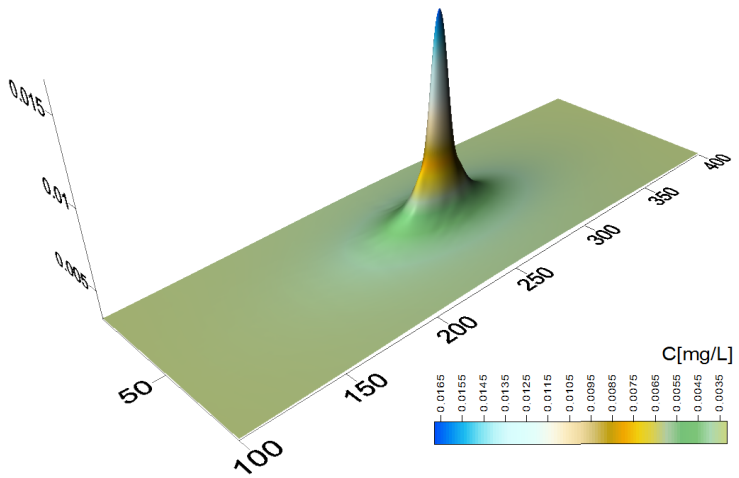
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΚΑΙ ΣΥΓΚΟΙΝΩΝΙΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΟΛΟΓΙΑΣ ΤΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

ΘΕΜΑ:
ΑΡΙΘΜΗΤΙΚΗ ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΣΥΜΜΕΤΑΦΟΡΑΣ ΒΙΟΚΟΛΛΟΕΙΔΩΝ - ΚΟΛΛΟΕΙΔΩΝ ΣΩΜΑΤΙΔΙΩΝ ΣΕ ΤΡΙΣΔΙΑΣΤΑ ΠΟΡΩΔΗ ΜΕΣΑ



ΑΠΟ
ΚΑΤΖΟΥΡΑΚΗ ΒΑΣΙΛΕΙΟ ΑΜ: 422

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
Κ. ΚΩΣΤΑΝΤΙΝΟΣ ΧΡΥΣΙΚΟΠΟΥΛΟΣ

ΠΑΤΡΑ
22/6/2012

Π Ρ Ο Λ Ο Γ Ο Σ

Η παρούσα εργασία εκπονήθηκε, στον χώρου του Εργαστηρίου Τεχνολογίας του Περιβάλλοντος. Το οποίο ανήκει στον τομέα τεχνολογίας του περιβάλλοντος και συγκοινωνιών, της πολυτεχνικής σχολής τμήμα πολιτικών μηχανικών.

Σημαντικό ρόλο στην εκπόνηση αυτής της εργασίας διαδραμάτισε ο κ. Κωνσταντίνος Χρυσικόπουλος, μέλος ΔΕΠ και υπεύθυνος του παραπάνω εργαστηρίου. Επιπλέον πρέπει να αναγνωριστεί η συμβολή του κ. Νικόλαου Ιωακειμίδη, εξωτερικού διδάσκοντα, ο οποίος βοήθησε στις εκάστοτε αριθμητικές επιλύσεις.

Π Ε Ρ Ι Λ Η Ψ Η

ΣΚΟΠΟΣ

Το θέμα το οποίο πραγματεύεται η παρούσα διπλωματική είναι η «Αριθμητική μοντελοποίηση συμμεταφοράς βιοκολοειδών-κολοειδών σωματιδίων σε τρισδιάστατα πορώδη μέσα ». Στο αντικείμενο αυτό ανήκουν τα φυσικά και χημικά φαινόμενα τα οποία παίρνουν μέρος στην μεταφορά ουσιών στο υπέδαφος αλλά και γενικότερα στα πορώδη μέσα. Με τον όρο προσομοίωση εννοούμε την προσπάθεια αριθμητικής περιγραφής των παραπάνω φαινομένων αλλά και των σταθερών, που διέπουν στην πραγματικότητα το φυσικό περιβάλλον εις το οποίο εκτυλίσσεται η μεταφορά.
Σκοπός της εργασίας αυτής είναι η αριθμητική επίλυση της εξίσωσης μεταφοράς-συμμεταφοράς ρύπων και η σύγκριση των αποτελεσμάτων αυτών με τις αντίστοιχες αναλυτικές λύσεις. Στην συνέχεια επιλέγοντας κατάλληλα μοντέλα, βασισμένα στις λύσεις που αναπτύχθηκαν, θα γίνει προσαρμογή δεδομένων (Fitting) από εργαστηριακά πειράματα ώστε να υπολογιστούν φυσικές σταθερές που υπεισέρχονται στην συμμεταφορά ρύπων. Τέλος σε ένα γενικότερο πλαίσιο θα γίνει προσπάθεια να αντιπαρατεθούν όλα τα θετικά και τα αρνητικά που έχουν οι αριθμητικές σε σχέση με τις αναλυτικές λύσεις ώστε να εξαχθεί ένα συμπέρασμα για την χρηστικότητα τους.

ΠΕΡΙΓΡΑΦΗ ΚΕΦΑΛΑΙΩΝ

1. Στο πρώτο κεφάλαιο παρατίθενται τα βασικά μεγέθη και οι εξισώσεις που συσχετίζονται με την μεταφορά ρύπων μέσα σε ένα πορώδες. Εδώ θα συναντήσει κανείς την διάχυση, διασπορά, μεταγωγή, προσρόφηση καθώς και την θεωρεία DLVO .

2. Στο δεύτερο κεφάλαιο γίνεται εξαγωγή της μονοδιάστατης εξίσωσης που περιγράφει πλέον την απλή μεταφορά ρύπων. Για να γίνει αυτό θα συνδυαστούν τα μεμονωμένα φαινόμενα που περιγράφονται στο πρώτο κεφάλαιο αλλά και θα χρησιμοποιηθεί κατάλληλα η εξίσωση της συνέχειας. Τέλος θα γίνει η γενίκευση της εξίσωσης αυτής στις τρείς διαστάσεις και θα περιγραφούν γενικά τα είδη των συνοριακών και αρχικών συνθηκών.

3. Στο τρίτο κεφάλαιο παρουσιάζεται το φαινόμενο της συμμεταφοράς μαζί με τις εξισώσεις της βιβλιογραφίας που το περιγράφουν. Επιπλέον θα δοθούν οι λόγοι για τους οποίους τα τελευταία χρόνια υπάρχει στροφή του επιστημονικού ενδιαφέροντος προς το φαινόμενο αυτό.

4. Στο τέταρτο κεφάλαιο αρχικά θα περιγραφούν συνοπτικά οι διαφορετικοί τρόποι επίλυσης διαφορικών εξισώσεων. Στην συνέχεια θα παρουσιαστούν έτοιμες αναλυτικές λύσεις για τρισδιάστατα μοντέλα μεταφοράς. Τέλος θα γίνει εμβάθυνση στις αριθμητικές λύσεις και πιο συγκεκριμένα στις πεπερασμένες διαφορές, όπου και θα γίνει ένα σχετικό αριθμητικό παράδειγμα.

5. Στο πέμπτο κεφάλαιο θα γίνει εφαρμογή των αριθμητικών μεθόδων, που περιγράφηκαν στο κεφάλαιο τέσσερα, στις εξισώσεις μεταφοράς και συμμεταφοράς ενώ παράλληλα θα προστεθούν οι κατάλληλες συνοριακές συνθήκες ώστε να ολοκληρωθεί η μοντελοποίηση των φαινομένων αυτών.

6. Στο έκτο κεφάλαιο θα περιγραφούν θέματα ευστάθειας, σύγκλισης και ακαμψίας που υπεισέρχονται στις αριθμητικές επιλύσεις. Επιπλέον θα δοθούν τρόποι οι οποίοι βοηθούν στο να ξεπεραστούν τα προβλήματα αυτά.

7. Στο έβδομο κεφάλαιο θα περιγραφεί η δομή του κώδικα (σε Fortran) ο οποίος υλοποιεί αναλυτικές και αριθμητικές λύσεις για τα διαφορετικά μοντέλα.

8. Στο όγδοο κεφάλαιο θα παρουσιαστούν αποτελέσματα από τις διαφορετικές εκτελέσεις των προγραμμάτων. Ακόμα θα περιγραφούν τα πειράματα μεταφοράς και συμμεταφοράς που χρησιμοποιήθηκαν ώστε να γίνει προσαρμογή δεδομένων και να υπολογιστούν οι δρώσες φυσικές σταθερές.

9. Στο ένατο κεφάλαιο θα παρουσιαστεί η βιβλιογραφία.

10. Στο δέκατο κεφάλαιο θα περιγράφει ο τρόπος με τον οποίο κανείς μπορεί να χρησιμοποιήσει τα προγράμματα που έχουν δημιουργηθεί για την επίλυση των διαφορικών εξισώσεων μεταφοράς. Με άλλα λόγια θα δοθεί το εγχειρίδιο χρήσης.

11. Στο εντέκατο παρουσιάζεται αυτούσιος ο πηγαίος κώδικας Fortran με τα κατάλληλα σχόλια.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΜΕΤΑΦΟΡΑΣ ΜΑΖΑΣ

1.1 ΔΙΑΧΥΣΗ (DIFUUSION)

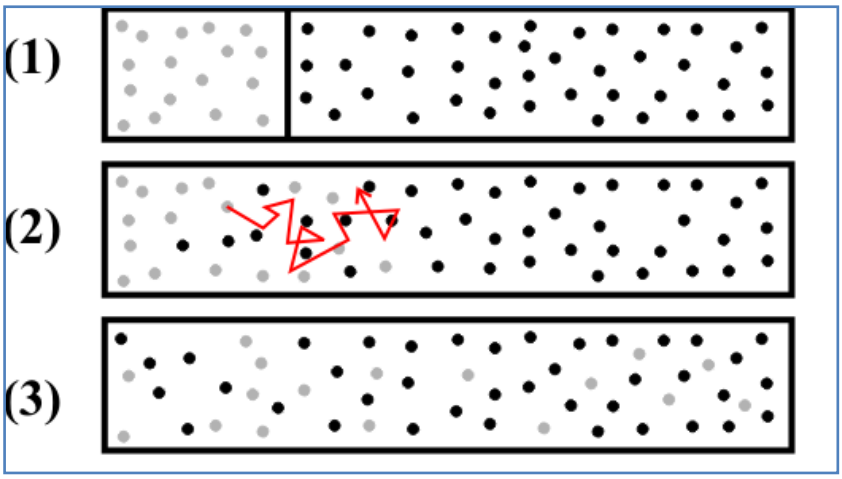
Ο Robert Brown παρατήρησε για πρώτη φορά το 1827 ότι πολύ μικρά σωματίδια αιωρούμενα στον αέρα ή στο νερό βρίσκονται σε μία συνεχή τυχαία κίνηση. Αργότερα ο Thomas Graham (1805-1869) προσδιόρισε ότι η διάχυση των αερίων είναι αντιστρόφως ανάλογη της τετραγωνικής ρίζας της πυκνότητας ή του μοριακού βάρους των αερίων . Η πρώτη ολοκληρωμένη μελέτη διάχυσης διαλυμένων στο νερό συστατικών έγινε από τον Adolf Fick (έτος γέννησης 1829). Η εργασία του Fick για την μελέτη διάχυσης δημοσιεύθηκε το 1855 και ορίζει ότι:

$$J = -D\frac{\partial C}{\partial X} \tag{1.1}$$

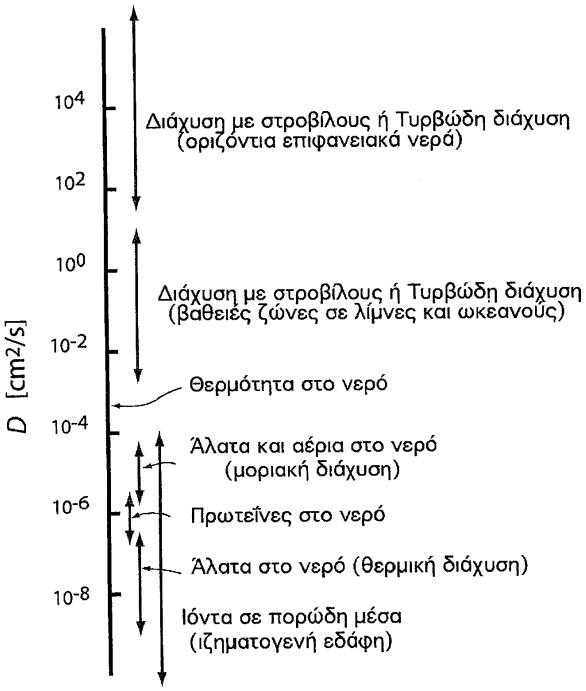
- J =Ροη μάζας ανά μονάδα επιφάνειας $\left[\frac{M}{tL^2}\right]$
- D=Συντελεστής μοριακής διάχυσης $\left[\frac{L^2}{t}\right]$
- C=Συγκέντρωση του διαλυμένου στο νερό συστατικού $\left[\frac{M}{L^3}\right]$

Ο νόμος αυτός δηλώνει ότι το διαλυμένο στο νερό συστατικό διαχέεται (δηλαδή κινείται σε σχέση με το διάλυμα ή μίγμα) προς την κατεύθυνση της αρνητικής βαθμίδας της συγκέντρωσης (δηλαδή από περιοχές με μεγαλύτερη συγκέντρωση σε αυτές με μικρότερη συγκέντρωση, γι΄αυτό υπάρχει και το "-", ώστε να δίδεται θετική ροή μάζας), ακριβώς όπως το νερό κινείται σε ένα πορώδες προς την κατεύθυνση της αρνητικής υδραυλικής κλίσης (νόμος Darcy). Όπως είναι φανερό από την εξ. (1.1) εάν δεν υπάρχει διαφορά συγκέντρωσης τότε και προφανώς δεν υπάρχει ροή μάζας.

Η διάχυση είναι διεργασία, που λαμβάνει χώρα εξαιτίας της τυχαίας κίνησης των μορίων (Random molecular motion), σύμφωνα με την οποία μεταφέρεται μάζα από ένα σημείο στο επόμενο (βλέπε Σχήμα 1.1). Συνεπώς και ο συντελεστής μοριακής διάχυσης “D” θα εξαρτάται από το μοριακό βάρος, την μοριακή δομή του διαλύτη και της διαλυμένης ουσίας (Πίνακας 1.1) αλλά και από την τυχαία κίνηση των μορίων. Σε κάθε περίπτωση όμως δεν εξαρτάται από την κατεύθυνση μεταφοράς. Ο Fick στην εξίσωσή του δεν έλαβε υπόψη την επίδραση που έχει η αλλαγή θερμοκρασίας και πίεσης ή άλλης δύναμης που συνεισφέρει στην μεταφορά μάζας, στην διάχυση.



Σχήμα 1.1: Σχηματική παρουσίαση μίξης δύο ουσιών σε τρείς φάσεις, μέσω διάχυσης.



Σχήμα 1.2 Συντελεστές μοριακής διάχυσης σε διάφορα περιβαλλοντικά συστήματα (Lerman,1971).

Πίνακας 1.1 : Συντελεστές μοριακής διάχυσης σε νερό για διάφορα ιόντα στους 25^0C (Υιοθετημένο από τους Li και Gregory 1974).

Κατιόν	D (10^{-6} cm^2/s)	Ανιόν	D (10^{-6} cm^2/s)
Υδρογόνο, H^+	93,1	Υδροξείδιο, OH^-	52,7
Νάτριο, Na^+	13,3	Φθόριο, F^-	14,6
Κάλιο, K^+	19,6	Χλώριο, Cl^-	20,6
Μαγνήσιο, Mg^{2+}	7,06	Βρώμιο, Br^-	20,1
Ασβέστιο, Ca^{2+}	7,93	Υδροθειικό, HS^-	17,3
Μαγγάνιο, Mn^{2+}	6,88	Διανθρακικό, HCO_3^-	11,8
Σίδηρος, Fe^{2+}	7,19	Ανθρακικό, CO_3^{2-}	9,55
Σίδηρος, Fe^{3+}	6,07	Θειικό, SO_4^{2-}	10,7

Όπως φαίνεται και στο Σχήμα 1.2 η διάχυση με ύπαρξη στροβίλων αυξάνει την τιμή του συντελεστή μοριακής διάχυσης D.

Οι Stokes-Einstein ανέπτυξαν μια θεωρητική σχέση εξ. (1.2) η οποία περιγράφει τον συντελεστή διάχυσης D που όμως τώρα εξαρτάται και από άλλους παράγοντες όπως θερμοκρασία. Πιο συγκεκριμένα η σχέση αυτή αναφέρεται στην διάχυση αραιών διαλυμάτων σφαιρικών αιωρούμενων κολλοειδών σωματιδίων A σε ρευστό B.

$$D_{AB} = \frac{k_B T}{3\pi \mu_B d_p} \tag{1.2}$$

(Einstein,1956; Polson,1950; Nird et al., 2002,p. 529)

K_B =Είναι η σταθερά Boltzmann

T = Απόλυτη θερμοκρασία

μ_β =Είναι το δυναμικό ιξώδες του ρευστού B

d_p=Είναι η διάμετρος των διαχεόμενων σωματιδίων

Αργότερα ο Fick εξέδωσε τον δεύτερο νόμο του, ο οποίος περιγράφει τη μεταβολή της συγκέντρωσης λόγω διάχυσης σε συνάρτηση του χρόνου σε καρτεσιανές συντεταγμένες με σταθερό συντελεστή μοριακής διάχυσης εξ. (1.3):

$$\frac{\partial C}{\partial t} = D \frac{\partial^2 C}{\partial X^2} \tag{1.3}$$

Εάν ο συντελεστής D δεν είναι σταθερός αλλά εξαρτάται από την καρτεσιανή συντεταγμένη X, τότε ο δεύτερος νόμος του Fick δίδεται από την σχέση

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial X}\left[D(x)\frac{\partial C}{\partial X}\right] \tag{1.4}$$

Αντίστοιχα ο δεύτερος νόμος του Fick για κυλινδρικές συντεταγμένες με σταθερό και μεταβλητό συντελεστή μοριακής διάχυσης δίνεται από την εξ. (1.5):

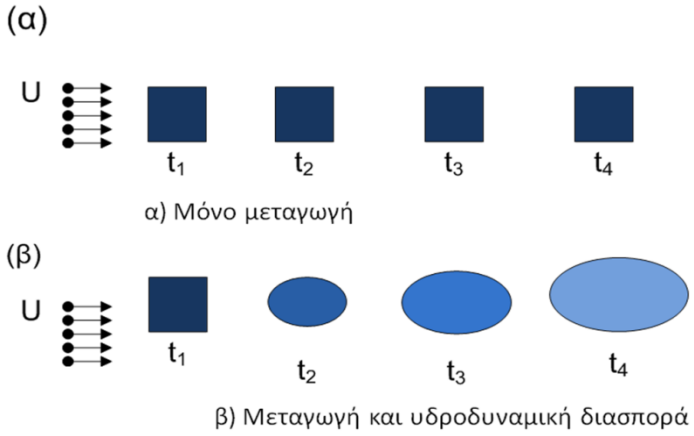
$$\frac{\partial C}{\partial t} = D\left(\frac{\partial^2 C}{\partial r^2} + \frac{2}{\mathrm{r}}\frac{\partial C}{\partial r}\right) \tag{1.5}$$

και για μεταβλητό συντελεστή :

$$\frac{\partial C}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left[r^2 D(r)\frac{\partial C}{\partial r}\right] \tag{1.6}$$

1.2 ΥΔΡΟΔΥΝΑΜΙΚΗ ΔΙΑΣΠΟΡΑ (HYDRONAMIC DISPERSION)

Η υδροδυναμική διασπορά είναι το φαινόμενο στο οποίο δυο αναμίξιμα υγρά εκτοπίζουν το ένα το άλλο μέσα σε ένα πορώδες μέσο . Ουσιαστικά στην διασπορά η μεταφορά της διαλυμένης ουσίας, από ένα σημείο στο άλλο, βασίζεται όχι μόνο στην διάχυση αλλά και στην ροή (κεκτημένη ταχύτητα) που έχει ο διαλύτης (βλέπε Σχήμα 1.3).



Σχήμα 1.3: Μεταφορά ρύπου με στιγμιαία εισαγωγή σε δισδιάστατο υδροφορέα με μεθόδους α, β. Η συγκέντρωση του ρύπου παρουσιάζεται σε διαφορετικές χρονικές στιγμές και είναι ανάλογη του χρώματος της ζώνης κάλυψης. Στην πρώτη περίπτωση (α) η συγκέντρωση του ρύπου είναι σταθερή και απλά μετατοπίζεται στον χώρο ενώ στην δεύτερη περίπτωση (β) ο ρύπος όχι μόνο μεταφέρεται αλλά και εξαπλώνεται με αποτέλεσμα να μειώνεται η συγκέντρωση του (Χρυσικόπουλος, 2010).

Πιο αναλυτικά η υδροδυναμική διασπορά εξαρτάται από ένα συνδυασμό φυσικοχημικών και μηχανικών διεργασιών. Οι φυσικοχημικές διεργασίες αποτελούνται από την διάχυση (μεταφορά μάζας λόγω κλίσης συγκέντρωσης) που βασίζεται στην τυχαία κίνηση των μορίων. Ενώ οι μηχανικές διεργασίες αποτελούνται από κινηματικούς και δυναμικούς μηχανισμούς οι οποίοι προκαλούνται από την ανομοιόμορφη ταχύτητα διήθησης στην κλίμακα των πόρων μέσα στο ανομοιόμορφο χώρο των διάκενων του πορώδους μέσου. Πρέπει να σημειωθεί ότι ενώ ο συντελεστής μοριακής διάχυσης δεν εξαρτάται από την κατεύθυνση της ροής (είναι ίδιος σε όλες τις διευθύνσεις), η υδροδυναμική διασπορά εξαρτάται. Τελικά μπορούμε να ισχυριστούμε ότι η υδροδυναμική διασπορά D_y είναι ίση με την αποτελεσματική διάχυση D_e και την μηχανική διασπορά D_m:

$$D_y = D_e + D_m \tag{1.7}$$

Ο συντελεστής αποτελεσματικής διάχυσης χρησιμεύει στο να αποδοθεί καλύτερα το φαινόμενο της διάχυσης στην πραγματικότητα. Όπως είναι προφανές όταν διαχέεται ένα υγρό, οι τροχιές που διαγράφουν τα ιόντα του καθορίζονται από το ίδιο το υγρό. Αντίθετα η ύπαρξη και άλλων σωματιδίων (πχ. κόκκοι άμμου) επιβάλει διαφορετικές τροχιές στον διαχεόμενο ρύπο. Τώρα τα ιόντα αναγκάζονται να κάνουν πιο μεγάλες διαδρομές γύρω από τους κόκκους Σχήμα 1.4 και η διάχυση περιορίζεται μόνο στα διάκενα του πορώδους μέσου. Όλα τα παραπάνω συνηγορούν στο ότι ο πραγματικός συντελεστής διάχυσης θα είναι μικρότερος στην δεύτερη περίπτωση.

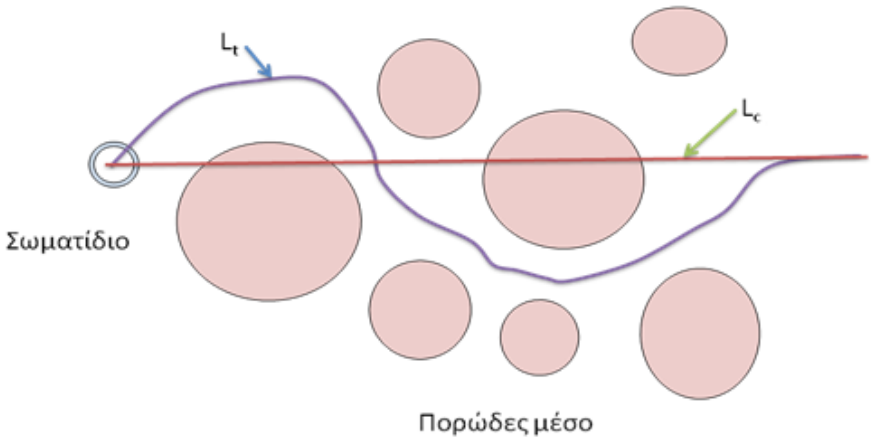
Έτσι ορίζεται το δαιδαλώδες των πόρων (tortuosity) τ:

$$\tau = \left(\frac{L_t}{L_c}\right)^2 \tag{1.8}$$

και ο συντελεστής αποτελεσματικής διάχυσης D_e:

$$D_e = \frac{D}{\tau} \tag{1.9}$$

Όπως προκύπτει και από το Σχήμα 1.4 το πραγματικό μήκος L_t είναι πάντα μεγαλύτερο από το χαρακτηριστικό μήκος L_c, οπότε και το δαιδαλώδες "τ" είναι πάντα μεγαλύτερο της μονάδας. Συνεπώς από εξ. (1.9) $D_e < D$.



Σχήμα 1.4:Πραγματική διαδρομή ιόντος μέσα σε πορώδες μέσο L_c και χαρακτηριστικό μήκος διαδρομής L_t.

Με την σειρά του ο Dm συντελεστής μηχανικής διασποράς εκφράζεται ως D_m=AU. Για ένα ανισότροπο πορώδες μέσο στις 3 διαστάσεις, όπου η κύρια κατεύθυνση της μέσης ενδοπορώδους ταχύτητας συμπίπτει με τον άξονα του συστήματος των συντεταγμένων, η εξίσωση αυτή παίρνει την μορφή (Bear, 1979):

$$D_m = AU = \begin{pmatrix} a_L U & 0 & 0 \\ 0 & a_T U & 0 \\ 0 & 0 & a_T U \end{pmatrix} \quad (1.10)$$

Όπου A είναι ο τανυστής τάσης διασποράς.

Με α_L να είναι η διαμήκης τάση διασποράς και α_T να είναι η εγκάρσια τάση διασποράς .

Τελικά με την χρήση των εξισώσεων (1.7), (1.9), (1.10) οι συντελεστές διαμήκης και εγκάρσιας υδροδυναμικής διασποράς D_L, D_T αντίστοιχα γράφονται:

$$D_L = D_e + a_L U \quad (1.11)$$

$$D_T = D_e + a_T U \quad (1.12)$$

Έτσι για την περίπτωση όπου η κύρια κατεύθυνση της μέσης ενδοπορώδους ταχύτητας συμπίπτει με τον άξονα του συστήματος συντεταγμένων ισχύει ότι ο συντελεστής υδροδυναμικής διασποράς είναι ένας διαγώνιος τανυστής της μορφής εξ. (1.13) (Bear, 1979):

$$D = \begin{pmatrix} D_{xx} & 0 & 0 \\ 0 & D_{yy} & 0 \\ 0 & 0 & D_{zz} \end{pmatrix} \quad (1.13)$$

Όπου τελικά $D_{xx} = D_L$, $D_{yy} = D_T$ και $D_{zz} = D_T$. Δηλαδή η κυρίως διαμήκης κατεύθυνση θεωρείται η X΄X ενώ οι εγκάρσιες κατευθύνσεις είναι οι Y΄Y και Z΄Z.

Διαφορετικά για να περιγραφεί ο συντελεστής D θα χρειαζόταν ένας τανυστής εννιά όρων της μορφής

$$D = \begin{pmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{pmatrix} \quad (1.14)$$

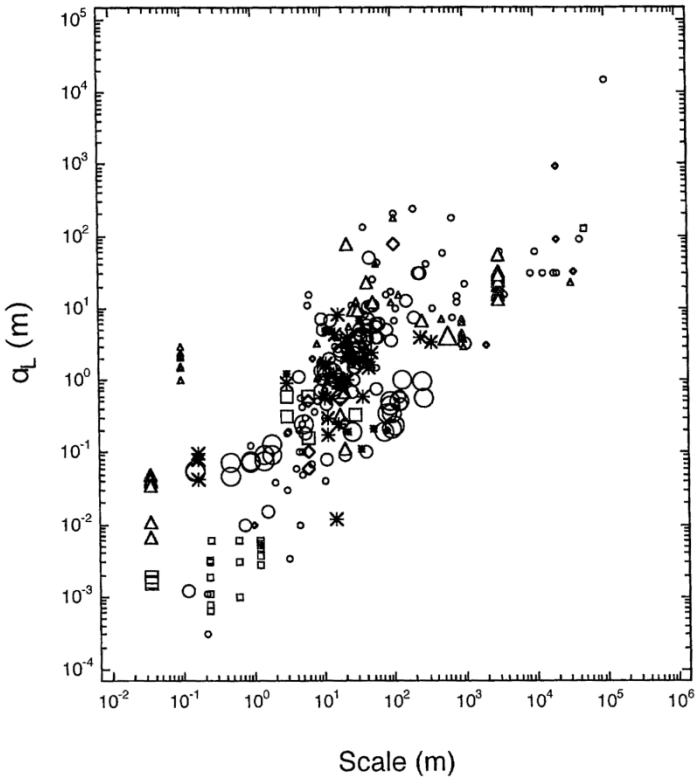
Ο Schulze-Makuch (2005) συγκέντρωσε πειραματικά δεδομένα από πολλές μελέτες για διαφορετικά πετρώματα και πρότεινε εμπειρικές συσχετίσεις σε συνάρτηση της κλίμακας του πεδίου. Αυτές φαίνονται στον πίνακα 1.2

Πίνακας 1.2 :Εμπειρικές συσχετίσεις του συντελεστή διαμήκης τάσης διασποράς για διάφορα πετρώματα. Υιοθετημένο από τον Schulze-Makuch (2005).

Πέτρωμα	Εμπειρική συσχέτιση
Μη-στερεοποιημένα πετρώματα (unconsolidated sediments)	$a_L=0.085(L)^{0.81}$
Ψαμμίτες (sandstones)	$a_L=0.01(L)^{0.92}$
Ανθρακικά πετρώματα (carbonates)	$a_L=0.80(L)^{0.40}$
Βασάλτες (basalts)	$a_L=0.15(L)^{0.61}$
Γρανίτες (granites)	$a_L=0.21(L)^{0.51}$

Ο συντελεστής υδροδυναμικής διασποράς εξαρτάται από την κλίμακα του πεδίου και αυτό έχει αποδειχθεί από δεδομένα πολλαπλών εργασιών. Στο Σχήμα 1.5 παρουσιάζονται τα πειραματικά δεδομένα για την διαμήκη τάση διασποράς από διάφορες μελέτες πεδίου που έχουν δημοσιευτεί στην διεθνή βιβλιογραφία, όπως τελικά συγκεντρώθηκαν από τους Gelhar (1986), Gelhar et al. (1992) και Schule-Makuch (2005). Παρατηρείται ότι ο συντελεστής a_L αυξάνει με την αύξηση της

κλίμακας του πεδίου. Η εξάρτηση του a_L στην κλίμακα του πεδίου αποδίδεται στην ύπαρξη τοπικών ανομοιογενειών του πεδίου. (Χρυσικόπουλος, 2009)



Σχήμα 1.5 : Διαμήκης τάση διασποράς από διάφορες μελέτες στο πεδίο οι οποίες έχουν συγκεντρωθεί από τους Gelhar (1986), Gelhar et al. (1992) και Schule-Makuch (2005). Η αξιοπιστία των δεδομένων αυξάνει με την αύξηση του μεγέθους των συμβόλων (Χρυσικόπουλος, 2009, σελ 131).

Ένα χρήσιμο μέγεθος στην μεταφορά διαλυμένης μάζας είναι ο "συντελεστής επιβράδυνσης "R" (retardation factor). Αυτός ισούται με τον λόγο της ενδοπορώδους ταχύτητας U, ως προς την ταχύτητα της διαλυμένης ουσίας στην υδατική φάση U_c.

$$R = \frac{U}{U_c} \tag{1.15}$$

Ο συντελεστής R παίρνει τιμές από 1 και επάνω καθώς είναι φανερό ότι η ταχύτητα της διαλυμένης ουσίας είναι πάντα μικρότερη από την ενδοπορώδη ταχύτητα. Θα μπορούσε να ισχυριστεί κανείς ότι εξαιτίας της διάχυσης προσδίδεται στην Uc μια επιπλέον ώθηση (ταχύτητα U_D) η οποία θα μπορούσε τελικά να υπερβεί την ενδοπορώδη ταχύτητα U. Στην πραγματικότητα όμως η μεταφορά διαλυμένης ουσίας λόγω μεταγωγής είναι πολύ μεγαλύτερη από αυτήν εξαιτίας της διάχυσης και γι' αυτό

θεωρείται ότι η συνεισφορά της διάχυσης στην ταχύτητα Uc είναι αμελητέα σε σχέση με την ενδοπορώδη ταχύτητα U (εξ. 1.16-1.18).

$$U_c = U_D + U \tag{1.16}$$

$$\frac{U_D}{U} \ll 1 \tag{1.17}$$

Και συνεπώς από τις Εξισώσεις (1.16) και (1.17) προκύπτει:

$$U_c \leq U \tag{1.18}$$

Επιπλέον υπάρχουν δυο ακόμα εξαιρέσεις στις οποίες το R<1. Η πρώτη συμβαίνει όταν έχουμε όχι μόνο μεταφορά διαλυμένης ουσίας αλλά και παραγωγή αυτής. Όποτε όταν ανιχνεύουμε συγκεντρώσεις, για να εκτιμήσουμε την ποσότητα της μάζας που μεταφέρθηκε (ώστε τελικά να βρούμε την ταχύτητα Uc), χωρίς να το θέλουμε έχουμε συνυπολογίσει και την συγκέντρωση που οφείλεται στην παραγωγή. Με αυτόν το τρόπο η ταχύτητα Uc είναι τεχνητά αυξημένη και ξεπερνά την ταχύτητα μεταφοράς U χωρίς όμως αυτό να ισχύει στην πραγματικότητα. Δεύτερη περίπτωση είναι όταν υπάρχει μεταφορά αιωρούμενων στερεών. Όπως έχει αποδειχτεί από εργασίες (James and Chrysikopoulos, 2003) τα αιωρούμενα στερεά μέσα σε κλειστό αγωγό προτιμούν να ταξιδεύουν στο κέντρο αυτού και με ταχύτητες μεγαλύτερες από την μέση ταχύτητα μεταφοράς του νερού U. Έτσι εδώ πραγματικά το R μπορεί να έχει τιμή μεγαλύτερη της μονάδας.

1.3 ΠΡΟΣΡΟΦΗΣΗ (ADSORPTION)

1.3.1 ΓΕΝΙΚΕΣ ΙΔΙΟΤΗΤΕΣ ΠΡΟΣΡΟΦΗΣΗΣ

Η προσρόφηση είναι φαινόμενο κατά το οποίο οι ρύποι σε μορφή ιόντων ή μορίων μίας διαλυμένης ουσίας συγκεντρώνονται στην επιφάνεια στερεών που αποτελούν το στερεό σκελετό εδαφικών στρωμάτων. Η προσρόφηση χωρίζεται σε δυο μεγάλες κατηγορίες

Α) Ισόθερμη

Β) Αντιστρέψιμη

Στην πρώτη κατηγορία οι συνθήκες όπως θερμοκρασία και πυκνότητα, παραμένουν σταθερές. Εδώ η διαλυμένη ποσότητα ρύπου έχει την δυνατότητα μόνο να προσροφηθεί χωρίς να μπορεί ποτέ να επιστρέψει σε διαλυμένη μορφή. Ακόμα η ταχύτητα των αντιδράσεων αυτών είναι πολύ μεγάλη σε σχέση με την ταχύτητα εξέλιξης του φαινομένου μεταφοράς των ρύπων. Συνεπώς δεν υπάρχει ανάγκη να γίνει περιγραφή του φαινομένου στον χρόνο.

Αντίθετα στην δεύτερη κατηγορία οι συνθήκες δεν παραμένουν σταθερές. Ένα μέρος της προσροφημένης ποσότητα ρύπου ξαναμετατρέπεται σε διαλυμένη μορφή καθώς εξελίσσεται το φαινόμενο. Η ταχύτητα των αντιδράσεων αυτών μπορεί να συγκριθεί με το φαινόμενο μεταφοράς ρύπων και γι' αυτό επιβάλλεται να γίνει περιγραφή του φαινομένου αυτού σε σχέση με τον χρόνο.

Ακόμα με βάση το είδος των μηχανισμών που παίρνουν μέρος στην προσρόφηση, αυτή μπορεί να χωριστεί σε τρείς κατηγορίες :

1) Φυσική προσρόφηση

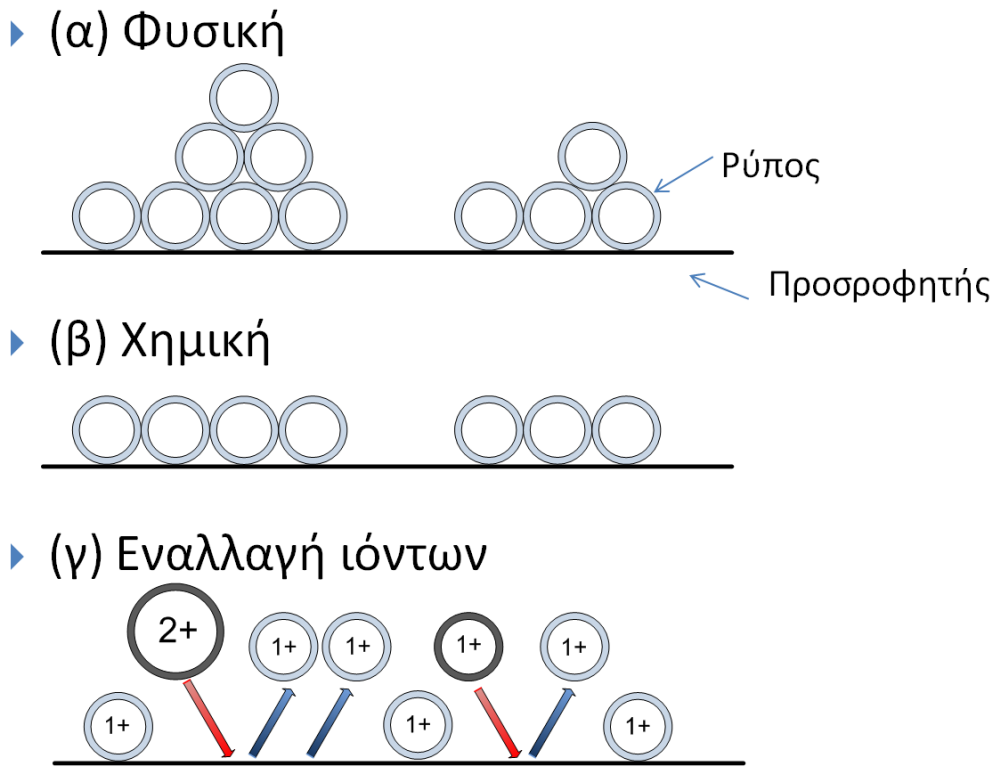
2) Χημική προσρόφηση

3) Εναλλαγή ιόντων

Στην φυσική αναπτύσσονται δυνάμεις Van der Waals που ασκούνται μεταξύ των ρύπων και της επιφάνειας των στερεών. Οι προσροφημένοι ρύποι δεν παραμένουν σε ένα συγκεκριμένο σημείο πάνω στον προσροφητή αλλά μπορούν και κινούνται ελεύθερα πάνω σε αυτό. Υπάρχει η δυνατότητα να δημιουργηθούν πολλές στρώσεις η μια πάνω στην άλλη (Σχ. 1.6) οι οποίες τελικά στηρίζονται στην επιφάνεια του στερεού. Εάν το επιτρέψουν οι συνθήκες (χαμηλή συγκέντρωση διαλυμένης ουσίας) η προσροφημένη ουσία έχει την δυνατότητα να ελευθερωθεί από τον προσροφητή και να ξανά επιστρέψει σε διαλυμένη μορφή. Η διαδικασία αυτή ονομάζεται εκρόφηση.

Στην χημική αναπτύσσονται αρκετά ισχυρές ελκτικές δυνάμεις μεταξύ των ρύπων και του προσροφητή και οδηγούν στον σχηματισμό χημικών ενώσεων. Εξαιτίας αυτού οι προσροφημένοι ρύποι δεν μπορούν κινούνται ελευθέρα επάνω στα στερεά ούτε μπορούν να σχηματίζουν πολλαπλά στρώματα. Όταν η επιφάνεια του προσροφητή γεμίσει τελείως τότε το φαινόμενο της προσρόφησης σταματά παντελώς Σχήμα 1.6. Η χημική προσρόφηση συνήθως δεν είναι αναστρέψιμη παρά μόνο όταν αυξηθεί η θερμοκρασία του προσροφητή.

Στην εναλλαγή ιόντων Σχήμα 1.6 ένα ιόν από την επιφάνεια του προσροφητή εναλλάσσεται με ένα η περισσότερα ιόντα της διαλυμένης ουσίας (ίσης αξίας). Η προσρόφηση αυτού του είδους οφείλεται σε ελκτικές ηλεκτροστατικές δυνάμεις που προκύπτουν εξαιτίας του αντίθετου ηλεκτρικού φορτίου μεταξύ του ρύπου και του προσροφητή.



Σχήμα 1.6: Κατηγορίες προσρόφησης (Χρυσικόπουλος 2010).

Τέλος οι φυσικοχημικές ιδιότητες των προσροφημένων ουσιών που καθορίζουν τον βαθμό προσρόφησης τους είναι οι εξής.

1. Η οξύτητα ή βασικότητα της ένωσης
2. Η διαλυτότητα στο νερό
3. Η κατανομή του φορτίου στα οργανικά κατιόντα
4. Η πολικότητα του μορίου

1.3.2 ΜΑΘΗΜΑΤΙΚΕΣ ΣΧΕΣΕΙΣ ΠΡΟΣΡΟΦΗΣΗΣ

Υπάρχουν αρκετά μοντέλα στην βιβλιογραφία που περιγράφουν διαφορετικά είδη προσρόφησης. Τα πιο γνωστά είναι η "ΓΡΑΜΜΙΚΗ ΙΣΟΘΕΡΜΙΚΗ ΠΡΟΣΡΟΦΗΣΗ", η "ΙΣΟΘΕΡΜΙΚΗ ΠΡΟΣΡΟΦΗΣΗ ΤΥΠΟΥ FREUNDLICH" και η "ΙΣΟΘΕΡΜΙΚΗ ΠΡΟΣΡΟΦΗΣΗ ΤΥΠΟΥ LANGMUIR".

Η γραμμική ισοθερμική προσρόφηση Σχήμα 1.7 δίδεται από τον τύπο εξ. (1.19):

$$c^* = K_d\, c_{eq} \tag{1.19}$$

c^*=η συγκέντρωση της προσροφημένης ουσίας $\left[\frac{M_{ουσ\,ίας}}{M_{στερε\,ών}}\right]$

K_d= συντελεστής κατανομής μάζας (distribution coefficient) $\left[\frac{L^3_{ρευστο\,ύ}}{M_{στερε\,ών}}\right]$

c_{eq}=συγκέντρωση διαλυμένης ουσίας στην κατάσταση ισορροπίας $\left[\frac{M_{ουσ\,ίας}}{L^3_{ρευστο\,ύ}}\right]$

Η Ισοθερμική προσρόφηση τύπου Langmuir Σχήμα 1.7 δίδεται από τον τύπο εξ. (1.20)

$$c^* = \frac{Q^0 a_1 c_{eq}}{1+a_1 c_{eq}} \tag{1.20}$$

c^*=η συγκέντρωση της προσροφημένης ουσίας $\left[\frac{M_{ουσ\,ίας}}{M_{στερε\,ών}}\right]$

Q^0= είναι η μέγιστη δυνατή ποσότητα προσροφημένης ουσίας σε μονοστρωματική διάταξη στα στερεά $\left[\frac{M_{ουσ\,ίας}}{M_{στερε\,ών}}\right]$

a_1=εμπειρική σταθερά η οποία σχετίζεται με την ενέργεια δέσμευσης (ενθαλπία προσρόφησης) $\left[\frac{L^3}{M_{ουσ\,ίας}}\right]$

c_{eq}=συγκέντρωση διαλυμένης ουσίας στην κατάσταση ισορροπίας $\left[\frac{M_{ουσ\,ίας}}{L^3_{ρευστο\,ύ}}\right]$

Η ισοθερμική προσρόφηση Σχήμα 1.7 τύπου Freundlich εξ. (1.21):

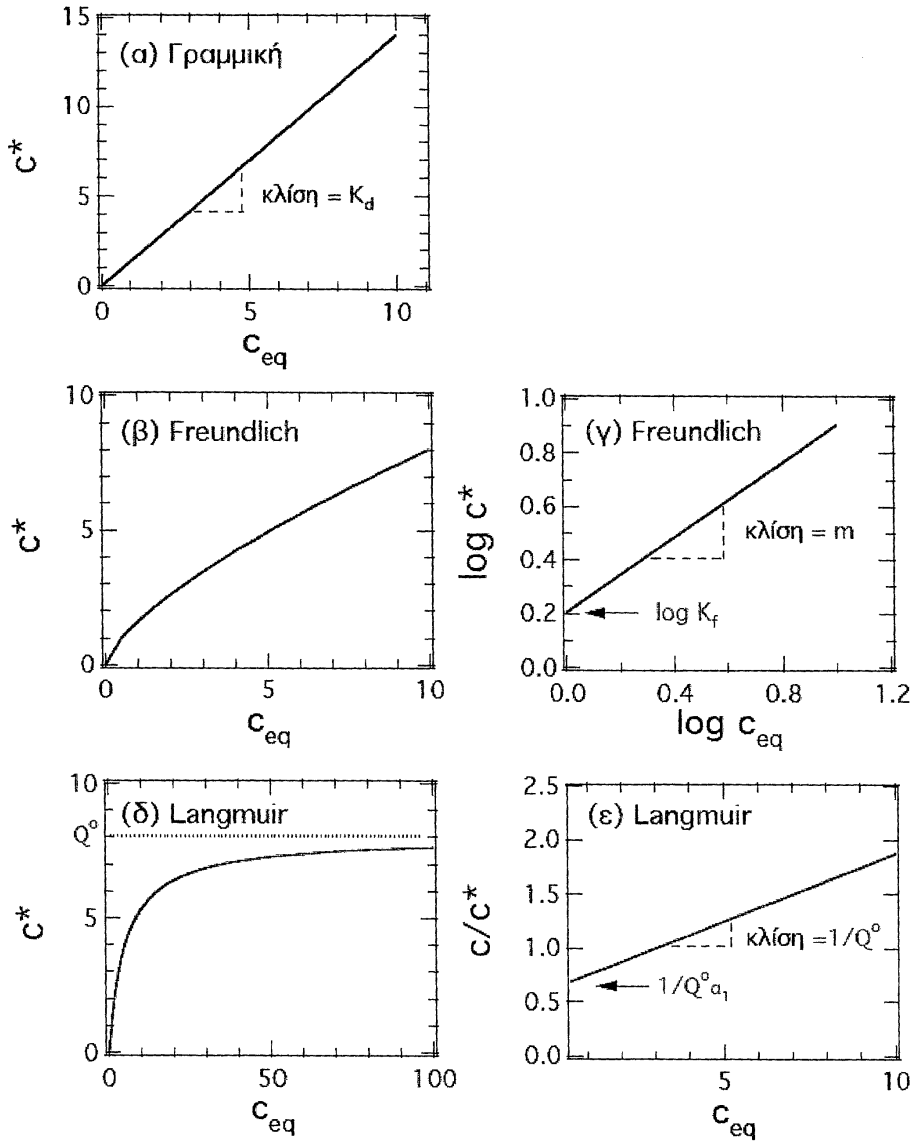
$$c^* = K_f\, c_{eq}^m \tag{1.21}$$

c^*=η συγκέντρωση της προσροφημένης ουσίας $\left[\frac{M_{ουσ\,ίας}}{M_{στερε\,ών}}\right]$

K_f=σταθερά$\left[\frac{\left(L^3_{ρευστο\,ύ}\right)^m}{M_{στερε\,ών}\left(M_{ουσ\,ίας}\right)^{m-1}}\right]$

c_{eq}=συγκέντρωση διαλυμένης ουσίας στην κατάσταση ισορροπίας $\left[\frac{M_{ουσ\,ίας}}{L^3_{ρευστο\,ύ}}\right]$

m= εκθέτης



Σχήμα 1.7: Ισόθερμες προσροφήσεις: (α) γραμμική με K_d=1.6 L/g, (β) & (γ) τύπου Freundlich με K_r=1.6 $(L/g)^m$και m=0.7, και (δ) & (ε) τύπου Langmuir με Q^0=8 mg/g και α_l=0.2 L/mg (Χρυσικόπουλος,2010).

Αντίστοιχα στην κινητική προσρόφηση τα πιο γνωστά μοντέλα παρατίθεται στον Πίνακα 1.6. Όπου α, α_1, α_2, n είναι εμπειρικές σταθερές ή συντελεστές παλινδρόμησης (regression coefficients) που εξαρτώνται από το σύστημα της διαλυμένης στο νερό ουσίας και των στερεών, θ είναι το πορώδες, c_g είναι η συγκέντρωση της ουσίας στην υγρή φάση σε άμεση επαφή με το στερεό προσροφητή, k, k_1, k_2 είναι συντελεστές ρυθμού αντίδρασης και φ_s είναι το κλάσμα των περιοχών προσρόφησης που καταλαμβάνονται από τη διαλυμένη στο νερό ουσία.

ΠΙΝΑΚΑΣ 1.6 :Κινητικά μοντέλα προσρόφησης (Travis and Etnier, 1981)

Όνομα	Μαθηματικό μοντέλο
Γραμμική αντιστρέψιμη (linear reversible)	$\frac{dc^*}{dt} = \frac{k_1\theta}{\rho_b}c - k_2c^*$
Μη-γραμμική αντιστρέψιμη (nonlinear reversible)	$\frac{dc^*}{dt} = \frac{k_1\theta}{\rho_b}c^n - k_2c^*$
Κινητικό γινόμενο (kinetic product)	$\frac{dc^*}{dt} = ac^{a_1}c^{a_2}$
Διγραμμική προσρόφηση (bilinear adsorption)	$\frac{dc^*}{dt} = k_1c\left(Q^o - c^*\right) - k_2c^*$
Μεταφορά μάζας (mass transfer)	$\frac{dc^*}{dt} = k\left(c - c_g\right)$
Μοντέλο Elovich (Elovich model)	$\frac{d\varphi_s}{dt} = a_1 \exp[-a\varphi_s]$

Αν χρησιμοποιήσουμε ένα από τα παραπάνω μοντέλα σε συνδυασμό με αποδόμηση τότε θα πάρουμε

$$\frac{\rho}{\theta}\frac{\partial C^*(t,x,y,z)}{\partial t} = K_1C(t,x,y,z) - K_2\frac{\rho}{\theta}C^*(t,x,y,z) - \frac{\lambda^*\rho}{\theta}C^*(t,x,y,z) \qquad (1.21)$$

Με λ^* συντελεστής αποδόμησης προσροφημένης ουσίας, C συγκέντρωση διαλυμένης ουσίας και C* συγκέντρωση προσροφημένης ουσίας, ρ= η πυκνότητα του πορώδους, θ = το πορώδες του στερεού, K1,K2 =συντελεστές ρυθμού αντίδρασης.

1.4 ΜΕΤΑΓΩΓΗ (ADVECTION)

Η πιο βασική συνιστώσα μεταφοράς ρύπων σε πορώδες μέσο είναι η μεταγωγή. Η οποία βασίζεται απλά και μόνο στην ταχύτητα της ροής που υφίσταται και είναι ανεξάρτητη της διάχυσης. Η εξίσωση που την αντιπροσωπεύει στην μονοδιάστατη περίπτωση είναι:

$$\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial X} = 0 \qquad (1.22)$$

C= συγκέντρωση διαλυμένης ουσίας $\frac{M}{L^3}$

t= χρόνος

x= χωρική μεταβλητή

1.5 ΘΕΩΡΕΙΑ ΠΡΟΣΡΟΦΗΣΗΣ ΑΙΩΡΟΥΜΕΝΩΝ ΣΤΕΡΕΩΝ

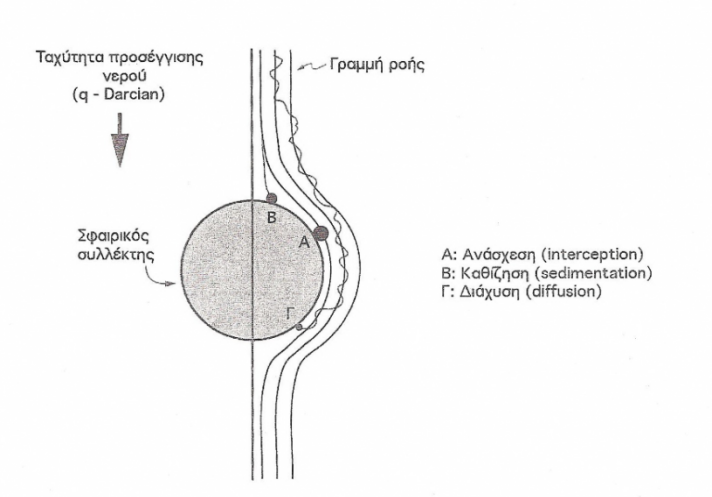
Τα τελευταία χρόνια έχουν γίνει πολλές προσπάθειες έτσι ώστε να περιγραφούν τα φαινόμενα που παίρνουν μέρος κατά την διάρκεια διήθησης ενός υγρού από ένα φίλτρο. Η πιο διαδεδομένη άποψη είναι αυτή των Yao et al. (1971), σύμφωνα με τους οποίους υφίστανται τρείς μηχανισμοί σύλληψης (ανάσχεση, καθίζηση, διάχυση). Υπάρχει όμως ακόμα ένας τέταρτος μηχανισμός (στράγγισης) ο οποίος όμως δεν είναι επιθυμητός. Η περιγραφή των τριών πρώτων θα γίνει θεωρώντας ιδανικό μοναδιαίο σφαιρικό συλλέκτη Σχήμα 1.8, ενώ για τον τέταρτο θα χρειαστούμε παραπάνω από ένα σφαιρικούς συλλέκτες Σχήμα 1.8.

α) Μηχανισμός ανάσχεσης (interception)

β) Μηχανισμός καθίζησης (sedimentation)

γ) Μηχανισμός διάχυσης (diffusion)

δ) Μηχανισμός στράγγισης (straining)



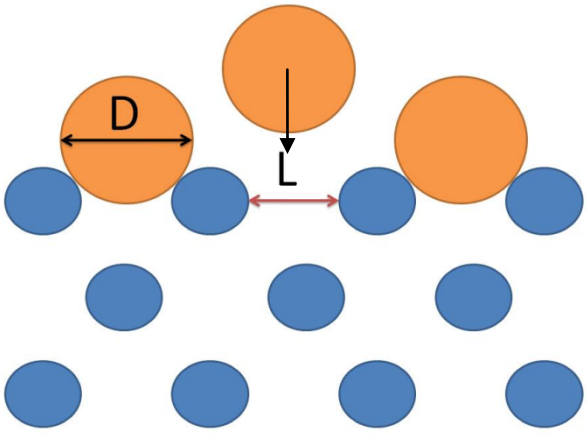
Σχήμα 1.8 :Παρουσιάζονται οι μηχανισμοί σύλληψης στερεών πάνω σε σφαιρικό συλλέκτη (Χρυσικόπουλος, 2012).

Σύμφωνα με τον πρώτο μηχανισμό (ανάσχεσης) η τροχιά ενός αιωρούμενου στερεού σωματιδίου συμβαίνει να περνάει πολύ κοντά από τον σφαιρικό συλλέκτη. Έτσι κατά την διάρκεια της πορείας του συγκρούεται με αυτόν και το πιθανό αποτέλεσμα είναι η σύλληψη.

Σύμφωνα με τον δεύτερο μηχανισμό (καθίζησης), η τροχιά του αιωρούμενου στερεού σωματιδίου τώρα δεν περνάει πολύ κοντά από τον σφαιρικό συλλέκτη, αλλά όμως εξαιτίας της βαρύτητας παρεκκλίνει της πορείας του και συγκρούεται με αυτόν. Για να συμβεί αυτού του είδος ο μηχανισμός είναι απαραίτητο να υπάρχει μεγάλη διαφορά στην πυκνότητα του υγρού με το στερεό σωματίδιο ώστε η επίδραση της βαρύτητας να είναι σημαντική.

Σύμφωνα με τον τρίτο μηχανισμό (διάχυσης) η τροχιά του αιωρούμενου στερεού πάλι δεν περνάει κοντά από τον σφαιρικό συλλέκτη αλλά αντίθετα φαίνεται να είναι και αρκετά μακριά του. Παρ όλα αυτά όμως εάν το σωματίδιο έχει μεγάλο συντελεστή διάχυσης δύναται τυχαία να υπερπηδήσει πολλαπλές τροχιές και τελικά να καταλήξει επάνω στον συλλέκτη.

Ο τέταρτος μηχανισμός σύλληψης, Σχήμα 1.9, συμβαίνει επειδή το μέγεθος του αιωρούμενου σωματιδίου είναι μεγαλύτερο από το διάκενο που δημιουργούν οι σφαιρικοί συλλέκτες. Το αποτέλεσμα είναι ότι ο ρύπος που θέλουμε να αφαιρέσουμε αδυνατεί να περάσει και έτσι έχουμε κατακράτηση του. Το πρόβλημα όμως με αυτό το μηχανισμό είναι ότι εάν συνεχίσει να συμβαίνει, τότε όλοι οι υπάρχων δίοδοι που δημιουργούνται στην επιφάνεια από τους συλλέκτες θα φράξουν τοπικά και έτσι το φίλτρο θα γίνει ακατάλληλο για επιπλέον χρήση. Στην πραγματικότητα όλα τα φίλτρα χρειάζονται αργά η γρήγορα αλλαγή, όμως δεν είναι επιθυμητό εξαιτίας μιας τοπικής απενεργοποίησης του φίλτρου, να καταστεί όλο ανενεργό.



Σχήμα 1.9 : Παρουσιάζεται ο μηχανισμός στράγγισης στερεών (πορτοκαλί κύκλοι) από ένα σύνολο σφαιρικών συλλεκτών (μπλε κύκλοι). Επειδή D (διάμετρος στερεού) >>L (διάκενο μεταξύ συλλεκτών) τα αιωρούμενα σωματίδια αδυνατούν να περάσουν και έχουμε συγκράτηση τους.

1.5.1 ΜΑΘΗΜΑΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΜΗΧΑΝΙΣΜΩΝ ΣΥΛΛΗΨΗΣ

Όλες οι συγκρούσεις των αιωρούμενων σωματιδίων με τους συλλέκτες δεν οδηγούν πάντα σε σύλληψη και συγκράτηση. Γι αυτό έχει οριστεί ο όρος η_c εξ. (1.23) (συντελεστής σύλληψης μοναδιαίου συλλέκτη) ο οποίος λαμβάνει υπόψη του την περίπτωση αποτυχίας συγκράτησης

$$\eta_c = \eta_o a \tag{1.23}$$

όπου a είναι ο συντελεστής απόδοσης συγκρούσεων (ο λόγος των επιτυχών συγκρούσεων, προς τις συνολικές συγκρούσεις που συνέβησαν), ενώ η_o εξ. (1.24) είναι ο συνολικός συντελεστής σύλληψης, ο οποίος βασίζεται στους 3 διαφορετικούς μηχανισμούς συγκράτησης (Yao et al., 1971):

$$\eta_o = \eta_D + \eta_I + \eta_G \tag{1.24}$$

με η_D να είναι ο συντελεστής σύλληψης ο οποίος εξαρτάται απο τον μηχανισμό διάχυσης, η_I συντελεστής ο οποίος εξαρτάται απο τον μηχανισμό ανάσχεσης και ομοιώς η_G συντελεστής σύλληψης ο οποίος εξαρτάται απο τον μηχανισμό καθίζησης.

Υποθέτοντας ότι το φίλτρο μας αποτελείται αποκλειστικά από σφαιρικούς συλλέκτες ομοιόμορφα διαταγμένους στον χώρο, μπορούμε να υπολογίσουμε θεωρητικά τους συντελεστές η_D, η_I και η_G εξ. (1.25, 1.30, 1.31) (Elimelech et al. 1995, p. 352).

$$\eta_D = 4.04\, A_s^{\frac{1}{3}} (Pe)^{-\frac{2}{3}} \tag{1.25}$$

Με Pe να είναι ο αδιάστατος αριθμός Peclet (1.26)

$$Pe = q\frac{d_c}{D} \tag{1.26}$$

Με d_c να είναι η διάμετρος του σφαιρικού συλλέκτη, q είναι η ταχύτητα κατά Darcy που δίδεται από την εξ. (1.27).

$$q = \frac{Q}{A_{κλίνης}} = U\Theta \tag{1.27}$$

Όπου $A_{κλίνης}$ είναι η επιφάνεια διατομής της κλίνης (φίλτρου), U πλέον είναι η ενδοπορώδης ταχύτητα (αυτή με την οποία το σωματίδιο πλησιάζει το συλλέκτη), θ είναι το πορώδες του φίλτρου και D είναι ο συντελεστής μοριακής διάχυσης. Ελλείψη πειραματικών δεδομένων κανείς μπορεί να υπολογίσει θεωρητικά το D χρησιμοποιώντας την εξ. (1.2) Stokes- Einstein.

Συνεχίζοντας στην εξ. (1.25) η παράμετρος A_ε , η οποία εξαρτάται από το πορώδες του φίλτρου δίδεται από την (1.28):

$$A_\varepsilon = \frac{2\,(1-\varepsilon^5)}{2-3\varepsilon\ +3\varepsilon^5-2\varepsilon^6} \tag{1.28}$$

Με το ε στην (1.28) να δίδεται από την εξ. (1.29).

$$\varepsilon = \ (1-\theta)^{\frac{1}{3}} \tag{1.29}$$

Με την σειρά του ο συντελεστής η_I (μηχανισμός ανάσχεσης) δίδεται (Yao et al. , 1971; Elimelech et al., 1995, p. 352):

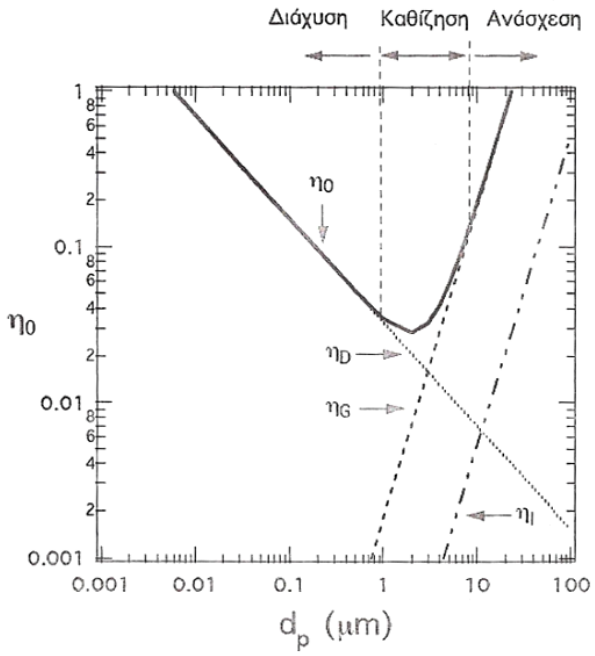
$$\eta_I = \frac{3}{2} A_\varepsilon \left(\frac{d_p}{d_c}\right)^2 \tag{1.30}$$

Όπου d_p να είναι η διάμετρος των αιωρούμενων στερεών, και d_c η διάμετρος του μοναδιαίου συλλέκτη.

Τέλος ο τρίτος όρος της εξίσωσης (1.24) μηχανισμός καθίζησης δίδεται από εξ. (1.31) (Yao et al., 1971):

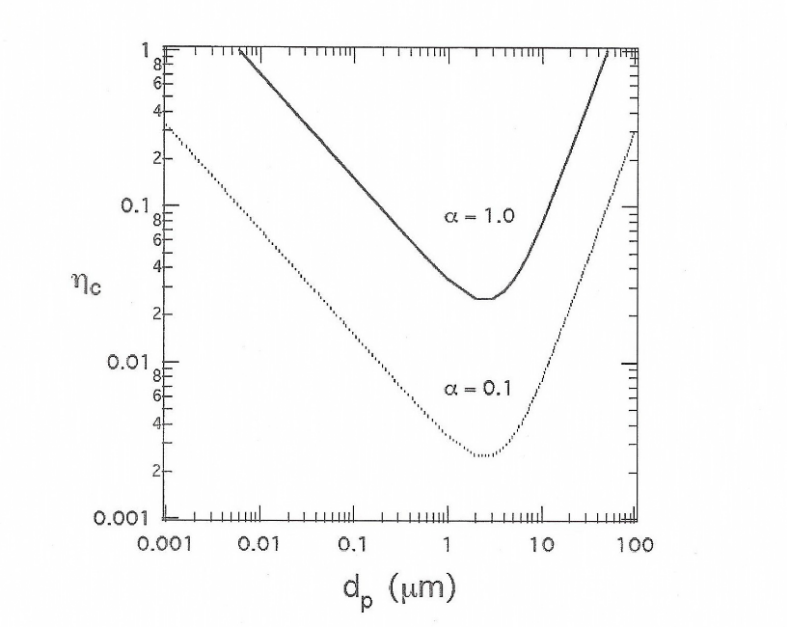
$$\eta_G = \frac{(\rho_p-\rho_w)g d_p^2}{18\ \mu_w q} \tag{1.31}$$

Το ποσοστό συνεισφοράς κάθε μηχανισμού στον τελικό συντελεστή σύλληψης η_o δεν είναι σταθερό αλλά εξαρτάται από την φύση του αιωρούμενου στερεού αλλά και του συλλέκτη. Έτσι στο Σχήμα 1.10 παρατηρείται μια εξάρτηση των η_D , η_I και η_G απο το μέγεθος των αιωρούμεων σωματιδίων. Πιο συγκεκριμένα παρατηρείται ότι όσο μεγαλώνει το μέγεθος, τόσο πιο σημαντικοί γίνονται οι μηχανισμοί ανάσχεσης (η_I) και καθίζησης (η_G) στον συνολικό συντελεστή σύλληψης η_o. Αντίθετα το μέγεθος επηρεάζει αρνητικά τον συντελεστή διάχυσης (η_D), φαινόμενο που εξηγείται φυσικά καθώς τα μεγαλύτερα μόρια είναι λιγότερο ευκίνητα από τα μικρότερα και έτσι είναι δυσκολότερα για τα πρώτα (μεγαλύτερα) να αλλάξουν τροχιά και να συγκρουστούν με το σφαιρικό συλλέκτη. Επιπλέον για την περίπτωση που εξετάζεται στο Σχήμα 1.10 φαίνεται ότι ο μηχανισμός διάχυσης επηρεάζει σωματίδια με μέγεθος μικρότερο από 1 μm. Με την σειρά του ο μηχανισμός ανάσχεσης παρουσιάζεται σε σωματίδια με μέγεθος μεγαλύτερο από 10 μm. Τέλος ο μηχανισμός ανάσχεσης είναι σημαντικός για σωματίδια με διάμετρο μεγαλύτερη από 1 μm.



Σχήμα 1.10: Συσχέτιση του συνολικού συντελεστή σύλληψης σωματιδίων από το μέσο διήθησης, με το μέγεθος των αιωρούμενων σωματιδίων ($d_c = 0.25\,mm,\quad g = 9.81\frac{m}{sec^2},\ T = 298\,k,\ U = 25.3\frac{cm}{hr},\ \theta = 0.3,\ \mu_w = 9.8 * 10^{-4}\ \frac{Kg}{m\ sec^2},\ \rho_w = \ 1000\ \frac{kg}{m^3},\ \rho_\rho = 1002\ \frac{kg}{m^3}$) (Χρυσικόπουλος, 2012).

Ο συντελεστής απόδοσης συγκρούσεων α εξ. (1.23) είναι δύσκολο να υπολογιστεί θεωρητικά και για αυτό χρησιμοποιείται συχνά η πειραματική προσέγγιση του. Ακόμα όπως παρουσιάζεται στο Σχήμα 1.11, για α<1 το ποσοστό σύλληψης σωματιδίων από τον συλλέκτη ελαττώνεται, γιατί ο συντελεστής η_c είναι μικρότερος του η_0 εξ. (1.23). Παρ' όλα αυτά όταν σε καθαρά φίλτρα διήθησης έχουμε προσθήκη (ικανής ποσότητας) κροκιδωτικού τότε ο συντελεστής α ισούται με την μονάδα. Αυτό συμβαίνει καθώς τα κροκιδωτικά εξαφανίζουν το διπλό στρώμα που περιβάλλει τα κολλοειδή (το οποίο δημιουργεί απωθητικές δυνάμεις), και κάθε σύγκρουση οδηγεί σε επιτυχή σύλληψη.



Σχήμα 1.11: Συσχέτιση του συντελεστή σύλληψης μοναδιαίου συλλέκτη η_c εξ. (1.23) με το μέγεθος αιωρούμενων σωματιδίων d_p για δύο διαφορετικές τιμές του συντελεστή α απόδοσης συγκρούσεων ($d_c = 0.25\,mm,\ g = 9.81\frac{m}{sec^2},\ T = 298\,k,\ U = 25.3\frac{cm}{hr},\ \theta = 0.3,\ \mu_w = 9.8 * 10^{-4}\frac{Kg}{msec^2}, \rho_w = \ 1000\ \frac{kg}{m^3},\ \rho_\rho = 1002\ \frac{kg}{m^3}$) (Χρυσικόπουλος, 2012).

Μια προσεγγιστική εξίσωση που περιγράφει επαρκώς τον όρο η_o εξ. (1.24) και χρησιμοποιείται αρκετά από την βιβλιογραφία είναι η (1.32) (Rajagopalan and Tien, 1976; Logan et al., 1995), για την 1.32 ισχύουν οι εξισώσεις (1.33-1.36):

$$\eta_0 = \ 4A_\varepsilon^{\frac{1}{3}}N_{pe}^{-\frac{2}{3}} + A_\varepsilon N_{LO}^{\frac{1}{8}}N_R^{\frac{15}{8}} + 0.00338A_\varepsilon N_G^{1.2}N_R^{-0.4} \quad (1.32)$$

$$N_{pe} = Pe \quad (1.33)$$

όπου Pe ο αριθμός Peclet,

$$N_R = \frac{d_p}{d_c} \quad (1.34)$$

N_R =λόγος των διαμέτρων αιωρούμενων σωματιδίων και συλλέκτη),

$$N_G = \eta_G \quad (1.35)$$

$$N_{Lo} = 4\frac{Ha}{9\,\pi\,\mu_w d_p^2 q} \quad (1.36)$$

Με $Ha = 6.6x10^{-21}kg\ \frac{m^2}{sec^2}$ να είναι ο συντελεστής Hamaker (Truesdail et al., 1998)

Με την ίδια λογική της εξ. (1.32) υφίσταται και μια εναλλακτική προσεγγιστική εξίσωση (1.37) (Tufenkji and Elimelech, 2004):

$$\eta_0 = 2.4\, A_\varepsilon^{\frac{1}{3}} N_R^{-0.081} N_{Pe}^{-0.715} N_{vdW}^{0.052} + 0.55\, A_\varepsilon N_R^{1.675} N_A^{0.125} + 0.22\, N_R^{-0.24} N_G^{1.11} N_{vdW}^{0.053} \quad (1.37)$$

Όπου για την εξ. (1.37) ισχύουν οι (1.38-1.39).

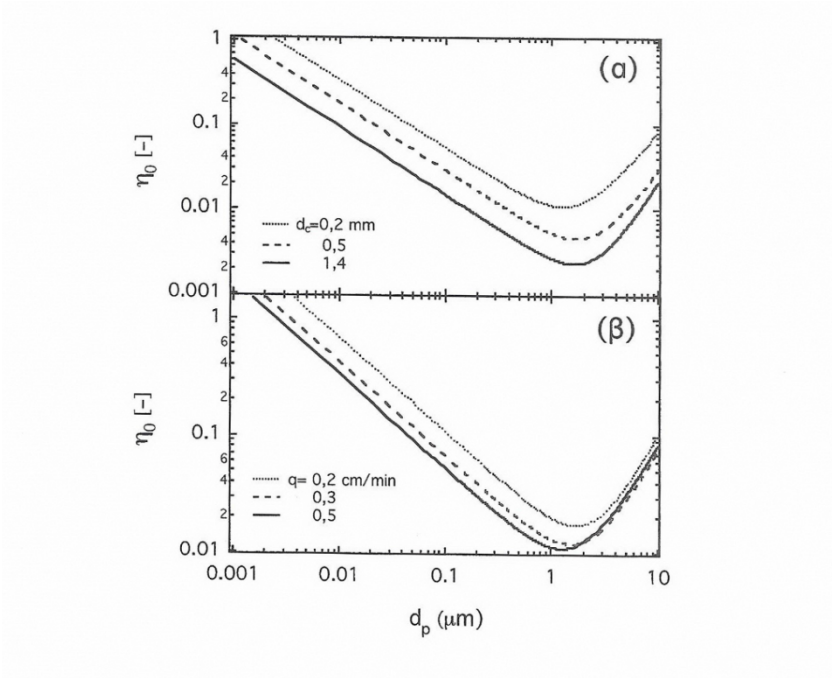
$$N_{vdW} = \frac{Ha}{K_B T} \quad (1.38)$$

K_B =Είναι η σταθερά Boltzmann

T= απόλυτη θερμοκρασία σε Kelvin

$$N_A = \frac{N_{vdW}}{N_R N_{PE}} \quad (1.39)$$

Στο Σχήμα 1.12 παρατηρούμε ότι έχουμε μείωση του συνολικού συντελεστή σύλληψης η_o εξ. (1.24) όταν η διάμετρος του σφαιρικού συλλέκτη d_c μεγαλώνει αλλά και η παροχή του νερού αυξάνεται (d_p διάμετρος αιωρούμενων στερεών).



Σχήμα 1.12: Συσχέτιση του συνολικού συντελεστή σύλληψης από το μέσο διήθησης με το μέγεθος των αιωρούμενων σωματιδίων για τρείς διαφορετικές για (α) τρείς διαφορετικές τιμές της διαμέτρου του σφαιρικού συλλέκτη και (β) τρείς διαφορετικές τιμές της επιφανειακής ταχύτητας του νερού (Χρυσικόπουλος, 2012).

Τέλος αφού με κάποιο τρόπο εξ. (1.32, 1.37) έχουν υπολογιστεί οι τιμές των η_0 και α εξ. (1.23-1.24) είναι δυνατή η χρήση τους στις εξισώσεις προσρόφησης. Πιο συγκεκριμένα ο όρος K1 της εξ. (1.21), μπορεί να γραφεί σε σχέση με το η_0 ώς εξής:

$$K1 = \frac{3}{2}\,(1-\theta)U\,\eta_0\alpha \tag{1.40}$$

(Anders and Chrysikopoulos, 2005; Harvey and Garabedian, 1991)

1.5.2 ΜΑΘΗΜΑΤΙΚΗ ΠΡΟΣΟΜΟΙΩΣΗ ΑΠΟΜΑΚΡΥΝΣΗΣ ΣΤΕΡΕΩΝ

Στο κεφάλαιο 1.5.1 είδαμε κυρίως θεωρητικές εξισώσεις που περιγράφουν την συγκράτηση αιωρούμενων στερεών από μοναδιαίο συλλέκτη. Για να γίνουμε λίγο πιο πρακτικοί, πλέον θεωρούμε κλίνη βαθειάς διήθησης, μήκους L, η οποία χρησιμοποιείτε για αφαίρεση σωματιδίων από το νερό. Στην περίπτωση αυτή μπορούμε να χρησιμοποιήσουμε την εξ. 1.40 ώστε να περιγραφεί η συγκράτηση των σωματιδίων (Iwasaki, 1937; Logan et al., 1995):

$$\frac{d_c}{d_L} = -\lambda\, C \tag{1.41}$$

C= συγκέντρωση στερεών

λ= συντελεστής φίλτρου

L=μήκος κλίνης διήθησης

Θεωρώντας ότι η κλίνη αποτελείται από πανομοιότυπους σφαιρικούς συλλέκτες, ομοιόμορφα διατεταγμένους στον χώρο, τότε συσχετίσουμε το η_c εξ. (1.23) με τον συντελεστή φίλτρου λ εξ.(1.41) χρησιμοποιώντας την εξ. (1.42)

$$\lambda = \frac{3}{2}\frac{1-\theta}{d_c}\,n_c \tag{1.42}$$

Λύνοντας την εξ. (1.40) σε συνδυασμό με την εξ. (1.41) προκύπτει η εξ. (1.43):

$$ln\left[\frac{c}{c_o}\right] = -\frac{3}{2}\,(1-\theta)\,\eta_c\,\left(\frac{\mathrm{L}}{\mathrm{d_c}}\right) \tag{1.43}$$

C_0 = αρχική συγκέντρωση αιωρούμενων στερεών

θ= πορώδες κλίνης

η_c = συντελεστής σύλληψης μοναδιαίου συλλέκτη, δίδεται από την εξίσωση (1.23).

Εδώ σημειώνεται ότι ο όρος $\frac{L}{d_c}$ αντιπροσωπεύει τον αριθμό διάκενων συλλεκτών σε σειρά.

1.6 ΔΥΝΑΜΕΙΣ ΜΕΤΑΞΥ ΚΟΛΛΟΕΙΔΩΝ

Είδαμε στο προηγούμενο κεφάλαιο 1.5, τους δυνατούς τρόπους σύλληψης. Ένας από αυτούς είναι και ο μηχανισμός καθίζησης εξ. (1.31). Το ερώτημα που τίθεται τώρα είναι εάν υπάρχει τρόπος ώστε να ενισχύσουμε και να βελτιώσουμε τον μηχανισμό αυτό. Για να γίνει κάτι τέτοιο θα πρέπει να περιγράψουμε και να κατανοήσουμε τις δυνάμεις που ασκούνται στα σωματίδια- κολλοειδή.

Με τον όρο Κολλοειδή εννοούμε σωματίδια που έχουν πολύ μικρό μέγεθος. Συνήθως η διάμετρος τους κυμαίνεται από 10 μm-0,001 μm (Chrysikopoulos and Sim, 1996).

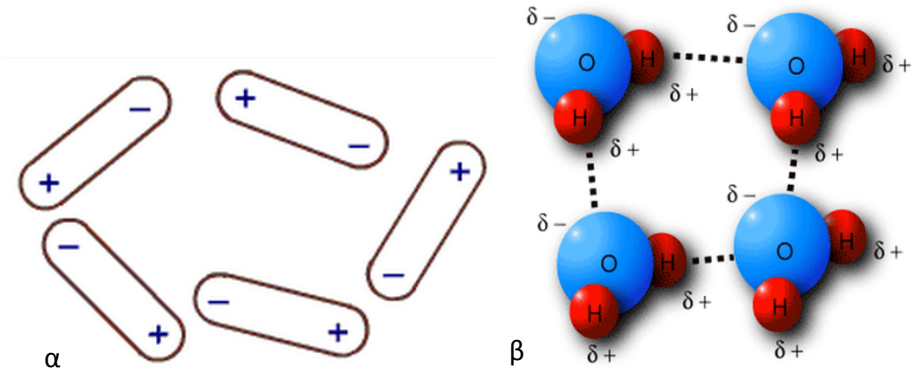
Τα κολλοειδή που αιωρούνται στο νερό συνήθως απαντώνται με αρνητικό φορτίο και για αυτό το λόγο απωθούνται μεταξύ τους. Όσο πιο μεγάλη είναι η μεταξύ τους απώθηση (δηλαδή μεγαλύτερα τα ομόσημα φορτία) τόσο πιο μεγάλη είναι η αντίσταση τους στην συσσωμάτωση και συνεπώς στην αποσταθεροποίηση τους.

Οι δυνάμεις που ασκούνται μεταξύ των κολλοειδών είναι κυρίως ηλεκτροστατικού τύπου, Van der Waals και θερμικής κίνησης Brown. Ασκούνται βέβαια και βαρυντικές δυνάμεις αλλά εξαιτίας του εξαιρετικά μικρού μεγέθους τους (μικρή μάζα), σε σχέση πάντα με τις άλλες τρείς κατηγορίες δυνάμεων, αυτές αμελούνται.

Οι δυνάμεις Brown οφείλονται στην τυχαία κίνηση των μορίων, η οποία όπως είναι φυσικό οδηγεί σε συγκρούσεις μεταξύ των μορίων. Όσο πιο μεγάλη είναι η θερμοκρασία τόσο πιο μεγάλη είναι η κινητική ενέργεια των μορίων και έτσι έχουμε πιο πολλές συγκρούσεις. Με την ίδια λογική οι συγκρούσεις αυξάνονται όταν το μέσο μας έχει μεγάλη πυκνότητα. Σημειώνεται ότι μέσω αυτών (των συγκρούσεων) αναπτύσσονται δυνάμεις στα μόρια οι οποίες δημιουργούν τυχαία συνισταμένη και οδηγούν τα μόρια σε άτακτη ευθύγραμμη ή και περιστροφική κίνηση.

Με την σειρά τους οι δυνάμεις Van der Waals περιγράφουν ένα σύνολο ασθενών διαμοριακών δυνάμεων: α) δυνάμεις διπόλου-διπόλου, β) δυνάμεις διπόλου εξ' επαγωγής, γ) δυνάμεις London. Για αρχή όλες οι δυνάμεις αυτές πήραν το όνομα τους από τον μεγάλο Ολλανδό φυσικό J.D. Van der Waals (1837-1923), ο οποίος τιμήθηκε με Νόμπελ Φυσικής το 1910, επειδή εξήγησε την απόκλιση που έχουν τα πραγματικά αέρια σε σχέση με τα ιδανικά. Για να το κάνει αυτό έπρεπε να περιγράψει τις διαμοριακές δυνάμεις που τελικά αναπτυσσόντουσαν. Έτσι διαμορφώθηκαν οι τρείς κατηγορίες αυτές δυνάμεων. Η πρώτη κατηγορία Σχήμα 1.13 υφίσταται εξαιτίας της ύπαρξης πολικών μορίων. Δηλαδή μόρια με ανομοιόμορφη κατανομή φορτίου, τα οποία και ευθυγραμμίζονται μεταξύ τους. Ο θετικός πόλος του ενός πλησιάζει τον αρνητικό πόλο του γειτονικού μορίου (τα ετερώνυμα έλκονται). Στην δεύτερη κατηγορία αρχικά έχουμε ένα μη πολικό μόριο το οποίο στην συνέχεια επηρεάζεται από το γειτονικό του (πολικό μόριο), πολώνεται, και έχουμε δυνάμεις μεταξύ δυο πολικών μορίων. Τέλος στην τρίτη κατηγορία οι δυνάμεις London, γνωστές ως δυνάμεις διασποράς, προκύπτουν από τα στιγμιαία δίπολα που δημιουργούνται εξαιτίας της κίνησης των ηλεκτρονίων γύρω από τους πυρήνες, με

αποτέλεσμα την ανομοιόμορφη κατανομή του φορτίου. Επισημαίνεται ότι οι δυνάμεις van der Waals είναι πολύ σημαντικές, καθώς καθορίζουν σε ποια κατάσταση θα βρίσκεται η ύλη σε συγκεκριμένη θερμοκρασία. Δηλαδή το εάν θα έχουμε στερεά, υγρή ή αέρια μορφή εξαρτάται από το μέγεθος των διαμοριακών δυνάμεων που επικρατούν εκείνη την στιγμή.



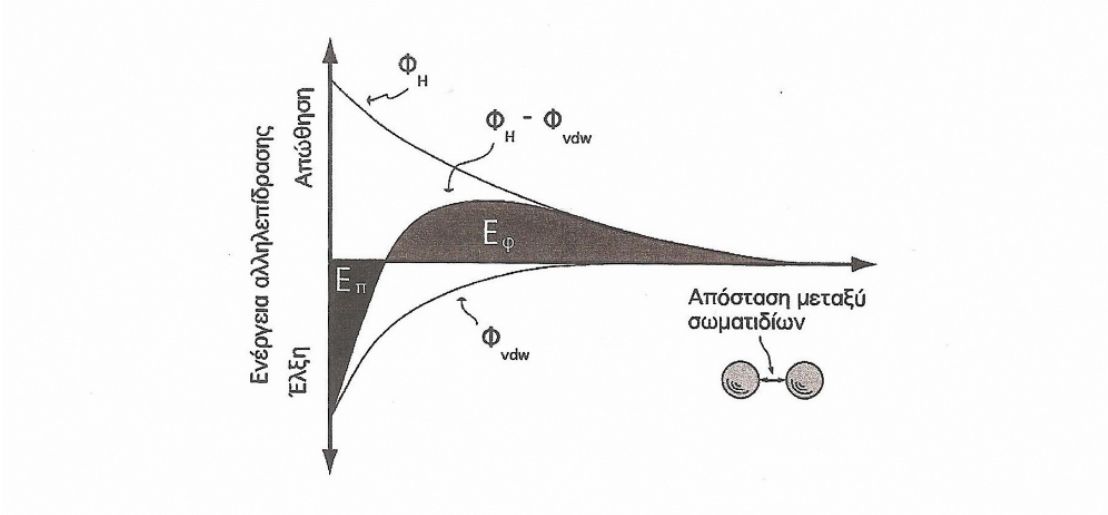
Σχήμα 1.13: Παρουσιάζονται μόρια με ανομοιόμορφη κατανομή φορτίου (πολικά): α) Ένα τυχαίο μόριο διατεταγμένο στο χώρο, β) Μόρια νερού (δ είναι τα φορτία που υφίστανται). Οι δυνάμεις που αναπτύσσονται είναι κυρίως van der Waals (Google Pics).

1.6.1 ΘΕΩΡΕΙΑ DLVO

Προσπαθώντας να περιγράψουν την δομή των κολλοειδών αλλά και παραμέτρους που διέπουν την συμπεριφορά τους στον χώρο οι Dejaquin, Landau, Verwey και Overbeek ανέπτυξαν την θεωρεία DLVO. Σε αυτήν περιορίζονται οι κατηγορίες των δυνάμεων που ασκούνται και τελικά μένουν δυο είδη: α) Απωστικές τα κολλοειδή απωθούνται μεταξύ τους, και είναι ηλεκτροστατικής φύσης, β) Ελκτικές, τα κολλοειδή έλκονται μεταξύ τους, και βασίζονται σε van der Waals δυνάμεις.

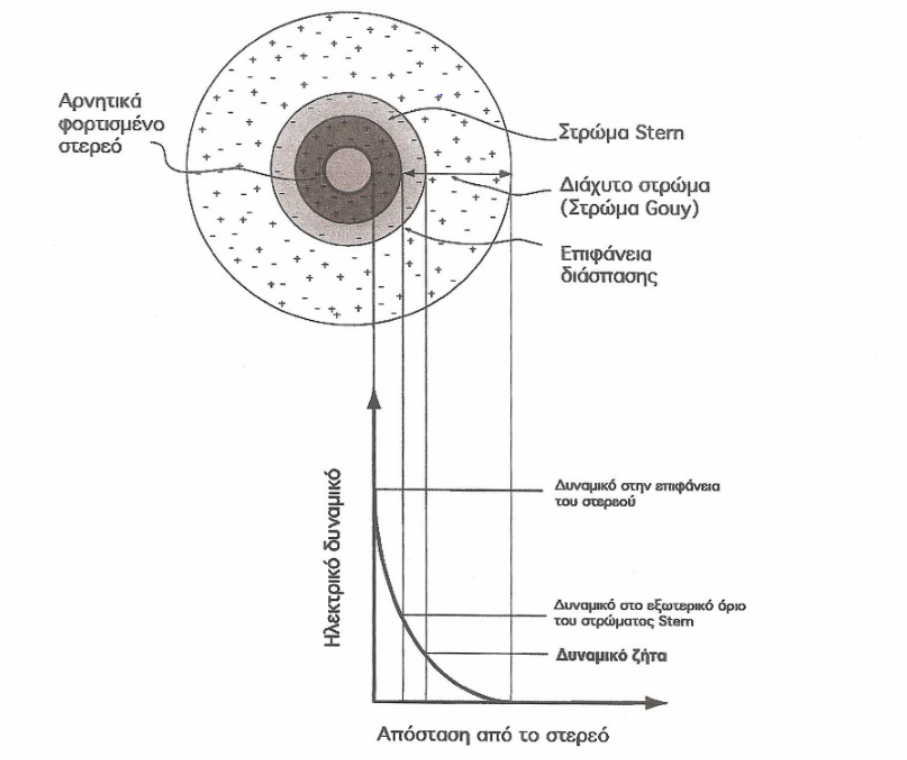
Αν κανείς προσπαθούσε να περιγράψει την συνισταμένη των δυνάμεων Σχήμα 1.14 (συνδυασμός ελκτικών και απωστικών) που αναπτύσσονται, σε σχέση με την απόσταση που έχουν δύο τυχαία κολλοειδή θα παρατηρούσε τα εξής. Όταν τα κολλοειδή είναι σχετικά μακριά οι δυνάμεις που επικρατούν είναι απωστικές, ενώ καθώς πλησιάζουν μετατρέπονται σε ελκτικές. Κάποια στιγμή αναπόφευκτα η συνολική δύναμη είναι μηδέν, η απόσταση που συμβαίνει αυτό είναι πολύ σημαντική. Καθώς υποδεικνύει την ελάχιστη απόσταση στην οποία πρέπει να φτάσουν τα κολλοειδή έτσι ώστε να επικρατήσουν οι ελκτικές δυνάμεις και να έχουμε συσσωμάτωση. Όμως για να φτάσουν στην απόσταση αυτή, χρειάζονται να έχουν αρκετή αρχική κινητική ενέργεια ώστε να την μετατρέψουν σε δυναμική και να πλησιάσουν μεταξύ τους. Επειδή το γινόμενο δύναμης επί απόσταση δίνει έργο, το

εμβαδό κάτω από την καμπύλη της συνισταμένης των δυνάμεων αντιπροσωπεύει το συνολικό έργο που έχει κάνει η συνισταμένη των δυνάμεων.



Σχήμα 1.14: Παρουσιάζεται σχηματική παράσταση ενέργειας αλληλεπίδρασης μεταξύ κολλοειδών εξαιτίας απωστικών και ελκτικών δυνάμεων (Φ_{vdw} είναι η ενέργεια εξαιτίας ελκτικών δυνάμεων Van der Waals και Φ_H είναι η ενέργεια της ηλεκτροστατικής άπωσης) (Χρυσικόπουλος, 2012).

Έχει παρατηρηθεί ότι στα φυσικά νερά τα κολλοειδή είναι αρνητικά φορτισμένα. Έτσι στην επιφάνεια τους όπως είναι αναμενόμενο, έλκονται θετικά ιόντα τα οποία και σχηματίζουν μια στοιβάδα, αυτή ονομάζεται Στρώμα Stern. Στην συνέχεια μια δεύτερη στοιβάδα ιόντων σχηματίζεται πάνω από το Στρώμα Stern, η οποία ονομάζεται διάχυτο στρώμα Gouy. Με την σειρά του το στρώμα Gouy αποτελείται από δύο μέρη. Ένα εσωτερικό, το οποίο και περιβάλει το (στρώμα) Stern, και ένα εξωτερικό που είναι ασθενέστερα συνδεδεμένο με το κολλοειδές. Εξαιτίας αυτού το τελευταίο (εξωτερικό στρώμα) δεν συνηθίζεται να συμπαρασύρεται μαζί με τις κινήσεις του κολλοειδούς. Το όριο μεταξύ του εσωτερικού και του εξωτερικού στρώματος ορίζεται από την διεπιφάνεια διάσπασης, της οποίας το ηλεκτρικό δυναμικό καλείται δυναμικό ζήτα. Όλα τα παραπάνω απεικονίζονται αναλυτικά στο Σχήμα 1.15.



Σχήμα 1.15: Παρουσιάζεται η δομή ενός κολλοειδούς σύμφωνα με την θεωρεία DVLO, καθώς και ένα διάγραμμα ηλεκτρικού δυναμικού σε συνάρτηση με την απόσταση από την επιφάνεια του στερεού (Χρυσικόπουλος, 2012).

Έχοντας πλέον κανείς κατανοήσει την δομή των κολλοειδών, είναι σε θέση να απαντήσει πώς είναι δυνατόν να υπάρξει βελτιστοποίηση στην διαδικασία της καθίζησης. Η απάντηση έρχεται με την χρήση των κροκιδωτικών. Ουσιών που είναι σε θέση να εξουδετερώνουν τα φορτία που περιβάλουν τα κολλοειδή και έτσι να μειώνεται το πάχος του διάχυτου στρώματος. Επιπλέον τα κροκιδωτικά μπορούν να προσροφηθούν πάνω στο ίδιο το κολλοειδές, στην αρνητικά φορτισμένη επιφάνεια τους και έτσι να τα εξουδετερώνουν. Ως εκ τούτου δεν επιτρέπουν το σχηματισμό του Στρώματος Stern και κάτω από αυτές τις συνθήκες τα κολλοειδή έχουν μειωμένο μέγεθος. Μπορούν και πλησιάζουν μεταξύ τους σε τέτοια απόσταση ώστε να επικρατούν οι ελκτικές δυνάμεις Van der Waals και όχι απωστικές ηλεκτροστατικής φύσης Σχήμα 1.14. Το αποτέλεσμα είναι συσσωμάτωση των κολλοειδών, αύξηση τοπικά της πυκνότητας τους και στην συνέχεια κάτω από την ισχυρή πλέον βαρυντική έλξη να έχουμε καθίζηση.

Το δυναμικό ζ που αναφέρθηκε ενωρίτερα μπορεί να υπολογιστεί πειραματικά στο εργαστήριο με ένα όργανο που ονομάζεται zetameter, το οποίο και μετατρέπει την ηλεκτροφορητική κινητικότητα (electrophoretic mobility), $U_E \rightarrow \left(\frac{m^2}{V*S}\right)$, σε δυναμικό ζ (Volts) χρησιμοποιώντας την ακόλουθη εξίσωση Smolouchoski εξ. (1.44) (Giese and Van Oss, 2002)

$$\zeta = \frac{4\pi\mu_w}{\varepsilon} U_E \qquad (1.44)$$

Όπου $\varepsilon \rightarrow \left(\frac{C^2}{j\ m} = \frac{C}{Vm}\right)$ είναι η διηλεκτρική σταθερά (dielectric constant) του υγρού. Η ηλεκτροφορητική κινητικότητα U_E εξ. (1.45) ορίζεται ως ο λόγος μεταξύ της ταχύτητας (m/s), του αιωρούμενου σωματιδίου, ως προς την ένταση, $E_e -> \frac{V}{m}$, του ηλεκτρικού πεδίου (Syngouna and Chrysikopoulos, 2010):

$$U_E = \frac{V}{E_e} \tag{1.45}$$

Το επιφανειακό δυναμικό, ψ[volts], αιωρούμενων σωματιδίων με ζ <60 mV, μπορεί να συσχετιστεί με το δυναμικό ζ με βάση την εξ. (1.46) (Van Oss eta al., 1990):

$$\psi = \zeta \left(1 + \frac{z}{r_p}\right) e^{kz} \tag{1.46}$$

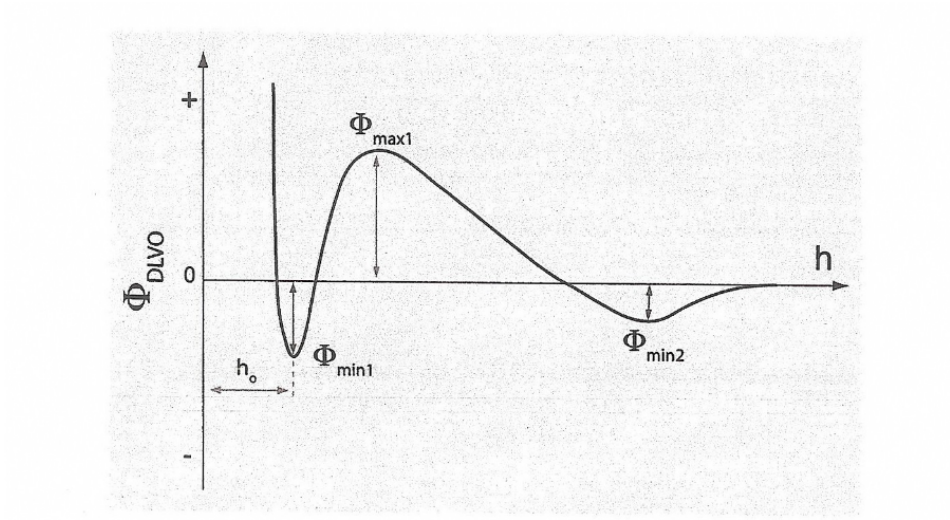
Με r_p[Å] να είναι η υδροδυναμική ακτίνα των αιωρούμενων στερεών, z≈3-5 Å είναι η απόσταση μεταξύ φορτισμένου σωματιδίου και της επιφάνειας διάσπασης (slipping plane) και κ^{-1} [Å] είναι το πάχος του διπλού στρώματος, το οποίο δίδεται από την παρακάτω εξ. (1.47) (Gouy, 1910, Ruckenstein and Priebe, 1976):

$$\kappa^{-1} = 10^{10} \left[2\ l_s\ N_A \frac{1000 e_c^2}{\varepsilon_r e_0 k_B T}\right]^{-\frac{1}{2}} \tag{1.47}$$

Με l_s να είναι η ιονική ισχύς [mol/L], $N_A = 6.022 * 10^{23}\ \frac{particles}{mol}$ είναι ο αριθμός Avogadro, $e_c = -1.6019 * 10^{-19} C$ είναι το φορτίο του ηλεκτρονίου, $k_B = 1.38066 * 10^{-23}\ \frac{J}{K}$ είναι η σταθερά Boltzmann, $\varepsilon_r = \frac{\varepsilon}{\varepsilon_0}$ (χωρίς διαστάσεις) είναι η διηλεκτρική σταθερά, $\varepsilon_0 \approx 8.854 * 10^{-12}\ \frac{C^2}{J\ m}$ είναι η διηλεκτρική σταθερά για το κενό, T είναι η απόλυτη θερμοκρασία σε [Kelvin] και οι όροι 10^{10} [Å/m], 1000 [L/m^3] χρησιμοποιούνται για την σωστή μετατροπή των μονάδων (οι σταθερές αυτές δεν είναι αδιάστατες).

Στην εξ. (1.46) υπολογίσαμε το επιφανειακό δυναμικό των αιωρούμενων σωματιδίων. Με την θεωρεία DLVO είναι δυνατό να περιγραφεί η συνολική ενέργεια διεπιφανειακού δυναμικού (interface potential energy), Σχήμα 1.16, μεταξύ ενός σφαιρικού σωματιδίου και μίας επίπεδης επιφάνειας που βρίσκονται σε απόσταση h [m], ή μεταξύ δυο σφαιρικών σωματιδίων. Τότε η ενέργεια αυτή ισούται με το άθροισμα των ενεργών διεπιφανειακών δυναμικών Van der Waals, Φ_{vdw}, διπλού στρώματος (double layer), Φ_{dl} και Born, Φ_{Born} (Loveland et al. 1996):

$$\Phi_{DLVO} = \Phi_{vdw} + \Phi_{dl} + \Phi_{Born} \tag{1.48}$$

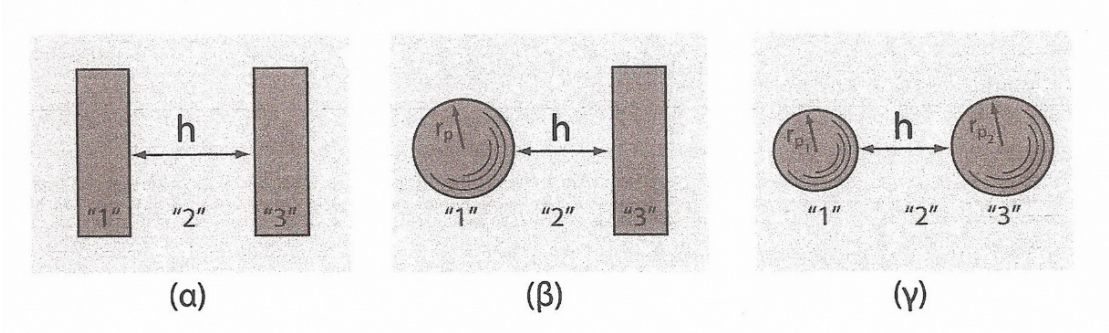


Σχήμα 1.16: Παρουσιάζεται η συνολική ενέργεια αλληλεπίδρασης μεταξύ δύο αιωρούμενων σωματιδίων συναρτήσει της μεταξύ τους απόστασης. Επίσης φαίνονται το πρωτοταγές ελάχιστο $\Phi_{min\,1} < 0$ το ενεργειακό φράγμα, $\Phi_{max\,1} > 0$ το δευτεροταγές ελάχιστο, $\Phi_{min\,2} < 0$ είναι το σημείο ελάχιστης απόστασης μεταξύ των δύο σωματιδίων, $h = h_0$, πέρα απο το οποίο κυριαρχούν ισχυρές απωθητικές δυνάμεις πυρηνικής φύσης (Χρυσικόπουλος, 2012).

Στο Σχήμα 1.16 παρουσιάζεται μια ποιοτική τυπική καμπύλη συνολικού διεπιφανειακού δυναμικού Φ_{DLVO}, μεταξύ δυο αιωρούμενων στο νερό σωματιδίων. Χαρακτηριστική είναι η ύπαρξη τοπικού ελαχίστου $\Phi_{\min 1}$ το οποίο εμφανίζεται σε μικρή απόσταση διαχωρισμού, το ενεργειακό φράγμα $\Phi_{\max 1}$ (το οποίο και πρέπει να ξεπεραστεί ώστε να βρεθούμε στο πρωτοταγές ελάχιστο) και το δευτεροταγές ελάχιστο $\Phi_{\min 2}$ που εμφανίζεται σε μεγαλύτερες αποστάσεις διαχωρισμού των σωματιδίων. Το πρωτοταγές ελάχιστο υφίσταται μόνο εάν σε κάποια θέση η $\Phi_{\mathrm{DLVO}} < 0$ και η ύπαρξη αυτού υποδηλώνει ότι είναι δυνατή η συσσωμάτωση των αιωρούμενων σωματιδίων. Στην περίπτωση που παντου $\Phi_{\mathrm{DLVO}} < 0$ τότε τα σωματίδια είναι ασταθή και στην πρώτη ευκαιρία, θα επέλθει συσσωμάτωση.

1.6.2 ΜΑΘΗΜΑΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΔΥΝΑΜΕΩΝ ΜΕΤΑΞΥ ΣΩΜΑΤΙΔΙΩΝ

Στην βιβλιογραφία υπάρχουν πολλές σχέσεις οι οποίες περιγράφουν εμπειρικά τις τιμές των Φ_{vdw} , Φ_{dl} και Φ_{Born} για διάφορα είδη σωματιδίων. Εμείς θα παρουσιάσουμε μόνο τις περιπτώσεις που φαίνονται στο Σχήμα 1.17.



Σχήμα 1.17: Παρουσιάζεται σχηματική παράσταση αλληλεπίδρασης μεταξύ α) δύο επίπεδων επιφανειών, β) μιας σφαίρας και μιας επιφάνειας, γ) μεταξύ δύο σφαιρών. Το σύμβολο "1" αντιπροσωπεύει το πρώτο σώμα που αλληλεπιδρά ενώ το σύμβολο "3" είναι το δεύτερο σώμα αλληλεπίδρασης. Το σύμβολο "2" είναι το μέσω αλληλεπίδρασης.

Η ενέργεια του δυναμικού van der Waals Φ_{vdw} για την περίπτωση όπου επιφάνεια αλληλεπιδρά με σφαίρα δίδεται από την παρακάτω σχέση εξ. 1.49 (Gregory, 1981):

$$\Phi_{vdw} = -\frac{A_{123}\, r_p}{6h}\left[1+\left(\frac{14h}{\lambda_w}\right)^{-1}\right] \tag{1.49}$$

Ενώ για την περίπτωση όπου έχουμε αλληλεπίδραση σφαίρας με σφαίρα η σχέση υπολογίστηκε από τον Hamaker και γίνεται (Feke et al., 1984, Ryan and Gschwend, 1994) εξ. (1.50):

$$\Phi_{vdw} = -\frac{A_{123}}{12}\left\{\frac{R_p}{\xi^2+\xi R_p+\xi+R_p}+2\ln\left[\frac{\xi^2+\xi R_p+\xi}{\xi^2+\xi R_p+\xi+R_p}\right]\right\} \tag{1.50}$$

Όπου στις εξ. (1.49, 1.50) ο όρος A_{123} είναι ο συντελεστής Hamaker $\left[\mathrm{J}=\mathrm{kg}\frac{\mathrm{m}^2}{\mathrm{s}^2}\right]$, $\lambda_w = 10^{-7}$ είναι το χαρακτηριστικό μήκος κύματος για την περίπτωση αλληλεπίδρασης σφαίρας με επίπεδη επιφάνεια (Loveland et al., 1996), r_p [m] είναι η ακτίνα σφαιρικού σωματιδίου, επιπλέον οι όροι R_p, ξ δίδονται απο τις εξ. (1.51, 1.52)

$$R_p = \frac{r_{p2}}{r_{p1}} \tag{1.51}$$

$$\xi = \frac{h}{2\, r_{p1}} \tag{1.52}$$

Οι κάτω δείκτες στις εξ. (1.51, 1.52) συμβολίζουν τα διαφορετικά σφαιρικά σωματίδια. Ο όρος A_{123} συμβολίζει τον συντελεστή Hamaker για τα σωματίδια "1" και "3" σε ένα μέσο διασποράς "2". Σημειώνεται ότι ο συντελεστής Hamaker δεν μπορεί να υπολογιστεί εύκολα και με ακρίβεια (Norde, 2003). Παρ όλα αυτά ένας συνηθισμένος τρόπος υπολογισμού του είναι μέσω του παρακάτω συνδυαστικού κανόνα (Israelachvili, 1992) εξ. (1.53):

$$A_{123} = \left(\sqrt{A_{11}} - \sqrt{A_{22}}\right)\left(\sqrt{A_{33}} - \sqrt{A_{22}}\right) \tag{1.53}$$

Με A_{11} να είναι ο συντελεστής Hamaker για το σωματίδιο 1 όταν αλληλεπιδρά με τον εαυτό του, το ίδιο ισχύει και για τα σύμβολα A_{22} και A_{33}. Επίσης ο συντελεστής Hamaker μπορεί να υπολογιστεί συναρτήσει ασύμμετρων αλληλεπιδράσεων A_{121} απο τον κανόνα εξ. (1.54) (Yoon et al., 1977). Προσοχή οι σχέσεις (1.52, 1.53) ισχύουν για δυναμικά <60 mV.

$$A_{123} = \left(\sqrt{A_{121}} - \sqrt{A_{323}}\right) \tag{1.54}$$

Στις εξ. (1.48, 1.49) υπολογίσαμε την ενέργεια του δυναμικού με βάση δυνάμεις Van der Waals, τώρα θα υπολογίσουμε την ενέργεια διεπιφανειακού δυναμικού Φ_{dl} [Jouls] για την περίπτωση σφαίρας με επίπεδη επιφάνεια εξ. (1.54) (Hogg et al., 1966) :

$$\Phi_{dl} = \pi\varepsilon_r\varepsilon_0 r_p \left[2\psi_p\psi_s \ln\left(\frac{1+e^{-kh}}{1-e^{-kh}}\right) + \left(\psi_p^2 + \psi_s^2\right)\ln\left(1-e^{-2kh}\right)\right] \tag{1.55}$$

ενώ για την περίπτωση που έχουμε αλληλεπίδραση σφαίρας με σφαίρα το Φ_{dl} δίδεται απο την εξ. (1.55):

$$\Phi_{dl} = \frac{\pi\varepsilon_r\varepsilon_0 r_{p_1} r_{p_2}}{r_{p_1}+r_{p_2}} \left[2\psi_{p_1}\psi_{P_2} \ln\left(\frac{1+e^{-kh}}{1-e^{-kh}}\right) + \left(\psi_{p_1}^2 + \psi_{p_2}^2\right)\ln\left(1-e^{-2kh}\right)\right] \tag{1.56}$$

όπου ψ_p [Volts] είναι το επιφανειακό δυναμικό (surface potential) του σφαιρικού σώματος και ψ_p [Volts] το επιφανειακό δυναμικό επιφανείας αντίστοιχα.

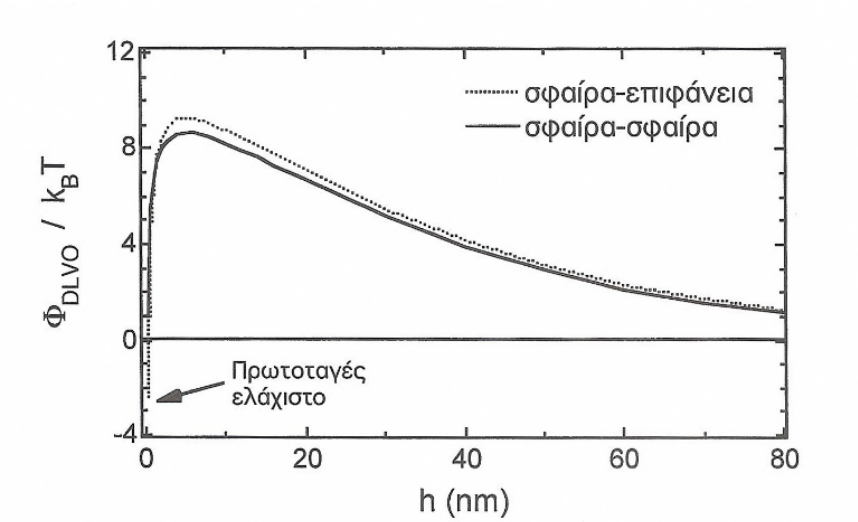
Τέλος ο τρίτος όρος της εξίσωσης (1.48), Φ_{Born} (επιφανειακό δυναμικο Born), για την περίπτωση σφαίρα-επιφάνεια, δίδεται από την εξ. (1.57) (Ruckenstein and Priebe, 1976):

$$\Phi_{Born} = \frac{A_{123}\sigma_{Born}^6}{7560}\left[\frac{8r_p+h}{\left(2\,r_p+h\right)^7} + \frac{6r_p-h}{h^7}\right] \tag{1.57}$$

Ενώ για την περίπτωση αλληλεπίδρασης σφαίρα με σφαίρα χρησιμοποιείτε η σχέση (1.58) (Feke et al., 1984, Ryan and Gschwend, 1994):

$$\Phi_{Born} = \frac{A_{123}}{7560\,\xi}\frac{\sigma_{Born}^2}{r_{p_1}^2}\left[\frac{-4\xi^2-14\left(R_p-1\right)\xi-6\left(R_p^2-7R_p+1\right)}{\left(2\xi-1+R_p\right)^7} + \frac{-4\xi^2+14\left(R_p-1\right)\xi-6\left(R_p^2-7R_p+1\right)}{\left(2\xi-1+R_p\right)^7} + \right.$$
$$\left.\frac{-4\xi^2+14\left(R_p-1\right)\xi+6\left(R_p^2+7R_p+1\right)}{\left(2\xi-1+R_p\right)^7} + \frac{-4\xi^2-14\left(R_p-1\right)\xi+6\left(R_p^2+7R_p+1\right)}{\left(2\xi-1+R_p\right)^7}\right] \tag{1.58}$$

Στις εξ. (1.57,1.58) ο όρος σ_{Born} [m] αντιπροσωπεύει την παράμετρο σύγκρουσης Born. Επιπλέον η Φ_{Born} σε σχέση με τους άλλους δύο όρους της εξ. (1.48) είναι αμηλετέα για όσο το h>1 nm.



Σχήμα 1.18: Παρουσιάζεται σχηματική παράσταση αλληλεπίδρασης μεταξύ σφαίρας-σφαίρας και σφαίρας-επιφάνειας συναρτήσει απόστασης διαχωρισμού h. Οι υπολογισμοί έγιναν με βάση τα ακόλουθα αριθμητικά δεδομένα: $T = 298\,K,\quad \sigma_{Born} = 0.5\,nm, r_p = r_{p_1} = 13nm, i_s = 0.0001\frac{mol}{L}$, $N_A = 6.022\,10^{-23}\frac{par.}{mol}$, $e_c = 1.602\,10^{-19}C$, $K_B = 1.38\frac{10^{-23}J}{K}, \varepsilon_r = \frac{\varepsilon}{\varepsilon_0} = 78.4, \varepsilon_0 = 8.85\frac{10^{-12}C^2}{J\,m}$, $\lambda_w = 10^{-7}m$, $\zeta_p = \zeta_{p_1} = -31.78\,mV$ *και* $\zeta_s = \zeta_{p_2} = 20.5\,mV$, $A_{123} = 7.5\,10^{-21}\,J$

Χρησιμοποιώντας τις απαραίτητες εξισώσεις (1.49-1.50), (1.55-1.56) και (1.57-1.58) υπολογίστηκε το συνολικό δυναμικό εξ.(1.48) Φ_{DLVO}, για δυο διαφορετικές αλληλεπιδράσεις: α) Σφαίρα με σφαίρα, β) Σφαίρα με επιφάνεια. Συγκρίνοντας τα αποτελέσματα Σχήμα 1.18 προκύπτει ότι οι δύο καμπύλες δεν έχουν ουσιαστικές διαφορές. Παρ' όλα αυτά μια προσεκτική εξέταση του σχήματος 1.18 δείχνει ότι η καμπύλη α (σφαίρα-σφαίρα) δεν έχει πρωτοταγές ελάχιστο, αντίθετα η καμπύλη β (σφαίρα - επιφάνεια) έχει. Στην ίδια λογική η καμπύλη α δεν έχει καθόλου αρνητικές τιμές. Πράγμα που σημαίνει ότι για τις συνθήκες που εμείς επιλέξαμε να μελετήσουμε τα σφαιρικά σωματίδια είναι πλήρως ευσταθή και δεν υπάρχει περίπτωση συσσωμάτωσης. Το αντίθετο ακριβώς ισχύει για την περίπτωση β. Τέλος επισημαίνεται ότι εάν οι δύο σφαίρες έχουν μεγάλη διαφορά μεγέθους, τότε για την περιγραφή τους είναι δυνατή η χρήση των εξισώσεων που διέπουν την αλληλεπίδραση σφαίρας-επιφάνειας (Syngouna and Chrysikopoulos, 2010, Vasiliadou and Chrysikopoulos, 2011).

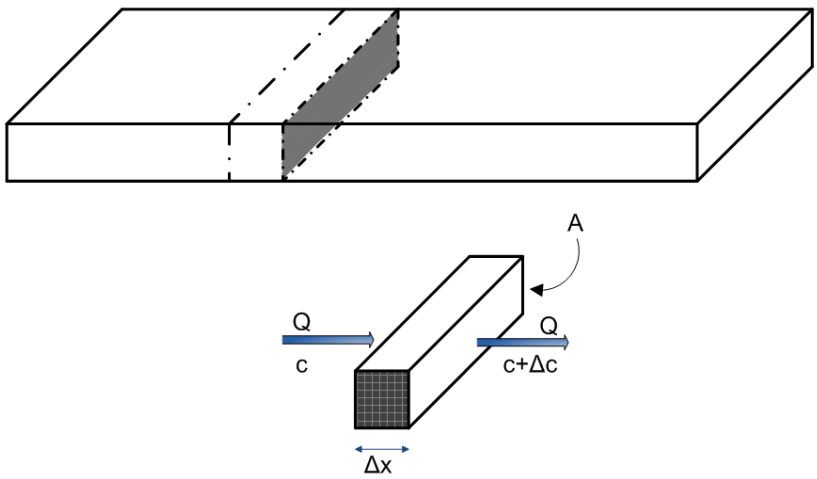
2. ΕΞΙΣΩΣΗ ΑΠΛΗΣ ΜΕΤΑΦΟΡΑΣ ΡΥΠΩΝ ΣΕ ΤΡΕΙΣ ΔΙΑΣΤΑΣΕΙΣ

2.1 ΑΠΛΗ ΜΟΝΟΔΙΑΣΤΑΤΗ ΕΞΙΣΩΣΗ ΜΕΤΑΦΟΡΑΣ ΡΥΠΩΝ

Για να εξαχθεί η μονοδιάστατη εξίσωση μεταφοράς ρύπων θα χρησιμοποιήσουμε την θεμελιώδη εξίσωση διατήρησης μάζας:

- [Συσσώρευση μάζας]=[εισροή]-[εκροή]+[παραγωγή]-[εξαφάνιση] (2.1)

Για να την εφαρμόσουμε θα απομονώσουμε ένα στοιχειώδες τμήμα, μήκους Δx (Σχ. 2.1), από το πορώδες μέσο και σε αυτό θα χρησιμοποιήσουμε το Ισοζύγιο μάζας σε χρόνο Δt.



Σχήμα 2.1:Στοιχειώδες απομονωμένο τμήμα μήκος Δx.

Για τους διάφορους όρους έχουμε:

Εισροή μάζας (διάχυση και μεταγωγή (in)):

$$\text{Εισροή}= Qc\Delta t+ J_x A\Delta t \quad (2.2)$$

Εκροή (διάχυση και μεταγωγή (out)):

$$\text{Εκροη}= Q(c+\Delta c)\Delta t + J_{x+\Delta x} A\Delta t \quad (2.3)$$

Θεωρείται ότι στο παρών παράδειγμα δεν υπάρχει Πηγή-Παραγωγή :

Πηγή=0 (2.4)

Εξαφάνιση (sink):

Εξαφάνιση=$V\rho_b\Delta c^*$ (2.5)

Συσσώρευση:

Συσσώρευση= $nV\Delta c$ (2.6)

Χρησιμοποιώντας τις εξισώσεις (2.1)-(2.6) έχουμε:

$$[nV\Delta c] = [Qc\Delta t + J_x A\Delta t] - [Q(c+\Delta c)\Delta t + J_{x+\Delta x} A\Delta t] + [0] - [V\rho_b \Delta c^*] \quad (2.7)$$

Όπου για την σχέση (2.7) ισχύει ότι ρ_b είναι η πυκνότητα των στερεών (μάζα στερεών ανά συνολικό όγκο), c είναι η συγκέντρωση της διαλυμένης ουσίας (μάζα ουσίας προς συνολικό όγκο), c^* είναι η συγκέντρωση της προσροφημένης ουσίας (μάζα ουσίας ανά μάζα στερεών), U είναι η ταχύτητα διήθησης ή ενδοπορώδης ταχύτητα, V αντιπροσωπευτικός όγκος του στοιχειώδες τμήματος μήκους Δx, J είναι η παροχή μάζας, n είναι το πορώδες, Δt το χρονικό βήμα στο οποίο μελετάμε την μεταφορά μάζας.

Χρησιμοποιώντας τον πρώτο νόμο του Fick (εξ. 1.1) με D όχι σαν συντελεστή μοριακής διάχυσης αλλά με D_x= συντελεστή υδροδυναμικής διασποράς (εξ. 1.7) μπορούμε να γράψουμε ότι:

$$J_x = -n\, D_x \frac{\partial c}{\partial x} \quad (2.8)$$

Διαιρώντας την (2.7) με $nV\Delta t$

$$\frac{\Delta c}{\Delta t} + \frac{\rho_b}{n}\frac{\Delta c^*}{\Delta \partial t} = -U\frac{\Delta c}{\Delta x} - \frac{J_{x+\Delta x} - J_x}{n\Delta x} \quad (2.9)$$

Εφαρμόζοντας και την εξ. (2.8) όταν $\Delta t \to 0$, $\Delta x \to 0$ προκύπτει ότι:

$$\frac{\partial c}{\partial t} + \frac{\rho_b}{n}\frac{\partial c^*}{\partial t} = -U\frac{\partial c}{\partial x} + \frac{\partial}{\partial x}\left(D_x \frac{\partial c}{\partial x}\right) \quad (2.10)$$

Χρησιμοποιώντας ένα γραμμικό μοντέλο ισόθερμης προσρόφησης (εξ. 1.9) τελικά παίρνουμε ότι:

$$\left(1 + \frac{\rho_b K_d}{n}\right)\frac{\partial c}{\partial t} = -U\frac{\partial c}{\partial x} + D_x \frac{\partial^2 C}{\partial X^2} \quad (2.11)$$

Όπου ο συντελεστής της μερικής παραγώγου της συγκέντρωσης ως προς τον χρόνο $\left(1 + \frac{\rho_b K_d}{n}\right)$ (εξ. 2.11) να ισούται με τον συντελεστή επιβράδυνσης R εξ. (2.12):

$$R = \left(1 + \frac{\rho_b K_d}{n}\right) \quad (2.12)$$

2.2 ΤΡΙΣΔΙΑΣΤΑΤΗ ΕΞΙΣΩΣΗ ΜΕΤΑΦΟΡΑΣ ΡΥΠΩΝ

Χρησιμοποιώντας την μορφή της εξίσωσης (2.11) μπορούμε να γενικεύσουμε στις τρεις διαστάσεις την μεταφορά ρύπου. Έτσι χρησιμοποιώντας:

I. Διάχυση σε τρείς διαστάσεις (Diffusion).

II. Μεταγωγή στην διεύθυνση X (Advection) (εξ 1.12).

III. Παρουσία πηγής.

IV. Αποδόμηση διαλυμένης και προσροφημένης ουσίας με συντελεστές λ και λ^* αντίστοιχα.

$$\frac{\partial C(t,x,y,z)}{\partial t} + \frac{\rho}{\theta}\frac{\partial C^*(t,x,y,z)}{\partial t} - D_x\frac{\partial^2 C(t,x,y,z)}{\partial x^2} - D_y\frac{\partial^2 C(t,x,y,z)}{\partial y^2} - D_z\frac{\partial^2 C(t,x,y,z)}{\partial z^2} + U\frac{\partial C(t,x,y,z)}{\partial x} + \lambda\, C(t,x,y,z) + \frac{\lambda^*\rho}{\theta}C^*(t,x,y,z) = F(t,x,y,z) \quad (2.13)$$

Για την εξίσωση (2.13) ισχύει ότι:

$$F(t,x,y,z) = G\,W \quad (2.14)$$

Όπου G είναι η ισχύς της πηγή. Έχει διαστάσεις ανάλογα με τον σχηματισμό αυτής. Έτσι εάν η πηγή είναι σημειακή τότε G=μάζα διαλυμένης ουσίας στην μονάδα του χρόνου. Εάν η πηγή είναι επιφανειακή τότε G=μάζα διαλυμένης ουσίας ανά Χρόνο ανά μονάδα επιφανείας. Αντίστοιχα εάν η πηγή έχει όγκο τότε G=μάζα διαλυμένης ουσίας ανά Χρόνο ανά μονάδα όγκου.

Με την σειρά του το W δηλώνει αυτόν ακριβώς τον σχηματισμό της πηγής. Στην περίπτωση που έχουμε σημειακή πηγή ισχύει (Sim and Chrysikopoulos 1998):

$$W(x,y,z) = \frac{1}{\theta}\,\delta(x - l_{x0})\delta(y - l_{y0})\delta(z - l_{z0}) \quad (2.15)$$

Εάν έχουμε ελλειπτική πηγή και ισχύει ότι $\frac{(x-l_{x0})^2}{\alpha^2} + \frac{(y-l_{y0})^2}{b^2} \le 1$ τότε

$$W(x,y,z) = \frac{\delta(z-l_{z0})}{\theta} \quad (2.16)$$

διαφορετικά εάν $\frac{(x-l_{x0})^2}{\alpha^2} + \frac{(y-l_{y0})^2}{b^2} \ge 1$ τότε

$$W(x,y,z) = 0 \quad (2.17)$$

Όπου για τις εξισώσεις (2.15)-(2.17) ισχύει ότι $\delta(x - l_{x0})$ είναι η εξίσωση του Dirac Delta, l_{x0}, l_{y0}, l_{z0} είναι οι Καρτεσιανές συντεταγμένες της πηγής για την οποία αναφερόμαστε κάθε φορά, α είναι ο ημιάξονας της ελλειπτικής πηγής παράλληλος στον x άξονα και b είναι ο ημιάξονας της ελλειπτικής πηγής παράλληλος στον yάξονα (Sim and Chrysikopoulos 1998).

2.3 ΑΡΧΙΚΕΣ ΚΑΙ ΣΥΝΟΡΙΑΚΕΣ ΣΥΝΘΗΚΕΣ

Για να λυθεί η εξ. (2.13) θα πρέπει να προσδιοριστούν οι αρχικές και συνοριακές συνθήκες. Οι αρχικές μας περιγράφουν την κατάσταση που επικρατούσε στο μοντέλο μας πριν αρχίσει να μετράει ο χρόνος. Ενώ οι συνοριακές συνθήκες μας περιγράφουν πως το εξωτερικό περιβάλλον επηρεάζει και επιδρά επάνω σε αυτό (στο μοντέλο μας).

Υπάρχουν τριών ειδών συνοριακές συνθήκες (Zheng and Bennett, 1995):

I. Οι συνθήκες τύπου Dirichlet. Σε αυτές, οι συγκεντρώσεις είναι ορισμένες κατά μήκος ενός συνόρου. Αυτού του είδους οι συνθήκες μπορούν να χρησιμοποιηθούν σαν πηγή η οποία εισάγει μάζα στο σύστημα ή σαν πηγάδι δηλαδή σαν μέσο αφαίρεσης διαλυμένης ουσίας από το σύστημα (2.18).

$$C(x,y,z,t)=C(x_i,y_i,z_i),\ t>0 \qquad (2.18)$$

II. Οι συνθήκες τύπου Neumann. Σε αυτέ οι κλίσεις συγκεντρώσεων είναι ορισμένες κατά μήκος ενός συνόρου (2.19).

$$-D_{ij}\frac{\partial c(x,y,z,t)}{\partial x_j} = f(x,y,z),\ t>0 \qquad (2.19)$$

III. Οι συνθήκες τύπου Cauchy. Σε αυτές είναι προσδιορισμένες όχι μόνο συγκεντρώσεις αλλά και οι κλίσεις αυτών κατά μήκος ενός συνόρου (2.20).

$$-D_{ij}\frac{\partial(x,y,z,t)}{\partial x_j} + v_i C = g_i(x_i,y_i,z_i),\ t>0 \qquad (2.20)$$

Όπου $g_i(x_i,y_i,z_i)$ είναι μια γνωστή συνάρτηση η οποία αντιπροσωπεύει την ολική ροή μάζας στον σύστημα η οποία οφείλεται και σε μεταγωγή και σε μοριακή διάχυση. D_{ij} = συντελεστής μοριακής διάχυσης και v_i= ενδοπορώδης ταχύτητα.

Όσο αναφορά τις αρχικές συνθήκες η πιο συνηθισμένη μορφή τους είναι αυτή της μηδενικής αρχικής συγκέντρωσης. Δηλαδή υπήρχε απουσία της διαλυμένης ουσίας σε όλη την έκταση του μοντέλου πριν αρχίσει να μετράει ο χρόνος εξ. (2.21).

$$C(0,x,y,z)=0 \qquad (2.21)$$

Οι συνθήκες που χρησιμοποιούνται συχνά για μοντελοποιήσεις και επίλυση αναλυτικών λύσεων είναι οι εξισώσεις (2.21)-(2.26).

$C(t, \pm\infty, y, z)=0$ (2.22)

$C(t, x, \pm\infty, z)=0$ (2.23)

$\frac{\partial C(t,x,y,0)}{\partial Z}=0$ (2.24)

$\frac{\partial C(t,x,y,\infty)}{\partial Z}=0$ (2.25)

$\frac{\partial C(t,x,y,H)}{\partial Z}=0$ (2.26)

Οι εξισώσεις (2.22) και (2.23) δείχνουν ότι το πορώδες μέσο είναι άπειρο κατά την διεύθυνση X και Y. Δηλαδή είναι τόσο μεγάλο ώστε για οποιονδήποτε χρόνο η διαλυμένη ουσία δεν έχει προλάβει να φτάσει ακόμα στα άκρα του. Η εξίσωση 2.24 δηλώνει ότι υπάρχει αδιαπέρατο στρώμα πάνω στο επίπεδο XY σε ύψος Z=0 από την αρχή των αξόνων. Ενώ η (2.26) περιγράφει αδιαπέρατο στρώμα σε ύψος Z=H . Τέλος η (2.25) μεταφράζεται σε ημιάπειρο πορώδες κατά την διεύθυνση Z.

3. ΣΥΜΜΕΤΑΦΟΡΑ ΚΑΙ ΜΑΘΗΜΑΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΗΣ

3.1 ΘΕΩΡΕΙΑ ΣΥΜΜΕΤΑΦΟΡΑΣ

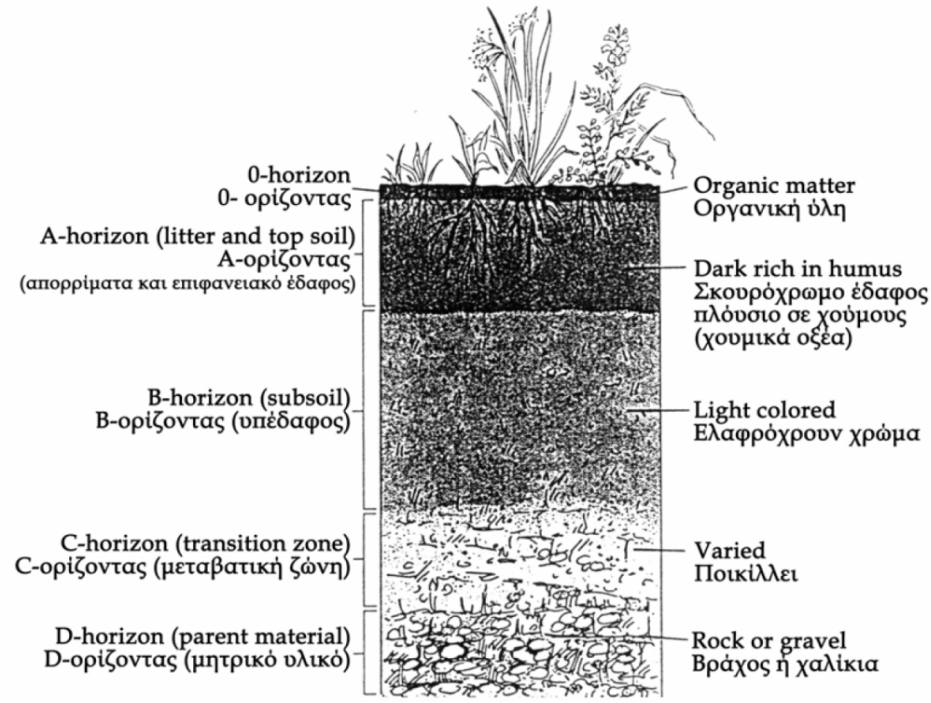
Ορισμοί βασικών εννοιών :

A. Συμμεταφορά ορίζεται ως η ταυτόχρονη μεταφορά δυο ή παραπάνω ρύπων που όμως, τουλάχιστον ο ένας επιδρά στη μεταφορά του άλλου.

B. Με τον όρο ρύπο εννοούμε κάθε διαλύτη (υδρόφιλη π.χ. ανόργανα άλατα) ή αδιάλυτη (υδρόφοβη, π.χ. υδρογονάνθρακες, διαλύτες κ.τ.λ.π) στο νερό ουσία, η οποία όταν εισάγεται στο περιβάλλον από ανθρώπινες δραστηριότητες, προκαλεί δυσμενείς επιπτώσεις.

C. Το έδαφος ορίζεται σαν ένα σύμπλοκο μίγμα ανόργανων υλικών, οργανικής ύλης που αποσυντίθεται ή σχηματίζει σύμπλοκα χουμικά οξέα, νερού, αέρα και ζωντανών μικροοργανισμών. Είναι δε ένα ανοικτό περιβαλλοντικό τμήμα που βρίσκεται σε συνεχή ανταλλαγή με την ατμόσφαιρα, την υδρόσφαιρα και την βιόσφαιρα. Σε αυτό διαμορφώνονται στιβάδες που καλούνται ορίζοντες (horizons), και οι οποίοι έχουν διαφορετική υφή και σύσταση Σχήμα 3.1:

   I. 0-Ορίζοντας: Ανώτατο στρώμα εδάφους με φυτά, οργανικά υπολείμματα, πεσμένα φύλλα δένδρων και μερικώς αποσυντιθέμενη οργανική ύλη.
   II. A-Ορίζοντας: Τα πρώτα 30-50 cm εδάφους (topsoil) με χουμικά οξέα, μερικά ανόργανα ορυκτά, ζωντανούς οργανισμούς, οργανική ύλη, με την μεγαλύτερη βιολογική δραστικότητα από όλες τις άλλες στιβάδες.
   III. E-Ορίζοντας: Η ζώνη που διαχωρίζει το επιφανειακό έδαφος από το υπέδαφος. Η διαλυμένη ή αιωρούμενη ύλη κινείται προς την στιβάδα αυτή και γι αυτό καλείται η ζώνη αποπλυμάτων (leaching zone)
   IV. B-Ορίζοντας: Το υπέδαφος είναι ορίζοντας εμπλουτισμού όπου συγκεντρώνονται τα χουμικά οξέα, ο άργιλος (πηλός), σίδηρος και αργίλιο μετά το στράγγισμα τους από τις επάνω ζώνες.
   V. C-Ορίζοντας: Ελαφρά διαβρωμένο βραχώδες έδαφος που περιέχει τα ορυκτά συστατικά του κυρίου εδάφους.



Σχήμα 3.1: Παρουσίαση στιβάδων που αποτελούν ένα συνηθισμένο εδαφικό στρώμα.

Γενικά στο υπέδαφος βρίσκονται πολλών ειδών ρύποι, όπως τοξικά χημικά, ιοί, βακτήρια ακόμα και σε μερικές περιπτώσεις ραδιενεργά απόβλητα. Δυστυχώς κάτω από ορισμένες συνθήκες είναι δυνατό λεπτά σωματίδια, που ίσως να μην είναι επιβλαβή, να συνεισφέρουν στην μεταφορά και περεταίρω εξάπλωση των παραπάνω επικίνδυνων χημικών.

Αυτό που συμβαίνει είναι ότι συχνά οι ρύποι όταν βρεθούν μέσα στο υπέδαφος δεν είναι απαραίτητο να ταξιδέψουν μακριά από την αρχική τους θέση. Ιδιαίτερα κάτι τέτοιο ισχύει για ρύπους με μεγάλη ικανότητα προσρόφησης. Αφού τότε προσκολλώνται γρήγορα στο στερεό σκελετό του υδροφορέα και παραμένουν εκεί. Έτσι η εξάπλωση της μόλυνσης περιορίζεται. Όμως όπως προσροφώνται στον στερεό σκελετό έτσι δύναται να προσροφηθούν και σε άλλα σωματίδια (κολλοειδή) τα οποία κινούνται. Αυτό σημαίνει ότι πλέον ο ρύπος μπορεί να αποκτήσει ένα νέο μέσο μεταφοράς που πριν δεν είχε, και η ικανότητα του για εξάπλωση να αυξηθεί κατακόρυφα. Με αυτόν το μηχανισμό δημιουργείται το φαινόμενο της συμμεταφοράς.

Με τον όρο κολλοειδή ονομάζουμε σωματίδια που η διάμετρος τους κυμαίνεται από 10 μm-0,001 μm. Στο υπέδαφος υπάρχει μεγάλη ποικιλία κολλοειδών είτε αυτά είναι ανόργανης είτε οργανικής προέλευσης. Σε αυτά ανήκουν

I. Ιζήματα σιδήρου
II. Οξείδια μαγνησίου
III. Πετρώματα και ορυκτά θραύσματα
IV. Βιοκολλοειδή όπως βακτήρια, πρωτόζωα και ιοί
V. Μακρομόρια σαν μέρος φυσικής οργανικής μάζας.

Οι μεγάλες επιφάνειες που διαθέτουν συχνά τα κολλοειδή, επιτρέπει σε αυτά να προσροφούν άλλους ρύπους με σχετική ευκολία. Για παράδειγμα κολλοειδή με σημαντικές περιεκτικότητες σε οργανικό άνθρακα (π.χ. αργιλικά ορυκτά και οξείδια σιδήρου) είναι ισχυροί προσροφητές για ραδιενεργούς και άλλους τοξικούς ρύπους (Ouyang et al., 1996; Flury et al., 2002).

Έχουν γίνει πολλές εργασίες στην βιβλιογραφία που ερευνούν την μεταφορά ρύπων παρουσία κολλοειδών (Eichholz et al., 1982; McCarthy and Zachara, 1989) και έχουν καταλήξει στο ότι όντως τα κολλοειδή μπορούν να διευκολύνουν την μεταφορά ρυπογόνων ουσιών, ιδιαίτερα αυτών που έχουν μεγάλη ικανότητα για προσρόφηση. Έτσι σε μελέτη πεδίου ο kersting et al., (1999) ο οποίος εξέτασε ραδιονουκλεοτίδια σε υπόγειο νερό, 1.3km κάτω από το έδαφος σε υπόγειες κοιλότητες, βρήκε ότι το 90-99% των ραδιονουκλεοτιδίων ήταν συνδεδεμένα με κολλοειδή. Παρομοίως η έντονη μεταφορά ραδιονουκλεοτιδίων τόσο στην περιοχή Chalk River Nuclear laboratories στον Καναδά (Walton και Merritt, 1980) όσο και το απόθεμα ουρανίου στην Αυστραλία έχουν συσχετισθεί με την μεταφορά κολλοειδών. Άλλες έρευνες στο πεδίο έχουν δείξει το μεγάλο εύρος των ρύπων που επηρεάζονται δραστικά από τα κολλοειδή, π.χ. συμμεταφορά καδμίου παρουσία βακτηρίων σε χαλικώδες υδροφορέα (Pang et al., 2005), συμμεταφορά καισίου με αργίλιο και διοξείδιο του πυριτίου (NOELL et al, 1998; Chen et al, 2005), συμμεταφορά πλουτώνιου με ανόργανα κολλοειδή (Cvetkovic et al., 2004) και ουρανίου από χουμικά κολλοειδή (Artinger et al., 2002). Όλες οι παραπάνω περιπτώσεις συνηγορούν στην ύπαρξη συμμεταφοράς και στον σημαντικό ρόλο που αυτή διαδραματίζει στην υπόγεια ρύπανση.

Τέλος να σημειώσουμε ότι προαπαιτούμενες για το φαινόμενο της συμμεταφοράς είναι οι παρακάτω τρείς διεργασίες: (Ryan and Elimelech, 1996):

1. Παραγωγή λεπτών σωματιδίων (ύπαρξη πηγής κολλοειδών).
2. Προσρόφηση (ή απορρόφηση) του ρύπου προς μεταφορά, από τα διαθέσιμα κολλοειδή.
3. Η μεταφορά των παραπάνω κολλοειδών.

3.2 ΜΑΘΗΜΑΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΣΥΜΜΕΤΑΦΟΡΑΣ

Η μονοδιάστατη απλή μεταφορά κολλοειδούς ρύπου μέσα σε ομογενές πορώδες μέσο, λαμβάνοντας υπόψη προσρόφηση πρώτης τάξεως καθώς και αποδόμηση, δίνεται από την παρακάτω μερική διαφορική εξ. (3.1) (την αποδείξαμε στο κεφ. 2):

$$\frac{\partial C_i}{\partial t} + \frac{\rho_m}{\theta}\frac{\partial C_i^*}{\partial t} = D_i \frac{\partial^2 C_i}{\partial x^2} - U\frac{\partial C_i}{\partial x} - \lambda C_i - \lambda^* \frac{\rho_m}{\theta} C_i^* \qquad (3.1)$$

Για λόγους απλότητας και αμεσότητας, στην παρούσα μαθηματική ανάλυση θα θεωρήσουμε ότι το κολλοειδές μας είναι μια άργιλος και ο ρύπος που συμμεταφέρεται είναι ιός. Οπότε στην εξ. (3.1) ο δείκτης i φανερώνει το είδος της ουσίας και : α) για i=t, έχουμε μεταφορά tracer (ιχνηθέτης), β) για i=v, έχουμε μεταφορά virus (ιός), γ) για i=c, έχουμε μεταφορά clay (αργίλου). Ο όρος C_i συμβολίζει την συγκέντρωση του εκάστοτε ρύπου; Ο όρος C_i* αναφέρεται στην συγκέντρωση του προσροφημένου ρύπου επάνω στο στερεό πορώδες, U είναι η ενδοπορώδης ταχύτητα του ρευστού διατηρείται σταθερά, ρ_m είναι η πυκνότητα του στερεού πορώδους (solid matrix), λ είναι ο ρυθμός αποδόμησης του διαλυμένου ρύπου. Η αποδόμηση αυτή μπορεί να οφείλεται σε διάφορους παράγοντες για παράδειγμα απενεργοποίηση των ιών (οι ιοί κάποια στιγμή πεθαίνουν) ή διάσπαση ραδιενεργών πυρήνων (χρόνος ημιζωής). Ό όρος λ* αναφέρεται στην αποδόμηση των προσροφημένων διαλυτών επάνω στο στερεό σκελετό του υδροφορέα, ενώ θ είναι το πορώδες του (του στερεού υδροφορέα). Τέλος t είναι ο χρόνος και D_i είναι η υδροδυναμική διασπορά εξ. (3.2) (Bear, 1979).

$$D_i = \alpha_L U + \mathcal{D}_{ei} \qquad (3.2)$$

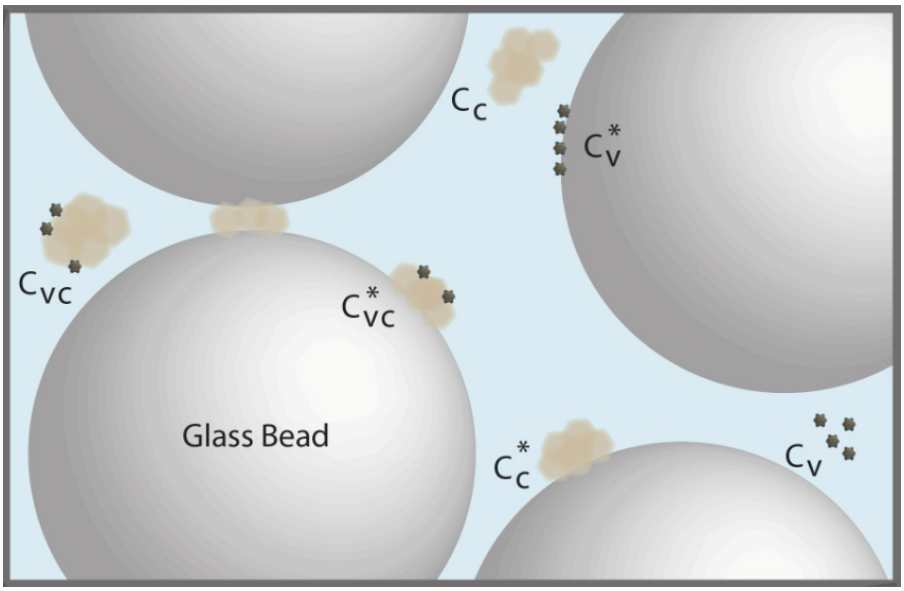
όπου αL είναι η διαμήκης τάση διασποράς και $D_{ei} = \frac{D_i}{\tau^*}$ είναι ο συντελεστής αποτελεσματικής μοριακή διάχυσης (είτε αυτός να αναφέρεται σε ιχνηθέτη, ιό ή άργιλο). Επιπλέον τ*≥1 είναι το δαιδαλώδες του πορώδους και D_i είναι η μοριακή διάχυση.

Ο ρυθμός με τον οποίο ο εκάστοτε διαλυμένος ρύπος προσροφάται επάνω στο στερεό πορώδες δίδεται από την γενικευμένη εξίσωση (3.3) (κινητική προσρόφηση):

$$\frac{\rho_m}{\theta}\frac{\partial C_i^*}{\partial t} = r_{i-i^*} C_i - \frac{\rho_m}{\theta} r_{i^*-i} C_i^* - \lambda^* \frac{\rho_m}{\theta} C_i^* \qquad (3.3)$$

Όπου r_{i-i*} είναι ο συντελεστής προσρόφησης, δηλαδή ο ρυθμός με τον οποίο διαλυμένος ρύπος προσροφάται και r_{i*-i} είναι ο ρυθμός με τον οποίο ήδη προσροφημένος ρύπος απελευθερώνεται και γίνεται ξανά διαλυμένος.

Ξεφεύγοντας από το πρότυπο της απλής μεταφοράς εξ. (3.1) και προσπαθώντας να περιγράψουμε την συμμεταφορά, το συνηθισμένο μοντέλο δύο φάσεων (διαλυτός ρύπος στο νερό ή προσροφημένος ρύπος επάνω στο στερεό πορώδες) παραχωρεί την θέση του σε αυτό των τριών φάσεων. Πλέον υπάρχει α) η διαλυτή φάση, που μπορεί να είναι είτε κάποιο κολλοειδές (άργιλος) είτε κάποιος ιός, β) προσροφημένος ιός επάνω σε κάποιο κολλοειδές, και γ) προσροφημένο κολλοειδές ή προσροφημένος ιός κατευθείαν επάνω στο στερεό σκελετό του πορώδους υδροφορέα ή προσροφημένος ιός επάνω σε κολλοειδές που στην συνέχεια ξανά προσροφάται επάνω στο στερεό σκελετό. Όλα τα παραπάνω περιγράφονται στο Σχήμα 3.1



Σχήμα 3.1: Απεικόνιση των έξι συστατικών του συστήματος συμμεταφοράς και κατανομή τους στις τρεις φάσεις. Οι όροι Cc, Cv, Cvc, Cc*, Cv* και Cvc* ορίζονται ως εξής: Cc είναι το αιωρούμενο κολλοειδές, Cv είναι ο αιωρούμενος ιός, Cvc είναι ο ιός που έχει προσροφηθεί επάνω στην επιφάνεια του κολλοειδούς, Cc* είναι το προσροφημένο κολλοειδές επάνω στο στερεό πορώδες (σφαίρες γυαλιού), Cv* είναι ο προσροφημένος ιός επάνω στο στερεό πορώδες (σφαίρες γυαλιού), Cvc* είναι ιός που έχει προσροφηθεί επάνω στην επιφάνεια του κολλοειδούς και στην συνέχεια ξανά προσροφάται επάνω στο στερεό πορώδες.

Παρατηρώντας το Σχήμα 3.1 αντιλαμβανόμαστε ότι ο συνολικός αριθμός των διαφορετικών αγνώστων που υπεισέρχονται στο πρόβλημα της συμμεταφοράς δύο φάσεων είναι έξι (Cc, Cv, Cvc, Cc*, Cv* και Cvc*). Άρα χρειαζόμαστε και έξι εξισώσεις για να το λύσουμε αποτελεσματικά.

Στην παρούσα προσέγγιση θα κάνουμε παραδοχή ότι τα κολλοειδή εξαιτίας του μεγέθους τους δεν επηρεάζονται από την μεταφορά των ιών (πολύ μικρότερα). Σαν αποτέλεσμα αυτού, η προσομοίωση της μεταφοράς τους θα γίνει σαν να μην υπήρχαν καθόλου ιοί. Οι εξ. (3.4, 3.5) περιγράφουν την μεταφορά και προσρόφηση των κολλοειδών αντίστοιχα. Έχουν προέλθει από τις εξ. (3.1, 3.2) θέτοντας όπου i=C:

$$\frac{\partial C_c}{\partial t} + \frac{\rho_m}{\theta}\frac{\partial C_c^*}{\partial t} = D_c \frac{\partial^2 C_c}{\partial x^2} - U\frac{\partial C_c}{\partial x} - \lambda C_c - \lambda^* \frac{\rho_m}{\theta} C_c^* \qquad (3.4)$$

$$\frac{\rho_m}{\theta}\frac{\partial C_c^*}{\partial t} = r_{c-c^*} C_c - \frac{\rho_m}{\theta} r_{c^*-c} C_c^* - \lambda^* \frac{\rho_m}{\theta} C_C^* \qquad (3.5)$$

Όπου C_c^* είναι η συγκέντρωση της προσροφημένης αργίλου, η σταθερά r_{c-c^*} δίνει το ρυθμό με τον οποίο η διαλυμένη άργιλος μετατρέπεται σε προσροφημένη, ενώ η σταθερά r_{c^*-c} δίνει το ρυθμό με τον οποίο η προσροφημένη άργιλος μετατρέπεται σε διαλυμένη. Τέλος ο όρος ρ_m είναι η πυκνότητα του στερεού πορώδους, θ είναι το πορώδες του στερεού μέσου και λ^* είναι η αποδόμηση της προσροφημένης αργίλου.

Πριν φτάσουμε στην τελική εξίσωση συμμεταφοράς για τους ιούς, σε ένα ενδιάμεσο βήμα θεωρούμε ότι ο ιός μεταφέρεται ανεξάρτητα από τα κολλοειδή. Έτσι χρησιμοποιούμε την εξ. (3.1) και θέτουμε όπου i=v, οπότε προκύπτει εξ. (3.6). Με την ίδια λογική το σύμπλεγμα C_{vc} (ιός προσροφημένος επάνω σε κολλοειδές) μεταφέρεται και αυτό ανεξάρτητα, οπότε με την χρήση της εξ. (3.1) (και θέτοντας όπου $i=C_{vc}$) εξάγεται η εξ. (3.7).

$$\frac{\partial C_v}{\partial t} + \frac{\rho_m}{\theta}\frac{\partial C_v^*}{\partial t} = D_v \frac{\partial^2 C_v}{\partial x^2} - U\frac{\partial C_V}{\partial x} - \lambda C_{vc} - \lambda^* \frac{\rho_m}{\theta} C_v^* \qquad (3.6)$$

$$\frac{\partial C_c C_{vc}}{\partial t} + \frac{\rho_m}{\theta}\frac{\partial C_c^* C_{vc}^*}{\partial t} = D_{vc} \frac{\partial^2 C_c C_{vc}}{\partial x^2} - U\frac{\partial C_c C_{vc}}{\partial x} - \lambda C_c C_{vc} - \lambda^* \frac{\rho_m}{\theta} C_c C_{vc}^* \qquad (3.7)$$

Για να φτάσουμε στην τελική εξίσωση συμμεταφοράς δεν έχουμε παρά να αθροίσουμε τις εξ. (3.6, 3.7) κατά μέλη. Οπότε προκύπτει η εξ. (3.8)

$$\frac{\partial}{\partial t}\left(C_v + \frac{\rho_m}{\theta} C_v^* + C_c C_{vc} + \frac{\rho_m}{\theta} C_c^* C_{vc}^* \right) = D_v \frac{\partial^2 C_v}{\partial x^2} + D_{vc} \frac{\partial^2}{\partial x^2}\left(C_c C_{vc}\right) - U\frac{\partial}{\partial x}\left(C_v + C_c C_{vc}\right)$$
$$- \lambda_v C_v - \lambda_{vc} C_c C_{vc} - \lambda_v^* \frac{\rho_m}{\theta} C_v^* - \lambda_{vc}^* \frac{\rho_m}{\theta} C_c^* C_{vc}^* \qquad (3.8)$$

Η εξ. (3.8) είναι η μονοδιάστατη εξίσωση συμμεταφοράς που περιγράφει την ταυτόχρονη μεταφορά αργίλου με ιό σε ομογενές πορώδες μέσο, παρουσία μεταγωγής, αποδόμησης και προσρόφησης (Abdel-Salam and Chrysikopoulos, 1995a, 1995b).

Στην εξ. (3.7) παρατηρείται ότι αντί για τον όρο C_{vc} έχει εμφανιστεί ο όρος $C_c C_{vc}$. Ο λόγος για τον οποίο συμβαίνει κάτι τέτοιο είναι οι μονάδες. Θέλουμε συμβατές μονάδες, όλοι οι διαλυτοί όροι να αναφέρονται σε ml ρευστού και οι προσροφημένοι να αναφέρονται σε μάζα στερεού σκελετού. Η ανάλυση των μονάδων φαίνεται παρακάτω.

$$C_c = \left[\frac{\text{mg αργίλου}}{\text{ml ρευστού}} \right] \tag{3.9}$$

$$C_{vc} = \left[\frac{\text{pfu ιού}}{\text{mg αργίλου}} \right] \tag{3.10}$$

$$C_c C_{vc} = \left[\frac{\text{mg αργίλου}}{\text{ml ρευστού}} \right] \left[\frac{\text{pfu ιού}}{\text{mg αργίλου}} \right] = \left[\frac{\text{pfu ιού}}{\text{ml ρευστού}} \right] \tag{3.11}$$

$$C_c^* = \left[\frac{\text{mg αργίλου}}{\text{mg στερεού σκελετού}} \right] \tag{3.12}$$

$$C_{vc}^* = \left[\frac{\text{pfu προσρόφ. ιού σε άργιλο}}{\text{mg αργίλου}} \right] \tag{3.13}$$

$$C_c^* C_{vc}^* = \left[\frac{\text{mg αργίλου}}{\text{mg στερεού σκελετού}} \right] \left[\frac{\text{pfu προσρόφ. ιού σε άργιλο}}{\text{mg αργίλου}} \right] \rightarrow$$
$$C_c^* C_{vc}^* = \left[\frac{\text{pfu προσρόφ. ιού σε άργιλο}}{\text{mg στερεού σκελετού}} \right] \tag{3.14}$$

Οπότε με το να πολλαπλασιάσουμε τους όρους C_{vc} και C_{vc}^* με τους C_c και C_c^* αντίστοιχα, καταφέραμε όλες οι μονάδες να αναφέρονται είτε σε ml ρευστού είτε σε μάζα στερεού σκελετού εξ. (3.9-3.14).

Οι όροι $\frac{\partial(C_c C_{vc})}{\partial t}$ και $\frac{\rho_m}{\theta}\frac{\partial(C_c^* C_{vc}^*)}{\partial t}$ που εμφανίζονται στην εξ. (3.8) δίνονται από την βιβλιογραφία (Bekhit et al., 2009) εξ. (3.15) και εξ. (3.16) αντίστοιχα :

$$\frac{\partial(C_c C_{vc})}{\partial t} = \Lambda_{v-vc} - \Lambda_{vc-v} + \Lambda_{v^*c^*-vc} - \Lambda_{vc-v^*c^*} - \lambda_{vc} C_c C_{vc} - \lambda_{vc}^* \frac{\rho_m}{\theta} C_c^* C_{vc}^* \quad (3.15)$$

$$\frac{\rho_m}{\theta}\frac{\partial(C_c^* C_{vc}^*)}{\partial t} = \Lambda_{v-v^*c^*} - \Lambda_{v^*c^*-v} + \Lambda_{vc-v^*c^*} - \Lambda_{v^*c^*-vc} - \lambda_{vc}^* \frac{\rho_m}{\theta} C_c^* C_{vc}^* \quad (3.16)$$

όπου Λ_{v-vc} [PFU /(cm^3 min)] είναι ο ρυθμός συσσώρευσης μάζας εξαιτίας της προσρόφησης διαλυμένων ιών πάνω σε διαλυμένη άργιλο, ο οποίος εκφράζεται από την ακόλουθη μη γραμμική εξίσωση εξ. (3.17):

$$\Lambda_{v-vc} = r_{v-vc}\left(C_{vc_{eq}} - C_{vc}\right)^2 C_c \quad (3.17)$$

με r_{v-vc} [mg clay /(PFU min)] να είναι η σταθερά προσρόφησης διαλυμένων ιών πάνω σε διαλυμένη άργιλο και $C_{vc_{eq}}$ (PFU/mg clay) είναι η συγκέντρωση των προσροφημένων ιών επάνω σε διαλυμένη άργιλο στην κατάσταση ισορροπίας. Συνεχίζοντας την περιγραφή των όρων της εξ. (3.15), Λ_{vc-v} [PFU /(cm^3 min)] είναι η συσσώρευση μάζας εξαιτίας της αποκόλλησης των προσροφημένων ιών επάνω από την διαλυμένη άργιλο, εκφρασμένη με την παρακάτω γραμμική σχέση εξ. (3.18).

$$\Lambda_{vc-v} = r_{vc-v}\left(C_c C_{vc}\right) \quad (3.18)$$

όπου r_{vc-v} (1/min) είναι η σταθερά ρυθμού αποκόλλησης των προσροφημένων ιών επάνω από την διαλυμένη άργιλο, $\Lambda_{vc-v^*c^*}$ [PFU /(cm^3 min)] είναι ο ρυθμός συσσώρευσης μάζας εξαιτίας της προσρόφησης διαλυμένων ιών πάνω σε διαλυμένη άργιλο και όλο το σύμπλεγμα αυτό ξανά προσροφάτε στο στερεό σκελετό (σφαίρες γυαλιού). Ο ρυθμός αυτός εκφράζεται από την ακόλουθη γραμμική σχέση εξ. (3.19).

$$\Lambda_{vc-v^*c^*} = r_{vc-v^*c^*}\left(C_c C_{vc}\right) \tag{3.19}$$

με $r_{vc-v^*c^*}$ (1/min) να είναι η σταθερά προσρόφησης του συμπλέγματος ιού-αργίλου επάνω το στερεό πορώδες, $\Lambda_{v^*c^*-vc}$ [PFU /(cm^3 min)] είναι ο ρυθμός συσσώρευσης μάζας εξαιτίας της αποκόλλησης του συμπλέγματος ιού-αργίλου από το στερεό πορώδες και η ξανά μετατροπή του σε διαλυτή μορφή. Ο ρυθμός αυτός εκφράζεται από την ακόλουθη γραμμική σχέση εξ. (3.20).

$$\Lambda_{v^*c^*-vc} = \frac{\rho_m}{\theta} r_{v^*c^*-vc}\left(C_c^* C_{vc}^*\right) \tag{3.20}$$

όπου $r_{v^*c^*-vc}$ (1/min) να είναι η σταθερά συσσώρευσης μάζας εξαιτίας της αποκόλλησης του συμπλέγματος ιού-αργίλου από το στερεό πορώδες και η ξανά μετατροπή του σε διαλυτή μορφή.

Συνεχίζοντας με την ανάλυση των όρων της εξ. (3.16), $\Lambda_{v-v^*c^*}$ [PFU /(cm^3 min)] είναι ο ρυθμός συσσώρευσης μάζας εξαιτίας της μετατροπής διαλυμένου ιού κατευθείαν σε σύμπλεγμα ιού-αργίλου (σύμπλεγμα= ιός προσροφημένος επάνω σε άργιλο) προσροφημένου επάνω στο στερεό πορώδες. Ο ρυθμός αυτός εκφράζεται από την ακόλουθη μη γραμμική σχέση εξ. (3.21).

$$\Lambda_{v-v^*c^*} = \frac{\rho_m}{\theta} r_{v-v^*c^*} \left(C^*_{vc_{eq}} - C^*_{vc} \right)^2 C^*_c \qquad (3.21)$$

όπου $r_{v-v^*c^*}$ [mg clay /(PFU min)] είναι η σταθερά συσσώρευσης μάζας εξαιτίας της μετατροπής διαλυμένου ιού κατευθείαν σε σύμπλεγμα ιού-αργίλου προσροφημένου επάνω στο στερεό πορώδες και $C^*_{vc_{eq}}$ (PFU/mg clay) είναι η συγκέντρωση του προσροφημένου συμπλέγματος ιού-αργίλου επάνω στο στερεό πορώδες, στην κατάσταση ισορροπίας.

Ακόμα $\Lambda_{v^*c^*-v}$ [PFU /(cm^3 min)] είναι ο ρυθμός συσσώρευσης μάζας εξαιτίας της μετατροπής προσροφημένου συμπλέγματος ιού-αργίλου (επάνω στο στερεό πορώδες) κατευθείαν σε διαλυμένο ιό. Ο ρυθμός αυτός εκφράζεται από την ακόλουθη μη γραμμική σχέση εξ. (3.22).

$$\Lambda_{v^*c^*-v} = \frac{\rho_m}{\theta} r_{v^*c^*-v} \left(C^*_c C^*_{vc} \right) \qquad (3.22)$$

όπου τέλος $r_{v^*c^*-v}$ (1/min) είναι η σταθερά που περιγράφει την μετατροπή προσροφημένου συμπλέγματος ιού-αργίλου (επάνω στο στερεό πορώδες) κατευθείαν σε διαλυμένο ιό.

Στο Σχήμα 3.1 είδαμε τους 6 διαφορετικούς όρους της συμμεταφοράς 3 φάσεων. Για να επιλυθούν μονοσήμαντα οι άγνωστοι αυτοί {Cc, Cv, Cvc, Cc*, Cv* και Cvc*} χρειαζόμαστε και 6 ανεξάρτητες εξισώσεις. Μέχρι τώρα έχουμε αναλύσει τις πρώτες πέντε εξ. 3.4, 3.5, 3.8, 3.15 και 3.16. Η τελευταία που μας λείπει είναι η εξίσωση προσρόφησης ιών επάνω στο στερεό πορώδες και δίνεται από την εξ. (3.23)

$$\frac{\rho_m}{\theta} \frac{\partial C^*_c}{\partial t} = r_{v-v^*} C_v - \frac{\rho_m}{\theta} r_{v^*-v} C^*_v - \lambda^* \frac{\rho_m}{\theta} C^*_v \qquad (3.23)$$

Όπου η σταθερά r_{v-v^*} δίνει το ρυθμό με τον οποίο ο διαλυμένος ιός μετατρέπεται σε προσροφημένο, ενώ η σταθερά r_{v^*-v} δίνει το ρυθμό με τον οποίο ο προσροφημένος ιός μετατρέπεται σε διαλυμένο.

3.3 ΠΕΡΙΛΗΨΗ ΤΩΝ ΕΞΙΣΩΣΕΩΝ ΣΥΜΜΕΤΑΦΟΡΑΣ

Συνοψίζοντας τις εξισώσεις συμμεταφοράς έχουμε :

1. Μονοδιάστατη εξίσωση μεταφοράς αργίλου, εξ. (3.4).
2. Εξίσωση προσρόφησης αργίλου με αποδόμηση, εξ. (3.5).
3. Εξίσωση που δίνει ρυθμό συσσώρευσης συμπλέγματος ιού-αργίλου σε διαλύτη μορφή $\frac{\partial(C_c C_{vc})}{\partial t}$, εξ. (3.15).
4. Εξίσωση που δίνει ρυθμό συσσώρευσης συμπλέγματος ιού-αργίλου προσροφημένου στο στερεό πορώδες $\frac{\partial(C_c^* C_{vc}^*)}{\partial t}$, εξ. (3.16).
5. Μονοδιάστατη εξίσωση συμμεταφοράς που περιγράφει την ταυτόχρονη μεταφορά αργίλου με ιό σε ομογενές πορώδες μέσο, παρουσία μεταγωγής, αποδόμησης και προσρόφησης, εξ.(3.8).
6. Εξίσωση προσρόφησης ιού με αποδόμηση, εξ. (3.23).

Συνεπώς έχουμε ένα σύστημα με 6 εξισώσεις και 6 αγνώστους το οποίο και λύνεται μονοσήμαντα . Η πιο συνετή και αποτελεσματική σειρά επίλυσης των παραπάνω σχέσεων, που βασίζεται στην διαδοχική επίλυση τριών 2x2 συστημάτων αντί ενός 6x6, είναι η εξής:

1. Επίλυση των εξισώσεων μεταφοράς κολλοειδών με προσρόφηση που δίνουν τους όρους Cc, Cc* εξ. (3.4, 3.5) (είναι ανεξάρτητες από οποιαδήποτε άλλη μεταφορά).

2. Λύση των εξισώσεων που δίνουν τους όρους CcCvc και Cc*Cvc* εξ. (3.15, 3.16) (βασίζονται στην μεταφορά κολλοειδών)

3. Επίλυση εξισώσεων συμμεταφοράς ιών με προσρόφηση και υπολογισμός όρων Cv, Cv* εξ. (3.8, 3.23) (βασίζονται στους όρους Cc, Cc*, CcCvc και Cc*Cvc*)

4. ΜΕΘΟΔΟΙ ΕΠΙΛΥΣΗΣ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

4.1 ΑΝΑΦΟΡΑ ΣΕ ΜΕΘΟΔΟΥΣ ΕΠΙΛΥΣΗΣ ΑΠΟ ΒΙΒΛΙΟΓΡΑΦΙΑ

Οι πιο γνωστοί μέθοδοι επίλυσης διαφορικών εξισώσεων με μερικές παραγώγους (Zheng and Bennett, 1995) παρουσιάζονται παρακάτω:

A. Αναλυτικοί μέθοδοι

   I. Laplace transformations
   II. Fourier transformations

B. EULERIAN METHODS

   I. Πεπερασμένες διαφορές
   II. Πεπερασμένα στοιχεία

- ✓ Παρέχουν "Σταθερό πλέγμα"-"fixed spatial grid"
- ✓ Λύνουν αποδοτικά και με ακρίβεια προβλήματα διασποράς
- Υπόκεινται σε τεχνητές διακυμάνσεις όταν λύνουν προβλήματα μεταγωγής (artificial oscillations)

C. LAGRANIAN METHODS
   I. Random Walk

- ✓ Παραμορφωμένο πλέγμα
- ✓ Λύνουν προβλήματα μεταγωγής με μεγάλες αιχμές συγκέντρωσης
- Προκαλούν αριθμητική διασπορά (numerical dispersion)

D. Mixed Eulerian – Lagranian methods

Στην πρώτη κατηγορία επίλυσης ανήκουν οι αναλυτικές μέθοδοι οι οποίοι εφαρμόζονται σε πολλά συνηθισμένα προβλήματα μεταφοράς ρύπων όπου με μετασχηματισμούς Laplace ή Fourier μπορούμε και μειώνουμε των αριθμό των άγνωστων μεταβλητών. Βασική διαφορά των δύο μεθόδων είναι ότι κατά την διάρκεια του μετασχηματισμού Laplace, η ολοκλήρωση που συμβαίνει έχει άκρα το μηδέν και το θετικό άπειρο. Αντίθετα με την μέθοδο Fourier η ολοκλήρωση έχει όρια το αρνητικό και το θετικό άπειρο. Πρακτικά αυτό σημαίνει ότι η μεταβλητή για την οποία γίνεται η ολοκλήρωση θα πρέπει να μπορεί και να πάρει τιμές σε όλο το

διάστημα που αναφέρεται η εκάστοτε μέθοδος αλλά και να έχει νόημα σε αυτό. Έτσι για την μεταβλητή του χρόνου δεν γίνεται να χρησιμοποιηθεί η μέθοδος Fourier αφού δεν υφίσταται αρνητικός χρόνος. Ενώ αντίθετα με άνεση θα μπορούσαμε να χρησιμοποιήσουμε τον μετασχηματισμό Laplace.

Στην δεύτερη κατηγορία επίλυσης ανήκουν οι αριθμητικές μέθοδοι. Κυριότεροι αντιπρόσωποι τους είναι οι λύσεις με βάση τον Euler και οι λύσεις με βάση τον Lagrance.

Στις πρώτες η επίλυση γίνεται σε σταθερό πλέγμα κάτι που επιτρέπει με ακρίβεια την επίλυση προβλημάτων όπου το κύριο χαρακτηριστικό τους είναι η διασπορά. Εξαιτίας αυτού όμως δημιουργούνται τεχνητές διακυμάνσεις όταν λύνονται προβλήματα οπου η μεταγωγή είναι το βασικό μέσο μεταφοράς. Οι πιο γνωστοί μέθοδοι με βάση τον Euler είναι οι "Πεπερασμένες διαφορές" και τα " Πεπερασμένα στοιχεία " (Zheng and Bennett, 1995).

Στις λύσεις με βάση τον Lagrance η επίλυση γίνεται σε μη σταθερό πλέγμα (deforming grid) ή σε μη σταθερές συντεταγμένες (deforming coordinates). Το αποτέλεσμα αυτού είναι η ακριβής επίλυση προβλημάτων όπου η μεταφορά βασίζεται κυρίως σε μεταγωγή, αλλά και η εμφάνιση αριθμητικής διασποράς στα προβλήματα με βάση την διάχυση. Ιδιαίτερα πολλές δυσκολίες συναντώνται όταν η εφαρμογή της μεθόδου γίνεται σε ανομοιόμορφα μοντέλα (πολλές πηγές και πολλά πηγάδια). Κυριότερος αντιπρόσωπος αυτής της κατηγορίας είναι η μέθοδος "Random Walk" (Zheng and Bennett 1995).

Τέλος υπάρχουν μέθοδοι οι οποίοι βασίζονται ταυτόχρονα και στον Euler αλλά και στον Lagrance. Όπου δηλαδή λύνεται ο όρος της μεταγωγής με τον Lagrance και ο όρος της διάχυσης με τον Euler. Αποτέλεσμα αυτού είναι ο συνδυασμός των θετικών των δυο μεθόδων. Δηλαδή η γενική αύξηση της ακρίβεια των αποτελεσμάτων χωρίς να υπάρχουν περιορισμοί στην εφαρμογή της. Δυστυχώς υπολογιστικά υπάρχουν πολλές δυσκολίες ως προς την υλοποίηση της όλης μεθόδου (Zheng and Bennett 1995) και συναντάται σπάνια στην βιβλιογραφία.

4.2 ΑΝΑΛΥΤΙΚΕΣ ΛΥΣΕΙΣ ΓΙΑ ΤΗΝ ΑΠΛΗ ΕΞΙΣΩΣΗ ΜΕΤΑΦΟΡΑΣ ΣΤΙΣ ΤΡΕΙΣ ΔΙΑΣΤΑΣΕΙΣ

Με το πέρασμα του χρόνου και την εξέλιξη της τεχνολογίας, η ισχύς των ηλεκτρονικών υπολογιστών αυξάνεται ραγδαία. Αποτέλεσμα αυτού είναι να γίνεται πιο δόκιμη η χρήση αριθμητικών μεθόδων για την επίλυση διαφορικών εξισώσεων. Ένα μεγάλο μέρος του επιστημονικού ενδιαφέροντος μετατοπίζεται όλο και περισσότερο στις αριθμητικές λύσεις, κατανοώντας τις επιπλέον επιλογές αλλά και την μεγάλη ευελιξία που προσφέρουν, σε αντίθεση πάντα με τις παραδοσιακές αναλυτικές λύσεις. Παρ' όλα αυτά κανείς θα πρέπει να είναι σε θέση να ελέγχει τα αποτελέσματα που λαμβάνει από τις αριθμητικές επιλύσεις καθώς είναι πολλές οι ευκαιρίες για λάθη και ανακρίβειες. Με αυτή την λογική συνηθίζεται να γίνεται πρώτα η επίλυση του προβλήματος με μια αναλυτική μέθοδο, στην συνέχεια να εφαρμόζονται οι αριθμητικές μέθοδοι και στο τέλος να ελέγχονται-συγκρίνονται τα αποτελέσματα. Σύμφωνα με αυτήν την λογική αναζητήθηκαν στην βιβλιογραφία αναλυτικές λύσεις για το πρόβλημα της μεταφοράς (με διαφορετικές συνοριακές συνθήκες) οι οποίες παρουσιάζονται παρακάτω.

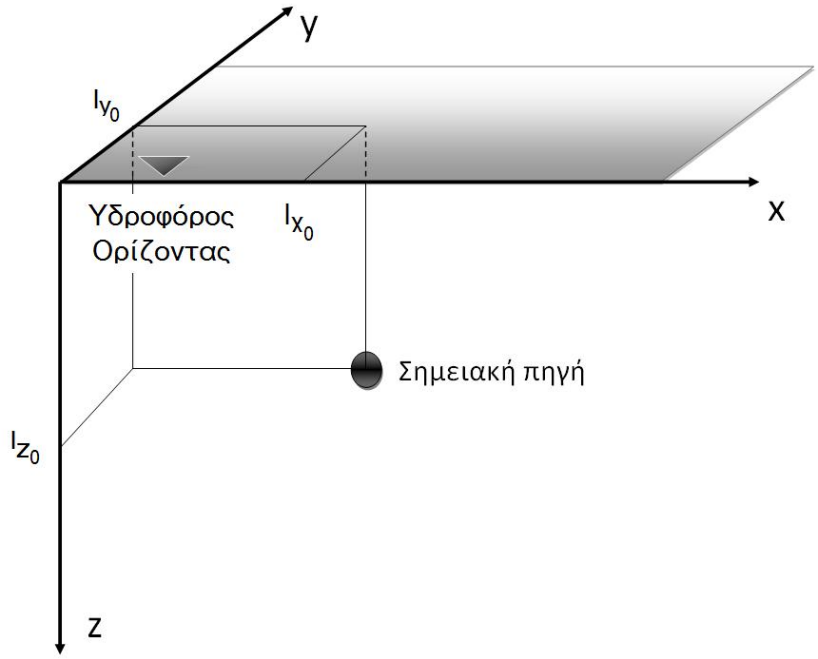
I. 1η Περίπτωση ημιάπειρος υδροφορέας με σημειακή πηγή.

II. 2η Περίπτωση περιορισμένος υδροφορέας με σημειακή πηγή.

III. 3η Περίπτωση ημιάπειρος υδροφορέας με ελλειπτική πηγή.

IV. 4η Περίπτωση περιορισμένος υδροφορέας με ελλειπτική πηγή.

1η Περίπτωση ημιάπειρος υδροφορέας με σημειακή πηγή:

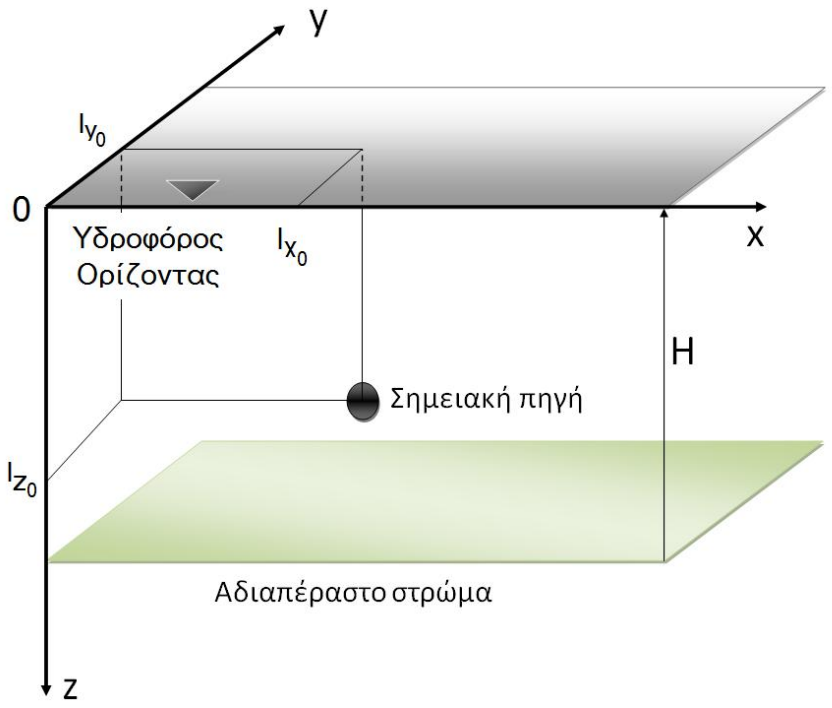


Σχήμα 4.1:Παρουσιάζεται υδροφόρος ορίζοντας με άπειρες διαστάσεις κατά την οριζόντια διεύθυνση. Στο πάνω μέρος του βρίσκεται η διεπιφάνεια νερού-αέρα η οποία δεν επιτρέπει την μεταφορά των ρύπων. Ενώ στο κάτω μέρος του δεν γνωρίζει περιορισμούς και επεκτείνεται στο άπειρο. Τέλος στο σημείο με συντεταγμένες l_{x0}, l_{y0}, l_{z0} υπάρχει σημειακή πηγή.

Χρησιμοποιώντας τις συνοριακές συνθήκες εξ. (2.21)-(2.25), την εξίσωση προσρόφησης με αποδόμηση (εξ. 1.21), την εξίσωση (εξ. 2.15) η οποία περιγράφει το είδος της πηγής και εφαρμόζοντας τις, στην εξίσωση (2.13) μεταφοράς ρύπων στις τρείς διαστάσεις παίρνουμε:

$$C(t,x,y,z) = \left(\frac{1}{64\pi^3 D_x D_y D_z}\right)^{\frac{1}{2}} \int_0^t \frac{G(t-\tau)}{\theta} \Lambda_1(\tau) \times \left\{ \int_0^\tau \frac{\Lambda_2(\tau)}{\zeta^{3/2}} \Lambda_3(\zeta, x-l_{x_0}, y-l_{y_0}) \times \right.$$
$$\left[\Lambda_4(\zeta, z+l_{z_0}) + \Lambda_4(\zeta, z-l_{z_0})\right] d\zeta + \frac{\Lambda_3(\tau, x+l_{x_0}, y-l_{y_0})}{\tau^{3/2}} \times \left[\Lambda_4(\tau, z+l_{z_0}) + \right.$$
$$\left.\left. \Lambda_4(\tau, z-l_{z_0})\right]\right\} d\tau \qquad (4.1)$$

2η Περίπτωση περιορισμένος υδροφορέας με σημειακή πηγή:

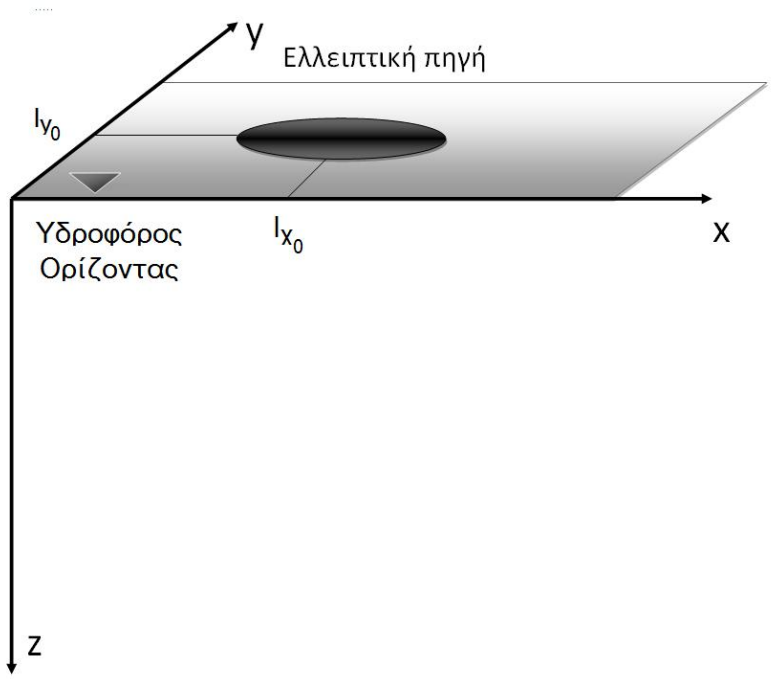


Σχήμα 4.2:Παρουσιάζεται υδροφόρος ορίζοντας με άπειρες διαστάσεις κατά την οριζόντια διεύθυνση. Στο πάνω μέρος του βρίσκεται η διεπιφάνεια νερού-αέρα η οποία δεν επιτρέπει την μεταφορά των ρύπων. Ενώ στο κάτω μέρος του εντοπίζεται αδιαπέρατο στρώμα το οποίο δρα όπως και η διεπιφάνεια νερού αέρα. Τέλος στο σημείο με συντεταγμένες l_{x0}, l_{y0}, l_{z0} υπάρχει σημειακή πηγή.

Χρησιμοποιώντας τις συνοριακές συνθήκες εξ. (2.21)-(2.24) και (2.26), την εξίσωση προσρόφησης με αποδόμηση (εξ. 1.21), την εξίσωση (εξ . 2.15) η οποία περιγράφει το είδος της πηγής και εφαρμόζοντας τις, στην εξ.(2.13) εξίσωση μεταφοράς ρύπων στις τρείς διαστάσεις παίρνουμε:

$$C(t,x,y,z) = \left(\frac{1}{16\pi^2 D_x D_y}\right)^{\frac{1}{2}} \int_0^t \frac{G(t-\tau)}{\theta} \Lambda_1(\tau) \left\{ \int_0^\tau \frac{\Lambda_2(\tau)}{\zeta} \Lambda_3\left(\zeta, x - l_{x0}, y - l_{y0}\right) \Lambda_6(\zeta, 1, \cos(\psi_m l_{z0}))\, d\zeta + \frac{\Lambda_3(\tau, x-l_{x0}, y-l_{y0})}{\tau} \Lambda_6(\tau, 1, \cos(\psi_m l_{z0})) \right\} d\tau \quad (4.2)$$

3η Περίπτωση ημιάπειρος υδροφορέας με ελλειπτική πηγή

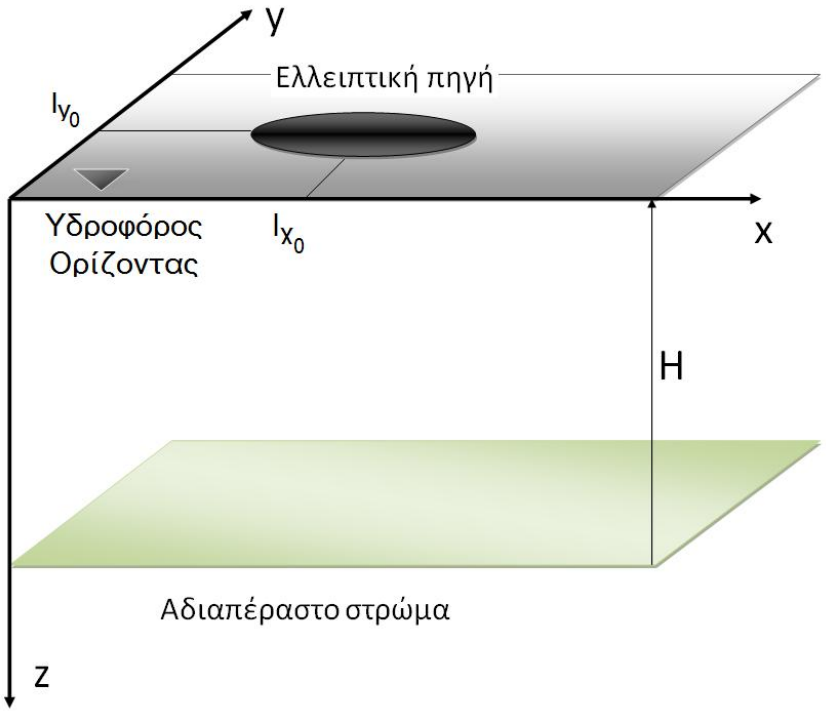


Σχήμα 4.3: Παρουσιάζεται υδροφόρος ορίζοντας με άπειρες διαστάσεις κατά την οριζόντια διεύθυνση. Στο πάνω μέρος του βρίσκεται η διεπιφάνεια νερού-αέρα η οποία δεν επιτρέπει την μεταφορά των ρύπων. Ενώ στο κάτω μέρος του δεν γνωρίζει περιορισμούς και επεκτείνεται στο άπειρο. Τέλος στο σημείο με συντεταγμένες l_{x0}, l_{y0}, l_{z0} υπάρχει ελλειπτική πηγή.

Χρησιμοποιώντας τις συνοριακές συνθήκες εξ. (2.21)-(2.25), την εξίσωση προσρόφησης με αποδόμηση (εξ. 1.21), τις εξισώσεις (εξ. 2.16)-(2.17) οι οποίες περιγράφουν το είδος της πηγής και εφαρμόζοντας τις, στην εξ. (2.13) εξίσωση μεταφοράς ρύπων στις τρείς διαστάσεις παίρνουμε:

$$C(t,x,y,z) = \left(\frac{1}{64\pi^2 D_x D_z}\right)^{\frac{1}{2}} \int_0^t \int_{\alpha_1}^{\alpha_2} \frac{G(t-\tau)}{\theta} \Lambda_1(\tau) \times \left\{ \int_0^{\tau} \frac{\Lambda_2(\tau)}{\zeta} \Lambda_3(\zeta, \mathrm{x} - \mathrm{q}, 0) \left[\Lambda_4\left(\zeta, \mathrm{z} + \mathrm{l}_{z_0}\right) + \Lambda_4\left(\zeta, \mathrm{z} - \mathrm{l}_{z_0}\right)\right] \Lambda_5(\zeta) \mathrm{d}\zeta + \frac{\Lambda_3(\tau, \mathrm{x}-\mathrm{q}, 0)}{\tau} \times \left[\Lambda_4\left(\tau, \mathrm{z} + \mathrm{l}_{z_0}\right) + \Lambda_4\left(\tau, \mathrm{z} - \mathrm{l}_{z_0}\right)\right] \Lambda_5(\tau) \right\} dq\, d\tau \quad (4.3)$$

4η Περίπτωση περιορισμένος υδροφορέας με ελλειπτική πηγή



Σχήμα 4.4: Παρουσιάζεται υδροφόρος ορίζοντας με άπειρες διαστάσεις κατά την οριζόντια διεύθυνση. Στο πάνω μέρος του βρίσκεται η διεπιφάνεια νερού-αέρα η οποία δεν επιτρέπει την μεταφορά των ρύπων. Ενώ στο κάτω μέρος του εντοπίζεται αδιαπέρατο στρώμα το οποίο δρα όπως και η διεπιφάνεια νερού αέρα. Τέλος στο σημείο με συντεταγμένες l_{x0}, l_{y0}, l_{z0} υπάρχει ελλειπτική πηγή.

Χρησιμοποιώντας τις συνοριακές συνθήκες εξ.(2.21)-(2.24) και (2.26), την εξίσωση προσρόφησης με αποδόμηση (εξ. 1.21), τις εξισώσεις (2.16)-(2.17) οι οποίες περιγράφουν το είδος της πηγής και εφαρμόζοντας τις, στην εξ. (2.13) εξίσωση μεταφοράς ρύπων στις τρείς διαστάσεις παίρνουμε:

$$C(t,x,y,z) = \left(\frac{1}{16\pi D_x}\right)^{\frac{1}{2}} \int_0^t \int_{\alpha_1}^{\alpha_2} \frac{G(t-\tau)}{\theta} \Lambda_1(\tau) \{\times \left\{ \int_0^{\tau} \frac{\Lambda_2(\tau)}{\zeta^{1/2}} \Lambda_3(\zeta, x-q, 0) \Lambda_5(\zeta) \times \Lambda_6(\zeta, 1, \cos(\psi_m l_{z0})) d\zeta + \frac{\Lambda_3(\tau, x-q, 0)}{\tau^{1/2}} \times \Lambda_5(\tau) \Lambda_6(\tau, 1, \cos(\psi_m l_{z0})) \} \right\} dq\, d\tau \quad (4.4)$$

Όπου για τις εξισώσεις 4.1-4.4 ισχύει ότι

$$\Lambda_1(\mathrm{t}) = exp[-H^* t] \quad (4.5)$$

$$\Lambda_2(t) = \left(\frac{B\zeta}{t-\zeta}\right)^{\frac{1}{2}} I_1\left[2\left(B\zeta(t-\zeta)\right)^{\frac{1}{2}}\right] \quad (4.6)$$

$$\Lambda_3(t,x,y) = \exp\left[\frac{U_x}{2D_x} - \frac{1}{4t}\left(\frac{x^2}{D_x} + \frac{y^2}{D_y}\right) - t\left(A - H^* + \frac{U^2}{4D_x}\right)\right] \quad (4.7)$$

$$\Lambda_4(t,z) = \exp\left[\frac{-z^2}{4D_z t}\right] \quad (4.8)$$

$$\Lambda_5(t) = \mathrm{erf}[\kappa_1(t,q,y)] - erf[\kappa_2(t,q,y)] \quad (4.9)$$

$$\alpha_1 = l_{x_0} - \alpha \quad (4.10)$$

$$\alpha_2 = l_{x_0} + \alpha \quad (4.11)$$

$$\kappa_1(t,q,y) = \left\{y - l_{y_0} + \left[b^2 - \frac{b^2(q-l_{x_0})^2}{\alpha^2}\right]^{\frac{1}{2}}\right\}\left(\frac{1}{4D_y t}\right)^{\frac{1}{2}} \quad (4.12)$$

$$\kappa_2(t,q,y) = \left\{y - l_{y_0} - \left[b^2 - \frac{b^2(q-l_{x_0})^2}{\alpha^2}\right]^{\frac{1}{2}}\right\}\left(\frac{1}{4D_y t}\right)^{\frac{1}{2}} \quad (4.13)$$

$$\Lambda_6(t,f1,f2) = \frac{f1}{H} + \frac{2}{H}\sum_{m=1}^{\infty} f_2 \exp[-\psi_m^2 D_z t]\, cos\,(\psi_m Z) \quad (4.14)$$

$$\psi_m = \frac{m\pi}{H} \quad (4.15)$$

$$A = r_1 + \lambda \quad (4.16)$$

$$B = r_1 r_2 \quad (4.17)$$

$$H^* = r_2 + \lambda^* \quad (4.18)$$

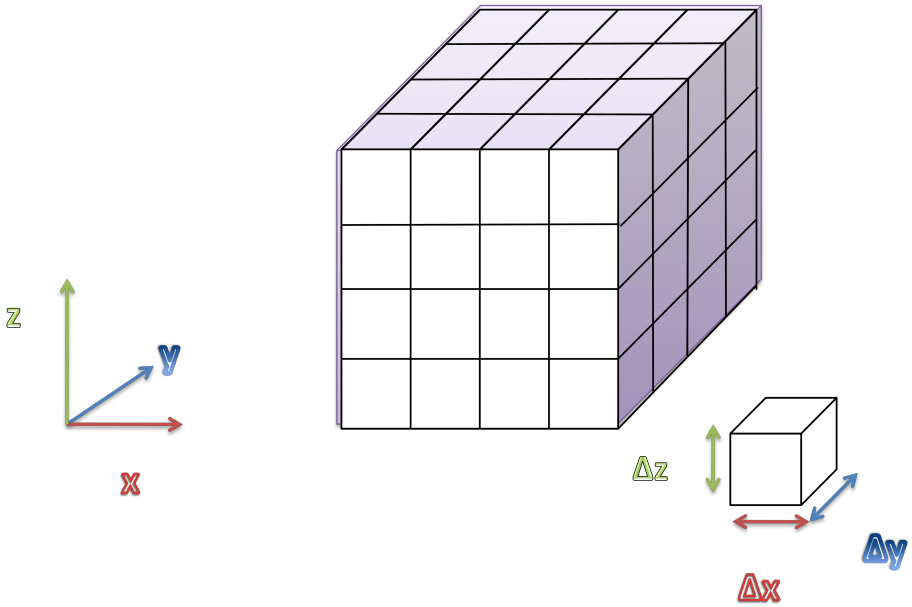
Για τα παραπάνω σύμβολα ισχύει ότι: α είναι ο ημιάξονας της ελλειπτικής πηγής παράλληλος στον x άξονα και b είναι ο ημιάξονας της ελλειπτικής πηγής παράλληλος στον y άξονα, C είναι η συγκέντρωση της διαλυμένης ουσίας [M L^{-3}], C_0 είναι η συγκέντρωση της πηγής, C^* είναι η συγκέντρωση του προσροφημένου ρύπου [MM^{-1}], D_x είναι η διαμήκης υδροδυναμική διασπορά [L^2t^{-1}], D_y είναι η πλευρική υδροδυναμική διασπορά [L^2t^{-1}], D_Z είναι η κατακόρυφη υδροδυναμική διασπορά [L^2t^{-1}], erf(x) είναι η συνάρτηση σφάλματος, f_1, f_2 είναι αυθαίρετες συναρτήσεις, F είναι η λειτουργική μορφή του είδους της πηγής (ορίζεται στις εξισώσεις 2.14-2.17), H είναι το πάχος του περιορισμένου υδροφορέα (Σχήμα 4.2, 4.4), I_1[ ] είναι η τροποποιημένη συνάρτηση του Bessel πρώτου είδους, πρώτης τάξεως, r_1, r_2 είναι τα K_1 και K_2 [t^{-1}] αντίστοιχα της εξίσωσης προσρόφησης με αποδόμηση (εξ. 1.21), t είναι η μεταβλητή του χρόνου, U είναι η μέση ενδοπορώδης ταχύτητα [Lt^{-1}], x,y,z είναι η χωρικές συντεταγμένες του σημείου στο οποίο ενδιαφερόμαστε να βρούμε την συγκέντρωση, Θ το πορώδες του υδροφορέα που μελετάμε, λ και λ^*[t^{-1}] είναι ο ρυθμός αποδόμησης διαλυμένης και προσροφημένης ουσίας αντίστοιχα, ρ είναι η πυκνότητα του σκελετού του πορώδους που μελετάμε [M L^{-3}].

4.3 ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ

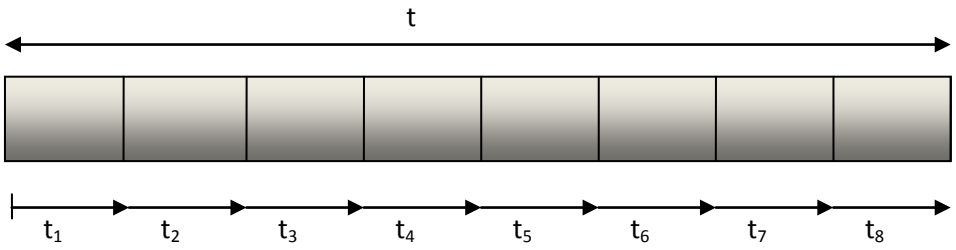
4.3.1 ΜΕΘΟΔΟΣ ΠΕΠΕΡΑΣΜΕΝΩΝ ΔΙΑΦΟΡΩΝ

Βασική ιδέα όλων των αριθμητικών μεθόδων είναι η αντικατάσταση μαθηματικών αναπαραστάσεων, οι οποίες είναι αδύνατο να υπολογιστούν με απόλυτη ακρίβεια (ολοκληρώματα η όρια που τείνουν στο 0), με άλλες που μπορούν να υπολογιστούν προσεγγιστικά, εντός επιτρεπτών ορίων πάντα ώστε να πλησιάζουν την πραγματικότητα. Μέσα στα πλαίσια αυτά ο χώρος Σχήμα 4.5 και ο χρόνος Σχήμα 4.6 χωρίζονται σε τμήματα (τεμάχια) στα οποία θεωρούμε ότι οι συνθήκες είναι σταθερές. Έτσι για τα τεμάχια του χώρου δεχόμαστε ότι η συγκέντρωση είναι σταθερή σε όλη την έκταση τους και αντιπροσωπεύεται από την συγκέντρωση στο κέντρου τους (block-centered discretization). Ενώ για τα τμήματα του χρόνου θεωρούμε οι αλλαγές της συγκέντρωσης στον χώρο γίνονται απότομα στο τέλος κάθε χρονικού βήματος.

Για την αριθμητική επίλυση των εξισώσεων μεταφοράς και συμμεταφοράς στις τρείς διαστάσεις επιλέχθηκαν "Οι Πεπερασμένες διαφορές". Οι συγκεκριμένες ανήκουν στις γενικότερες μεθόδους του Euler και συνδυάζουν αφενός εύκολη υλοποίηση (σχετικά γρήγορος ο προγραμματισμός τους), αλλά και αφετέρου προσομοιώνουν με ευκολία ανομοιόμορφα μοντέλα με πολλές πηγές και πηγάδια. Είναι δυνατό δε όταν ζητηθεί αύξηση της ακρίβεια των αποτελεσμάτων αυτή να γίνει απλά αυξάνοντας την διακριτοποίηση του χώρου Σχήμα 4.5 και του χρόνου Σχήμα 4.6. Τέλος για λόγους απλότητας και αμεσότητας η παρουσίαση των παραπάνω αριθμητικών μεθόδων θα γίνει με εφαρμογή τους στους όρους απλής μεταφοράς (εξισώσεις κεφαλαίου 2).

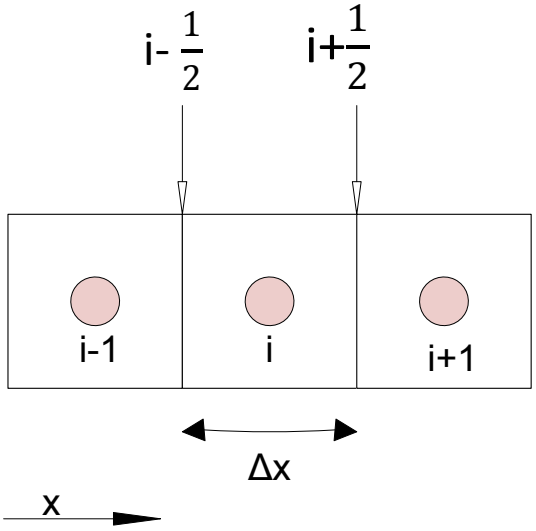


Σχήμα 4.5: Παρουσιάζεται τρισδιάστατος χώρος ο οποίος έχει διακριτοποιηθεί με κελιά διαστάσεων Δx, Δy, Δz.



Σχήμα 4.6: Παρουσιάζεται χρόνος ο οποίος έχει διακριτοποιηθεί.

4.3.1.1 ΑΝΑΝΤΗ ΚΑΙ ΚΕΝΤΡΙΚΕΣ ΔΙΑΦΟΡΕΣ



Σχήμα 4.7: Παρουσιάζεται μονοδιάστατος χώρος ο οποίος έχει διακριτοποιηθεί με μήκος κελιού Δx.

Χρησιμοποιώντας τον κατακερματισμένο χώρο Σχήμα 4.7 θα προσπαθήσουμε να σχηματίσουμε τους διάφορους όρους της εξίσωσης μεταφοράς εξ. (2.13) συναρτήσει πεπερασμένων διαφορών. Έτσι κανείς θα μπορούσε να γράψει τον όρο της μεταγωγής εναλλακτικά (εξ. 4.18 ή 4.19) ως εξής:

$$U\frac{\partial C}{\partial X} = U\frac{C_{i+\frac{1}{2}} - C_{i-\frac{1}{2}}}{\Delta X} \tag{4.18}$$

ή

$$U\frac{\partial C}{\partial X} = U\frac{C_{i+1} - C_{i-1}}{2\,\Delta X} \tag{4.19}$$

Επιπλέον επειδή τα κατακερματισμένα τμήματα έχουν σταθερές διαστάσεις καθ' όλο το μήκος του χώρου μπορούμε να γράψουμε ότι:

$$C_{i+\frac{1}{2}} = \frac{(C_{i+1} + C_i)}{2} \tag{4.20}$$

$$C_{i-\frac{1}{2}} = \frac{(C_i + C_{i-1})}{2} \tag{4.21}$$

Συνδυάζοντας τις εξισώσεις (4.20) και (4.21) προκύπτει:

$$U\frac{\partial C}{\partial X} = U\frac{C_{i+1} - C_{i-1}}{2\,\Delta X} \tag{4.22}$$

Παρ όλα αυτά κάποιος θα μπορούσε να είχε γράψει την εξίσωση (4.22) με διαφορετικό τρόπο. Δηλαδή όχι σαν συνδυασμός των συγκεντρώσεων στα σημεία $i - \frac{1}{2}$ και $i + \frac{1}{2}$ (Σχήμα 4.7) αλλά σαν συνδυασμός των i+1 και i, οπότε προκύπτει:

$$U\frac{\partial C}{\partial X} = U\frac{C_{i+1} - C_i}{\Delta X} \tag{4.23}$$

Οι εξισώσεις 4.20 και 4.21 ανήκουν στην γενικότερη μορφή

$$C_{i+\frac{1}{2}} = (1-\alpha)C_i + \alpha C_{i+1} \tag{4.24}$$

$$C_{i-\frac{1}{2}} = (1-\alpha)C_{i-1} + \alpha C_i \tag{4.25}$$

Όπου α είναι ένας συντελεστής χωρικής βαρύτητας (spatial weighting factor) (Zheng and Bennett, 1995) και μπορεί να πάρει διάφορες τιμές όπως 0 και 1. Ανάλογα με την τιμή που έχει δημιουργεί διαφορετικές προσεγγίσεις της εκάστοτε παραγώγου.

Έτσι όταν ισχύουν οι εξισώσεις (4.26) και (4.27) (U η ενδοπορώδης ταχύτητα)

α =0 εάν U>0 (4.26)

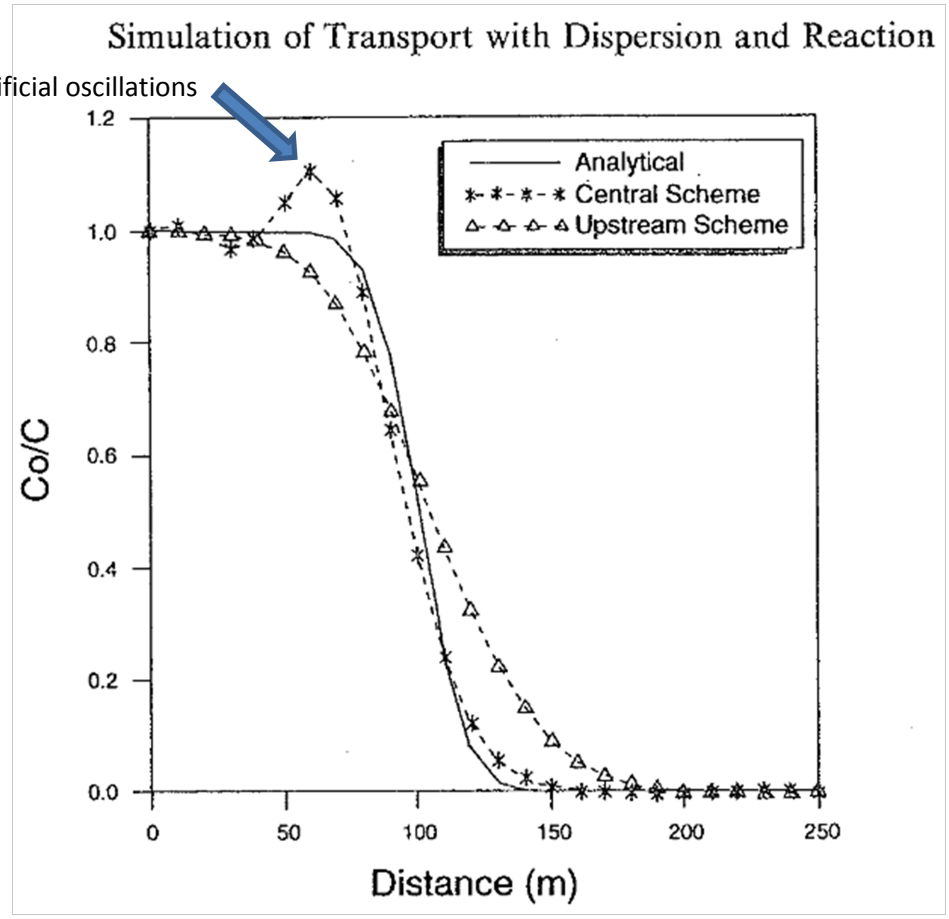
α =1 εάν U<0 (4.27)

προκύπτει η μέθοδος των ΑΝΑΝΤΗ ΔΙΑΦΟΡΩΝ (UPSTREAM DIFFERENCES SCHEME)

Τώρα η μεταγωγή γράφεται

$$U\frac{\partial C}{\partial X} = U\frac{C_i - C_{i-1}}{\Delta X} \tag{4.28}$$

Το πλεονέκτημα της μεθόδου αυτής είναι ότι δεν δημιουργεί τεχνητές ταλαντώσεις (artificial oscillations) Σχήμα 4.8 αλλά όμως εισαγάγει αριθμητική διασπορά (numerical dispersion). Αυτό οφείλεται στο γεγονός ότι η ανάντη μέθοδος έχει ακρίβεια μόνο πρώτης τάξης και έτσι προκύπτει ένα λάθος αποκοπής δεύτερης τάξης (second order truncation error) (Zheng and Bennett, 1995). Το λάθος αυτό έχει το ίδιο αποτέλεσμα στην μεταφορά του ρύπου όπως έχει και η φυσική διασπορά. Γι αυτό άλλωστε ονομάστηκε και "Τεχνητή διασπορά".



Σχήμα 4.8: Παρουσιάζεται η επίλυσης μεταφοράς ρύπων σε μια διάσταση με τρείς διαφορετικούς τρόπους: Αναλυτική μέθοδος, κεντρικές διαφορές και ανάντη διαφορές. Υπογραμμίζεται το φαινόμενο των τεχνητών ταλαντώσεων (artificial oscillations) (Zheng and Bennett, 1995).

Τώρα εάν α=0.5 τότε προκύπτει η μέθοδος των ΚΕΝΤΡΙΚΩΝ ΔΙΑΦΟΡΩΝ (CENTRAL DIFFERENCES SCHEME).

Η μεταγωγή πλέον γράφεται:

$$U\frac{\partial C}{\partial X} = U\frac{C_{i+1} - C_{i-1}}{2\,\Delta X} \tag{4.29}$$

Και η διασπορά γράφεται

$$D_x\frac{\partial^2 C}{\partial X^2} = D_x\frac{C_{i+1} - 2\,C_i + C_{i-1}}{\Delta X^2} \tag{4.30}$$

Το πλεονέκτημα της μεθόδου αυτής είναι ότι δεν εισαγάγει αριθμητική διασπορά (numerical dispersion). Αυτό οφείλεται στο γεγονός ότι η κεντρική μέθοδος έχει ακρίβεια δεύτερης τάξης (Zheng and Bennett 1995). Παρ όλα αυτά δημιουργεί τεχνητές ταλαντώσεις (artificial oscillations) Σχήμα 4.8.

4.3.1.2 ΡΗΤΟ ΚΑΙ ΥΠΟΝΟΟΎΜΕΝΟ ΣΧΗΜΑ

Μέχρι τώρα έχουμε δει πώς η διαφορετική επιλογή τμήματος κελιού μπορεί δημιουργεί διαφορετικές μεθόδους (ανάντη - κεντρικές). Όμως δεν αναφέραμε καθόλου πώς ο χρόνος επηρεάζει τις μεθόδους. Εάν με τον δείκτη n συμβολίζουμε τον παρών χρόνο και με n+1 τον επόμενο χρόνο τότε μπορούμε να γράψουμε την εξίσωση μεταφοράς εξ. (4.31), με διασπορά και μεταγωγή, στο κεντρικό σχήμα ότι ως εξής εξ. (4.32):

$$\frac{\partial c}{\partial t} = D_x \frac{\partial^2 c}{\partial x^2} - U \frac{\partial c}{\partial x} \tag{4.31}$$

$$\frac{c_i^{n+1} - c_i^n}{\Delta t} = D_x \frac{c_{i+1}^n - 2c_i^n - c_{i-1}^n}{\Delta x^2} - U \frac{c_{i+1}^n - c_{i-1}^n}{2\,\Delta x} \tag{4.32}$$

Ό μόνος άγνωστος στην εξίσωση (4.32) είναι η συγκέντρωση του κελιού i στον επόμενο χρόνο t= n+1. Έτσι επιλύοντας ως προς c_i^{n+1}προκύπτει ότι:

$$c_i^{n+1} = \frac{D_x\,\Delta t}{\Delta x^2}(c_{i+1}^n - 2\,c_i^n - c_{i-1}^n) - \frac{U\Delta t}{2\,\Delta x}(c_{i+1}^n - c_{i-1}^n) + c_i^n \tag{4.33}$$

Η εξίσωση 4.33 ανήκει στο ΡΗΤΟ ΣΧΗΜΑ (EXPLICIT SCHEME) .

Στην μέθοδο αυτή για να υπολογίσει κανείς την συγκέντρωση σε ένα κελί στο επόμενο χρόνο θα πρέπει να γνωρίζει μόνο τις συγκεντρώσεις στον προηγούμενο χρόνο. Κάτι που διευκολύνει πολύ την επίλυση του προβλήματος αφού οι συγκεντρώσεις στον χρόνο n είναι γνωστές είτε από προηγούμενους υπολογισμούς είτε από τις αρχικές συνθήκες. Και με την σειρά τους οι συγκεντρώσεις γειτονικών κελιών C_{i-1} και C_{i+1} σε χρόνο n είναι γνωστές είτε από προηγούμενους υπολογισμούς πάλι είτε από συνοριακές συνθήκες. Έτσι με πολύ εύκολο τρόπο χρησιμοποιούμε τις συνοριακές και τις αρχικές συνθήκες και υπολογίζουμε σε πρώτο χρόνο n (ο προηγούμενος χρόνος ήταν το t_0=0) όλες τις συγκεντρώσεις του χώρου και στην συνέχεια με γνωστές αυτές εφαρμόζουμε την εξ. (4.33) και υπολογίζουμε τις συγκεντρώσεις n+1. Εργαζόμαστε με αυτήν την μέθοδο μέχρι να φτάσουμε στον χρόνο που ενδιαφερόμαστε t_n.

Αντίθετα κάποιος θα μπορούσε να είχε γράψει την συγκέντρωση στον επόμενο χρόνο n+1 συναρτήσει συγκεντρώσεων πάλι στον επόμενο χρόνο n+1. Έτσι για τον όρο της διασποράς και της μεταγωγής στις κεντρικές διαφορές έχουμε εξ. (4.34, 4.35)

$$D_x \frac{\partial^2 C}{\partial X^2} = D_x \frac{c_{i+1}^{n+1} - 2\, c_i^{n+1} - c_{i-1}^{n+1}}{\Delta X^2} \quad (4.34\alpha)$$

$$U \frac{\partial c}{\partial x} = U \frac{c_{i+1}^{n+1} - c_{i-1}^{n+1}}{2\, \Delta x} \quad (4.34\beta)$$

$$\frac{\partial c}{\partial t} = \frac{c_i^{n+1} - c_i^{n}}{\Delta t} \quad (4.35)$$

Χρησιμοποιώντας τις εξισώσεις (4.34α)-(4.35) και εφαρμόζοντας τις, στην εξ. (4.31) και λύνοντας ως προς την συγκέντρωση c_i^{n+1} του κελιού i προκύπτει ότι:

$$c_i^{n+1} = \frac{D_x \Delta t}{\Delta x^2}(c_{i+1}^{n+1} - 2\, c_i^{n+1} - c_{i-1}^{n+1}) - \frac{U \Delta t}{2\, \Delta x}(c_{i+1}^{n+1} - c_{i-1}^{n+1}) \; + c_i^{n} \quad (4.36)$$

Η εξίσωση 4.36 ανήκει στο <u>ΥΠΟΝΟΟΎΜΕΝΟ ΣΧΗΜΑ</u> (IMPLICIT SCHEME).

Στην μέθοδο αυτή για να υπολογίσει κανείς την συγκέντρωση σε ένα κελί στο επόμενο χρόνο θα πρέπει να γνωρίζει όχι μόνο τις συγκεντρώσεις στον προηγούμενο χρόνο αλλά και τις συγκεντρώσεις στον επόμενο. Κάτι πού δυσκολεύει πολύ την επίλυση του γενικότερου προβλήματος. Αφού οι συγκεντρώσεις στον επόμενο χρόνο είναι άγνωστες.

Έτσι για να λυθεί η εξίσωση μεταφοράς απαιτείται σύστημα γραμμικών εξισώσεων:

$$[A]\ \{C^{n+1}\} = \{f(C^n)\} \quad (4.38)$$

Όπου Α είναι τετράγωνος πίνακας με διαστάσεις [nxn], με n να είναι ο αριθμός των αγνώστων συγκεντρώσεων (συνήθως αυτός ο αριθμός είναι ίσος με τον αριθμό των κελιών με τα οποία έχουμε κατακερματίσει τον χώρο μας), $f(C^n)$ να είναι το διάνυσμα των σταθερών όρων το οποίο αποτελείται από μια συνάρτηση των γνωστών συγκεντρώσεων C^n στον παρόν χρόνο $t=t_n$, και C^{n+1} να είναι το διάνυσμα των αγνώστων των γραμμικών εξισώσεων το οποίο αποτελείται από τις συγκεντρώσεις C^{n+1} στον επόμενο χρόνο $t=t_{n+1}$.

Τελικά αφού κανείς σχηματίσει το σύστημα των γραμμικών εξισώσεων και το επιλύσει μπορεί να έχει στην διάθεσή του τις συγκεντρώσεις C^{n+1} στον επόμενο χρόνο $t=t_{n+1}$. Στην συνέχεια τις χρησιμοποιεί ώστε να υπολογίσει το καινούριο $\{f(C^n)\}$ (θεωρώντας ότι οι συγκεντρώσεις που μόλις υπολόγισε πλέον αναφέρονται στον παρόν χρόνο $t=t_n$) και ξανά λύνει το σύστημα εξ. (4.38) ώστε να υπολογίσει πλέον τις νέες συγκεντρώσεις C^{n+1} στον επόμενο χρόνο $t_{n+1}=t_n+\Delta t$. Αυτή η διαδικασία συνεχίζεται με χρονικά βήματα Δt μέχρι ο χρόνος t να γίνει ίσος με τον επιθυμητό χρόνο t_0όπου και οι υπολογισμοί σταματάνε.

4.3.1.2 CRANK NICOLSON ΣΧΗΜΑ

Το ρητό εξ. (4.33) και το υπονοούμενο εξ. (4.36) σχήμα δεν είναι οι μοναδικές χρονικές διακριτοποιήσεις που υπάρχουν, αλλά ανήκουν σε μια γενικότερη κατηγορία η οποία δίδεται από την εξ. (4.37).

$$c_i^{n+1} = (1-wm)\left(\frac{D_x\,\Delta t}{\Delta x^2}(c_{i+1}^n - 2\,c_i^n - c_{i-1}^n) - \frac{U\Delta t}{2\,\Delta x}(c_{i+1}^n - c_{i-1}^n)\right) + wm\left(\frac{D_x\Delta t}{\Delta x^2}(c_{i+1}^{n+1} - 2\,c_i^{n+1} - c_{i-1}^{n+1}) - \frac{U\Delta t}{2\,\Delta x}(c_{i+1}^{n+1} - c_{i-1}^{n+1})\right) + c_i^n \qquad (4.37)$$

Ο καινούριος όρος που εισάγεται στην εξ. (4.37) είναι ο συντελεστής wm ο οποίος καθορίζει και την χρονική διακριτοποίηση. Εάν wm=0 τότε έχουμε Ρητό σχήμα, εάν wm=1 έχουμε Υπονοούμενο σχήμα και εάν wm=0.5 τότε έχουμε Crank Nicolson σχήμα (Zheng and Bennett, 1995). Αυτό που προσφέρει η τελευταία μέθοδος (Crank Nicolson) είναι ο συνδυασμός των θετικών των δυο γνωστών μεθόδων. Δηλαδή αυξημένη ακρίβεια από το ρητό σχήμα (δευτέρας τάξης ακρίβεια) αλλά και αυξημένη σταθερότητα από το υπονοούμενο σχήμα. Παρ' όλα αυτά δεν συνηθίζεται πολύ η χρήση του γιατί ενώ σε απλές περιπτώσεις μεταφοράς (διασπορά-μεταγωγή) είναι σταθερό (unconditionally stable Peaceman (1977)), σε πιο πολύπλοκες κάτι τέτοιο δεν ισχύει και υπάρχει απαίτηση για πολύ μικρό χρονικό βήμα dt. Αποτέλεσμα αυτού να είναι η χρήση του Υπονοούμενου σχήματος που απαιτεί λιγότερες επαναλήψεις (μεγαλύτερο dt) και παρέχει ικανοποιητική ακρίβεια.

4.3.2.1 ΜΕΘΟΔΟΙ ΕΠΙΛΥΣΗΣ ΣΥΣΤΗΜΑΤΩΝ ΓΡΑΜΜΙΚΏΝ ΕΞΙΣΏΣΕΩΝ

Τα συστήματα γραμμικών εξισώσεων συναντώνται συνεχώς στις αριθμητικές επιλύσεις και χαρακτηρίζουν την αποτελεσματικότητα των μεθόδων αυτών. Γι αυτό είναι απαραίτητο να έχει κανείς στην κατοχή του γρήγορες και ακριβείς ρουτίνες επίλυσης (συστημάτων).

Συνολικά υπάρχουν δύο κατηγορίες μεθόδων επίλυσης:

A. Οι επαναληπτικές μέθοδοι
B. Οι κατευθείαν μέθοδοι

Στην πρώτη κατηγορία ανήκουν μέθοδοι οι οποίοι προσπαθούν να λύσουν τις εξισώσεις εξ. (4.38) ύστερα από πολλές προσεγγίσεις. Αρχικά ξεκινάνε με ένα διάνυσμα το οποίο αποτελεί μια πιθανή λύση {possible C^{n+1}} και στην συνέχεια με επαναλήψεις το βελτιώνουν. Το πότε θα σταματήσει η μέθοδος αυτή εξαρτάται από το κριτήριο σύγκλισης. Συνήθως αυτό είναι ότι δύο διαδοχικές λύσεις δεν θα πρέπει να έχουν σχετικό σφάλμα μεγαλύτερο από 5% (εξ. 4.39).

$$\frac{(C_{n+1}-C_n)}{C_{n+1}} \leq 5\% \qquad (4.39)$$

Στην δεύτερη κατηγορία ανήκουν μέθοδοι οι οποίοι επιχειρούν απευθείας να βρουν την τελική λύση $\{C^{n+1}\}$. Οι πιο συνηθισμένοι βασίζονται είτε στην αντιστροφή του πίνακα [A] εξ. (4.38) είτε στην δημιουργία ενδιάμεσων πινάκων οι οποίοι μπορούν με αποτελεσματικότητα σε δύο βήματα να βρουν την τελική λύση. Η τελευταία μέθοδος ονομάζεται "αποσύνθεση σε LU πίνακες" (LU decomposition). Επειδή όμως η αντιστροφή πίνακα κοστίζει πολύ χρόνο χωρίς πάντα να είναι απαραίτητη προτιμάται τελικά η Lu decompotision.

4.3.2.2 ΣΥΝΔΥΑΣΜΟΣ ΚΑΤΕΥΘΕΙΑΝ ΚΑΙ ΕΠΑΝΑΛΗΠΤΙΚΩΝΜΕΘΟΔΩΝ

Όλες οι μέθοδοι επίλυσης γραμμικών συστημάτων έχουν πλεονεκτήματα και μειονεκτήματα. Έτσι για τις επαναληπτικές κανείς θα μπορούσε να πει ότι είναι αρκετά γρήγορες, εφαρμόζονται με εύκολο τρόπο σε συστήματα μεγάλου μεγέθους, αλλά όμως είναι λιγότερο ακριβείς. Με την σειρά τους οι κατευθείαν μέθοδοι είναι αργές αλλά η ακρίβεια τους είναι πολύ καλή. Γι αυτό θα ήταν καλό κανείς να μπορούσε να συνδυάσει τις δύο μεγάλες κατηγορίες επίλυσης ώστε να έχει και καλή ταχύτητα και καλή ακρίβεια.

Γενικά υπάρχουν διάφορα είδη πινάκων. Η πιο συνηθισμένη μορφή τους είναι αυτή του σχήματος Σχήμα 4.9α, τυχαίοι μη μηδενικοί πίνακες. Αυτοί ιδιαίτερα όταν έχουν μεγάλο μέγεθος δύσκολά επιλύονται. Παρόμοια συμπεριφορά έχουν και οι πίνακες του σχήματος Σχήμα 4.9β. Όπου αν και υπάρχουν μηδενικά που κανείς θα μπορούσε να τα αξιοποιήσει, δυστυχώς αυτά είναι διατεταγμένα με τυχαίο τρόπο. Σε αντίθεση με τις προηγούμενες δυο μορφές, οι εξισώσεις μεταφοράς σχηματίζουν συνήθως διαγώνιους πίνακες Σχήμα 4.9γ ή Σχήμα 4.9δ. Σε αυτές τις περιπτώσεις παρατηρούνται πολλά μηδενικά στα πάνω δεξιά και κάτω αριστερά άκρα ενώ όλοι οι διαγώνιοι όροι είναι μη μηδενικοί. Στην πρώτη περίπτωση Σχήμα 4.9γ κανείς μπορεί να εκμεταλλευτεί το γεγονός αυτό, να μην κάνει καθόλου πράξεις με τους μηδενικούς όρους (να μην τους αποθηκεύσει καν στην μνήμη του υπολογιστή) και να λύσει τον πίνακα πολύ γρήγορα και με μεγάλη ακρίβεια. Η επίλυση διαγώνιων πινάκων είναι τάξης μεγέθους πιο γρήγορη από αυτήν των τυχαίων πινάκων Σχήμα 4.9α. Γι αυτό σε κάθε περίπτωση είναι επιθυμητός ο σχηματισμός πινάκων 4.9γ. Παρ όλα αυτά συνήθως αντιμετωπίζονται πίνακες όπως Σχήμα 4.9δ. Σε αυτές τις περιπτώσεις είναι χρήσιμός ο συνδυασμός επαναληπτικών και κατευθείαν μεθόδων επίλυσης.

Είναι δυνατή η χρήση επαναληπτικών μεθόδων με τις οποίες κανείς μπορεί να αφαιρέσει τα στοιχεία (α_{25}) και (α_{51}) από τον πίνακα 4.9δ και να τον φέρει στην μορφή του πίνακα Σχήμα 4.9γ. Τώρα είναι πολύ πιο εύκολο με κατευθείαν μεθόδους να λυθεί όχι σαν τυχαίος πίνακας Σχήμα 4.9α η 4.9β αλλά περίπου σαν διαγώνιος (band matrix) Σχήμα 4.9ε.

$$\begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ l_{21} & l_{22} & l_{23} & l_{24} \\ l_{31} & l_{32} & l_{33} & l_{34} \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}$$

(α)

$$\begin{bmatrix} l_{11} & 0 & l_{13} & 0 \\ 0 & 0 & l_{23} & 0 \\ l_{31} & l_{32} & l_{33} & l_{34} \\ 0 & l_{42} & 0 & l_{44} \end{bmatrix}$$

(β)

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ 0 & l_{22} & 0 & 0 \\ 0 & 0 & l_{33} & 0 \\ 0 & 0 & 0 & l_{44} \end{bmatrix}$$

(γ)

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & \{a_{25}\} & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ \{a_{51}\} & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

(δ)

$$\begin{bmatrix} d_{11} & d_{12} & 0 & 0 & 0 & 0 \\ d_{21} & d_{22} & d_{23} & 0 & 0 & 0 \\ 0 & d_{32} & d_{33} & d_{34} & 0 & 0 \\ 0 & 0 & d_{43} & d_{44} & d_{45} & 0 \\ 0 & 0 & 0 & d_{54} & d_{55} & d_{56} \\ 0 & 0 & 0 & 0 & d_{65} & d_{66} \end{bmatrix}$$

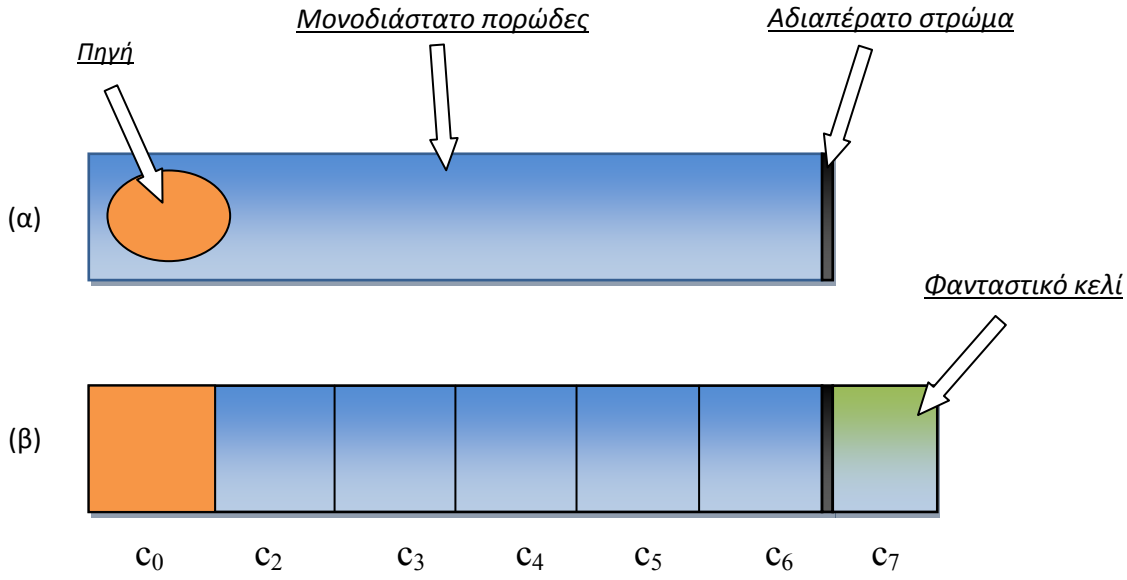
(ε)

Σχήμα 4.9: Παρουσιάζονται διαφορετικά είδη πινάκων.

4.4 ΠΑΡΆΔΕΙΓΜΑ ΕΦΑΡΜΟΓΗΣ ΥΠΟΝΟΟΥΜΕΝΟΥ ΣΧΗΜΑΤΟΣ (IMPLICIT SCHEME)

Για να γίνει περισσότερο αντιληπτό πως δουλεύει το υπονοούμενο σχήμα (implicit none), θα γίνει εφαρμογή του σε ένα παράδειγμα.

Έστω ότι έχουμε την μονοδιάστατη εξίσωση απλής μεταφοράς ρύπου με μεταγωγή και διάχυση εξ. (4.40), και θέλουμε να την εφαρμόσουμε σε ένα πορώδες Σχήμα 4.10. Το οποίο στην αρχή του έχει μία πηγή με σταθερή συγκέντρωση C_0 και στο τέλος του αδιαπέρατο στρώμα.



Σχήμα 4.10: Παρουσιάζεται πεπερασμένο πορώδες μέσο (α) στην αρχή του οποίου βρίσκεται πηγή C_0 και στο τέλος του αδιαπέρατο στρώμα. Στην συνέχεια το πορώδες αυτό κατακερματίζεται σε μικρά τμήματα (β).

Για το σχήμα 4.10 μπορεί κανείς να γράψει τις εξισώσεις:

$$\frac{\partial c}{\partial t} = D_x \frac{\partial^2 c}{\partial x^2} - U \frac{\partial c}{\partial x} \tag{4.40}$$

$$C_1 = C_0 \ = \sigma\tau\alpha\theta\varepsilon\rho\acute{o} \tag{4.41}$$

$$-\mathrm{D}_x \frac{\mathrm{c}_7 - \mathrm{c}_6}{\mathrm{x}_7 - \mathrm{x}_6} = 0 \qquad (4.42)$$

Η εξίσωση 4.40 με την κεντρική μέθοδο για το υπονοούμενο σχήμα (central -implicit scheme) μπορεί να γραφτεί σαν εξίσωση εξ. (4.43). Με την σειρά της η εξ. (4.41) αποτελεί μια συνθήκη Dirichlet και συμπεριφέρεται σαν πηγή. Τέλος η εξ. (4.42) είναι η υλοποίηση της συνοριακής συνθήκης του Neumann εξ. (2.19) και περιγράφει ότι στο τέλος, το πορώδες μέσο έχει αδιαπέρατο στρώμα Σχήμα 4.10α.

$$c_i^{n+1} = \frac{D_x\, \Delta t}{\Delta x^2}(c_{i+1}^{n+1} - 2\, c_i^{n+1} - \, c_{i-1}^{n+1}) \, - \frac{U \Delta t}{2\, \Delta x}(c_{i+1}^{n+1} \, - c_{i-1}^{n+1}) \, + c_i^n \qquad (4.43)$$

Παρατηρώντας το Σχ. 4.10α και ταυτόχρονα το 4.10β εντοπίζεται μια σημαντική διαφορά. Το δεύτερο προεκτείνεται κατά την διεύθυνση του x άξονα και περιέχει ένα επιπλέον κελί το C_7. Το κελί αυτό δεν αντιστοιχεί σε καμία περιοχή του αρχικού πορώδους. Η εξήγηση για την διαφορά αυτή βασίζεται στο γεγονός ότι για να εφαρμοσθεί σωστά η συνοριακή συνθήκη του Neumann εξ. (4.42) απαιτείται να προστεθεί στο πορώδες ένα επιπλέον "φανταστικό" κελί. Τον ρόλο του κελιού αυτού τον διαδραματίζει το C7.

Η διαδικασία που ακολουθείται για την επίλυση αυτού του προβλήματος είναι

- Έστω ότι έχουμε χωρίσει σε "6 " κελιά τον χώρο μας

1. Γράφεται για κάθε κελί η εξίσωση μεταφοράς (4.43).
2. Εφαρμόζονται οι αρχικές –συνοριακές συνθήκες (4.41, 4.42).
3. Σχηματίζεται σύστημα εξισώσεων,

   $[A]\ \{C^{n+1}\}=\{f(C^n)\}$

4. Επιλύεται το σύστημα.
5. Εάν έχουμε ορίζουσα που πλησιάζει το 0 , χρησιμοποιούμε κάποια μέθοδο για την βελτίωση αποτελεσμάτων (μέθοδος SVD).
6. Ελέγχουμε τα αποτελέσματα για πιθανές αστοχία σύγκλισης.

Έτσι τελικά αποκτάμε τον πίνακα που βρίσκεται στο Σχήμα 4.11. Το λύνουμε και λαμβάνουμε σε οποιαδήποτε χρόνο χρειαζόμαστε όλες τις διακεκριμένες συγκεντρώσεις του πορώδους. Στο βήμα 5 αναφερόμαστε στην περίπτωση που η ορίζουσα ενός πίνακα πλησιάζει το 0. Τότε τα αποτελέσματα από την επίλυση του συστήματος των εξισώσεων είναι εσφαλμένα και κρίνεται απαραίτητο να γίνει κάποια βελτίωση γι αυτά. Υπάρχουν διάφοροι τρόποι για να γίνει αυτό ο πιο

κλασικός και γρήγορος περιγράφεται από τους (Teukolsky et. al, 1997) με τον όρο "Iterative refinement". Το σημαντικό είναι να γνωρίζει κανείς πως ακόμα και εάν βγάλει κάποια αποτελέσματα με τις διάφορες μεθόδους επίλυσης γραμμικών συστημάτων, αυτά δεν είναι απαραίτητα σωστά. Για αυτό θα πρέπει να υπάρχει ένας σχετικός έλεγχος των λύσεων ώστε να εντοπισθούν τυχόν αστοχίες και εάν κριθεί απαραίτητο να επιστρατευτούν μέθοδοι που βελτιώνουν την ακρίβεια τους.

$$
\begin{aligned}
-3C_2^{n+1} + 0.5C_3^{n+1} &= -C_2^n - 1.5C_0 \\
1.5C_2^{n+1} - 3C_3^{n+1} + 0.5C_4^{n+1} &= -C_3^n \\
1.5C_3^{n+1} - 3C_4^{n+1} + 0.5C_5^{n+1} &= -C_4^n \\
1.5C_4^{n+1} - 3C_5^{n+1} + 0.5C_6^{n+1} &= -C_5^n \\
1.5C_5^{n+1} - 2.5C_6^{n+1} &= -C_6^n
\end{aligned}
$$

Σχήμα 4.11: Παρουσιάζεται ο πίνακας που σχηματίζεται εάν γραφτεί για κάθε κελί του Σχήματος 4.10 η εξ. (4.40) και συνάμα εφαρμοσθούν οι απαραίτητες συνοριακές συνθήκες (4.41), (4.42) (Zheng and Bennett, 1995).

5. ΕΠΙΛΥΣΗ ΔΙΑΦΟΡΙΚΩΝ ΕΞΙΣΩΣΕΩΝ ΑΠΛΗΣ ΜΕΤΑΦΟΡΑΣ ΚΑΙ ΣΥΜΜΕΤΑΦΟΡΑΣ

5.1 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΑΙ ΕΦΑΡΜΟΓΗ ΑΡΙΘΜΗΤΙΚΩΝ ΜΕΘΟΔΩΝ ΣΤΗΝ ΤΡΙΣΔΙΑΣΤΑΤΗ ΕΞΙΣΩΣΗ ΑΠΛΗΣ ΜΕΤΑΦΟΡΑΣ

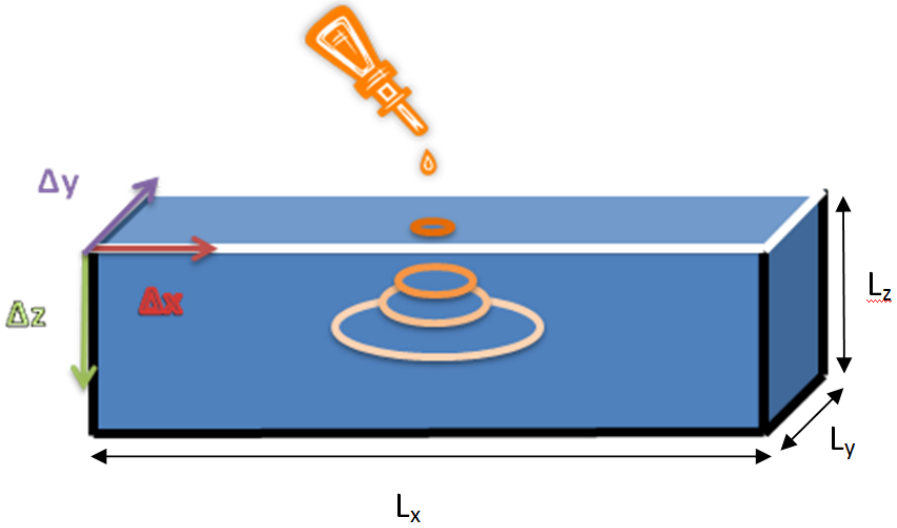
Στην περίπτωση που έχουμε απλή μεταφορά ρύπων (και όχι συμμεταφορά) το μαθηματικό μοντέλο μας περιγράφεται επαρκώς από τις εξ. (5.1, 5.2) (μεταφορά και προσρόφηση αντίστοιχα).

$$\frac{\partial C(t,x,y,z)}{\partial t} + \frac{\rho}{\theta}\frac{\partial C^*(t,x,y,z)}{\partial t} - D_x\frac{\partial^2 C(t,x,y,z)}{\partial x^2} - D_y\frac{\partial^2 C(t,x,y,z)}{\partial y^2} - D_z\frac{\partial^2 C(t,x,y,z)}{\partial z^2} + U\frac{\partial C(t,x,y,z)}{\partial x} + \lambda\, C(t,x,y,z) + \frac{\lambda^*\rho}{\theta}C^*(t,x,y,z) = F(t,x,y,z) \quad (5.1)$$

$$\frac{\rho}{\theta}\frac{\partial C^*(t,x,y,z)}{\partial t} = r_1 C(t,x,y,z) - r_2\frac{\rho}{\theta}C^*(t,x,y,z) - \frac{\lambda^*\rho}{\theta}C^*(t,x,y,z) \quad (5.2)$$

Πριν εφαρμόσουμε τις παραπάνω εξισώσεις θα πρέπει να ορίσουμε το μοντέλο μας ώστε να επιλέξουμε τις σωστές συνοριακές συνθήκες. Σε αντίθεση με τις αναλυτικές λύσεις που εφαρμόσθηκαν σε πορώδη όπως Σχ. (4.1)-(4.4), στις αριθμητικές λύσεις δεν υπάρχουν άπειρα και ημιάπειρα μεγέθη. Όλα είναι πεπερασμένα γι αυτό άλλωστε και η αντίστοιχη μέθοδος που θα υλοποιήσουμε ονομάζεται "ΠΕΠΕΡΑΣΜΕΝΕΣ ΔΙΑΦΟΡΕΣ ".

Το μοντέλο μας αποτελείται από πορώδες κυβικού σχήματος, το οποίο είναι περιορισμένο σε όλες του τις πλευρές από αδιαπέραστα στρώματα. Στο πάνω μέρος του υπάρχει επιφανειακή πηγή κυκλικού σχήματος η οποία δεν έχει σταθερή συγκέντρωση και παρέχει στο μοντέλο μας ρύπο, με σταθερή ροή (g/day). Στην αρχή του χρόνου σε όλο το πορώδες υπήρχε απουσία ρύπου. Τέλος κατά μήκος του άξονα x υπάρχει ροή με σταθερή ταχύτητα u (cm/day). Όλα αυτά περιγράφονται στο Σχήμα 5.1, στο οποίο εμείς έχουμε ήδη επιλέξει την φορά των αξόνων.



Σχήμα 5.1: Παρουσιάζεται πορώδες στο οποίο εισέρχεται ρύπος με σταθερή ροή μάζας (απλή μεταφορά). Οι μαύρες γραμμές περιγράφουν αδιαπέραστα εδαφικά στρώματα ενώ οι άσπρες την διεπιφάνεια του πορώδους με τον αέρα.

Οι αρχικές και συνοριακές συνθήκες που περιγράφουν το μοντέλο μας είναι:

$$C(0,x,y,z)=0 \tag{5.3}$$

$$\frac{\partial C(t,0,y,z)}{\partial x}=0 \tag{5.4}$$

$$\frac{\partial C(t,L_x,y,z)}{\partial x}=0 \tag{5.5}$$

$$\frac{\partial C(t,x,0,z)}{\partial y}=0 \tag{5.6α}$$

$$\frac{\partial C(t,x,L_y,z)}{\partial y}=0 \tag{5.6β}$$

$$\frac{\partial C(t,x,y,0)}{\partial Z}=0 \tag{5.7α}$$

$$\frac{\partial C(t,x,y,L_z)}{\partial Z}=0 \tag{5.7β}$$

Για όλα τα κελιά εκτός από αυτά της πηγής ισχύει η εξίσωση (5.8). Ενώ μόνο για τα κελία τα οποία λειτουργούν σαν πηγή ισχύει η εξ. (5.9) με την (5.10). Όπου G=μάζα διαλυμένης ουσίας ανά χρόνο ανά μονάδα όγκου ($\frac{g}{ml\ day}$).

$$\mathrm{F(t,x,y,z)} = 0 \tag{5.8}$$

$$\mathrm{F(t,x,y,z)} = \mathrm{G\ W} \tag{5.9}$$

$$\mathrm{w} = \frac{1}{\theta} \tag{5.10}$$

Χρησιμοποιώντας central scheme για τον όρο της διασποράς και upstream scheme για το όρο της μεταγωγής μετατρέπουμε όλες τις παραγώγους σε πεπερασμένες διαφορές. Έτσι τελικά φτάνουμε σε δυο ομάδες συστημάτων εξισώσεων που όμως πρέπει να λυθούν μαζί. Η πρώτη αντιστοιχεί στην εξίσωση μεταφοράς και έχει την μορφή εξ. (5.11):

$$A_1 C_{i,j,w}^{n+1} + A_2 C_{i,j,w}^{*\,n+1} + A_3 C_{i-1,j,w}^{n+1} + A_4 C_{i+1,j,w}^{n+1} + A_5 C_{i,j-1,w}^{n+1} + A_6 C_{i,j+1,w}^{n+1} + A_7 C_{i,j,w-1}^{n+1} + A_8 C_{i,j,w+1}^{n+1} = b_1 \tag{5.11}$$

$$A_1 = 1 + dt\left(2\frac{D_x}{dx^2} + 2\frac{D_y}{dy^2} + 2\frac{D_z}{dz^2} + \lambda + \frac{u}{dx}\right) \tag{5.12}$$

$$A_2 = \frac{1+dt\ \lambda^*}{\theta}\rho \tag{5.13}$$

$$A_3 = -dt\frac{D_x}{dx^2} - dt\frac{U}{dx} \tag{5.14}$$

$$A_4 = -dt\frac{D_x}{dx^2} \tag{5.15}$$

$$A_5 = -dt\frac{D_y}{dy^2} \tag{5.16}$$

$$A_6 = -dt\frac{D_y}{dy^2} \tag{5.17}$$

$$A_7 = -dt\frac{D_z}{dz^2} \tag{5.18}$$

$$A_8 = -dt\frac{D_z}{dz^2} \tag{5.19}$$

$$b_1 = c_{i,j,w}^{n} + \frac{\rho}{\theta} c_{i,j,w}^{*n} \tag{5.20}$$

Ενώ η δεύτερη ομάδα εξισώσεων αντιστοιχεί στη προσρόφηση και έχει την μορφή εξ. (5.21):

$$A_9 C_{i,j,w}^{n+1} + A_{10} C_{i,j,w}^{*\,n+1} = b_2 \tag{5.21}$$

Όπου :

$$A_9 = -dt \frac{r_1 \theta}{\rho} \tag{5.22}$$

$$A_{10} = 1 + dt\ (\lambda^* + r_2) \tag{5.23}$$

$$b_2 = c_{i,j,w}^{*n} \tag{5.24}$$

Για όλα τα παραπάνω ισχύει ότι $c_{i,j,w}^{*n}$ είναι η προσροφημένη συγκέντρωση στον γνωστό παρόν χρόνο n του κελιού με συντεταγμένες i,j,w. Ενώ $c_{i,j,w}^{n}$ είναι η συγκέντρωση του ρύπου πάλι στον παρόν χρόνο n του κελιού με συντεταγμένες i,j,w. Χρησιμοποιώντας τις συνοριακές και τις αρχικές συνθήκες (5.1)-(5.10) σε συνδυασμό με τις ομάδες εξισώσεων (5.11) και (5.21) μεταφοράς και προσρόφησης σχηματίζεται ένα σύστημα εξισώσεων, με 2 nx*ny*nz αγνώστους (nx, ny, nz αριθμός κελιών στις 3 διαστάσεις αντίστοιχα). Το οποίο έχει πίνακα (μητρώο) της μορφής Σχήμα 4.9δ και επιλύεται εύκολα με την μέθοδο LU Decomposition (κεφ. 4.3.2.1). Το αποτέλεσμα αυτής της επίλυσης είναι η παραλαβή όλων των συγκεντρώσεων του ρύπου σε διακριτές χωρικές θέσεις και συγκεκριμένες χρονικές στιγμές που επιβάλλονται από το χωρικό και χρονικό βήμα που έχει χρησιμοποιηθεί, αντίστοιχα.

5.2 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΑΙ ΕΦΑΡΜΟΓΗ ΑΡΙΘΜΗΤΙΚΩΝ ΜΕΘΟΔΩΝ ΣΤΗΝ ΤΡΙΣΔΙΑΣΤΑΤΗ ΕΞΙΣΩΣΗ ΣΥΜΜΕΤΑΦΟΡΑΣ

Όταν όμως συμβεί να έχουμε συμμεταφορά δυο ρύπων (π.χ. ιός-άργιλος) τότε το μαθηματικό μοντέλο μας χρειάζεται έξι εξισώσεις για να περιγραφεί σωστά: εξ. (3.4, 3.5), (3.15, 3.16) και εξ. (3.8, 3.23). Δεν θα επαναλάβουμε εδώ τις έξι εξισώσεις, αντίθετα θα γενικεύσουμε το πρόβλημα στις τρείς διαστάσεις. Τώρα η εξίσωση μεταφοράς της αργίλου εξ. (3.4) και η εξίσωση συμμεταφοράς ιού εξ. (3.8) γράφεται ως εξ. (5.25) και εξ. (5.26) αντίστοιχα.

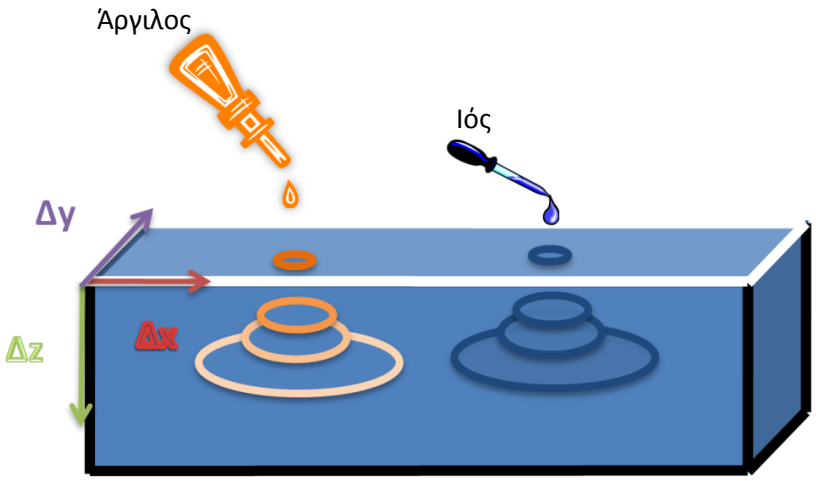
$$\frac{\partial C_c(t,x,y,z)}{\partial t} + \frac{\rho}{\theta}\frac{\partial C_c^*(t,x,y,z)}{\partial t} - D_{cx}\frac{\partial^2 C_c(t,x,y,z)}{\partial x^2} - D_{cy}\frac{\partial^2 C_c(t,x,y,z)}{\partial y^2} - D_{cz}\frac{\partial^2 C_c(t,x,y,z)}{\partial z^2} + U\frac{\partial C_c(t,x,y,z)}{\partial x} + L_c\, C_c(t,x,y,z) + \frac{L_c^*\rho}{\theta}C_c^*(t,x,y,z) = F_c(t,x,y,z) \qquad (5.25)$$

$$\frac{\partial C_v(t,x,y,z)}{\partial t} + \frac{\rho}{\theta}\frac{\partial C_v^*(t,x,y,z)}{\partial t} + \frac{\partial C_C C_{vc}(t,x,y,z)}{\partial t} + \frac{\rho}{\theta}\frac{\partial C_C C_{vc}^*(t,x,y,z)}{\partial t} = D_{vx}\frac{\partial^2 C_v(t,x,y,z)}{\partial x^2} + D_{vy}\frac{\partial^2 C_v(t,x,y,z)}{\partial y^2} + D_{vz}\frac{\partial^2 C_v(t,x,y,z)}{\partial z^2} + D_{vcx}\frac{\partial^2 C_C C_{vc}(t,x,y,z)}{\partial x^2} + D_{vcy}\frac{\partial^2 C_C C_{vc}(t,x,y,z)}{\partial y^2} + D_{vcz}\frac{\partial^2 C_C C_{vc}(t,x,y,z)}{\partial z^2} - U\frac{\partial C_v(t,x,y,z)}{\partial x} - U\frac{\partial C_C C_{vc}(t,x,y,z)}{\partial x} - L_v C_v(t,x,y,z) - \frac{L_v^*\rho}{\theta}C_v^*(t,x,y,z) - L_{vc}\, C_C C_{vc}(t,x,y,z) - \frac{L_{vc}^*\rho}{\theta}C_C C_{vc}^*(t,x,y,z) + F_v \qquad (5.26)$$

Η εξ. (5.26) είναι η γενικευμένη εξίσωση συμμεταφοράς στις τρείς διαστάσεις όταν λαμβάνεται υπόψη διασπορά, μεταγωγή, προσρόφηση και αποδόμηση. Οι όροι που εμφανίζονται στις εξ. (5.25, 5.26) έχουν τους εξής συμβολισμούς: C είναι όρος συγκέντρωσης οπότε, C_c είναι το αιωρούμενο κολλοειδές, C_v είναι ο αιωρούμενος ιός, C_{vc} είναι ο ιός που έχει προσροφηθεί επάνω στην επιφάνεια του κολλοειδούς, C_c^* είναι το προσροφημένο κολλοειδές επάνω στο στερεό πορώδες, C_v^* είναι ο προσροφημένος ιός επάνω στο στερεό πορώδες (σφαίρες γυαλιού), C_{vc}^* είναι ιός που έχει προσροφηθεί επάνω στην επιφάνεια του κολλοειδούς και στην συνέχεια ξανά προσροφάται επάνω στο στερεό πορώδες. Συνεχίζοντας D είναι όρος υδροδυναμικής διασποράς οπότε D_{vx} είναι διασπορά του ιού στην διεύθυνση x, D_{vy} , D_{vz} όροι διασποράς στις άλλες δύο διευθύνσεις, D_{vcx} διασπορά του συμπλέγματος C_{vc} στην διεύθυνση x, D_{vy}, D_{vcz} όροι διασποράς στις άλλες δυο διευθύνσεις. Ενώ λ είναι όροι ρυθμού αποδόμησης, συνεπώς λ_v αποδόμηση ιού, λ_v^* αποδόμηση προσροφημένου

ιού επάνω στο στερεό πορώδες, λ_{vc} αποδόμηση του συμπλέγματος C_{vc} ενώ λ_{vc}^{*} είναι ρυθμός αποδόμησης του συμπλέγματος C_{vc}*. Τέλος U είναι η ενδοπορώδης ταχύτητα του ρευστού στο οποίο γίνεται η συμμεταφορά και F_v είναι ο ρυθμός εισαγωγής ιού στο πορώδες.

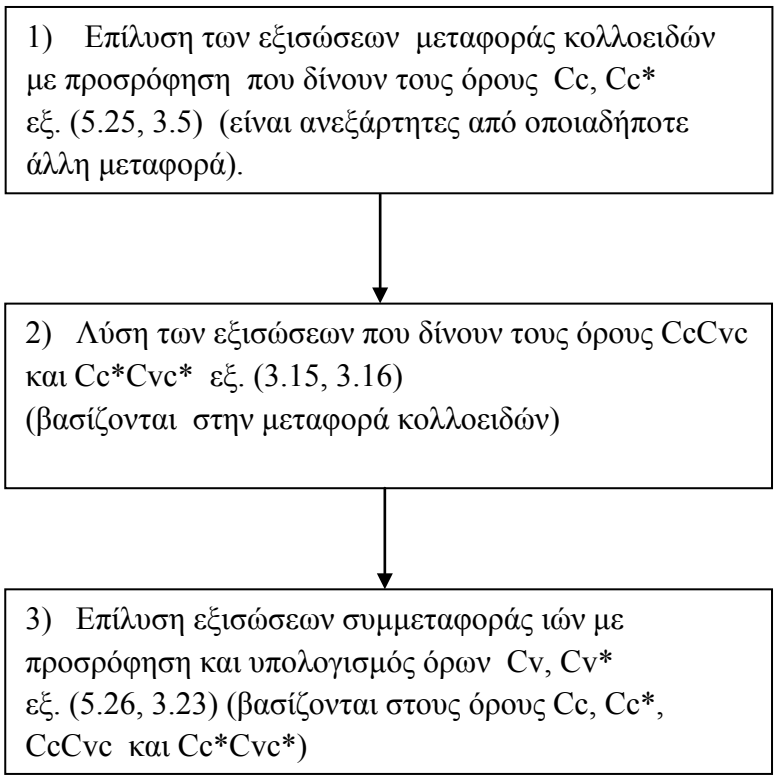
Όπως και στο κεφ. 5.1 πριν εφαρμόσουμε τις παραπάνω εξισώσεις (5.25, 5.26) θα πρέπει να ορίσουμε το μοντέλο μας ώστε να επιλέξουμε τις σωστές συνοριακές συνθήκες. Θεωρώντας ότι το Σχήμα 5.2 περιγράφει επαρκώς το πρόβλημα συμμεταφοράς που έχουμε, θα χρησιμοποιήσουμε τις εξ. (5.1-5.7), που δηλώνουν μηδενική αρχική συγκέντρωση και αδιαπέραστα όρια σε όλες τις διευθύνσεις.



Σχήμα 5.2: Παρουσιάζεται πορώδες στο οποίο εισέρχονται δύο διαφορετικών ειδών ρύποι, ιός και άργιλος, με σταθερή ροή μάζας. Η δυνατότητα που έχει ο ιός να προσροφάται στην άργιλο καταδεικνύει την ύπαρξη του φαινομένου συμμεταφοράς. Οι μαύρες γραμμές περιγράφουν αδιαπέραστα εδαφικά στρώματα ενώ οι άσπρες την διεπιφάνεια του πορώδους με τον αέρα.

Αφού πλέον είναι διαθέσιμες και οι τρισδιάστατες εξισώσεις μεταφοράς-συμμεταφοράς το μόνο που μένει είναι να τις συνδυάσουμε σωστά και να εφαρμόσουμε τις απαραίτητες αριθμητικές μεθόδους. Παρακάτω φαίνεται η σειρά επίλυσης:

Η διαδικασία επίλυσης της συμμεταφοράς επιβάλει τρείς διαδοχικές επιλύσεις



1η ΕΠΙΛΥΣΗ

Η πρώτη λύση αναφέρεται στην απλή μεταφορά αργίλου (κολλοειδές). Την ξανά έχουμε δει στο κεφ. 5.1 παρ' όλα αυτά εδώ θα γίνει μια λίγο διαφορετική επίλυση η οποία περιέχει μέσα και τον όρο "wm", εάν wm=0.5 τότε αυτή ονομάζεται μέθοδος Implicit Crank Nikolson. Κρίνεται σκόπιμη η χρήσης της εδώ καθώς μπορεί σε μερικές περιπτώσεις με μικρότερη χρονική διακριτοποίηση (μεγαλύτερο dt συνεπάγεται λιγότερες επαναλήψεις) να αποφέρει μικρότερο σφάλμα σε σχέση με την Full-Implicit. Γενικότερα στην συμμεταφορά υπάρχουν πολλοί άγνωστοι (έξι σε σχέση με δύο της απλή μεταφοράς) οπότε και αυξάνεται εκθετικά το μέγεθος των συστημάτων προς επίλυση. Αυτό σημαίνει πολλαπλάσιος χρόνος υπολογισμών πριν τα αποτελέσματα είναι διαθέσιμα. Συνεπώς ακόμα και η πιο μικρή οικονομία χρόνου μπορεί να κάνει την διαφορά μεταξύ ωρών ή λεπτών (υπολογιστικών πράξεων).

Χρησιμοποιώντας το κεντρικό σχήμα (central scheme) τόσο για τον όρο της διασποράς όσο και για το όρο της μεταγωγής, μετατρέπουμε όλες τις παραγώγους της εξ. (5.1, 5.2) σε πεπερασμένες διαφορές. Έτσι φτάνουμε σε δυο ομάδες εξισώσεων που πρέπει να λυθούν μαζί. Η πρώτη αντιστοιχεί στην απλή εξίσωση μεταφοράς και έχει την μορφή εξ. (5.27).

$$A_1 Cc_{i,j,w}^{n+1} + A_2 Cc_{i,j,w}^{*\,n+1} + A_3 Cc_{i-1,j,w}^{n+1} + A_4 Cc_{i+1,j,w}^{n+1} + A_5 Cc_{i,j-1,w}^{n+1} + A_6 Cc_{i,j+1,w}^{n+1} + A_7 Cc_{i,j,w-1}^{n+1} + A_8 Cc_{i,j,w+1}^{n+1} = b_1 \quad (5.27)$$

Όπου $A_{n,n=1-8}$ είναι οι συντελεστές των αγνώστων συγκεντρώσεων $Cc_{i-1,j,w}^{n+1}$, και με την σειρά του $Cc_{i-1,j,w}^{n+1}$ είναι η συγκέντρωση της διαλυμένης αργίλου στο πορώδες στον επόμενο χρόνο n+1 στο κελί με διακριτικό θέσης $i-1, j, w$. Οι οχτώ συντελεστές $A_{n,n=1-8}$ δίδονται παρακάτω.

$$A_1 = L_c\, wm + \frac{1}{dt} + 2\frac{D_{cx}}{dx^2} + 2\frac{D_{cy}}{dy^2} + 2\frac{D_{cz}}{dz^2} \quad (5.28)$$

$$A_2 = \frac{\left(\frac{1}{dt} + L_{ca}\, wm\right)\rho}{\theta} \quad (5.29)$$

$$A_3 = -\frac{D_{cx}\, wm}{dx^2} - \frac{U\ wm}{2\, dx} \quad (5.30)$$

$$A_4 = -\frac{D_{cx}\, wm}{dx^2} + \frac{U\ wm}{2\, dx} \quad (5.31)$$

$$A_5 = -\frac{D_{cy}\, wm}{dy^2} \quad (5.32)$$

$$A_6 = -\frac{D_{cy}\, wm}{dy^2} \quad (5.33)$$

$$A_7 = -\frac{D_{cz}\, wm}{dz^2} \quad (5.34)$$

$$A_8 = -\frac{D_{cz}\, wm}{dz^2} \quad (5.35)$$

$$\begin{aligned} b_1 = & -\frac{Cc_{i,j,w}^{*n} L_c^*\, rd\ (1-wm)}{\theta} + \frac{Cc_{i,j,w}^{n}}{dt} + \frac{Cc_{i,j,w}^{*n}\, rd}{\theta\ dt} + Cc_{i+1,j,w}^{n}\left(\frac{D_{cx}\ (1-wm)}{dx^2} - \frac{U\ (1-wm)}{2\, dx}\right) + \\ & Cc_{i-1,j,w}^{n}\left(\frac{D_{cx}\ (1-wm)}{dx^2} + \frac{U\ (1-wm)}{2\, dx}\right) + \frac{Cc_{i,j-1,w}^{n}\, D_{cy}\ (1-wm)}{dy^2} + \frac{Cc_{i,j+1,w}^{n}\, D_{cy}\ (1-wm)}{dy^2} + \\ & Cc_{i,j,w}^{n}\left(-\left(L_c\ (1-\ wm)\right) - \frac{2\ D_{cx}\ (1-wm)}{dx^2} - \frac{2\ D_{cy}\ (1-wm)}{dy^2} - \frac{2\ D_{cz}\ (1-wm)}{dz^2}\right) + \frac{Cc_{i,j,w-1}^{n}\, D_{cz}\ (1-wm)}{dz^2} + \\ & \frac{Cc_{i,j,w+1}^{n}\, D_{cz}\ (1-wm)}{dz^2} + F_c \end{aligned} \quad (5.36)$$

Ενώ η δεύτερη ομάδα εξισώσεων αντιστοιχεί στη προσρόφηση και έχει την μορφή:

$$A_9 Cc_{i,j,w}^{n+1} + A_{10} Cc_{i,j,w}^{*\,n+1} = b_2 \quad (5.37)$$

Όπου :

$$A_9 = wm \frac{R_{c-c^*}\,\theta}{\rho} \quad (5.38)$$

$$A_{10} = -\frac{1}{dt} - (L_c^* + R_{c^*-c}) \quad (5.39)$$

$$b_2 = -\left(Cc_{i,j,w}^{*n}\left(\frac{1}{dt} - L_c^*(1 - wm) - R_{c^*-c}\,(1 - wm)\right)\right) - \frac{Cc_{i,j,w}^{n}\,R_{c-c^*}\,\theta\,(1-wm)}{\rho} \quad (5.40)$$

Στις εξ. (5.27-5.40) Cc είναι η αιωρούμενη άργιλος, Cc^* είναι η προσροφημένη άργιλος επάνω στο στερεό πορώδες, D είναι όροι υδροδυναμικής διασποράς οπότε D_{cx} είναι διασπορά του ιού στην διεύθυνση x, D_{cy} , D_{cz} όροι διασποράς στις άλλες δύο διευθύνσεις. Ενώ L είναι όροι ρυθμού αποδόμησης, συνεπώς L_c αποδόμηση αργίλου, L_c^* αποδόμηση προσροφημένης αργίλου επάνω στο στερεό πορώδες, R είναι ρυθμοί προσρόφησης συνεπώς R_{c^*-c} είναι ο ρυθμός με τον οποίο προσροφημένη άργιλος ξαναμετατρέπεται σε αιωρούμενη και R_{c-c^*} είναι ακριβώς το αντίθετο αυτού. Τέλος U είναι η ενδοπορώδης ταχύτητα του ρευστού στο οποίο γίνεται η συμμεταφορά και F_c είναι ο ρυθμός εισαγωγής αργίλου στο πορώδες.

2η ΕΠΙΛΥΣΗ

Σε αντίθεση με το πρώτο βήμα οι εξ. (3.15, 3.16) που περιγράφουν την δημιουργία των συμπλεγμάτων CcCvc και Cc*Cvc*, δεν έχουν μερικές παραγώγους αλλά μόνο μια παράγωγο ως προς το χρόνο.

Κάτω από κανονικές συνθήκες οι συνήθεις διαφορικές λύνονται πολύ εύκολα, ακόμα και χωρίς την χρήση αριθμητικών μεθόδων. Μια εκθετική αντικατάσταση θα ήταν υπέρ αρκετή ώστε να βρεθεί η αναλυτική λύση. Τώρα όμως έχουμε σύστημα μη γραμμικών εξισώσεων με επιπλέον κίνδυνο την εμφάνιση ακαμψίας (κεφ. 6). Οπότε για να περιορίσουμε την αρνητική επίδραση που έχει το φαινόμενο αυτό στην

πορεία της επίλυσης πρέπει να χρησιμοποιήσουμε υπονοούμενο σχήμα (Implicit Scheme, κεφ. 4). Που συνεπάγεται σχηματισμός μεγάλων συστημάτων εξισώσεων. Κάτι τέτοιο δεν το βλέπουμε για πρώτη φορά αλλά το ξανά χρησιμοποιήσαμε για να επιλύσουμε την απλή μεταφορά της αργίλου εξ. (5.25, 3.5). Όμως τώρα έχουμε μη γραμμικές εξισώσεις, και συστήματα με τέτοιες εξισώσεις δεν λύνονται εύκολα και μερικές φορές δεν λύνονται ποτέ. Η πιο συνηθισμένη μέθοδος για επίλυση αυτών συστημάτων είναι χρήση ελαχίστων τετραγώνων. Σε αυτή επιλέγεται μια αρχική προσέγγιση για την λύση, υπολογίζεται το σφάλμα που προκύπτει σαν άθροισμα των διαφορών στο τετράγωνο και στην συνέχεια γίνεται προσπάθεια ώστε να μειωθεί το συνολικό σφάλμα μεταβάλλοντας στοχευμένα (jacob matrix) την αρχική προσέγγιση. Όταν το συνολικό σφάλμα μειωθεί αρκετά ή δεν μπορεί να γίνει καλύτερη προσέγγιση για την λύση τότε η διαδικασία αυτή σταματά (http://en.wikipedia.org/wiki/Non-linear_least_squares). Η μέθοδος των ελαχίστων τετραγώνων παρ' ότι ξεπερνά το φαινόμενο της ακαμψίας, δυστυχώς έχει προβλήματα σύγκλισης. Ιδιαίτερα όταν εφαρμόζεται σε μη γραμμικές εξισώσεις ακόμα και να συγκλίνει είναι πολύ αργή. Η εφαρμογή της πρέπει να πραγματοποιείται μόνο όταν είναι τελείως απαραίτητη διαφορετικά υπάρχει κίνδυνος πολύ μεγάλης καθυστέρησης στους υπολογισμούς αν όχι διακοπή των πράξεων εξαιτίας της μη σύγκλισης.

Στις εξ. (3.15, 3.16) οι όροι της μορφής $\left(C^{*}_{vc_{eq}} - C^{*}_{vc}\right)^2$ αφαιρούν την γραμμικότητα και δυσχεραίνουν πάρα πολύ την επίλυση. Για να ξεπεραστεί το εμπόδιο αυτό χρησιμοποιήθηκαν συγκεκριμένες μέθοδοι. Η dodesol_rkm9mkn (intel ode solver subroutine) είναι μια εξειδικευμένη υπορουτίνα η οποία υλοποιεί επίλυση συστημάτων με συνήθεις διαφορικές χωρίς να γνωρίζει εκ των προτέρων την γραμμικότητα ή το μέγεθος της ακαμψίας του συστήματος. Ελέγχει σε κάθε βήμα την παρούσα δυσκαμψία και εναλλάσσει αυτόματα σε ρητό ή υπονοούμενο σχήμα όταν το κρίνει απαραίτητο. Στην περίπτωση που χρησιμοποιεί ρητό σχήμα εφαρμόζει την μέθοδο Merson's ενώ όταν εφαρμόζει υπονοούμενο σχήμα χρησιμοποιεί, L-stable (5,2) 4th order μέθοδο. Για πιο πολλές λεπτομέρειες σχετικά με την dodesol παρακαλώ ανατρέξτε στο εγχειρίδιο χρήσης intel ode solver manual (http://software.intel.com/en-us/articles/intel-ordinary-differential-equations-solver-library/).

Υπάρχει όμως ένα ακόμα πρόβλημα που πρέπει να ξεπεραστεί πριν ολοκληρωθεί η 2η ΕΠΙΛΥΣΗ. Οι εξισώσεις (3.17) και (3.21) λαμβάνουν υπόψη του μόνο την περίπτωση που η συγκέντρωση του εισαγόμενου ιού αυξάνεται ή είναι σταθερή. Αντίθετα αγνοεί την περίπτωση που ο ιός ελαττώνεται η έχει μηδενιστεί τελείως. Έτσι γενικεύονται οι εξ. (3.17, 3.21) και γίνονται εξ. (5.55, 5.42).

$$\Lambda_{v-v^*c^*} = +\frac{\rho_m}{\theta} r_{v-v^*c^*}\left(C^*_{vc_{eq}} - C^*_{vc}\right)^2 C^*_c \qquad \Delta C^*_v \geq 0$$

$$\Lambda_{v-v^*c^*} = -\frac{\rho_m}{\theta} r_{v-v^*c^*}\left(C^*_{vc_{eq}} - C^*_{vc}\right)^2 C^*_c \qquad \Delta C^*_v \prec 0 \tag{5.41}$$

$$\Lambda_{v-vc} = +r_{v-vc}\left(C_{vc_{eq}} - C_{vc}\right)^2 C_c \qquad \Delta C_v \geq 0$$

$$\Lambda_{v-vc} = -r_{v-vc}\left(C_{vc_{eq}} - C_{vc}\right)^2 C_c \qquad \Delta C_v \prec 0 \tag{5.42}$$

Τώρα χρησιμοποιώντας τις κατάλληλες μεθόδους και υπορουτίνες είναι δυνατή η λύση των εξισώσεων (3.15-3.22) μαζί με τις γενικεύσεις (5.41, 5.42) οπότε και παραλαμβάνονται οι συγκεντρώσεις των συμπλεγμάτων CcCvc και Cc*Cvc* .

<u>3η ΕΠΙΛΥΣΗ</u>

Στο τρίτο και τελευταίο βήμα θα επιλύσουμε την πιο σημαντική εξίσωση, την εξίσωση συμμεταφοράς εξ. (5.26). Μαζί με αυτήν βέβαια θα λυθεί και η εξίσωση προσρόφησης εξ. (3.23). Για να γίνει κάτι τέτοιο εφαρμόζουμε το κεντρικό σχήμα (central scheme) τόσο για τον όρο της διασποράς όσο και για το όρο της μεταγωγής. Μετατρέπουμε με αυτόν τον τρόπο όλες τις παραγώγους σε πεπερασμένες διαφορές. Τελικά φτάνουμε σε δυο ομάδες εξισώσεων που πρέπει να λυθούν μαζί. Η πρώτη αντιστοιχεί στην εξίσωση συμμεταφοράς και έχει την μορφή εξ. (5.43).

$$A_1 Cv^{n+1}_{i,j,w} + A_2 Cv^{*\,n+1}_{i,j,w} + A_3 Cv^{n+1}_{i-1,j,w} + A_4 Cv^{n+1}_{i+1,j,w} + A_5 Cv^{n+1}_{i,j-1,w} + A_6 Cv^{n+1}_{i,j+1,w} + A_7 Cv^{n+1}_{i,j,w-1} + A_8 Cv^{n+1}_{i,j,w+1} = b_1 \tag{5.43}$$

Όπου $A_{n,n=1-8}$ είναι οι συντελεστές των αγνώστων συγκεντρώσεων $C^{n+1}_{i-1,j,w}$, και με την σειρά του $C^{n+1}_{i-1,j,w}$ είναι η συγκέντρωση του διαλυμένου ιού στο πορώδες στον επόμενο χρόνο n+1 στο κελί με διακριτικό θέσης $i-1, j, w$.

Οι οχτώ συντελεστές $A_{n,n=1-8}$ δίδονται παρακάτω.

$$A_1 = L_v\ wm + \frac{1}{dt} + 2\frac{D_{vx}}{dx^2} + 2\frac{D_{vy}}{dy^2} + 2\frac{D_{vz}}{dz^2} \tag{5.44}$$

$$A_2 = \frac{\left(\frac{1}{dt} + L_{va}\ wm\right)\rho}{\theta} \tag{5.45}$$

$$A_3 = -\frac{D_{vx}\ wm}{dx^2} - \frac{U\ wm}{2\ dx} \tag{5.46}$$

$$A_4 = -\frac{D_{vx}\ wm}{dx^2} + \frac{U\ wm}{2\ dx} \tag{5.47}$$

$$A_5 = -\frac{D_{vy}\ wm}{dy^2} \tag{5.48}$$

$$A_6 = -\frac{D_{vy}\ wm}{dy^2} \tag{5.49}$$

$$A_7 = -\frac{D_{vz}\ wm}{dz^2} \tag{5.50}$$

$$A_8 = -\frac{D_{vz}\ wm}{dz^2} \tag{5.51}$$

$$\begin{aligned} b_1 = & -\frac{C^{*n}_{i,j,w}\ L_{va}\ \rho\ (1-wm)}{\theta} - \frac{BB^{n}_{i,j,w}\ L^{*}_{vc}\ \rho\ (1-wm)}{\theta} - \frac{BB^{n+1}_{i,j,w}\ L^{*}_{vc}\ \rho\ wm}{\theta} + \frac{C^{*n}_{i,j,w}\ \rho}{\theta\ dt} - \\ & \frac{-AA^{n}_{i,j,w} + AA^{(*n)p1}_{i,j,w}}{dt} - \frac{\rho\ \left(-BB^{n}_{i,j,w} + BB^{n+1}_{i,j,w}\right)}{\theta\ dt} + AA^{n}_{i+1,j,w}\ \left(\frac{D_{vcx}\ (1-wm)}{dx^2} - \frac{U\ (1-wm)}{2\ dx}\right) + \\ & C^{n}_{i+1,j,w}\ \left(\frac{D_{vx}\ (1-wm)}{dx^2} - \frac{U\ (1-wm)}{2\ dx}\right) + AA^{n}_{i-1,j,w}\ \left(\frac{D_{vcx}\ (1-wm)}{dx^2} + \frac{U\ (1-wm)}{2\ dx}\right) + \\ & C^{n}_{i-1,j,w}\ \left(\frac{D_{vx}\ (1-wm)}{dx^2} + \frac{U\ (1-wm)}{2\ dx}\right) + \frac{AA^{n+1}_{i-1,j,w}\ D_{vcx}\ wm}{dx^2} + \frac{AA^{n+1}_{i+1,j,w}\ D_{vcx}\ wm}{dx^2} - \\ & \frac{\left(-AA^{n+1}_{i-1,j,w} + AA^{n+1}_{i+1,j,w}\right)\ U\ wm}{2\ dx} + \frac{AA^{n}_{i,j-1,w}\ D_{vcy}\ (1-wm)}{dy^2} + \frac{AA^{n}_{i,j+1,w}\ D_{vcy}\ (1-wm)}{dy^2} + \\ & \frac{C^{n}_{i,j-1,w}\ Dvy\ (1-wm)}{dy^2} + \frac{C^{n}_{i,j+1,w}\ Dvy\ (1-wm)}{dy^2} + \frac{AA^{n+1}_{i,j-1,w}\ D_{vcy}\ wm}{dy^2} + \frac{AA^{n+1}_{i,j+1,w}\ D_{vcy}\ wm}{dy^2} + \\ & AA^{n}_{i,j,w}\ \left(-\left(L_{vc}\ (1\ -\ wm)\right) - \frac{2\ D_{vcx}\ (1-wm)}{dx^2} - \frac{2\ D_{vcy}\ (1-wm)}{dy^2} - \frac{2\ D_{vcz}\ (1-wm)}{dz^2}\right) + \\ & C^{n}_{i,j,w}\ \left(-\left(L_{v}\ (1\ -\ wm)\right) + \frac{1}{dt} - \frac{2\ D_{vx}\ (1-wm)}{dx^2} - \frac{2\ Dvy\ (1-wm)}{dy^2} - \frac{2\ Dvz\ (1-wm)}{dz^2}\right) + \\ & AA^{*n+1}_{i,j,w}\ \left(-(L_{vc}\ wm) - \frac{2\ D_{vcx}\ wm}{dx^2} - \frac{2\ D_{vcy}\ wm}{dy^2} - \frac{2\ D_{vcz}\ wm}{dz^2}\right) + \frac{AA^{n}_{i,j,w-1}\ D_{vcz}\ (1-wm)}{dz^2} + \\ & \frac{AA^{n}_{i,j,w+1}\ D_{vcz}\ (1-wm)}{dz^2} + \frac{C^{n}_{i,j,w-1}\ D_{vz}\ (1-wm)}{dz^2} + \frac{C^{n}_{i,j,w+1}\ Dvz\ (1-wm)}{dz^2} + \frac{AA^{n+1}_{i,j,w-1}\ D_{vcz}\ wm}{dz^2} + \\ & \frac{AA^{n+1}_{i,j,w+1}\ D_{vcz}\ wm}{dz^2} \end{aligned} \tag{5.52}$$

Ενώ η δεύτερη ομάδα εξισώσεων αντιστοιχεί στη προσρόφηση και έχει την μορφή:

$$A_9 C\nu_{i,j,w}^{n+1} + A_{10} C\nu_{i,j,w}^{*\,n+1} = b_2 \tag{5.53}$$

Όπου :

$$A_9 = wm\ \frac{R_{\nu-\nu^*}\theta}{\rho} \tag{5.54}$$

$$A_{10} = -\frac{1}{dt} - (\ L_\nu^* + R_{\nu^*-\nu}) \tag{5.55}$$

$$b_2 = \left(C\nu_{i,j,w}^{*n} \left(\frac{1}{dt} - L_\nu^* (1 - wm) - R_{\nu^*-\nu} (1 - wm) \right) \right) - \frac{C\nu_{i,j,w}^{*n}\ R_{\nu-\nu^*}\ \theta\ (1 - wm)}{\rho} \tag{5.56}$$

Στις εξ. 5.43-5.56 ισχύει ότι, $C\nu^*$ είναι ό προσροφημένος ιός επάνω στο στερεό πορώδες, D είναι όροι υδροδυναμικής διασποράς οπότε $D_{\nu x}$ είναι διασπορά του ιού στην διεύθυνση x, $D_{\nu y}$, $D_{\nu z}$ όροι διασποράς στις άλλες δύο διευθύνσεις, $D_{\nu cx}$ είναι διασπορά του προσροφημένου ιού στην διεύθυνση x, $D_{\nu cy}$, $D_{\nu cz}$ όροι διασποράς στις άλλες δύο διευθύνσεις. Ενώ L είναι όροι ρυθμού αποδόμησης, συνεπώς L_ν αποδόμηση ιού, L_ν^* αποδόμηση προσροφημένου ιού επάνω στο στερεό πορώδες, ομοίως L_{vc} και L_{vc}^* όροι αποδόμησης του συμπλέγματος ιού προσροφημένου σε άργιλο και συνεχεία προσροφημένου στο στερεό πορώδες αντίστοιχα, R είναι ρυθμοί προσρόφησης συνεπώς $R_{\nu^*-\nu}$ είναι ο ρυθμός με τον οποίο προσροφημένος ιός ξαναμετατρέπεται σε αιωρούμενο και $R_{\nu-\nu^*}$ είναι ακριβώς το αντίθετο αυτού. Τέλος οι όροι AA, BB αντιστοιχούν σε $AA = CcC_{vc}$ και $BB = Cc^*C_{vc}^*$, με C_{vc} να είναι το σύμπλοκο προσροφημένου ιού επάνω στην άργιλο ενώ C_{vc}^* να είναι η προσροφημένη μορφή του επάνω στο στερεό πορώδες.

Χρησιμοποιώντας τις συνοριακές και τις αρχικές συνθήκες (5.3)-(5.10) σε συνδυασμό με τις ομάδες εξισώσεων μεταφοράς εξ. (5.43) και προσρόφησης εξ. (5.53) αλλά και εισάγοντας τα αποτελέσματα C_{vc}, C_{vc}^*, Cc και Cc^* από τα δυο προηγούμενα βήματα (1η επίλυση, 2η επίλυση) σχηματίζεται ένα σύστημα εξισώσεων, με 2 nx*ny*nz αγνώστους (nx, ny, nz αριθμός κελιών στις 3 διαστάσεις αντίστοιχα). Το οποίο έχει πίνακα (μητρώο) της μορφής Σχήμα 4.9δ και λύνεται εύκολα με την μέθοδο LU Decomposition (κεφ. 4.3.2.1). Το αποτέλεσμα αυτής είναι η παραλαβή όλων των συγκεντρώσεων του ιού σε διακριτές χωρικές θέσεις και συγκεκριμένες χρονικές στιγμές που επιβάλλονται από το χωρικό και χρονικό βήμα που έχει χρησιμοποιηθεί, αντίστοιχα.

6. ΣΤΑΘΕΡΟΤΗΤΑ ΚΑΙ ΑΚΡΙΒΕΙΑ

6.1 ΑΚΑΜΨΙΑ ΚΑΙ ΜΗ ΓΡΑΜΜΙΚΟΤΗΤΑ

Δυστυχώς δεν είναι δυνατό για όλες τι μεθόδους να συγκλίνουν πάντα ή να έχουν ακριβή αποτελέσματα. Υπάρχουν πολύ παράγοντες οι οποίοι δυσχεραίνουν τις επιλύσεις ή τις καθιστούν τελείως αδύνατες. Μια αιτία είναι η γραμμικότητα. Εάν δεν υπάρχει αυτή στις εξισώσεις αλλά αντίθετα υπάρχουν όροι όπως τετράγωνα ή ημίτονα ή γινόμενα μεταξύ των άγνωστων μεταβλητών τότε η εύρεση της λύσης δεν είναι απλή υπόθεση. Αυτού του είδους οι εξισώσεις δεν έχουν μια και μοναδική λύση αλλά ένα σύνολο λύσεων που κάθε φορά μεταβάλλονται ανάλογα με τις οριακές συνθήκες. Ιδιαίτερα μη γραμμικά προβλήματα που περιέχουν τετράγωνα (όρους υψουμένους στην δευτέρα δύναμη) ή γενικότερα σε κάποια άρτια δύναμη αντιμετωπίζουν πολλά προβλήματα ως προς την σύγκλιση των λύσεων τους καθώς εμποδίζεται η απαιτούμενη αλλαγή προσήμου. Έτσι εάν για κάποιο λόγο κατά την διάρκεια της επίλυσης, η αριθμητική λύση πάρει σε κάποιο σημείο μια τιμή μεγαλύτερη από ότι θα έπρεπε, τότε είναι αδύνατο να επιστρέψει σε αποδεκτές τιμές εξαιτίας του τετραγώνου που κρατάει πάντα θετικό πρόσημο. Μια τέτοια περίπτωση είναι και η εξ. (6.1) η οποία έχει την μορφή των εξ. (3.15, 3.16) που περιγράφουν δημιουργία των συγκεντρώσεων C_{vc} και C^{*}_{vc} στο κεφ. 3. Η φυσική συνθήκη η οποία θα πρέπει να ισχύει είναι $C_{eq} \geq Y(t)$.

$$Y'(t) = \left(C_{eq} - Y(t)\right)^2 - 0.1 * Y(t) + 10 \qquad (6.1)$$

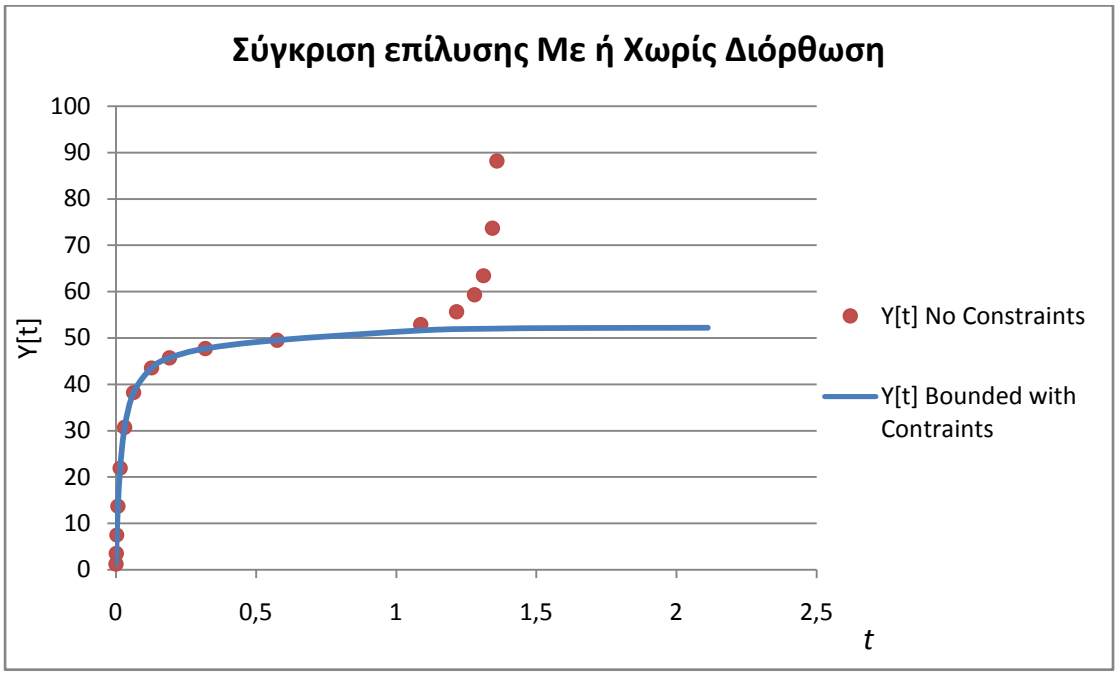
$$Y(0) = 0, \quad C_{eq} = 50 \qquad (6.2)$$

Κοιτάζοντας τις αρχικές συνθήκες εξ. (6.2) παρατηρείται ότι για t=0 το $Y'(t) > 0$ που σημαίνει ότι αφού υπάρχει θετική παράγωγο η τιμή της $Y(t)$ θα αυξάνεται. Καθώς όμως περνάει ο χρόνος ο όρος $\left(C_{eq} - Y(t)\right)^2$ μειώνεται όλο και περισσότερο με αποτέλεσμα ο ρυθμός αύξησης του $Y(t)$ να μειώνεται. Αυτή η διαδικασία θα συνεχιστεί μέχρι το $Y'(t)$ να πάρει τιμή 0. Τότε θα σταματήσει η αύξηση του $Y(t)$, θα διατηρηθεί η τρέχουσα τιμή του στο χρόνο και σε κάθε περίπτωση το $C_{eq} > Y(t)$ (Σχήμα 6.1 μπλε γραμμή).

Τι γίνεται όμως όταν λόγο των αριθμητικών σφαλμάτων το $Y(t)$ γίνει μεγαλύτερο του C_{eq}. Τότε αντί η $Y'(t)$ να διατηρήσει μηδενική τελική τιμή όπως θα όφειλε θα ξαναποκτήσει θετική και μάλιστα με απότομη αύξουσα πορεία εξαιτίας του όρου $\left(C_{eq} - Y(t)\right)^2$. Αποτέλεσμα αυτού είναι όχι μόνο προσωρινά να παραβιασθεί η φυσική συνθήκη $C_{eq} \geq Y(t)$ αλλά να χαθεί τελείως η σύγκλιση του προβλήματος (Σχήμα 6.1 κόκκινοι κύκλοι). Εάν το τετράγωνο μέσα στην εξ. (6.1) δεν κρατούσε πάντα το πρόσημο θετικό, τότε δεν θα υπήρχε τέτοιος κίνδυνος. Μια κατά λάθος υπέρβαση του $Y(t)$ πάνω από το C_{eq} θα οδηγούσε σε αρνητικό πρόσημο την σχέση $\left(C_{eq} - Y(t)\right)$, θα είχαμε αρνητική παράγωγο $Y'(t) < 0$ και φθίνουσα πορεία του $Y(t)$.

Φαίνεται λοιπόν πόσο καταστρεπτικές συνέπειες μπορεί να έχει μια άρτια δύναμη σε μια αριθμητική λύση.

Για να αποφευχθούν τα παραπάνω προβλήματα συχνά στην βιβλιογραφία χρησιμοποιούνται ειδικές ρουτίνες οι οποίες φροντίζουν να κάνουν εξαιρετικά μικρά βήματα σε κρίσιμες περιοχές ή να θέτουν περιορισμούς στο πόσο μεγάλες μπορούν να γίνουν οι τιμές των μεταβλητών.



Σχήμα 6.1: Παρουσιάζονται δυο επιλύσεις της εξ. $Y'(t) = \left(C_{eq} - Y(t)\right)^2 - 0.1Y(t) + 10$. Η συνεχόμενη μπλε γραμμή έχει προέλθει φροντίζοντας να αλλάζει η αρχική εξίσωση σε $Y'(t) = -\left(C_{eq} - Y(t)\right)^2 - 0.1Y(t) + 10$ όταν το $Y(t) > C_{eq}$. Αντίθετα οι κόκκινες κουκίδες δείχνουν τι συμβαίνει όταν κατά λάθος εξαιτίας των αριθμητικών σφαλμάτων το $Y(t)$ ξεπερνάει σε ένα τυχαίο σημείο το C_{eq}. Η σύγκλιση χάνεται και η φυσική συνθήκη $C_{eq} \geq Y(t)$ παραβιάζεται.

Εκτός όμως από την γραμμικότητα ένας δεύτερος παράγοντας που δυσχεραίνει την αριθμητική επίλυση είναι η ακαμψία των εξισώσεων (equations stiffness). Αν και δεν υπάρχει ακριβής ορισμός για το φαινόμενο αυτό, θεωρούμε ότι μια εξίσωση έχει ακαμψία όταν ισχύει ένα από τα παρακάτω:

I. Περιέχει πολλές διαφορετικές χρονικές κλίμακες. Δηλαδή ορισμένοι όροι της π.χ. αποσύνθεση, αλλάζουν-φθίνουν πολύ πιο γρήγορα από τους άλλους.
II. Στην αριθμητική λύση της το βήμα υπαγορεύεται από τις απαιτήσεις σταθερότητας και όχι από τις απαιτήσεις ακρίβειας.
III. Εάν η εφαρμογή ρητής μεθόδου σε αυτή δεν είναι αποδοτική ή λειτουργεί πολύ αργά και έτσι κρίνεται απαραίτητη η χρήση υπονοούμενου σχήματος

IV. Το σύνολο των ιδιοτιμών της, έχουν αρνητικό πραγματικό μέρος, και ο δείκτης δυσκαμψίας (ο λόγος του πραγματικού μέρους της μεγαλύτερης ιδιοτιμής προς το πραγματικό μέρος της μικρότερης) είναι μεγάλος.

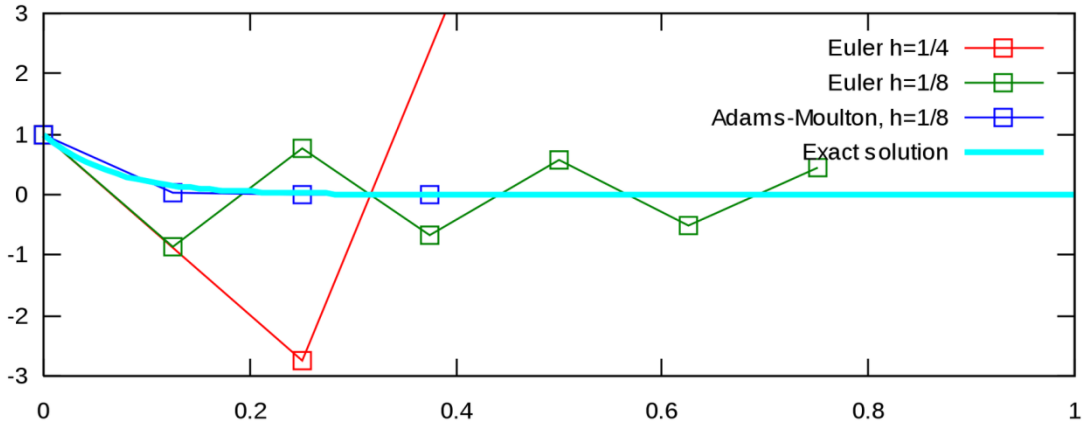
V. Γενικότερα μια εξίσωση (ή σύνολο εξισώσεων) έχει δυσκαμψία όταν οι ιδιοτιμές του μητρώου Jacob που σχηματίζεται από αυτήν, έχει ιδιοτιμές που διαφέρουν πολύ μεταξύ του σε μέγεθος.

Εξισώσεις που έχουν ακαμψία μέσα τους προκύπτουν σε πολλές πρακτικές εφαρμογές, όπως μοντελοποίηση χημικών διεργασιών ή ανάλυση ηλεκτρικών κυκλωμάτων. Η συνέπειες του φαινομένου αυτού μεγεθύνονται σε μεγάλο βαθμό όταν γίνεται ταυτόχρονη επίλυση πολλών εξισώσεων. Σε αυτήν την περίπτωση το βήμα της αριθμητικής ανάλυσης μπορεί να επιβάλλεται όχι από την εξίσωση με την πιο σημαντική επίδραση στο τελικό αποτέλεσμα αλλά από αυτήν με την πιο γρήγορη μεταβολή. Δηλαδή θα μπορούσε μια εξίσωση να μην διαδραματίζει σημαντικό ρόλο στο αποτέλεσμα, παρ' όλα αυτά να είναι η συγκεκριμένη η οποία θα επιβάλει το βήμα της συνολικής επίλυσης. Τα ίδια ισχύουν και για το θέμα ευστάθειας σύγκλισης. Μπορεί μια εξίσωση να μην έχει προβλήματα σταθερότητας όταν λύνεται μόνη της, αλλά όταν επιλύεται μέσα σε σύστημα τότε να προκύπτει μεγάλη αστάθεια και να χρειάζεται να χρησιμοποιηθεί πολύ μικρό βήμα για να αποφευχθεί η αστοχία. Οι συνέπειες της ακαμψίας δεν είναι πάντα ότι χρειαζόμαστε πιο λεπτή διακριτοποίηση για να αποφύγουμε την αστοχία άρα χρειαζόμαστε πιο πολύ υπολογιστικό χρόνο. Στην πραγματικότητα υπάρχει περίπτωση όσο και μικρό βήμα να πάρουμε η σύγκλιση να μη έρθει ποτέ ή ακόμα και να έρθει, αυτή να οδηγεί σε λάθος λύση. Παράδειγμα συστήματος με μεγάλη ακαμψία είναι οι εξ. (6.3-6.5) που περιγράφουν χημική αντίδραση Robertson (Robertson chemical reaction). Παρατηρούνται πολύ μεγάλες διαφορές στο μέγεθος των συντελεστών των αγνώστων.

$$y_1^{'} = -0.04\, y_1 + 10^4 y_2\, y_3 \tag{6.3}$$
$$y_2^{'} = \ \ 0.04\, y_1 - 10^4 y_2\, y_3 - \ 3 * 10^7 y_2^2 \tag{6.4}$$
$$y_3^{'} = \ \ 3 * 10^7 y_2^2 \tag{6.5}$$

Ομοίως η εξ. (6.6) έχει πολύ μικρές κλίσεις (Σχήμα 6.2 κυανή γραμμή) και λογικά λίγα βήματα θα ήταν αρκετά για την επίλυση της. Παρ' όλα αυτά εξαιτίας της ακαμψίας χρειάζεται αρκετή προσπάθεια ώστε να βρεθεί το σωστό αποτέλεσμα Σχήμα 6.2 (http://en.wikipedia.org/wiki/Stiff_equation).

$$y^{'}(t) = -\lambda y(t) \tag{6.6}$$



Σχήμα 6.2: Παρουσιάζονται διαφορετικές επιλύσεις της εξ. (6.6) για λ=-15, μαζί με την αρχική συνθήκη $y(0) = 1$. Παρ' ότι η κλίση της ακριβής λύσης είναι πολύ μικρή, δυο από τις τρείς προσπάθειες επίλυσης με ρητές μεθόδους απέτυχαν. Η μέθοδος Euler βασίζεται στην σχέση $y_{n+1} = y_n + hf(t\_n\ , y\_n)$ ενώ η Adams-Moulton στην $y_{n+1} = y_n + \frac{1}{2}\, h\big(f(t_n\ , y_n) + f(t_{n+1}\ , y_{n+1})\big)$.

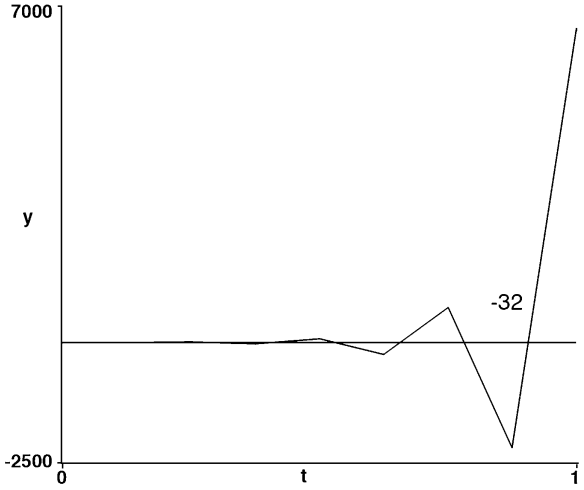
Βιβλιογραφικά στο παρακάτω πίνακα 6.1 φαίνονται ένα σύνολο συνηθισμένων αριθμητικών μεθόδων που χρησιμοποιούνται για την επίλυση συνήθη διαφορικών εξισώσεων, και στην συνέχεια παρουσιάζονται μερικές από τις γραφικές παραστάσεις αυτών.

Πίνακας 6.1: Αλγόριθμοι γνωστών αριθμητικών μεθόδων, όπου m είναι τα βήματα που χρειάζονται σε κάθε r στάδιο ώστε να πραγματοποιηθεί ο υπολογισμός του επόμενου σημείου y_{n+1} και p είναι η τάξη του διαφορικού που προσομοιώνεται (Richard Palais and Robert Palais, 2003).

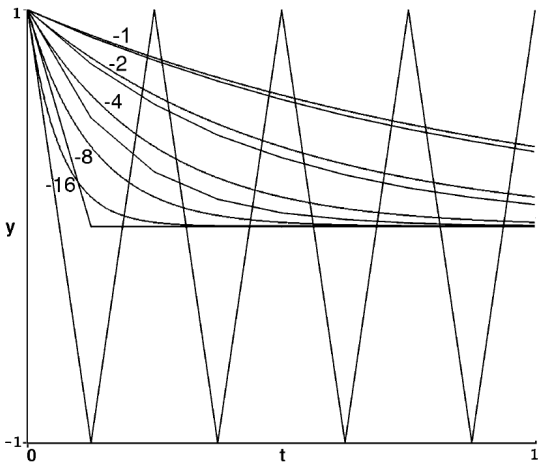
Method (order p)	$y_{n+1} =$ (m-steps, r-stages)
Euler (1)	$y_n + hf(t_n, y_n)$ $(1,1)$
Backward Euler*(1)	$y_n + hf(t_{n+1}, y_{n+1})$ $(1,1)$
Midpoint (2)	$y_n + hf(t_n + h/2, y_n + hf(t_n, y_n)/2)$ $(1,2)$
Leapfrog (2)	$y_{n-1} + 2hf(t_n, y_n)$ $(2,1)$
Trapezoidal*(2)	$y_n + h(f(t_n, y_n) + f(t_{n+1}, y_{n+1}))/2$ $(1,2)$
Heun (2)	$y_n + h(f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)))/2$ $(1,2)$
y-Midpoint*(2)	$y_n + hf(t_n + h/2, (y_n + y_{n+1})/2)$ $(1,2)$

*Implicit method

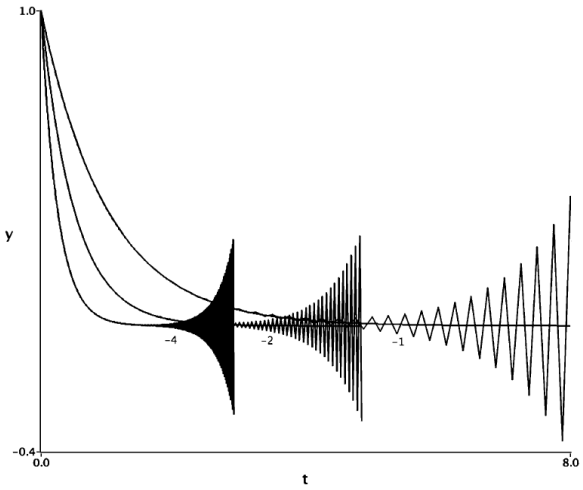
Στο Σχήμα 6.3 φαίνονται οι μέθοδοι του πίνακα 6.1 σε εφαρμογή. Χαρακτηριστική περίπτωση είναι η μέθοδος Leapfrog που ενώ καταφέρνει να προσομοιώσει την απότομη κλίση σωστά Σχήμα 6.3γ, αποτυγχάνει στην ευθεία εξαιτίας της ακαμψίας που υφίσταται στην εξίσωση προς επίλυση εξ.(6.6).



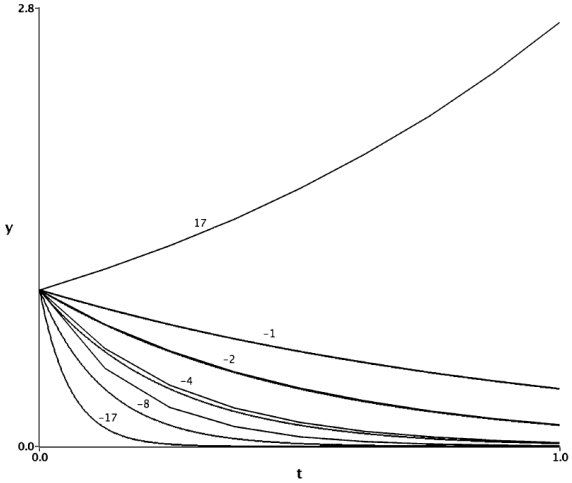
α) Μέθοδος Euler: Αστάθεια



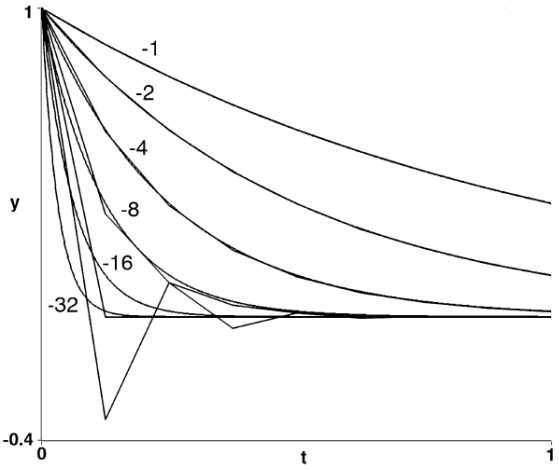
β) Μέθοδος Euler: Ευστάθεια-Αστάθεια



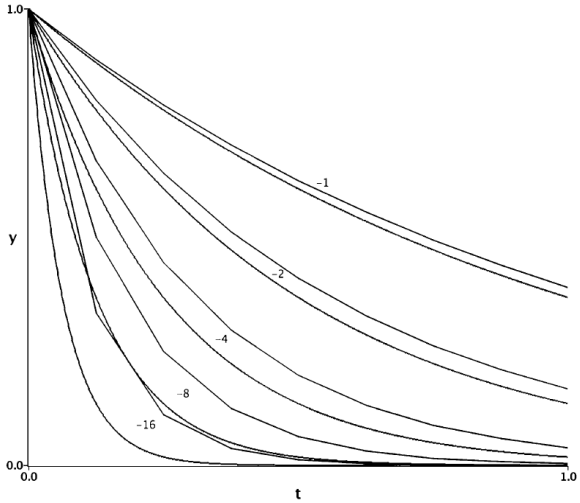
γ) Μέθοδος Leapfrog: Αστάθεια



δ) Μέθοδος Midpoint: Ευστάθεια



ε) Μέθοδος Trapezoidal: Αστάθεια



ζ) Μέθοδος Backward Euler: Ευστάθεια

Σχήμα 6.3: Παρουσιάζονται διαφορετικές μέθοδοι σε καταστάσεις ευστάθειας και αστάθειας. Σε μικρές κλίσεις φαίνονται να αντεπεξέρχονται αποτελεσματικά αλλά όταν τα γραφήματα γίνονται πιο απότομα τότε αποτυγχάνουν. Η εξίσωση που λύνεται είναι η $y' = \lambda y$, $y(0) = 1$, με $\lambda = -1, -2, -4, -8, -16, -32$.

6.2 ΠΕΡΙΟΡΙΣΜΟΙ ΣΤΙΣ ΕΞΙΣΩΣΕΙΣ ΜΕΤΑΦΟΡΑΣ

Οι μέθοδοι που χρησιμοποιήθηκαν ώστε να λυθούν οι εξισώσεις μεταφοράς αλλά συμμεταφοράς κεφ. 5, επιλέχθηκαν με το κριτήριο της ταχύτητας αλλά και της ευστάθειας. Παρακάτω δίδονται μερικοί περιορισμοί από την βιβλιογραφία που διασφαλίζουν την σύγκλιση.

Ρητό σχήμα (Explicit scheme)

Το ρητό σχήμα όταν χρησιμοποιείται για το όρο της διασποράς συγκλίνει εάν ισχύει η εξίσωση 6.7. Διαφορετικά έχουμε το φαινόμενο που παρατηρείται στο Σχήμα 6.4 (Zheng and Wang ,1999):

$$\Delta t \leq 0.5 \frac{R}{\frac{D_{xx}}{\Delta_x^2}+\frac{D_{yy}}{\Delta_y^2}+\frac{D_{zz}}{\Delta_Z^2}} \tag{6.7}$$

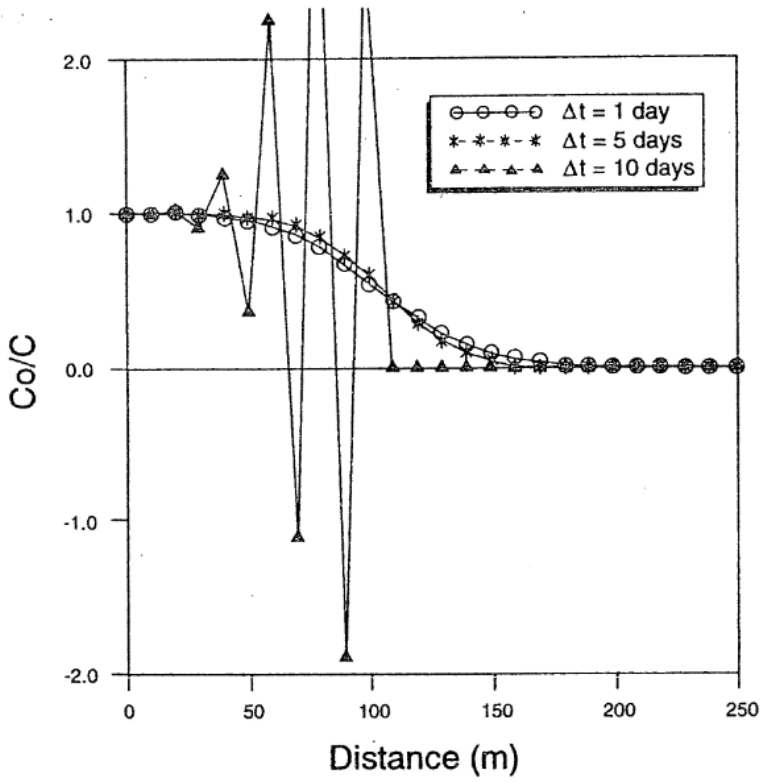
Το ρητό σχήμα σε συνδυασμό με την ανάντη μέθοδο (upstream scheme), όταν χρησιμοποιείται για τον όρο της μεταγωγής συγκλίνει εάν ισχύει η εξ. (6.8):

$$\Delta t \leq \frac{R}{\left|\frac{v_x}{\Delta_x}\right|+\left|\frac{v_y}{\Delta_y}\right|+\left|\frac{v_z}{\Delta_z}\right|} \tag{6.8}$$

(Zheng and Wang ,1999)

Σε κάποιες περιπτώσεις δεν έχουμε καμία σύγκλιση όσο μικρό και να γίνει το χρονικό βήμα Δt. Μια από αυτές είναι το ρητό σχήμα σε συνδυασμό με κεντρικές διαφορές όταν χρησιμοποιείται για να επιλυθεί ο όρος της μεταγωγής (Zheng and Bennett, 1995).

Στις εξισώσεις (6.7) και (6.8) ισχύουν ότι: Δt είναι το χρονικό βήμα, v_x είναι η ενδοπορώδης ταχύτητα στην διεύθυνση x, D_{xx} είναι ο συντελεστής υδροδυναμικής διασποράς στην διεύθυνση x, R είναι ο συντελεστής επιβράδυνσης (retardation factor εξ.1.15).



Impact of the time step size on the explicit finite-difference solution.

Σχήμα 6.4: Παρουσιάζεται η επίπτωση που έχει το χρονικό βήμα στην μονοδιάστατη εξίσωση μεταφοράς. Παρατηρείται ότι για Δt=10 το σύστημα γίνεται ασταθές και δεν υπάρχει σύγκλιση (Zheng and Bennett, 1995).

Υπονοούμενο Σχήμα (Implicit scheme)

Σε αντίθεση με το ρητό το υπονοούμενο σχήμα δεν έχει προβλήματα σύγκλισης. Παρόλα αυτά αντιμετωπίζει προβλήματα ακρίβειας, τα οποία ξεπερνιούνται για την περίπτωση μονοδιάστατης μεταφοράς όταν ισχύουν οι εξ. (6.9, 6.10) (Zheng and Bennett, 1995):

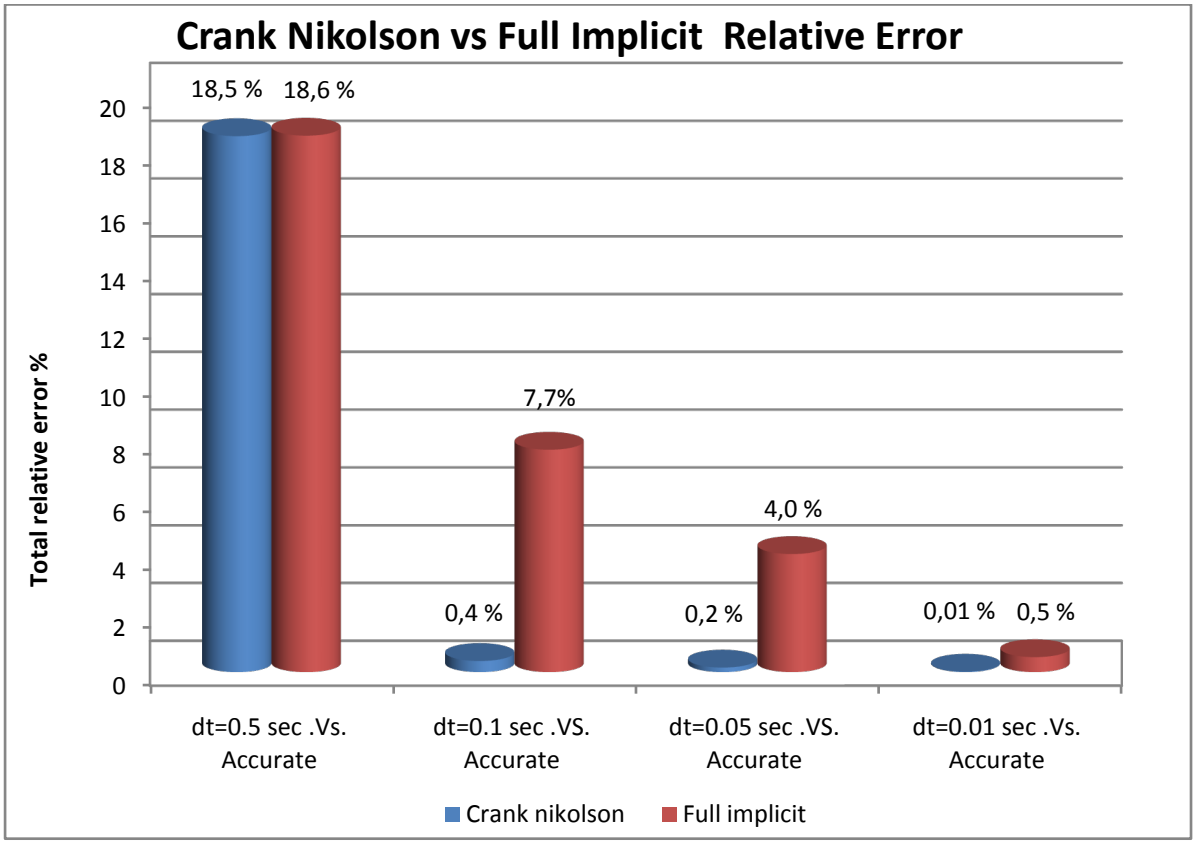
$$P_e = \frac{U\Delta x}{D} \leq 2 \qquad (6.9)$$

$$C_r = \frac{U\,\Delta t}{\Delta x} \leq 1 \qquad (6.10)$$

Η εξίσωση (6.9) διασφαλίζεται με πυκνή διακριτοποίηση του χώρου (μικρό μήκος κελιού Δx). Ενώ με την σειρά της η εξ. (6.10) ικανοποιείται όταν έχουμε μικρό χρονικό βήμα Δt.

Crank-Nicolson Σχήμα

Στο κεφ. 4.3.1.2 ορίσαμε μια ενδιάμεση μέθοδο μεταξύ του ρητού και του υπονοούμενου σχήματος. Αυτή ονομάστηκε Crank-Nicolson. Η σύγκλιση της για την περίπτωση απλής μεταφοράς πηγάζει από το υπονοούμενο σχήμα και ως εκ τούτου είναι εξασφαλισμένη (Zheng and Bennett, 1995). Η χρησιμότητα αυτής της μεθόδου έγκειται στο ότι θυσιάζοντας ευστάθεια αυξάνουμε αρκετά την ακρίβεια. Συνεπώς με την ίδια διακριτοποίηση εάν υπάρχει σύγκλιση είναι πιθανό να έχουμε καλύτερη ακρίβεια από ότι θα είχαμε στο πλήρως υπονοούμενο σχήμα. Η παγίδα στην χρήση της μεθόδου αυτής είναι ότι το εύρος διακριτοποίησης όπου εξασφαλίζεται η αποτελεσματικότητα της είναι πολύ μικρό. Με αποτέλεσμα τα επιπλέον θετικά χαρακτηριστικά που προσφέρει να μην μπορούν να χρησιμοποιηθούν και αντίθετα τα αρνητικά της να υπερισχύουν. Παράδειγμα αυτού είναι το Σχήμα 6.5



Σχήμα 6.5: Παρουσιάζονται τα σχετικά σφάλματα μεταξύ της ακριβής λύσης και δυο αριθμητικών μεθόδων: α) Του ρητού σχήματος και β) Του σχήματος Crank-Nicolson. Το πρόβλημα που επιλύθηκε αναφέρεται σε συμμεταφορά χρησιμοποιώντας ίδια διακριτοποίηση στο χώρο και ίδιες φυσικές παραμέτρους. Το συνολικό σχετικό σφάλμα υπολογίστηκε σαν ο μέσος όρος των επιμέρους σχετικών σφαλμάτων όλων το κελιών του αριθμητικού μοντέλου. Με το σχετικό σφάλμα δίνεται από την σχέση $e = \frac{C_{num} - C_{acc}}{C_{acc}}$.

Στο Σχήμα 6.5 φαίνεται μια σύγκριση ως προς την ακρίβεια μεταξύ του πλήρως υπονοούμενου σχήματος (Implicit Scheme) και της μεθόδου Crank-Nicolson. Όπως παρατηρείται το σφάλμα είναι το ίδιο για διακριτοποίηση dt=0.5 sec, αλλά καθώς το dt παίρνει όλο και μικρότερες τιμές το σφάλμα μειώνεται πολύ πιο γρήγορα στην Crank-Nicolson από ότι στην Full Implicit. Η περιοχή διακριτοποίησης που πραγματικά συμφέρει να χρησιμοποιήσουμε Crank-Nicolson είναι μεταξύ dt=0.5 sec και dt=0.1 sec. πέρα από το διάστημα αυτό το σχετικό σφάλμα μεταξύ της ακριβής λύσης και της αριθμητικής είναι το ίδιο και για τις δύο μεθόδους. Γι' αυτό χρειάζεται κανείς να εξετάζει προσεκτικά εάν αξίζει να εφαρμοστεί το σχήμα Crank Nicolson. Υπάρχει πολύ μεγάλη πιθανότητα να μειωθεί υπερβολικά η ευστάθεια της αριθμητικής λύσης, καθιστώντας την ευάλωτη σε μικρές αλλαγές των παραμέτρων εισόδου, χωρίς να δοθεί σαν αντάλλαγμα η αναμενόμενη επιπλέον ακρίβεια.

7. ΔΟΜΗ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ

7.1 ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΚΩΔΙΚΑ ΟΛΟΚΛΗΡΩΣΕΙΣ

Όπως αναφέρθηκε και στα προηγούμενα κεφάλαια το πρόβλημα της απλής μεταφοράς εξ. (5.1, 5.2) λύθηκε με δύο διαφορετικούς μεθόδους: α) χρησιμοποιώντας αναλυτικούς τύπους εξ. (4.4-4.18) και ολοκληρώνοντας τους αριθμητικά β) χρησιμοποιώντας αριθμητικούς μεθόδους κεφ. 5. Με αυτόν το τρόπο έγινε σύγκριση των αποτελεσμάτων των δύο μεθόδων και εξήχθησαν χρήσιμα συμπεράσματα για την αξιοπιστία της τελευταίας. Με σκοπό την υλοποίηση των παραπάνω επιλύσεων, γράφτηκαν στην γλώσσα προγραμματισμού Fortran (Intel fortran 11.1.0.61 with MKL 10.2, Visual studio 2008) οι απαραίτητες υπορουτίνες, δίνοντας μεγάλο βάρος τόσο στην ταχύτητα όσο και στην ακρίβεια τους.

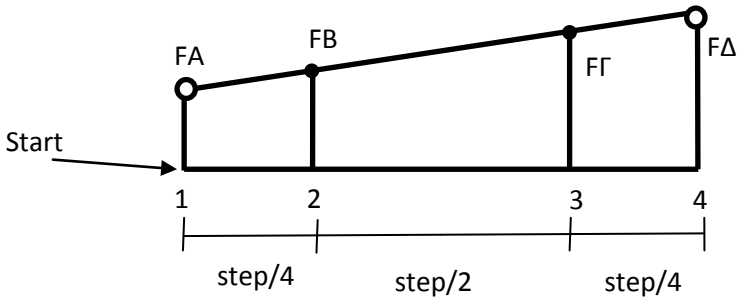
Η διαδικασία που ακολουθήθηκε βασίζεται στην αριθμητική ολοκλήρωση οπού στοιχειώδες επιφάνεια θεωρείται το τραπέζιο με ανοικτά άκρα. Το κυριότερο πρόβλημα που αντιμετωπίζουν οι αναλυτικές λύσεις θεωρείται ο χρόνος. Οι εξ. (4.1)-(4.4) περιέχουν μέσα τους πολλαπλά ολοκληρώματα $\iiint$, με αποτέλεσμα για να υπολογιστεί μια τιμή για την συνάρτηση του εξωτερικού ολοκληρώματος να πρέπει να γίνουν δύο και τρεις ακόμα διαδοχικές ολοκληρώσεις. Αυτή η διαδικασία είναι εξαιρετικά χρονοβόρα και η επιλογή του σχήματος ολοκλήρωσης είναι καθοριστική για την ακρίβεια και την αποτελεσματικότητα των πράξεων.

Το σχήμα ολοκλήρωσης για την παρούσα εργασία επιλέχθηκε να είναι "Τραπέζιο με ανοικτά άκρα" Σχήμα 7.1. Χαρακτηριστικά αυτού είναι ότι μπορεί να υπολογίζει επιφάνειες που έχουν ολοκληρώσιμες ασυνέχειες. Παρατηρείστε την εξίσωση 4.1, για t=0 δεν μπορεί να υπολογιστεί η τιμή του εσωτερικού ολοκληρώματος εξαιτίας του όρου $\frac{\Lambda_2(\tau)}{\zeta^{\frac{3}{2}}}$ ο οποίος τείνει στο άπειρο για $\zeta \dashrightarrow 0$. Επιπλέον η μέθοδος αυτή παρέχει την δυνατότητα ώστε οι υπολογισμοί που γίνονται σε διαδοχικά στάδια να μην χάνονται αλλά να συνεργάζονται. Αυτό διευκολύνει την χρήση του προσαρμοζόμενου βήματος (adaptive step).

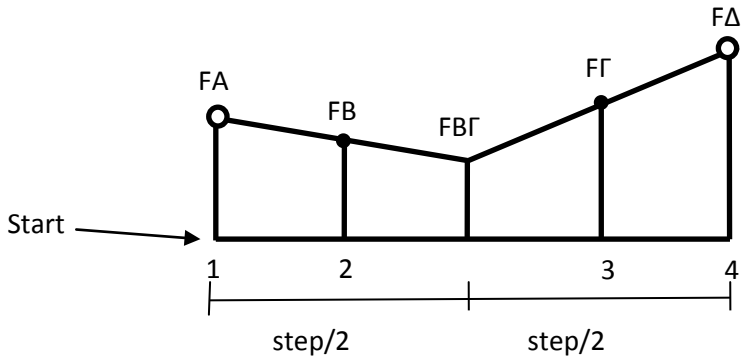
Πιο αναλυτικά :

1. θεωρείται αρχικό τυχαίο βήμα step.
2. Υπολογίζονται στο πρώτο στάδιο οι τιμές της συνάρτησης FB, FΓ στα σημεία step/4 και 3 step/4 αντίστοιχα.
3. Υπολογίζονται οι τιμές της συνάρτησης FAκαι FΔ στις θέσεις 0*step και 1*step αντίστοιχα, θεωρώντας ότι η ευθεία που συνδέει τα FA και FΔ περνάει από τα FB και FΓ.
4. Υπολογίζεται το εμβαδό Sum_1 όλου του τραπεζίου που έχει βάση μήκους step (τραπέζιο FA, FΔ, 1,2) Σχήμα 7.1, με βάση τον γνωστό τύπο: βάση μικρή + βάση μεγάλη επί ύψος δια δύο εξ. 7.1.
5. Σε δεύτερο στάδιο υπολογίζεται η τιμή της συνάρτησης FBΓ στο μέσο του διαστήματος step Σχήμα 7.2.
6. Υπολογίζονται τα σημεία FA και FΔ με βάση την λογική ότι πλέον υπάρχει μια ευθεία που ενώνει τα σημεία FA, FB, FBΓ και μια ξεχωριστή η οποία ενώνει τα FBΓ, FΓ, FΔ.
7. Υπολογίζεται πλέον το συνολικό εμβαδό του σχήματος με βάση $stepsum_2$ χρησιμοποιώντας την εξ. (7.2), Σχήμα 7.2.
8. Συγκρίνονται τα αποτελέσματα Sum_1 και Sum_2 με την εξ. (7.3):
    a. Εάν `division< 0.00001.and.division>=0.000001`,τότε το βήμα-step πού έχουμε επιλέξει είναι καλό και μπορούμε να προχωρήσουμε στην επόμενη περιοχή διατηρώντας το σταθερό.
    b. Εάν `division<0.000001D0 .or. abs(sum1)<1.D-18`,τότε το βήμα-step πού έχουμε επιλέξει είναι πολύ μικρό. Αυτό μπορεί να οφείλεται είτε επειδή η συνάρτηση που ολοκληρώνεται στην περιοχή αυτή είναι σταθερή, είτε επειδή το αποτέλεσμα της ολοκλήρωσης είναι πολύ μικρό σε σχέση με το τελικό αποτέλεσμα και δεν μας ενδιαφέρει να έχουμε καλύτερη ακρίβεια. Έτσι με το παρόν βήμα έχουμε μείωση της αποτελεσματικότητας του προγράμματος και καθυστερούμε χωρίς λόγο. Γι αυτό το επόμενο βήμα θα είναι μεγαλύτερο κατά 10% .
    c. Εάν `division> 0.00001` τότε η το βήμα μας είναι αρκετά μεγάλο και έχει μειωμένη ακρίβεια. Κρίνεται απαραίτητος ο υποδιπλασιασμός του βήματος και η επανάληψη των βημάτων από το βήμα 2 μέχρι το βήμα 8.

9. Υπολογίζουμε καινούριο Start =start+step, και ξεκινάμε την διαδικασία από το νούμερο 2.



Σχήμα 7.1: Παρουσιάζεται τραπέζιο με γνωστές τις τιμές FB και FΓ και άγνωστες τις FAκαι FΔ.



Σχήμα 7.2: Παρουσιάζεται τραπέζιο με γνωστές τις τιμές FB, FBΓ και FΓ και άγνωστες τις FA και FΔ.

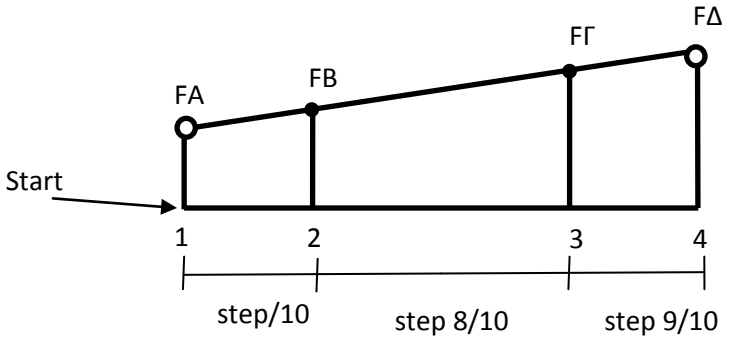
$$Sum_1 = \frac{FA+F\Delta}{2} step \tag{7.1}$$

$$Sum_2 = \frac{FA+FB\Gamma}{4} step + \frac{FB\Gamma+F\Delta}{4} step \tag{7.2}$$

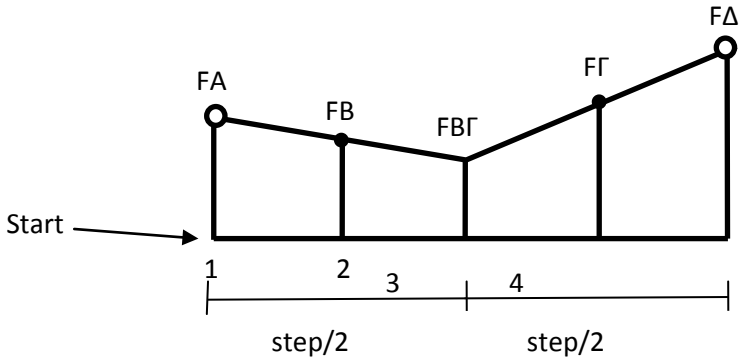
$$division = \frac{Absolute\ (Sum_2 - Sum_1)}{sum_2} \tag{7.3}$$

Επειδή υπάρχει περίπτωση για κάποιο λόγο να μην μπορεί να βρει το πρόγραμμα κατάλληλο step που να πληροί όλες τις προϋποθέσεις (καλή ακρίβεια κυρίως) υπάρχει περιορισμός ύστερα από 1000 επαναλήψεις, να σταματάει κάθε προσπάθεια για καλύτερο step να εμφανίζεται μήνυμα λάθους και να συνεχίζει στο επόμενο σημείο προς ολοκλήρωση.

Για να βελτιωθεί η ακρίβεια ολοκλήρωσης, χωρίς σοβαρό κόστος στον χρόνο, στο πρόγραμμα που λύνει την εξ. (4.2), γίνεται ολοκλήρωση στον χρόνο από 0 έως 1 με τα τραπέζια Σχήμα 7.1, 7.2, ενώ για χρόνο μεγαλύτερο του ενός χρησιμοποιούνται παραπλήσια τραπέζια της μορφής Σχήμα 7.3 και 7.4.



Σχήμα 7.3: Παρουσιάζεται τραπέζιο με γνωστές τις τιμές ΦΒ και ΦΓ και άγνωστες τις ΦΑκαι ΦΔ.

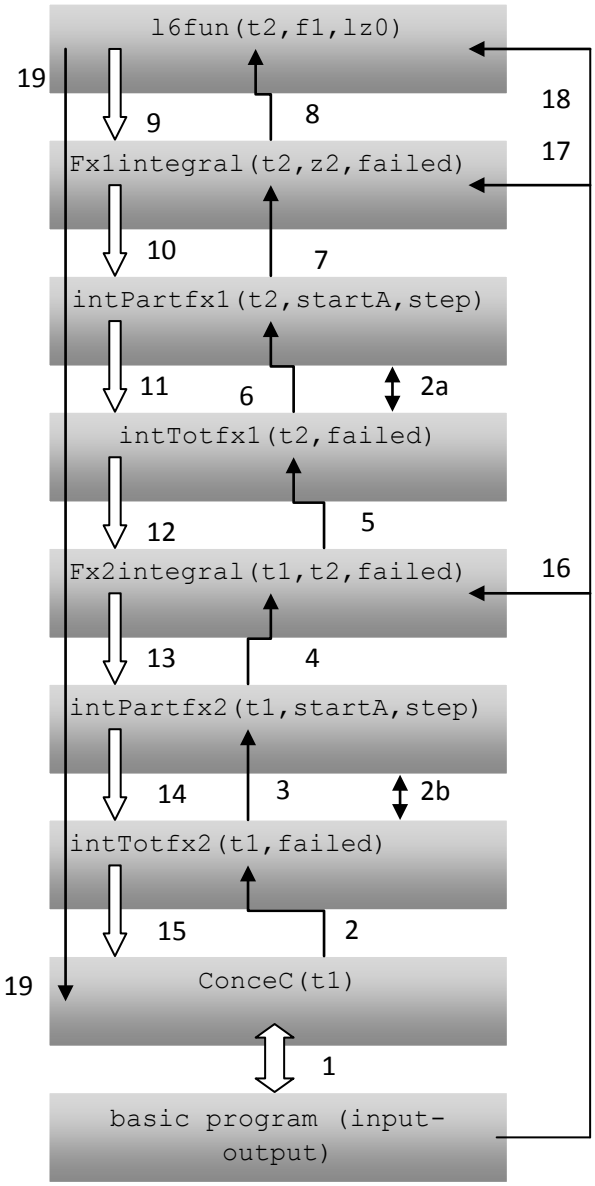


Σχήμα 7.4: Παρουσιάζεται τραπέζιο με γνωστές τις τιμές ΦΒ,ΦΒΓ και ΦΓ και άγνωστες τις ΦΑκαι ΦΔ.

7.2 ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΓΕΙΝΚΟΥ ΚΩΔΙΚΑ ΑΝΑΛΥΤΙΚΩΝ ΛΥΣΕΩΝ

Η εξ. (4.2) λύνεται από τα πρόγραμμα με όνομα: "pFin_Program". Η γενική του λειτουργία είναι η εξής:

Υπάρχει ένα βασικό αρχείο πηγαίου κώδικα (basic program.F90) το οποίο αναλαμβάνει να ανοίγει τα κατάλληλα αρχεία και να διαβάζει τιμές από αυτά. Αυτές μπορεί να είναι είτε βασικά μεγέθη που περιγράφουν το μοντέλο, είτε οι συντεταγμένες των σημείων στα οποία απαιτείται να ευρεθεί η συγκέντρωση C ύστερα από κάποιον χρόνο t. Στην συνέχεια υπάρχει κώδικας (input_check.F90) ο οποίος ελέγχει εάν τα δεδομένα πληρούν κάποιους συγκεκριμένους όρους (μη αρνητικός χρόνος...). Εάν όλα είναι όπως πρέπει τότε έρχεται η σειρά των αριθμητικών πράξεων (subfunctions.F90, bessel modified function.F90). Η υπορουτίνα που υπολογίζει την bessel modified function εξ. (4.6) δίδεται από τις γενικές μεθόδους της σελίδας www.netlib.org (γενική βιβλιογραφία). Επιπλέον υπάρχει κώδικας (interpolation.f90) ο οποίος κάνει γραμμική παρεμβολή στην περίπτωση όπου ύστερα από 1000 επαναλήψεις το πρόγραμμα αδυνατεί να βρει step το οποίο να πληροί τις απαιτήσεις ακριβείας. Η παρεμβολή γίνεται μεταξύ δύο γειτονικών σημείων C_2, C_3 με συντεταγμένες z_2=z+0.001, x_2=x+0.001, y_2=y+0.001, z_3=z-0.001, x_3=x-0.001, y_3=y-0.001 αντίστοιχα. Ενώ X,Y,Z είναι οι συντεταγμένες του αρχικού σημείου όπου απέτυχε να βρεθεί η συγκέντρωση με αναλυτικές λύσεις. Η βασική δομή που ακολουθεί ο κώδικας "sub functions.F90" είναι:



Σχήμα 7.5: Παρουσιάζεται η δομική οργάνωση του κώδικα υπεύθυνου για την επίλυση σημειακής πηγής με περιορισμένο υδροφορέα εξ. (4.2).

Όπως παρατηρείται στο Σχήμα 7.5 όλη η διαδικασία ξεκινάει από το basicprogram.f90. Από εκεί γίνεται η τροφοδοσία δεδομένων (σταθερών που περιγράφουν το μοντέλο κυρίως) σε όποια υπολογιστική μονάδα χρειαστεί. Έτσι μέσο των 1, 16, 17, 18 η βασική μονάδα παρέχει τις απαραίτητες σταθερές στις υπορουτίνες `ConceC(t1)`,`Fx2integral(t1,t2,failed`,`Fx1integral(t2,z2,failed)` και `l6fun(t2,f1,lz0)`. Η `Fx1integral` και η `Fx2integral` αποτελούν τις συναρτήσεις οι οποίες πρόκειται να ολοκληρωθούν (εσωτερικό και εξωτερικό ολοκλήρωμα της εξ. (4.2) αντίστοιχα). Ο μηχανισμός της κάθε ολοκλήρωσης βασίζεται στις ρουτίνες `intPartfx2(t1,startA,step)` και

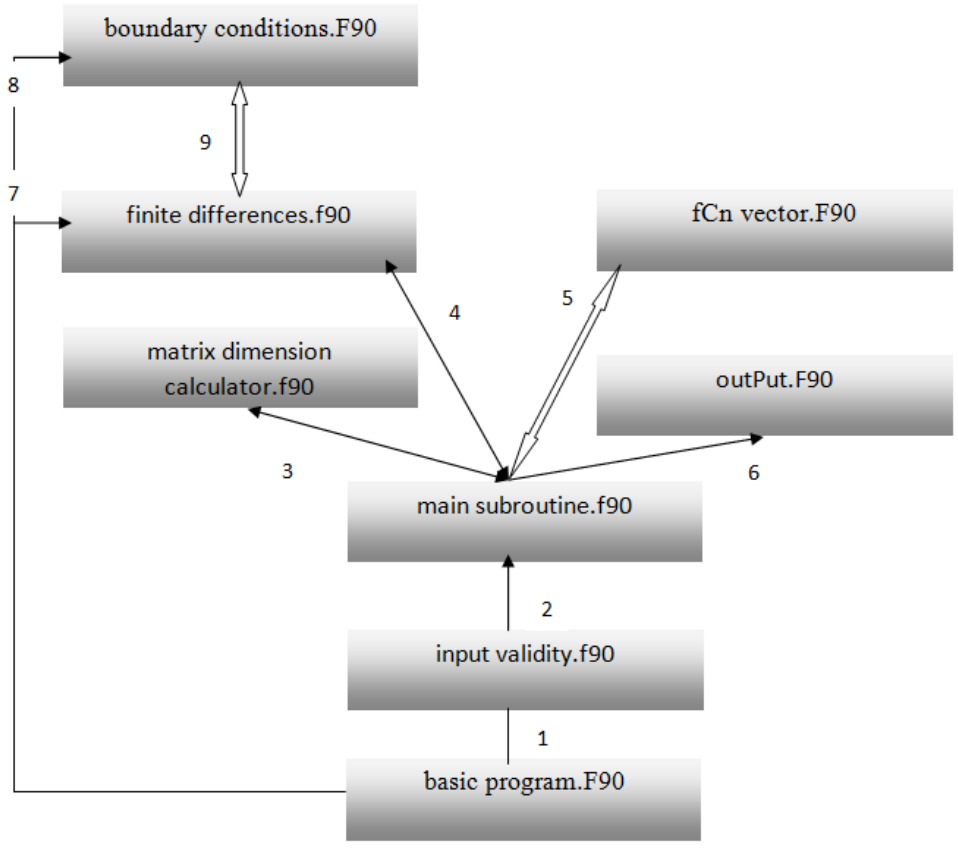
`intPartfx1(t1,startA,step)`. Αυτές υλοποιούν τον μηχανισμό με τα τραπέζια που περιγράφθηκε στην παράγραφο 7.1. Με την σειρά τους οι `intTotfx2(t1,failed)` και `intTotfx1(t1,failed)` φροντίζουν ώστε να αθροίζουν τα εμβαδά από τα τραπέζια που τους παρέχονται, μέσω των συνδέσεων 2b και 2a (από τις ρουτίνες `intPartfx2(t1,startA,step)` `intPartfx2(t1,startA,step)` αντίστοιχα) αλλά και ταυτόχρονα φροντίζουν ώστε το βήμα από το προηγούμενο τραπέζιο να θεωρείται σαν πιθανό αρχικό στο επόμενο.

Η `l6fun(t2,f1,lz0)` είναι η δομική μονάδα που πραγματοποιεί τον μετασχηματισμό Fourier ο οποίος περιγράφεται από την εξίσωση (4.14). Η υπορουτίνα αυτή θέλει ιδιαίτερη προσοχή καθώς ο μηχανισμός εξόδου περιγράφεται από την εξ. (7.3), με division< 0.000000001. Κανείς θα μπορούσε να θεωρήσει υπερβολικό το μέγεθος αυτό, να το μειώσει να τρέξει το πρόγραμμα και να δει ότι τα αποτελέσματα δεν διαφέρουν πολύ. Στην πραγματικότητα όμως υπάρχουν τυχαία σημεία στα οποία η διαφορά είναι αρκετά μεγάλη σε τέτοιο βαθμό ώστε η συγκέντρωση καθώς απομακρυνόμαστε από την πηγή να μεγαλώνει αντί να μικραίνει. Οπότε χρειάζεται ιδιαίτερη προσοχή στην μείωση του συντελεστή αυτού. Κρίνεται επιθυμητό γενικά να χρησιμοποιούνται έτοιμοι κώδικές FFT (fast Fourier transformations (TeukolskyS.Aet. al, 1997) που επιτυγχάνουν μεγάλες ταχύτητες στις μετατροπές διασφαλίζοντας πάντα την ακρίβεια. . Όμως στην παρούσα περίπτωση δεν ήταν αναγκαίο κάτι τέτοιο καθώς υπάρχει ο όρος $\exp[-\psi_m^2 D_z t]$ στην εξ. (4.14) πλησιάζει γρήγορα το 0 και ως εκ τούτου δεν χρειάζονται πολλά αθροίσματα ώστε να υπάρχει η απαιτούμενη ακρίβεια. Για τιμές δε του $\psi_m^2 D_z t$>700 η τιμή της παράστασης $\exp[-\psi_m^2 D_z t] = 10^{-305} = 0$ είναι πρακτικά μηδέν.

Μέσω των συνδέσεων 2-8 (Σχ. 7.5) κάθε μονάδα μεταφέρει στην επόμενη τον χρόνο t_1 ή t_2. Όπου t_1 είναι το πάνω όριο ολοκλήρωσης (του εξωτερικού ολοκληρώματος) της εξ. 4.2 και t_2 ομοίως του αντιστοίχου εσωτερικού ολοκληρώματος. Μέσω των συνδέσεων 9-15 κάθε ρουτίνα μεταφέρει τα αποτελέσματα των πράξεων της σε αυτήν που την κάλεσε. Τέλος η σύνδεση 19 φροντίζει να ενημερώσει κατευθείαν το βασικό πρόγραμμα ότι κάποια από τις υπορουτίνες ολοκλήρωσης ή ο μετασχηματισμός Fourier απέτυχαν μέσα σε κάποιες προκαθορισμένες επαναλήψεις να βρουν αποτέλεσμα με την απαιτούμενη ακρίβεια. Πλέον οι υπολογισμοί πρέπει να σταματήσουν ή να ξεκινήσει η γραμμική παρεμβολή όταν αυτό είναι δυνατό από τον κώδικα.

7.3 ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΓΕΝΙΚΟΥ ΚΩΔΙΚΑ ΓΙΑ ΑΡΙΘΜΗΤΙΚΕΣ ΛΥΣΕΙΣ

Οι αναλυτικές λύσεις παρά την ακρίβεια που διαθέτουν αντιμετωπίζουν προβλήματα χρόνου και αποτελεσματικότητας. Οι πολλαπλές ολοκληρώσεις είναι αρκετά χρονοβόρες και ταυτόχρονα δεν υπάρχουν για όλες τις περιπτώσεις αναλυτικές λύσεις (π.χ. πορώδες με πολλαπλές πηγές και πηγάδια). Για αυτόν τον λόγο δημιουργήθηκε πηγαίος κώδικας ο οποίος επιλύει τόσο το πρόβλημα της απλής μεταφοράς Σχήμα 5.1, εξ. (5.1-5.24) αλλά και το πρόβλημα της συμμεταφοράς Σχήμα 5.2, εξ. (5.25-5.56). Χρησιμοποιήθηκαν δε πέραν του Intel Fortran Compiler και έτοιμες υπορουτίνες που λύνουν διαφορικές εξισώσεις με ακαμψία προερχόμενες από το κλειστό πακέτο "w_ode_a_1.0.0.006, Intel differential equation solvers" (http://software.intel.com/en-us/articles/pre-release-license-agreement-for-intel-ordinary-differential-equations-solver-library-accept-end-user-license-agreement-and-download/). Εξαιτίας της πολυπλοκότητας των εξισώσεων της συμμεταφοράς εξ. (5.25-5.56) στο παρακάτω Σχήμα 7.6 θα παρουσιαστεί μόνο ένα σκαρίφημα του βασικού σκελετού επίλυσης απλής μεταφοράς εξ. (5.1-5.24).



Σχήμα 7.6: Παρουσιάζεται η δομική οργάνωση του κώδικα υπεύθυνου για την επίλυση του αριθμητικού μοντέλου Σχήμα 5.1.

Η επίλυση αποτελείται από 8 δομικές μονάδες τις: "outPut.F90, boundaryconditions.F90, fCnvector.F90, finitedifferences.f90, input validity.f90, main subroutine.f90, matrix dimension calculator.f90 outPut.F90" Η κάθε μια από αυτές εκτελεί και από μια διαφορετική εργασία.

Στο Σχήμα 7.6 φαίνεται η δομική οργάνωση του κώδικα υπεύθυνου για την αριθμητική λύση της απλής μεταφοράς. Όπως παρατηρείται όλα ξεκινούν από το βασικό πρόγραμμα το οποίο αναλαμβάνει να διαβάσει από το αρχείο εισόδου τα δεδομένα (σταθερές) που περιγράφουν το μοντέλο προς επίλυση. Στην συνέχεια μέσο του καναλιού 1 ελέγχεται κατά πόσο τα δεδομένα που διαβάστηκαν είναι αποδεκτά η όχι. Για παράδειγμα δεν γίνεται να βρίσκεται το κέντρο της πηγής εξωτερικά από την περιοχή του πορώδους. Στην περίπτωση που κάποιο δεδομένο δεν είναι σωστό, η εκτέλεση του προγράμματος σταματάει και εμφανίζεται το σχετικό μήνυμα. Αφού όλα είναι εντάξει την συνέχεια αναλαμβάνει το κυρίως πρόγραμμα (main subroutine). Από εδώ και πέρα η υπορουτίνα αυτή συντονίζει την εκτέλεση των υπολοίπων δομικών μονάδων μέσω των καναλιών 3, 4, 5, 6. Πρώτη κίνηση για την επίλυση του προβλήματος μεταφοράς (όπως περιγράφεται από τις εξισώσεις (3.44)-(3.69) είναι η δημιουργία του πίνακα ο οποίος περιέχει όλες τις σταθερές του συστήματος των γραμμικών εξισώσεων. Πριν γίνει αυτό πρέπει να γνωρίζουμε ποιο ακριβώς θα είναι το μέγεθος του ώστε να το περιγράψουμε στις υπορουτίνες επίλυσης.

Η δομική μονάδα matrix dimension calculator.f90 έχει τα απαραίτητα ώστε να μετρήσει το μέγεθος των διανυσμάτων που χρειάζονται για να αποθηκευτεί ο απαιτούμενος πίνακας. Η πληροφορία αυτή μεταβιβάζεται στην κυρίως υπορουτίνα (main subroutine.f90) μέσω της διόδου 3 Σχήμα 7.6, η οποία στην συνέχεια καλεί την finite differences.f90 μέσω του καναλιού 4. Η τελευταία αναλαμβάνει να δημιουργήσει τον πίνακα των σταθερών με το μέγεθος όπως προσδιορίστηκε από την matrix dimension calculator.f90. Στην ουσία για κάθε κελί του κατακερματισμένου χώρου γράφονται οι εξισώσεις (5.11)-(5.24). Είναι όμως απαραίτητο να προστεθούν και οι συνοριακές συνθήκες οι οποίες περιγράφουν το μοντέλο Σχήμα 5.1. Για αυτόν το λόγο υπάρχει αμφίδρομη επικοινωνία με την boundary conditions.F90 μέσω του καναλιού 9. Η οποία φροντίζει να εφαρμόζει τις συνοριακές εξισώσεις (5.3)-(5.7β). Πλέον ο πίνακας είναι σχηματισμένος και έτοιμος προς επίλυση.

Για την επίλυση του συστήματος γραμμικών εξισώσεων επιλέχθηκε κλειστός εμπορικός κώδικας, ο οποίος εμπεριέχεται στις βιβλιοθήκες MKLV10.2.2.025 (MATH KERNEL LIBRARY) της εταιρίας INTEL (http://software.intel.com/en-us/intel-compilers/) με την ονομασία PARDISO* (Parallel Direct Sparse Solver Interface). Ο κώδικας αυτός μπορεί πολύ εύκολα να λύσει μεγάλους πίνακες εκμεταλλευόμενος κάθε ιδιαιτερότητα που μπορεί να υπάρχει (π.χ. πολλά μηδενικά,

διαγώνιος πίνακας...). Για να λειτουργήσει σωστά όμως απαιτεί συγκεκριμένο τρόπο εισαγωγής και αποθήκευσης του πίνακα προς επίλυση. Κανείς μπορεί να διαβάσει αναλυτικά το εγχειρίδιο χρήσης από την εταιρία που το παράγει . Συνοπτικά μόνο στο Σχήμα 7.7 φαίνεται ένα παράδειγμα αποθήκευσης σε στήλες τυχαίου μη διαγώνιου πίνακα με πολλά μηδενικά.

$$B = \begin{bmatrix} 1 & -1 & * & -3 & * \\ -2 & 5 & * & * & * \\ * & * & 4 & 6 & 4 \\ -4 & * & 2 & 7 & * \\ * & 8 & * & * & -5 \end{bmatrix}$$

The matrix B has 13 non-zero elements, and all of them are stored as follows:

Table A-2 Storage Arrays for a Non-Symmetric Matrix

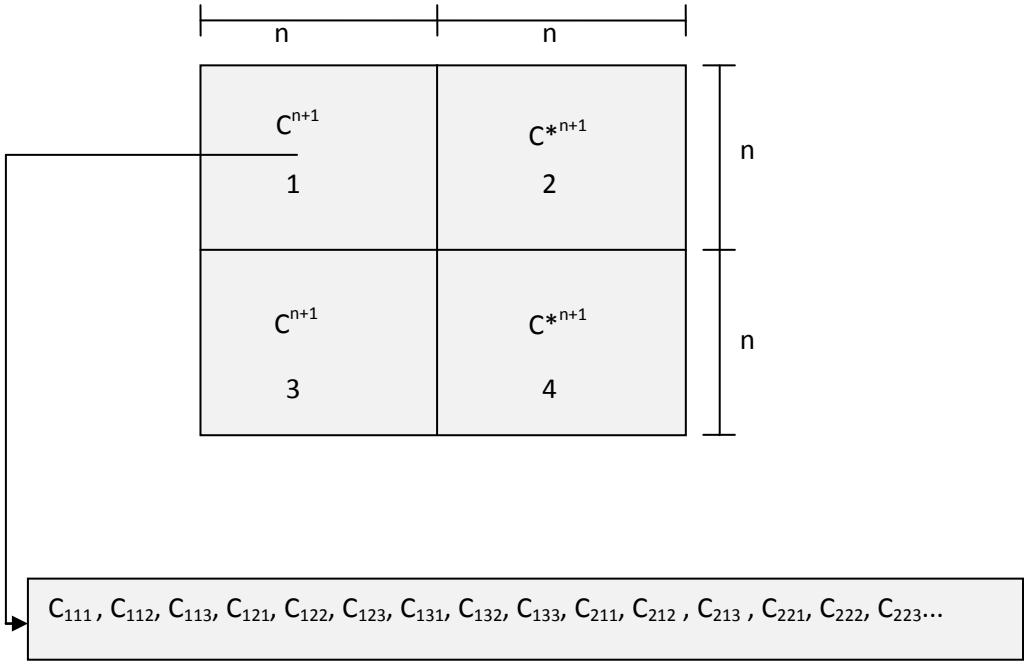
values	=	(1	-1	-3	-2	5	4	6	4	-4	2	7	8	-5)
columns	=	(1	2	4	1	2	3	4	5	1	3	4	2	5)
rowIndex	=	(1	4	6	9	12	14)							

Σχήμα 7.7: Παρουσιάζεται ο τρόπος αποθήκευσης μη διαγώνιου πίνακα από τον κώδικα PARDISO (πηγή mklman.pdf, intel MKLV10.2).

Πριν όμως φτάσουμε στην επίλυση πρώτα πρέπει να σχηματιστεί το δεξί μέλος (γνωστό και σταθερό), εξ. (5.20, 5.24) ($\{f(C^n)\}$), των γραμμικών εξισώσεων ώστε να γίνει συνολικά και σωστά η επίλυση. Αυτόν το ρόλο τον αναλαμβάνει η μονάδα fCnvector.F90 μέσο της διόδου 5. Έτσι δημιουργείται το δεξί μέλος που πρόκειται να επιλυθεί. Την πρώτη φορά που καλείται ουσιαστικά αποτελεί την εφαρμογή της αρχικής συνθήκης εξ (5.3). Πρόσθετα η fCnvector.F90 ενεργοποιείται στο τέλος κάθε επίλυσης του συστήματος των εξισώσεων, ώστε το δεξί μέλος των σταθερών να γνωρίζει τα αποτελέσματα των προηγουμένων πράξεων (οι αρχικές συγκεντρώσεις πλέον δεν είναι μηδέν όπως υπαγόρευε η εξ. (5.3) στην αρχή). Αφού γίνουν όλα αυτά, αναλαμβάνει τώρα η μονάδα outPut.F90 να καταγράψει στα κατάλληλα αρχεία τα αποτελέσματα που ζήτησε ο χρήστης.

Στο Σχήμα 7.8 φαίνεται ο τρόπος με τον οποίο οι άγνωστοι έχουν κατανεμηθεί μέσα στον τετραγωνικό πίνακα. Όπου n= ο αριθμός των συνολικών κελιών που έχει ο κατακερματισμένος χώρος, C^{n+1} είναι οι αναζητούμενες συγκεντρώσεις της διαλυμένης ουσίας και C^{*n+1} είναι συγκεντρώσεις της προσροφημένης ουσίας. Οι περιοχές 1 και 2 του πίνακα προέρχονται από την εφαρμογή της εξ. (5.1) ενώ οι 3 και 4 προέρχονται από την (5.2). Προσοχή σε κάθε γραμμή του πίνακα ένας μικρός

αριθμός σταθερών είναι μη μηδενικός. Αυτός ισούται με τον αριθμό των όρων των εξισώσεων (5.11) ή (5.21) (ανάλογα με το αν είμαστε στο πάνω μισό ή στο κάτω μισό του πίνακα). Συνολικά περιέχονται $4n^2$ στοιχεία.



Σχήμα 7.8: Παρουσιάζεται ο τρόπος με τον οποίο οι τέσσερις άγνωστοι έχουν κατανεμηθεί μέσα στον τετραγωνικό πίνακα.

.

8. ΑΠΟΤΕΛΕΣΜΑΤΑ: FITTING-ΑΝΑΛΥΣΗ ΕΥΑΙΣΘΗΣΙΑΣ

8.1 FITTING

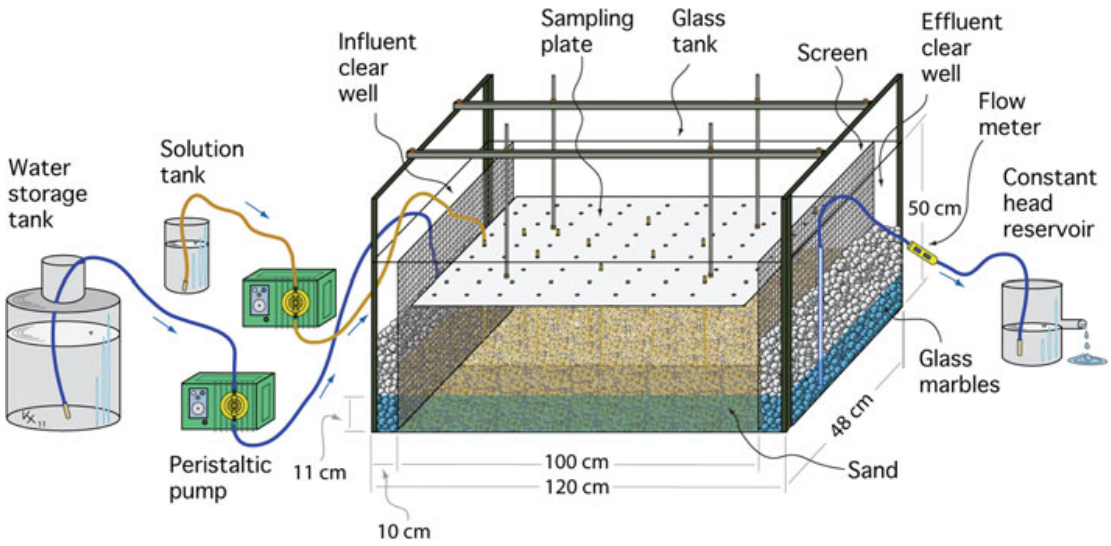
Ορισμός:
Με τον όρο Fitting εννοούμε την διαδικασία υπολογισμού φυσικών σταθερών ενός μοντέλου, έχοντας γνωστές τιμές μιας ή περισσοτέρων μεταβλητών σε διάφορα σημεία ενός χώρου.

Αυτό που συμβαίνει στην πράξη είναι ότι πολλές φορές έχουμε κάποιο υπόγειο ορίζοντα στο οποίον κατά λάθος χύθηκε κάποιος ρύπος. Η πρώτη σκέψη είναι να υπολογίσουμε πόσο μακριά θα μεταφερθεί αυτός ο ρύπος, σε πόσο χρόνο θα σταματήσει να μεταφέρεται και ποιά θα είναι η συγκέντρωση του τότε. Για να γίνει κάτι τέτοιο κανείς δεν χρειάζεται μόνο την αρχική συγκέντρωση του ρύπου αλλά και το ποιές είναι οι φυσικές ιδιότητες του. Για παράδειγμα ποιός είναι ο συντελεστής διασποράς του, με ποιόν ρυθμό προσκολλάται στο πορώδες μέσο ή πόσο γρήγορα καταστρέφεται. Όλες αυτές οι παράμετροι είναι πολύ σημαντικοί για την μεταφορά του και ο τρόπος υπολογισμού τους είναι μέσω Fitting. Η διαδικασία που ακολουθείται είναι η εξής. Πρώτα συλλέγεται ένα εδαφικό τμήμα από την περιοχή που έγινε η απόθεση του ρύπου. Σχηματίζεται ένα ομοίωμα αυτού στο εργαστήριο και επαναλαμβάνεται η προσθήκη ρύπου σε αυτό φροντίζοντας να διατηρηθούν οι συνθήκες του πεδίου. Επιλέγεται ένα μαθηματικό μοντέλο που περιγράφει την μεταφορά του ρύπου λαμβάνοντας υπόψη τις συνοριακές συνθήκες που επικρατούν. Στην συνέχεια λαμβάνονται δείγματα από το ομοίωμα, υπολογίζεται η συγκέντρωση του ρύπου και ταυτόχρονα συγκρίνονται τα αποτελέσματα με τις τιμές που δίνει το μαθηματικό μοντέλο. Φυσικά στην αρχή δίδονται τυχαίες τιμές στις φυσικές παραμέτρους που χρησιμοποιεί το μοντέλο αλλά με την χρήση ειδικού λογισμικού οι παράμετροι αυτοί διορθώνονται κατάλληλα έτσι ώστε οι διαφορές μεταξύ των τιμών που δίνει το πείραμα στο εργαστήριο, με τις τιμές που δίνει το μαθηματικό μοντέλο, να μειωθούν όσο περισσότερο γίνεται. Στο τέλος οι φυσικοί παράμετροι του (του μαθηματικού μοντέλου) είναι αυτοί που αντιπροσωπεύουν τις συνθήκες στο ομοίωμα και συνεπώς τις συνθήκες του πεδίου. Έχοντας τις παραμέτρους αυτούς, κανείς γυρίζει στο αρχικό του πρόβλημα και υπολογίζει αποτελεσματικά την μεταφορά του ρύπου στο πεδίο.

Παρακάτω παρουσιάζονται δεδομένα από πειράματα που έγιναν στην βιβλιογραφία και με την βοήθεια των ρουτινών που παρουσιάστηκαν στο κεφάλαιο 7 αλλά και κατάλληλου λογισμικού Fitting έγινε ο υπολογισμός των φυσικών παραμέτρων τούς. Το λογισμικό που χρησιμοποιήθηκε είναι το "Pest". Υλοποιεί μεθόδους Gauss Marquardt Levenberg with Broyden Jacobian updating και Shuffled Complex Evolution Covariance Matrix Adaptation με αποτέλεσμα να μπορεί να βρίσκει παραμέτρους με μεγάλη σταθερότητα και ακρίβεια. Ταυτόχρονα είναι σε θέση να κάνει ανάλυση αβεβαιότητας σε Linear over-determined ή Nonlinear over-determined through calibration-constrained predictive maximization/minimization problems. Πιο πολλά για το "Pest" μπορεί να βρει κανείς στο http://www.pesthomepage.org/Highly-parameterized_inversion.php.

Πρώτο πείραμα: Απλή μεταφορά βακτηρίων στις τρείς διστάσεις.

Η μεταφορά βακτηρίων στο υπέδαφος είναι ένα πολύ σημαντικός μολυσματικός παράγοντας που τα τελευταία χρόνια έχει αρχίσει να τραβά την προσοχή των επιστημόνων. Για αυτό κρίθηκε ενδιαφέρον χρησιμοποιώντας το αριθμητικό μοντέλο που αναπτύχθηκε κεφ. 7 σε συνδυασμό με το Pest να υπολογιστούν οι φυσικοί παράμετροι σε πείραμα απλής μεταφοράς βακτηρίων, μέσα σε άμμο στις τρείς διαστάσεις Σχήμα 8.1.



Σχήμα 8.1: Παρουσιάζεται η πειραματική διάταξη που χρησιμοποιήθηκε στην μεταφορά ιχνηθέτη και βακτηρίων (Chrysikopoulos et al., 2012). Ο υδροφορέας αποτελείτο από καλά διαβαθμισμένη, ομογενοποιημένη άμμο (quartz sand), ενώ στην είσοδο και στην έξοδο του υπήρχαν πηγάδια εισαγωγής και αφαίρεσης ρύπου.

Όπως κρίνεται απαραίτητο σε τέτοιες περιπτώσεις πριν την μεταφορά των βακτηρίων Pseudomonas (P.) putida γίνεται μια δοκιμαστική μεταφορά ιχνηθέτη (Tracer) ώστε να εκτιμηθεί η συμπεριφορά του υδροφορέα. Παρακάτω ακολουθούν οι οριακές συνθήκες εξ. (8.3-8.9) καθώς και οι εξισώσεις που χρησιμοποιήθηκαν για την δημιουργία του μαθηματικού μοντέλου απλής μεταφοράς εξ. (8.1, 8.2):

$$\frac{\partial C(t,x,y,z)}{\partial t} + \frac{\rho}{\theta}\frac{\partial C^*(t,x,y,z)}{\partial t} - D_x\frac{\partial^2 C(t,x,y,z)}{\partial x^2} - D_y\frac{\partial^2 C(t,x,y,z)}{\partial y^2} - D_z\frac{\partial^2 C(t,x,y,z)}{\partial z^2} + U\frac{\partial C(t,x,y,z)}{\partial x} + \lambda\, C(t,x,y,z) + \frac{\lambda^*\rho}{\theta}C^*(t,x,y,z) = F(t,x,y,z) \quad (8.1)$$

$$\frac{\rho}{\theta}\frac{\partial C^*(t,x,y,z)}{\partial t} = r_1 C(t,x,y,z) - r_2\frac{\rho}{\theta}C^*(t,x,y,z) - \frac{\lambda^*\rho}{\theta}C^*(t,x,y,z) \quad (8.2)$$

Οι αρχικές και συνοριακές συνθήκες που περιγράφουν το μοντέλο μας είναι εξ. (8.3-8.9):

$$C(0,x,y,z) \tag{8.3}$$

$$C(t,0,y,z) = 0 \tag{8.4}$$

$$\frac{\partial C(t,L_x,y,z)}{\partial x} = 0 \tag{8.5}$$

$$\frac{\partial C(t,x,0,z)}{\partial y} = 0 \tag{8.6}$$

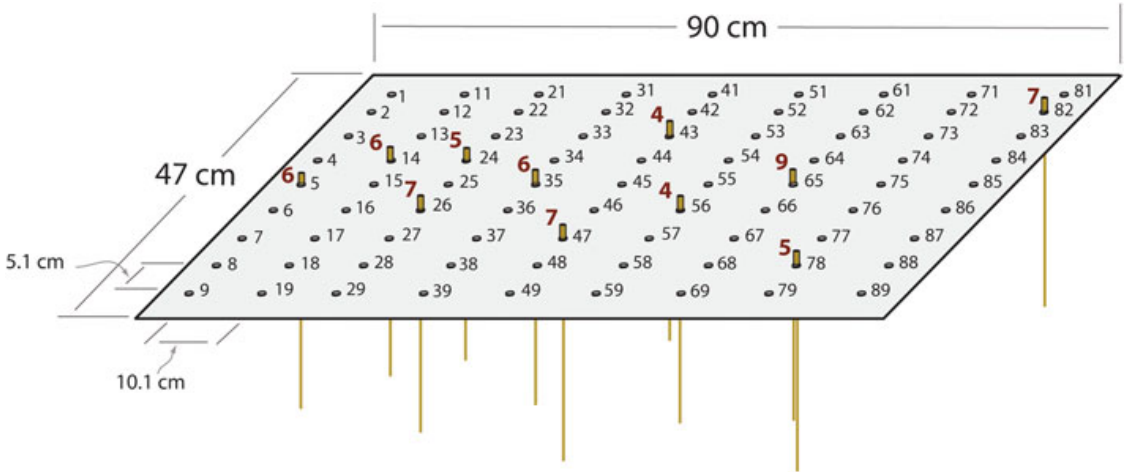
$$\frac{\partial C(t,x,L_y,z)}{\partial y} = 0 \tag{8.7}$$

$$\frac{\partial C(t,x,y,0)}{\partial Z} = 0 \tag{8.8}$$

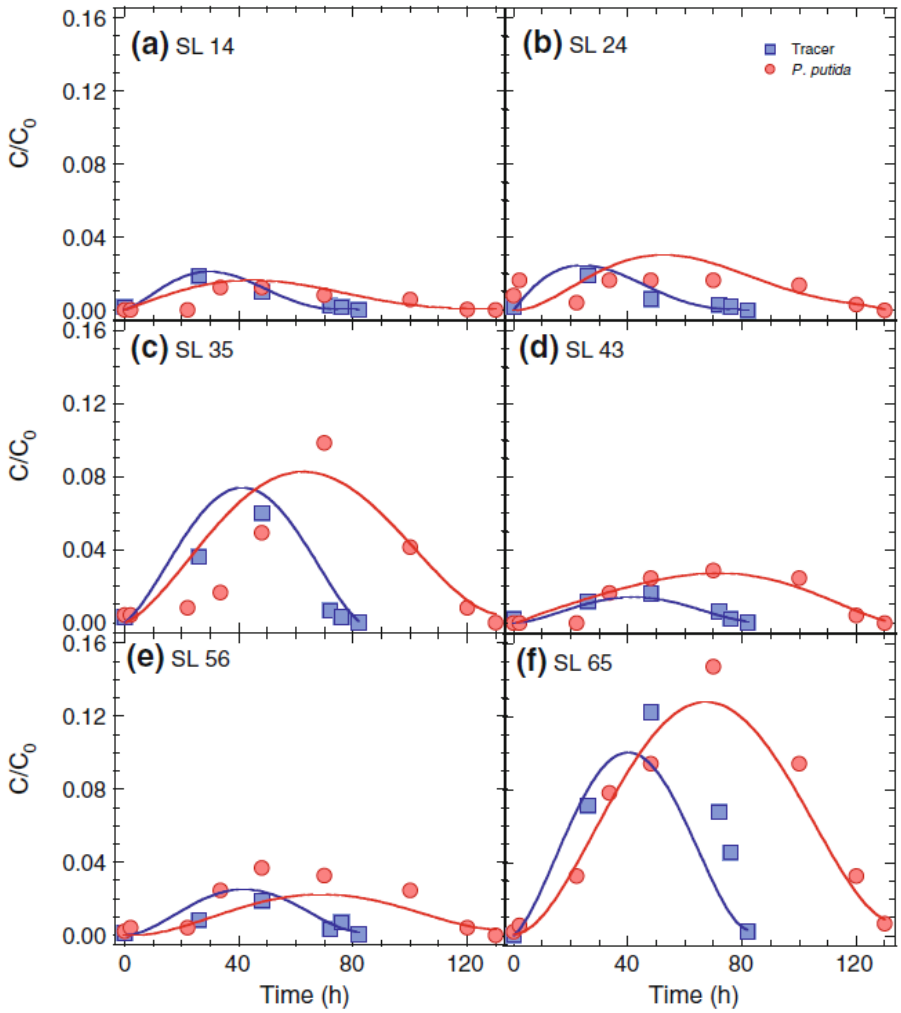
$$\frac{\partial C(t,x,y,L_z)}{\partial Z} = 0 \tag{8.9}$$

Με τις παραπάνω εξισώσεις να έχουν την εξής ερμηνεία: Η εξ. (8.3) σημαίνει ότι στην αρχή του χρόνου δεν υπήρχε καθόλου ρύπος στον υδροφορέα μας Σχήμα 8.1, η εξ. (8.4) ότι στην είσοδο του υδροφορέα εισάγεται καθαρό νερό με μηδενική συγκέντρωση ρύπου, η εξ. (8.4) ότι ο ρύπος που εξέρχεται από το υδροφορέα δεν μπορεί να ξαναμπεί σε αυτόν και απομακρύνεται, ενώ τέλος οι εξ. (8.6-8.9) σημαίνουν ότι τα τοιχώματα στις διευθύνσεις y και z είναι αδιαπέραστα και ρύπος δεν μπορεί να εισέλθει μέσα από αυτά.

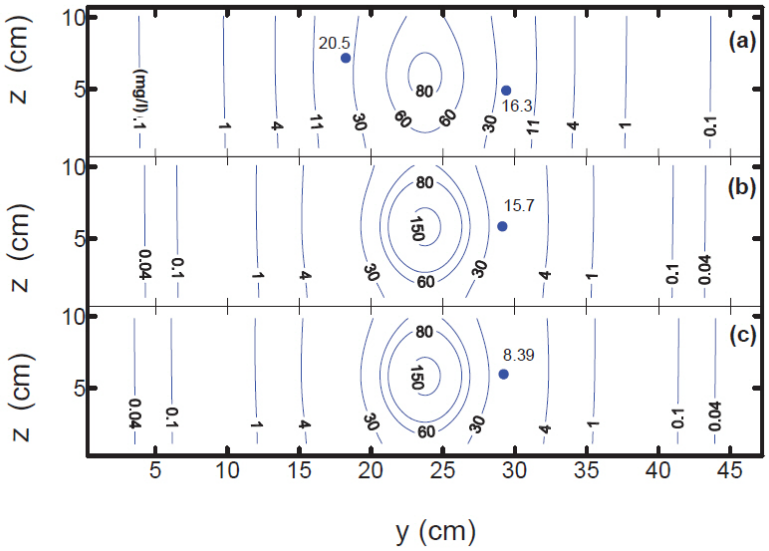
Στο Σχήμα 8.2 παρουσιάζονται οι θέσεις των σημείων από όπου λήφθηκαν τα δείγματα για το "fitting". Ενώ στα Σχήματα 8.3-8.7 που ακολουθούν φαίνονται τόσο τα πειραματικά όσο και τα αποτελέσματα από την μαθηματική προσομοίωση.



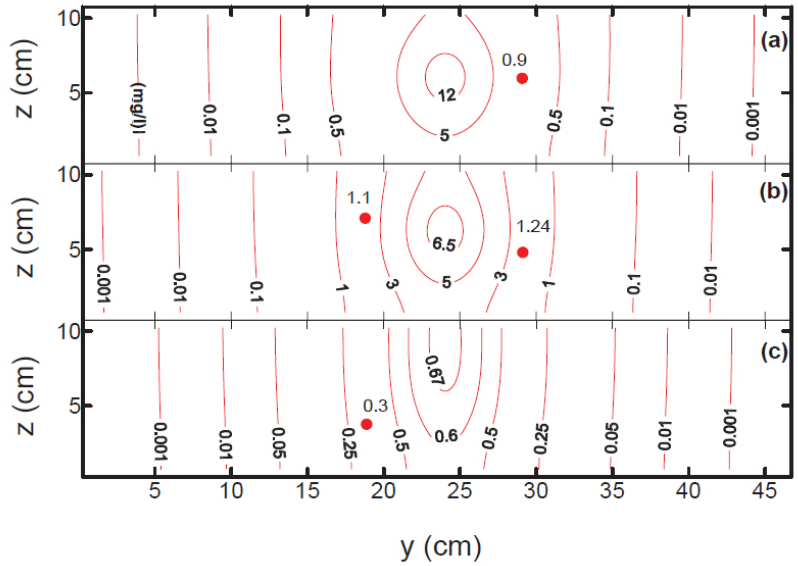
Σχήμα 8.2: Παρουσιάζεται η πλάκα που συγκρατούσε τις δειγματοληπτικές βελόνες για την λήψη δειγμάτων. Τα νούμερα που είναι πιο έντονα (bold) δείχνουν το βάθος στο οποίο έφταναν οι βελόνες (υψόμετρο μηδέν z=0 βρίσκεται πάνω στην ίδια την πλάκα).



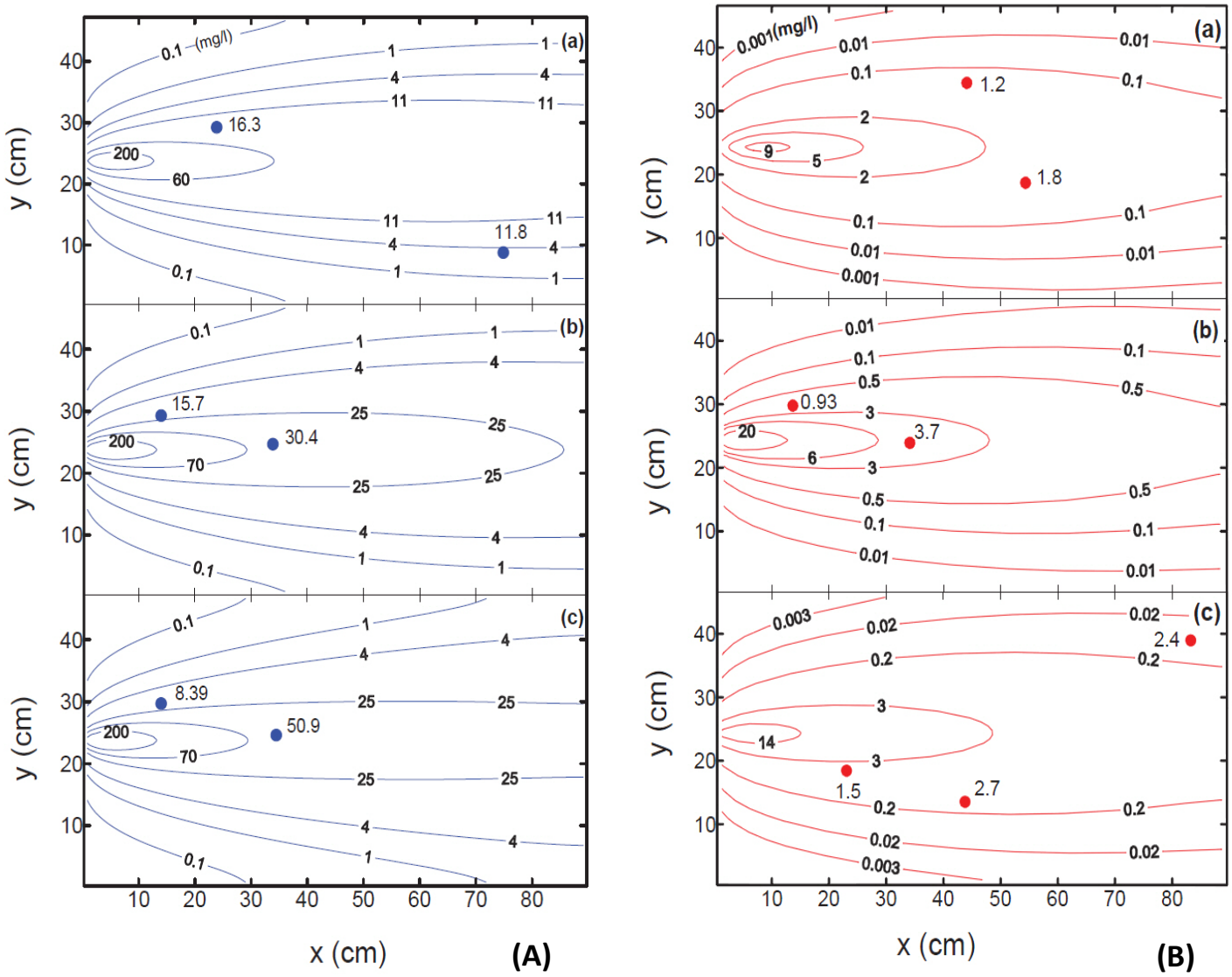
Σχήμα 8.3: Παρουσιάζονται αδιάστατα αποτελέσματα τόσο από τα πειράματα μεταφοράς βακτηρίων (κόκκινες κουκίδες), ιχνηθέτη (μπλε τετράγωνα) όσο και από την αριθμητική προσομοίωση (κόκκινη συνεχομένη καμπύλη αντιστοιχεί σε βακτήρια, μπλε συνεχομένη καμπύλη αντιστοιχεί σε ιχνηθέτη). Οι θέσεις από όπου έγινε η δειγματοληψία υπολογίζεται από τους όρους SL (sampling positions) σε συνδυασμό με το Σχήμα 8.2.



Σχήμα 8.4: Παρουσιάζονται ισοϋψείς συγκεντρώσεων (mg/L) ιχνηθέτη (Chloride) όπως υπολογίστηκαν από το μαθηματικό μοντέλο, για διάφορα κάθετα y-z επίπεδα του υδροφόρου ορίζοντα. Συγχρόνως οι μπλε κύκλοι συμβολίζουν τα πειραματικά δεδομένα (θέσεις SL 24, SL 26 SL 14) από την δειγματοληψία σε a) t = 26 h, x = 23,9 cm, b) t = 26 h, x = 13.75 cm, c) t = 48 h, x = 13.75 cm.



Σχήμα 8.5: Παρουσιάζονται ισοϋψείς συγκεντρώσεων (mg/L) βακτηρίων (P. putida) όπως υπολογίστηκαν από το μαθηματικό μοντέλο, για διάφορα κάθετα y-z επίπεδα του υδροφόρου ορίζοντα. Συγχρόνως οι κόκκινοι κύκλοι συμβολίζουν τα πειραματικά δεδομένα (θέσεις SL 24, SL 26 και SL 56) από την δειγματοληψία σε a) t = 48 h, x = 13.75 cm, b) t = 48 h, x = 23.9 cm, c) t = 22 h, x = 54.35 cm



Σχήμα 8.6Α: Παρουσιάζονται ισοϋψείς συγκεντρώσεων (mg/L) ιχνηθέτη (Chloride) όπως υπολογίστηκαν από το μαθηματικό μοντέλο, για διάφορα οριζόντια y-z επίπεδα του υδροφόρου ορίζοντα. Συγχρόνως οι μπλε κύκλοι συμβολίζουν τα πειραματικά δεδομένα (θέσεις SL 24, SL 78, SL 14 και SL 35 και SL 56) από την δειγματοληψία σε a) t = 26 h, z = 5 cm, b) t = 26 h, z = 6 cm, c) t = 48 h, z = 6 cm.
Σχήμα 8.6Β: Παρουσιάζονται ισοϋψείς συγκεντρώσεων (mg/L) βακτηρίων (P. putida) όπως υπολογίστηκαν από το μαθηματικό μοντέλο, για διάφορα οριζόντια x-y επίπεδα του υδροφόρου ορίζοντα. Συγχρόνως οι κόκκινοι κύκλοι συμβολίζουν τα πειραματικά δεδομένα (θέσεις SL 43, SL 56, SL 14 και SL 35, SL 26, SL 47 και SL 82) από την δειγματοληψία σε a) t = 33.5 h, z = 4 cm, b) t = 48 h, z = 6 cm και c) t = 48 h, z = 7 cm.

Τα πειραματικά δεδομένα από την μεταφορά της P. putida έγιναν "Fit" με την βοήθεια των εξισώσεων (8.1-8.9) αλλά και του λογισμικού Pest που αναφέρθηκε στην αρχή του κεφαλαίου. Το Pest χρησιμοποιώντας γνωστές τιμές των παραμέτρων $r_1 = 0{,}78\ h^{-1}$, $\lambda = 0{,}0109\ h^{-1}$, και $\lambda^* = \lambda/2 = 0{,}0055\ h^{-1}$, αλλά και θεωρώντας ότι κατά προσέγγιση Dy = DZ, υπολόγισε τις τιμές των U, Dx, Dy, DZ, και r_2. Οι σχέσεις $\lambda^* = \lambda/2$ και Dy = DZ επιβλήθηκαν από την ανάγκη ο μέγιστος αριθμός άγνωστων παραμέτρων που θα υπολογιστεί να μην ξεπερνά του τρείς (Chrysikopoulos et al., 2012)! Τα αποτελέσματα από το Fitting παρουσιάζονται στον πίνακα 8.1 μαζί με τα απαραίτητα διαστήματα εμπιστοσύνης 95%. Τέλος με την χρήση των υδροδυναμικών διασπορών Dx και Dy υπολογίστηκε η διασπαρσιμότητα (dispersivity) α ($D = \alpha U$) του υδροφορέα για την περίπτωση των βακτηρίων και βρέθηκε $\alpha_L = 4.54$ cm για την x' διεύθυνση , και $\alpha_{Ty} = \alpha_{Tz} = 0.31$ cm για τις άλλες δύο διευθύνσεις.

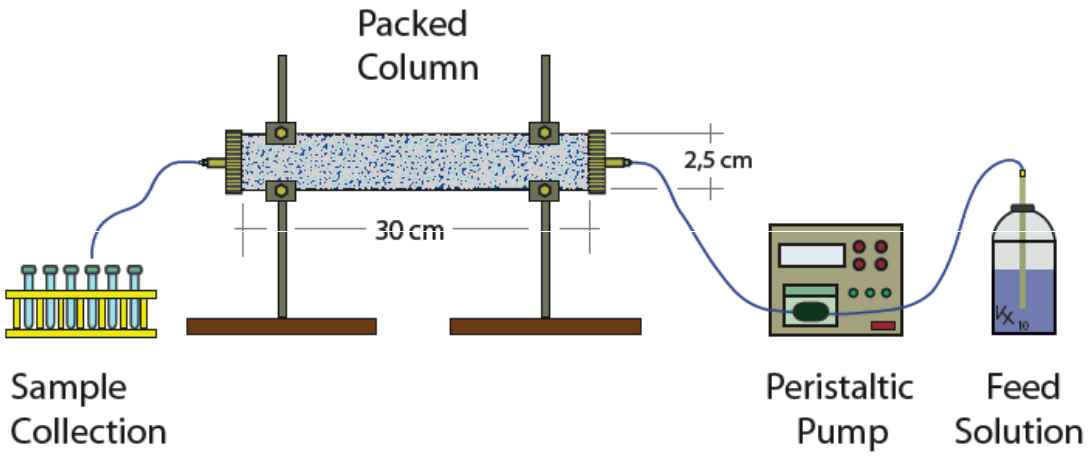
Πίνακας 8.1. Φυσικοί παράμετροι του μαθηματικού μοντέλου για μεταφορά ιχνηθέτη (KCl) και βακτηρίων (P. putida). Στις παραμέτρους που υπολογίστηκαν με τη διαδικασία του Fitting παρουσιάζονται μαζί και τα 95% διαστήματα εμπιστοσύνης.

Parameter	Status	KCl	Status	*P. putida*
D_x	Fitted	26.3±10 cm^2/h	Fitted	14.1±7.91 cm^2/h
$D_y = D_z$	Fitted	1.46±0.84 cm^2/h	Fitted	0.97±0.37 cm^2/h
α_x	Estimated	8.46 cm	Estimated	4.54 cm
$\alpha_{Ty}=\alpha_{Tz}$	Estimated	0.45 cm	Estimated	0.31 cm
F	Fixed	20.4mg/(h·cm^3)	Fixed	1.8 mg/(h·cm^3)
r_1	Fixed	0 1/h	Fixed	0.78 1/h
r_2	Fixed	0 1/h	Fitted	0.26±0.14 1/h
U	Fitted	3.1±1.94 cm/hr	Fixed	3.1 cm/hr
θ	Fixed	0.38	Fixed	0.38
λ	Fixed	0 1/h	Fixed	0.0109 1/h
λ^*	Fixed	0 1/h	Fitted	0.00545 1/h
ρ_b	Fixed	1.63 kg/L	Fixed	1.63 kg/L

Δεύτερο πείραμα: Συμμεταφορά Ιών με άργιλο στις τρείς διστάσεις.

Πέρα από την απλή μεταφορά βακτηρίων και ιών, τα τελευταία χρόνια έχει αρχίσει να μελετάται και το φαινόμενο συμμεταφοράς των παραπάνω οργανικών σωματιδίων μαζί με κολλοειδή. Όπως αναφέρθηκε και στο κεφ. 3.1 τα κολλοειδή είναι σε θέση να διευκολύνουν δραματικά την μεταφορά των ιών κάτι που τα καθιστά επικίνδυνα για την δημόσια υγεία. Με αυτήν την σκέψη παρακάτω θα παρουσιαστεί όλη η διαδικασία του Fitting που έγινε σε πείραμα συμμεταφοράς από την βιβλιογραφία (Πανεπιστήμιο Πατρών Εργαστήριο "Τεχνολογία του Περιβάλλοντος", Τμήμα Πολιτικών Μηχανικών).

Το πείραμα περιείχε ταυτόχρονη μεταφορά ιών (bacteriphages MS2) και αργίλου (Kaolinite KGa-1b). Οι ιοί ανήκουν στην κατηγορία F-Specific με μια έλικα RNA και μέγεθος περίπου 24-26 nm. Αντίστοιχα η άργιλος KGa-1b προέρχεται από την περιοχή Washington Country, Georgia και είναι καλά κρυσταλλοποιημένη (Syngouna and Chrysikopoulos, 2011). Στην συνέχεια δημιουργήθηκε μίγμα ιού με την άργιλο αραιώθηκε κατάλληλα και με την χρήση μιας περισταλτικής αντλίας εισήλθε σε οριζόντια στήλη μήκους 30 cm. Μέσα στην στήλη αυτή δημιουργήθηκε πορώδες προσθέτοντας σφαίρες γυαλιού (d=2 mm) και υπολογίστηκε πειραματικά η τιμή του. Στο τέλος έγινε δειγματοληψία από την έξοδο της στήλης και υπολογίστηκαν οι πυκνότητες του διαλυμένου Ιού, της διαλυμένης αργίλου αλλά και η πυκνότητα των προσροφημένων ιών επάνω στην άργιλο. Το πείραμα αυτό έγινε τρεις φορές για τρεις διαφορετικές παροχές Σχήμα 8.7.
Ο υπολογισμός του πορώδους έγινε με τα εξής βήματα 1) Ζυγίστηκε η στήλη περιέχοντας μόνο σφαίρες γυαλιού, 2) Ζυγίστηκε η στήλη περιέχοντας σφαίρες γυαλιού αλλά και νερό, 3) Γίνεται η διαφορά των παραπάνω δύο μαζών ώστε να υπολογιστεί το βάρος του νερού που γέμιζε τα κενά του πορώδους και 4) Με χρήση της πυκνότητας του νερού υπολογίστηκε ο όγκος των κενών και συνεπώς το πορώδες.



Σχήμα 8.7. Πειραματική διάταξη που χρησιμοποιήθηκε για την συμμεταφορά Ιών - Αργίλου (Syngouna and Chrysikopoulos, 2011).

Έχοντας τα αποτελέσματα από το πείραμα συμμεταφοράς καθώς και τις οριακές συνθήκες αυτού, δημιουργήθηκε το αριθμητικό μοντέλο που θεωρητικά υπολογίζει τις συγκεντρώσεις στον χρόνο. Οι εξισώσεις που το αποτελούν είναι οι εξ. (8.10-8.15). Αυτές λύνονται με την σειρά όπως παρουσιάζονται, σε συστήματα των 2x2 . Δηλαδή πρώτα οι εξ. (8.10-8.11) στην συνέχεια οι εξ. (8.12-8.13) με την βοήθεια των εξ. (3.17-3.22) και στο τέλος λύνονται οι εξ. (8.14-8.15). Βέβαια απαραίτητες για την οποιαδήποτε επίλυση είναι και οι συνοριακές συνθήκες εξ. (8.16-8.18).

$$\frac{\partial C_c}{\partial t}+\frac{\rho_m}{\theta}\frac{\partial C_c^*}{\partial t}=D_c\frac{\partial^2 C_c}{\partial x^2}-U\frac{\partial C_c}{\partial x}-\lambda C_c-\lambda^*\frac{\rho_m}{\theta}C_c^*$$

(8.10)

$$\frac{\rho_m}{\theta}\frac{\partial C_c^*}{\partial t}=r_{c-c^*}C_c-\frac{\rho_m}{\theta}r_{c^*-c}C_c^*-\lambda^*\frac{\rho_m}{\theta}C_C^* \qquad (8.11)$$

$$\frac{\partial(C_cC_{vc})}{\partial t}=\Lambda_{v-vc}-\Lambda_{vc-v}+\Lambda_{v^*c^*-vc}-\Lambda_{vc-v^*c^*}-\lambda_{vc}C_cC_{vc}-\lambda_{vc}^*\frac{\rho_m}{\theta}C_c^*C_{vc}^* \qquad (8.12)$$

$$\frac{\rho_m}{\theta}\frac{\partial(C_c^*C_{vc}^*)}{\partial t}=\Lambda_{v-v^*c^*}-\Lambda_{v^*c^*-v}+\Lambda_{vc-v^*c^*}-\Lambda_{v^*c^*-vc}-\lambda_{vc}^*\frac{\rho_m}{\theta}C_c^*C_{vc}^* \qquad (8.13)$$

$$\frac{\partial}{\partial t}\left(C_v+\frac{\rho_m}{\theta}C_v^*+C_cC_{vc}+\frac{\rho_m}{\theta}C_c^*C_{vc}^*\right)=D_v\frac{\partial^2 C_v}{\partial x^2}+D_{vc}\frac{\partial^2}{\partial x^2}(C_cC_{vc})-U\frac{\partial}{\partial x}(C_v+C_cC_{vc})$$

$$-\lambda_vC_v-\lambda_{vc}C_cC_{vc}-\lambda_v^*\frac{\rho_m}{\theta}C_v^*-\lambda_{vc}^*\frac{\rho_m}{\theta}C_c^*C_{vc}^* \qquad (8.14)$$

$$\frac{\rho_m}{\theta}\frac{\partial C_c^*}{\partial t}=r_{v-v^*}C_v-\frac{\rho_m}{\theta}r_{v^*-v}C_v^*-\lambda^*\frac{\rho_m}{\theta}C_v^* \qquad (8.15)$$

$$C(0,x) \qquad (8.16)$$

$$C(t,0)=C_0 \qquad (8.17)$$

$$\frac{\partial C(t,L_x)}{\partial x}=0 \qquad (8.18)$$

Όπου οι εξισώσεις (8.10-8.11) περιγράφουν την απλή μεταφορά του αργίλου, οι εξ. (8.12-8.13) ορίζουν το ρυθμό δημιουργίας του συμπλόκου ιός-άργιλος (ιός προσροφημένος πάνω στην επιφάνεια της αργίλου) και ιός-άργιλος προσροφημένος επάνω στις σφαίρες γυαλιού, ενώ οι εξ. (8.14-8.15) περιγράφουν την συμμεταφορά ιού αργίλου. Με την σειρά τους οι οριακές συνθήκες εξ. (8.16-8.18) ορίζουν ότι στην αρχή του χρόνου στο πορώδες υπήρχε απουσία ιών και αργίλου, στο σημείο x=0 εισάγεται ο ρύπος (μίγμα ιού αργίλου) και για όσο διαρκεί η παροχή η συγκέντρωση εκεί είναι σταθερή, τέλος στην έξοδο της στήλης ($x = L_x$) η μεταβολή της συγκέντρωσης είναι μηδέν που σημαίνει ότι ο ρύπος που εξέρχεται αδυνατεί με οποιαδήποτε τρόπο να ξανά εισέρθει στο πορώδες.

Στο πίνακα 8.2 δίδονται οι βασικοί παράμετροι του πειράματος συμμεταφοράς, για τις τρείς διαφορετικές παροχές. Οι πιο σημαντικοί από αυτούς είναι οι αρχικές συγκεντρώσεις των διαλυμένων ιών **Cinv** και της αργίλου **Cinc** που εισέρχονται στην στήλη. Εξίσου σημαντική είναι και η διάρκεια κατά την οποία έχουμε εισροή ρύπων, καθώς ύστερα από χρόνο tp η πηγή γίνεται ανενεργή και εισέρχεται μόνο καθαρό νερό στο πορώδες.

Πίνακας 8.2. Φυσικοί παράμετροι του μαθηματικού μοντέλου συμμεταφοράς.

Parameter	Exp 1.	Exp 2.	Exp 3.	Περιγραφή όρων
Cinv (PFU/mL)	51500	2425	4738	Αρχική συγκέντρωση ιών.
Cinc (mg/L)	62,838	69,122	63,781	Αρχική συγκέντρωση αργίλου.
Q (mL/min)	0,8	1,5	2,5	Παροχή νερού.
ρ (mg/cm^3)	1610	1610	1610	Πυκνότητα υδροφορέα.
θ	0,42	0,42	0,42	Πορώδες.
tp (min)	190	117	75	Διάρκεια παροχής μάζας .
U (cm/min)	0,38	0,74	1,21	Ενδοπορώδης ταχύτητα.
lxx (cm)	30	30	30	Μήκος στήλης.
nx (cells)	500	500	500	Μέγεθος διακριτοποίησης στήλης.

Διαδικασία και Αριθμός Παραμέτρων για Fitting

Η σειρά της επίλυσης δείχνει ότι οι εξ. (8.10, 8.11) δεν είναι ανεξάρτητες από οποιαδήποτε άλλη εξίσωση και συνεπώς μπορεί να θεωρηθεί ότι η μεταφορά της αργίλου είναι όντως ένα ξεχωριστό πείραμα. Άρα είναι δυνατό να γίνει Fit σε τρείς παραμέτρους από τις συνολικά δέκα που εισέρχονται στις εξ. (8.10, 8.11).
Με την ίδια λογική οι εξισώσεις (8.12, 8.13) είναι ανεξάρτητες από τις εξ. (8.14, 8.15) Οι όροι Cvc και Cvc* όταν σχηματίζονται, αγνοούν τελείως την ύπαρξη των Cv και Cv*. Αυτό επιβάλεται από τις εξισώσεις (8.12, 8.13) στις οποίες δεν υπάρχει πουθενά ο όρος Cv ή Cv* . Οι εξ. (8.12, 8.13) έχουν δοθεί από βιβλιογραφία (κεφ. 3).

Συνεπώς μπορεί να γίνει Fit σε ακόμα 3 παραμέτρους από τις συνολικά 15 παραμέτρους που εισάγουν οι εξ. (8.12, 8.13).
Παρομοίως με την ίδια σκέψη και οι εξ. (8.14, 8.15) εισάγουν 7 παραμέτρους από τις οποίες οι 3 μπορούν να γίνουν fitting.
Έτσι ο μέγιστος αριθμός παραμέτρων που μπορεί να γίνει Fit είναι 9. Δηλαδή 3 για κάθε ξεχωριστό διάγραμμα (Cc, Cv και Cvc).

Οι τιμές που έγιναν τελικά fit φαίνονται στον παρακάτω πίνακα **8.3** και είναι χρωματισμένες. Με πράσινο χρώμα είναι το fit για τα κολλοειδή Cc, γαλάζιο οι ιοί Cv και κόκκινο είναι οι προσροφημένοι ιοί επάνω στα κολλοειδή. Όμως και πάλι οι εννιά διαθέσιμες παράμετροι για Fit δεν είναι αρκετοί για να περιγραφεί συνολικά το μοντέλο καθώς πολλές σταθερές ακόμα είναι πρακτικά άγνωστες. Γι αυτόν το λόγο προστέθηκαν και οι πειραματικές εξισώσεις (8.19-8.26) οι οποίες ισχύουν και είναι κοντά στην πραγματικότητα. Για παράδειγμα η εξ. (8.19) απορρέει από ότι το σύμπλοκο ιού με άργιλο έχει περίπου το ίδιο μέγεθος με τον απλό άργιλο γι αυτό και θεωρείται ότι έχουν παρόμοια υδροδυναμική διασπορά. Η εξ. (8.23) προέρχεται από πειραματικά δεδομένα και φαίνεται να ισχύει για ένα ευρύ φάσμα αρχικών συγκεντρώσεων στο οποίο ανήκουν και οι συγκεντρώσεις του πειράματος συμμεταφοράς που προσομοιώνεται. Μόνο οι όροι R_{c-c^*}, R_{v-v^*} και R_{vc^*-v} δεν έγιναν fit ούτε και προέρχονται από καμία άλλη σχέση, οπότε οι τιμές τους δόθηκαν από την βιβλιογραφία.

Επιπλέον σχέσεις που ισχύουν εξ. (8.19-8.26):

$$D_{vcx} = D_{cx} \tag{8.19}$$

$$R_{c^*-c} = R_{vc^*-vc} \tag{8.20}$$

$$R_{c-c^*} = R_{vc-vc^*} \tag{8.21}$$

$$R_{v-vc} = R_{v-vc^*} \tag{8.22}$$

$$C_{vce\,q^*} = \frac{1}{3} C_{vceq} \tag{8.23}$$

$$L_c = 2\,L_c^* \tag{8.24}$$

$$L_{vc} = L_c \tag{8.25}$$

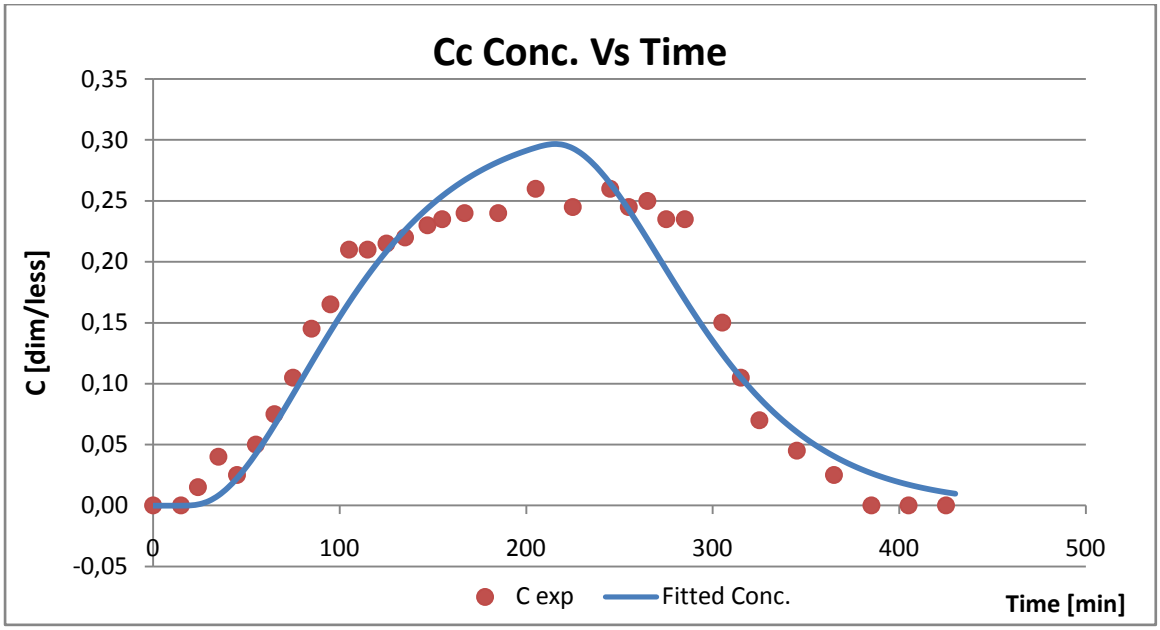
$$L_{vc} = 2\,L_{vc}^* \tag{8.26}$$

Πίνακας 8.3. Φυσικοί παράμετροι του μαθηματικού μοντέλου μεταφοράς και συμμεταφοράς.

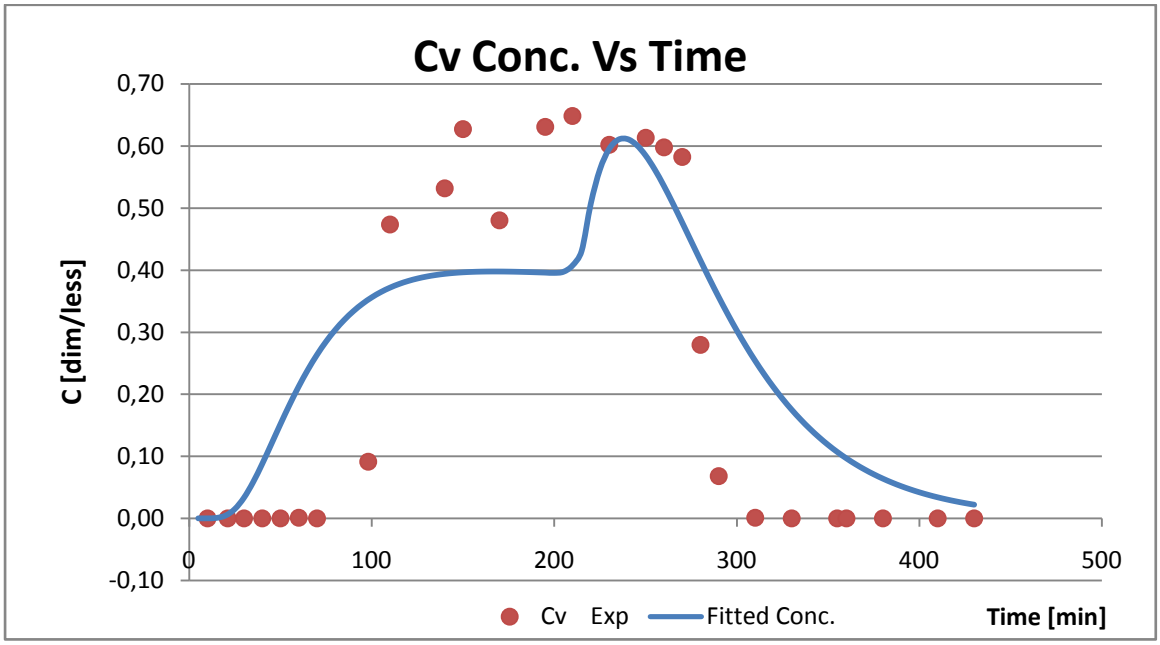
	Transport	Exp. 1	Exp. 2	Exp. 3	
1	D_{cx}	1,080	2,200	3,500	cm^2/min
2	D_{cy}	0	0	0	cm^2/min
3	D_{cz}	0	0	0	cm^2/min
4	U	0,380	0,740	1,210	cm/min
5	L_c	0,012	0,014	0,026	1/min
6	L_c^*	0,006	0,007	0,013	1/min
7	θ	0,420	0,420	0,420	
8	R_{c-c^*}	0,073	0,855	1,954	1/min
9	R_{c^*-c}	0,068	2,328	3,001	1/min
10	ρ (mg/cm^3)	1610	1610	1610	(mg/cm^3)

	Cootranport	Exp. 1	Exp. 2	Exp. 3	
1	D_{vx}	2,3802	4,5	7,3	cm^2/min
2	D_{vy}	0	0	0	cm^2/min
3	D_{vz}	0	0	0	cm^2/min
4	D_{vcx}	1,080	2,2	3,5	cm^2/min
5	D_{vcy}	0	0	0	cm^2/min
6	D_{vcz}	0	0	0	cm^2/min
7	L_v	0,002	0,034	0,102	1/min
8	L_v^*	0,001	0,017	0,051	1/min
9	L_{vc}	0,012	0,014	0,026	1/min
10	L_{vc}^*	0,006	0,007	0,013	1/min
11	R_{v-v^*}	0,946	1,880	3,382	1/min
12	R_{v^*-v}	2,281	0,659	2,836	1/min
13	R_{v-vc}	0,002	0,018	0,099	1/min
14	R_{vc-v}	0,031	0,010	0,010	1/min
15	R_{v-vc^*}	0,002	0,018	0,099	1/min
16	R_{vc-vc^*}	0,072	0,855	1,954	1/min
17	R_{vc^*-v}	0,010	1,1E-04	4,1E-05	1/min
18	R_{vc^*-vc}	0,069	2,328	3,001	1/min
19	C_{vceq}	7,100	1,687	3,902	ml water/mg clay
20	$C_{vce\,q^*}$	2,350	0,595	1,300	ml water/mg clay

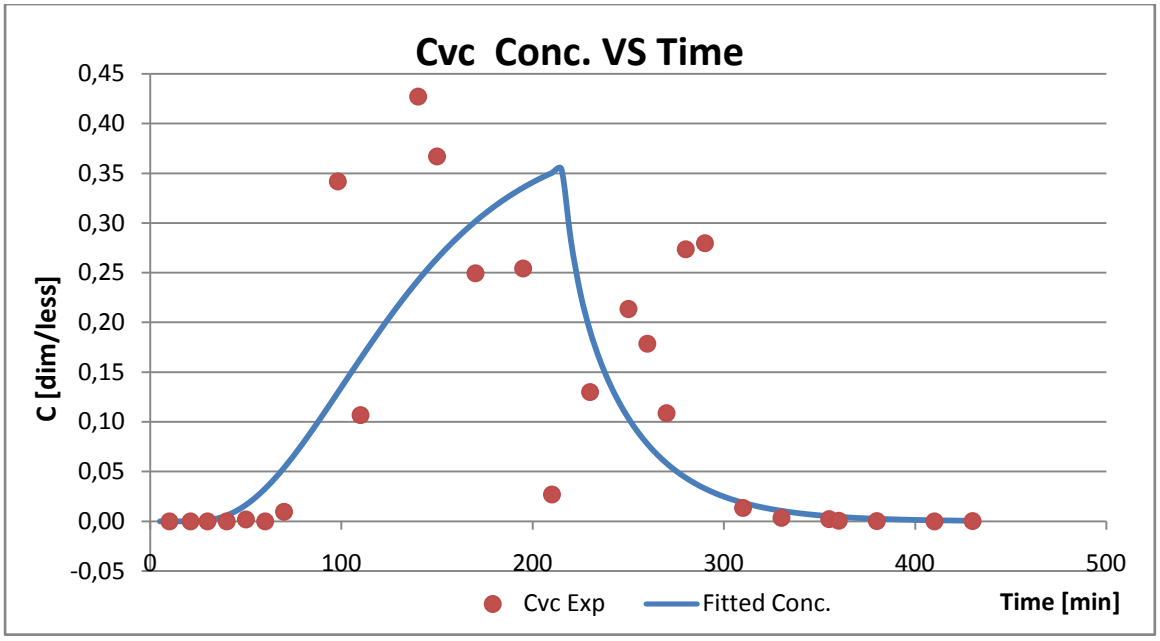
Στην συνέχεια ακολουθούν διαγράμματα Σχήματα 8.8-8.16 συμμεταφοράς τα οποία προέρχονται από τα δεδομένα των πινάκων 8.2, 8.3 και περιέχουν τόσο τα αποτελέσματα της μαθηματικής προσομοίωσης όσο και τις πειραματικές τιμές του εργαστηρίου.



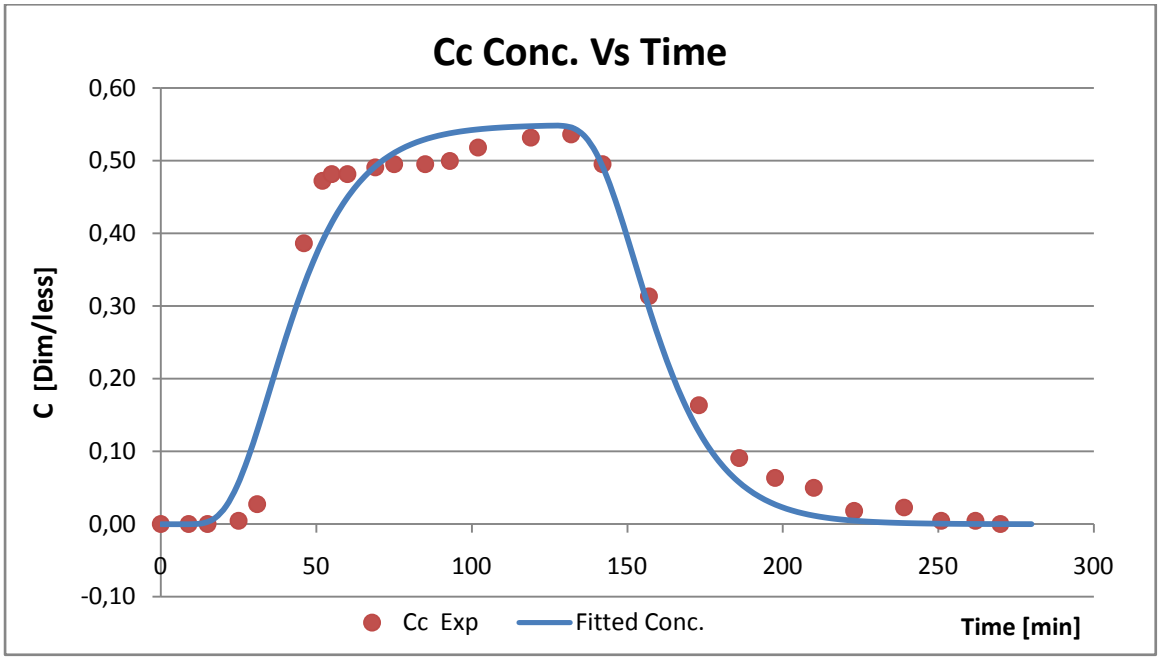
Σχήμα 8.8. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση της αργίλου με παροχή Q=0.8 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



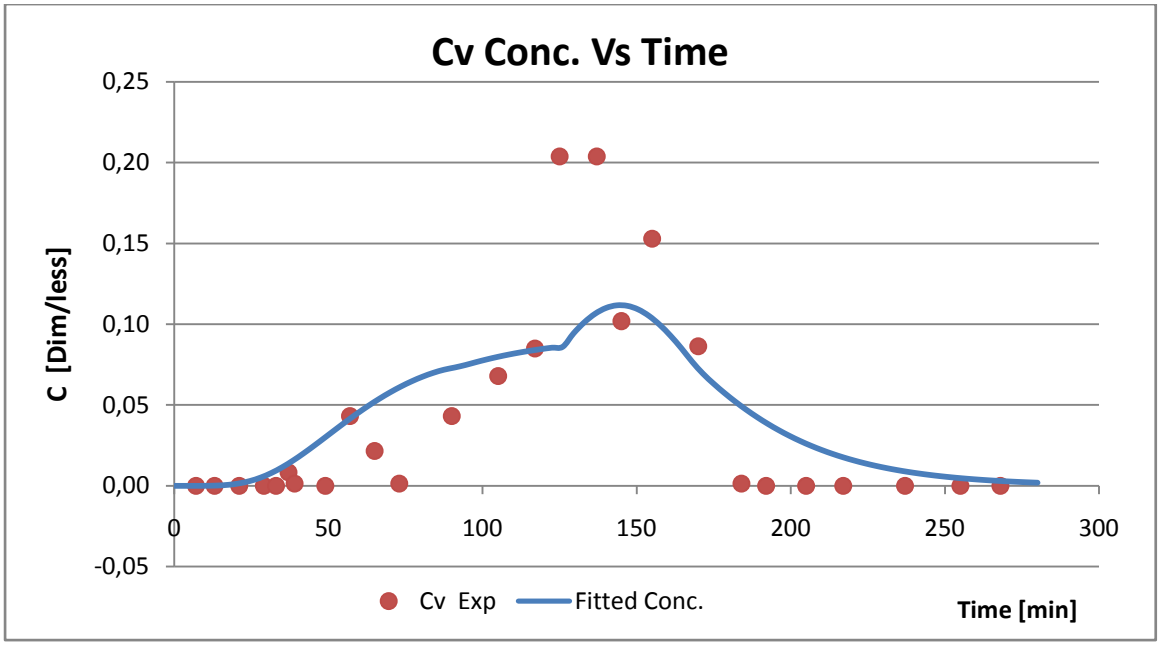
Σχήμα 8.9. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση του ιού με παροχή Q=0.8 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



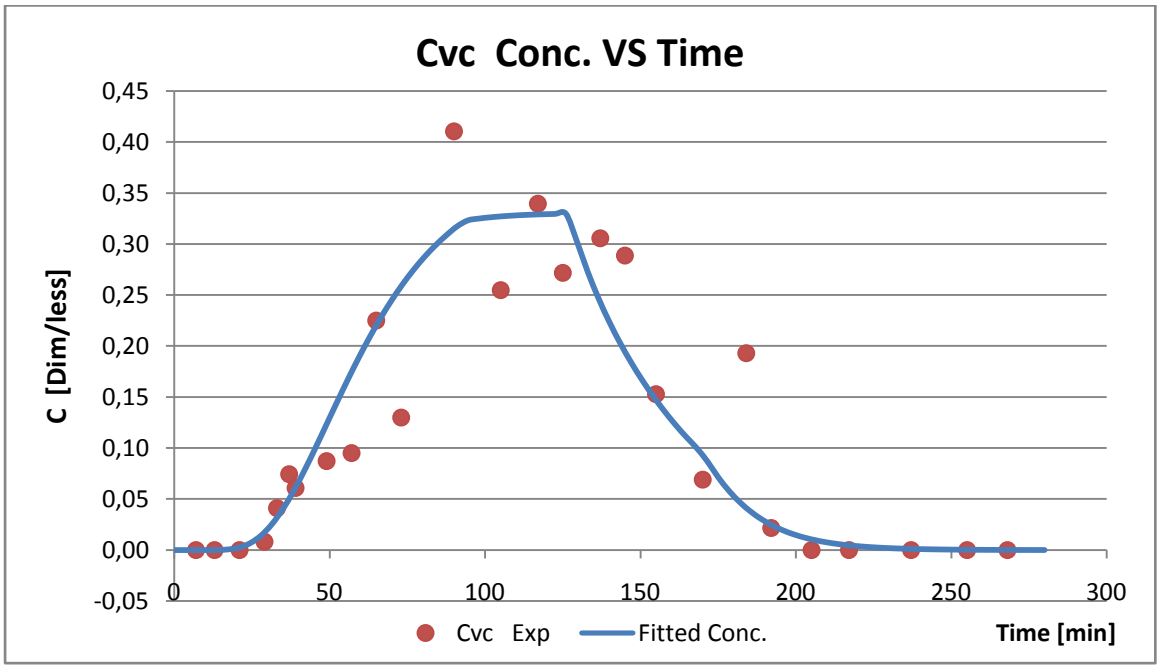
Σχήμα 8.10. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση του συσσωματώματος ιού-αργίλου με παροχή Q=0.8 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



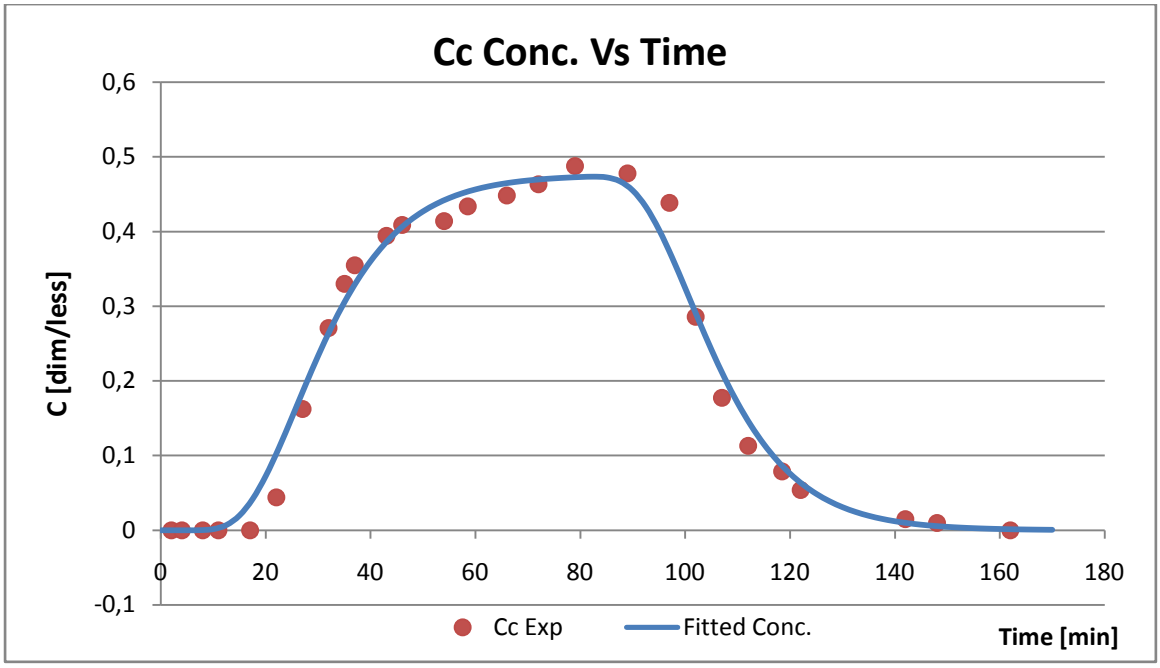
Σχήμα 8.11. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση της αργίλου με παροχή Q=1.5 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



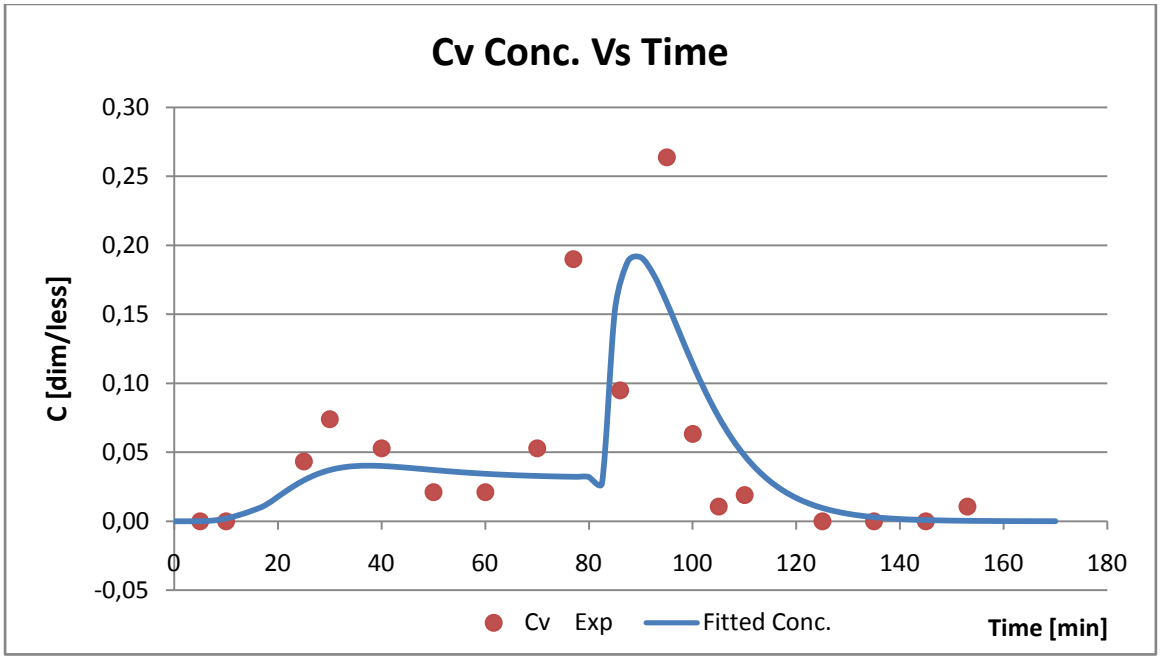
Σχήμα 8.12. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση του ιού με παροχή Q=1.5 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



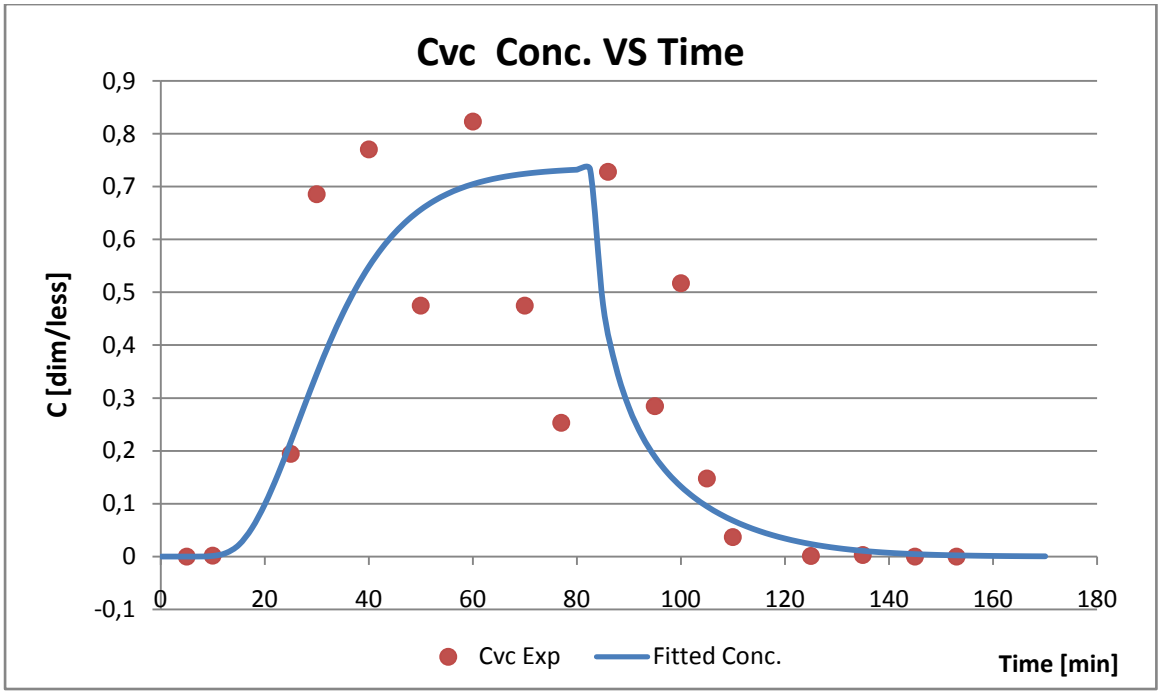
Σχήμα 8.13 Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση του συσσωματώματος ιού-αργίλου με παροχή Q=1.5ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



Σχήμα 8.14. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση της αργίλου με παροχή Q=2.5 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.

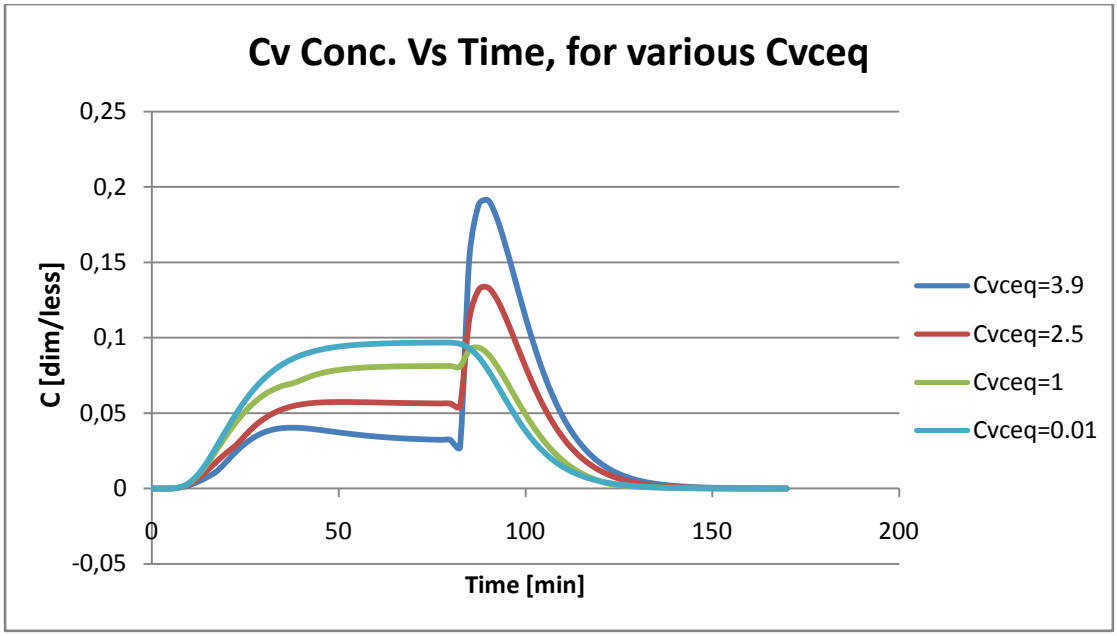


Σχήμα 8.15. Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση του ιού με παροχή Q=2.5 ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.



Σχήμα 8.16 Παρουσιάζονται τόσο τα πειραματικά δεδομένα (κόκκινοι κύκλοι) όσο και τα αποτελέσματα από το Fitting (μπλε καμπύλη) για την περίπτωση του συσσωματώματος ιού-αργίλου με παροχή Q=2.5ml/min. Οι τιμές του χρόνου δίνονται σε min, ενώ αυτές των συγκεντρώσεων είναι αδιάστατες.

Όπως παρατηρείται στα παραπάνω Σχήματα 8.8-8.16 το μοντέλο μας έχει καταφέρει πολύ καλά να περιγράψει το πείραμα συμμεταφοράς. Κρίνεται χρήσιμο να δοθεί μια εξήγηση για το φαινόμενο στο Σχήμα 8.16. Πιο συγκεκριμένα στο σχήμα αυτό φαίνεται μια αρχική αύξηση της συγκέντρωσης του ιού και στην συνέχεια μια μείωση της όπως ακριβώς είναι αναμενόμενο. Αυτό που δεν είναι αναμενόμενο είναι ότι στην συνέχεια στον χρόνο t=85 min υπάρχει μια απότομη αύξηση της ενώ εμείς αντίθετα ήδη από t=75 min έχουμε σταματήσει την παροχή οποιαδήποτε ρύπου στην στήλη. Φαίνεται περίεργο να συμβαίνει κάτι τέτοιο παρ όλα αυτά έχει μια λογική εξήγηση. Ο ιός όταν για πρώτη φορά μπαίνει στην στήλη προσροφάται επάνω στην άργιλο που βρίσκει σχηματίζοντας σύμπλοκα ιού-αργίλου Cvc μέχρι να φτάσει την μέγιστη τιμή Cvceq (κατάστασης ισορροπίας). Όμως έρχεται η ώρα που η παροχή αργίλου σταματάει, τότε αρχίζει με έντονο ρυθμό η αντίστροφη διαδικασία και ο ιός αποκολλείται από την άργιλο σχηματίζοντας όρους Cv με ρυθμό R_{vc-v}. Αυτός είναι και ο λόγος που ενώ έχει σταματήσει η παροχή ιού, η συγκέντρωση του αυξάνεται. Η εξήγηση αυτή στηρίζεται και από το Σχήμα 8.17, το οποίο δείχνει πως μεταφέρεται ο ιός για διαφορετικές τιμές της παραμέτρου Cvceq (μέγιστη συγκέντρωση του συμπλόκου ιού-αργίλου στην κατάσταση ισσοροπίας). Όπως παρατηρείται καθώς η μέγιστη τιμή της Cvc μειώνεται τόσο και το φαινόμενο της απότομης αύξησης εξασθενεί.



Σχήμα 8.17. Παρουσιάζονται διαγράμματα συγκεντρώσεων ιού (παροχή 2.5 ml/min) για διαφορετικές τιμές της παραμέτρου Cvceq. Τα υπόλοιπα δεδομένα δίνονται στους πίνακες 8.2, 8.3 για παροχή Q=2.5 ml/min (Exp. 3).

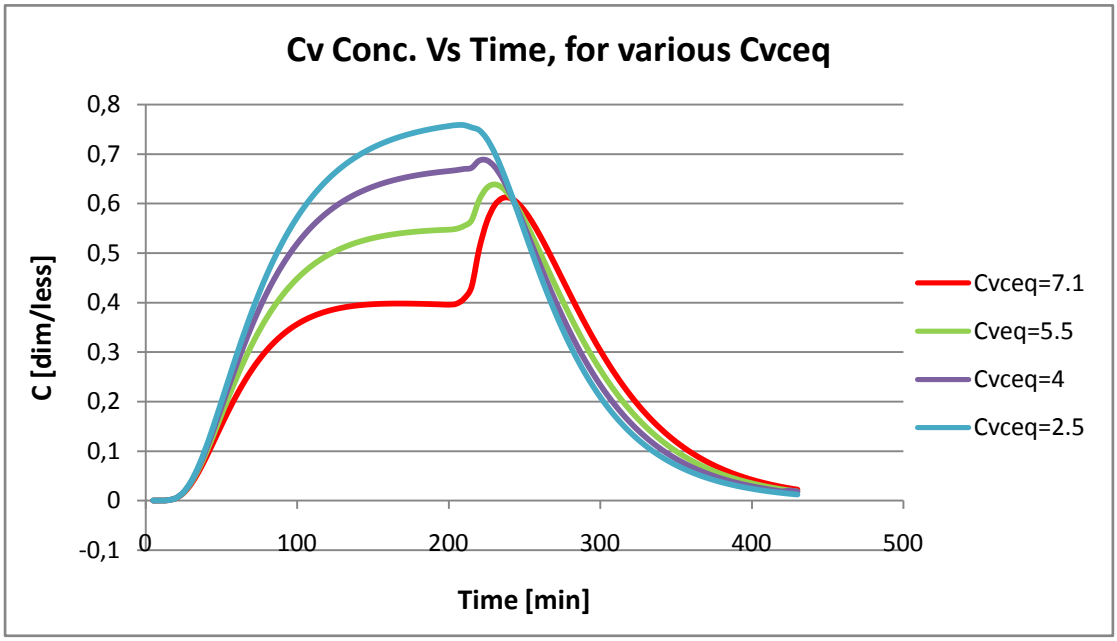
8.2 ΑΝΑΛΥΣΗ ΕΥΑΙΣΘΗΣΙΑΣ

Ορισμός:

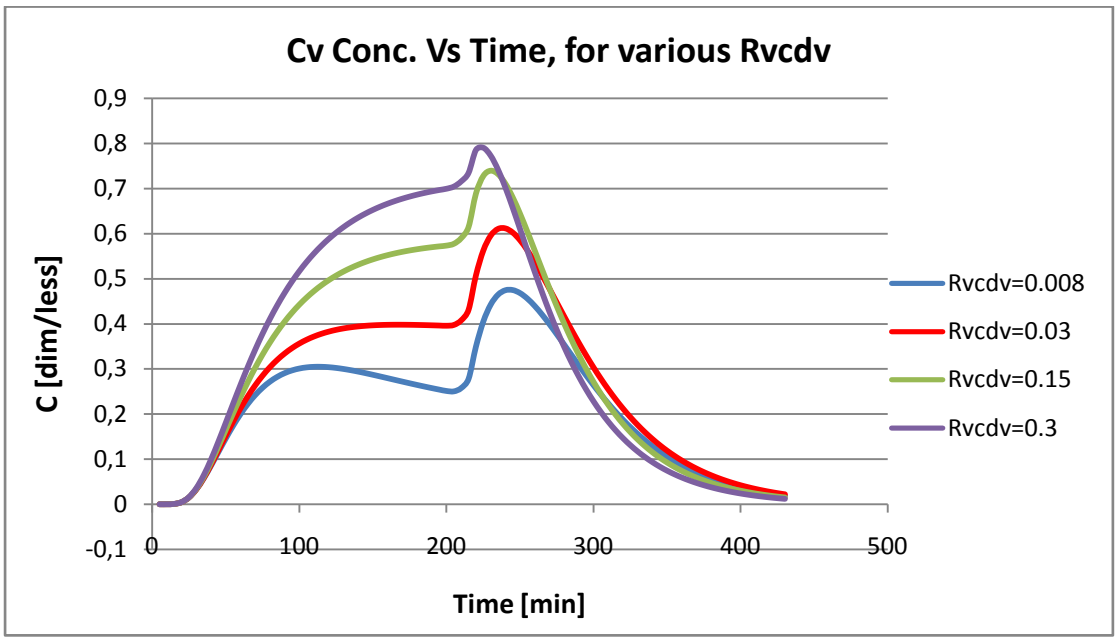
Η μέθοδος ή τεχνική μεταβολής των τιμών ορισμένων επιλεκτικών μεταβλητών, για να διερευνηθεί η επίπτωσή τους σε ορισμένες άλλες μεταβλητές που έχουν εξέχουσα σημασία στην αξιολόγηση ή επιλογή αποφάσεων (http://www.infosoc.gr/infosoc/el-GR/services/leksiko/332.htm).

Πιο συγκεκριμένα ένας μηχανικός βλέποντας ένα διάγραμμα συγκέντρωσης χρόνου σε ένα πείραμα μεταφοράς, θα πρέπει να είναι σε θέση να εκτιμήσει ποιοί παράμετροι είναι σημαντικοί σε αυτό και ποιοί δεν είναι. Έτσι ώστε εάν θελήσει να ενισχύσει την απομάκρυνση του ρύπου να ξέρει ποιούς παράγοντες να επηρεάσει. Για παράδειγμα σε πείραμα όπου το φαινόμενο της διασποράς είναι ο κύριος μηχανισμός μεταφοράς, η παραμικρή αύξηση της οριζόντιας ταχύτητας μπορεί να έχει πολύ σημαντικά αποτελέσματα στην αφαίρεση του ρύπου. Αντίθετα εάν η μεταγωγή είναι το κυρίως φαινόμενο όσο και να αυξηθεί την ταχύτητα, δραματικά αποτελέσματα δεν θα υπάρξουν. Γι αυτόν το λόγο θα παρουσιαστούν παρακάτω διαγράμματα Σχήμα 8.18-8.25 στα οποία θα φαίνονται οι μεταβολές στην συγκέντρωση που προκαλούνται από αλλαγές στις τιμές των σημαντικότερων παραμέτρων.

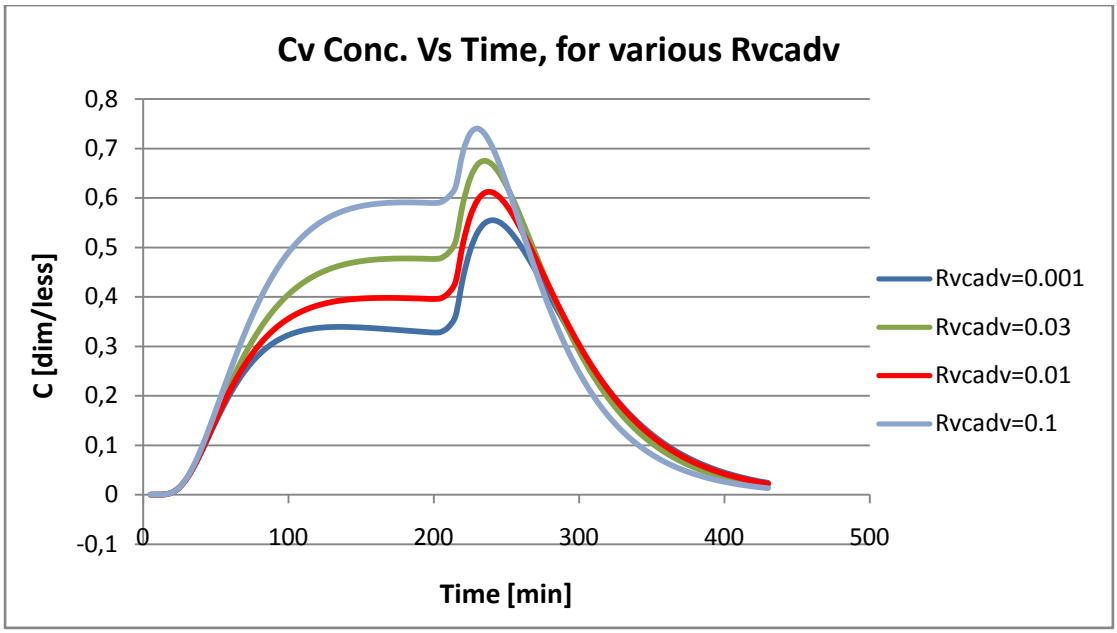
Το σύνολο των δεδομένων για την αριθμητική προσομοίωση δίδονται στου πίνακες 8.2, 8.3 για την περίπτωση συμμεταφοράς με παροχή 0.8 ml/min (Exp. 1).



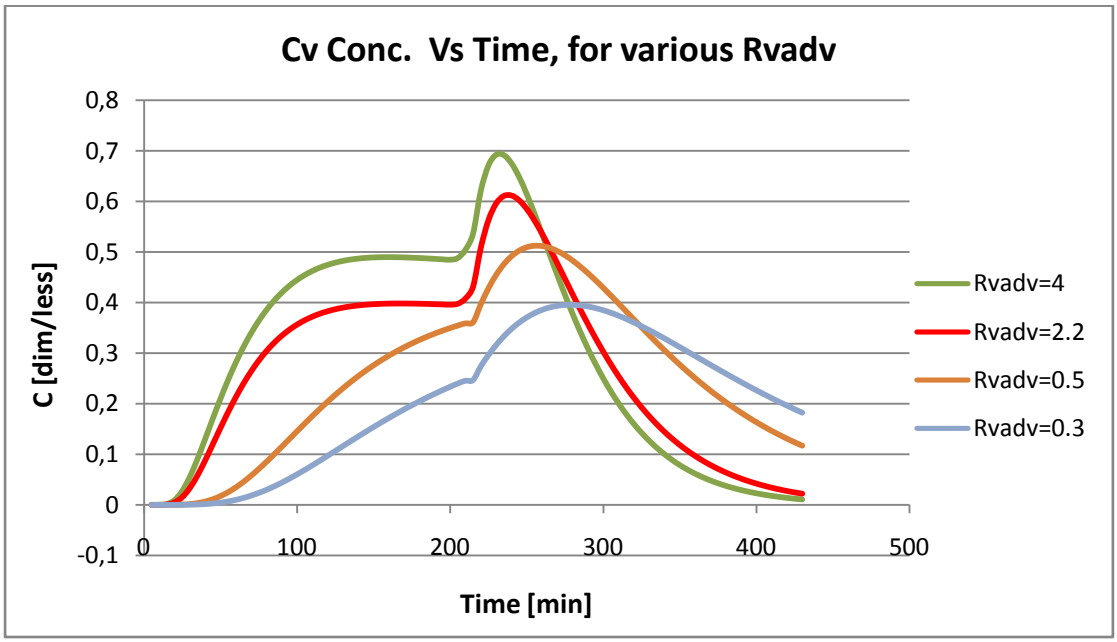
Σχήμα 8.18. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Cvceq**. Όπου **Cvceq η** μέγιστη συγκέντρωση του συμπλόκου ιού-αργίλου στην κατάσταση ισορροπίας. Ταυτόχρονα ισχύει ότι $C_{vce\,q^*} = \frac{1}{3}\, C_{vceq}$.



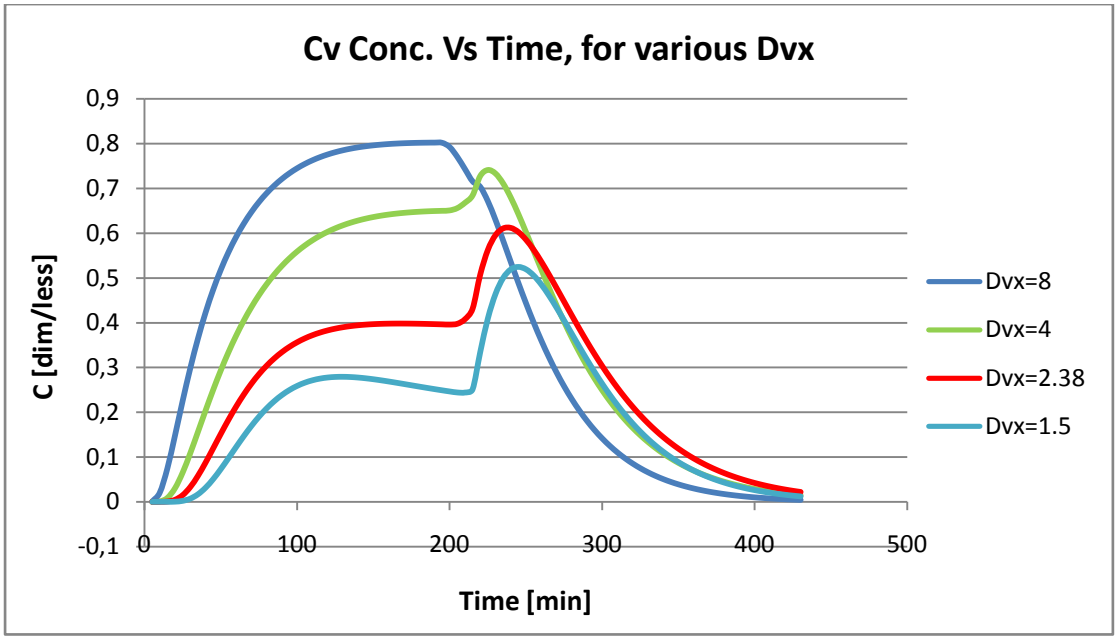
Σχήμα 8.19. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Rvcdv**. Όπου **Rvcdv=R_{vc-v}** είναι ο ρυθμός μετατροπής του συμπλόκου ιού-αργίλου σε διαλυμένο ιό.



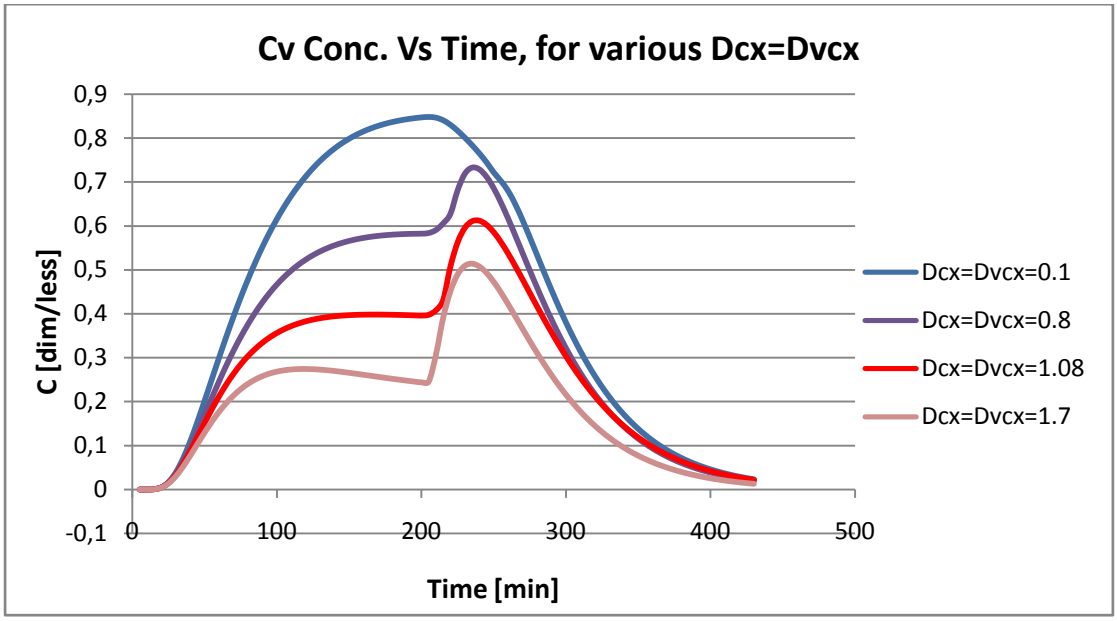
Σχήμα 8.20. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Rvcdv**. Όπου **Rvcadv=**R_{vc^*-v} είναι ο ρυθμός μετατροπής του συμπλόκου ιού-αργίλου, προσροφημένου στο στερεό πορώδες, σε διαλυμένο ιό.



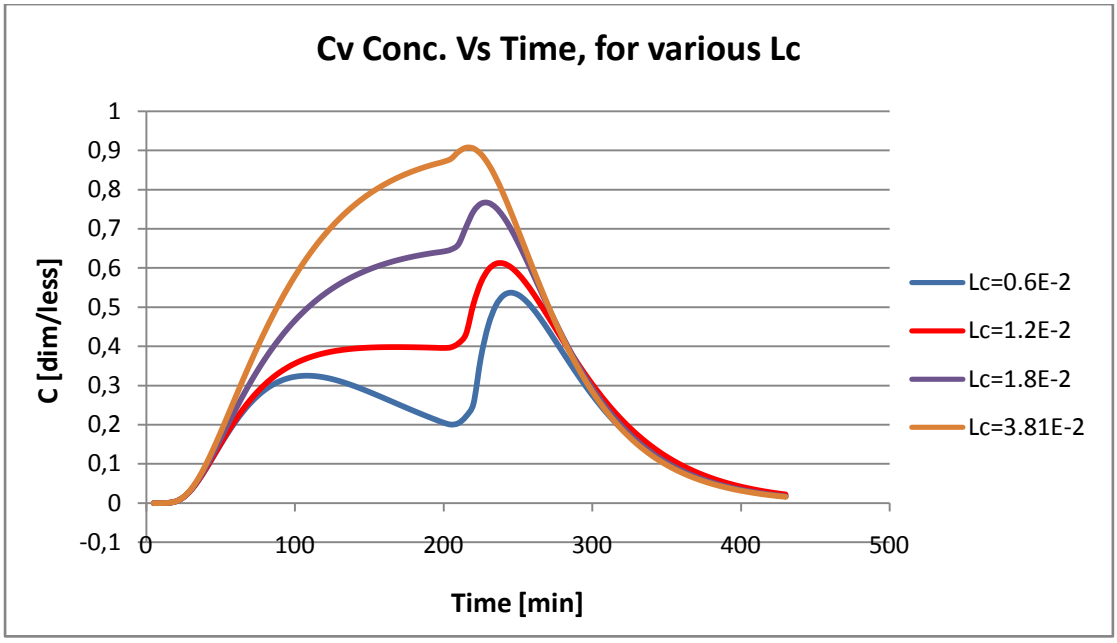
Σχήμα 8.21. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Rvadv**. Όπου **Rvadv=**R_{v^*-v} είναι ο ρυθμός μετατροπής του προσροφημένου ιού στο στερεό πορώδες, σε διαλυμένο ιό.



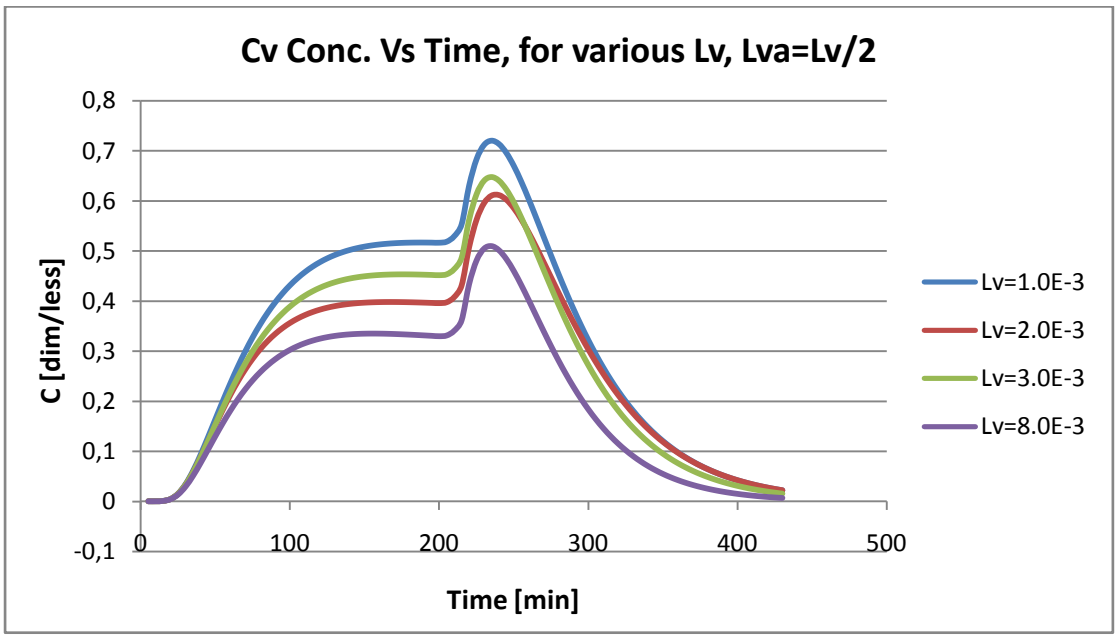
Σχήμα 8.22. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Dvx**. Όπου **Dvx** είναι ο συντελεστής υδροδυναμικής διασποράς του ιού.



Σχήμα 8.23. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Dvcx**. Όπου **Dvcx** είναι ο συντελεστής υδροδυναμικής διασποράς του σύμπλοκου ιού-αργίλου.



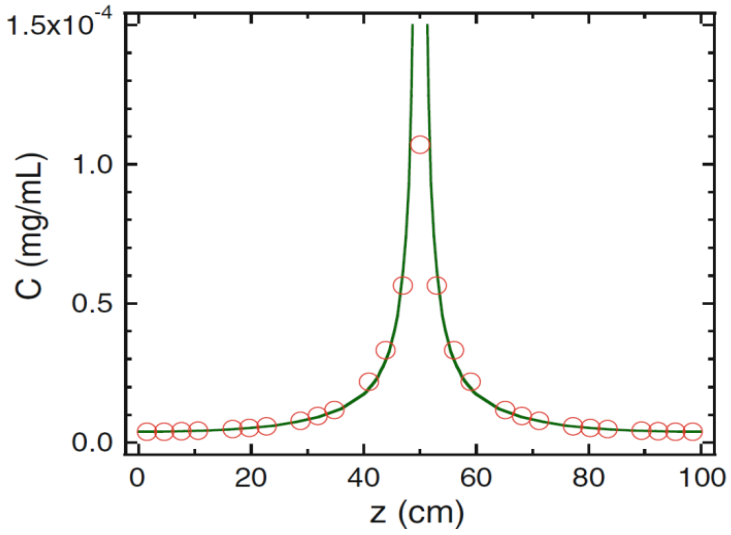
Σχήμα 8.24. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Lc**. Όπου **Lc** είναι ο συντελεστής αποδόμησης της διαλυμένης αργίλου. Ταυτόχρονα ισχύει ότι $L_c^* = \frac{L_c}{2}$.



Σχήμα 8.25. Παρουσιάζονται οι τιμές των συγκεντρώσεων ιού για διαφορετικές τιμές της παραμέτρου **Lv**. Όπου **Lv** είναι ο συντελεστής αποδόμησης του διαλυμένου ιού. Ταυτόχρονα ισχύει ότι $L_v^* = \frac{L_v}{2}$.

8.3 ΣΥΓΚΡΙΣΗ ΑΝΑΛΥΤΙΚΩΝ ΚΑΙ ΑΡΙΘΜΗΤΙΚΩΝ ΜΕΘΟΔΩΝ

Έχοντας στην διάθεση μας τόσο αναλυτικές όσο και αριθμητικές λύσεις για το πρόβλημα της απλής μεταφοράς κεφ. 7 έγινε δυνατή η σύγκριση και η αξιολόγηση του μοντέλου που αναπτύχθηκε. Παρακάτω στο Σχήμα 8.25 παρουσιάζονται ταυτόχρονα οι δύο αυτές λύσεις και εξάγονται χρήσιμα συμπεράσματα. Όπως παρατηρείται όσο πιο μακριά είμαστε από την πηγή (z=50 cm) τόσο καλύτερη ακρίβεια έχουμε. Καθώς πλησιάζουμε φαίνεται κάποια αναντιστοιχία μεταξύ των δύο μεθόδων. Ο λόγος για κάτι τέτοιο έγκειται στην φύση της αναλυτικής λύσης εξ. (4.2). Η συγκεκριμένη εξίσωση θεωρεί ότι για z=50 cm υπάρχει σημειακή πηγή μηδενικών διαστάσεων, και γι αυτό το λόγο η συγκέντρωση της πηγαίνει στο άπειρο. Κάτι τέτοιο εκτός από ότι παραβιάζει την φυσική του προβλήματος, καθώς η μάζα του ρύπου είναι πεπερασμένη, δεν μπορεί να προσομοιωθεί με κανέναν τρόπο από τις αριθμητικές μεθόδους. Βασική αρχή αυτών είναι ότι ο χώρος διακριτοποιείται σε πεπερασμένους όγκους, με αυστηρώς ορισμένες μη μηδενικές διαστάσεις. Συνεπώς αυτή η απόκλιση των αριθμητικών από τις αναλυτικές λύσεις είναι απόλυτα αναμενόμενη και δεν θεωρείται ότι ακυρώνει την ακρίβεια του μαθηματικού μοντέλου.



Σχήμα 8.25. Παρουσιάζεται σύγκριση μεταξύ αναλυτικών και αριθμητικών λύσεων. Οι παράμετροι του αναλυτικού μοντέλου είναι: περιοχή σύγκρισης (x, y, z)=(200, 250 cm, z), για χρόνο t = 5 days, σε τρισδιάστατο περιορισμένο υδροφορέα με πάχος H = 100 cm, για σημειακή πηγή σε θέση x = 200 cm,y = 250 cm, z = 50 cm. Ενώ το αριθμητικό μοντέλο θεώρησε υδροφορέα στο χώρο με αδιαπέρατα τοιχώματα με διαστάσεις Lx = Ly = 500 cm, Lz = H = 100 cm. Η διακριτοποίηση έγινε με αριθμό κελιών nx = ny = 71, nz = 33, και χρονικό βήμα t = 0.25 d. Ταυτοχρόνως για τις δύο λύσεις ισχύουν: παροχή μάζας πηγής F = 4 g/(cm^3 day), διασπορά Dx = 1950 cm^2/d, Dy = Dz = 1450 cm^2/d, ταχύτητα U =15 cm/d, αποδόμηση με ρυθμό $\lambda = \lambda* = 0$ d^{-1}, προσρόφηση με ρυθμό $r_1 = 0.72$ d^{-1}, αποκόλληση με ρυθμό $r_2 = 0.4$ d^{-1} και πορώδες $\theta = 0.25$.

ΣΥΓΚΡΙΣΗ ΠΛΕΟΝΕΚΤΗΜΑΤΩΝ ΑΝΑΛΥΤΙΚΩΝ ΚΑΙ ΑΡΙΘΜΗΤΙΚΩΝ ΜΕΘΟΔΩΝ

ΑΝΑΛΥΤΙΚΕΣ ΜΕΘΟΔΟΙ

1. Ακριβή αποτελέσματα:

   Με εύκολο τρόπο κανείς μπορεί να έχει ακρίβεια 0.001%.

2. Δεν χρειάζεται να βρεθούν όλες οι συγκεντρώσεις του χώρου, ώστε να βρεθεί η συγκέντρωση ενός σημείου.

   Κάθε σημείο του χώρου είναι ανεξάρτητο από κάθε άλλο και έτσι μπορεί και υπολογίζεται ξεχωριστά.

ΑΡΙΘΜΗΤΙΚΕΣ ΜΕΘΟΔΟΙ

1. Γρήγορα αποτελέσματα

   Τα χρονικά βήματα dt υπολογισμού μπορεί να είναι αρκετά μεγάλα και συνεπώς επιτρέπουν την γρήγορη εξαγωγή αποτελεσμάτων.

2. Εύκολη μοντελοποίηση σύνθετων προβλημάτων.

   Κανείς μπορεί να μοντελοποιήσει σύνθετες πηγές, με πολύπλοκο σχήμα, αλλά και με δύσκολες εξισώσεις προσρόφησης, απορρόφησης και αποδόμησης με σχετική ευκολία. Κάτι τέτοιο είναι πολύ δύσκολο εάν όχι αδύνατο για αναλυτικές λύσεις.

3. Επίλυση προβλημάτων για τα οποία δεν υπάρχει αναλυτική λύση.

   Για όλα τα προβλήματα που συναντώνται στην πράξη δεν είναι απαραίτητο να υπάρχουν αναλυτικές λύσεις. Ακόμα όμως και να υπάρχουν τις περισσότερες φορές είναι δύσχρηστες αφού δεν είναι ευέλικτες και δεν προσαρμόζονται σε καινούρια δεδομένα, που ίσως τελικά να προκύψουν στο άμεσο μέλλον.

Συμπέρασμα:

Κρίνεται κατά την γνώμη του συγγραφέα ότι όταν είναι προσβάσιμες οι αναλυτικές λύσεις καλό θα είναι να χρησιμοποιούνται εξαιτίας της ακρίβειας τους αλλά και της δυνατότητας να υπολογίζουν μεμονωμένες συγκεντρώσεις στον χώρο. Παρόλα αυτά στην πράξη αντιμετωπίζονται προβλήματα που οι αναλυτικές λύσεις μόνες τους δεν μπορούν να δώσουν ακριβή αποτελέσματα εξαιτίας της πολυπλοκότητας τους. Τέτοια προβλήματα είναι αυτά που απαιτούν επίλυση πρόσθετων εξισώσεων που υπολογίζουν υδραυλικές κλίσεις, συγχρόνως με τις εξισώσεις που περιγράφουν μεταφορά ρύπου. Σε αυτές τις περιπτώσεις οι αριθμητικές λύσεις είναι μονόδρομος. Άλλωστε σπάνια οι σταθερές (όπως το πορώδες) του φυσικού πεδίου είναι ομοιόμορφες σε όλη την έκταση του και γι αυτό στα πραγματικά προβλήματα που αντιμετωπίζονται σήμερα οι αριθμητικές λύσεις στην πλειοψηφία των περιπτώσεων είναι αν όχι μονόδρομος, σίγουρα η καλύτερη επιλογή.

9. ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Abdel-Salam A. and C.V. Chrysikopoulos (1995a), "Analysis of a model for contaminant transport in fractured media in the presence of colloids", Journal of Hydrology, Vol. 165, pp. 261–281.
2. Anders, R., Chrysicopoulos, C.V., 2005. Virus fate and transport during artificial recharge with recycled water. Water Resources Research 41, W10415. doi:10.1029/2004WR003419
3. Artinger, R., Rabung, T., Kim, J.I., Sachs, S., Schmeide, K., Heise, K.H., Bernhard, G., Nitshe, H., 2002. Humic colloid-borne migration of uranium in sand columns. J. Contam. Hydrol. 58, 1–12.
4. Bear, J., Hydraulics of Groundwater, McGraw-Hill, 569 pp., 1979.
5. Bekhit, H. M., M. A. El-Kordy, and A. E. Hassan (2009), Contaminant transport in groundwater in the presence of colloids and bacteria: Model development and verification, J. Contam. Hydrol., 108, 152 – 167
6. Chen, G., Flury, M., Harsh, J.B., Lichtner, P.C., 2005. Colloid facilitated transport of cesium in variably-saturated Hanford sediments. Environ. Sci. Technol. 39, 3435–3442.
7. Chrysicopoulos, C.V., and Y, Sim, One dimensional virus transport in homogenus porous media with time dependent distribution coefficient, Hournal of Hydrology 185, 199-219, 1996.
8. Chunmiao Zheng and Gordon D. Bennet , Applied contaminant transport modeling , 1995 , ISBN 0-442-01348-5.
9. Chunmiao Zheng and P. Patrick Wang ,1999 , A Modular Three-Dimensional Multispecies Transport Model mt3dms, User guide manual.pdf.
10. Constantinos V. Chrysikopoulos.; Vasiliki I. Syngouna.;
11. Cvetkovic, V., Painter, S., Turner, D., Pickett, D., Bertetti, P., 2004. Parameter and model sensitivities for colloid-facilitated radionuclide transport on the field scale. Water Resour. Res. 40, 1–14.
12. Eichholz, G.G., Wahling, B.G., Powell, G.F., Craft, T.F., 1982. Subsurface migration of radioactive waste materials by particulate transport. Nucl. Technol. 58, 511–520
13. Einstein, A., Investigations on the Theory of the Brownian Movement, Dover, New York 1956.
14. Elimelech, M., J. Gregory, X. Jia, and R.A Williams, Particle Depotision and Aggregation : Measurement, Modeling and Simulation, Buterworth-Heinmann Ltd, Oxford, Great Britain, 1995.
15. Feke D.L., N.D. Prabhu, J.A. Mann, Jr. and J.A. Mann (1984), "A formulation of the short-range repulsion between spherical colloidal particles", Journal of Physical Chemistry, Vol. 8, pp. 5735-5739.

16. Flury, M., Mathison, J.B., Harsh, J.B., 2002. In situ mobilization of colloids and transport of cesium in Hanford sediments. Environ. Sci. Technol. 36, 5335–5341.
17. Gelhar, L.W., C. Welty and K.R. Rehfeldt, A critical review of data on field-scale dispersion in aquifers, Water Resources Research, 28(7), 1955-1974, 1992.
18. Gelhar, L.W., Stochastic subsurface hydrology from theory to applications, Water Resources Research, 22(9), 135S-145S, 1986.
19. Giese , R.F., and C.J. van Oss, Colloid and surface properties of clays and related minerals Marcel Dekker Inc., New York, 119-139, 2002.
20. Gouy, G., Sur la constitution de la charge electriquea la surface, d'um elecrolyte, Journal of Physical Chemistry, 9, 457-467, 1910.
21. Harvey R.W. and Garabedian S.P. (1991), "Use of colloid filtration theory in modelling movement of bacteria through a contaminated sandy aquifer", Environmental Science and Technology, Vol. 25, pp.178-185.
22. Hogg R., Healy T.W. and Fuerstenau D.W. (1966), "Mutual coagulation of colloidal dispersions", Transactions of the Faraday Society, Vol.62, pp. 1638-1651.
Ioanna A. Vasiliadou.; Vasileios E. Katzourakis (2012). Transport of Pseudomonas putida in 3-D Bench Scale Experimental Aquifer. Transp. Porous Med.
23. Israelachvili J.N., "Intermolecular and Surface Forces", 2nd Edition, Academic Press, London, 1992
24. Iwasaki, T., Some notes on sand filtration, Journal of American Water Worksm Association, 29, 1591, 1937
25. Kersting, A.B., Efurd, D.W., Finnegan, D.L., Rokop, D.J., Smith, D.K., Thompson, J.L., 1999. Migration of plutonium in ground water at the Nevada test site. Nature 397, 56–59.
26. Lerman, A., Non-Equilibrium Systems in Water Chemistry, J.D. Hem, ed., ACS Advances in Chemistry Series 106, American Chemical Society, Washington DC, p.32, 1971.
27. Li Y.-H, and S. Gregory, Diffusion of ions in sea water and in deep-sea sediments, Geochim. Cosmochim. Acta, 38, 703-714, 1974.
28. Logan, B.E., D.G. Jewitt, R.G. Arnlod, E.J. Bouwer, and C.R. O'Melia, Clarification of clean-bed filtration models, Journal of Enviromental Engineering, 121, 869-873, 1995.
29. Logan, B.E., D.G. Jewitt, R.G. Arnlod, E.J. Bouwer, and C.R. O'Melia, Clarification of clean-bed filtration models, Journal of Enviromental Engineering, 121, 869-873, 1995.
30. Loveland, J.P., J.N Ryan, G.L.Amy, and R.W. Harver, The revesibility of attachment to mineral surfaces. Colloids and Surfaces A: Physicochemical and Ebgubeerubg Asoects, 107, 205-221, 1996.
31. McCarthy, J.F., Zachara, J.M., 1989. Subsurface transport of contaminants. Environ. Sci. Technol. 23, 496–502.

32. Noell, A.L., Thompson, J.L., Corapcioglu, M.Y., Triay, I.R., 1998. The role of silica colloids on facilitated cesium transport through glass bead columns and modeling. J. Contam. Hydrol. 31, 23–56
33. Ouyang, Y., Shinde, D., Mansell, R.S., Harris, W., 1996. Colloidenhanced transport of chemicals in subsurface environments: a review. Crit. Rev. Environ. Sci. Technol. 26, 189–204
34. Pang, L., Close, M.E., Noonan, M.J., Flintoft, M.J., Van den Brink, P., 2005. Heavy metals in the environment – a laboratory study of bacteria-facilitated Cadmium transport in alluvial gravel aquifer media. J. Environ. Qual. 34, 237–247.
35. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical recipes in Fortran 77 (CUP, 1997)(ISBN 052143064X)(1160s).
36. Rajagopalan R., and C. Tien, Trajectory analysis of deep-bed filtration with the sphere-in-cell porous media model, AIChE Journal, 22, 523-533,1976
37. Richard S. Palais and Robert A. Palais Differential equations, Mechanics, and Computation, 2009, ISBN-13: 978-0-8218-2138-1
38. Ruckenstein, E., and D.C. Prieve, Adsirption and desorption of particles and their chromatographic separation. AIChE J. 22, 276-283, 1976
39. Ryan J.N. and P.M. Gschwend (1994), "Effects of ionic strength and flow rate on colloid release: Relating kinetics to intersurface potential energy", Journal of Colloid and Interface Science, Vol. 164, pp. 21-34.
40. Ryan, J.N., and M. Elimelech, Colloid mobilization and transport in groundwater, Colloid Surfaces, 107, 1-56, 1996
41. Schulze-Makuch, D., Longitudinal dispersivity data and implications for scaling behavior, Ground Water, 43(3), 443-456, 2005.
42. Scott C. James and Constantinos V. Chrysikopoulos, *Effective velocity and effective dispersion coefficient for finite-sized particles flowing in a uniform fracture,* Journal of Colloid and Interface Science 263 (2003) 288-295.
43. Syngouna V.I. and Chrysikopoulos C.V. (2010), "Interaction between viruses and clays in static and dynamic batch systems", Environmental Science and Technology, Vol. 44, pp. 4539-4544.
44. Syngouna, V.I., and C.V. Chrysicopoulos, Interaction between viruses and clays in static and dynamic batch systems, Environmental Science and Technology, 44 (12), 4539-4544, 2010.
45. Truesdail, S.E., J. Lukasik, S.R. Farra, D.O. Shah, abd R.B. Dickinson, Analysis of bacterial deposition on metal (hydr) oxide-coated sand filter media, Journal of Colloid and Interface Science , (203)(2), 369-378, 1998.
46. Van Oss, C.J., R.F. Giese, and P.<. Costanzo, DLVO and NON-DLVO interactions in hectorite. Clays Clay Miner., 38(2), 151-159, 1990.
47. Vasiliadou I.A. and Chrysikopoulos C.V. (2011),"Cotransport of Pseudomonas putida and kaolinite particles through water saturated

columns packed with glass beads", Water Resources Research, Vol. 47, doi: 10.1029/2010WR009560.
48. Walton, F.B., Merritt, W.F., 1980. Long-term extrapolation of laboratory glass leaching data for the prediction of fission product release under actual groundwater conditions. Sci. Basis Nucl. Waste Manage. 2, 155–166
49. Yao, K.-M., M.T. Habibian, and C.R. O'Melia, Water and waste water filtration: Concepts and applications, Enviromental Science and Technolodgy, 5(11), 1105-1112, 1971.
50. Youn Sim and Constantinos V. Chrysikopoulos, Analytical solutions for solute transport saturated porous media with semi-infinite orfinite thickness,1998.
51. Ian Scott ,Chrysicopoulos, Retardation factor,Paper
52. Χρυσικόπουλος Κ. Β., Ειδικά Θέματα Τεχνολογίας του Περιβάλλοντος, Τμήμα Πολιτικών Μηχανικών, Πανεπιστήμιο Πατρών, Πάτρα 2010.
53. Χρυσικόπουλος Κ. Β., Ειδικά Θέματα Τεχνολογίας του Περιβάλλοντος, Τμήμα Πολιτικών Μηχανικών, Πανεπιστήμιο Πατρών, Πάτρα 2010.
54. Syngouna V.I. and Chrysikopoulos C.V. (2011)," Transport of biocolloids in water saturated columns packed with sand": Effect of grain size and pore water velocity". Journal of Contaminant Hydrolodgy

10. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ MANUAL

10.1 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΛΟΓΙΣΜΙΚΟΥ

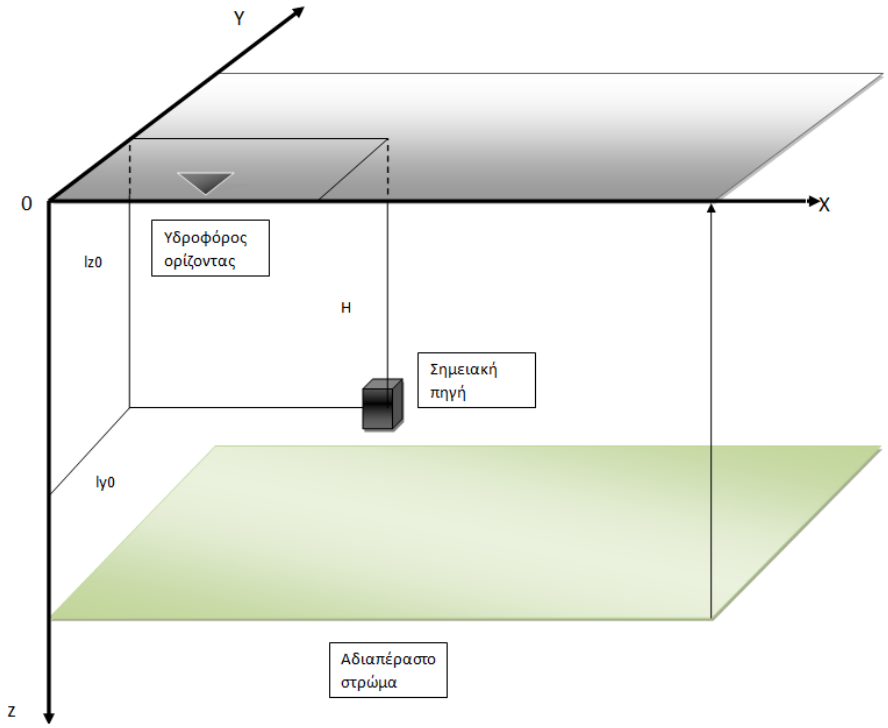
Για το αριθμητικό μοντέλο που δημιουργήθηκε κεφ. 5.2, ώστε να περιγραφεί η συμμεταφορά ρύπων σε τρισδιάστατο πορώδες μέσο, αναπτύχθηκε ο κατάλληλος κώδικας σε γλώσσα Fortran (κεφ. 7) και στην συνέχεια έγινε compilation (Intel Fortran Compiler 11.0.6) ώστε να γίνει η μετατροπή του σε εκτελέσιμο αρχείο "Cootransport_Simulation.exe".

Το "Cootransport_Simulation.exe" μπορεί να αντιμετωπίσει ένα σύνολο προβλημάτων που παρουσιάζονται τόσο σε πειράματα στο εργαστήριο όσο και σε πραγματικές καταστάσεις στο πεδίο. Είναι σε θέση να προσομοιώσει ένα μεγάλο εύρος γεωμετρίας πηγών που αρχίζει από σημειακή Σχήμα 10.1 και τελειώνει σε ελλειψοειδή πηγή Σχήμα 10.2. Η χρήση ελλειπτικής γεωμετρίας είναι επίσης δυνατή, με τον περιορισμό ότι αυτή θα βρίσκεται σε μια από τις τρείς κύριες επιφάνειες yz, xz και xy του υδροφορέα Σχήμα 10.3. Σε κάθε περίπτωση τόσο η ελλειπτική όσο και η ελλειψοειδής πηγή αυτόματα χωρίζονται από το πρόγραμμα σε ένα σύνολο σημειακών πηγών Σχήμα 10.4 και στην συνέχεια ξεκινάει η εκτέλεση της προσομοίωσης.

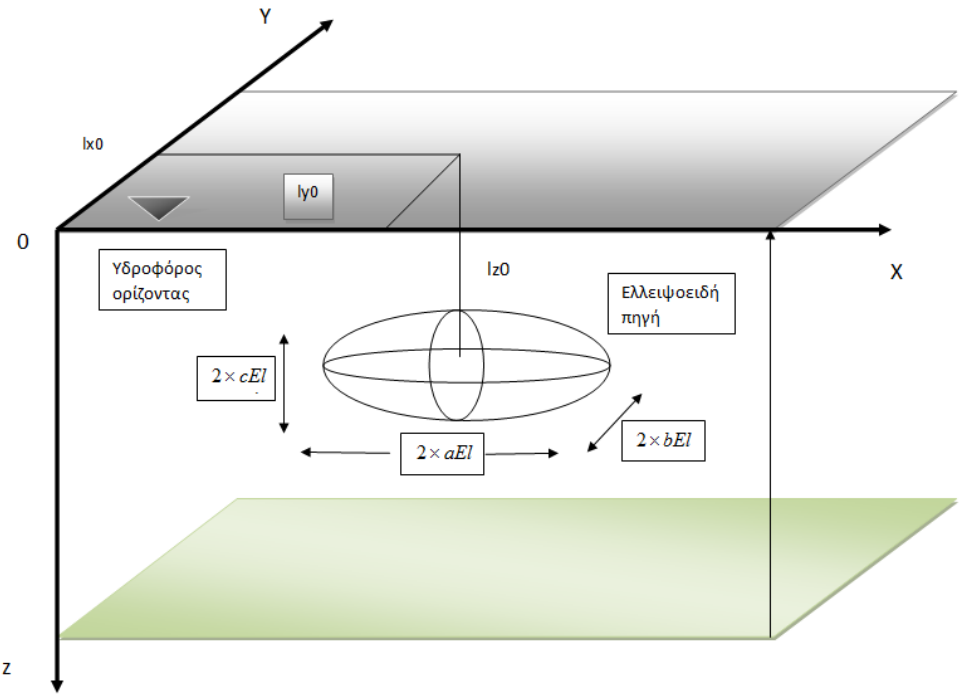
Πέρα όμως από το σχήμα της πηγής, ακόμα ένα μεγάλο σύνολο παραμέτρων βρίσκονται στην διάθεση του τελικού χρήση ώστε μοντελοποιήσει όσο καλύτερα γίνεται το φαινόμενο της συμμεταφοράς. Μερικοί από αυτούς είναι οι φυσικές σταθερές του υδροφορέα (π.χ. πυκνότητα, πορώδες), η διάρκεια και ισχύ της εκάστοτε πηγής, οι αριθμητικές σταθερές (π.χ. χρονικό και χωρικό βήμα) αλλά και πολύ άλλοι ακόμα που περιγράφουν την μορφοποίηση και το είδος των αποτελεσμάτων που ζητούνται.

Τέλος με σκοπό να γίνει η χρήση του "Cootransport_Simulation.exe" πιο εύκολη, δημιουργήθηκε στην γλώσσα προγραμματισμού Visual Basic 2008 γραφικό περιβάλλον διεπιφάνειας χρήστη (Graphical User Interface). Το οποίο αναλαμβάνει να μεσολαβήσει μεταξύ του χρήση και του προγράμματος προσομοίωσης "Cootransport_Simulation.exe" ώστε με εύκολο τρόπο να γίνει εισαγωγή των απαραίτητων παραμέτρων στο τελευταίο. Το αναλυτικό εγχειρίδιο του "Cootransport_Simulation.exe" μέσο του Interface δίνεται στο κεφ. 8.2.

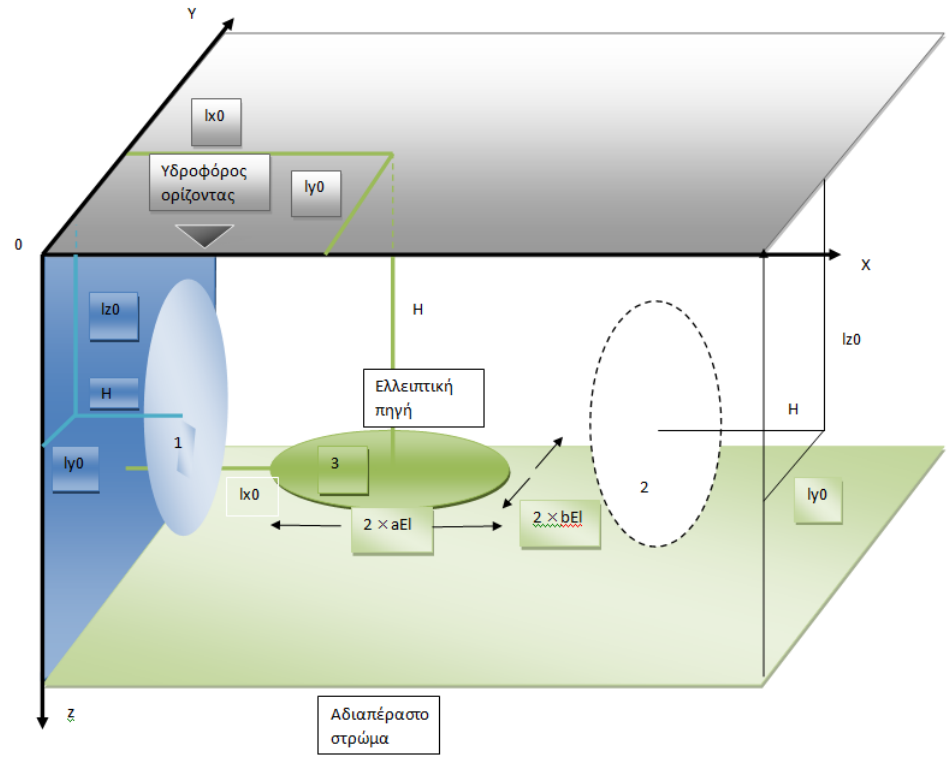
Παρακάτω στα Σχήματα 10.1-10.4 φαίνονται τα δυνατά είδη πηγών που μπορεί να προσομοιώσει το λογισμικό "Cootransport_Simulation.exe".



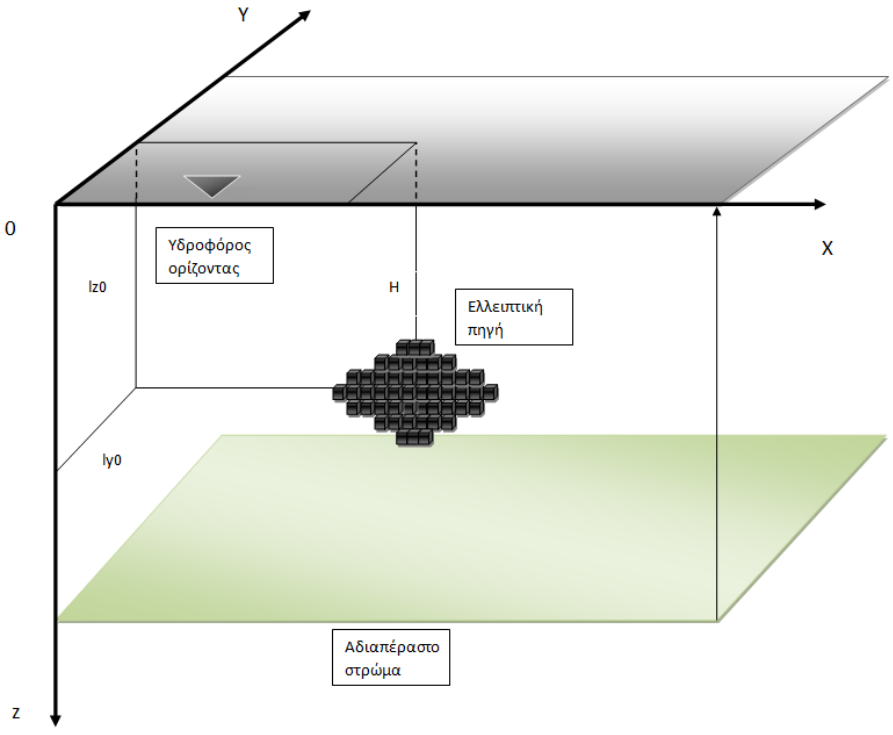
Σχήμα 10.1: Παρουσιάζεται μοντέλο που μπορεί να επιλύει ο κώδικας "Cootransport_Simulation.exe". Η πηγή εδώ θεωρείται σημειακή αλλά με πεπερασμένο όγκο.



Σχήμα 10.2: Παρουσιάζεται μοντέλο ελλειψοειδής πηγής που μπορεί να επιλύει ο κώδικας "Cootransport_Simulation.exe". Η πηγή εδώ θεωρείται ελλειψοειδής με τυχαίες διαστάσεις στον X, Y, Z, άξονα.



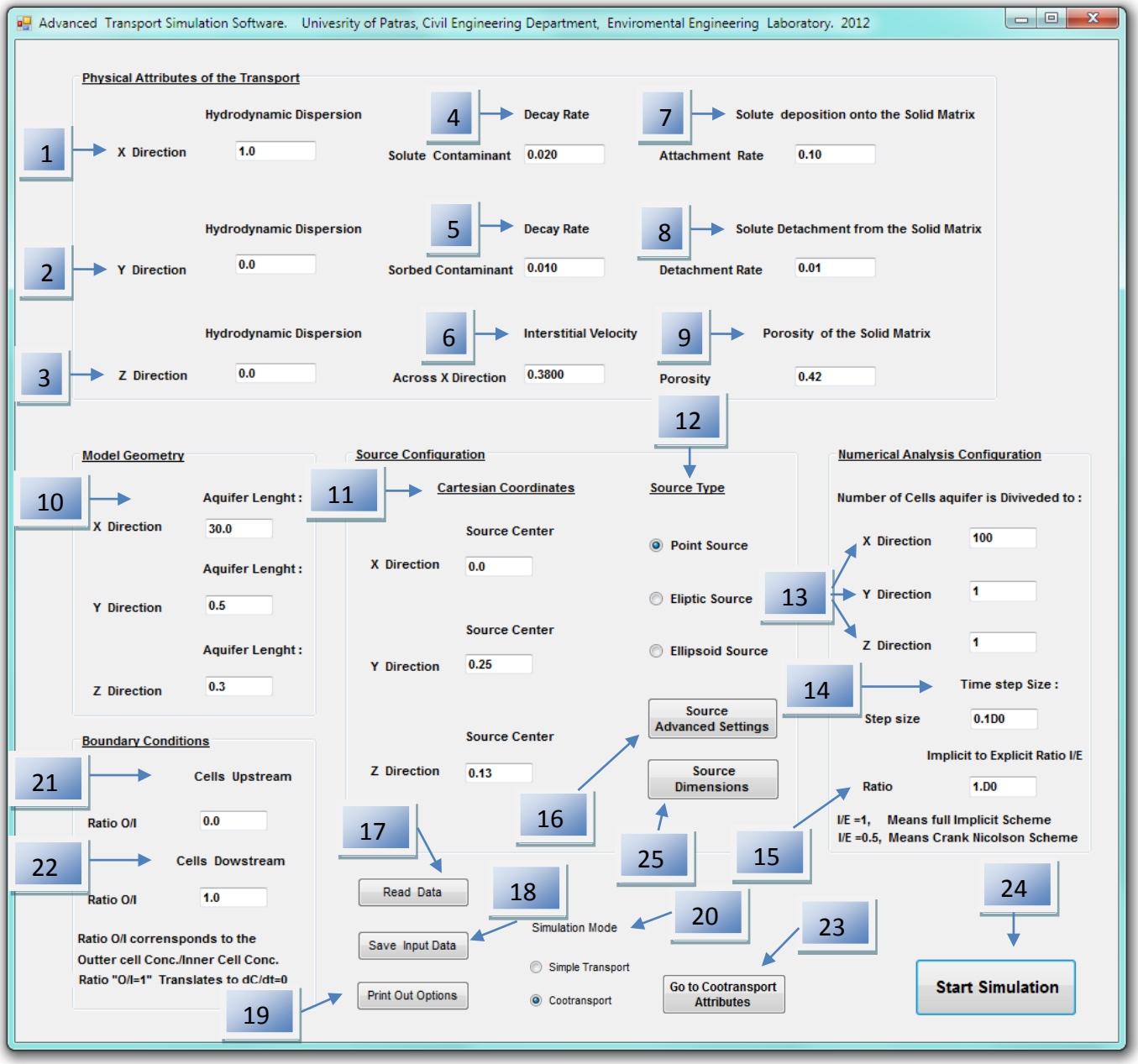
Σχήμα 10.3: Παρουσιάζεται μοντέλο με επιφανειακές ελλειπτικές πηγές, σε διαφορετικά επίπεδα, που μπορεί να επιλύει ο κώδικας "Cootransport_Simulation.exe". Η ελλειπτική πηγή 1 ανήκει στο επίπεδο YZ, η πηγή 2 στο XZ και τέλος η πηγή 3 ανήκει στο επίπεδο XY.



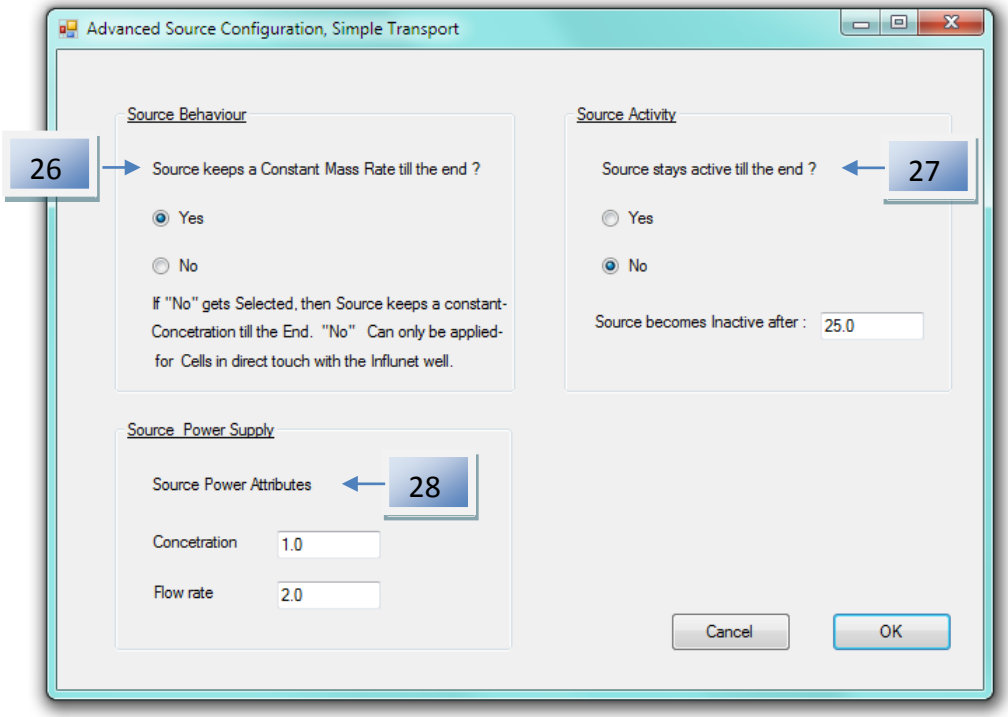
Σχήμα 10.4: Παρουσιάζεται μοντελοποίηση ελλειπτικής πηγής, που ανήκει στο επίπεδο XZ, σύμφωνα με τον κώδικα "Cootransport_Simulation.exe".

10.2 User Manual "Cootransport_Simulation.exe"

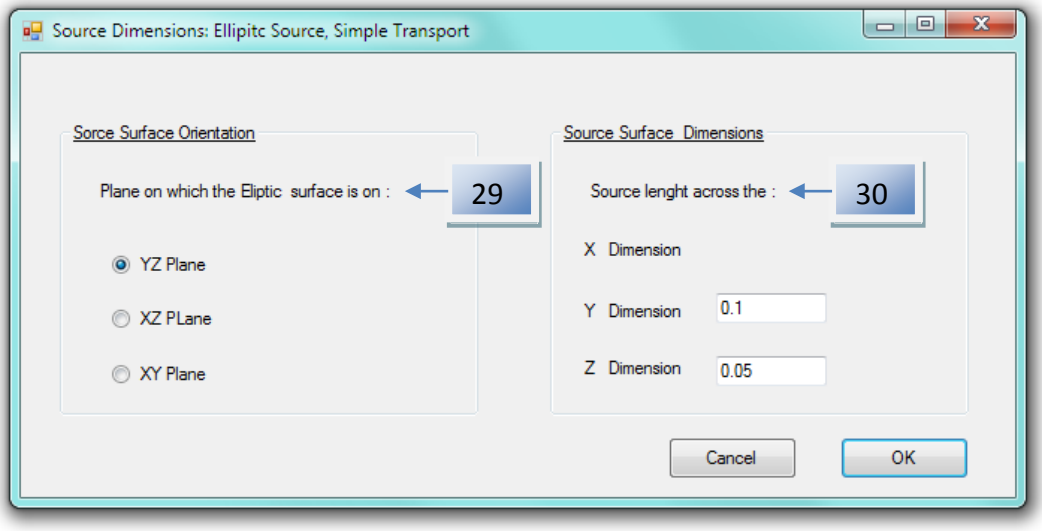
Το αρχείο εισόδου που περιμένει για εισαγωγή το "Cootransport_Simulation.exe" ονομάζεται input.txt και βρίσκεται στον ίδιο υποφάκελο με το εκτελέσιμο αρχείο Cootransport_Simulation.exe. Στο Σχήμα 10.5(a-d) φαίνεται το Interface που διαβάζει από το αρχείο αυτό και παρομοίως (το interface) βρίσκεται στο ίδιο υποφάκελο με το "Cootransport_Simulation.exe".



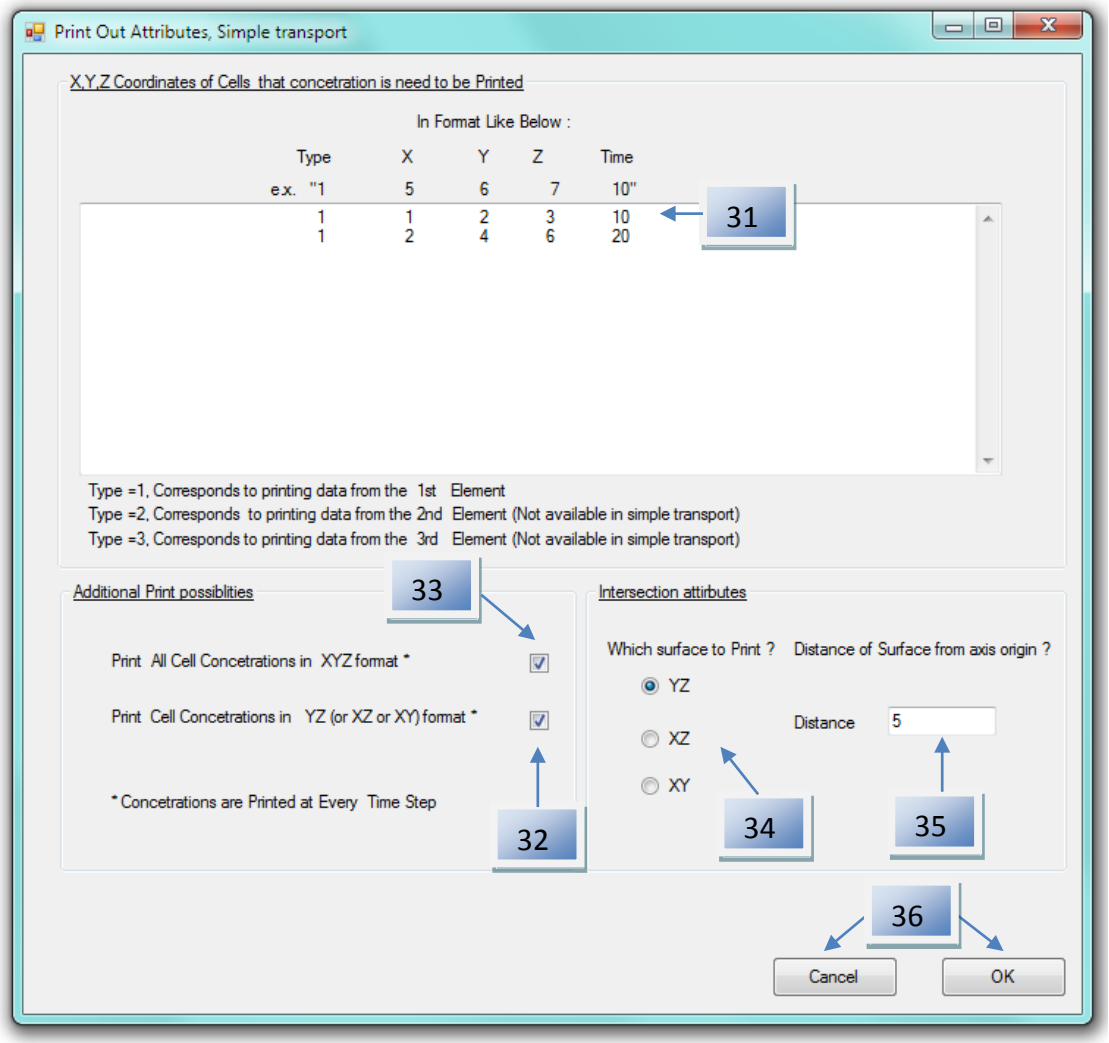
Σχήμα 10.5a: Παρουσιάζεται η δομή του Interface για το πρόγραμμα "Cootransport_Simulation.exe". Το συγκεκριμένο είναι αρκετό για να επιλύσει ένα πρόβλημα απλής μεταφοράς αλλά για την περίπτωση συμμεταφοράς χρειάζεται και το αντίστοιχο Interface Σχήμα 10.6.



Σχήμα 10.5b: Παρουσιάζεται η συνέχεια του Interface για το πρόγραμμα "Cootransport_Simulation.exe". Το συγκεκριμένο είναι υπεύθυνο για να περιγράφει την διάρκεια και την ισχύ της πηγής (απλή μεταφορά μόνο).



Σχήμα 10.5c: Παρουσιάζεται η συνέχεια του Interface για το πρόγραμμα "Cootransport_Simulation.exe". Το συγκεκριμένο είναι υπεύθυνο για να περιγράφει την γεωμετρία σε περίπτωση που έχουμε ελλειπτική πηγή (απλή μεταφορά μόνο).



Σχήμα 10.5d: Παρουσιάζεται η συνέχεια του Interface για το πρόγραμμα "Cootransport_Simulation.exe" . Το συγκεκριμένο είναι υπεύθυνο για να περιγράφει το είδος των αποτελεσμάτων που θέλει να λάβει ο χρήστης (απλή μεταφορά μόνο).

10.2.1 ΑΝΑΛΥΣΗ ΤΟΥ INTERFACE ΑΠΛΗΣ ΜΕΤΑΦΟΡΑΣ ΚΑΙ Η ΧΡΗΣΗ ΤΟΥ

Όταν καλεί για πρώτη φορά ο χρήστης το "Cootransport_Simulation.exe" εμφανίζεται μπροστά του το Σχήμα 10.5a. Από αυτό μπορεί μέσω των κατάλληλων κουμπιών να μεταβεί σε οποιαδήποτε άλλο μέρος του προγράμματος αυτός επιθυμεί (π.χ. Σχήμα 10.5b ή 10.5d) και από εκεί να εισάγει τις κατάλληλες εκάστοτε παραμέτρους. Όταν θελήσει η αλλαγή που έκανε σε κάποια παράμετρο να αποθηκευτεί στο σκληρό δίσκο δεν έχει παρά να πατήσει το κουμπί 18 Σχήμα 10.5a " Save input data" και εάν θελήσει να τρέξει το μοντέλο που μέχρι εκείνη την στιγμή έχει ορίσει δεν έχει παρά να πατήσει το κουμπί 24 Σχήμα 10.5a " Start Simulation". Τότε θα αρχίσουν οι υπολογισμοί. Μόλις αυτοί τελειώσουν μπορεί να πάει στον φάκελο όπου βρίσκεται το "Cootransport_Simulation.exe" και εκεί να αναζητήσει τα αποτελέσματα στα αρχεία print.txt ή printSurWCoord.txt και printSur.txt.

Όλα οι δυνατές επιλογές που μπορεί να κάνει ο χρήστης θα αναλυθούν μία προς μια αρχίζοντας με το βασικό interface Σχήμα 10.5a.

Έτσι για το Σχήμα 10.5a οι αριθμοί στα πλαίσια καθορίζουν τις εξής παραμέτρους (οι μονάδες είναι ενδεικτικές όπως ενδεικτικός είναι και ο ρύπος που μεταφέρεται π.χ. άργιλος.) :

1. Dcx =Longitudinal hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
2. Dcy =Lateral hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
3. Dcz =vertical hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
4. L_c= decay rate of liquid phase solute $[\frac{1}{days}]$
5. L_{c^*}= decay rate of sorbed solute $[\frac{1}{days}]$
6. U =average interstitial velocity $[\frac{cm}{day}]$. (ενδοπορώδης ταχύτητα στην διεύθυνση x)
7. R_{v-v^*} Forward rate coefficient $[\frac{1}{days}]$
8. R_{v^*-v} Reverse rate coefficient $[\frac{1}{days}]$
9. Th=porosity of the solid matrix
10. lxx =dimension of the aquifer parallel to the x direction (cm)
    lyy= dimension of the aquifer parallelto the y direction (cm)
    lzz= finite aquifer thickness (cm)
11. Lx0 = Cartesian coordinates of the point source X direction [cm]
    Ly0 = Cartesian coordinates of the point source Y direction [cm]
    Lz0 = Cartesian coordinates of the point source Z direction [cm]
12. Indicates the type of the source. The possible options are Point or Eliptic or Ellipsoid Source Figs. 10.3, 10.2.

13. The number of cells each x, y, z direction will be split to. Required for discretization of the aquifer (Numerical methods demand it).
14. The number of pieces the time will be divided to. Required by the discretization scheme (Numerical methods demand it).
15. Ratio between the Full implicit and the explicit scheme. Full implicit is very stable but sometimes can be less accurate. On the other hand explicit scheme tends to be unstable but when it converges it does it with more accuracy. A value for the Ratio 0.5 creates the Crank Nicolson method which is known to be stable for the simple transport problems (no adsorption or decay)
16. Button that opens the Advanced Source Settings Interface Fig 10.5b.
17. Button that starts reading the input.txt file and passes data read to all possible interfaces. Usefull in case someone do an unwanted change and wants to reset all parameters to their initial data. Don't forget the reading of the file is automatically done when the program opens for the first time.
18. All changes done to the interface are saved to the input.txt file. This action is also automatically done when the start simulation button 24 is clicked.
19. Opens the interface required to describe the kind and the number of wanted output data fig. 10.5d.
20. Option that defines whether we have a simple transport or we have a full cootransport (two different solutes the same time).
21. Defines the ratio of the outer cell concentration versus the inner boundary cell (Upstream). Really usefull when an upstream clear water well exists that creates a flow of zero concentration solute. In that case the ratio must be set to zero. When set to one it corresponds to the dC/dt=0 boundary condition.
22. Defines the ratio of the outer cell concentration versus the inner boundary cell (downstream). It is usually set to one to indicate that the solutes that exit the aquifer never come in again. Also this setting corresponds to the dC/dt=0 boundary condition.

23. It opens the Cootransport interface fig. 10.6a (to do so option 20 must be set to Cootransport). The specific interface is needed when we want to simulate Cootranport.
24. Maybe the most important button. It start the simulation with the parameters defined till now. When the simulation is finished the user can go and check the results in the respective print.txt files.
25. In case option 12 (Source type) is set to Elliptic or Ellipsoid source there is a need to define the geometry of the Source. This is done with the option 25 which opens the needed interface

Επεξήγηση του Interface Σχήμα 10.5b (Advanced source Configuration):

26. Current Version of the model allows two types of source cell. The first one keeps constant concentration till the end. It is usefull only when there is an influent well at the start of the x direction (x=0) which has non zero solute concentration and thus it allows pollutants to enter the aquifer. In that case the upstream boundary cells keep constant concentration and the option 21 fig. 10.5a must be set to zero. The second source type allows to have constant mass rate. This is usefull when we inject our pollutant in a random point of the aquifer. When done so the options 21 or 22 fig. 10.5a can take any positive value.
27. The source can be inactivated after some time. If done so we must define the time after which the inactivation occurs.
28. In case we have constant mass release from the source cells we must define the magnitude of the flow rate and the concentration of the incoming injection fluid. Otherwise if we have constant concentration we need only to define the concentration magnitude.

Επεξήγηση του Interface Σχήμα 10.5c (Source dimensions):

29. When the option 12 fig 105a is set to either Elliptic or Ellipsoid we need to define the geometry of source. This can be done with the current interface fig. 10.5c and the options 29, 30. Option 29 Indicates the plane the Elliptic surface is set on fig. 10.3.
30. Defines the length of the Elliptic source in the corresponding x, y, z dimension fig. 10.3.

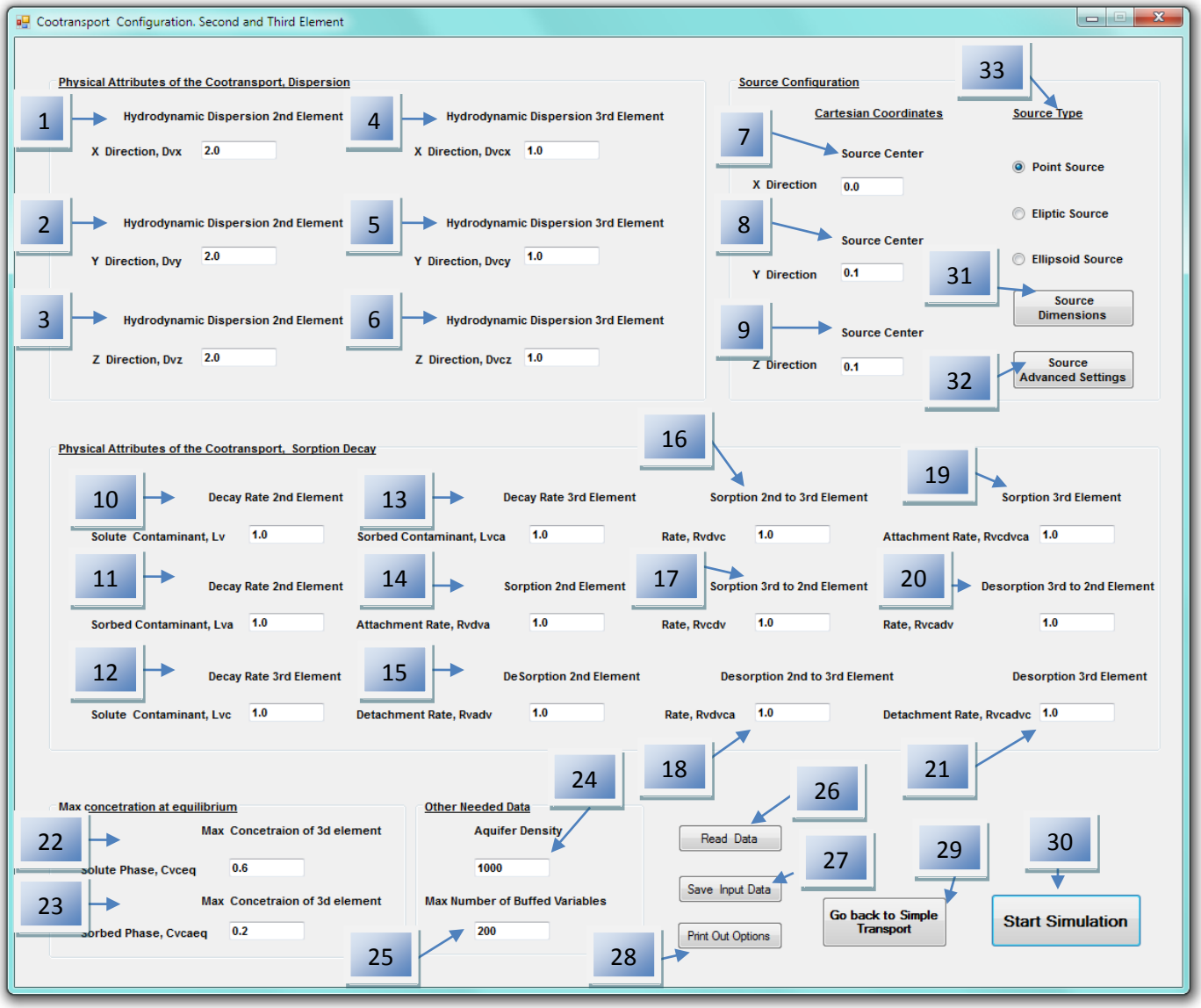
Επεξήγηση του Interface Σχήμα 10.5d (Print out Attributes):

31. Till now we described many things but we never talked about time. We need our concentrations at an explicit point, after a specific amount of time. These parameters are inputted in the 31 option fig. 10.5d. For example if we enter "1 1 2 3 10" we ask the model to find the concentration (of the simple transport problem) at a point with x=1, y=2, z=3 cm after 10 mins have passed. The first parameter with value "1" defines the kind of the Concentration to be printed. While we are on Simple Transport this parameter can occupy only the value "1". This isn't true for the case of Cootransport, where values "2" and "3" are also supported. For further details check Cootranport interface.

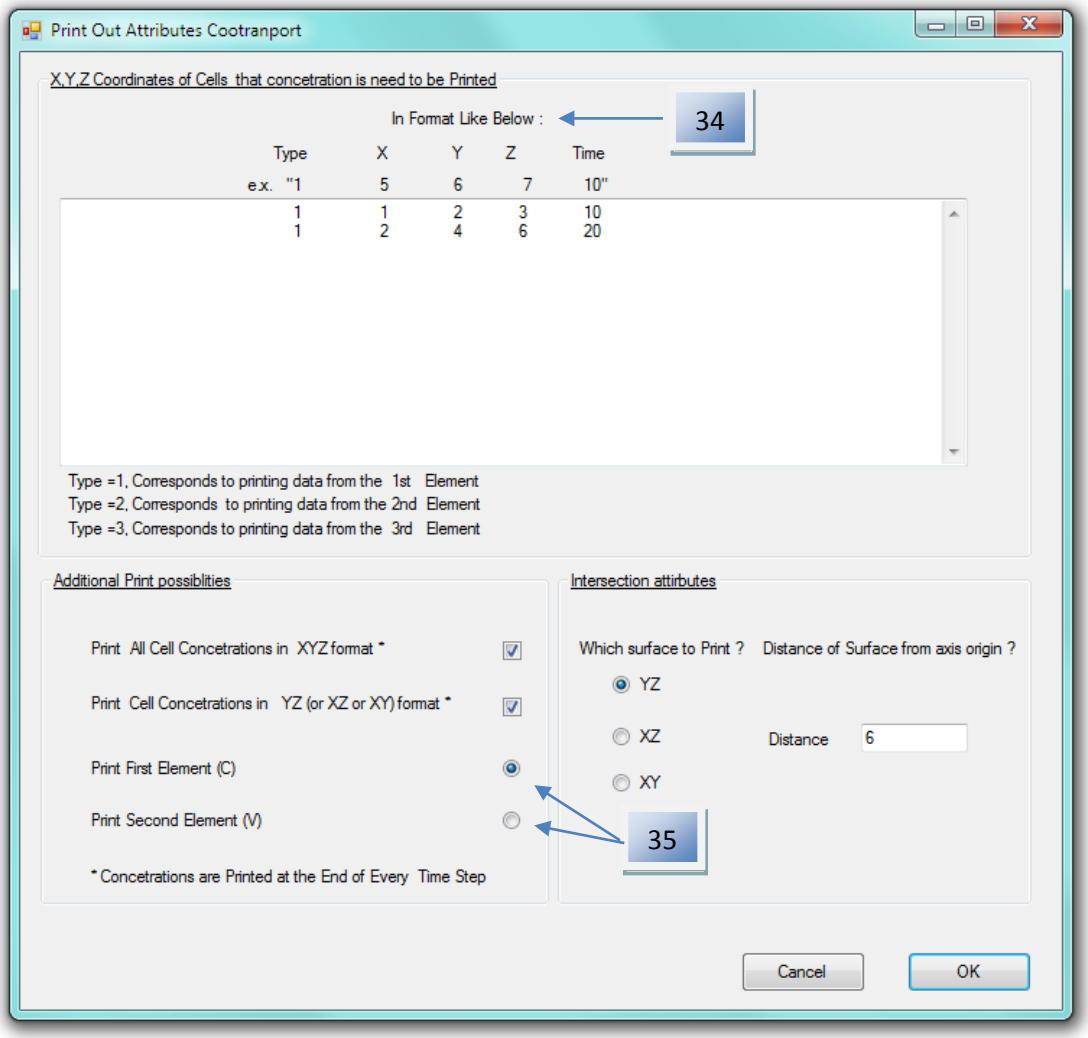
32. Sometimes we require more info from just the concentration at a specific point. That can be achieved by doing intersections and printing the whole surface from a 3d aquifer. So option 32 allows us to get concentrations from an intersection in two different formats: a) printSurWCoord (with coordinates), b) printSur (without coordinates). "Yes" means we want to do an intersection and print the results for every time step (which is defined from option 14 fig. 10.5a) in the respective output file, while "no" means we just get the results asked from option 31. Choice 32 is really usefull when we need to create contours of a particular surface.
33. Like option 32, option 33 allows us to get in a "XYZ C" format all the cell concentrations of simple transport, at the end of every time step. Each time step corresponds to a different file (named by increasing number, in the multiprints subfolder) .
34. Allows us to select which plane to print (YZ, XZ, or XY) while option 35,
35. Allows us to define the distance of the plane from the origin of the axis.
36. In case all changes made to the current (Print out Attributes) interface, are valid then we just have to push the "Ok" button to save them. If we don't want to keep them we just push Cancel. Don't forget Clicking "Ok" keeps the changes but this doesn't mean changes are written to the hard disc. Which leads to the fact that if the program closes without pushing "save data" option 18 fig. 10.5a any changes done are lost and the next time the program starts the last saved data will be loaded.

10.2.2 ΑΝΑΛΥΣΗ ΤΟΥ INTERFACE ΣΥΜΜΕΤΑΦΟΡΑΣ ΚΑΙ Η ΧΡΗΣΗ ΤΟΥ

Μέχρι τώρα περιγράψαμε την απλή μεταφορά ενός ρύπου (Element 1). Στο Σχήμα 10.6 που ακολουθεί παρουσιάζεται το Interface που βοηθάει τον χρήστη να εισάγει τις παραμέτρους για την περίπτωση της συμμεταφοράς (Cootransport).



Σχήμα 10.6a: Παρουσιάζεται η δομή του Interface για το πρόγραμμα "Cootransport_Simulation.exe". Περιγράφει τις επιπλέον παραμέτρους που χρειάζονται ώστε να λυθεί το πρόβλημα της συμμεταφοράς.



Σχήμα 10.6b: Παρουσιάζεται η συνέχεια του Interface για το πρόγραμμα "Cootransport_Simulation.exe". Το συγκεκριμένο είναι υπεύθυνο για να περιγράφει το είδος των αποτελεσμάτων που θέλει να λάβει ο χρήστης (Συμμεταφορά).

Παρακάτω φαίνονται οι δυνατές επιλογές του Interface. Έχουν γίνει οι εξής δύο σιωπηλές παραδοχές: α) Έχουμε ταυτόχρονη μεταφορά δύο ρύπων εκ των οποίων ο πρώτος είναι απλή άργιλος (σύμβολο C) η οποία και εκτελεί απλή μεταφορά ενώ ο δεύτερος είναι ιός (σύμβολο V), ο οποίος συμμεταφέρεται β) Οι σταθερές που αφορούν τον πρώτο ρύπο θεωρούνται ότι είναι αυτές που έχουν δοθεί στο Interface απλής μεταφοράς Σχήμα 10.5. Ακόμα βάση των παραπάνω "Element 1" θεωρείται ότι είναι η άργιλος, "Element 2 " θεωρείται ότι είναι ο ιός και "Element 3 " θεωρείται ότι είναι ο ιός προσροφημένος επάνω στην άργιλο (Virus-Clay=Cvc).

Διαθέσιμες επιλογές του Cootransport Interface Σχήμα 10.6:

1. Dvx =Virus Longitudinal hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
2. Dvy = Virus Lateral hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
3. Dvz = Virus Vertical hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
4. Dvcx =Virus-Clay Longitudinal hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
5. Dvcy = Virus-Clay Lateral hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
6. Dvcz = Virus-Clay Vertical hydrodynamic dispersion coefficient $[\frac{cm^2}{day}]$
7. Coordinates for the source x-direction [cm].
8. Coordinates for the source y-direction [cm].
9. Coordinates for the source y-direction [cm].
10. L_v= virus decay rate of liquid phase solute $[\frac{1}{days}]$
11. L_{v^*}= Virus decay rate of the sorbed phase $[\frac{1}{days}]$
12. L_{vc}= Virus-Clay decay rate $[\frac{1}{days}]$
13. L_{vc^*}= Virus-Clay sorbed on the solid matrix decay rate $[\frac{1}{days}]$
14. R_{v-v^*}= Forward rate coefficient (v converts to v^*) $[\frac{1}{days}]$
15. R_{v^*-v} =Reverse rate coefficient $[\frac{1}{days}]$
16. R_{v-vc} = Virus to Virus-Clay conversion rate $[\frac{1}{days}]$
17. R_{vc-v} = Virus-Clay to Virus conversion rate $[\frac{1}{days}]$
18. R_{v-vc^*} = Virus to sorbed Virus-Clay conversion rate $[\frac{1}{days}]$
19. R_{vc-vc^*} = Virus-Clay to sorbed Virus-Clay conversion rate $[\frac{1}{days}]$
20. R_{vc^*-v} = Sorbed Virus-Clay to Virus conversion rate $[\frac{1}{days}]$
21. R_{vc^*-vc} = Sorbed Virus-Clay to Virus-Clay conversion rate $[\frac{1}{days}]$
22. Cvceq= Max concentration of Virus-Clay complex at equilibrium (Pfu virus/mg clay)
23. Cvceq*= Max concentration of sorbed Virus-Clay complex at equilibrium (Pfu virus/mg clay)
24. rD=bulk density of the solid matrix(solid mass/aquifer volume) $\left[\frac{mg}{cm^3}\right]$
25. In order to increase speed, some data is buffed onto memory. Option 25 allows to define the max capacity of the buffed data. So in our example we set the capacity to be 200 which means 200 variables (8 byte each) are buffed. More capacity decreases time. Though one must be careful with the memory usage since a large number in this option can result in several Gigabytes of memory usage (sometimes more than 12 gigabytes of memory maybe required).

26. Initiates again reading input data from file. Useful in case there is a need to undone changes to the interface. Don't forget though that all input files are read once during the load of the initial interface fig 10.5a.
27. Saves all changes done to the respective input files on the hard disk. The input files are held on the same subfolder as the main interface executable and are named "input.txt", "inputC.txt" and "inputMisc.txt".
28. Opens the interface that will define what kind of data to print fig. 10.6b. Q= is the flow rate with which we transfer solute mass (with concentration Cm0) to the source cell $\left[\frac{cm^3}{day}\right]$.
29. Saves to memory all changes done and return us to the initial simple transport interface fig. 10.5a.
30. Starts the Cootransport simulation.
31. Specifies the source geometry in case we have selected in option 33 Elliptic or Ellipsoid Source (opens the required interface).
32. Opens the required interface that will define the type (constant mass or constant Concentration) of our second pollutant source (e.x. Virus mass). The first source was defined on the simple source interface and nothing is needed to be done again.
33. Specifies the geometry of our second (e.x. virus) source.

Μέχρι εδώ περιγράψαμε το Interface Σχήμα 10.6a, τώρα συνεχίσουμε αναλύοντας το Interface Σχήμα 10.6b που ορίζει ποιά αποτελέσματα θέλει να δει στην έξοδο χρήστης.

34. Option 34 allows the user to enter in a simple and straightforward manner the position and the time after which he wants to have the concentration printed in print.txt file as seen in fig. 10.9. So in our example fig. 10.6b user asked to get the concentration of the first Element (here Clay) at position with coordinates x=1, y=6 and z=7 after 10 days have passed which corresponds to input "1 1 6 7 10 ". The user could also have inputted "2 1 6 7 10 " which translates to print Virus concentration from a point with coordinates x=1, y=6 and z=7 after 10 days have passed. Because we are on Cootransport more options are supported and the first parameter can be valid for values "1", "2" or "3". The value "1" indicates Clay concentration (first element), value "2" indicates Virus concentration (second element) and lastly value "3" indicates Virus concentration sorbed on clays (third element).
35. Like it was discussed on fig. 10.5d, option 35 allows to select whether to print the concentrations of the first element (Clay) or the second Element (virus). The format can be either xy or xyz.

10.2.3 ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΞΟΔΟΥ

Έχοντας χρησιμοποιήσει σωστά το Interface για την εισαγωγή δεδομένων ο χρήστης καλείται να ξεκινήσει την προσομοίωση. Αυτό γίνεται κάνοντας click στο κουμπί "Start Simulation" Σχήμα 10.5α. Μόλις η προσομοίωση τελειώσει τα αποτελέσματα γράφονται στα απαραίτητα αρχεία στον σκληρό δίσκο του συστήματος. Κανείς μπορεί να τα βρει πηγαίνοντας στον ίδιο υποφάκελο με το αρχείο "Cootransport_Simulation.exe" και ψάχνοντας για τα ονόματα PrintSURWCOORD.txt ή PrintSurXYZ.txt και multipleprints.

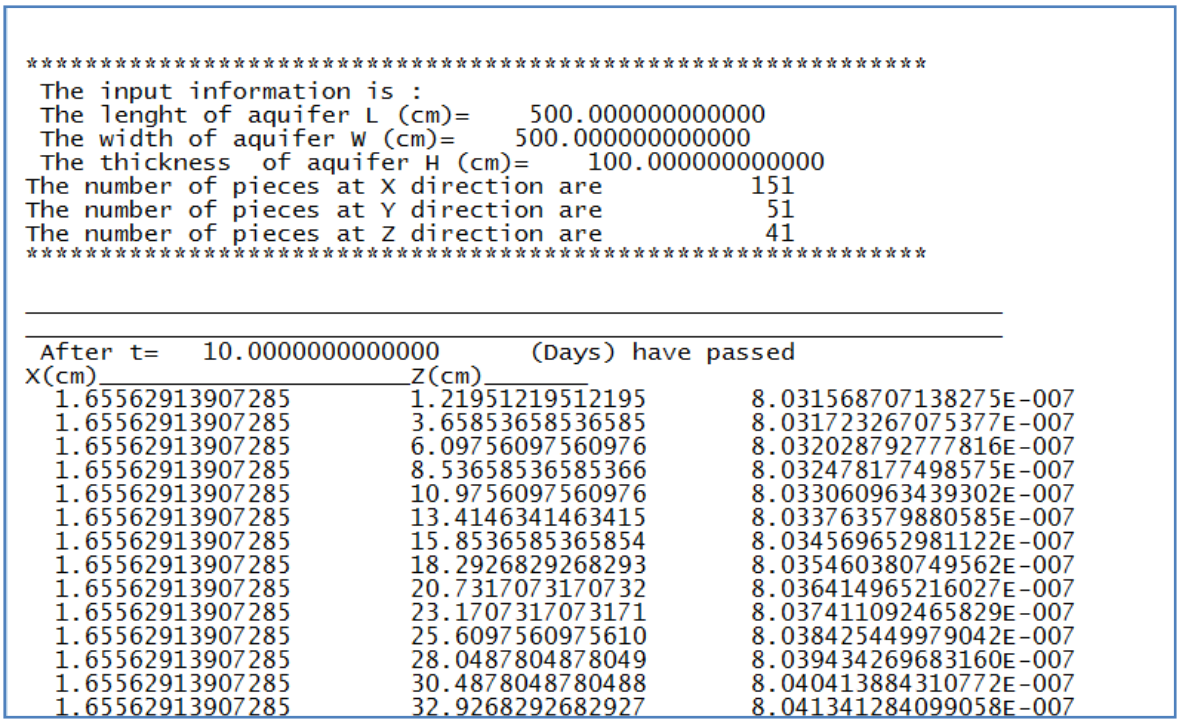
Ενδεικτικά στα παρακάτω Σχήματα 10.7-10.11 φαίνονται τα αρχεία εξόδου. Στο αρχείο PrintSur.txt (που βρίσκεται στον ίδιο υποφάκελο με το input.txt) Σχημα 10.7a, 10.7b αλλά και ταυτόχρονα στο αρχείο PrintSURWCOORD.txt (στον ίδιο υποφάκελο με το "Cootransport_Simulation.exe") Σχήμα 10.8 φαίνονται οι συγκεντρώσεις μιας τομής που επέλεξε ο χρήστης. Η διαφορά τους είναι ότι στο πρώτο βλέπουμε μόνο τις συγκεντρώσεις την μια κάτω από την άλλη ενώ στο δεύτερο παρατηρούμε και τις συντεταγμένες του επιπέδου που εκτυπώνονται σε μορφή x, y, C (οι δύο από τις τρείς καταγράφονται, η τρίτη είναι ο αριθμός Η). Επιπλέον στο αρχείο PrintSur.txt αποτυπώνονται οι σταθερές που περιγράφουν το μοντέλο μας όπως διαβάστηκαν από το αρχείο input.txt.

```
**********************************************
Input file was succesfully read
Dx=     31950.0000000000
Dy=     6450.00000000000
Dz=     6450.00000000000
U=     15.0000000000000
l1=    0.000000000000000E+000
l2=    0.000000000000000E+000
th=    0.250000000000000
lx0=     249.000000000000
ly0=     249.000000000000
lz0=     49.8000000000000
r1=    0.720000000000000
r2=    0.408000000000000
rD=     1.50000000000000
Q=     1.00000000000000
Cm0=     1.00000000000000
sourceSurf=              0
aEl=     25.0000000000000
bEl=    25.0000000000000
cEl=    0.100000000000000
lxx=     500.000000000000
lyy=     500.000000000000
lzz=     100.000000000000
nx=           151
ny=            51
nz=            41
dt=     10.0000000000000
printSurMode=             1
PrintSurXYZ=              2
h=     249.000000000000
**********************************************
```

Σχήμα 10.7a: Παρουσιάζεται αρχείο εξόδου printSur.txt για το πρόγραμμα "Cootransport_Simulation.exe" (πρώτο μέρος).

	**********										
	current	results	are	based	on	time	t=10days				
	_____	Horizontal	---->>>(cm)__	vectical	(cm)_____						
0.80316D-06	0.80333D-06	0.80367D-06	0.80418D-06	0.80486D-06	0.80571D-06	0.80673D-06	0.80793D-06	0.80930D-06	0.81085D-06	0.81257D-06	0.81447D-06
0.80317D-06	0.80334D-06	0.80368D-06	0.80419D-06	0.80487D-06	0.80573D-06	0.80675D-06	0.80795D-06	0.80932D-06	0.81087D-06	0.81259D-06	0.81450D-06
0.80320D-06	0.80337D-06	0.80371D-06	0.80422D-06	0.80491D-06	0.80576D-06	0.80679D-06	0.80799D-06	0.80936D-06	0.81092D-06	0.81264D-06	0.81455D-06
0.80325D-06	0.80342D-06	0.80376D-06	0.80427D-06	0.80496D-06	0.80581D-06	0.80684D-06	0.80805D-06	0.80943D-06	0.81098D-06	0.81272D-06	0.81463D-06
0.80331D-06	0.80348D-06	0.80382D-06	0.80433D-06	0.80502D-06	0.80588D-06	0.80691D-06	0.80812D-06	0.80951D-06	0.81107D-06	0.81281D-06	0.81473D-06
0.80338D-06	0.80355D-06	0.80389D-06	0.80441D-06	0.80510D-06	0.80596D-06	0.80700D-06	0.80821D-06	0.80960D-06	0.81117D-06	0.81292D-06	0.81485D-06
0.80346D-06	0.80363D-06	0.80397D-06	0.80449D-06	0.80519D-06	0.80605D-06	0.80710D-06	0.80832D-06	0.80972D-06	0.81129D-06	0.81305D-06	0.81499D-06
0.80355D-06	0.80372D-06	0.80407D-06	0.80459D-06	0.80528D-06	0.80616D-06	0.80721D-06	0.80843D-06	0.80984D-06	0.81143D-06	0.81320D-06	0.81515D-06
0.80364D-06	0.80382D-06	0.80416D-06	0.80469D-06	0.80539D-06	0.80627D-06	0.80732D-06	0.80856D-06	0.80997D-06	0.81157D-06	0.81335D-06	0.81532D-06
0.80374D-06	0.80392D-06	0.80427D-06	0.80480D-06	0.80550D-06	0.80638D-06	0.80744D-06	0.80869D-06	0.81011D-06	0.81172D-06	0.81351D-06	0.81549D-06
0.80384D-06	0.80402D-06	0.80437D-06	0.80490D-06	0.80561D-06	0.80650D-06	0.80757D-06	0.80882D-06	0.81025D-06	0.81187D-06	0.81368D-06	0.81567D-06
0.80394D-06	0.80412D-06	0.80448D-06	0.80501D-06	0.80572D-06	0.80662D-06	0.80769D-06	0.80895D-06	0.81039D-06	0.81202D-06	0.81384D-06	0.81585D-06
0.80404D-06	0.80422D-06	0.80458D-06	0.80511D-06	0.80583D-06	0.80673D-06	0.80781D-06	0.80908D-06	0.81053D-06	0.81217D-06	0.81400D-06	0.81602D-06
0.80413D-06	0.80431D-06	0.80467D-06	0.80521D-06	0.80593D-06	0.80684D-06	0.80792D-06	0.80920D-06	0.81066D-06	0.81231D-06	0.81415D-06	0.81618D-06
0.80422D-06	0.80440D-06	0.80476D-06	0.80530D-06	0.80603D-06	0.80694D-06	0.80803D-06	0.80931D-06	0.81078D-06	0.81244D-06	0.81429D-06	0.81633D-06
0.80430D-06	0.80448D-06	0.80484D-06	0.80538D-06	0.80611D-06	0.80702D-06	0.80812D-06	0.80941D-06	0.81088D-06	0.81255D-06	0.81441D-06	0.81647D-06
0.80436D-06	0.80454D-06	0.80491D-06	0.80545D-06	0.80618D-06	0.80710D-06	0.80820D-06	0.80949D-06	0.81097D-06	0.81265D-06	0.81451D-06	0.81658D-06
0.80441D-06	0.80459D-06	0.80496D-06	0.80551D-06	0.80624D-06	0.80716D-06	0.80826D-06	0.80956D-06	0.81105D-06	0.81272D-06	0.81460D-06	0.81667D-06
0.80445D-06	0.80463D-06	0.80500D-06	0.80555D-06	0.80628D-06	0.80720D-06	0.80831D-06	0.80961D-06	0.81110D-06	0.81278D-06	0.81466D-06	0.81674D-06
0.80447D-06	0.80466D-06	0.80502D-06	0.80557D-06	0.80631D-06	0.80723D-06	0.80834D-06	0.80964D-06	0.81113D-06	0.81282D-06	0.81470D-06	0.81678D-06
0.80448D-06	0.80466D-06	0.80503D-06	0.80558D-06	0.80632D-06	0.80724D-06	0.80835D-06	0.80965D-06	0.81114D-06	0.81283D-06	0.81471D-06	0.81679D-06
0.80447D-06	0.80466D-06	0.80502D-06	0.80557D-06	0.80631D-06	0.80723D-06	0.80834D-06	0.80964D-06	0.81113D-06	0.81282D-06	0.81470D-06	0.81678D-06
0.80445D-06	0.80463D-06	0.80500D-06	0.80555D-06	0.80628D-06	0.80720D-06	0.80831D-06	0.80961D-06	0.81110D-06	0.81278D-06	0.81466D-06	0.81674D-06
0.80441D-06	0.80459D-06	0.80496D-06	0.80551D-06	0.80624D-06	0.80716D-06	0.80826D-06	0.80956D-06	0.81105D-06	0.81272D-06	0.81460D-06	0.81667D-06
0.80436D-06	0.80454D-06	0.80491D-06	0.80545D-06	0.80618D-06	0.80710D-06	0.80820D-06	0.80949D-06	0.81097D-06	0.81265D-06	0.81451D-06	0.81658D-06
0.80430D-06	0.80448D-06	0.80484D-06	0.80538D-06	0.80611D-06	0.80702D-06	0.80812D-06	0.80941D-06	0.81088D-06	0.81255D-06	0.81441D-06	0.81647D-06
0.80422D-06	0.80440D-06	0.80476D-06	0.80530D-06	0.80603D-06	0.80694D-06	0.80803D-06	0.80931D-06	0.81078D-06	0.81244D-06	0.81429D-06	0.81633D-06
0.80413D-06	0.80431D-06	0.80467D-06	0.80521D-06	0.80593D-06	0.80684D-06	0.80792D-06	0.80920D-06	0.81066D-06	0.81231D-06	0.81415D-06	0.81618D-06
0.80404D-06	0.80422D-06	0.80458D-06	0.80511D-06	0.80583D-06	0.80673D-06	0.80781D-06	0.80908D-06	0.81053D-06	0.81217D-06	0.81400D-06	0.81602D-06
0.80394D-06	0.80412D-06	0.80448D-06	0.80501D-06	0.80572D-06	0.80662D-06	0.80769D-06	0.80895D-06	0.81039D-06	0.81202D-06	0.81384D-06	0.81585D-06
0.80384D-06	0.80402D-06	0.80437D-06	0.80490D-06	0.80561D-06	0.80650D-06	0.80757D-06	0.80882D-06	0.81025D-06	0.81187D-06	0.81368D-06	0.81567D-06
0.80374D-06	0.80392D-06	0.80427D-06	0.80480D-06	0.80550D-06	0.80638D-06	0.80744D-06	0.80869D-06	0.81011D-06	0.81172D-06	0.81351D-06	0.81549D-06
0.80364D-06	0.80382D-06	0.80416D-06	0.80469D-06	0.80539D-06	0.80627D-06	0.80732D-06	0.80856D-06	0.80997D-06	0.81157D-06	0.81335D-06	0.81532D-06
0.80355D-06	0.80372D-06	0.80407D-06	0.80459D-06	0.80528D-06	0.80616D-06	0.80721D-06	0.80843D-06	0.80984D-06	0.81143D-06	0.81320D-06	0.81515D-06
0.80346D-06	0.80363D-06	0.80397D-06	0.80449D-06	0.80519D-06	0.80605D-06	0.80710D-06	0.80832D-06	0.80972D-06	0.81129D-06	0.81305D-06	0.81499D-06

Σχήμα 10.7b: Παρουσιάζεται αρχείο εξόδου PrintSur.txt για το πρόγραμμα "Cootransport_Simulation.exe" (δεύτερο μέρος).

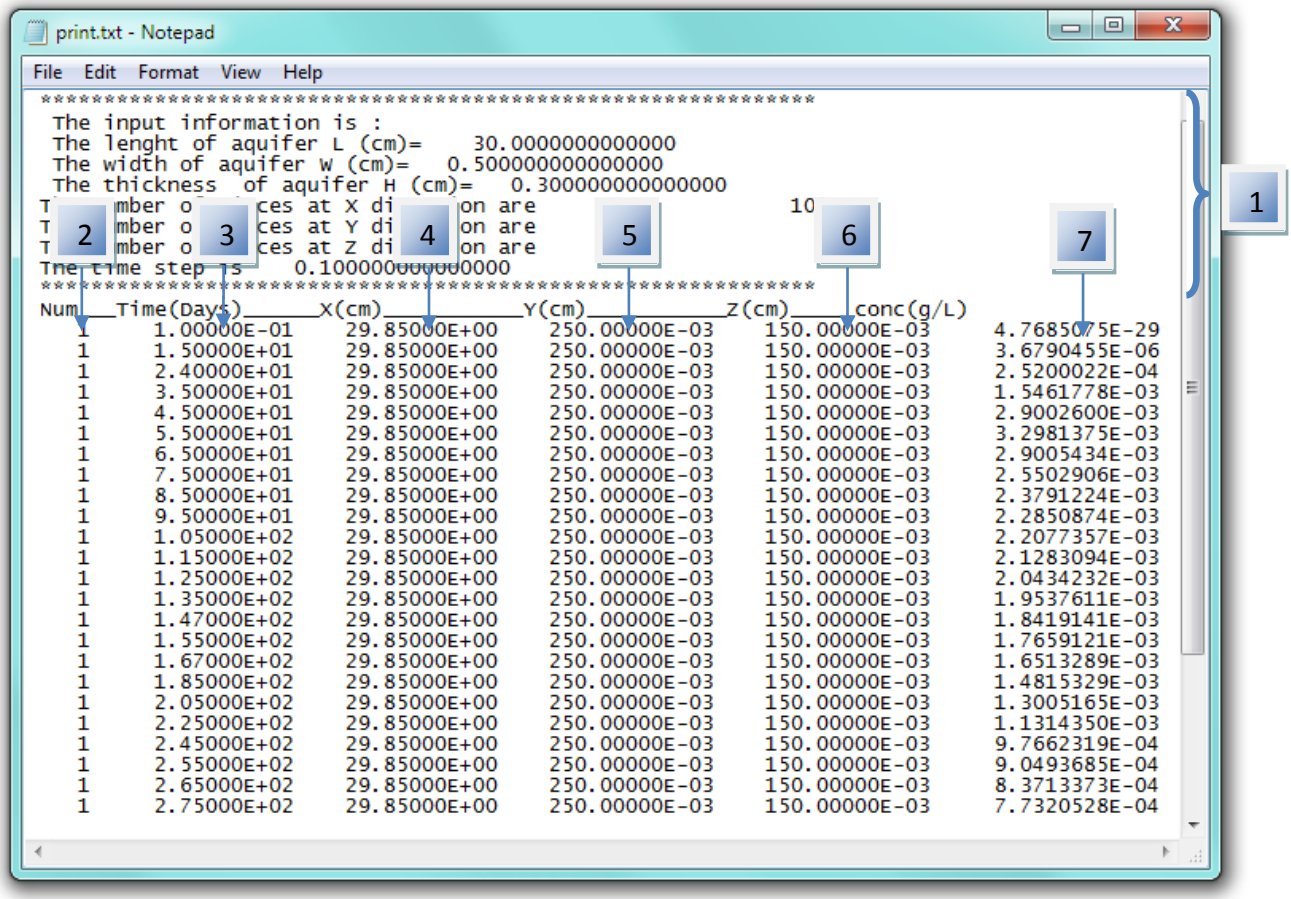


```
************************************************************
 The input information is :
 The lenght of aquifer L (cm)=    500.000000000000
 The width of aquifer W (cm)=    500.000000000000
 The thickness  of aquifer H (cm)=    100.000000000000
The number of pieces at X direction are          151
The number of pieces at Y direction are           51
The number of pieces at Z direction are           41
************************************************************

_____________________________________________________________
_____________________________________________________________
 After t=   10.0000000000000      (Days) have passed
X(cm)_____________________Z(cm)_______
  1.65562913907285        1.21951219512195        8.031568707138275E-007
  1.65562913907285        3.65853658536585        8.031723267075377E-007
  1.65562913907285        6.09756097560976        8.032028792777816E-007
  1.65562913907285        8.53658536585366        8.032478177498575E-007
  1.65562913907285        10.9756097560976        8.033060963439302E-007
  1.65562913907285        13.4146341463415        8.033763579880585E-007
  1.65562913907285        15.8536585365854        8.034569652981122E-007
  1.65562913907285        18.2926829268293        8.035460380749562E-007
  1.65562913907285        20.7317073170732        8.036414965216027E-007
  1.65562913907285        23.1707317073171        8.037411092465829E-007
  1.65562913907285        25.6097560975610        8.038425449979042E-007
  1.65562913907285        28.0487804878049        8.039434269683160E-007
  1.65562913907285        30.4878048780488        8.040413884310772E-007
  1.65562913907285        32.9268292682927        8.041341284099058E-007
```

Σχήμα 10.8: Παρουσιάζεται αρχείο εξόδου PrintSURWCOORD.txt για το πρόγραμμα "Cootransport_Simulation.exe".

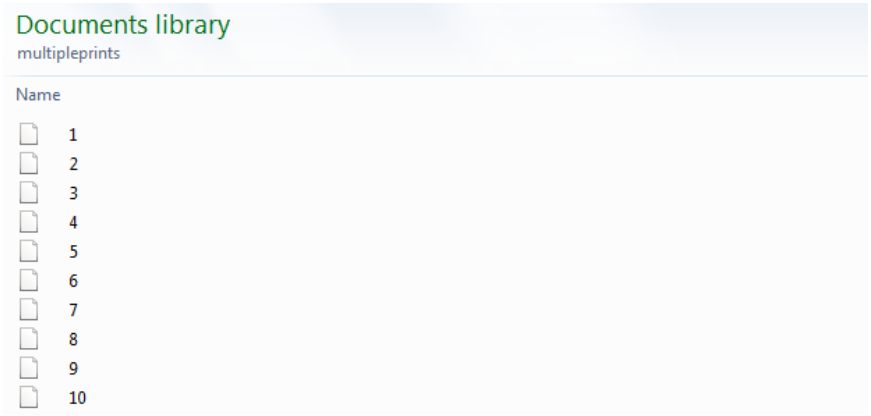
Εάν χρησιμοποιηθεί σωστά το Interface και εισαχθούν τα απαραίτητα δεδομένα θα ξεκινήσει η προσομοίωση. Εάν υπάρξει κάποιο σφάλμα στα δεδομένα (αρνητικός χρόνος) εμφανίζεται το ανάλογο μήνυμα και ο χρήστης καλείται να διορθώσει το interface και να ξανατρέξει το πρόγραμμα από την αρχή. Στην περίπτωση πού δεν υπάρχει κανένα σφάλμα τότε ξεκινούν οι υπολογισμοί και βρίσκονται οι συγκεντρώσεις στο τέλος του χρονικού βήματος dt. Στην συνέχεια διαβάζονται οι γραμμές της επιλογής 34 Σχήμα 10.6b, που περιγράφουν τα σημεία στα οποία ζητείται να βρεθεί η συγκέντρωση. Εάν ο χρόνος που έχει περάσει είναι μικρότερος από τον χρόνο τον οποίο ζητείται το σημείο τότε γίνονται και άλλοι υπολογισμοί ώστε να βρεθεί τα αποτέλεσμα ύστερα από ένα ακόμα dt. Αυτό γίνεται μέχρι ο χρόνος που σχηματίζεται από το άθροισμα των βημάτων dt να είναι μεγαλύτερος από τον χρόνο που απαιτεί το σημείο. Στο τέλος γίνεται η εκτύπωση των αποτελεσμάτων σε αρχείο print.txt Σχήμα 10.9.



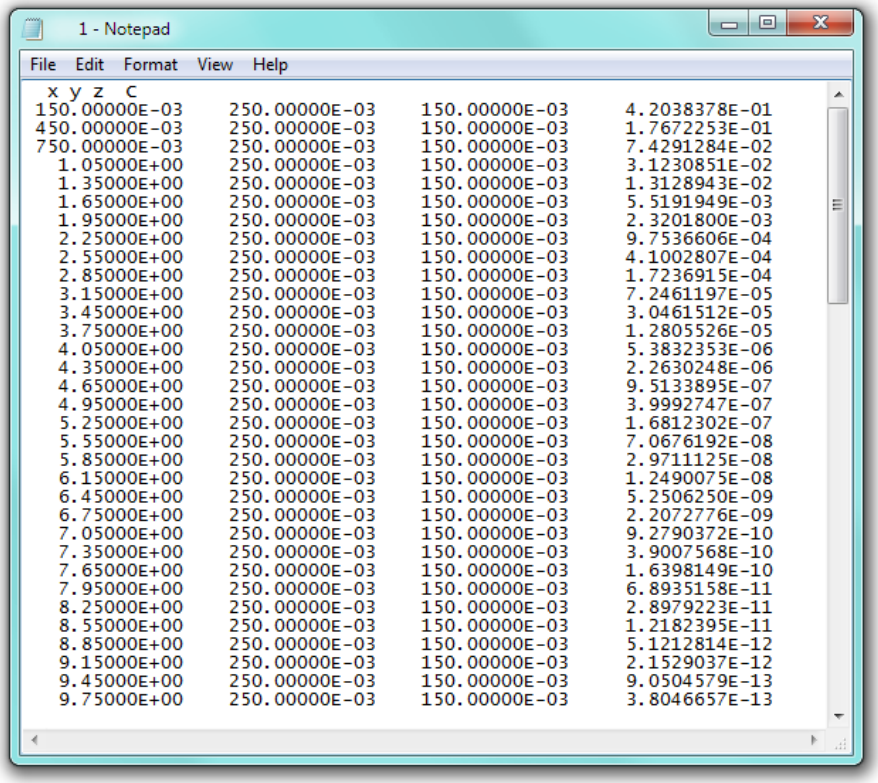
Σχήμα 10.9: Παρουσιάζεται αρχείο εξόδου Print.txt για το πρόγραμμα "Cootransport_Simulation.exe".

Στο αρχείο "print.txt" συγκεντρώνονται τα αποτελέσματα που ορίστηκαν στην επιλογή 34 Σχήμα 10.6b. Ίσως είναι το πιο σημαντικό αρχείο εξόδου αφού παρουσιάζει αναλυτικά και σε μορφή "kind time x y z Concentration" τις συγκεντρώσεις που ζητήθηκαν από τον χρήστη. Πιο συγκεκριμένα στο Σχήμα 10.9 το σύνολο δεδομένων (1) περιέχει συνοπτικά τις κυριότερες παραμέτρους που δόθηκαν από το Interface. Η στήλη (2) δείχνει το είδος της συγκέντρωσης που εκτυπώθηκε στην παρούσα σειρά. Τα δυνατά είδη είναι (1 ή 2 ή 3), το πρώτο αντιστοιχεί στον ρύπο που εκτελεί απλή μεταφορά π.χ. άργιλος, το δεύτερο αντιστοιχεί στον ρύπο που εκτελεί συμμεταφορά π.χ. ιός ενώ το τελευταίος είδος αντιστοιχεί στο σύμπλοκο μεταξύ των δύο πρώτων ρύπων π.χ. ιός προσροφημένος επάνω στην άργιλο. Στην συνέχεια η στήλη 3 δίνει τον χρόνο στο τέλος του οποίου γίνεται ο υπολογισμός των συγκεντρώσεων. Οι στήλες 4, 5, και 6 δίνουν τις συντεταγμένες του σημείο (x,y,z αντίστοιχα) του οποίου έγινε ο υπολογισμός της συγκέντρωσης . Τέλος η συγκέντρωση που ζητήθηκε φαίνεται στην στήλη 7, είναι δε γραμμένη σε εκθετική μορφή.

Η μορφή της τελευταίας κατηγορίας αρχείων εξόδου φαίνεται στο Σχήμα 10.10a,b.



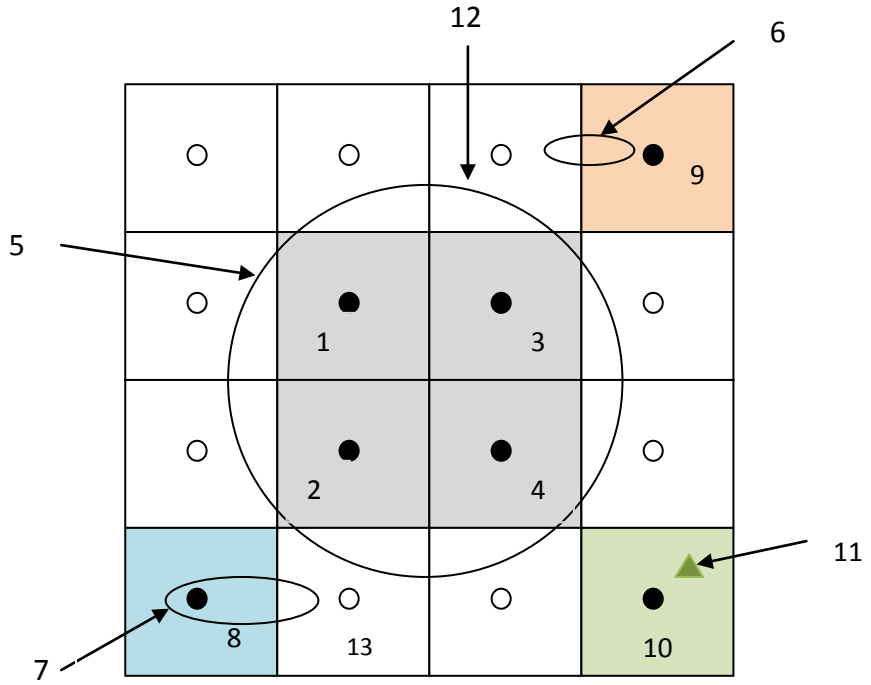
Σχήμα 10.10a: Παρουσιάζεται η μορφή της τελευταίας κατηγορίας αρχείων εξόδου από το πρόγραμμα "Cootransport_Simulation.exe". Η ονομασία τους είναι απλή και διέπεται από μια αύξων αρίθμηση. Το επόμενο αρχείο περιέχει τις συγκεντρώσεις του ακριβώς επόμενου χρονικού βήματος. Στο εσωτερικό των αρχείων περιέχονται όλες οι συγκεντρώσεις των κελιών του υδροφορέα Σχήμα 10.10b.



Σχήμα 10.10b: Παρουσιάζεται αρχείο εξόδου για το πρόγραμμα "Cootransport_Simulation.exe". Περιέχει συγκεντρώσεις σε μορφή X, Y, Z, C. Ο χρόνος στον οποίο αναφέρονται όλες οι συγκεντρώσεις είναι ίδιος (το τέλος κάθε χρονικού βήματος).

Στο Σχήμα 10.10a,b παρουσιάζονται αρχεία εξόδου με την εξής λογική. Στο τέλος κάθε χρονικού βήματος δημιουργείται αρχείο με απλό όνομα έναν αύξων αριθμό, το οποίο περιέχει όλες τις συγκεντρώσεις των κελιών του υδροφορέα. Η θέση που βρίσκονται τα αρχεία αυτά είναι μέσα στον υποφάκελο multiplePrints, ο οποίος βρίσκεται μέσα στον ίδιο φάκελο με τον αρχείο "Cootransport_Simulation.exe". Το κάθε αρχείο **Σχήμα 10.10b** περιέχει τέσσερις στήλες, εκ των οποίων οι τρείς πρώτες δίνουν τις συντεταγμένες του σημείου, του οποίου η συγκέντρωση εκτυπώνεται, ενώ η τελευταία δίνει ακριβώς αυτήν την συγκέντρωση (το είδος της οποίας καθορίζεται από την επιλογή 35 Σχήμα 10.6b).

Ακόμα η συγκέντρωση η οποία εκτυπώνεται ανήκει στο κέντρο το κελιού που είναι πιο κοντά στο σημείο που ζητείται. Και αντίστοιχα οι συντεταγμένες που καταγράφονται ανήκουν ακριβώς στο κελί αυτό και δεν είναι οι συντεταγμένες που διαβάστηκαν από την επιλογή 34 Σχήμα 10.6b. Στο Σχήμα 10.12 παρουσιάζονται διαφορετικές περιπτώσεις όπου μια ελλειπτική πηγή προσομοιώνεται από ένα πεπερασμένο αριθμό κελιών. Θεωρούμε ότι ένα κελί ανήκει στην πηγή όταν το κέντρο του βρίσκεται μέσα σε αυτήν. Έτσι η πηγή 5 προσομοιώνεται από τα κελιά 1, 2, 3 και 4. Στην περίπτωση που η πηγή (πχ. πηγή 7) έχει μέσα της μόνο ένα κέντρο κελιού (κελί 8), τότε αυτή προσομοιώνεται μόνο από το κελί αυτό. Αν για κάποιο λόγο στην περίμετρο της πηγής δεν περικλείεται κανένα κέντρο, όπως στην περίπτωση πηγή 6 με κελιά 12 και 9, τότε αυτόματα επιλέγεται το πιο δεξί κελί, σαν αυτό που πρέπει να αντιπροσωπεύει την ελλειπτική πηγή, και στην περίπτωση μας το κελί 9. Η ίδια λογική ισχύει και για την περίπτωση που προσομοιώνουμε ελλειψοειδή πηγή (ελλειπτική πηγή στον χώρο).



Σχήμα 10.11: Παρουσιάζονται διαφορετικές περιπτώσεις όπου η ελλειπτική πηγή προσομοιώνεται από ένα πεπερασμένο αριθμό κελιών.

Τέλος οι δυνατότητες του ParDiso όπως εφαρμόζεται στον παρόν κώδικα (σε αυτήν την έκδοση MKL 10.2.2.025) έχουν όριο διακριτοποίησης 101x101x101 στην διεύθυνση X,Y και Z αντίστοιχα. Το νούμερο αυτό μπορεί να φαίνεται μικρό αλλά αντιστοιχεί σε πίνακα με 4 $x10^9$ στοιχεία κα απαιτεί περισσότερα από 12 gigabyte σκληρού δίσκου.

Αν για κάποιο λόγο η εκτέλεση σταματήσει και εμφανιστεί κάποιος αριθμός σφάλματος τότε το manual του Pardiso ορίζει τα εξής Σχήμα 10.12:

error INTEGER

The error indicator according to the below table:

error	**Information**
0	no error
-1	input inconsistent
-2	not enough memory
-3	reordering problem
-4	zero pivot, numerical factorization or iterative refinement problem
-5	unclassified (internal) error
-6	preordering failed (matrix types 11, 13 only)
-7	diagonal matrix problem
-8	32-bit integer overflow problem
-9	not enough memory for OOC
-10	problems with opening OOC temporary files
-11	read/write problems with the OOC data file

Σχήμα 10.12: Παρουσιάζονται τα είδη των σφαλμάτων που μπορεί να εμφανίσει το ParDiso.

Κανείς βλέποντας την πιθανή σημασία των λαθών πράττει τις απαραίτητες διορθώσεις ώστε αυτά να σταματήσουν να εμφανίζονται. Σε γενικές γραμμές είναι σχετικά δύσκολο να παρουσιαστούν αλλά και να γίνει κάτι τέτοιο μια μείωση στην διακριτοποίηση του πορώδους συνήθως λύνει το πρόβλημα. Σε κάθε περίπτωση η χρήση του pardiso είναι παραπάνω από ικανοποιητική, καθώς καταφέρνει να χρησιμοποιήσει στο έπακρον τις δυνατότητες της τελευταίας γενιάς υπολογιστών (multiple core cpus, large ram capacity) μειώνοντας έτσι τους χρόνους επίλυσης εκθετικά. Μια απλή σύγκριση του pardiso με έναν συνηθισμένο open source code (Lu decomposition) δίνει τα εξής αποτελέσματα. Για τον ίδιο πίνακα το pardiso χρειάστηκε περίπου 1 sec, ενώ ο ελεύθερος κώδικας παραπάνω από 45 mins για την πλήρη επίλυση του. Η χρήση του "Cootransport_Simulation.exe" μπορεί να φαίνεται πολύπλοκη αλλά στην πράξη είναι πολύ εύχρηστη καθώς έχει μεγάλη δυνατότητα για παραμετροποίηση. Επιπλέον όλοι οι φυσικοί παράμετροι του μοντέλου αποθηκεύονται στο σκληρό δίσκο με αποτέλεσμα η επαναφορά και η επεξεργασία τους να είναι πολύ εύκολη. Ακόμα και για τους δύσπιστους μια απλή εκτέλεση του Interface.exe θα λύσει οποιαδήποτε αμφιβολία περί της χρηστικότητα του παρόντος λογισμικού.

11. ΚΩΔΙΚΑΣ FORTRAN

Στις επόμενες σελίδες παρουσιάζεται ο πηγαίος κώδικας του λογισμικού "Cootransport_Simulation.exe" (fortran source code) όπως αναπτύχθηκε και περιγράφθηκε στα κεφάλαια της διατριβής αυτής. Δεν κρίνεται σκόπιμο να παρουσιαστεί ο κώδικας για το interface (Visual basic 2008) καθώς είναι άνευ αριθμητικής (μαθηματικής) σημασίας. Παρ' όλα αυτά εάν κάποιος επιθυμεί να τον δει μπορεί να τον βρει στον οπτικό δίσκο που συνοδεύει την εργασία αυτή.

```fortran
Program finite_differences_implicit
use  AA_BB_Solver , only: dt,Cvceq,Cvcaeq  ! uses the module to share globally some intresting variables
implicit none



real*8 Rcadc,Lca  !Rcadc=r2= reverse rate coefficient    ,Lca=l2=l*=decay rate of sorbed solute
real*8 Rcdca,Lc   !Rcdca=r1=forward rate coefficient ,=Lc=l1=decay rate of liquid phase solute
real*8 Rvdva,Rvadv,Rvcdv,Rvdvc,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc  ! forward or reverse rates cooefficients  (Virus case)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
real*8 Lv,Lva      ! Lva=l2=l*=decay rate of sorbed solute, Lv=l1=decay rate of liquid phase solute
real*8 U,Dcx,Dcy,th,thh,rD !Dcx,Dcy  hydrodynamic dispersion coefficient of the colloid component
                      !th=thh=porosity (or th=th* the effective porosity can be used as well  page 509) ,rD= bulk density of the solid matrix(solid mass/aquifer volume)

real*8 lx0_C,ly0_C,lz0_C ,Dcz    !lx0,ly0,lz0 =cartesian coordinates of the point source colloid case
                         !Dcz=Vertical hydronamic dispersion coefficient of the colloid component
real*8 lx0_V,ly0_V,lz0_V !lx0,ly0,lz0 =cartesian coordinates of the point source Virus case
real*8 Dvx,Dvy,Dvz      !Dvx,Dvy  hydrodynamic dispersion coefficient of the virus component
                        !Dvz=Vertical hydronamic dispersion coefficient of the virus component

real*8 Dvcx,Dvcy,Dvcz !Dcvx,Dcvy,Dcvz =hydrodynamic components (x,y,z direction) for the case when virus is attached to the colloid
real*8 Lvc,Lvca ! decay rates coefficients  when virus is attached to the colloid, and virus is attached to the colloid and all together sorbed on the solid matrix


real*8 lxx,lyy,lzz ! lxx,lyy,lzz the dimensions of the aquifer
integer nX,nY,nZ ! nX,nY,nZ the number of pieces the user wants to split the aquifer to ,at  x,y,z directions
real*8  F_C, Cm0_C,Q_C ! F_C= general functional form of colloid source configuration [ML-3 t-1]=[gr cm-3 Day-1], colloid case
                      ! Cm0_C  source (source cell)  constant conccetration ,Colloid case
                      ! Q_C is the flow rate with which we transfer soltuted mass (Concetration Cm0)to the source cell Colloid case
real*8  F_V    !F_v= general functional form of colloid source configuration [ML-3 t-1]=[gr cm-3 Day-1], Virus case
real*8  Q_V    ! is the flow rate with  which we transfer soltuted mass (Concetration Cm0)to  the source  cell    [cm^3/day]For the virus case

real*8  Cm0_V  !The constant    concetration    of  the flow rate   [mgr/ml] Virus case



Integer printSurMode !Allows to print  into the file PrintSur.txt
                     ! a particular surface of the aquifer at a requested height H, for every time step Dt  (if 0 we print nothing ,if 1 we print for every time step).
integer PrintSurXYZ   !Mode (1 =yz plane is being printed,2=xz plane is being printed, =3 yz planed is being printed)
real*8 h !H determines the height of the surface to be printed. (cm)
real*8 wm ! wm determines the numerical mode that will be  used for the soliving of the tranport or cootranport problem. If wm=0.5 we have crank nikolson . If wm=1 we have full implicit scheme
       ! wm should be within the range  0.5-1. Else we fall to explicit scheme the and stability is not ensured
integer sourceSurf_C ! sourceSurf_C determines on which plane the elliptic source is based  (Colloid source)
                    !(sourceSurf=0  we dont have an elliptic source but a point one, =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is based on XY plane)
integer sourceSurf_V ! sourceSurf_V determines on which plane the elliptic source is
```

```
    based  (Virus source)
42                  !(sourceSurf_V=0  we dont have an elliptic source but a point one,
    =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is
    based on XY plane)
43 integer Source_Type_C  ! the type of the Colloid Source,=1 means constant source
    concetration,=2 means constant mass flow! attention constant source concetration
    can be valid only for influent cells
44
45 integer Source_Type_V  ! the type of the Virus Source,=1 means constant source
    concetration,=2 means constant mass flow! attention constant source concetration
    can be valid only for influent cells
46
47 real*8 aEl_C,bEl_C,cEl_C  !aEl semi-axis of the elliptic source Colloid case! check the
     manual for more info
48                 !bEl semi-axis of the elliptic source !Check the manual for more info
49                 !cEl semi-axis of the elliptic source !Check the manual for more info

50 real*8 aEl_V,bEl_V,cEl_V  !aEl semi-axis of the elliptic source Virus case ! check the
    manual for more info
51                 !bEl semi-axis of the elliptic source !Check the manual for more info
52                 !cEl semi-axis of the elliptic source !Check the manual for more info

53 real*8 time_begin ,time_end
54 real*8 tp_C ! tp is the active time of the Colloid source (time within the source
    provides with mass the model)
55 real*8 tp_V ! tp is the active time of the Virus source (time within the source
    provides with mass the model)
56 real*8 dboundaryFactor,uboundaryFactor !dboundary factor indicates how many times the
    concetration of the outter imaginary boundary cell,
57                    ! at the end off the x direction, is smaller than the
    concetration of the previous cell (which is in the aquifer limits)
58                    ! uboundaryFactor= is the same like dboundaryFactor ,but for the
    upstream boundary cells
59 integer CooTransp     ! Indicates wether we have simple transport (=0) or Cootransport
    (=1)
60 integer nMaxCcVectors ! Responsible for holding the max allowed Cc vectors. Since we
    require to aquire the solutions for the colloid component and then
61                    ! use them to get the solutions for the Virus Component. We can
    store the results that Colloid transportation produced and then use them
62                    ! to solve virus component. By doing so we avoid factorizing
    again and again the same matrix, without having to keep in memory the same time
63                    ! and the A matrix for the colloid component and the A matrix for
     the Virus component. Keep in mind that nMaxVectors require as well a lot of
64                    ! memory. By giving  larger nMaxCcVectors we gain speed but we
    need larger amount of memory. A valuearound 50 should be good.
65
66
67
68 common/VarTrans1/Dcx,Dcy,Dcz,Lc,Lca,Rcdca,Rcadc
69 common/VarTrans1a/u
70 common/VarTran2/rD,th
71 common/vartrans4/dboundaryFactor,uboundaryFactor
72 common/vartrans5/wm
73 common/vartrans6/Lv,Lva,Rvdva,Rvadv
74 common/vartrans7/Dvx,Dvy,Dvz,Dvcx,Dvcy,Dvcz
75 Common/vartrans8/Rvdvc,Rvcdv,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc
76 Common/vartrans9/Lvc,Lvca
77
78 common/printMode/printSurMode,PrintSurXYZ,h
79 common/sourceGeom_C/sourceSurf_C
80 common/sourceGeom2_C/aEl_C,bEl_C,cEl_C
81 common/sourceCoord_C/lx0_C,ly0_C,lz0_C
82 common/sourcePower_C/Q_C,Cm0_C,thh
83
84 common/sourceGeom_V/sourceSurf_V
85 common/sourceGeom2_V/aEl_V,bEl_V,cEl_V
86 common/sourceCoord_V/lx0_V,ly0_V,lz0_V
87 common/sourcePower_V/Q_V,Cm0_V,nMaxCcVectors
88
89 common/Source_Type_trans/Source_Type_C,Source_Type_v
90
91
92
93   ! opens files for input and output
```

```
 94  open(111,File="input.txt",form="Formatted")
 95  open(112,File="inputC.txt",form="Formatted") ! cootranport extra input file
 96  open(222,File="printSurWCoord.txt",Form="Formatted")
 97  open(333,File="print.txt",Form="Formatted")
 98  open(444,File="printSur.txt",Form="Formatted")
 99  open(555,File="temp.txt",Form="Formatted",access="direct", RECL=140) ! temp file for
        direct access
100  open(777,File="AA_BB_results.txt",Form="Formatted")
101  open(889,File="inputMisc.txt",Form="Formatted")
102 ! intialization in case reading failed
103  !_________________Input Data_____________________!
104
105
106 Dcx=1331.25D0*24.D0
107 Dcy=268.75D0*24.D0 !lateral hydrodynamic dispersion coefficient cm^2/day
108 Dcz=Dcy
109 Dvx=Dcx
110 Dvy=Dcy
111 Dvz=Dcz
112 Dvcx=1
113 Dvcy=1
114 Dvcz=1
115 Lv=1
116 Lva=1
117 Lvc=1
118 Lvca=1
119 Rvcdv=1
120 Rvdvca=1
121 Rvcdvca=1
122 Rvcadv=1
123 Rvcadvc=1
124 Cvcaeq=1
125 Rvdva=1
126 Rvadv=1
127 nMaxCcVectors=50
128
129 U=0.625D0*24.D0  ! average interstitial velocity cm/day
130
131  Lc=0.D0 !decay rate of liquid phase solute  1/days
132  Lca=0.D0 !decay rate of sorbed solute        1/days
133
134 th=0.25D0 !porosity
135 thh=th
136 lx0_C=200.D0   ! Cartesian coordinates of the point source cm
137 ly0_C=0.D0
138 lz0_C=50.D0
139 lx0_V=200.D0   ! Cartesian coordinates of the point source cm
140 ly0_V=0.D0
141 lz0_V=50.D0
142
143 Rcdca=0.03*24.D0  ! forward rate coefficient
144 Rcadc=0.017*24.D0 !reverse rate coefficient
145 lzz=100.D0        ! finite aquifer thickness (cm)
146 Cm0_C=1.0D0
147 Cm0_V=1.0D0
148 dt=1
149 F_C=1.D0
150 F_V=1.0D0
151 Q_C=1
152 Q_V=1
153 printSurMode =1
154 PrintSurXYZ=2
155 h=0
156 sourceSurf_C=0
157 sourceSurf_V=0
158
159 aEl_C=0.D0
160 bEl_C=0.D0
161 cEl_C=0.D0
162
163 aEl_v=0.D0
164 bEl_v=0.D0
165 cEl_V=0.D0
166
```

```
167 tp_C=1.D0
168 tp_V=1.D0
169 dboundaryFactor =0.1D0
170 uboundaryFactor=1.D0
171 CooTransp=0
172 ! _____________________End of input Data_____________________!
173
174
175 ! Reading from input file
176 read(111,*) ! reads the first useless line ,contains names
177 read(111,*)Dcx
178 read(111,*) ! reads the first useless line ,contains names
179 read(111,*)Dcy
180 read(111,*) ! reads the first useless line ,contains names
181 read(111,*)Dcz
182 read(111,*) ! reads the first useless line ,contains names
183 read(111,*)U
184 read(111,*) ! reads the first useless line ,contains names
185 read(111,*)Lc
186 read(111,*) ! reads the first useless line ,contains names
187 read(111,*)Lca
188 read(111,*) ! reads the first useless line ,contains names
189 read(111,*)th
190 read(111,*) ! reads the first useless line ,contains names
191 read(111,*)lx0_C
192 read(111,*) ! reads the first useless line ,contains names
193 read(111,*)ly0_C
194 read(111,*) ! reads the first useless line ,contains names
195 read(111,*) lz0_C
196 read(111,*) ! reads the first useless line ,contains names
197 read(111,*)Rcdca
198 read(111,*) ! reads the first useless line ,contains names
199 read(111,*)Rcadc
200 read(111,*) ! reads the first useless line ,contains names
201 read(111,*)rD
202 read(111,*) ! reads the first useless line ,contains names
203 read(111,*)Q_C
204 read(111,*) ! reads the first useless line ,contains names
205 read(111,*)Cm0_C
206 read(111,*) ! reads the first useless line ,contains names
207 read(111,*)sourceSurf_C
208 read(111,*) ! reads the first useless line ,contains names
209 read(111,*)aEl_C
210 read(111,*) ! reads the first useless line ,contains names
211 read(111,*)bEl_C
212 read(111,*) ! reads the first useless line ,contains names
213 read(111,*)cEl_C
214 read(111,*) ! reads the first useless line ,contains names
215 read(111,*)lxx
216 read(111,*) ! reads the first useless line ,contains names
217 read(111,*)lyy
218 read(111,*) ! reads the first useless line ,contains names
219 read(111,*)lzz
220 read(111,*) ! reads the first useless line ,contains names
221 read(111,*)nx
222 read(111,*) ! reads the first useless line ,contains names
223 read(111,*)ny
224 read(111,*) ! reads the first useless line ,contains names
225 read(111,*)nz
226 read(111,*) ! reads the first useless line ,contains names
227 read(111,*)dt
228 read(111,*) ! reads the first useless line ,contains names
229 read(111,*)printSurMode
230 read(111,*) ! reads the first useless line ,contains names
231 read(111,*)PrintSurXYZ
232 read(111,*) ! reads the first useless line ,contains names
233 read(111,*)h
234 read(111,*) ! reads the first useless line ,contains names
235 read(111,*)tp_C
236 read(111,*) ! reads the first useless line ,contains names
237 read(111,*)wm
238 read(111,*) ! reads the first useless line ,contains names
239 read(111,*) dboundaryFactor
240 read(111,*) ! reads the first useless line ,contains names
```

```
241 read(111,*) uboundaryFactor
242 read(111,*) ! reads the first useless line ,contains names
243 read(111,*) CooTransp
244  thh=th ! updates the thh variable
245
246
247
248
249
250 read(112,*) ! reads the first useless line ,contains names
251 read(112,*)Dvx
252 read(112,*) ! reads the first useless line ,contains names
253 read(112,*)Dvy
254 read(112,*) ! reads the first useless line ,contains names
255 read(112,*)Dvz
256 read(112,*) ! reads the first useless line ,contains names
257 read(112,*)Dvcx
258 read(112,*) ! reads the first useless line ,contains names
259 read(112,*)Dvcy
260 read(112,*) ! reads the first useless line ,contains names
261 read(112,*)Dvcz
262 read(112,*) ! reads the first useless line ,contains names
263 read(112,*)Lv
264 read(112,*) ! reads the first useless line ,contains names
265 read(112,*)Lva
266 read(112,*) ! reads the first useless line ,contains names
267 read(112,*)Lvc
268 read(112,*) ! reads the first useless line ,contains names
269 read(112,*)Lvca
270 read(112,*) ! reads the first useless line ,contains names
271 read(112,*)Rvdva
272 read(112,*) ! reads the first useless line ,contains names
273 read(112,*)Rvadv
274 read(112,*) ! reads the first useless line ,contains names
275 read(112,*)Rvdvc
276 read(112,*) ! reads the first useless line ,contains names
277 read(112,*)Rvcdv
278 read(112,*) ! reads the first useless line ,contains names
279 read(112,*)Rvdvca
280 read(112,*) ! reads the first useless line ,contains names
281 read(112,*)Rvcdvca
282 read(112,*) ! reads the first useless line ,contains names
283 read(112,*)Rvcadv
284 read(112,*) ! reads the first useless line ,contains names
285 read(112,*)Rvcadvc
286 read(112,*) ! reads the first useless line ,contains names
287 read(112,*)Cvceq
288 read(112,*) ! reads the first useless line ,contains names
289 read(112,*)Cvcaeq
290 read(112,*) ! reads the first useless line ,contains names
291 read(112,*)Q_V
292 read(112,*) ! reads the first useless line ,contains names
293 read(112,*)Cm0_V
294 read(112,*) ! reads the first useless line ,contains names
295 read(112,*)tp_V
296 read(112,*) ! reads the first useless line ,contains names
297 read(112,*)lx0_V
298 read(112,*) ! reads the first useless line ,contains names
299 read(112,*)ly0_V
300 read(112,*) ! reads the first useless line ,contains names
301 read(112,*)lz0_V
302 read(112,*) ! reads the first useless line ,contains names
303 read(112,*)sourceSurf_V
304 read(112,*) ! reads the first useless line ,contains names
305 read(112,*)aEl_V
306 read(112,*) ! reads the first useless line ,contains names
307 read(112,*)bEl_V
308 read(112,*) ! reads the first useless line ,contains names
309 read(112,*)cEl_V
310 read(112,*) ! reads the first useless line ,contains names
311 read(112,*)nMaxCcVectors
312
313
314 read(889,*) ! reads the first useless line ,contains names
```

```
315 read(889,*) ! reads the first useless line ,contains names
316 read(889,*) ! reads the first useless line ,contains names
317 read(889,*) Source_Type_C
318 read(889,*) ! reads the first useless line ,contains names
319 read(889,*) Source_Type_V
320
321
322
323  ! Prints to file what was read to ensure  input file was read correctly
324  ! we check the quality of the input data
325 call   validating_input(lxx,lyy,lzz,nx,ny,nz,tp_C,tp_V,CooTransp)
326
327
328
329  ! Reading from input file
330 write(444,*)"************************************************"
331 write(444,*) "Input file was succesfully read"
332 write(444,*)"Dx= ",Dcx
333 write(444,*)"Dy= ",Dcy
334 write(444,*)"Dz= ",Dcz
335 write(444,*)"Dx= ",Dvx
336 write(444,*)"Dy= ",Dvy
337 write(444,*)"Dz= ",Dvz
338 write(444,*)"U= ",U
339 write(444,*)"l1= ",Lc
340 write(444,*)"l2= ",Lca
341 write(444,*)"th= ",th
342 write(444,*)"lx0= ",lx0_C
343 write(444,*)"ly0= ",ly0_C
344 write(444,*)"lz0= ",lz0_C
345 write(444,*)"r1= ",Rcdca
346 write(444,*)"r2= ",Rcadc
347 write(444,*)"rD= ",rD
348 write(444,*)"Q= ",Q_C
349 write(444,*)"Cm0= ",Cm0_C
350 write(444,*)"sourceSurf= ",sourceSurf_C
351 write(444,*)"aEl= ",aEl_C
352 write(444,*)"bEl=",bEl_C
353 write(444,*)"cEl= ",cEl_C
354 write(444,*)"lxx= ",lxx
355 write(444,*)"lyy= ",lyy
356 write(444,*)"lzz= ",lzz
357 write(444,*)"nx= ",nx
358 write(444,*)"ny= ",ny
359 write(444,*)"nz= ",nz
360 write(444,*)"dt= ",dt
361 write(444,*)"printSurMode= ",printSurMode
362 write(444,*)"PrintSurXYZ= ",PrintSurXYZ
363 write(444,*)"h= ",h
364 write(444,*)"************************************************"
365
366
367 write(*,*) "nx read",nx,"ny read", ny,"nz read", nz
368
369
370
371 !_____________________________________________________________________________
372 CALL CPU_TIME ( time_begin )
373
374
375 write(222,*) "************************************************************"
376 write(222,*) " The input information is :"
377 write(222,*) " The lenght of aquifer L (cm)= ",lxx
378 write(222,*) " The width of aquifer W (cm)= ",lyy
379 write(222,*) " The thickness  of aquifer H (cm)= ",lzz
380 write(222,*)"The number of pieces at X direction are ",nx
381 write(222,*)"The number of pieces at Y direction are ",ny
382 write(222,*)"The number of pieces at Z direction are ",nz
383 write(222,*) "************************************************************"
384 write(333,*) "************************************************************"
385 write(333,*) " The input information is :"
386 write(333,*) " The lenght of aquifer L (cm)= ",lxx
387 write(333,*) " The width of aquifer W (cm)= ",lyy
388 write(333,*) " The thickness  of aquifer H (cm)= ",lzz
```

```
389 write(333,*)"The number of pieces at X direction are ",nx
390 write(333,*)"The number of pieces at Y direction are ",ny
391 write(333,*)"The number of pieces at Z direction are ",nz
392 write(333,*)"The time step is  ",dt
393 write(333,*) "************************************************************"
394 write(333,*)"Num___Time(Days)______X(cm)___________Y(cm)___________Z(cm)_____conc(g/L)"
395
396 call main(nx,ny,nz,lxx,lyy,lzz,tp_C,tp_V)! calls the main subroutine to solve the ↙
        problem
397
398 CALL CPU_TIME ( time_end ) !ends the counter
399
400 write(222,*)"All the calculations where finished in",int((time_end - time_begin)/60.0), ↙
        "mins  +",((time_end - time_begin)/60.0-int((time_end - time_begin)/60.0))*60,"secs↙
        "
401 write(*,*)"All the calculations where finished in",int((time_end - time_begin)/60.0),   ↙
        "mins  +",((time_end - time_begin)/60.0-int((time_end - time_begin)/60.0))*60,"secs↙
        "
402
403 !_____________________________________________________________________________________
404
405
406
407  write(*,*) "Reading completed"
408  !read(*,*) ! slows down exit
409  end
410
411
```

```
1  subroutine  isBoundaryCell (BoundaryCell,i0,j0,w0,nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2,dt,A1,
       A2,A3,A4,A5,A6,A7,A8,flag3,flag4,flag5,flag6,flag7,flag8)
2    ! this function is responsible for definiing whether a cell is boundary or not and
       accordingly enforces the neccessary
3    ! boundary conditions so the A matrix can properly be created
4    ! AxC=fCn   .fCn vector conctains a function of C (non sorbed ) concetrations and
       then Cs (sorbed) Concetrations
5    ! fCn_vector depends of Ccnp1 to produce fCn!!!
6  implicit none
7    logical  BoundaryCell  ! It  will indicate  if this is a boundary cell or not .  True
        means this is actually a boundary cell
8                           !
9    integer i0,j0,w0 ! i0,j0,w0 are the coordinates of the cell ,for which we want to
       create the  A
10             ! coeffiecients (equations (1) and (2) are being silently written for each
        cell ,check page 509)
11   integer nx,ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer
       to,at  x,y,z directions.
12  real*8 A1,A2,A3,A4,A5,A6,A7,A8 ! coeffiecients
13
14  real*8 Dx,Dy,Dz !Dx,Dy,Dz longitudinal hydrodynamic dispersion coefficient
15 real*8 r1,r2,l1,l2 !r1=forward rate coefficient ,r2= reverse rate coefficient,l1=decay
       rate of liquid phase solute
16                   ! l2=l*=decay rate of sorbed solute
17 real*8 rD,th  !th=porosity (or th=th* the effective porosity can be used as well  page
       509) ,rD= bulk density of the solid matrix(solid mass/aquifer volume)
18 real*8 U !U average interstitial velocity
19 real*8 dxx,dyy,dzz,dt ! dxx=the lenght of  each cell the total aquifer is splitted to.
20                       ! dyy=the witdth of  each cell the total aquifer is splitted to.
21                       ! dzz=the height of  each sell the total aquifer is splitted to.
22                       ! dt= the  (time) distance between two sequential time moments t1
       ,t2  =dt=t2-t1
23 real*8 dboundaryFactor !boundary factor indicates how many times the concetration of
       the outter imaginary boundary cell,
24                       ! at the end off the x direction, is smaller than the
       concetration of the previous cell (which is in the aquifer limits)
25 real*8 uBoundaryFactor ! the same like dboundaryFactor but for the upstream boundary
       cells
26 real*8 wm ! wm determines the numerical mode that will be  used for the soliving of the
        tranport or cootranport problem. If wm=0.5 we have crank nikolson . If wm=1 we
       have full implicit scheme
27 logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out if the according
       coefficients should be added to the A matrix or not .! true means they should be
       added
28
29
30 common/VarTrans1a/u
31 common/VarTran2/rD,th
32 common/VarTrans3/dxx,dyy,dzz
33 common/vartrans4/dboundaryFactor,uboundaryFactor
34 common/vartrans5/wm
35  !initializing values in case something went wrong during reading
36 A1=0
37 A2=0
38 A3=0
39 A4=0
40 A5=0
41 A6=0
42 A7=0
43 A8=0
44 flag3=.True.
45 flag4=.True.
46 flag5=.True.
47 flag6=.True.
48 flag7=.True.
49 flag8=.True.
50  BoundaryCell=.False.
51 ! end of initialization
52
53 if (i0==1 .or.i0==nx.or.j0==1 .or.j0==ny.or.w0==1 .or.w0==nz) then ! whenever a cell is
        at the edges it is supposed to be boundary
54   BoundaryCell=.True.
55 else
56   BoundaryCell=.False.
```

```
goto 11 ! the cell wasnt   boundary   and  so we exit the function
end if

            ! calculating A1-A8 concetration coefficients

            A1=(1/dt+wm*(2.D0*Dx/dxx**2.D0   +2.D0*Dy/dyy**2.D0  +2.D0*Dz/dzz**2.D0 +l1
     ))

            A2=(1/dt+wm*l2)*rD/th

            A3=wm*(-Dx/dxx**2.D0 -U/(2.D0*dxx))

            A4=wm*(-Dx/dxx**2.D0  +U/(2.D0*dxx))

            A5=-wm*Dy/dyy**2.D0

            A6=-wm*Dy/dyy**2.D0

            A7=-wm*Dz/dzz**2.D0

            A8=-wm*Dz/dzz**2.D0

   ! applying boundary conditions

if (i0==1 ) then
flag3=.false. ! will be united with A1 and wont be writted seperately in the A matrix
A1=A1+ uboundaryFactor*A3 !+0.001*A3 !A1=A1+A3 would normally be but A3 tends to zero ,
    since we suppose the that i-1,j,w cells tend to have zero concetration A1=A1+
    boundaryFactor*A3
      ! upstream boundary condition tends to infinite aquifer ! we could also use a
    factor
end if

if (i0==nx ) then
! in case we have v dc/dx =0 boundary
! the hydrodynamic part of the A4(coeffiecient of the Ci+1 cell) will be united with
    the A1 and wont be written seperately though
! the advection part of A4 will be subtracted from A4 because we suppose a boundary
    downstream  condition V(C+1-C-1)/(2dx)=g(x) exists.
! this condition obviously effects  the A3 (coefficient fo the Ci-1 cell) and again we
    need to remove the advection part from it
! But A3 and A4 have Hydrodynamic diffusion part which in order to work properly we
    need toconsider that
! the Ci+1 cells(we are out of the aquifer so Ci+1 is imaginary) have only a portion of
     the concetration i,j,w cells have
! this portion is determined by the boundaryFactor parameter
 !A3=-dt*Dx/dxx**2.D0  !Normally A3=-dt*Dx/dxx**2.D0 -dt*U/(2.D0*dxx)
 !A4=-dt*Dx/dxx**2.D0  !Normally A4=-dt*Dx/dxx**2.D0  +dt*U/(2.D0*dxx)
 ! though here we dont have

 flag4=.false. ! the hydrodynamic part of the A4(coeffiecient of the Ci+1 cell) will be
     united with the A1 and wont be written seperately
 A1=A1+dBoundaryFactor*A4  !the  i+1 cells have only a portion of the concetration i,j,
    w cells have    A1=A1+boundaryFactor*A4
                        ! downstream boundary condition tends to infinite aquifer
end if

if (j0==1 ) then
flag5=.false. ! will be united with A1 and wont be writted seperately in the A matrix
A1=A1+A5
end if

if (j0==ny ) then
flag6=.false. ! will be united with A1 and wont be writted seperately in the A matrix
A1=A1+A6
end if

if (w0==1 ) then
flag7=.false. ! will be united with A1 and wont be writted seperately in the A matrix
A1=A1+A7
end if

if (w0==nz ) then
flag8=.false. ! will be united with A1 and wont be writted seperately in the A matrix
```

```
120 A1=A1+A8
121 end if
122
123  ! A2 isnt affected at all from the boundary conditions because it is interested only in the i0,j0,w0 coordinates
124  11 end subroutine
125
126
127
128
129
130
131
132
133
134
135
136
137
138  !For the colloid, fCn boundaries
139
140  subroutine  isBoundaryCellfCn (i0,j0,w0,nx,ny,nz,Ccijktn,Ccim1jktn,Ccip1jktn,Ccijm1ktn,Ccijp1ktn,Ccijkm1tn,Ccijkp1tn,flag3,flag4,flag5,flag6,flag7,flag8)
141   ! this function is responsible for defininig whether a cell is boundary or not and accordingly enforces the neccessary
142   ! boundary conditions so the fcn vector can properly be created
143   ! AxC=fCn   .fCn vector conctains a function of C (non sorbed ) concetrations and then Cs (sorbed) Concetrations
144   ! fCn_vector depends of Ccnp1 to produce fCn!!!
145  implicit none
146
147   integer i0,j0,w0 ! i0,j0,w0 are the coordinates of the cell ,for which we want to create the  A
148   integer nx,ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions.
149   real*8  Ccijktn, Ccim1jktn, Ccip1jktn, Ccijm1ktn, Ccijp1ktn, Ccijkm1tn, Ccijkp1tn !
150   ! Ccijktn, Ccim1jktn, Ccip1jktn, Ccijm1ktn, Ccijp1ktn, Ccijkm1tn, Ccijkp1tn= usefull temp variables holding concetrations of the present t=tn time
151   logical  BoundaryCell  ! It  will indicate  if this is a boundary cell or not .  True means this is actually a boundary cell
152   real*8 dboundaryFactor !boundary factor indicates how many times the concetration of the outter imaginary boundary cell,
153                          ! at the end off the x direction, is smaller than the concetration of the previous cell (which is in the aquifer limits)
154   real*8 uBoundaryFactor ! the same like dboundaryFactor but for the upstream boundary cells
155   logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out   whether boudary conditions the corresponding Concetrations
156         !Ccijktn, Ccim1jktn, Ccip1jktn, Ccijm1ktn, Ccijp1ktn, Ccijkm1tn, Ccijkp1tn will be calculated without any changes from the boundary conditions or not
157         ! Flag=True means no boundary effect is needed
158   common/vartrans4/dboundaryFactor,uboundaryFactor
159
160   !initializing variables
161   BoundaryCell=.False. ! in the begining we assume that the cell isnt a boundary one
162   flag3=.True. ! All Flags are true which means that for the calculations of the fcn no boundary conditions need to be enforced
163   flag4=.True.
164   flag5=.True.
165   flag6=.True.
166   flag7=.True.
167   flag8=.True.
168   ! End of initialization
169   if (i0==1 .or.i0==nx.or.j0==1 .or.j0==ny.or.w0==1 .or.w0==nz) then ! whenever a cell is at the edges it is supposed to be boundary
170     BoundaryCell=.True.
171   else
172     BoundaryCell=.False.
173     goto 14 ! the cell wasnt   boundary   and  so we exit the function
174   end if
175
176
177
178 if (i0==1 ) then ! we got upstream cell
```

```
179 flag3=.false. ! Cell position requires that Ccim1jktn=Ccijktn*uboundaryFacto
180 Ccim1jktn=Ccijktn*uboundaryFactor !Ccim1jktn=Ccijktn  would normally be but Ccim1jktn
        tends to zero ,since we suppose the that i-1,j,w cells tend to have zero
        concetration Ccim1jktn=Ccijktn*uboundaryFactor
181                                  ! upstream boundary condition tends to infinite
        aquifer ! For that we could set the factor uboundaryFactor=1
182                                  ! Ccip1jktn is imaginary cell when i0==1
183 end if
184
185 if (i0==nx ) then  ! we got downstram cell
186  flag4=.false.  ! Cell position requires that Ccip1jktn=dBoundaryFactor*Ccijktn
187  Ccip1jktn=dBoundaryFactor*Ccijktn   !Ccip1jktn= Ccijktn  would normally be but
        Ccim1jktn tends to zero ,since we suppose the that i-1,j,w cells tend to have zero
        concetration Ccim1jktn=Ccijktn*uboundaryFactor
188                                  ! downstream boundary condition tends to infinite
        aquifer ! For that we could set the factor uboundaryFactor=1
189                                  !Ccip1jktn is imaginary cell when i0==nx
190 end if
191
192 if (j0==1 ) then
193 flag5=.false. ! Ccijm1ktn is imaginary cell.  We can apply boundary condition dc/dy=0
        and so on
194 Ccijm1ktn=Ccijktn
195 end if
196
197 if (j0==ny ) then
198 flag6=.false. ! Ccijp1ktn is imaginary cell.  We can apply boundary condition dc/dy=0
        and so on
199 Ccijp1ktn=Ccijktn
200 end if
201
202 if (w0==1 ) then
203 flag7=.false. ! Ccijkm1tn is imaginary cell.  We can apply boundary condition dc/dz=0
        and so on
204 Ccijkm1tn=Ccijktn
205 end if
206
207 if (w0==nz ) then
208 flag8=.false. ! Ccijkp1tn is imaginary cell.  We can apply boundary condition dc/dz=0
        and so on
209 Ccijkp1tn=Ccijktn
210 end if
211
212  14 end subroutine
213
214  !End For the colloid, fCn boundaries
215
216
217
218
219
220  !For the colloid-virus Cootransport, fCnC boundaries
221
222  subroutine  isBoundaryCellfCnC (i0,j0,w0,nx,ny,nz,Cvijktn,Cvim1jktn,Cvip1jktn,
        Cvijm1ktn, Cvijp1ktn,Cvijkm1tn,Cvijkp1tn,AAijktn,AAim1jktn, AAip1jktn,AAijm1ktn,
        AAijp1ktn,AAijkm1tn,AAijkp1tn,AAijktnp1,AAim1jktnp1,AAip1jktnp1,AAijm1ktnp1,
        AAijp1ktnp1,AAijkm1tnp1,AAijkp1tnp1,flag3,flag4,flag5,flag6,flag7,flag8)
223   ! this function is responsible for defininig whether a cell is boundary or not and
        accordingly enforces the neccessary
224   ! boundary conditions so the fCn and fCnC vector can properly be created
225   ! AxC=fCnc   .fCnc vector conctains a function of C (non sorbed ) concetrations and
        then Cs (sorbed) Concetrations
226   ! fCnC_vector depends of Cvnp1 to produce fCnC, for the case of virus-colloid
        Cootranport
227  implicit none
228
229   integer i0,j0,w0 ! i0,j0,w0 are the coordinates of the cell ,for which we want to
        create the  A
230   integer nx,ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer
        to,at  x,y,z directions.
231   real*8    Cvijktn, Cvim1jktn, Cvip1jktn, Cvijm1ktn, Cvijp1ktn, Cvijkm1tn, Cvijkp1tn !
232             !  Cvijktn, Cvaijktn, Cvim1jktn, Cvip1jktn, Cvijm1ktn, Cvijp1ktn, Cvijkm1tn,
         Cvijkp1tn = usefull temp variables holding concetrations of the present t=tn time
233   real*8    AAijktn,  AAim1jktn, AAip1jktn, AAijm1ktn, AAijp1ktn, AAijkm1tn, AAijkp1tn
```

```
    !usefull temp variables holding concetrations of the present on t=tn
234   real*8    AAijktnp1, AAim1jktnp1, AAip1jktnp1, AAijm1ktnp1, AAijp1ktnp1, AAijkm1tnp1,
       AAijkp1tnp1 !usefull temp variables holding concetrations on t=tnp1 time
     !AA values as well need boundary conditions since they represent AA=CcCcv
236   logical  BoundaryCell  ! It  will indicate  if this is a boundary cell or not .  True
       means this is actually a boundary cell
237   real*8 dboundaryFactor !boundary factor indicates how many times the concetration of
      the outter imaginary boundary cell,
238                       ! at the end off the x direction, is smaller than the
      concetration of the previous cell (which is in the aquifer limits)
239   real*8 uBoundaryFactor ! the same like dboundaryFactor but for the upstream boundary
      cells
240   logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out   whether boudary
       conditions the corresponding Concetrations
241         !Cvijktn, Cvim1jktn, Cvip1jktn, Cvijm1ktn, Cvijp1ktn, Cvijkm1tn, Cvijkp1tn
      will be calculated without any changes from the boundary conditions or not
242         ! Flag=True means no boundary effect is needed
243   common/vartrans4/dboundaryFactor,uboundaryFactor
244
245   !initializing variables
246   BoundaryCell=.False. ! in the begining we assume that the cell isnt a boundary one
247   flag3=.True. ! All Flags are true which means that for the calculations of the fcn no
       boundary conditions need to be enforced
248   flag4=.True.
249   flag5=.True.
250   flag6=.True.
251   flag7=.True.
252   flag8=.True.
253   ! End of initialization
254   if (i0==1 .or.i0==nx.or.j0==1 .or.j0==ny.or.w0==1 .or.w0==nz) then ! whenever a cell
      is at the edges it is supposed to be boundary
255     BoundaryCell=.True.
256   else
257     BoundaryCell=.False.
258     goto 14 ! the cell wasnt   boundary   and  so we exit the function
259   end if
260
261
262
263 if (i0==1 ) then ! we got upstream cell
264 flag3=.false. ! Cell position requires that Cvim1jktn=Cvijktn*uboundaryFacto
265 Cvim1jktn=Cvijktn*uboundaryFactor !Cvim1jktn=Cvijktn  would normally be but Cvim1jktn
      tends to zero ,since we suppose the that i-1,j,w cells tend to have zero
      concetration Cvim1jktn=Cvijktn*uboundaryFactor
266                                   ! upstream boundary condition tends to infinite
      aquifer ! For that we could set the factor uboundaryFactor=1
267                                   ! Cvip1jktn is imaginary cell when i0==1
268 AAim1jktn=AAijktn*uboundaryFactor ! The above can be applied as well for the AA values.
       Since AA values represent concetrations AA=CcCcv
269 AAim1jktnp1=AAijktnp1*uboundaryFactor
270 end if
271
272 if (i0==nx ) then  ! we got downstram cell
273  flag4=.false.  ! Cell position requires that Cvip1jktn=dBoundaryFactor*Cvijktn
274  Cvip1jktn=dBoundaryFactor*Cvijktn   !Cvip1jktn= Cvijktn  would normally be but
      Cvim1jktn tends to zero ,since we suppose the that i-1,j,w cells tend to have zero
      concetration Cvim1jktn=Cvijktn*uboundaryFactor
275                                   ! downstream boundary condition tends to infinite
      aquifer ! For that we could set the factor uboundaryFactor=1
276                                   !Cvip1jktn is imaginary cell when i0==nx
277 AAip1jktn=AAijktn*dBoundaryFactor ! The above can be applied as well for the AA values.
       Since AA values represent concetrations AA=CcCcv
278 AAip1jktnp1=AAijktnp1*dBoundaryFactor
279 end if
280
281 if (j0==1 ) then
282 flag5=.false. ! Cvijm1ktn is imaginary cell.  We can apply boundary condition dc/dy=0
      and so on
283 Cvijm1ktn=Cvijktn
284 AAijm1ktn=AAijktn
285 AAijm1ktnp1=AAijktnp1
286 end if
287
288 if (j0==ny ) then
```

```
289 flag6=.false. ! Cvijp1ktn is imaginary cell.  We can apply boundary condition dc/dy=0 ↙
        and so on
290 Cvijp1ktn=Cvijktn
291 AAijp1ktn=AAijktn
292 AAijp1ktnp1=AAijktnp1
293 end if
294
295 if (w0==1 ) then
296 flag7=.false. ! Cvijkm1tn is imaginary cell.  We can apply boundary condition dc/dz=0 ↙
        and so on
297 Cvijkm1tn=Cvijktn
298 AAijkm1tn=AAijktn
299 AAijkm1tnp1=AAijktnp1
300 end if
301
302 if (w0==nz ) then
303 flag8=.false. ! Cvijkp1tn is imaginary cell.  We can apply boundary condition dc/dz=0 ↙
        and so on
304 Cvijkp1tn=Cvijktn
305 AAijkp1tn=AAijktn
306 AAijkp1tnp1=AAijktnp1
307 end if
308
309  14 end subroutine
310
311
312  !End For the colloid-virus Cootransport, fCnC boundaries
```

```
1  module fcn_Vector_Cootrans
2  contains
3  subroutine  fCnC_vector( nX,nY,nZ,nx0,ny0,nz0,fCnC,Cvnp1,AAVector,BBVector,F_V,dt)
4  !fCnC_vector   takes in account the changes that were made from the source conditions
5  implicit none
6                          ! AxC=fCnC   .fCnC vector conctains a function of C (non sorbed ) concetrations and then Cs (sorbed) Concetrations
7                          ! fCnC_vector depends of Cvnp1 to produce fCnC!!!
8   integer  nX,nY,nZ  ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions
9   real*8 th,rD  !th=porosity (or th=th* the effective porosity can be used as well  page 509) ,rD= bulk density of the solid matrix(solid mass/aquifer volume)
10
11
12
13
14  real*8 , allocatable :: fCnC(:)       !  fCnC will be holding  the stable vector of equation   AxC=fCnCC   (fCnC is afunction of Cvn =known concetrations at time t=tn, for the virus case)
15  real*8 , allocatable :: Cvnp1(:)      ! Cvnp1 matrix will hold all the unknown concetrations (sorbed and non sorbed ) at time tn+1
16  real*8 , allocatable :: AAVector(:,:) ! Vector holding the solutions of the 8, 13 eqs . Where aa=CcCvc and BB=Cc*Cvc* ,
17                                        ! 2 dimensional  matrix,because we need the  AA values on time t=tn time as well as t=tnp1 time
18  real*8 , allocatable :: BBVector(:,:) ! Vector holding the solutions of the 8, 13 eqs . Where aa=CcCvc and BB=Cc*Cvc*
19                                        ! 2 dimensional  matrix,because we need the  BB values on time t=tn time as well as t=tnp1 time
20                                        ! Paper Colloid-Facilitated virus transport in saturated porus media . Vasiliki i.Syngouna Constantinos V.Chrysicopoulos
21             !  All the c values are  located at the  beginning of each row and then the Csorbed values follow
22           !C=c+cs   ! non sorbed and sorbed concetrations
23  real*8 wm ! wm determines the numerical mode that will be  used for the soliving of the tranport or cootranport problem. If wm=0.5 we have crank nikolson . If wm=1 we have full implicit scheme
24           ! wm should be within the range  0.5-1. Else we fall to explicit scheme the and stability is not ensured
25  real*8    Cvijktn, Cvaijktn, Cvim1jktn, Cvip1jktn, Cvijm1ktn, Cvijp1ktn, Cvijkm1tn, Cvijkp1tn !
26             !  Cvijktn, Cvaijktn, Cvim1jktn, Cvip1jktn, Cvijm1ktn, Cvijp1ktn, Cvijkm1tn, Cvijkp1tn = usefull temp variables holding concetrations of the present t=tn time
27  real*8    AAijktn,  AAim1jktn, AAip1jktn, AAijm1ktn, AAijp1ktn, AAijkm1tn, AAijkp1tn ! usefull temp variables holding concetrations of the present on t=tn
28  real*8    AAijktnp1, AAim1jktnp1, AAip1jktnp1, AAijm1ktnp1, AAijp1ktnp1, AAijkm1tnp1, AAijkp1tnp1 !usefull temp variables holding concetrations on t=tnp1 time
29  real*8    BBijktn,BBijktnp1 !usefull temp variables holding concetrations of the present t=tn and t=tnp1 time
30
31
32  integer rowPoss(8) ! useFull integers that will hold the position of the needed concetrations Cvijktn,Cvip1jktn ... in the Cvnp1 Vector
33
34
35 real*8 Dvx,Dvy,Dvz      !Dvx,Dvy  hydrodynamic dispersion coefficient of the virus component
36                         !Dvz=Vertical hydronamic dispersion coefficient of the virus component
37 real*8 Dvcx,Dvcy,Dvcz !Dcvx,Dcvy,Dcvz =hydrodynamic components (x,y,z direction) for the case when virus is attached to the colloid
38 real*8 Lvc,Lvca ! decay rates coefficients  when virus is attached to the colloid, and virus is attached to the colloid and all together sorbed on the solid matrix
39 real*8 Lv,Lva ! decay ratse coefficients when just a simple virus  is soluted in the water and  when a vius is sorbed on the solid matrix
40
41  real*8 F_V !F_V= general functional form of virus source configuration [ML-3 t-1]=[gr cm-3 Day-1]
42  integer nx0,ny0,nz0 ! nx0,ny0,nz0  cell coordinates of the point source
43  integer  rowPos,rowCsPos,row ! rowPos,rowCsPos  the distance from the first element for a single row (check the corresponding functions)
44                               ! "row" will hold the row at the current fCnC vector
```

```
45  integer ii,jj,ww ,ir ! iterators
46  real*8 dxx,dyy,dzz,dt ! dxx=the lenght of  each cell the total aquifer is splitted to.
47                        ! dyy=the witdth of  each cell the total aquifer is splitted to.
48                        ! dzz=the height of  each sell the total aquifer is splitted to.
49                        ! dt= the  (time) distance between two sequential time moments t1 ,t2  =dt=t2-t1
50  integer sourceSurf_V ! sourceSurf_V determines on which plane the elliptic source is based
51                  !(sourceSurf_V=0  we dont have an elliptic source but a point one, =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is based on XY plane)
52  real*8    U ! average  interstitial    velocity ! Dx,Dy,Dz,  ...defined in basic program hydrodynamic dispersion coefficients
53  real*8 dboundaryFactor,uboundaryFactor !dboundary factor indicates how many times the concetration of the outter imaginary boundary cell,
54                        ! at the end off the x direction, is smaller than the concetration of the previous cell (which is in the aquifer limits)
55                        !uboundaryFactor= is the same like dboundaryFactor ,but for the upstream boundary cells
56
57  real*8 Rvdva,Rvadv  ! forward or reverse rates cooefficients  (Virus case)!!!!!!!!!!!! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
58
59  logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out   whether boudary conditions the corresponding Concetrations
60        !Cvijktn, Cvim1jktn, Cvip1jktn, Cvijm1ktn, Cvijp1ktn, Cvijkm1tn, Cvijkp1tn will be calculated without any changes from the boundary conditions or not
61        ! Flag=True means no boundary effect is needed
62  common/VarTrans1a/u
63  common/VarTran2/rD,th
64  common/VarTrans3/dxx,dyy,dzz
65  common/vartrans4/dboundaryFactor,uboundaryFactor
66  common/vartrans5/wm
67  common/vartrans6/Lv,Lva,Rvdva,Rvadv
68  common/vartrans7/Dvx,Dvy,Dvz,Dvcx,Dvcy,Dvcz
69  Common/vartrans9/Lvc,Lvca
70  common/sourceGeom_V/sourceSurf_V
71
72
73
74  !initializing values
75  ii=1
76  jj=1
77  ww=1
78  ir=0
79  Cvijktn=0
80  Cvaijktn=0
81  Cvim1jktn=0
82  Cvip1jktn=0
83  Cvijm1ktn=0
84  Cvijp1ktn=0
85  Cvijkm1tn=0
86  Cvijkp1tn=0
87  rowPoss=0
88  ! end of initializations
89
90
91    do ii=1,nx  ! iterating through all the cells of the aquifer
92      do jj=1,ny
93         do ww=1,nz
94
95           ir=ir+1
96
97
98             !We calculate and store the corresponding Concetration on the present time t=tn based on their position in the Cvnp1 Vector
99             !Cvijktn and Cvaijktn are based on the i,j,k cell only which means that they are not affected by the boundary conditions at all
100
101            rowPoss(1)= rowPos(ii,jj,ww,ny,nz) ! Just for this case  RowPoss(1) = ir , but in order to be less errorProne we use a general sheme with the function rowPos
102            Cvijktn=Cvnp1(rowPoss(1))
103
104            AAijktn=AAVector(rowPoss(1),1)
```

```
105             AAijktnp1=AAVector(rowPoss(1),2)
106
107             BBijktn=BBVector(rowPoss(1),1)
108             BBijktnp1=BBVector(rowPoss(1),2)
109
110             rowPoss(2)= rowCsPos(ii,jj,ww,nx,ny,nz)
111             Cvaijktn=Cvnp1(rowPoss(2))
112
113
114          ! Checking the whether the cell for which we try to create the fCnC part is a boundary cell or not. If it is, it takes into aCvount  any extra boundary conditions that may exist
115             call isBoundaryCellfCnC (ii,jj,ww,nx,ny,nz,Cvijktn,Cvim1jktn,Cvip1jktn, Cvijm1ktn,Cvijp1ktn,Cvijkm1tn,Cvijkp1tn,AAijktn,AAim1jktn,AAip1jktn,AAijm1ktn, AAijp1ktn,AAijkm1tn,AAijkp1tn,AAijktnp1,AAim1jktnp1,AAip1jktnp1,AAijm1ktnp1, AAijp1ktnp1,AAijkm1tnp1,AAijkp1tnp1,flag3,flag4,flag5,flag6,flag7,flag8)
116
117
118          ! if flags return with value =False this means boundary conditions have already being applied and the corresponding concetrations are already calculated

119          ! If flags return True this means we will calculate them based on their position in the fCnC vector. AA values as well need boundary conditions since they represent AA=CcCcv
120
121          if (flag3==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCnC vector or on the AAVectors
122           rowPoss(3)=rowPos(ii-1,jj,ww,ny,nz)
123           Cvim1jktn=Cvnp1( rowPoss(3))
124           AAim1jktn=AAVector( rowPoss(3),1)
125           AAim1jktnp1=AAVector( rowPoss(3),2)
126          end if
127
128          if (flag4==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCnC vectorr or on the AAVectors
129           rowPoss(4)=rowPos(ii+1,jj,ww,ny,nz)
130           Cvip1jktn=Cvnp1(rowPoss(4))
131           AAip1jktn=AAVector(rowPoss(4),1)
132           AAip1jktnp1=AAVector(rowPoss(4),2)
133          end if
134
135          if (flag5==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCnC vector or on the AAVectors
136           rowPoss(5)=rowPos(ii,jj-1,ww,ny,nz)
137           Cvijm1ktn=Cvnp1(rowPoss(5))
138           AAijm1ktn=AAVector(rowPoss(5),1)
139           AAijm1ktnp1=AAVector(rowPoss(5),2)
140          end if
141
142          if (flag6==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCnC vector or on the AAVectors
143           rowPoss(6)=rowPos(ii,jj+1,ww,ny,nz)
144           Cvijp1ktn=Cvnp1(rowPoss(6))
145           AAijp1ktn=AAVector(rowPoss(6),1)
146           AAijp1ktnp1=AAVector(rowPoss(6),2)
147          end if
148
149          if (flag7==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCnC vector or on the AAVectors
150           rowPoss(7)=rowPos(ii,jj,ww-1,ny,nz)
151           Cvijkm1tn=Cvnp1(rowPoss(7))
152           AAijkm1tn=AAVector(rowPoss(7),1)
153           AAijkm1tnp1=AAVector(rowPoss(7),2)
154          end if
155
156          if (flag8==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCnC vector or on the AAVectors
157           rowPoss(8)= rowPos(ii,jj,ww+1,ny,nz)
```

```
158           Cvijkp1tn=Cvnp1(rowPoss(8))
159           AAijkp1tn=AAVector(rowPoss(8),1)
160           AAijkp1tnp1=AAVector(rowPoss(8),2)
161          end if
162
163
164
165
166
167
168         ! for the crank nikolson , simple transport should be used  ! directlty exctracted from mathematica **********************************
169           !fCnC(ir)= - (Cvaijktn*Lva*rd*(1.D0 - wm))/th + Cvijktn/dt + (Cvaijktn*rd)/(th*dt) + Cvip1jktn*((Dvx*(1.D0 - wm))/dxx**2.D0 - (U*(1.D0 - wm))/(2.D0*dxx)) +  &
170           !Cvim1jktn*((Dvx*(1.D0 - wm))/dxx**2.D0 + (U*(1.D0 - wm))/(2.D0*dxx)) + (Cvijm1ktn*Dvy*(1.D0 - wm))/dyy**2.D0 + (Cvijp1ktn*Dvy*(1.D0 - wm))/dyy**2.D0 + &
171           !Cvijktn*(-(Lv*(1.D0 - wm)) - (2.D0*Dvx*(1.D0 - wm))/dxx**2.D0 - (2.D0*Dvy*(1.D0 - wm))/dyy**2.D0 - (2.D0*Dvz*(1.D0 - wm))/dzz**2.D0) + (Cvijkm1tn*Dvz*(1.D0 - wm))/dzz**2.D0 +  &
172           !(Cvijkp1tn*Dvz*(1.D0 - wm))/dzz**2.D0
173         ! simple tranport crank nikolson ends here*************************************************************************
174
175         ! for the crank nikolson , cootransport should be used ! directlty exctracted from mathematica ***************************************
176           fCnC(ir)=     - (Cvaijktn*Lva*rd*(1.D0 - wm))/th - (BBijktn*Lvca*rd*(1.D0 - wm))/th - (BBijktnp1*Lvca*rd*wm)/th + (Cvaijktn*rd)/(th*dt) - &
177          (-AAijktn + AAijktnp1)/dt - (rd*(-BBijktn + BBijktnp1))/(th*dt) + AAip1jktn*((Dvcx*(1.D0 - wm))/dxx**2.D0 - (U*(1.D0 - wm))/(2.D0*dxx)) + &
178           Cvip1jktn*((Dvx*(1.D0 - wm))/dxx**2.D0 - (U*(1.D0 - wm))/(2.D0*dxx)) + AAim1jktn*((Dvcx*(1.D0 - wm))/dxx**2.D0 + (U*(1.D0 - wm))/(2.D0*dxx)) +  &
179           Cvim1jktn*((Dvx*(1.D0 - wm))/dxx**2.D0 + (U*(1.D0 - wm))/(2.D0*dxx)) + (AAim1jktnp1*Dvcx*wm)/dxx**2.D0 + (AAip1jktnp1*Dvcx*wm)/dxx**2.D0 - &
180          ((-AAim1jktnp1 + AAip1jktnp1)*U*wm)/(2.D0*dxx) + (AAijm1ktn*Dvcy*(1.D0 - wm))/dyy**2.D0 + (AAijp1ktn*Dvcy*(1.D0 - wm))/dyy**2.D0 +  &
181           (Cvijm1ktn*Dvy*(1.D0 - wm))/dyy**2.D0 + (Cvijp1ktn*Dvy*(1.D0 - wm))/dyy**2.D0 + (AAijm1ktnp1*Dvcy*wm)/dyy**2.D0 + (AAijp1ktnp1*Dvcy*wm)/dyy**2.D0 + &
182           AAijktn*(-(Lvc*(1.D0 - wm)) - (2.D0*Dvcx*(1.D0 - wm))/dxx**2.D0 - (2.D0*Dvcy*(1.D0 - wm))/dyy**2.D0 - (2.D0*Dvcz*(1.D0 - wm))/dzz**2.D0) +  &
183           Cvijktn*(-(Lv*(1.D0 - wm)) + 1.D0/dt - (2.D0*Dvx*(1.D0 - wm))/dxx**2.D0 - (2.D0*Dvy*(1.D0 - wm))/dyy**2.D0 - (2.D0*Dvz*(1.D0 - wm))/dzz**2.D0) + &
184           AAijktnp1*(-(Lvc*wm) - (2.D0*Dvcx*wm)/dxx**2.D0 - (2.D0*Dvcy*wm)/dyy**2.D0 - (2.D0*Dvcz*wm)/dzz**2.D0) + (AAijkm1tn*Dvcz*(1.D0 - wm))/dzz**2.D0 + &
185           (AAijkp1tn*Dvcz*(1.D0 - wm))/dzz**2.D0 + (Cvijkm1tn*Dvz*(1.D0 - wm))/dzz**2.D0 + (Cvijkp1tn*Dvz*(1.D0 - wm))/dzz**2.D0 + (AAijkm1tnp1*Dvcz*wm)/dzz**2.D0 + &
186           (AAijkp1tnp1*Dvcz*wm)/dzz**2.D0
187         ! cootransport crank nikolson ends here*************************************************************************
188
189
190
191        end do
192      end do
193    end do
194
195    !ir contains the row ,which the previous iterations finished with
196    do ii=1,nx  ! iterating through all the cells of the aquifer
197       do jj=1,ny
198          do ww=1,nz
199         ir=ir+1
200
201           Cvijktn=Cvnp1(rowPos(ii,jj,ww,ny,nz)) ! Calculating the neCvessary Concetrations using rowPos to find position, from Cvnp1 vector
202           Cvaijktn=Cvnp1(rowCsPos(ii,jj,ww,nx,ny,nz))! Calculating the neccessary Concetrations using rowPos to find position, from Cvnp1 vector
203           fCnC(ir)=-(Cvaijktn*(1.D0/dt - Lva*(1.D0 - wm) - Rvadv*(1.D0 - wm))) - (Cvijktn*Rvdva*th*(1.D0 - wm))/rd ! fCnC vector created based on  a general implicit method
    !wm=1=> full implicit, wm=0.5=> Crank Nikolson

205         end do
206      end do
207    end do
208
```

```
209
210
211  if (sourceSurf_V==0)then ! in case we have point source
212    ! because the way C are distributed the same way across the rows and the columns , Column=Cow
213    row= rowPos (nx0,ny0,nz0,ny,nz) ! rowPos give the column in the matrix
214    ! source conditions applied
215    ! we add the constant mass income at the source cell (located on the "row" row)
216    fCnC(row)=fCnC(row)+F_V
217  else ! in case we have elliptic  or ellipsoid source
218
219    call  sourceConf_CooTr(nX,nY,nZ,nx0,ny0,nz0,row,fCnC,F_V)
220  end if
221
222  end subroutine
223
224
225
226
227 subroutine sourceConf_CooTr(nX,nY,nZ,nx0,ny0,nz0,row,fCnC,F_V)
228 implicit none
229 ! sourceConf is the responsible subroutine to implement the source geometry ,in case we have an elliptic source
230 integer  nX,nY,nZ  ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions
231 integer nx0,ny0,nz0 ! nx0,ny0,nz0   coordinates of the  elliptic source center
232 integer  row ,rowPos ! rowPos  the distance from the first element for a single row !  "row" will hold the row at the current fCnC vector
233 integer rowSourcePos  ! converts with the help of sourceSurf_V,the 2 dimension plane into a 3 dimension space
234                       ! and returns the row of the particular elliptic source cell
235 real*8, allocatable :: fCnC(:) !  fCnC will be holding  the stable vector of equation AxC=fCnC   (fCnC is afunction of Cn =known concetrations at time t=tn)
236 real*8 F_V !F_V= general functional form of virus source configuration [ML-3 t-1]=[gr cm-3 Day-1]
237 real*8 dxx,dyy,dzz  ! dxx=the lenght of  each cell the total aquifer is splitted to.
238                      ! dyy=the witdth of  each cell the total aquifer is splitted to.
239                      ! dzz=the height of  each sell the total aquifer is splitted to.
240 integer sourceSurf_V ! sourceSurf_V determines on which plane the elliptic source is based
241                     !(sourceSurf_V=0  we dont have an elliptic source but a point one, =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is based on XY plane)
242                     !(sourceSurf_V=4 => ellipse in tree dimensions =ellipsoid)
243 real*8 aEl_V,bEl_V,cEl_V !aEl_V semi-axis of the elliptic source ! check the manual for more info
244                      !bEl_V semi-axis of the elliptic source !Check the manual for more info
245                      !cEl_V semi-axis of the elliptic source !Check the manual for more info
246 real*8 lHorMax,lVerMax,lHorMin,lVerMin ! usefull variables that indicate the limits of the square that surrounds the elliptic source
247 real*8 lzMax ,lzMin ! usefull variables that indicate the limits of the cube that surrounds the ellipsoid  source
248 integer nHor,nVer ! usefull temp variables to hold nx ny or nz depending on the current plane
249 integer nHorMax,nVerMax  ! usefull variables that indicate the limits (in cells) of the square that surrounds the elliptic source
250 integer nHorMin,nVerMin  ! usefull variables that indicate the limits (in cells)of the square that surrounds the elliptic source
251 integer nzMax, nzMin ! ! usefull variables that indicate the limits (in cells)of the cube that surrounds the ellipsoid source
252 real*8 lx0_V,ly0_V,lz0_V  !lx0_V,ly0_V,lz0_V =cartesian coordinates of the elliptic source center
253 real*8 lHor0 ,lVer0  ! usefull temp variables to hold lx0_V or ly0_V or lz0_V depending on the current plane
254 real*8   dHor,dVer ! usefull variables that describe the dimensions of a cell
255 real*8 lR1 ,lR2 ,lR3 ! random lenghts used as temp variables
256 integer sourCellNum ! the number of total cells the elliptic source was divided to
257 integer ir,irr ,irrr ! iterators
258
259 common/VarTrans3/dxx,dyy,dzz
260 common/sourceGeom_V/sourceSurf_V
261 common/sourceGeom2_V/aEl_V,bEl_V,cEl_V
```

```
262
263 common/sourceCoord_V/lx0_V,ly0_V,lz0_V
264 common/numOfsourcecells/sourCellNum
265
266 ! initializing values
267 lHorMax=0.D0
268 lVerMax=0.D0
269 lzMax=0.D0
270
271 lHorMin=0.D0
272 lVerMin=0.D0
273 lzMin=0.D0
274
275 nHorMax=0
276 nVerMax=0
277 nzMax=0
278
279 nHorMin=0
280 nVerMin=0
281 nzMin=0
282
283
284 dHor=0
285 dVer=0
286
287 lHor0=0
288 lVer0=0
289
290 ir=0
291 irr=0
292 irrr=0
293 lR1=0.D0
294 lR2=0.D0
295 lR3=0.D0
296 sourCellNum=0
297 !end if initialization
298
299
300 if (sourceSurf_V==1) then ! we are on yz plane
301   lHor0=ly0_V
302   lVer0=lz0_V
303
304   nHor=ny
305   nVer=nz
306
307   dHor=dyy
308   dver=dzz
309
310 else if(sourceSurf_V==2)then ! we are on xz plane
311   lHor0=lx0_V
312   lVer0=lz0_V
313
314   nHor=nx
315   nVer=nz
316
317   dHor=dxx
318   dver=dzz
319 else if(sourceSurf_V==3)then ! we are on xy plane
320   lHor0=lx0_V
321   lVer0=ly0_V
322
323   nHor=nx
324   nVer=ny
325
326   dHor=dxx
327   dver=dyy
328 else if(sourceSurf_V==4)then ! Ellipsoid = Ellipse in three dimensions
329   lHor0=lx0_V
330   lVer0=ly0_V
331
332   nHor=nx
333   nVer=ny
334
335   dHor=dxx
```

```
336   dver=dyy
337 goto 10 ! all the ellipoid code could be inserted here instead of the goto statement,
      but it would be more error prone
338
339 else
340   write(*,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
341   write(111,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
342   write(222,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
343   write(333,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
344   write(444,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
345   write(*,*)"The system will now stop"
346   read(*,*)! slows down system exit
347   pause
348   stop
349 end if
350
351
352
353 ! creating a surrounding  square outside of the elliptic source
354  ! we will then check which cells of the square are really inside the Ellipse
355
356  ! calculating max Limits
357  lHorMax=lHor0+aEl_V
358  lVerMax=lver0+bEl_V
359
360  nHorMax=int(lHorMax/dHor) +1 ! the integer part of the division is returned and the
       next closest cell is considered to be valid
361  nVerMax=int(lVerMax/dVer) +1
362
363  if(nHorMax>nHor) nHorMax=nHor ! the position of the source cant be outside of the
       aquifer
364  if(nVerMax>nVer) nVerMax=nVer ! the position of the source cant be outside of the
       aquifer
365  if(nHorMax<1) nHorMax=1 ! the position of the source cant be outside of the aquifer
366  if(nVerMax<1) nVerMax=1 ! the position of the source cant be outside of the aquifer
367
368  ! calculating min Limits
369   lHorMin=lHor0-aEl_V
370   lVerMin=lVer0-bEl_V
371
372   nHorMin=int(lHorMin/dHor) +1 ! the integer part of the division is returned and the
        next closest cell is considered to be valid
373   nVerMin=int(lVerMin/dVer) +1
374
375  if(nHorMin>nHor) nHorMin=nHor ! the position of the source cant be outside of the
       aquifer
376  if(nVerMin>nVer) nVerMin=nVer ! the position of the source cant be outside of the
       aquifer
377  if(nHorMin<1) nHorMin=1 ! the position of the source cant be outside of the aquifer
378  if(nVerMin<1) nVerMin=1 ! the position of the source cant be outside of the aquifer
379  ! square borders are now determined
380
381   do ir=nHorMin,nHorMax
382    do irr=nVerMin,nVerMax
383     ! cell is considered to be inside the source Ellipse ,if its cencter is inside
       (inside the Ellipse)
384     lR1=(ir-1)*dHor +dHor/2.D0 ! we locate the center of each valid cell
385     lR2=(irr-1)*dVer+dVer/2.D0 ! we locate the center of each valid cell
386
387
388       if(nVerMin==nVerMax .and.nHorMin==nHorMax) then ! at the Horizontal, and at the
       Vertical direction
389                                                      !only one cell is intersected by
       the elliptic source
390
391
392               row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_V) ! rowSourcePos
       give the row number in the matrix
393               fCnC(row)=fCnC(row)+F_V
```

```
394             sourCellNum=sourCellNum+1
395
396       else  if(nHorMin==nHorMax) then ! at the Horizontal  direction only one cell is intersected by the elliptic source
397
398             if (lR2>lVerMin .And.lR2<=lVerMax)then
399                 row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_V) ! rowSourcePos give the row number in the matrix
400                 fCnC(row)=fCnC(row)+F_V
401                 sourCellNum=sourCellNum+1
402             end if
403
404       else if(nVerMin==nVerMax) then ! at the Vertical direction only one cell is intersected by the elliptic source
405
406             if (lR1>lHorMin .And.lR1<=lHorMax) then
407                 row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_V) ! rowSourcePos give the row number in the matrix
408                 fCnC(row)=fCnC(row)+F_V
409                 sourCellNum=sourCellNum+1
410             end if
411
412
413       else if( ((lR1-lHor0)/aEl_V)**2.D0 +((lR2-lVer0)/bEl_V)**2.D0 <=1.D0  )then ! checking if the ellipse equation is being respected or not
414             ! it is respect we add the proper effects of the fCnC vector
415               row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_V) ! rowSourcePos give the row number in the matrix
416               fCnC(row)=fCnC(row)+F_V
417               sourCellNum=sourCellNum+1
418
419       end if
420
421    end do
422   end do
423
424
425  goto 9 ! protects it from reading the  ellipsoid code (elliptic source already used)
426
427
428
429   ! Ellipsoid source code starts Here
430  ! creating a surrounding cube outside of the elliptic source
431  ! we will then check which cells of the square are really inside the Ellipse
432
433  ! calculating max Limits
434  10 lHorMax=lHor0+aEl_V
435     lVerMax=lver0+bEl_V
436     lzMax=lz0_V+cEl_V
437
438  nHorMax=int(lHorMax/dHor) +1 ! the integer part of the division is returned and the next closest cell is considered to be valid
439  nVerMax=int(lVerMax/dVer) +1
440  nzMax=int(lzMax/dzz) +1
441
442  if(nHorMax>nHor) nHorMax=nHor ! the position of the source cant be outside of the aquifer
443  if(nVerMax>nVer) nVerMax=nVer ! the position of the source cant be outside of the aquifer
444  if(nzMax>nz) nzMax=nz ! the position of the source cant be outside of the aquifer
445
446  if(nHorMax<1) nHorMax=1 ! the position of the source cant be outside of the aquifer
447  if(nVerMax<1) nVerMax=1 ! the position of the source cant be outside of the aquifer
448  if(nzMax<1) nzMax=1 ! the position of the source cant be outside of the aquifer
449
450  ! calculating min Limits
451   lHorMin=lHor0-aEl_V
452   lVerMin=lVer0-bEl_V
453   lzMin=lz0_V-cEl_V
454
455   nHorMin=int(lHorMin/dHor) +1 ! the integer part of the division is returned and the next closest cell is considered to be valid
456   nVerMin=int(lVerMin/dVer) +1
457   nzMin=int(lzMin/dzz) +1
```

```
458
459  if(nHorMin>nHor) nHorMin=nHor ! the position of the source cant be outside of the aquifer
460  if(nVerMin>nVer) nVerMin=nVer ! the position of the source cant be outside of the aquifer
461  if(nzMin>nz) nzMin=nz ! the position of the source cant be outside of the aquifer
462
463  if(nHorMin<1) nHorMin=1 ! the position of the source cant be outside of the aquifer
464  if(nVerMin<1) nVerMin=1 ! the position of the source cant be outside of the aquifer
465  if(nzMin<1) nzMin=1 ! the position of the source cant be outside of the aquifer
466  ! cube borders are now determined
467
468   do ir=nHorMin,nHorMax
469    do irr=nVerMin,nVerMax
470     do irrr=nzMin,nzMax
471     ! cell is considered to be inside the source Ellipsoid ,if its center is inside (inside the ellipsoid)
472     lR1=(ir-1)*dHor  +dHor/2.D0 ! we locate the center of each valid cell
473     lR2=(irr-1)*dVer+dVer/2.D0 ! we locate the center of each valid cell
474     lR3=(irrr-1)*dzz+dzz/2.D0 ! we locate the center of each valid cell
475
476         if(nVerMin==nVerMax .and.nHorMin==nHorMax.and.nzMin==nzMax) then ! at the Horizontal, and at the  2 Vertical directions
477  ! only one cell is intersected by the elliptic source
478
479
480              row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
481              fCnC(row)=fCnC(row)+F_V
482              sourCellNum=sourCellNum+1
483
484       else if(nHorMin==nHorMax .and. nVerMin==nVerMax) then ! at the  xy plane only one cell is intersected by the ellipsoid source
485
486              if (lR3>lzMin .And.lR3<=lzMax ) then ! restrictions only on z direction
487                  row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
488                  fCnC(row)=fCnC(row)+F_V
489                  sourCellNum=sourCellNum+1
490              end if
491       else if(nHorMin==nHorMax .and. nzMin==nzMax) then ! at the  xz plane only one cell is intersected by the ellipsoid source
492
493              if ( lR2>lVerMin .And.lR2<=lVerMax) then ! restrictions only on  y direction
494                  row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
495                  fCnC(row)=fCnC(row)+F_V
496                  sourCellNum=sourCellNum+1
497              end if
498       else if(nVerMin==nVerMax .and. nzMin==nzMax) then ! at the  yz plane only one cell is intersected by the ellipsoid source
499
500              if (lR1>lHorMin .And.lR1<=lHorMax ) then ! restrictions only on  x direction
501                  row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
502                  fCnC(row)=fCnC(row)+F_V
503                  sourCellNum=sourCellNum+1
504              end if
505
506      else if(nHorMin==nHorMax) then ! at the x  direction only one cell is intersected by the elliptic source
507
508              if (lR2>lVerMin .And.lR2<=lVerMax .And.lR3>lzMin .And.lR3<=lzMax)then
509                  row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
510                  fCnC(row)=fCnC(row)+F_V
511                  sourCellNum=sourCellNum+1
512              end if
513
514       else if(nVerMin==nVerMax) then ! at the y direction only one cell is intersected by the elliptic source
515
```

```
516                 if (lR1>lHorMin .And.lR1<=lHorMax.And.lR3>lzMin .And.lR3<=lzMax) then
517                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
518                     fCnC(row)=fCnC(row)+F_V
519                     sourCellNum=sourCellNum+1
520                 end if
521           else if(nzMin==nzMax) then ! at the  z direction only one cell is intersected by the elliptic source
522
523                 if (lR1>lHorMin .And.lR1<=lHorMax.And.lR2>lVerMin .And.lR2<=lVerMax) then
524                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
525                     fCnC(row)=fCnC(row)+F_V
526                     sourCellNum=sourCellNum+1
527                 end if
528           else if( ((lR1-lHor0)/aEl_V)**2.D0 +((lR2-lVer0)/bEl_V)**2.D0  +((lR3-lz0_V)/cEl_V)**2.D0 <=1  )then
529                               ! checking if the ellipsoid equation is being respected or not
530                               ! If it is respected, we add the proper effects of the fCnC vector
531                 row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
532                 fCnC(row)=fCnC(row)+F_V
533                 sourCellNum=sourCellNum+1
534           end if
535         end do
536        end do
537       end do
538
539
540
541     9 if (sourCellNum==0) then ! the elliptic or ellipsoid source failed to intesect the center of any cell ,and so we will use a single point source
542         !because the C are distributed the same way across the rows and the columns ,Column=Row
543         row= rowPos (nx0,ny0,nz0,ny,nz) ! rowPos give the column in the matrix
544         ! source conditions applied
545         ! we add the constant mass income at the source cell (located on the "row" row)
546         fCnC(row)=fCnC(row)+F_V
547         sourCellNum=1
548       end if
549
550
551 end subroutine
552
553
554
555
556
557 end module
```

```
1
2  module fcn_Vector_Simple_Trans
3  contains
4
5  subroutine  fCn_vector( nX,nY,nZ,nx0,ny0,nz0,fcn,Ccnp1,F_C,dt)
6  !fCn_vector   takes in account the changes that were made from the source conditions
7  implicit none
8                          ! AxC=fCn   .fCn vector conctains a function of C (non sorbed ) concetrations and then Cs (sorbed) Concetrations
9                          ! fCn_vector depends of Ccnp1 to produce fCn!!!
10 integer  nX,nY,nZ  ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions
11 real*8 th,rD  !th=porosity (or th=th* the effective porosity can be used as well  page 509) ,rD= bulk density of the solid matrix(solid mass/aquifer volume)

12 real*8  ,allocatable :: fCn(:) !  fCn will be holding  the stable vector of equation AxC=fCn   (fCn is afunction of Cn =known concetrations at time t=tn)
13 real*8 Ccnp1(2*nx*ny*nz)  ! Ccnp1 matrix will hold all the unknown concetrations (sorbed and non sorbed ) at time tn+1
14           !  All the c values are  located at the  beginning of each row and then the Csorbed values follow
15           !C=c+cs   ! non sorbed and sorbed concetrations
16 real*8 wm ! wm determines the numerical mode that will be  used for the soliving of the tranport or cootranport problem. If wm=0.5 we have crank nikolson . If wm=1 we have full implicit scheme
17          ! wm should be within the range  0.5-1. Else we fall to explicit scheme the and stability is not ensured
18 real*8    Ccijktn, Ccaijktn, Ccim1jktn, Ccip1jktn, Ccijm1ktn, Ccijp1ktn, Ccijkm1tn, Ccijkp1tn !
19           ! Ccijktn, Ccaijktn, Ccim1jktn, Ccip1jktn, Ccijm1ktn, Ccijp1ktn, Ccijkm1tn, Ccijkp1tn= usefull temp variables holding concetrations of the present t=tn time
20 real*8 F_C !F_C= general functional form of Colloid source configuration [ML-3 t-1]= [gr cm-3 Day-1]
21 integer nx0,ny0,nz0 ! nx0,ny0,nz0  cell coordinates of the point source
22 integer  rowPos,rowCsPos,row ! rowPos,rowCsPos  the distance from the first element for a single row (check the corresponding functions)
23                            ! "row" will hold the row at the current fCn vector

24 integer ii,jj,ww ,ir ! iterators
25 real*8 dxx,dyy,dzz,dt ! dxx=the lenght of  each cell the total aquifer is splitted to.
26                       ! dyy=the witdth of  each cell the total aquifer is splitted to.
27                       ! dzz=the height of  each sell the total aquifer is splitted to.
28                       ! dt= the  (time) distance between two sequential time moments t1 ,t2  =dt=t2-t1
29 integer sourceSurf_C ! sourceSurf_C determines on which plane the elliptic source is based, Colloid case
30                      !(sourceSurf_C=0  we dont have an elliptic source but a point one, =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is based on XY plane)
31
32 real*8 Dcx,Dcy,Dcz,U ! average interstitial    velocity ! Dcx,Dcy,Dcz,  ...defined in basic program hydrodynamic dispersion coefficients
33 real*8 Rcadc,Lca  !Rcadc=r2= reverse rate coefficient    ,Lca=l2=l*=decay rate of sorbed solute
34 real*8 Rcdca,Lc   !Rcdca=r1=forward rate coefficient ,=Lc=l1=decay rate of liquid phase solute
35 real*8 dboundaryFactor,uboundaryFactor !dboundary factor indicates how many times the concetration of the outter imaginary boundary cell,
36                       ! at the end off the x direction, is smaller than the concetration of the previous cell (which is in the aquifer limits)
37                       !uboundaryFactor= is the same like dboundaryFactor ,but for the upstream boundary cells
38
39
40 logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out   whether boudary conditions the corresponding Concetrations
41          !Ccijktn, Ccim1jktn, Ccip1jktn, Ccijm1ktn, Ccijp1ktn, Ccijkm1tn, Ccijkp1tn will be calculated without any changes from the boundary conditions or not
42          ! Flag=True means no boundary effect is needed
43
44
45
46
47 common/VarTran2/rD,th
```

```
 48  common/VarTrans3/dxx,dyy,dzz
 49  Common/vartrans4/dboundaryFactor,uboundaryFactor
 50  common/vartrans5/wm
 51  common/VarTrans1/Dcx,Dcy,Dcz,Lc,Lca,Rcdca,Rcadc
 52  common/VarTrans1a/u
 53  common/sourceGeom_C/sourceSurf_C
 54  !initializing values
 55  ii=1
 56  jj=1
 57  ww=1
 58  ir=0
 59  Ccijktn=0
 60  Ccaijktn=0
 61  Ccim1jktn=0
 62  Ccip1jktn=0
 63  Ccijm1ktn=0
 64  Ccijp1ktn=0
 65  Ccijkm1tn=0
 66  Ccijkp1tn=0
 67  ! end of initializations
 68
 69
 70    do ii=1,nx  ! iterating through all the cells of the aquifer
 71      do jj=1,ny
 72         do ww=1,nz
 73
 74           ir=ir+1
 75
 76
 77             !We calculate and store the corresponding Concetration on the present time t=tn based on their position in the Ccnp1 Vector
 78             !Ccijktn and Ccaijktn are based on the i,j,k cell only which means that they are not affected by the boundary conditions at all
 79
 80
 81             Ccijktn=Ccnp1(rowPos(ii,jj,ww,ny,nz))
 82             Ccaijktn=Ccnp1(rowCsPos(ii,jj,ww,nx,ny,nz))
 83
 84
 85          ! Checking the whether the cell for which we try to create the fCn part is a boundary cell or not. If it is, it takes into account  any extra boundary conditions that may exist
 86            call isBoundaryCellfCn (ii,jj,ww,nx,ny,nz,Ccijktn,Ccim1jktn,Ccip1jktn,Ccijm1ktn,Ccijp1ktn,Ccijkm1tn,Ccijkp1tn,flag3,flag4,flag5,flag6,flag7,flag8)
 87          ! if flags return with value =False this means boundary conditions have already being applied and the corresponding concetrations are already calculated
 88          ! If flags return True this means we will calculate them based on their position in the fCn vector
 89
 90          if (flag3==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCn vector
 91
 92           Ccim1jktn=Ccnp1(rowPos(ii-1,jj,ww,ny,nz))
 93          end if
 94
 95          if (flag4==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCn vector
 96           Ccip1jktn=Ccnp1(rowPos(ii+1,jj,ww,ny,nz))
 97          end if
 98
 99          if (flag5==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCn vector
100           Ccijm1ktn=Ccnp1(rowPos(ii,jj-1,ww,ny,nz))
101          end if
102
103          if (flag6==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCn vector
104           Ccijp1ktn=Ccnp1(rowPos(ii,jj+1,ww,ny,nz))
105          end if
106          if (flag7==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCn vector
107           Ccijkm1tn=Ccnp1(rowPos(ii,jj,ww-1,ny,nz))
108          end if
```

```
109
110         if (flag8==.True. ) then ! True means no boundary need to be applied and so we calculate the needed concetration based her position on the fCn vector
111          Ccijkp1tn=Ccnp1(rowPos(ii,jj,ww+1,ny,nz))
112         end if
113
114
115
116
117
118
119
120        ! for the crank nikolson , simple transport should be used  ! directlty exctracted from mathematica **********************************
121
122
123           fCn(ir)= - (Ccaijktn*Lca*rd*(1.D0 - wm))/th + Ccijktn/dt + (Ccaijktn*rd)/(th*dt) + Ccip1jktn*((Dcx*(1.D0 - wm))/dxx**2.D0 - (U*(1.D0 - wm))/(2.D0*dxx)) +  &
124           Ccim1jktn*((Dcx*(1.D0 - wm))/dxx**2.D0 + (U*(1.D0 - wm))/(2.D0*dxx)) + (Ccijm1ktn*Dcy*(1.D0 - wm))/dyy**2.D0 + (Ccijp1ktn*Dcy*(1.D0 - wm))/dyy**2.D0 + &
125           Ccijktn*(-(Lc*(1.D0 - wm)) - (2.D0*Dcx*(1.D0 - wm))/dxx**2.D0 - (2.D0*Dcy*(1.D0 - wm))/dyy**2.D0 - (2.D0*Dcz*(1.D0 - wm))/dzz**2.D0) + (Ccijkm1tn*Dcz*(1.D0 - wm))/dzz**2.D0 +  &
126           (Ccijkp1tn*Dcz*(1.D0 - wm))/dzz**2.D0
127
128        ! simple tranport crank nikolson ends here************************************ ****************************************************
129
130
131
132
133
134       end do
135     end do
136   end do
137
138   !ir contains the row ,which the previous iterations finished with
139   do ii=1,nx  ! iterating through all the cells of the aquifer
140       do jj=1,ny
141          do ww=1,nz
142         ir=ir+1
143
144              Ccijktn=Ccnp1(rowPos(ii,jj,ww,ny,nz)) ! Calculating the neccessary Concetrations using rowPos to find position, from Ccnp1 vector
145              Ccaijktn=Ccnp1(rowCsPos(ii,jj,ww,nx,ny,nz))! Calculating the neccessary Concetrations using rowPos to find position, from Ccnp1 vector
146              fCn(ir)=-(Ccaijktn*(1.D0/dt - Lca*(1.D0 - wm) - Rcadc*(1.D0 - wm))) - (Ccijktn*Rcdca*th*(1.D0 - wm))/rd ! fcn vector created based on  a general implicit method
147   !wm=1=> full implicit, wm=0.5=> Crank Nikolson
148         end do
149     end do
150   end do
151
152
153
154   if (sourceSurf_C==0)then ! in case we have point source
155     ! because the way C are distributed the same way across the rows and the columns , Column=Cow
156     row= rowPos (nx0,ny0,nz0,ny,nz) ! rowPos give the column in the matrix
157     ! source conditions applied
158     ! we add the constant mass income at the source cell (located on the "row" row)
159     fCn(row)=fCn(row)+F_C
160   else ! in case we have elliptic  or ellipsoid source
161
162     call  sourceConf(nX,nY,nZ,nx0,ny0,nz0,row,fCn,F_C)
163   end if
164
165   end subroutine
166
167
168
169
```

```
170
171 subroutine sourceConf(nX,nY,nZ,nx0,ny0,nz0,row,fCn,F_C)
172 implicit none
173 ! sourceConf is the responsible subroutine to implement the source geometry ,in case we have an elliptic source
174 integer  nX,nY,nZ  ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions
175 integer nx0,ny0,nz0 ! nx0,ny0,nz0   coordinates of the  elliptic source center
176 integer  row ,rowPos ! rowPos  the distance from the first element for a single row !  "row" will hold the row at the current fCn vector
177 integer rowSourcePos  ! converts with the help of sourceSurf,the 2 dimension plane into a 3 dimension space
178                       ! and returns the row of the particular elliptic source cell
179 real*8, allocatable :: fCn(:) !  fCn will be holding  the stable vector of equation AxC=fCn   (fCn is afunction of Cn =known concetrations at time t=tn)
180 real*8 F_C !F_C= general functional form of colloid source configuration [ML-3 t-1]=[gr cm-3 Day-1]
181 real*8 dxx,dyy,dzz  ! dxx=the lenght of  each cell the total aquifer is splitted to.
182                       ! dyy=the witdth of  each cell the total aquifer is splitted to.
183                       ! dzz=the height of  each sell the total aquifer is splitted to.
184 integer sourceSurf_C ! sourceSurf_C determines on which plane the elliptic source is based,Colloid case
185                     !(sourceSurf_C=0  we dont have an elliptic source but a point one, =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is based on XY plane)
186                     !(sourceSurf_C=4 => ellipse in tree dimensions =ellipsoid)
187 real*8 aEl_C,bEl_C,cEl_C  !aEl_C semi-axis of the elliptic source ! check the manual for more info. Colloid case
188                       !bEl semi-axis of the elliptic source !Check the manual for more info.Colloid case
189                       !cEl semi-axis of the elliptic source !Check the manual for more info.
190 real*8 lHorMax,lVerMax,lHorMin,lVerMin ! usefull variables that indicate the limits of the square that surrounds the elliptic source
191 real*8 lzMax ,lzMin ! usefull variables that indicate the limits of the cube that surrounds the ellipsoid  source
192 integer nHor,nVer ! usefull temp variables to hold nx ny or nz depending on the current plane
193 integer nHorMax,nVerMax  ! usefull variables that indicate the limits (in cells) of the square that surrounds the elliptic source
194 integer nHorMin,nVerMin  ! usefull variables that indicate the limits (in cells)of the square that surrounds the elliptic source
195 integer nzMax, nzMin ! ! usefull variables that indicate the limits (in cells)of the cube that surrounds the ellipsoid source
196 real*8 lx0_C,ly0_C,lz0_C  !lx0_C,ly0_C,lz0_C =cartesian coordinates of the elliptic source center,Colloid case
197 real*8 lHor0 ,lVer0  ! usefull temp variables to hold lx0 or ly0 or lz0  depending on the current plane
198 real*8   dHor,dVer ! usefull variables that describe the dimensions of a cell
199 real*8 lR1 ,lR2 ,lR3 ! random lenghts used as temp variables
200 integer sourCellNum ! the number of total cells the elliptic source was divided to
201 integer ir,irr ,irrr ! iterators
202 common/VarTrans3/dxx,dyy,dzz
203
204 common/sourceGeom_C/sourceSurf_C
205 common/sourceGeom2_C/aEl_C,bEl_C,cEl_C
206
207 common/sourceCoord_C/lx0_C,ly0_C,lz0_C
208 common/numOfsourcecells/sourCellNum
209
210 ! initializing values
211 lHorMax=0.D0
212 lVerMax=0.D0
213 lzMax=0.D0
214
215 lHorMin=0.D0
216 lVerMin=0.D0
217 lzMin=0.D0
218
219 nHorMax=0
220 nVerMax=0
221 nzMax=0
222
223 nHorMin=0
```

```
224 nVerMin=0
225 nzMin=0
226
227
228 dHor=0
229 dVer=0
230
231 lHor0=0
232 lVer0=0
233
234 ir=0
235 irr=0
236 irrr=0
237 lR1=0.D0
238 lR2=0.D0
239 lR3=0.D0
240 sourCellNum=0
241 !end if initialization
242
243
244 if (sourceSurf_C==1) then ! we are on yz plane
245   lHor0=ly0_C
246   lVer0=lz0_C
247
248   nHor=ny
249   nVer=nz
250
251   dHor=dyy
252   dver=dzz
253
254 else if(sourceSurf_C==2)then ! we are on xz plane
255   lHor0=lx0_C
256   lVer0=lz0_C
257
258   nHor=nx
259   nVer=nz
260
261   dHor=dxx
262   dver=dzz
263 else if(sourceSurf_C==3)then ! we are on xy plane
264   lHor0=lx0_C
265   lVer0=ly0_C
266
267   nHor=nx
268   nVer=ny
269
270   dHor=dxx
271   dver=dyy
272 else if(sourceSurf_C==4)then ! Ellipsoid = Ellipse in three dimensions
273   lHor0=lx0_C
274   lVer0=ly0_C
275
276   nHor=nx
277   nVer=ny
278
279   dHor=dxx
280   dver=dyy
281 goto 10 ! all the ellipoid code could be inserted here instead of the goto statement,
      but it would be more error prone
282
283 else
284   write(*,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
285   write(111,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
286   write(222,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
287   write(333,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
288   write(444,*)"Unknown internal error occured when trying to determine the plane of the
        epiptic source"
289   write(*,*)"The system will now stop"
290   read(*,*)! slows down system exit
291   stop
```

```
292 end if
293
294
295
296 ! creating a surrounding  square outside of the elliptic source
297  ! we will then check which cells of the square are really inside the Ellipse
298
299  ! calculating max Limits
300  lHorMax=lHor0+aEl_C
301  lVerMax=lver0+bEl_C
302
303  nHorMax=int(lHorMax/dHor) +1 ! the integer part of the division is returned and the
        next closest cell is considered to be valid
304  nVerMax=int(lVerMax/dVer) +1
305
306  if(nHorMax>nHor) nHorMax=nHor ! the position of the source cant be outside of the
        aquifer
307  if(nVerMax>nVer) nVerMax=nVer ! the position of the source cant be outside of the
        aquifer
308  if(nHorMax<1) nHorMax=1 ! the position of the source cant be outside of the aquifer
309  if(nVerMax<1) nVerMax=1 ! the position of the source cant be outside of the aquifer
310
311  ! calculating min Limits
312   lHorMin=lHor0-aEl_C
313   lVerMin=lVer0-bEl_C
314
315   nHorMin=int(lHorMin/dHor) +1 ! the integer part of the division is returned and the
        next closest cell is considered to be valid
316   nVerMin=int(lVerMin/dVer) +1
317
318  if(nHorMin>nHor) nHorMin=nHor ! the position of the source cant be outside of the
        aquifer
319  if(nVerMin>nVer) nVerMin=nVer ! the position of the source cant be outside of the
        aquifer
320  if(nHorMin<1) nHorMin=1 ! the position of the source cant be outside of the aquifer
321  if(nVerMin<1) nVerMin=1 ! the position of the source cant be outside of the aquifer
322  ! square borders are now determined
323
324   do ir=nHorMin,nHorMax
325    do irr=nVerMin,nVerMax
326     ! cell is considered to be inside the source Ellipse ,if its cencter is inside
        (inside the Ellipse)
327     lR1=(ir-1)*dHor +dHor/2.D0 ! we locate the center of each valid cell
328     lR2=(irr-1)*dVer+dVer/2.D0 ! we locate the center of each valid cell
329
330
331       if(nVerMin==nVerMax .and.nHorMin==nHorMax) then ! at the Horizontal, and at the
        Vertical direction
332                                                       !only one cell is intersected by
        the elliptic source
333
334
335              row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_C) ! rowSourcePos
        give the row number in the matrix
336              fCn(row)=fCn(row)+F_C
337              sourCellNum=sourCellNum+1
338
339       else  if(nHorMin==nHorMax) then ! at the Horizontal  direction only one cell is
        intersected by the elliptic source
340
341              if (lR2>lVerMin .And.lR2<=lVerMax)then
342                  row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_C) !
        rowSourcePos give the row number in the matrix
343                  fCn(row)=fCn(row)+F_C
344                  sourCellNum=sourCellNum+1
345              end if
346
347       else if(nVerMin==nVerMax) then ! at the Vertical direction only one cell is
        intersected by the elliptic source
348
349              if (lR1>lHorMin .And.lR1<=lHorMax) then
350                  row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_C) !
        rowSourcePos give the row number in the matrix
351                  fCn(row)=fCn(row)+F_C
```

```
352                 sourCellNum=sourCellNum+1
353               end if
354
355
356       else if( ((lR1-lHor0)/aEl_C)**2.D0 +((lR2-lVer0)/bEl_C)**2.D0 <=1.D0  )then ! checking if the ellipse equation is being respected or not
357             ! it is respect we add the proper effects of the fCn vector
358               row= rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf_C) ! rowSourcePos give the row number in the matrix
359               fCn(row)=fCn(row)+F_C
360               sourCellNum=sourCellNum+1
361
362       end if
363
364     end do
365    end do
366
367
368   goto 9 ! protects it from reading the  ellipsoid code (elliptic source already used)
369
370
371
372    ! Ellipsoid source code starts Here
373   ! creating a surrounding cube outside of the elliptic source
374   ! we will then check which cells of the square are really inside the Ellipse
375
376   ! calculating max Limits
377   10 lHorMax=lHor0+aEl_C
378      lVerMax=lver0+bEl_C
379      lzMax=lz0_C+cEl_C
380
381   nHorMax=int(lHorMax/dHor) +1 ! the integer part of the division is returned and the next closest cell is considered to be valid
382   nVerMax=int(lVerMax/dVer) +1
383   nzMax=int(lzMax/dzz) +1
384
385   if(nHorMax>nHor) nHorMax=nHor ! the position of the source cant be outside of the aquifer
386   if(nVerMax>nVer) nVerMax=nVer ! the position of the source cant be outside of the aquifer
387   if(nzMax>nz) nzMax=nz ! the position of the source cant be outside of the aquifer
388
389   if(nHorMax<1) nHorMax=1 ! the position of the source cant be outside of the aquifer
390   if(nVerMax<1) nVerMax=1 ! the position of the source cant be outside of the aquifer
391   if(nzMax<1) nzMax=1 ! the position of the source cant be outside of the aquifer
392
393   ! calculating min Limits
394    lHorMin=lHor0-aEl_C
395    lVerMin=lVer0-bEl_C
396    lzMin=lz0_C-cEl_C
397
398    nHorMin=int(lHorMin/dHor) +1 ! the integer part of the division is returned and the next closest cell is considered to be valid
399    nVerMin=int(lVerMin/dVer) +1
400    nzMin=int(lzMin/dzz) +1
401
402   if(nHorMin>nHor) nHorMin=nHor ! the position of the source cant be outside of the aquifer
403   if(nVerMin>nVer) nVerMin=nVer ! the position of the source cant be outside of the aquifer
404   if(nzMin>nz) nzMin=nz ! the position of the source cant be outside of the aquifer
405
406   if(nHorMin<1) nHorMin=1 ! the position of the source cant be outside of the aquifer
407   if(nVerMin<1) nVerMin=1 ! the position of the source cant be outside of the aquifer
408   if(nzMin<1) nzMin=1 ! the position of the source cant be outside of the aquifer
409   ! cube borders are now determined
410
411    do ir=nHorMin,nHorMax
412     do irr=nVerMin,nVerMax
413      do irrr=nzMin,nzMax
414      ! cell is considered to be inside the source Ellipsoid ,if its center is inside (inside the ellipsoid)
415      lR1=(ir-1)*dHor  +dHor/2.D0 ! we locate the center of each valid cell
416      lR2=(irr-1)*dVer+dVer/2.D0 ! we locate the center of each valid cell
```

```
417     lR3=(irrr-1)*dzz+dzz/2.D0 ! we locate the center of each valid cell
418
419         if(nVerMin==nVerMax .and.nHorMin==nHorMax.and.nzMin==nzMax) then ! at the Horizontal, and at the  2 Vertical directions
420  ! only one cell is intersected by the elliptic source
421
422
423                 row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
424                 fCn(row)=fCn(row)+F_C
425                 sourCellNum=sourCellNum+1
426
427       else if(nHorMin==nHorMax .and. nVerMin==nVerMax) then ! at the  xy plane only one cell is intersected by the ellipsoid source
428
429                 if (lR3>lzMin .And.lR3<=lzMax ) then ! restrictions only on z direction
430                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
431                     fCn(row)=fCn(row)+F_C
432                     sourCellNum=sourCellNum+1
433                 end if
434       else if(nHorMin==nHorMax .and. nzMin==nzMax) then ! at the  xz plane only one cell is intersected by the ellipsoid source
435
436                 if ( lR2>lVerMin .And.lR2<=lVerMax) then ! restrictions only on  y direction
437                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
438                     fCn(row)=fCn(row)+F_C
439                     sourCellNum=sourCellNum+1
440                 end if
441       else if(nVerMin==nVerMax .and. nzMin==nzMax) then ! at the  yz plane only one cell is intersected by the ellipsoid source
442
443                 if (lR1>lHorMin .And.lR1<=lHorMax ) then ! restrictions only on  x direction
444                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
445                     fCn(row)=fCn(row)+F_C
446                     sourCellNum=sourCellNum+1
447                 end if
448
449      else if(nHorMin==nHorMax) then ! at the x  direction only one cell is intersected by the elliptic source
450
451                 if (lR2>lVerMin .And.lR2<=lVerMax .And.lR3>lzMin .And.lR3<=lzMax)then
452                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
453                     fCn(row)=fCn(row)+F_C
454                     sourCellNum=sourCellNum+1
455                 end if
456
457       else if(nVerMin==nVerMax) then ! at the y direction only one cell is intersected by the elliptic source
458
459                 if (lR1>lHorMin .And.lR1<=lHorMax.And.lR3>lzMin .And.lR3<=lzMax) then
460                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
461                     fCn(row)=fCn(row)+F_C
462                     sourCellNum=sourCellNum+1
463                 end if
464       else if(nzMin==nzMax) then ! at the  z direction only one cell is intersected by the elliptic source
465
466                 if (lR1>lHorMin .And.lR1<=lHorMax.And.lR2>lVerMin .And.lR2<=lVerMax) then
467                     row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
468                     fCn(row)=fCn(row)+F_C
469                     sourCellNum=sourCellNum+1
470                 end if
471       else if( ((lR1-lHor0)/aEl_C)**2.D0 +((lR2-lVer0)/bEl_C)**2.D0  +((lR3-lz0_C)/cEl_C)**2.D0 <=1  )then
472                             ! checking if the ellipsoid equation is being respected or not
```

```
473                             ! If it is respected, we add the proper effects of the fCn vector
474                 row= rowPos (ir,irr,irrr,ny,nz)  ! rowPos give the row number in the matrix
475                 fCn(row)=fCn(row)+F_C
476                 sourCellNum=sourCellNum+1
477               end if
478             end do
479            end do
480           end do
481
482
483
484 9 if (sourCellNum==0) then ! the elliptic or ellipsoid source failed to intesect the center of any cell ,and so we will use a single point source
485             !because the C are distributed the same way across the rows and the columns ,Column=Row
486             row= rowPos (nx0,ny0,nz0,ny,nz) ! rowPos give the column in the matrix
487             ! source conditions applied
488             ! we add the constant mass income at the source cell (located on the "row" row)
489             fCn(row)=fCn(row)+F_C
490             sourCellNum=1
491           end if
492
493
494 end subroutine
495
496
497
498
499
500 end module
501
502
503
504
505
506
507
508
509
510 function rowSourcePos(ir,irr,ny,nz,nx0,ny0,nz0,sourceSurf) ! converts with the help of sourceSurf,the 2 dimension plane into a 3 dimension space
511 implicit none                                               ! and returns the row of the particular elliptic source cell
512 ! calculating the distance from the first element for a single row (positioning the c elements)
513 ! supposed that each row is written by the formula write(*,*) "C" (((i,j,w , w=1,nz),j=1,ny),i=1,nx) for the c and
514 ! then the same for Csorbed write(*,*) "Cs" (((i,j,w , w=1,nz),j=1,ny),i=1,nx)
515 ! we would get  111 -112 -113 -114 -115 -121 -122 -123 -124 -125 -131-132 ....
516
517 integer rowSourcePos ! like function rowPos but for the elliptic source cells
518 integer ir, irr ! the number of pieces the distance from the (0.0.0) each  source cell is divided to
519 integer ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at x,y,z directions.
520 integer nx0,ny0,nz0 ! nx0,ny0,nz0  cell coordinates of the elliptic source center
521 integer i,j,w !The coordinates  i,j,w of  the cell each A coefficient is multipliying
522 integer sourceSurf ! sourceSurf determines on which plane the elliptic source is based
523  !(sourceSurf=0  we dont have an elliptic source but a point one, =1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is based on XY plane)
524 integer rowPos ! rowPos  the distance from the first element for a single row
525 i=0
526 j=0
527 w=0
528 rowSourcePos=0
529
530 if (sourceSurf==1)then ! we are on yz plane
531  i=nx0
532  j=ir
533  w=irr
534  rowSourcePos=rowPos(i,j,w,ny,nz)
535
```

```
536 else if (sourceSurf==2)then ! we are on xz plane
537  i=ir
538  j=ny0
539  w=irr
540  rowSourcePos=rowPos(i,j,w,ny,nz)
541 else if (sourceSurf==3)then ! we are on xy plane
542  i=ir
543  j=irr
544  w=nz0
545  rowSourcePos=rowPos(i,j,w,ny,nz)
546 else
547   write(*,*)"Unknown internal error 2 occured when trying to determin the plane of the epiptic source"
548   write(111,*)"Unknown internal error 2  occured when trying to determin the plane of the epiptic source"
549   write(222,*)"Unknown internal error 2  occured when trying to determin the plane of the epiptic source"
550   write(333,*)"Unknown internal error 2  occured when trying to determin the plane of the epiptic source"
551   write(444,*)"Unknown internal error 2  occured when trying to determin the plane of the epiptic source"
552   write(*,*)"The system will now stop"
553   read(*,*)! slows down system exit
554   stop
555 end if
556
557 end function
558
559
560
561
562 function rowPos(i,j,w,ny,nz)
563 implicit none
564
565 ! calculating the distance from the first element for a single row (positioning the c elements)
566 ! supposed that each row is written by the formula write(*,*) "C" (((i,j,w , w=1,nz),j=1,ny),i=1,nx)
567 !for the c and then the same for Csorbed write(*,*) "Cs" (((i,j,w , w=1,nz),j=1,ny),i=1,nx)
568 ! we would get  111 -112 -113 -114 -115 -121 -122 -123 -124 -125 -131-132 ....
569
570 integer i,j,w !The coordinates  i,j,w of  the cell each A coefficient is multipliying
571 integer rowPos ! rowPos  the distance from the first element for a single row
572 integer ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at x,y,z directions.
573
574     rowPos= (i-1)*ny*nz+(j-1)*nz+w
575
576
577  ! (i-1)*ny*nz+(j-1)*nz+w = is like writting nested do loops
578  !  do ii=1,nx  ! iterating through all the cells of the aquifer
579  !    do jj=1,ny
580  !      do ww=1,nz
581  !        rowPos=rowPos+1
582  !        if (ii==i.and.jj==j.and.ww==w) goto 1 ! position located, we end the function
583  !       end do
584  !    end do
585  !end do
586  ! the difference is that it really faster
587  end function
588
589
590
591
592
593
594  function rowCsPos (i,j,w,nx,ny,nz)
595 implicit none
596
597 ! calculating the distance from the first element for a single row  (positioning the Csorbed elements)
598 ! supposed the each row is written by the formula write(*,*) "C" (((i,j,w , w=1,nz),j=1
```

```
    ,ny),i=1,nx) for the c
!and then the same for Csorbed write(*,*) "Cs" (((i,j,w , w=1,nz),j=1,ny),i=1,nx)
! we would get  111 -112 -113 -114 -115 -121 -122 -123 -124 -125 -131-132 ....

integer i,j,w !The coordinates  i,j,w the cell each A coefficient is multiplicating
integer rowCsPos  ! ii,jjj,ww iterators
                        ! rowPos  the distance from the first element for a single
    row
integer nx,ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer to
    ,at  x,y,z directions.

 !  initializing values  in case something happens with the reading of the nx,ny,nz
    values

 rowCsPos=nx*ny*nz  !  All the C values are  located at the  beginning of each row  and
     then the Csorbed values follow
rowCsPos=rowCsPos+(i-1)*ny*nz+(j-1)*nz+w


 end function
```

```
1
2
3
4
5
6
7  module finite_differences
8  contains
9
10
11 subroutine aConstructor(nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2,dt,vDim,A,ia,ja)  ! responsible
       subroutine for creating A coefficients who will fill up the A vector
12 implicit none                                          ! also creates the neccessary vectors
        ia,ja needed to solve the 2*nx*ny*nz equations
13
14 integer i0,j0,w0 ! i0,j0,w0 are the coordinates of the cell ,for which we want to
       create the  A
15             ! coeffiecients (equations (1) and (2) are being silently written for each
        cell ,check page 509).
16 integer nx,ny,nz  ! nX,nY,nZ the number of pieces the user wants to split the aquifer
       to,at  x,y,z directions.
17
18
19 real*8 Dx,Dy,Dz !Dx,Dy,Dz longitudinal hydrodynamic dispersion coefficient
20 real*8 r1,r2,l1,l2 !r1=forward rate coefficient ,r2= reverse rate coefficient,l1=decay
       rate of liquid phase solute
21                   ! l2=l*=decay rate of sorbed solute
22 real*8 rD,th  !th=porosity (or th=th* the effective porosity can be used as well  page
       509) ,rD= bulk density of the solid matrix(solid mass/aquifer volume)
23 real*8 U !U average interstitial velocity
24 real*8 dxx,dyy,dzz,dt ! dxx=the lenght of  each cell the total aquifer is splitted to.
25                       ! dyy=the witdth of  each cell the total aquifer is splitted to.
26                       ! dzz=the height of  each sell the total aquifer is splitted to.
27                       ! dt= the  (time) distance between two sequential time moments t1
       ,t2  =dt=t2-t1
28 real*8 wm ! wm determines the numerical mode that will be  used for the soliving of the
        tranport or cootranport problem. If wm=0.5 we have crank nikolson . If wm=1 we
       have full implicit scheme
29 real*8 A1,A2,A3,A4,A5,A6,A7,A8,A9,A10 ! coeffiecients
30 integer rowPoss(10),rowPosss(8)  ! rowPoss,rowPosss  the distance from the first
       element (column) for a single row
31 integer rowPos,rowCsPos ! rowPos, rowCsPos =check the corresponding functions
32 integer ii,jj,ww ! ii,jj,ww=iterators
33 integer currentRow ! currentRow holds the number of the current row for which the
       coefficients are been written
34                   !(equations (1) and (2) are being silently written for each cell ,
       check page 509).
35 logical BoundaryCell  !  will indicate if this is a boundary cell or not .  True means
       this is actually a boundary cell
36 logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out if the according
       coefficients should be added to the A matrix or not .
37                                                ! true means they should be added
38 integer vDim ! this   holds the lenght of the vector that will store  the coefficient
       matrix A
39 real*8 ,allocatable :: A(:)  ! A vector will store the matrix that holds all the known
       concetration coefficients at time tn+1 .
40             !  All the c values are  located at the  beginning of each row and then the
       Csorbed values follow
41             !A=c+cs   ! non sorbed and sorbed concetrations coefficients
42 integer  ,allocatable :: ia(:),ja(:) ! vectors that are used by the sparse matrix solver
        ! please check the manual of parDiso page 2351  (mkl 10.2.2.25)
43 integer rowIndex ! please check the manual of parDiso page 2351
44 integer elementIndex ! the index of each non zero element in the A vector
45 integer iaIndex   ! used on ia vector
46
47 common/VarTrans1a/u
48 common/ VarTran2/rD,th
49 common/VarTrans3/dxx,dyy,dzz
50 common/vartrans5/wm
51
52 !initializing values in case something went wrong during reading
53 A1=0
54 A2=0
55 A3=0
```

```
A4=0
A5=0
A6=0
A7=0
A8=0
A9=0
A10=0
rowPoss=0
rowPosss=0
currentRow=0
BoundaryCell=.False.
flag3=.False.
flag4=.False.
flag5=.False.
flag6=.False.
flag7=.False.
flag8=.False.
A=0.D0
ia=0
ja=0
rowIndex=0
elementIndex=0
iaIndex=0
! end of initializations




 !The way c and cs are dirtibuted on each row the same way the are distibuted for each column  (check rowPos function)

     do ii=1,nx  ! iterating through all the cells of the aquifer
        do jj=1,ny
         do ww=1,nz


           ! setting the current cell coordinates
           i0=ii
           j0=jj
           w0=ww
           currentRow=currentRow+1  ! increasing the row ....everytime the cell is changing

           call  isBoundaryCell (BoundaryCell,i0,j0,w0,nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2,dt,A1,A2,A3,A4,A5,A6,A7,A8,flag3,flag4,flag5,flag6,flag7,flag8)  ! calls the respective subroutine to verify if cell is boundary or not

           if ( BoundaryCell ==.False.) then
           ! this means we can use the default coefficients


            !writting and saving the a coeffiecients
            ! all the coefficients are written in the order they appear in any row of the matrix

            A3=(-Dx/dxx**2.D0 -U/(2.D0*dxx))*wm
            !for upstream speed scheme  !A3=-dt*Dx/dxx**2.D0  -dt*U/dxx
            ! A3 multiplies Ci0-1,j0,w0
            rowPoss(1)= rowPos(i0-1,j0,w0,ny,nz)
            elementIndex=elementIndex+1             ! each time we write an element we increase the elementIndex
            A(elementIndex)=A3
            ja(elementIndex)=rowPoss(1)            ! filling the neccessary vectors for the inversions
            iaIndex=iaIndex+1                      ! iaIndex is used to position elements on the Ia vector
            ia(iaIndex)= elementIndex  ! the index in the general matrix of the first non zero element in a row.

            A5=-wm*Dy/dyy**2.D0
            ! A5 multiplies Ci0,j0-1,w0
            rowPoss(2)=  rowPos(i0,j0-1,w0,ny,nz)
            elementIndex=elementIndex+1
```

```
121             A(elementIndex)=A5
122             ja(elementIndex)=rowPoss(2)
123
124
125             A7=-wm*Dz/dzz**2.D0
126             ! A7 multiplies Ci0,j0,w0-1
127             rowPoss(3)=  rowPos(i0,j0,w0-1,ny,nz)
128             elementIndex=elementIndex+1
129             A(elementIndex)=A7
130             ja(elementIndex)=rowPoss(3)            ! filling the neccessary vectors for
        the inversions
131
132
134             A1=(1/dt+wm*(2.D0*Dx/dxx**2.D0   +2.D0*Dy/dyy**2.D0  +2.D0*Dz/dzz**2.D0 +l1
        ))
135                !A1=(1+dt*(2.D0*Dx/dxx**2.D0   +2.D0*Dy/dyy**2.D0  +2.D0*Dz/dzz**2.D0 +
       l1+U/dxx)) !for upstream speed scheme
136             ! calculating the column where every A  should be written to
137             ! A1 multiplies Ci0,j0,w0
138             rowPoss(4)= rowPos(i0,j0,w0,ny,nz)
139             elementIndex=elementIndex+1
140             A(elementIndex)=A1
141             ja(elementIndex)=rowPoss(4)            ! filling the neccessary vectors for
        the inversions
142
143
144             A8=-wm*Dz/dzz**2.D0
145             ! A8 multiplies Ci0,j0,w0+1
146             rowPoss(5)=  rowPos(i0,j0,w0+1,ny,nz)
147             elementIndex=elementIndex+1
148             A(elementIndex)=A8
149             ja(elementIndex)=rowPoss(5)            ! filling the neccessary vectors for
        the inversions
150
151
152             A6=-wm*Dy/dyy**2.D0
153             ! A6 multiplies Ci0,j0+1,w0
154             rowPoss(6)= rowPos(i0,j0+1,w0,ny,nz)
155             elementIndex=elementIndex+1
156             A(elementIndex)=A6
157             ja(elementIndex)=rowPoss(6)            ! filling the neccessary vectors for
        the inversions
158
159
160
161             A4=-wm*Dx/dxx**2.D0  +wm*U/(2.D0*dxx)
162             !for upstream speed scheme !A4=-dt*Dx/dxx**2.D0
163             ! A4 multiplies Ci0+1,j0,w0
164             rowPoss(7)= rowPos(i0+1,j0,w0,ny,nz)
165             elementIndex=elementIndex+1
166             A(elementIndex)=A4
167             ja(elementIndex)=rowPoss(7)            ! filling the neccessary vectors for
        the inversions
168
169
170             A2=(1/dt+wm*l2)*rD/th
171             ! A2 multiplies Ci0,j0,w0
172             rowPoss(8)= rowCsPos(i0,j0,w0,nx,ny,nz) ! Csorbed coefficient
173             elementIndex=elementIndex+1
174             A(elementIndex)=A2
175             ja(elementIndex)=rowPoss(8)            ! filling the neccessary vectors for
        the inversions
176
177
178
179          else
180
181
182                       ! finding the first non zero element in a row
183              iaIndex=iaIndex+1                      ! iaIndex is used to position
       elements on the Ia vector
184             ia(iaIndex)=elementIndex+1  ! the index in the general matrix of the first
```

```
     non zero element in a row.
185                                     !+1 because the    elementIndex holded the
    last non zero element in the previous row
186                                     !( while we want the fist non zero element in
    the next row)
187
189
190             if(flag3==.True.) then
191             ! A3 multiplies Ci0-1,j0,w0
192             rowPosss(1)= rowPos(i0-1,j0,w0,ny,nz) ! calculating the column where every
    A  should be written to
193             elementIndex=elementIndex+1
194             A(elementIndex)=A3
195             ja(elementIndex)=rowPosss(1)            ! filling the neccessary vectors
    for the inversions
196
197             end if
198
199             if(flag5==.True.) then
200             ! A5 multiplies Ci0,j0-1,w0
201             rowPosss(2)=  rowPos(i0,j0-1,w0,ny,nz)
202             elementIndex=elementIndex+1
203             A(elementIndex)=A5
204             ja(elementIndex)=rowPosss(2)            ! filling the neccessary vectors
    for the inversions
205             end if
206
207             if(flag7==.True.) then
208             ! A7 multiplies Ci0,j0,w0-1
209             rowPosss(3)=  rowPos(i0,j0,w0-1,ny,nz)
210             elementIndex=elementIndex+1
211             A(elementIndex)=A7
212             ja(elementIndex)=rowPosss(3)            ! filling the neccessary vectors
    for the inversions
213             end if
214
215             ! A1 multiplies Ci0,j0,w0
216             rowPosss(4)= rowPos(i0,j0,w0,ny,nz)
217             elementIndex=elementIndex+1
218             A(elementIndex)=A1
219             ja(elementIndex)=rowPosss(4)            ! filling the neccessary vectors
    for the inversions
220
221
222             if(flag8==.True.) then
223             ! A8 multiplies Ci0,j0,w0+1
224             rowPosss(5)=  rowPos(i0,j0,w0+1,ny,nz)
225             elementIndex=elementIndex+1
226             A(elementIndex)=A8
227             ja(elementIndex)=rowPosss(5)            ! filling the neccessary vectors
    for the inversions
228
229             end if
230
231
232             if(flag6==.True.) then
233             ! A6 multiplies Ci0,j0+1,w0
234             rowPosss(6)= rowPos(i0,j0+1,w0,ny,nz)
235             elementIndex=elementIndex+1
236             A(elementIndex)=A6
237             ja(elementIndex)=rowPosss(6)            ! filling the neccessary vectors
    for the inversions
238             end if
239
240             if(flag4==.True.) then
241             ! A4 multiplies Ci0+1,j0,w0
242             rowPosss(7)= rowPos(i0+1,j0,w0,ny,nz)
243             elementIndex=elementIndex+1
244             A(elementIndex)=A4
245             ja(elementIndex)=rowPosss(7)            ! filling the neccessary vectors
    for the inversions
246             end if
247
```

```
248             ! A2 multiplies Ci0,j0,w0
249             rowPosss(8)= rowCsPos(i0,j0,w0,nx,ny,nz) ! Csorbed coefficient
250             elementIndex=elementIndex+1
251             A(elementIndex)=A2
252             ja(elementIndex)=rowPosss(8)            ! filling the neccessary vectors for the inversions
253
254
255
256
257           end if
258
259          call rightOrdered(BoundaryCell ,rowPoss) ! checking whether the coefficients are placed correctly on the matrix
260
261         end do
262       end do
263     end do
264
265
266
267
268   !carefull  currentRow now  holds the last row that the previous iterations finished with  (C non sorbed concetration coefficient rows)
269
270      do ii=1,nx  ! iterating through all the cells of the aquifer (including the boundary cells)
271         do jj=1,ny  ! Cs sorbed equations  (eq 2 page 509) are based on the same cell without affecting the neighbouring cells ...
272           do ww=1,nz  ! so no attention should be given whether the  the cell is boundary or not  .
273
274                   ! Creating a coefficients for the Cs (sorbed) concetrations and adding them
275                   ! accordingly after C (non sorbed) rows.
276                   ! the cs rows are after the c rows  .
277                   !equation  (2) is being silently written for each cell ,check page 509.
278
279             ! setting the current cell coordinates
280             i0=ii
281             j0=jj
282             w0=ww
283             currentRow=currentRow+1  !carefull  currentRow  holds the last row that the previous iterations  (C rows (non sorbed rows)) finished with
284
285             !writting and saving the a coeffiecients
286              A9=+wm*r1*th/rD
287             ! A9 multiplies Ci0,j0,w0
288             rowPoss(9)=  rowPos(i0,j0,w0,ny,nz)
289             elementIndex=elementIndex+1
290             A(elementIndex)=A9
291             ja(elementIndex)=rowPoss(9)            ! filling the neccessary vectors for the inversions
292             iaIndex=iaIndex+1                      ! iaIndex is used to position elements on the Ia vector
293             ia(iaIndex)= elementIndex  ! the index in the general matrix of the first non zero element in a row.
294
295
296             A10=-(1.D0/dt +wm*(l2+r2))
297             ! A10 multiplies Ci0,j0,w0
298             rowPoss(10)=  rowCsPos(i0,j0,w0,nx,ny,nz) ! Csorbed coefficient
299             elementIndex=elementIndex+1
300             A(elementIndex)=A10
301             ja(elementIndex)=rowPoss(10)            ! filling the neccessary vectors for the inversions
302
303             end do
304          end do
305       end do
306
307
308
```

```
309     ! finally we add the last element to ia holding the maximum number of non zeroes
         elements plus 1
310    iaIndex=iaIndex+1
311    ia(iaIndex)=vDim+1
312
313
314
315  end subroutine
316
317
318
319
320
321
322
323
324
325   subroutine rightOrdered(BoundaryCell ,rowPoss)
326   ! this subroutine checks if the coefficient are properly positioned on any row
327   implicit none
328
329   integer rowPoss(10)
330   logical BoundaryCell
331   integer i
332   i=0
333
334   if ( BoundaryCell==.false.)   then ! we are not on a boundary cell
335
336          do i=1,7         ! 1-7 and not 1-8 because the last because when i=7 ,i+1=8
337
338             if(rowPoss(i)<rowPoss(i+1) ) then
339             else
340              goto 11
341             end if
342
343          end do
344
345
346    end if
347
348   goto 8 ! everything went as expected
349
350
351   11 write(*,*) "critical error please try again ,the A coefficients wont take the right
       place in the A matrix "
352      write(*,*) "the system will now stop"
353       pause
354
355   8 end subroutine
356
357
358
359
360
361  end module finite_differences
```

```
1  Subroutine validating_input(lxx,lyy,lzz,nx,ny,nz,tp_C,tp_V,CooTransp)
2  use  AA_BB_Solver, only: dt  ! uses the module to share globally some intresting
       variables
3  implicit none
4  ! this routine checks the quality of the input data
5  !ex. source cell is out off the aquifer.
6  !real*8 dt ! dt= the  (time) distance between two sequential time moments t1,t2  =dt=t2
       -t1 ! defined in the  AA_BB_Solver module
7
8  real*8 Rcadc,Lca   !Rcadc=r2= reverse rate coefficient    ,Lca=l2=l*=decay rate of
       sorbed solute
9  real*8 Rcdca,Lc    !Rcdca=r1=forward rate coefficient ,=Lc=l1=decay rate of liquid
       phase solute
10 real*8 Rvdva,Rvadv ! forward or reverse rates cooefficients  (Virus case)!!!!!!!!!!!!!!
       !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
11 real*8 Lv,Lva      ! Lva=l2=l*=decay rate of sorbed solute, Lv=l1=decay rate of liquid
       phase solute
12 real*8 U,Dcx,Dcy,th,rD !Dcx,Dcy longitudinal hydrodynamic dispersion coefficient
       (colloid case)
13                       !th=porosity (or th=th* the effective porosity can be used as
       well  page 509) ,rD= bulk density of the solid matrix(solid mass/aquifer volume)
14 real*8 lx0_C,ly0_C,lz0_C ,Dcz    !lx0,ly0,lz0 =cartesian coordinates of the point
       source
15                         !Dcz=Vertical hydronamic dispersion coefficient (colloid
       case)
16 real*8 Dvx,Dvy,Dvz      !Dvx,Dvy  hydrodynamic dispersion coefficient of the virus
       component
17                         !Dvz=Vertical hydronamic dispersion coefficient of the virus
       component
18 real*8  Dcvx,Dcvy,Dcvz !Dcvx,Dcvy,Dcvz =hydrodynamic components (x,y,z direction) for
       the case when virus is attached to the colloid
19 real*8 lx0_V,ly0_V,lz0_V !lx0,ly0,lz0 =cartesian coordinates of the point source Virus
       case
20 real*8 lxx,lyy,lzz ! lxx,lyy,lzz the dimensions of the aquifer
21 integer  nX,nY,nZ ! nX,nY,nZ the number of pieces the user wants to split the aquifer
       to,at  x,y,z directions
22 real*8  Cm0_C,Q_C  !Cm0_C source (source cell)  constant conccetration  (Colloid case)
23                    ! Q_C is the flow rate with which we transfer soltuted mass
       (Concetration Cm0)to the source cell (Colloid case)
24 real*8 Cm0_V  ! The constant    concetration    of  the flow rate   [mgr/ml] Virus case

25 real*8 Q_V    ! is the flow rate with    which we transfer soltuted mass (Concetration
         Cm0)to  the source  cell    [cm^3/day]For the virus case
26 Integer printSurMode !Allows to print  a file PrintSur.txt a surface of the aquifer at
       any requested height H, for every time step Dt  (if 0 we print nothing ,if 1 we
       print for every time step).
27 integer PrintSurXYZ   !Mode (1 =yz plane is being printed,2=xz plane is being printed,
       =3 yz planed is being printed)
28 real*8 h
29 integer sourceSurf_C ! sourceSurfe determines on which plane the elliptic source is
       based
30                     !(sourceSurf_C=0  we dont have an elliptic source but a point one, =
       1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is
       based on XY plane)
31 integer sourceSurf_V ! sourceSurfe determines on which plane the elliptic source is
       based
32                     !(sourceSurf_V=0  we dont have an elliptic source but a point one, =
       1 Ellipse is based on YZ plane, =2 Ellipse is based on XZ plane  ,=3 Ellipse is
       based on XY plane)
33
34 real*8 aEl_C,bEl_C,cEl_C  !aEl semi-axis of the elliptic source ! check the manual for
       more info ! Colloid case
35                     !bEl semi-axis of the elliptic source !Check the manual for more
       info
36                     !cEl semi-axis of the elliptic source !Check the manual for more
       info
37
38 real*8 aEl_V,bEl_V,cEl_V  !aEl semi-axis of the elliptic source ! check the manual for
       more info !Virus case
39                     !bEl semi-axis of the elliptic source !Check the manual for more
       info
40                     !cEl semi-axis of the elliptic source !Check the manual for more
       info
41 real*8 wm ! wm determines the numerical mode that will be  used for the soliving of the
```

```
     tranport or cootranport problem. If wm=0.5 we have crank nikolson . If wm=1 we have full implicit scheme
42         ! wm should be within the range  0.5-1. Else we fall to explicit scheme the and stability is not ensured
43 real*8 tp_C ! tp is the active time of the Colloid source (time within the source provides with mass the model)
44 real*8 tp_V ! tp is the active time of the Virus source (time within the source provides with mass the model)
45 integer nMaxCcVectors ! Responsible for holding the max allowed Cc vectors. Since we require to aquire the solutions for the colloid component and then
46                       ! use them to get the solutions for the Virus Component. We can store the results that Colloid transportation produced and then use them
47                       ! to solve virus component. By doing so we avoid factorizing again and again the same matrix, without having to keep in memory the same time
48                       ! and the A matrix for the colloid component and the A matrix for the Virus component. Keep in mind that nMaxVectors require as well a lot of
49                       ! memory. By giving  larger nMaxCcVectors we gain speed but we need larger amount of memory. A valuearound 50 should be good.


51 integer CooTransp     ! Indicates wether we have simple transport (=0) or Cootransport (=1)
52
53
54
55 common/VarTrans1/Dcx,Dcy,Dcz,Lc,Lca,Rcdca,Rcadc
56 common/VarTrans1a/u
57 common/VarTran2/rD,th
58
59 common/vartrans5/wm
60 common/vartrans6/Lv,Lva,Rvdva,Rvadv
61 common/vartrans7/Dvx,Dvy,Dvz,Dcvx,Dcvy,Dcvz
62 common/printMode/printSurMode,PrintSurXYZ,h
63
64 common/sourceGeom_C/sourceSurf_C
65 common/sourceGeom2_C/aEl_C,bEl_C,cEl_C
66 common/sourceCoord_C/lx0_C,ly0_C,lz0_C
67 common/sourcePower_C/Q_C,Cm0_C
68
69 common/sourceGeom_V/sourceSurf_V
70 common/sourceGeom2_V/aEl_V,bEl_V,cEl_V
71 common/sourceCoord_V/lx0_V,ly0_V,lz0_V
72 common/sourcePower_V/Q_V,Cm0_V,nMaxCcVectors
73
74
75
76
77
78
79
80
81
82
83 if (lx0_C>lxx) then
84  write(*,*) " Error found in the input"
85  Write(*,*)" The position  _x_ of the source cant be outside of the aquifer _colloids_"
86  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
87  read(*,*)  ! slows down the system exit
88  stop
89 end if
90 if (ly0_C>lyy) then
91  write(*,*) " Error found in the input"
92  Write(*,*)" The position  _y_ of the source cant be outside of the aquifer _colloids_"
93  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
94  read(*,*) ! slows down the system exit
95  stop
96 end if
97 if (lz0_C>lzz) then
98  write(*,*) " Error found in the input"
99  Write(*,*)" The position  _z_ of the source cant be outside of the aquifer _colloids_"
100  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
101  read(*,*) ! slows down the system exit
102  stop
103 end if
```

```
104
105
106 if (lx0_V>lxx) then
107  write(*,*) " Error found in the input"
108  Write(*,*)" The position  _x_ of the source cant be outside of the aquifer _Virus_"
109  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
110  read(*,*)  ! slows down the system exit
111  stop
112 end if
113 if (ly0_V>lyy) then
114  write(*,*) " Error found in the input"
115  Write(*,*)" The position  _y_ of the source cant be outside of the aquifer  _Virus_"
116  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
117  read(*,*) ! slows down the system exit
118  stop
119 end if
120 if (lz0_V>lzz) then
121  write(*,*) " Error found in the input"
122  Write(*,*)" The position  _z_ of the source cant be outside of the aquifer  _Virus_"
123  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
124  read(*,*) ! slows down the system exit
125  stop
126 end if
127
128
129
130 if(lx0_C<0.D0 .or.ly0_C<0.or.lz0_C<0) then
131  write(*,*) " Error found in the input  (lx0,ly0,lz0)_Colloids_"
132  write(*,*)" The coordinates of the source cell cant be negative"
133  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
134  read(*,*) ! slows down the system exit
135  stop
136 end if
137
138 if(lx0_V<0.D0 .or.ly0_V<0.or.lz0_V<0) then
139  write(*,*) " Error found in the input  (lx0,ly0,lz0) _Virus_"
140  write(*,*)" The coordinates of the source cell cant be negative"
141  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
142  read(*,*) ! slows down the system exit
143  stop
144 end if
145
146
147 if(lxx<=0.D0 .or.lyy<=0.or.lzz<=0) then
148  write(*,*) " Error found in the input  (lxx,lyy,lzz)"
149  write(*,*)" The dimensions of the aquifer cant be negative"
150  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
151  read(*,*) ! slows down the system exit
152  stop
153 end if
154
155 if (Q_C<=0.or.Cm0_C<=0)then
156 write(*,*)"Possible error found during reading input file (Q,Cm0) _Colloids_"
157 write(*,*)"Source flow rate or Concetration of the source flow  shouldnt be zero or
        negative."
158 write(*,*)"The only case this is allowed is when   |you|  have given initial source
        concetraton other than zero to the aquifer."
159 write(*,*) "And now you want to observe  the way the concetrations will be distributed
        if you stop the source flow rate,or you begin to remove"
160 write(*,*)"solutes from the aquifer."
161 write(*,*)"Please push enter to continue with the excisting input file"
162 read(*,*)
163 end if
164
165
166 if (Q_V<=0.or.Cm0_V<=0)then
167 write(*,*)"Possible error found during reading input file (Q,Cm0 _Virus_)"
168 write(*,*)"Source flow rate or Concetration of the source flow  shouldnt be zero or
        negative."
169 write(*,*)"The only case this is allowed is when   |you|  have given initial source
        concetraton other than zero to the aquifer."
170 write(*,*) "And now you want to observe  the way the concetrations will be distributed
        if you stop the source flow rate,or you begin to remove"
171 write(*,*)"solutes from the aquifer."
```

```
172 write(*,*)"Please push enter to continue with the excisting input file"
173 read(*,*)
174 end if
175
176
177
178 if(nx<=0.D0 .or.ny<=0.or.nz<=0) then
179  write(*,*) " Error found in the input (nx,ny,ny)"
180  write(*,*)" You cant split the aquifer in a |negative| or |zero| number of pieces"
181  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
182  read(*,*) ! slows down the system exit
183  stop
184 end if
185
186
187 if(Rcadc<0.D0 .or.Rcdca<0) then
188  write(*,*) " Error found in the input (r2,r1)"
189  write(*,*)" The reverse rate (r2) and forward rate (r1) coefficient cant be negative"
190  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
191  read(*,*) ! slows down the system exit
192  stop
193 end if
194
195 if(Rvadv<0.D0 .or.Rvdva<0) then
196  write(*,*) " Error found in the input (Rvadv=r2,Rvdva=r1)"
197  write(*,*)" The reverse rate (r2) and forward rate (r1) coefficient cant be negative"
198  write(*,*)"please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
199  read(*,*) ! slows down the system exit
200  stop
201 end if
202
203
204 if(Lc<0.D0 .or.Lca<0) then
205  write(*,*) " Error found in the input (Lc=l1,Lca=l2)"
206  write(*,*)" The l2=l*=decay rate of sorbed solute and the l1=decay rate of liquid
        phase solute cant be negative "
207  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
208  read(*,*) ! slows down the system exit
209  stop
210 end if
211
212 if(Lv<0.D0 .or.Lva<0) then
213  write(*,*) " Error found in the input (Lv=l1,Lva=l2)"
214  write(*,*)" The l2=l*=decay rate of sorbed solute and the l1=decay rate of liquid
        phase solute cant be negative "
215  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
216  read(*,*) ! slows down the system exit
217  stop
218 end if
219
220  if(Dcx<0.D0 .or.Dcy<0.or.Dcz<0) then
221   write(*,*) " Error found in the input (Dx,Dy,Dz)"
222   write(*,*)" The Hydrodynamic coefficient cant be negative  ."
223   write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
224   read(*,*) ! slows down the system exit
225   stop
226 end if
227
228  if(Dvx<0.D0 .or.Dvy<0.or.Dvz<0) then
229   write(*,*) " Error found in the input (Dx,Dy,Dz)"
230   write(*,*)" The Hydrodynamic coefficient cant be negative  ."
231   write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
232   read(*,*) ! slows down the system exit
233   stop
234 end if
235
236
237  if(Dcvx<0.D0 .or.Dcvy<0.or.Dcvz<0) then
238   write(*,*) " Error found in the input (Dx,Dy,Dz)"
239   write(*,*)" The Hydrodynamic coefficient cant be negative  ."
240   write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
241   read(*,*) ! slows down the system exit
242   stop
243 end if
```

```
244
245
246
247
248 if(rD<=0.D0 .or.th<=0) then
249  write(*,*) " Error found in the input (rD,th)"
250  write(*,*)" The th=porosity  OR the rD= bulk density of the solid matrix(solid mass/
        aquifer volume) cant be&
251  negative or zero"
252  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
253  read(*,*) ! slows down the system exit
254  stop
255 end if
256
257
258
259 if(dt<=0.D0) then
260  write(*,*) " Error found in the input (dt)"
261  write(*,*)" The time step Dt cant be zero or negative."
262  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
263  read(*,*) ! slows down the system exit
264  stop
265 end if
266 if(tp_C<0.D0.or.tp_V<0) then
267  write(*,*) " Error found in the input (tp_C)or (tp_V)"
268  write(*,*)" The tp_C or tp_V (  time   source is active)cant be negative ,time must be
         positive number "
269  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
270  read(*,*) ! slows down the system exit
271  stop
272 end if
273
274 if(h<0.D0) then
275  write(*,*) " Error found in the input (h)"
276  write(*,*)" The height of the surface to be printed should be  a non negative number "
277  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
278  read(*,*) ! slows down the system exit
279  stop
280 end if
281 ! U average intersistial velocity has no restrictions.... can be negadive ,zero or a
        positive number
282
283 if (printSurMode==0.or.printSurMode==1.or.printSurMode==2.or.printSurMode==3)then !
        Colloid case
284 else if(printSurMode==11.or.printSurMode==22.or.printSurMode==33)then  ! Virus case
285 else
286  write(*,*) " Error found in the input (printSurMode)"
287  write(*,*)"PrintSurMod determines whether we want to print a surface or not...Thus
        should be 0 or 1 or 2 or 3 only "
288  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
289  read(*,*) ! slows down the system exit
290  stop
291 end if
292
293 if(PrintSurXYZ==1 .or.PrintSurXYZ==2.or.PrintSurXYZ==3  ) then
294 else
295 write(*,*) " Error found in the input (PrintSurXYZ)"
296  write(*,*)"PrintSurXYZ determines which plane we want to print .Mode 1 =yz plane is
        being printed,2=xz plane is being printed, =3 yz planed is being printed "
297  write(*,*)" Thus PrintSurXYZ should be only 1 or 2 or 3 "
298  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
299  read(*,*) ! slows down the system exit
300  stop
301 end if
302
303 if(sourceSurf_C==0 .or.sourceSurf_C==1 .or.sourceSurf_C==2.or.sourceSurf_C==3 .or.
        sourceSurf_C ==4) then
304 else
305 write(*,*) " Error found in the input (sourceSurf_C)"
306  write(*,*)"sourceSurf determines on which plane the elliptic source is on ."
307  write(*,*)"If sourceSurf_C=0 our source isnt elliptic  but apoint one ,"
308  write(*,*)"if sourceSurf_C=1  Ellipse is on xz plane   ,  =2 xz plane , 3 yz plane ,4
        ellipsoid source"
309  write(*,*)" Thus sourceSurf should be only 0 or  1 or 2 or 3 only"
```

```
310  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
311  read(*,*) ! slows down the system exit
312  stop
313 end if
314
315 if(sourceSurf_C==1 .or.sourceSurf_C==2.or.sourceSurf_C==3 .or.sourceSurf_C==4 ) then
316  if(aEl_C<=0.D0 .or. bEl_C<=0.or. cEl_C<=0) then
317   write(*,*) " Error found in the input (aEl_C or bEl_C or cEl_C)"
318   write(*,*)" The dimensions of the elliptic source cant be zero or negative , set
        sourceSurf=0 if you want a point source "
319   write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
320   read(*,*) ! slows down the system exit
321   stop
322  end if
323 end if
324
325
326
327
328 if(sourceSurf_V==0 .or.sourceSurf_V==1 .or.sourceSurf_V==2.or.sourceSurf_V==3 .or.
      sourceSurf_V==4 ) then
329 else
330 write(*,*) " Error found in the input (sourceSurf_V)"
331  write(*,*)"sourceSurf_V determines on which plane the elliptic source is on ."
332  write(*,*)"If sourceSurf_V=0 our source isnt elliptic  but apoint one ,"
333  write(*,*)"if sourceSurf_V=1  Ellipse is on xz plane   ,  =2 xz plane , 3 yz plane ,4
      ellipsoid source"
334  write(*,*)" Thus sourceSurf should be only 0 or  1 or 2 or 3 only"
335  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
336  read(*,*) ! slows down the system exit
337  stop
338 end if
339
340 if(sourceSurf_C==1 .or.sourceSurf_C==2.or.sourceSurf_C==3 .or.sourceSurf_C==4 ) then
341  if(aEl_C<=0.D0 .or. bEl_C<=0.or. cEl_C<=0) then
342   write(*,*) " Error found in the input (aEl_V or bEl_V or cEl_V)"
343   write(*,*)" The dimensions of the elliptic source cant be zero or negative , set
        sourceSurf=0 if you want a point source "
344   write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
345   read(*,*) ! slows down the system exit
346   stop
347  end if
348 end if
349
350
351 if(Wm<0.5D0.or.Wm>1) then
352  write(*,*) " Error found in the input (Wm)"
353  write(*,*)" wm determines the numerical mode that will be  used for the soliving of
      the tranport or cootranport problem"
354  write(*,*)"wm should be within the range  0.5-1. Else we fall to explicit scheme and
      the stability is not ensured"
355  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN",
      "Wm found=",Wm
356  read(*,*) ! slows down the system exit
357  stop
358 end if
359
360 if(nMaxCcVectors<=0) then
361  write(*,*) " Error found in the input nMaxCcVectors"
362  write(*,*) " Responsible for holding the max allowed Cc vectors. Since we require to
      aquire the solutions for the colloid component and then"
363  write(*,*) " use them to get the solutions for the Virus Component. We can store the
      results that Colloid transportation produced and then use them"
364  write(*,*) " to solve virus component. By doing so we avoid factorizing again and
      again the same matrix, without having to keep in memory the same time"
365  write(*,*) " and the A matrix for the colloid component and the A matrix for the Virus
       component. Keep in mind that nMaxVectors require as well a lot of"
366  write(*,*) " memory. By giving  larger nMaxCcVectors we gain speed but we need larger
      amount of memory. A value around 50 should be good."
367  write(*,*) " nMaxCcVectors should be a non negative integer value, Instead the value
      nMaxCcVectors= ",nMaxCcVectors, "was found"
368  write(*,*) " Please change the inputC file and try again ,THE SYSTEM WILL NOW SHUT
      DOWN"
369  read(*,*) ! slows down the system exit
```

```
370  stop
371 end if
372
373
374 if(CooTransp==0 .or.CooTransp==1) then
375 else
376  write(*,*) " Error found in the input (CooTransp)"
377  write(*,*) " Indicates wether we have simple transport (=0) or Cootransport(=1) "
378  write(*,*)" Thus CooTransp should be only 0 or  1 , While it Was found to be",
        Cootransp
379  write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN"
380  read(*,*) ! slows down the system exit
381  stop
382 end if
383
384 end subroutine
385
386
387
```

```
 1
 2 subroutine  main(nx,ny,nz,lxx,lyy,lzz,tp_C,tp_V)
 3 ! main subroutin connects the input andoutput from the basic program with the finite
       differences functions
 4 use finite_differences  ! Loading neccessary modules, modules allow passing allocatable
        vetors without the need to
 5 use prntOutPut          ! explicitly determine the vector length (Fortran passes
       vectors lenghts automatically when comes to modules)
 6 use visitOutPut         ! Also vectors created this way dont use static memory and can
       dynamically use all available ram
 7 use fcn_Vector_Simple_Trans
 8 use fcn_Vector_Cootrans
 9 use  AA_BB_Solver
10 implicit none
11
12 integer nx,ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer to
       ,at  x,y,z directions.
13 real*8 lxx,lyy,lzz    ! lxx,lyy,lzz the dimensions of the aquifer
14 real*8 dxx,dyy,dzz    ! dxx=the lenght of  each cell the total aquifer is splitted to.
15                       ! dyy=the witdth of  each cell the total aquifer is splitted to.
16                       ! dzz=the height of  each sell the total aquifer is splitted to.
17
18
19  real*8 r1,r2,l1,l2 ! r1,r2,l1,l2=temp variables that either will hold values from
       Colloids or Virus ! l2=l*=decay rate of sorbed solute
20                     !r1=forward rate coefficient ,r2= reverse rate coefficient,l1=decay
        rate of liquid phase solute
21  real*8 U           ! interstitial speed
22  real*8 Rcadc,Lca   ! Rcadc=r2= reverse rate coefficient    ,Lca=l2=l*=decay rate of
       sorbed solute
23  real*8 Rcdca,Lc    ! Rcdca=r1=forward rate coefficient ,Lc=l1=decay rate of liquid
       phase solute
24  real*8 Lv,Lva      ! Lva=l2=l*=decay rate of sorbed solute, Lv=l1=decay rate of liquid
        phase solute
25  real*8 Rvdva,Rvadv ! forward or reverse rates cooefficients  (Virus case)
26
27
28  real*8 F_C !F= general functional form of virus source configuration [ML-3 t-1]=[gr cm
       -3 Day-1], colloid case
29  real*8 F_V !F= general functional form of virus source configuration [ML-3 t-1]=[gr cm
       -3 Day-1], Virus case
30  real*8 lx0_C,ly0_C,lz0_C  !lx0_C,ly0_C,lz0_C =cartesian coordinates of the point
       source (Colloid case)
31  real*8 lx0_V,ly0_V,lz0_V  !lx0_V,ly0_V,lz0_V =cartesian coordinates of the point
       source (Virus case)
32  real*8  t1   ! t1= the total time after which we want to calculate the concetrations
33
34  real*8 Dvx,Dvy,Dvz      !Dvx,Dvy  hydrodynamic dispersion coefficient of the virus
       component
35                          !Dvz=Vertical hydronamic dispersion coefficient of the virus
       component
36  real*8 Dcx,Dcy,Dcz      !Dcx,Dcy  hydrodynamic dispersion coefficient of the Colloid
       component
37                          !Dcz=Vertical hydronamic dispersion coefficient of the Colloid
       component
38
39  real*8 Dx,Dy,Dz     !Dx,Dy  hydrodynamic dispersion coefficient Will other hold Virus
       or Colloid component
40                          !Dz =Vertical hydronamic dispersion coefficient  Will other
       hold Virus or Colloid component
41  real*8 tc,tmpTc ! tc = current time= total time since the source begun providing mass,
       tmpTc temporary variable holding tc value
42  integer vDim ! this   holds the lenght of the vector that will store  the coefficient
       matrix A
43  integer nX0_C,ny0_C,nZ0_C ! nX0_C,ny0_C,nZ0_C  cell coordinates of the point source
       (Colloid source)
44  integer nX0_V,ny0_V,nZ0_V ! nX0_V,ny0_V,nZ0_V  cell coordinates of the point source
       (Virus  source)
45  integer sourCellNum ! the number of total cells the elliptic source was divided to
46  integer ii,iii,ir !iterators
47  logical  allCcConcPrinted ! Indicates if all colloid concetrations were properly
       printed . If true then all colloid questions where answered
48                            ! and  prntOutCv subroutine ,if ready, can convert the temp.
       txt record file to print.txt formatted file
```

```
49  Integer printSurMode !Allows to print  a file PrintSur.txt a surface of the aquifer at
        any requested height H, for every time step Dt
50                       !(if 0 we print nothing ,if 1 we print for every time step).
51  integer PrintSurXYZ   !Mode (1 =yz plane is being printed,2=xz plane is being printed,
        =3 yz planed is being printed)
52  real*8 h !H determines the height of the surface to be printed. (cm)
53  integer rowPos ,rowPosSource ! rowPos  the distance from the first element for a
       single row
54                               !rowPosSource is the rowPos for the source cell
55  real*8   Q_C,Cm0_C,thh !Cm0 source (source cell)  constant conccetration  ! Q is the
       flow rate with which we transfer soltuted mass (Concetration Cm0)to the source cell

56                      !th=thh==porosity (or th=th* the effective porosity can be used as
       well  page 509)
57  real*8 Q_V    ! is the flow rate with  which we transfer soltuted mass (Concetration
           Cm0)to  the source  cell    [cm^3/day]For the virus case

58  real*8 Cm0_V  ! The constant    concetration    of  the flow rate   [mgr/ml] Virus case

59
60  integer Source_Type_C  ! the type of the Colloid Source,=1 means constant source
       concetration,=2 means constant mass flow! attention constant source concetration
       can be valid only for influent cells
61  integer Source_Type_V  ! the type of the Virus Source,=1 means constant source
       concetration,=2 means constant mass flow! attention constant source concetration
       can be valid only for influent cells
62
63  real*8 tp_C ! tp is the active time of the Colloid source (time within the source
       provides with mass the model)
64  real*8 tp_V ! tp is the active time of the Virus source (time within the source
       provides with mass the model)
65
66 !neccessary vectors for the cootranpotrtp
67  real*8 , allocatable :: fCnC(:)        !  fCn will be holding  the stable vector of
       equation   AxC=fCnC   (fCn is afunction of Cvn =known concetrations at time t=tn,
       for the virus case)
68  real*8 , allocatable :: Cvnp1(:)       ! Cvnp1 matrix will hold all the unknown
       concetrations (sorbed and non sorbed ) at time tn+1  (Virus case)
69  real*8 , allocatable :: Ccn(:)         ! Ccn matrix will hold all the unknown
       concetrations (sorbed and non sorbed ) at time tn    (Colloid case)
70  real*8 , allocatable :: AAVector(:,:)  ! Vector holding the solutions of the 8, 13 eqs
       . Where aa=CcCvc and BB=Cc*Cvc*  ! can hold values for t=tn and t=tn+1
71  real*8 , allocatable :: BBVector(:,:)  ! Vector holding the solutions of the 8, 13 eqs
       . Where aa=CcCvc and BB=Cc*Cvc*  ! can hold values for t=tn and t=tn+1
72                                          ! Paper Colloid-Facilitated virus transport in
       saturated porus media . Vasiliki i.Syngouna Constantinos V.Chrysicopoulos
73  real*8 , allocatable :: AAVectors(:,:) ! Vector holding the solutions of the 8, 13 eqs
       . Where aa=CcCvc and BB=Cc*Cvc*  ! can hold values for t=tn ...t=tnmax (nmax=
       nMaxCcVectors)
74  real*8 , allocatable :: BBVectors(:,:) ! Vector holding the solutions of the 8, 13 eqs
       . Where aa=CcCvc and BB=Cc*Cvc*  ! can hold values for t=tn ...t=tnmax (nmax=
       nMaxCcVectors)
75                                          ! Paper Colloid-Facilitated virus transport in
       saturated porus media . Vasiliki i.Syngouna Constantinos V.Chrysicopoulos

76  real*8 , allocatable :: CcVectors(:,:) ! Responsible for holding  Cc solutions ,for
       successive times . Since we require to aquire the solutions for the colloid
       component and then
77                       ! use them to get the solutions for the Virus Component. We can
       store the results that Colloid transportation produced and then use them
78                       ! to solve virus component. By doing so we avoid factorizing
       again and again the same matrix, without having to keep  in memory the same time
79                       ! and the A matrix for the colloid component and the A matrix for
        the Virus component. Keep in mind that nMaxVectors require as well a lot of
80                       ! memory. By giving  larger nMaxCcVectors we gain speed but we
       need larger amount of memory. A valuearound 50 should be good.

81 real*8,allocatable::AAn(:),BBn(:)! AAn vector will hold all (all cells) initial known
       concetrations (AAn=CcCvc ) at time tn
82                                   ! BBn vector will hold all (all cells) initial known
       concetrations (BBn=Cc*Cvc* ) at time tn,
83                                   ! So we can calculate values on t=tnp1 (ex.t1) based
       on values t=tn (ex. t0)
84 !end of neccessary vectors for the cootranport
```

```
 85
 86 ! Start of <<<<<<<PARDISO>>>>>>>> variables
 87  integer*8 ,allocatable:: pt(:)
 88  real*8,allocatable:: A(:)
 89  integer,allocatable:: ia(:),ja(:),perm(:),iparm(:)
 90  integer maxfct ,mnum,mtype,phase,nrhs,msglvl
 91  integer n ,error
 92  integer mkl_get_max_threads
 93  integer mkl_num_threads
 94  real*8 ,allocatable :: Ccnp1(:)  ! Ccnp1 matrix will hold all the unknown concetrations (sorbed and non sorbed ) at time tn+1
 95            !  All the c values are  located at the  beginning of each row and then the Csorbed values follow
 96            ! C=c+cs   ! non sorbed and sorbed concetrations ,Colloid case
 97  real*8 ,allocatable ::  fCn(:)   !  fCn will be holding  the stable vector of equation A[C]=fCn
 98            !(fCn is afunction of Cn =known concetrations at time t=tn)
 99            !All the c values are  located at the  beginning of each row and then the Csorbed values follow
100            !A=c+cs  = non sorbed and sorbed concetrations coefficients
101
102 integer nMaxCcVectors ! Responsible for holding the max allowed Cc vectors. Since we require to aquire the solutions for the colloid component and then
103                       ! use them to get the solutions for the Virus Component. We can store the results that Colloid transportation produced and then use them
104                       ! to solve virus component. By doing so we avoid factorizing again and again the same matrix, without having to keep in memory the same time
105                       ! and the A matrix for the colloid component and the A matrix for the Virus component. Keep in mind that nMaxVectors require as well a lot of
106                       ! memory. By giving  larger nMaxCcVectors we gain speed but we need larger amount of memory. A valuearound 50 should be good.
107
108
109
110 ! End of <<<<<<<PARDISO>>>>>>>> variables
111
112
113 common/VarTrans1/Dcx,Dcy,Dcz,Lc,Lca,Rcdca,Rcadc
114 common/VarTrans1a/u
115 common/VarTrans3/dxx,dyy,dzz
116 common/vartrans6/Lv,Lva,Rvdva,Rvadv
117 common/vartrans7/Dvx,Dvy,Dvz
118
119 common/printMode/printSurMode,PrintSurXYZ,h
120 common/sourceCoord_C/lx0_C,ly0_C,lz0_C
121 common/sourceCoord_V/lx0_V,ly0_V,lz0_V
122 common/numOfsourcecells/sourCellNum
123 common/sourcePower_C/Q_C,Cm0_C,thh
124 common/sourcePower_V/Q_V,Cm0_V,nMaxCcVectors
125 common/Source_Type_trans/Source_Type_C,Source_Type_v
126 ! early Initializing
127
128 dxx=1.D0
129 dyy=1.D0
130 dzz=1.D0
131 r1=1. !
132 r2=1.
133 l1=1.
134 l2=1.
135 ! End of early  Initialization
136
137
138 ! calculating the size Vdim of the A vector
139
140 call matrixDimCal(vDim,nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2) ! call the specified subroutine to calculate all the possible non zero elements ,
141                                    ! in order to allocate the vectors needed .
142
143
144
145
146 ! <<<<<<<PARDISO>>>>>>>> allocation
147  allocate(a(vDim))
148  allocate(ja(vDim))
```

```
149  allocate(ia(2*nx*ny*nz+1))
150  allocate(perm(2*nx*ny*nz))
151  allocate(iparm(64))
152  allocate(pt(64))
153  !End of <<<<<<<PARDISO>>>>>>>> allocation
154
155  ! allocating the neccessary vectors
156  allocate(Ccnp1(2*nx*ny*nz))
157  allocate(Cvnp1(2*nx*ny*nz))
158  allocate(fCn(2*nx*ny*nz))
159  allocate(fCnC(2*nx*ny*nz))
160  allocate(AAVector(nx*ny*nz,2)) ! 2 vectors  matrix,because we need the  AA values on time t=tn time as well as t=tnp1 time
161                                 ! Its lenght though is half from other Vectors since its doesnt participate in the a matrix that solve 2 serious of equations the same time
162  allocate(BBVector(nx*ny*nz,2)) ! 2 vectors  matrix,because we need the  BB values on time t=tn time as well as t=tnp1 time
163                                 ! Its lenght though is half from other Vectors since its doesnt participate in the a matrix that solve 2 serious of equations the same time
164  allocate(AAVectors(nx*ny*nz,nMaxCcVectors)) ! 2 dimensional  matrix,because we need the  AA values on time t=tn time as well as t=tnp1 time  (n=1....nMaxCcVectors)
165                                 ! Its lenght though is half from other Vectors since its doesnt participate in the a matrix that solve 2 serious of equations the same time
166  allocate(BBVectors(nx*ny*nz,nMaxCcVectors)) ! 2 dimensional  matrix,because we need the  BB values on time t=tn time as well as t=tnp1 time  (n=1....nMaxCcVectors)
167                                 ! Its lenght though is half from other Vectors since its doesnt participate in the a matrix that solve 2 serious of equations the same time
168  allocate(CcVectors(2*nx*ny*nz,nMaxCcVectors)) !2 dimensional, First dimension=all colloids concetrations, Second dimension=diffent times (sequential times though)
169  allocate(Ccn(2*nx*ny*nz))
170  allocate(AAn(nx*ny*nz)) ! Allocate based on twice the total number of cells
171  allocate(BBn(nx*ny*nz)) ! Allocate based on twice the total number of cells
172 ! End of allocating the neccessary vectors
173
174
175
176
177
178
179    ! initialization
180 ii=0
181 iii=0
182 ir=0
183 nX0_C=0
184 ny0_C=0
185 nZ0_C=0
186 nX0_V=0
187 ny0_V=0
188 nZ0_V=0
189 dxx=1.D0
190 dyy=1.D0
191 dzz=1.D0
192 tc=0.D0
193 tmpTc=0.D0
194 t1=dt !one iteration at least will occur
195 rowPosSource=0
196 F_C=0.D0
197 F_V=0.D0
198
199 Dx=0
200 Dy=0
201 Dz=0
202 allCcConcPrinted=.False.
203 CcVectors=0
204
205 ! <<<<<<<PARDISO>>>>>>>> initialization
206 pt=0 !for more details check the ParDiso Manual page 2348.
207 maxfct=1 ! we store one matrix only
208 mnum=1
209 mtype=11 ! real and unsymmetric matrix
```

```
210 phase=12 ! phase the most important parameter ,determines the way " pardiso solver "
        works
211          ! 12  means "Analysis,numerical factorization" !for more details check the
        ParDiso Manual page 2350 (mkl 10.2.2.25)
212 n=2*nx*ny*nz ! number of total equations to be solved  simultaneously
213 A=0.D0
214 ia=0
215 ja=0
216 perm=0 ! current configuration doesnt use it ,and so we can set it to 0.
217 nrhs=1 ! we solve each time one side hand vector
218 iparm=0 ! we use the default factory configuration of ParDiso
219 !!! temp only for debugging
220 !iparm(8)=30 ! number of max iterative refinements, if iparm(8)=0 no refinement is
        happening
221 !!! temp only for debugging
222
223
224
225
226
227 msglvl=0 !1= we will  get  statical info on the screen every time Pardiso solver is
        called,0= we get nothing
228 error=1 ! error is an output variable and shows if the inversion succeeded. ! error =0
        success , else something went wrong
229 sourCellNum=1 ! in case we dont have elliptic or ellipsoid source ,sourCellNum will
        never be calculated by fcn_fector and so must have
230              ! value "one" in order not to affect the point source
231
232
233 Ccnp1=10.D-30 ! Appying initial conditions to the aquifer .All concetrations at t=t0=0
        Are zero at the begining of time
234 Ccn=10.D-30   ! Appying initial conditions to the aquifer .All concetrations at t=t0=0
        Are zero at the begining of time
235 AAn=0.D0      ! Appying initial conditions to the aquifer .All concetrations at t=t0=0
        Are zero at the begining of time
236 BBn=0.D0      ! Appying initial conditions to the aquifer .All concetrations at t=t0=0
        Are zero at the begining of time
237 Cvnp1=0.D0    ! All concetrations sorbed or not... blocks of virus with colloids are
        also zero
238 AAVectors=0.  ! Appying initial conditions to the aquifer .All concetrations at t=t0=0
        Are zero at the begining of time
239 AAVectors=0.  ! Appying initial conditions to the aquifer .All concetrations at t=t0=0
        Are zero at the begining of time
240 AAVector(:,:)=0.D0
241 BBVector(:,:)=0.D0
242 fCn=0.D0
243 fCnC=0.D0
244 ! end of initialization
245
246
247
248
249
250 ! in case of 32 bit its better!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!
251
252 !iparm(60)=1 ! If the memory of the system is not enough we will start using the hard
        disc to store the matrices
253 ! Current version of parDiso (v10.2.2.25) out core solving requires :
254 !mkl_num_threads=1  ! single thread
255 !iparm(3)=1         ! single thread
256 ! if iparm(60)=0 we can ignore the above 2 statements and let iparm(3) =
        mkl_get_max_threads() do the job
257 ! multithreading..... multicores...
258 ! in case of 32 bit its better!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!
259
260
261 ! in case of 64 bit its better!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        !!!!!!!!!!!!!
262 iparm(60)=0
```

```
mkl_num_threads= mkl_get_max_threads() ! numbers of processors, value of MKL_NUM_THREADS
iparm(3) = mkl_num_threads ! We will use all the available processors of the system
! multithreading..... multicores...




!calcualting the dimensions of the cells
dxx=lxx/nx
dyy=lyy/ny
dzz=lzz/nz


!calculating source position Colloid
nX0_C=int(lx0_C/dxx) +1 ! the integer part of the division is returned and the next closest cell is considered to be the point source
ny0_C=int(ly0_C/dyy) +1
nZ0_C=int(lz0_C/dzz) +1

if(nX0_C>nx) nX0_C=nx ! the position of the source cant be outside of the aquifer
if(ny0_C>ny) ny0_C=ny ! the position of the source cant be outside of the aquifer
if(nZ0_C>nz) nZ0_C=nz ! the position of the source cant be outside of the aquifer
if(nX0_C<1) nX0_C=1 ! the position of the source cant be outside of the aquifer
if(ny0_C<1) ny0_C=1 ! the position of the source cant be outside of the aquifer
if(nZ0_C<1) nZ0_C=1 ! the position of the source cant be outside of the aquifer

!calculating source position Virus
nX0_V=int(lx0_V/dxx) +1 ! the integer part of the division is returned and the next closest cell is considered to be the point source
ny0_V=int(ly0_V/dyy) +1
nZ0_V=int(lz0_V/dzz) +1

if(nX0_V>nx) nX0_V=nx ! the position of the source cant be outside of the aquifer
if(ny0_V>ny) ny0_V=ny ! the position of the source cant be outside of the aquifer
if(nZ0_V>nz) nZ0_V=nz ! the position of the source cant be outside of the aquifer
if(nX0_V<1) nX0_V=1 ! the position of the source cant be outside of the aquifer
if(ny0_V<1) ny0_V=1 ! the position of the source cant be outside of the aquifer
if(nZ0_V<1) nZ0_V=1 ! the position of the source cant be outside of the aquifer





!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Calculating the sourCellNum >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
write(*,*)" ******* Program started *******"
write(*,*)"Calculating  the number of cells the Source consists of"

call fCn_vector( nX,nY,nZ,nX0_C,ny0_C,nZ0_C,fcn,Ccnp1,F_C,dt)! the first time it is called  calculates the sourCellNum  (in case of elliptic or ellipsoid source)
                                                     ! if we have point source it wont be affected by the fCn_vector ,and will keep its initial value 1.
fcn=0.D0 ! we set again fCn to zero in order to be calculated properly (in case of elliptic or ellipsoid source)

!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of Calculating the sourCellNum >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>






!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  Calculating the sourcePower __F___>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 if (Source_Type_C ==2) then
 !F=G*W =Q*Cm0*W/dxdydz=Q*Cm0*Cm0/(th*dxdydz)   ! check eq 8 ,page 509 Analytical Solution for solte transport
 !in saturated porous media with semi-infinite or finite thickness 1998
 ! W=dirac(x-lx0_C) Dirac(y-ly0_C) dirac(z-lz0_C)/th
   F_C=Q_C*Cm0_C/thh

 !Calculating F based on the geometry of the source
```

```
 ! If we inject through a needle to a specific cell (of the aquifer)  mass then the F_C are as below. This specific cell wont have constant concetration through time
   F_C=F_C/(dxx*dyy*dzz*sourCellNum) ! Colloid case

 end if

 if (Source_Type_V ==2) then
  !F=G*W =Q*Cm0*W/dxdydz=Q*Cm0/(th*dxdydz)   ! check eq 8 ,page 509 Analytical Solution for solte transport
  !in saturated porous media with semi-infinite or finite thickness 1998
  ! W=dirac(x-lx0_C) Dirac(y-ly0_C) dirac(z-lz0_C)/th
   F_V=Q_V*Cm0_V/thh
  !Calculating F based on the geometry of the source
  ! If we inject through a needle to a specific cell (of the aquifer)  mass then the F_C are as below. This specific cell wont have constant concetration through time
                                  ! F=F/(dxx*dyy*dzz*sourCellNum) ,When we have 2 dimensional trasport F=F/(dxx*dyy*sourCellNum)!F=F/(tp_V*dxx*dyy**sourCellNum) also here 3 min is the normal time the source is active and transfer 1 ml of tracer with Q=2ml/hr.
   F_v=F_V/(dxx*dyy*dzz*sourCellNum) ! Virus case
 end if



 if (Source_Type_C ==1) then
 !If we inject the mass into the influent flux, and through this flux the mass is tranfered into the aquifer, then all the cells who are in direct touch with the flux have constant concetration
 ! keep in mind that we assume that the cells with direct touch to the infuent flux are only the ones which are considered to be source cells.
 ! the last assumption is usefull when a secondary flux (within the main flux) is also provided and so not all the cells at x=0, are in touch with contaminant flux

  F_C=(+Dcx/dxx**2.D0 +U/(2.D0*dxx))*Cm0_C
 end if

 if (Source_Type_v ==1) then
 !If we inject the mass into the influent flux, and through this flux the mass is tranfered into the aquifer, then all the cells who are in direct touch with the flux have constant concetration
 ! keep in mind that we assume that the cells with direct touch to the infuent flux are only the ones which are considered to be source cells.
 ! the last assumption is usefull when a secondary flux (within the main flux) is also provided and so not all the cells at x=0, are in touch with contaminant flux
  F_V=(+Dvx/dxx**2.D0 +U/(2.D0*dxx))*Cm0_V
 end if


!calculating F at the next time (tc+dt) ,Since the next printing time will be t+dt
if(tc+dt>tp_C)then
F_C=0.D0   ! Insert here the neccessary source loading, By default if we cross the active time of the source, mass supply rate becomes zero
end if
if(tc+dt>tp_V)then
F_V=0.D0   ! Insert here the neccessary source loading, By default if we cross the active time of the source, mass supply rate becomes zero
end if
!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of   Calculating the sourcePower __F___>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


 !___________________________Time will now start to count____________________________
  do


    !<<<<<<<<<<<<<<<<<<<<<<<<<Creating right hand vector colloid case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    ! creates new fCn vector based on the initial Ccnp1 cell concetration (Colloid case)
    call fCn_vector( nX,nY,nZ,nX0_C,ny0_C,nZ0_C,fcn,Ccnp1,F_C,dt)
    write(*,*)"  ******* Right hand vector ready !!!! ******* "
    !<<<<<<<<<<<<<<<<<<<<<<<<<<End of creating  right handvector colloid case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

```

```
!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Creating  A matrix for the colloid case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
!calling the neccessary subroutines to construct the coefficient matrix
write(*,*)"Creating the A coefficient matrix For the colloid case"
Dx=Dcx
Dy=Dcy
Dz=Dcz
l1=Lc
l2=Lca
r1=Rcdca
r2=Rcadc
call  aConstructor(nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2,dt,vDim,A,ia,ja)! A matrix Ready
write(*,*)" ******* The A coefficient matrix is ready ,Colloid case !!!  ******* "

!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of A matrix Creation for the colloid case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Factorizing the A matrix, Colloid case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
write(*,*)"  ******* Factorizing the A matrix ******* "
phase=12 ! 12  means "Analysis,numerical factorization  only"
CALL pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja,perm,nrhs,iparm,msglvl,fcn,Ccnp1,error)
                       ! analysis numerical factorization only ! we do not modify fcn or Ccnp1
write(*,*)" *******  Factorization ready !!!  ******* "
write(*,*)"___________________________________________"
write(*,*)"*************Bellow source values are being printed*****************"
!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Factorizing the A matrix, Colloid case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

phase=33  ! complete solve in one step
rowPosSource=rowPos(nX0_C,ny0_C,nZ0_C,ny,nz) ! locating the source cell




tmpTc=Tc
do ii=1,nMaxCcVectors ! We try to fill all the available CcVectors

    ! tc=0 the very first time this is read
    tc=tc+dt
    ! after the first time above argument is read ,time will be tc=dt
    ! we calculate concetrations at the end of the tc time
    CALL pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja,perm,nrhs,iparm,msglvl,fcn,Ccnp1,error)
    CcVectors(:,ii)=Ccnp1 ! Filling the CcVectors with the results of Ccnp1



    !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Applying Colloid Source Inactivity, Colloid case >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    if(tc+dt>tp_C)then
        F_C=0.D0  ! Insert here the neccessary source loading
    end if
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of  Applying  Source changes, Colloid case >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>




   !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<We calculate Cc concetrations and fill CcVecotrs (Colloids)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   ! creates new Ccnp1 based on previous fcn  (Colloid case)

   CALL  fCn_vector( nX,nY,nZ,nX0_C,ny0_C,nZ0_C,fcn,Ccnp1,F_C,dt)  ! creates new fCn vector based on the previous Ccnp1 results
   write(*,*)"At the end of time t=",tc,"Source concetration Colloids is  ",Ccnp1(rowPosSource)
```

```
      !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of Calculating and filling CcVectors(Colloids)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>




       !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Printing Colloids Concetrations  values >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
       call prntCcOut (Ccnp1,dxx,dyy,dzz,dt ,nx,ny,nz,tc,t1,allCcConcPrinted)
       !Notice that  prntCcOut Subroutine may have finished with value allCcConcPrinted==.True. or .False.
       ! If true then we printed all possible Cc concetrations and we also reached the end of input file, and also we reached maximum time t1.
       !were t1 is max time after we want concetrations either Colloid ones or Virus ones
       ! If false then we just found end of file and also we printed all colloids concetrations till time t=tc
       if(allCcConcPrinted==.True.) Then
        goto 430 ! we exit the loops for filling the CcVectors since we reached and of time
       end if

       !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< End of printing Colloids Concetrations values>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>





     IF (error .NE. 0) THEN  ! checking for errors
       WRITE(*,*) 'The following ERROR was detected: ', error
       write(*,*) "check the Pardiso manual for more info"
       pause
     END IF



 End do

  ii=ii-1 ! Very important. Do ii=1,nmax statement  allows ii to go beyond nmax (basicly ii=nmax+1), but when this happens the do loop exists... But still has gone ii=nmax+1.
          ! So we bring ii to its proper value






!_________________________________________________________________________________________________________________________________________________________
!_____________________________________________________________Virus ___________Case__________________________________________________________________________


  430 tc=tmpTc ! Restores tc to its original value (negates iterations made to fill CcVectors) so Cootransport (Virus) simulations can work properly below
     phase =-1 ! realease all internal memory the next time pardiso is called. Pardiso now must solve a new matrix.
     CALL pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja,perm,nrhs,iparm, msglvl,fcn,Ccnp1,error)


   write(*,*)"<<<<<<<<<Solving equations to calculate CcCvc and Cc*Cvc*, Plz wait!!! >>>>>>>>>>>>> "
 !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Calculates all AAValues And BBvalues for all times steps t=1,t=nmax (nmax=nmaxCcvectors)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
! calculates all AAVector and BBVector through iterations
! Based on initial conditions AAn,BBn,Ccn. Returns AAVectors,BBVectors
! first time its read AAn=BBn=Ccn=0. these are the initial conditions that we have assumed
    call AA_BB_Point_Over_Time_Calc(nx,ny,nz,Ccn,AAn,BBn,CcVectors,AAVectors,BBVectors,ii)
```

```
  write(*,*)"<<<<<<<<<<<<<<<<<<<< Concetrations  CcCvc and Cc*Cvc* were successfully calculated >>>>>>>>>>>>>>>>>>>>>>>>>>>>>> "

 !<<<<<<<<<<<<<<<<<<<<<<<<<End of Calculating all AAValues And BBvalues for all times steps t=1,t=nmax (nmax=nmaxCcvectors)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

                                                ! for all the available CcVectors
 !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<creates new CVnp1 based on previous fcnC (Virus case) >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

 !AAVector(:,1)=AAVector(:,2) ! Time t=tn+1 becomes t=tn ! the first time this is read AAVector(:,1)=AAVector(:,2)=0 since we initialized it this way
 !BBVector(:,1)=BBVector(:,2) ! Time t=tn+1 becomes t=tn ! the first time this is read AAVector(:,1)=AAVector(:,2)=0 since we initialized it this way
 !Ccn(:)=Ccnp1(:)! The first time this is "Read" Ccn=0, the other times Ccn(:)=Ccnp1(:) is automatically done by the system and so... we can leave it as a comment...
 !Ccnp1(:)=CcVectors(:,1)!And thus the first vector holds t=tn and the second holds t=tnp1
 !CALL AA_BB_Creator( Ccn,Ccnp1,AAVector,BBvector,Cvceq,Cvcaeq,nX,nY,nZ,dt)  ! creates the vectors AAVector(:,2) and BBVector(:,2),based on:
!Ccn, Ccnp1,AAVector(:,1)
 AAVector(:,1)=AAn(:)

 AAVector(:,2)=AAVectors(:,1)
 BBVector(:,1)=BBn(:)
 AAVector(:,2)=BBVectors(:,1)
 CALL  fCnC_vector( nX,nY,nZ,nX0_V,ny0_V,nZ0_V,fCnC,Cvnp1,AAVector,BBVector,F_V,dt)  ! creates new fCn vector based on the previous CVnp1 results and on

  !initializing AAn,BBn,and Ccn for the next  batch of iterations with lenght=ii. This batch will be activated after the virus transport simulation has finished
  !AAVector(:,1),BBVector(:,1),AAVector(:,2) and BBVector(:,2)
  AAn(:)=AAVectors(:,ii) ! the last value of of the vectors will the initial to calculate the next batch of concetrations
  BBn(:)=BBVectors(:,ii) ! the last value of of the vectors will the initial to calculate the next batch of concetrations
  Ccn(:)=CcVectors(:,ii) ! the last value of of the vectors will the initial to calculate the next batch of concetrations


 write(*,*)"*************bellow source values are being printed*****************"
 write(*,*)"At the end of time t=",tc,"Source concetration Virus case is  ",Cvnp1(rowPosSource)
 !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of creatings new CVnp1 based on previous fcnC  (Virus case)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>


 !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Creating  A matrix Creation for the Virus case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 !calling the neccessary subroutines to construct the coefficient matrix
 write(*,*)"Creating the A coefficient matrix For the Virus case"
 Dx=Dvx
 Dy=Dvy
 Dz=Dvz
 l1=Lv
 l2=Lva
 r1=Rvdva
 r2=Rvadv
 call  aConstructor(nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2,dt,vDim,A,ia,ja)! A matrix Ready
 write(*,*)" ******* The A coefficient matrix is ready !!!  ******* "
 !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of A matrix Creation for the Virus case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>




 !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Factorizing the A matrix, Virus case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 write(*,*)" Factorizing the A matrix, Virus Case"
 phase=12 ! 12  means "Analysis,numerical factorization  only"
 CALL pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja,perm,nrhs,iparm,msglvl, fcnC,Cvnp1,error)
 ! analysis numerical factorization only ! we do not modify fcn or Ccnp1
 write(*,*)" *******  Factorization ready !!!  ******* "
```

```fortran
!<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Factorizing the A matrix, Virus case>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>




phase=33  ! complete solve in one step

Do iii=1,ii ! For the available Ccvectors,We calculate Cv vectors and do the printing
      tc=tc+dt
      ! after the first time above argument is read ,time will be tc=dt
      ! we calculate concetrations at the end of the tc time
      CALL pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja,perm,nrhs,iparm, msglvl,fcnC,Cvnp1,error)  ! complete solve in one step

      write(*,*)"At the end of time t=",tc,"Source concetration of Virus  is  ",Cvnp1(rowPosSource)




      !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Applying Source Inactivity, Virus case >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      if(tc+dt>tp_V)then
          F_V=0.D0 ! Insert here the neccessary source loading
      end if
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of  Applying Source Changes, Virus case >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>




       !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Various print routines are activated>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
       Call  prntCvOut (Cvnp1,AAVector,dxx,dyy,dzz,dt ,nx,ny,nz,tc,t1, allCcConcPrinted) !Cvnp1 corresponds to Virus, Print one concetation at a time

       if (printSurMode==1)then
         Ccnp1(:)=Ccvectors(:,iii) ! we update Ccnp1 so we can print properly
         call surface_print(Ccnp1,dxx,dyy,dzz,nx,ny,nz,tc,PrintSurxyz,h) ! Prints all colloid surface at point h
       else if (printSurMode==2) then
         Ccnp1(:)=Ccvectors(:,iii) ! we update Ccnp1 so we can print properly
         call VisitLineData(Ccnp1,dxx,dyy,dzz,nx,ny,nz,tc)  ! at each time step we try to plot the volume  concetrations,Colloid, Ready for Visit software
       else if (printSurMode==3) then
         Ccnp1(:)=Ccvectors(:,iii) ! we update Ccnp1 so we can print properly
         call surface_print(Ccnp1,dxx,dyy,dzz,nx,ny,nz,tc,PrintSurxyz,h) ! Prints all colloid surface at point h
         call VisitLineData(Ccnp1,dxx,dyy,dzz,nx,ny,nz,tc)  ! at each time step we try to plot the volume  concetrations,Colloid, Ready for Visit software
       else if (printSurMode==11) then
         call surface_print(Cvnp1,dxx,dyy,dzz,nx,ny,nz,tc,PrintSurxyz,h) ! Prints all Virus surface at point h
       else if (printSurMode==22) then
         call VisitLineData(Cvnp1,dxx,dyy,dzz,nx,ny,nz,tc)  ! at each time step we try to plot the volume  concetrations,virus, Ready for Visit software
       else if (printSurMode==33) then
         call surface_print(Cvnp1,dxx,dyy,dzz,nx,ny,nz,tc,PrintSurxyz,h) ! Prints all Virus surface at point h
         call VisitLineData(Cvnp1,dxx,dyy,dzz,nx,ny,nz,tc)  ! at each time step we try to plot the volume  concetrations,Virus, Ready for Visit software
       end if

      if (tc==-1.333D0)then
          goto 332 ! end of program
       end if

     !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<creates new CVnp1 based on previous fcnC(Virus case) >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
        if(iii<ii)then ! needed because CcVectors(:,iii+1)  will create error for array bounds,if iii=ii we dont need to calculate next fcnc vectr, either because we reached end of time ,or because our nmaxCcVectors is reached and we run out of Cc vectors
            AAVector(:,1)=AAVector(:,2) ! Time t=tn+1 becomes t=tn
            BBVector(:,1)=BBVector(:,2) ! Time t=tn+1 becomes t=tn
            AAVector(:,2)=AAVectors(:,iii+1)! we exctract from the already calculated AAVectors the AAvector that the next  fCnC_vector will need
            BBVector(:,2)=BBVectors(:,iii+1)! we exctract from the already calculated BBVectors the BBvector that the next  fCnC_vector will need

            CALL  fCnC_vector( nX,nY,nZ,nX0_V,ny0_V,nZ0_V,fCnC,Cvnp1,AAVector,BBVector,F_V,dt) ! creates new fCn vector based on the previous Ccnp1 results

       !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<End of creatings new CVnp1 based on previous fcnC  (Virus case)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
          end if


      IF (error .NE. 0) THEN  ! checking for errors
        WRITE(*,*) 'The following ERROR was detected: ', error
        write(*,*) "check the Pardiso manual for more info"
        pause
      END IF





      ir=ir+1 ! iterations counter


      if(ir>1000000000)then ! protection from infinite loops
         goto 330
      end if




  end do
   !!!!!!!! need check maybe instead of ii we should have nMaxCcVectors
  ! The last time the above "do" ran, it didnt made the neccessaries changes because of the   if(iii<ii)then .... statement, so now we do it seperately
   AAn(:)=AAVectors(:,ii) ! the last value of of the vectors will the initial to calculate the next batch of concetrations
   BBn(:)=BBVectors(:,ii) ! the last value of of the vectors will the initial to calculate the next batch of concetrations
   Ccn(:)=CcVectors(:,ii) ! the last value of of the vectors will the initial to calculate the next batch of concetrations
   Ccnp1(:)=Ccn(:) ! we update Ccnp1 since we will now solve again the colloid transport and we need remind Ccnp1 where it had stopped
                  ! the above statement isnt really needed...Because either Ccnp1 isnt modified at all during computations or either is updated at every time step virus transport does
                  ! when the printSurMode's are checked


end do





332 write(*,*)"all calculations finished with success"
goto 331 ! protects from printing without reason the below error message
330 write(*,*)"Error max time steps reached"
    write(*,*)"Check the way time steps are added in main subroutine"
    write(*,*) "the system will now close"
    read(*,*)!slows down the system exit


331 end  subroutine
```

```
1  subroutine matrixDimCal(vDim,nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2)
2  !this subroutine calculates the size of the vector that will store  the coefficient matrix A
3  use  AA_BB_Solver, only: dt ! uses the module to share globally some intresting variables
4  implicit none
5
6  integer i0,j0,w0 ! i0,j0,w0 are the coordinates of the cell ,for which we want to create the  A
7               ! coeffiecients (equations (1) and (2) are being silently written for each cell ,check page 509).
8  integer nx,ny,nz  ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions.
9  !real*8  dt  ! dt= the  (time) distance between two sequential time moments t1,t2  =dt= t2-t1 ! defined in the  AA_BB_Solver module
10 real*8 A1,A2,A3,A4,A5,A6,A7,A8,A9,A10 ! coeffiecients
11 integer rowPoss  ! rowPoss  the distance from the first element (column) for a single row
12 integer ii,jj,ww ! ii,jj,ww=iterators
13 integer currentRow ! currentRow holds the number of the current row
14   !for which the coefficients are been written (equations (1) and (2) are being silently written for each cell ,check page 509).
15 logical BoundaryCell  !  will point if this is a boundary cell or not .  True means this is actually a boundary cell
16 logical flag3,flag4,flag5,flag6,flag7,flag8 !  flags will point out if the according coefficients should be added to the A matrix or not .
17                                               ! true means they should be added
18 integer vDim ! this will hold the lenght of the vector that will store  the coefficient matrix A
19 real*8 Dx,Dy,Dz      !Dx,Dy  hydrodynamic dispersion coefficient Will either hold Virus or Colloid component
20                      !Dz =Vertical hydronamic dispersion coefficient  Will either hold Virus or Colloid component
21 real*8 R2,L2 !r2= reverse rate coefficient ,l2=l*=decay rate of sorbed solute, Will either hold Virus or Colloid component
22 real*8 R1,L1 !r1=forward rate coefficient  ,l1=decay rate of liquid phase solute, Will either hold Virus or Colloid component
23 !initializing values in case something went wrong during reading
24 A1=0
25 A2=0
26 A3=0
27 A4=0
28 A5=0
29 A6=0
30 A7=0
31 A8=0
32 A9=0
33 A10=0
34 rowPoss=0
35 currentRow=0
36 BoundaryCell=.False.
37 flag3=.False.
38 flag4=.False.
39 flag5=.False.
40 flag6=.False.
41 flag7=.False.
42 flag8=.False.
43 vDim=0
44 ! end of initializations
45
46
47  !The way c and cs are dirtibuted for each row the same way the are distibuted for each column  (check rowPos function)
48
49      do ii=1,nx  ! iterating through all the cells of the aquifer
50         do jj=1,ny
51          do ww=1,nz
52
53
54            ! setting the current cell coordinates
55            i0=ii
56            j0=jj
57            w0=ww
58            currentRow=currentRow+1  ! increasing the row ....everytime the cell is
```

```
    changing
59
60            call isBoundaryCell (BoundaryCell,i0,j0,w0,nx,ny,nz,Dx,Dy,Dz,r1,r2,l1,l2,dt,
    A1,A2,A3,A4,A5,A6,A7,A8,flag3,flag4,flag5,flag6,flag7,flag8)
61            ! calls the respective subroutine to verify if cell is boundary or not
62
63            if ( BoundaryCell ==.False.) then
64            ! this means we can use the default coefficients
65
66
67             vDim=vDim+8  ! every time we find a usual cell we write 8  (A1-A8)
    coefficients to describe the C concetrations (C non sorbed concetrations)
68
69
70          else
71
72            ! we are on a boundary cell
73
74            vDim=vDim+2    ! space for coefficients (A1,A2).These are always written
    for any kind of cell (boundary or not)
75
76
77            if(flag3==.True.) then
78            vDim=vDim+1 ! space for A3 coefficient
79            end if
80
81            if(flag4==.True.) then
82            vDim=vDim+1 ! space for A4 coefficient
83            end if
84
85            if(flag5==.True.) then
86            vDim=vDim+1 ! space for A5 coefficient
87            end if
88
89            if(flag6==.True.) then
90            vDim=vDim+1 ! space for A6 coefficient
91            end if
92
93            if(flag7==.True.) then
94            vDim=vDim+1 ! space for A7 coefficient
95            end if
96
97            if(flag8==.True.) then
98            vDim=vDim+1 ! space for A8 coefficient
99            end if
100
101         end if
102
103
104
105       end do
106     end do
107   end do
108
109
110
111
112  !carefull  currentRow now  holds the last row that the previous iterations finished
    with  (C non sorbed concetration coefficient rows)
113
114    do ii=1,nx  ! iterating through all the cells of the aquifer (including the
    boundary cells)
115       do jj=1,ny  ! Cs sorbed equations  (eq 2 page 509) are based on the sdame cell
    without affecting the neighbouring cells ...
116        do ww=1,nz  ! so no attention should be given whether the  the cell is
    boundary or not  .
117
118                  ! Creating a coefficients for the Cs (sorbed) concetrations and
    adding them
119                  ! accordingly after C (non sorbed) rows.
120                  ! the cs rows are after the c rows  .
121                  !equation  (2) is being silently written for each cell ,check page
    509.
122
```

```
123             ! setting the current cell coordiantes
124             i0=ii
125             j0=jj
126             w0=ww
127             currentRow=currentRow+1  !carefull  currentRow  holds the last row that the↙
      previous iterations  (C rows (non sorbed rows)) finished with
128             vDim=vDim+1 ! space for A9 coefficient
129             vDim=vDim+1 ! space for A10 coefficient
130             end do
131          end do
132       end do
133
134
135
136
137
138 end subroutine
```

```
1
2
3
4  module prntOutPut
5  ! this module is responsible for Reading the rest of the input.txt and print to file the results
6  contains
7  subroutine   prntCcOut (Ccnp1,dxx,dyy,dzz,dt ,nx,ny,nz,tc,t1,allCcConcPrinted)
8  implicit none
9  real*8  x,y,z1 ! the coordinates of the point we want to get the concetration
10 real*8 Ccnp1(:)    ! will be holding all the  colloid   concetrations (Ccnp1 is a vector)
11 real*8 dxx,dyy,dzz,dt ! dxx=the lenght of  each cell the total aquifer is splitted to.
12                       ! dyy=the witdth of  each cell the total aquifer is splitted to.
13                       ! dzz=the height of  each sell the total aquifer is splitted to.
14                       ! dt= the  (time) distance between two sequential time moments t1 ,t2  =dt=t2-t1
15 integer nx,ny,nz  ! nX,nY,nZ the number of pieces the user wants to split the aquifer to,at  x,y,z directions
16 real*8 t1,tc  ! t1 will be the total time after we want to get the concetrations
17               !tc=current t1  =the time already passed since the beginning
18 integer nxi,nyi,nzi !are the coordinates of the point we cant to get the concetration (nxi...are expressed in pieces) after t1 time
19 integer row  ! "row"    will hold the row number ,of the point for which we calculate concetration,of B vector
20 integer num ! num will hold the number of the read line ,If num begins with the digit one "1" then we print colloid concetratiosn
21              ! If num begins with the digit "2" we print Virus Concetrations,
22              ! If num begins with the digit "3" we print Concetrations of Viruses attached onto Colloid
23              ! Else we should get error
24              ! check reading from files.
25 Character*11 StrNum ! Usefull string used to store the num value, as a string variable
26 integer ii,jj,ww ! iterators
27 integer ip ! ip is the position of the line the current concetration at t=t1 should be written at (temp.txt file)
28 integer ir ! total number of values read from input.txt
29 logical  allCcConcPrinted ! Indicates if all colloid concetrations were properly printed . If true then all colloid questions where answered
30                           ! and  prntOutCv subroutine ,if ready, can convert the temp. txt record file to print.txt formatted file
31 logical  FirstTimeReading !checks whether we have read from the input file again in the past
32
33 real*8 temp ,temp1,temp2,temp3,temp4 ! just for converting temp.txt to print.txt
34 real*8 aDt ! when we read for first time we want all times  from  0 to 3dt/2 to be printed and not only dt/2 to 3dt/2
35           ! adt helps this happen when printing conditions are caclulated
36 save  FirstTimeReading ! we want to remember the outcome of the last subroutine
37 save ir
38
39
40  !initializing values
41  x=0 ;y=0; z1=0; num=0;
42  nxi=0; nyi=0; nzi=0;
43  ii=0 ; jj=0; ww=0;
44  aDt=0.D0
45  row=0
46  allCcConcPrinted=.True.
47  ip=0
48  temp4=0.D0 ;temp1=0.D0 ;temp2=0.D0 ;temp3=0.D0 ;
49  StrNum="12345678" ! empty string
50  data FirstTimeReading/.True./ ! this is only read once per program run.
51  !end of initialization
52
53 if(FirstTimeReading==.True.) then
54  ir=0
55  adt=-dt/2.D0
56   !and temporary lines are added to the print.txt (based on the required lines of input.txt)
57    FirstTimeReading=.False. ! we have already read from this file
58 end if
59
60
```

```
61
62 rewind 111 ! we go at the beggining of our input file to read again ...Time now is tc=
       tc+dt
63
64 do ii=1,70
65 read(111,*)! we jump the first lines of the print txt. Contains only text
       spexifications
66 end do
67
68
69  9 read(111,*,end=12) num,t1,x,y,z1 ! if end of file is reached we go to end
70       ip=ip+1
71
72   if(t1<=0) then ! checking the time read.
73    write(*,*) " Error found in the input (t)"
74    write(*,*)" The time after which we cant to calculate the concetration, cant be
       negative or zero"
75    write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN
       "
76    read(*,*) ! slows down the system exit
77    stop
78   end if
79
80   if(tc<t1 .or.tc==dt)then
81     allCcConcPrinted=.false. ! it is the first time we read the file or we have more
       calculations to do
82   end if                    !before we declare that we have all the concetrations
       printed
83
84
85
86  if((tc -dt/2.D0 +aDt)<t1 .and. t1<=(tc+dt/2.D0) )then ! current tc approcahed best t1
87     allCcConcPrinted=.false. ! we have to notify the system that we will now print at
       least one concetration
88
89    nxi=int(x/(dxx))+1
90    nyi=int(y/(dyy))+1
91    nzi=int(z1/(dzz))+1
92
93    if(nxi>nx) nxi=nx ! the position of the source cant be outside of the aquifer
94    if(nyi>ny) nyi=ny ! the position of the source cant be outside of the aquifer
95    if(nzi>nz) nzi=nz ! the position of the source cant be outside of the aquifer
96    if(nxi<1) nxi=1 ! the position of the source cant be outside of the aquifer
97    if(nyi<1) nyi=1 ! the position of the source cant be outside of the aquifer
98    if(nzi<1) nzi=1 ! the position of the source cant be outside of the aquifer
99
100
101
102      row=0 ! initializes row counter, sequential reads make it imperative
103
104     do ii=1,nx
105         do jj=1,ny
106          do ww=1,nz
107
108             row=row+1 ! row counter
109              if (ii==nxi.and.jj==nyi.and.ww==nzi) then
110               if (row==0) row=1 ! in case we try to print the cell C(1,1,1) this would
       lead Ccnp1 to Ccnp1(0) which is not allowed
111                 goto 11 ! position located, we end the function
112               end if
113
114          end do
115         end do
116      end do
117
118      ! We will try to indetify the Concetration that needs to be printed based on the
       num value
119      ! num will hold the number of the read line ,If num begins with the digit one "1"
       then we print colloid concetratiosn
120             ! If num begins with the digit "2" we print Virus Concetrations,
121             ! If num begins with the digit "3" we print Concetrations of Viruses
       attached onto Colloid
122             ! Else we should get error
123      11 write(StrNum,'(i10)')num ! we convert the integer num into string so we can
```

```
    check its digits one by one
124         StrNum=trim(adjustl(StrNum)) ! Removes preceding blanks (from left)
125
126         if (Scan(StrNum,"1")==1) then     ! Digit one "1" was found to be first on num
    integer
127          temp=Ccnp1(row)   ! We print colloid concetrations
128         else if (Scan(StrNum,"2")==1)then ! Digit one "2" was found to be first on num
    integer
129           goto 9 ! we do nothing since this subroutine prints Colloid values only
130                  ! Check the beggining of this routine for the meanings of Num

131         else if (Scan(StrNum,"3")==1)then ! Digit one "3" was found to be first on num
    integer
132           goto 9 ! we do nothing since this subroutine prints Colloid values only
133                  ! Check the beggining of this routine for the meanings of Num
134         else
135           write(*,*) "The system failed to understand the kind of Concetration you want
     to print"
136           write(*,*) "Based on the input data in the input.txt"
137           write(*,*) "Please check again the first number of every row, for the rows
    after the "
138           write(*,*) "parameter input section. The numbers should start from the digits
     1,2 or 3 only"
139           write(*,*) "For more check manual: "
140           write(*,*) "Critical error found the system will now exit"
141           pause
142           stop
143         end if
144
145
146
147
148
149                    ! we print current time
150     write(555,33,REC=ip)num,tc,(nxi-1)*dxx+dxx/2.D0 ,(nyi-1)*dyy+dyy/2.D0 ,(nzi-1)*dzz
    +dzz/2.D0,temp ! concetration is pointed at the middle of each printing cell
151     33 format(I5,"  ",Es14.5,"  ",En14.5 ,"   ",En14.5,"   ",En14.5 ,"  ",Es16.7)
152     ir=ir+1 ! we count the values printed to the temp.txt
153
154   end if
155
156   goto 9 ! again and again we read till we find end
157
158
159       !Notice that  prntCcOut Subroutine may have finished with value  allCcConcPrinted=
    =.True. or .False.
160       ! If true then we printed all possible Cc concetrations and we also reached the
    end of input file, and also we reached maximum time t1.
161       !were t1 is max time after we want concetrations either Colloid ones or Virus ones
162       ! If false then we just found end of file and also we printed all colloids
    concetrations till time t=tc
163   12 end subroutine
164
165
166
167
168
169
170
171   subroutine   prntCvOut (Cvnp1,AAVector,dxx,dyy,dzz,dt ,nx,ny,nz,tc,t1,
    allCcConcPrinted)
172 implicit none
173 real*8  x,y,z1 ! the coordinates of the point we want to get the concetration
174 real*8 Cvnp1(:)    ! will be holding all the  virus     concetrations (Cvnp1 is a
    vector)
175 real*8 AAvector(:,:) ! will be holding all the  viruses attached on Colloids
    concetrations (Ccnp1 is a vector)
176 real*8 dxx,dyy,dzz,dt ! dxx=the lenght of  each cell the total aquifer is splitted to.
177                        ! dyy=the witdth of  each cell the total aquifer is splitted to.
178                        ! dzz=the height of  each sell the total aquifer is splitted to.
179                        ! dt= the  (time) distance between two sequential time moments t1
    ,t2  =dt=t2-t1
180 integer nx,ny,nz  ! nX,nY,nZ the number of pieces the user wants to split the aquifer
    to,at  x,y,z directions
```

```
181 real*8 t1,tc  ! t1 will be the total time after we want to get the concetrations
182               !tc=current t1  =the time already passed since the beginning
183 integer nxi,nyi,nzi !are the coordinates of the point we cant to get the concetration
    (nxi...are expressed in pieces) after t1 time
184 integer row  ! "row"    will hold the row number ,of the point for which we calculate
    concetration,of B vector
185 integer num ! num will hold the number of the read line ,If num begins with the digit
    one "1" then we print colloid concetratiosn
186               ! If num begins with the digit "2" we print Virus Concetrations,
187               ! If num begins with the digit "3" we print Concetrations of Viruses
    attached onto Colloid
188               ! Else we should get error
189               ! check reading from files.
190 Character*11 StrNum ! Usefull string used to store the num value, as a string variable
191 integer ii,jj,ww ! iterators
192 integer ip ! ip is the position of the line the current concetration at t=t1 should be
    written at (temp.txt file)
193 integer ir ! total number of values read from input.txt
194 logical  allCvConcPrinted ! Indicates if all virus concetrations were properly printed
    . If true then all virus questions where answered
195                           ! and  prntOutCv subroutine ,if allCcConcPrinted= .True. as
    well, can convert the temp.txt record file to print.txt formatted file
196 logical  allCcConcPrinted ! Indicates if all colloid concetrations were properly
    printed . If true then all colloid questions where answered
197                           ! and  prntOutCv subroutine ,if ready, can convert the temp.
    txt record file to print.txt formatted file
198 logical  FirstTimeReading !checks whether we have read from the input file again in the
     past
199 real*8 temp ,temp1,temp2,temp3,temp4 ! just for converting temp.txt to print.txt
200 real*8 aDt ! when we read for first time we want all times  from  0 to 3dt/2 to be
    printed and not only dt/2 to 3dt/2
201          ! adt helps this happen when printing conditions are caclulated
202  save  FirstTimeReading ! we want to remember the outcome of the last subroutine
203  save ir
204
205
206  !initializing values
207  x=0 ;y=0; z1=0; num=0;
208  nxi=0; nyi=0; nzi=0;
209  ii=0 ; jj=0; ww=0;
210  aDt=0.D0
211  row=0
212  allCvConcPrinted=.True.
213  ip=0
214  temp4=0.D0 ;temp1=0.D0 ;temp2=0.D0 ;temp3=0.D0 ;
215  StrNum="               " ! empty string
216  data FirstTimeReading/.True./ ! this is only read once per program run.
217  !end of initialization
218
219 if(FirstTimeReading==.True.) then
220  ir=0
221  adt=-dt/2.D0
222   !and temporary lines are added to the print.txt (based on the required lines of input
      .txt)
223    FirstTimeReading=.False. ! we have already read from this file
224 end if
225
226
227
228 rewind 111 ! we go at the beggining of our input file to read again ...Time now is tc=
    tc+dt
229
230 do ii=1,70
231 read(111,*)! we jump the first lines of the print txt. Contains only text
    spexifications
232 end do
233
234
235  9 read(111,*,end=12) num,t1,x,y,z1 ! if end of file is reached we go to end
236       ip=ip+1
237
238   if(t1<=0) then ! checking the time read.
239    write(*,*) " Error found in the input (t)"
240    write(*,*)" The time after which we cant to calculate the concetration, cant be
```

```
     negative or zero"
241    write(*,*)"Please change the input file and try again ,THE SYSTEM WILL NOW SHUT DOWN
        "
242    read(*,*) ! slows down the system exit
243    stop
244   end if
245
246   if(tc<t1 .or.tc==dt)then
247     allCvConcPrinted=.false. ! it is the first time we read the file or we have more
        calculations to do
248   end if                    !before we declare that we have all the concetrations
        printed
249
250
251
252  if((tc -dt/2.D0 +aDt)<t1 .and. t1<=(tc+dt/2.D0) )then ! current tc approched best t1
253     allCvConcPrinted=.false. ! we have to notify the system that we will now print at
        least one concetration
254
255    nxi=int(x/(dxx))+1
256    nyi=int(y/(dyy))+1
257    nzi=int(z1/(dzz))+1
258
259    if(nxi>nx) nxi=nx ! the position of the source cant be outside of the aquifer
260    if(nyi>ny) nyi=ny ! the position of the source cant be outside of the aquifer
261    if(nzi>nz) nzi=nz ! the position of the source cant be outside of the aquifer
262    if(nxi<1) nxi=1 ! the position of the source cant be outside of the aquifer
263    if(nyi<1) nyi=1 ! the position of the source cant be outside of the aquifer
264    if(nzi<1) nzi=1 ! the position of the source cant be outside of the aquifer
265
266
267
268      row=0 ! initializes row counter, sequential reads make it imperative
269
270     do ii=1,nx
271         do jj=1,ny
272          do ww=1,nz
273
274             row=row+1 ! row counter
275              if (ii==nxi.and.jj==nyi.and.ww==nzi) then
276               if (row==0) row=1 ! in case we try to print the cell C(1,1,1) this would
        lead Ccnp1 to Ccnp1(0) which is not allowed
277                goto 11 ! position located, we end the function
278              end if
279
280          end do
281         end do
282       end do
283
284       ! We will try to indetify the Concetration that needs to be printed based on the
        num value
285       ! num will hold the number of the read line ,If num begins with the digit one "1"
        then we print colloid concetratiosn
286             ! If num begins with the digit "2" we print Virus Concetrations,
287             ! If num begins with the digit "3" we print Concetrations of Viruses
        attached onto Colloid
288             ! Else we should get error
289       11 write(StrNum,'(i10)')num ! we convert the integer num into string so we can
        check its digits one by one
290         StrNum=trim(adjustl(StrNum)) ! Removes preceding blanks
291
292         if (Scan(StrNum,"1")==1) then     ! Digit one "1" was found to be first on num
        integer
293           goto 9   ! we do nothing since this subroutine prints Virus values only
294                    ! Check the beggining of this routine for the meanings of Num
295         else if (Scan(StrNum,"2")==1)then ! Digit one "2" was found to be first on num
        integer
296           temp=Cvnp1(row)   ! We print Virus concetrations
297         else if (Scan(StrNum,"3")==1)then ! Digit one "3" was found to be first on num
        integer
298           temp=AAVector(row,2)! We print Viruses attached on colloids concetrations
299         else
300           write(*,*) "The system failed to understand the kind of Concetration you want
        to print"
```

```
301           write(*,*) "Based on the input data in the input.txt"
302           write(*,*) "Please check again the first number of every row, for the rows
      after the "
303           write(*,*) "parameter input section. The numbers should start from the digits
       1,2 or 3 only"
304           write(*,*) "For more check manual: "
305           write(*,*) "Critical error found the system will now exit"
306           pause
307           stop
308         end if
309
310
311
312
313
314                     ! we print current time
315     write(555,33,REC=ip)num,tc,(nxi-1)*dxx+dxx/2.D0 ,(nyi-1)*dyy+dyy/2.D0 ,(nzi-1)*dzz
      +dzz/2.D0,temp ! concetration is pointed at the middle of each printing cell
316     33 format(I5,"  ",Es14.5,"  ",En14.5 ,"   ",En14.5,"   ",En14.5 ,"  ",Es16.7)
317     ir=ir+1 ! we count the values printed to the temp.txt
318
319   end if
320
321   goto 9 ! again and again we read till we find end
322
323
324   12 if (allCvConcPrinted==.True. .And. allCcConcPrinted==.True.)then  ! Both virus and
      Colloid concetrations must be already printed
325                         ! we will transfer our outputs from the temp.txt to the print.
      txt (from a record file... to a formatted file)
326
327           do  ir=1,ip
328             read(555,33,REC=ir)  num,tc,dxx,dyy , dzz ,temp
329             write(333,33)        num,tc,dxx,dyy , dzz ,temp   ! concetration is pointed
      at the middle of each printing cell  !num,temp ,temp1,temp2,temp3,temp4
330           end do
331           write(*,*)"conversion finished"
332
333         tc=-1.333D0 ! a negative value is given in order the main program not to call
      again the  prntCcOut or prntCvOut subroutine
334               ! all times t1  already have their concetrations printed
335
336    end if
337
338
339
340    end subroutine
341
342
343
344
345
346
347   subroutine surface_print(b,dxx,dyy,dzz,nx,ny,nz,tc,PrintSurxyz,h)
348   implicit none
349    !responsible subroutine for printing whole  the surface of the aquifer at a
        particular time
350    real*8 b(:) ! will be holding all the  concetrations (b is a vector)
351    real*8 dxx,dyy,dzz  ! dxx=the lenght of  each cell the total aquifer is splitted to.
352                        ! dyy=the witdth of  each cell the total aquifer is splitted to.
353                        ! dzz=the height of  each sell the total aquifer is splitted to.
354    integer nx,ny,nz  ! nX,nY,nZ the number of pieces the user wants to split the aquifer
       to,at  x,y,z directions
355    real*8 tc !tc=current t1  =the time already pass since the beginning
356    integer PrintSurxyz !indicator  ,shows which surface we want to print :  PrintSurxyz=
       1 => yz is printed,PrintSurxyz=2 => xz is printed ,PrintSurxyz=3 => xy is printed
357    real*8 h ! surface position (distance from  (x,y,z)=(0,0,0))
358    integer row  ! "row"    will hold the row number ,of the point for which we calculate
       concetration,of B vector
359    integer i,ii,jj,ww ! iterators
360    integer nh0 ! h (surface position) measured with cell units
361    integer nxyz,nnxyz ! usefull iterators for the printing mechanism
362    real*8 ,Allocatable::cSurf(:,:)
363
```

```
364     !initializing values
365     ii=0 ; jj=0; ww=0
366     row=0 ; nh0=1; i=0
367     nxyz=0 ; nnxyz=0
368     !end of initialization
369
370
371      write(444,*) " current results are based on time t =",tc,"(days)"
372
373      if (PrintSurxyz==1) then
374
375        allocate(cSurf(ny,nz))  ! allocates the neccessary matrix based on which surface we want to print, surface yz
376        cSurf=0 ! initializes the allocated matrix
377        row=0 ! initializes row counter, sequential reads make it imperative
378        nh0=int(h/(dxx))+1 ! converting h to cell units
379        if(nh0>nx) nh0=nx ! the position of the source cant be outside of the aquifer
380        if(nh0<1) nh0=1 ! the position of the source cant be outside of the aquifer
381
382
383              do ii=1,nx  ! we convert the b vector to a cSurf(ni,nii) (i ,ii could be any of x,y,z ) matrix,which contains all of the surface concetrations
384                 do jj=1,ny
385                   do ww=1,nz
386
387                         row=row+1
388                          if(ii==nh0) then !  current cell is part of the printing surface
389                           cSurf(jj,ww)=b(row)
390                          end if
391
392                    end do
393                  end do
394                end do
395
396        nxyz=nz
397        nnxyz=ny
398        write(444,*)"____________Y---->>>(cm)__________________Z vectical (cm)________"
399      end if
400
401
402
403
404     if (PrintSurxyz==2)then
405
406           allocate(cSurf(nx,nz))  ! allocates the neccessary matrix based on which surface we want to print, surface xz
407           cSurf=0 ! initializes the allocated matrix
408           row=0
409           nh0=int(h/(dyy))+1 ! converting h to cell unitis
410           if(nh0>ny) nh0=ny ! the position of the source cant be outside of the aquifer
411           if(nh0<1) nh0=1 ! the position of the source cant be outside of the aquifer
412
413
414              do ii=1,nx  ! we convert the b vector to a cSurf(ni,nii) (i ,ii could be any of x,y,z ) matrix,which contains all of the surface concetrations
415                 do jj=1,ny
416                   do ww=1,nz
417
418
419                         row=row+1
420                         if(jj==nh0) then !  current cell is part of the printing surface
421                           cSurf(ii,ww)=b(row)
422                         end if
423
424                    end do
425                  end do
426                end do
427
428        nxyz=nz
429        nnxyz=nx
430        write(444,*)"____________X Horizontal ---->>>(cm)_____________z vectical (cm)_________"
```

```
431     end if
432
433
434   if (PrintSurxyz==3) then
435
436           allocate(cSurf(nx,ny))  ! allocates the neccessary matrix based on which surface we want to print, surface xy
437           cSurf=0 ! initializes the allocated matrix
438           row =0
439           nh0=int(h/(dzz))+1 ! converting h to cell units
440           if(nh0>nz) nh0=nz ! the position of the source cant be outside of the aquifer
441           if(nh0<1) nh0=1 ! the position of the source cant be outside of the aquifer
442
443             do ii=1,nx  ! we convert the b vector to a cSurf(ni,nii) (i ,ii could be any of x,y,z ) matrix,which contains all of the surface concetrations
444                do jj=1,ny
445                  do ww=1,nz
446
447
448                           row=row+1
449                           if(ww==nh0) then !  current cell is part of the printing surface
450                             cSurf(ii,jj)=b(row)
451                           end if
452
453                  end do
454                end do
455              end do
456     nxyz=ny
457     nnxyz=nx
458     write(444,*)"____________X Horizontal ---->>>(cm)_____________Y vectical (cm)_________"
459   end if
460
461
462
463   !cSurf now contains the concetrations of the particular surface ,based on   height h (or nh0 in cell units)
464   !printing cSurf
465
466   do ii=1,nxyz
467    write(444, 122) (cSurf(i,ii),i=1,nnxyz )
468    write(444,*)
469   end do
470 122 format(  (D11.5,", ")$  )
471
472
473    write(222,*) ! we jump a row
474    write(222,*)"_____________________________________________________________________"
475    write(222,*)"_____________________________________________________________________"
476    write(222,*)" After t=",tc, "(Days) have passed"
477
478    ! printing surfac with Coordinates
479   if (PrintSurxyz==1)then ! we print yz plane coordinates per column
480     write(222,*)"Y(cm)_____________________Z(cm)_______"
481     do jj=1,ny
482        do ww=1,nz
483           write(222,*)(jj-1)*dyy+dyy/2.D0 ,(ww-1)*dzz +dzz/2.D0, cSurf(jj,ww) ! concetration is pointed at the center of each cell
484        end do
485     end do
486
487   else if (PrintSurxyz==2)then ! we print xz plane coordinates per column
488     write(222,*)"X(cm)_____________________Z(cm)_______"
489     do ii=1,nx
490        do ww=1,nz
491          write(222,*)(ii-1)*dxx+dxx/2.D0 ,(ww-1)*dzz+dzz/2.D0 ,cSurf(ii,ww) ! concetration is pointed at the center of each cell
492        end do
493     end do
494
495   else if (PrintSurxyz==3)then  ! we print xy plane coordinates per column
496    write(222,*)"X(cm)_____________________Y(cm)_______"
497    do ii=1,nx
```

```
498         do jj=1,ny
499           write(222,*)(ii-1)*dxx+dxx/2.D0,(jj-1)*dyy+dyy/2.D0 ,cSurf(ii,jj) !
        concetration is pointed at the center of each cell
500         end do
501       end do
502
503
504   else
505       write(*,*)"Critical error the system will now stop,we cant print the neccessary
         plane with the neccessary coordianates"
506       write(*,*)"please check the input file about printing and try again"
507   stop
508   end if
509
510
511
512
513    end subroutine
514
515
516
517
518
519
520
521
522
523    end module prntOutPut
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542    module visitOutPut
543    contains
544
545
546    subroutine VisitLineData(b,dxx,dyy,dzz,nx,ny,nz,tc)
547    implicit none
548     !responsible subroutine for printing whole the volume of the aquifer and at different
          files
549     real*8 b(:) ! will be holding all the  concetrations (b is a vector)
550
551     integer nx,ny,nz  ! nX,nY,nZ the number of pieces the user wants to split the aquifer
          to,at  x,y,z directions
552     real*8 tc !tc=current t1  =the time already pass since the beginning
553     integer row  ! "row"    will hold the row number ,of the point for which we calculate
          concetration,of B vector
554     real*8 dxx,dyy,dzz  ! dxx=the lenght of  each cell the total aquifer is splitted to.
555                         ! dyy=the witdth of  each cell the total aquifer is splitted to.
556                         ! dzz=the height of  each sell the total aquifer is splitted to.

557     integer i,ii,jj,ww ! iterators ! i specifies the time at which we print. Only
          intgeres for now
558     CHARACTER fileName*35 ! filename the name of the file to be written
559      save i ! we will remember the previous call of the subroutine
560
561      data i/0/
562          !initializing values
563      ii=0
564      jj=0
```

```
565     ww=0
566     row=0
567     !end of initialization
568
569
570     if(i==0) then
571     ! we open a file the first time this subroutine is called in order to create a .
         visit file
572     !.Visit file describes Visit software the names of   files that   part of a greater
         database
573       open(888,File="multiplePrints\visitDatabase.visit",Form="Formatted")
574     end if
575
576
577
578      i=i+1
579   ! if( tc-Int(tc)==0)then
580
581       call fileNameCreator(fileName,i,tc) ! grabs the name of file to be printed
582
583        open(778,File=fileName,Form="Formatted") ! opens the file to be written att
584        write(778,*)" x y z  C " ! needed filespecification to work C(m/v), x,y,z=
        (Lenght)
585             do ii=1,nx  ! we convert the b vector to a cSurf(ni,nii) (i ,ii could be any
        of x,y,z ) matrix,which contains all of the surface concetrations
586               do jj=1,ny
587                 do ww=1,nz
588
589                         row=row+1
590
591                         write(778,44)(ii-1)*dxx+dxx/2.D0 ,(jj-1)*dyy+dyy/2.D0 ,(ww-1)*
        dzz +dzz/2.D0,b(row)
592                          44 format(En14.5 ,"   ",En14.5,"   ",En14.5 ,"  ",Es16.7)
593
594
595               end do
596             end do
597           end do
598     ! we closed the written file
599
600     close (778)
601
602   !else
603    !i=i-1 ! we dont need to print yet we revert the change on i
604   !end if
605
606
607 end subroutine
608
609
610
611
612
613
614 subroutine fileNameCreator(fileName,i,tc)
615 implicit none
616
617  CHARACTER fileName*35 ! filename the name of the file to be written
618  character  number*7
619  character baseFileName*21
620  PARAMETER (baseFileName = "multiplePrints\")
621  integer i ! indicates the number of files previously printed
622  real*8 tc ! current time
623  !write(number,100) tc
624  !number=adjustl(number) ! moves to right and removes leading blacnks-adding them to
       the end
625  !i=len_trim(number) ! Retuns the lenght of the string without the ending blanks
626  !fileName=baseFileName//number(1:i)//".dat"
627  !write(888,*)"print t= "//number(1:i)//".dat" ! we fill the visit database file with
       the names of the files that we create .
628   write(number,"(i6)") i
629   number=adjustl(number)
630   fileName=baseFileName//number
631
```

```
632    write(888,*)"print"//number//".dat" ! we fill the visit database file with the names
         of the files that we create .
633
634                          ! this is needed in order to make visit read them as time
         varying database
635     100 FORMAT(   F8.2 )! converts the blank spaces before the 1 to zeros ex. "  1"="001"

636   end subroutine
637
638   end module  visitOutPut
639
```

```
1
2  Module AA_BB_Solver
3
4
5  double Precision , allocatable :: CcvectorT(:) ! Vector that will hold Cc and Cca concetrations for a specific x,y,z cell over time...t=tn+1.....+n  (were n=nmax)
6                                    ! The first nmax rows are Cc concetrations and the rest are Cca concetrations
7  double Precision Cc0,Cca0 ! Cc0 and Cca0 are the initial conditions for the case of Solute colloids and Sorbed on the solid matrix colloids
8  double Precision   Cvceq,Cvcaeq !Cveq max holds the concetration of viruses attached onto clay collids  at equilibrium (M viruses)/(M clay)
9                     !Cvcaeq max holds the concetration of viruses attached onto clay collids already attached onto glass beads at equilibrium (M viruses)/(M clay)
10 integer nmax ! nmax holds the number of time steps we will perform. Based on that nmax determines the size of the jacoby matrix (nmax could be set= nMaxCcVectors)
11 double Precision    dt          !dt= the  (time) distance between two sequential time moments t1,t2  =dt=t2-t1
12 logical   negSlope  ! indicates whether we have negative slope or not on Cc values
13
14
15  !CcVectorT,Cc0,Cca0,Cvceq,Cvcaeq,dt,nmax variables are made common to all routines inside AA_BB_Solver
16
17 Contains
18
19 subroutine AA_BB_Point_Over_Time_Calc(nx,ny,nz,Ccn,AAn,BBn,CcVectors,AAVectors, BBVectors,nMaxCcVectorsorII)
20 IMPLICIT NONE
21       ! The AA_BB_Point_Over_Time_Calc sub calculates for a specific point the AA=CcCcv and BB=Cc*Ccv* Concetrations over time
22
23
24
25
26 double Precision AA0,BB0  ! AA0 and BB0 are initial conditions. Each cell has its own AA0=CcCvc,BB0=Cc*Cvc*
27
28 integer i,j,k,ir,it !iterators,it= time iterator
29 integer nx,ny,nz ! nX,nY,nZ the number of pieces the user wants to split the aquifer to ,at  x,y,z directions.
30 integer nMaxCcVectorsorII ! Responsible for holding the max allowed Cc vectors. Since we require to aquire the solutions for the colloid component and then
31                    ! use them to get the solutions for the Virus Component. We can store the results that Colloid transportation produced and then use them
32                    ! to solve virus component. By doing so we avoid factorizing again and again the same matrix, without having to keep in memory the same time
33                    ! and the A matrix for the colloid component and the A matrix for the Virus component. Keep in mind that nMaxVectors require as well a lot of
34                    ! memory. By giving  larger nMaxCcVectors we gain speed but we need larger amount of memory. A valuearound 50 should be good.
35
36 double Precision , allocatable :: CcVectors(:,:) ! Responsible for holding  Cc solutions ,for successive times . Since we require to aquire the solutions for the colloid component and then
37                    ! use them to get the solutions for the Virus Component. We can store the results that Colloid transportation produced and then use them
38                    ! to solve virus component. By doing so we avoid factorizing again and again the same matrix,and also we avoid having to keep  in memory the same time
39                    ! and the A matrix for the colloid component and the A matrix for the Virus component. Keep in mind that nMaxVectors require as well a lot of
40                    ! memory. By giving  larger nMaxCcVectors we gain speed but we need larger amount of memory. A valuearound 100 should be good.
41 double Precision , allocatable :: Ccn(:)   ! Ccn matrix will hold all the unknown concetrations (sorbed and non sorbed ) at time tn    (Colloid case)
42                                 ! Basicly Ccn will hold the initial coniditions so AA_BB_Point_Over_Time_Calc can calculate
43                                 ! values on t=tnp1 (ex.t1) based on values t=tn (ex. t0)
44 double Precision , allocatable::AAn(:),BBn(:)! AAn vector will hold all (all cells) initial known concetrations (AAn=CcCvc ) at time tn
45                                 ! BBn vector will hold all (all cells) initial known concetrations (BBn=Cc*Cvc* ) at time tn,
```

```
46                                  ! So we can calculate values on t=tnp1 (ex.t1) based
    on values t=tn (ex. t0)
47
48 double Precision , allocatable :: AAVectors(:,:) ! Vectors holding the solutions of the
    8, 13 eqs . Where aa=CcCvc and BB=Cc*Cvc* ,
49                                      ! 2 dimensional  matrix,because we need the  AA
    values on time t=tn time as well as t=tn+1......n
50 double Precision , allocatable :: BBVectors(:,:) ! Vector holding the solutions of the 8
    , 13 eqs . Where aa=CcCvc and BB=Cc*Cvc*
51                                      ! 2 dimensional  matrix,because we need the  BB
    values on time t=tn time as well as t=tn+1......n
52                                      ! Paper Colloid-Facilitated virus transport in
    saturated porus media . Vasiliki i.Syngouna Constantinos V.Chrysicopoulos


53
54 double Precision CcAver,CcaAver     ! because Cc and Cca change with time, we intepolate
     the closest Cc values and Cca Values to get CcAver and CcaAver.
55 double Precision tmpCvceq,tmpCvcaeq ! tmp variables holding the Values Cvceq and Cvcaeq
56 double Precision  slopeAA,slopeBB ! Slopes of AA1 and BB1 graphs, they are checked and
    indicate if we have reached results that dont change significally over time
57 double Precision slopeCc,slopeCca   ! Slopes of Cc and Cca  graphs, they are checked and
     indicate if we have reached results that dont change significally over time
58 !_____________________________________________________________________________________
    _____________________________________
59 !<<<<<<<<<<<<<<<<<<<<<<Defining valiables for the INTEL DODESOL
    approach>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
60 INTEGER ierr
61    INTEGER NEQ
62 ! It is higly recommended to declare ipar array of size 128
63 ! for compatibility with future versions of ODE solvers
64       INTEGER kd(2), ipar(128)
65       double Precision T,h, t_end,  hm, ep, tr
66 ! As ODE system has size n=2, than the size of dpar array is equal to
67 ! max{13*n,(7+2*n)*n}=max{26,22}=26. More details on dpar array can be
68 ! found in the Manual
69       double Precision y(2), dpar(26) ! dpar(26) normally for a 2 equations
70       EXTERNAL FEX, JDUM
71
72
73 !<<<<<<<<<<<<<<<<<<<<End of Defining valiables for the INTEL DODESOL
    approach>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
74
75 !_____________________________________________________________________________________
    _____________________________________
76
77
78
79
80
81
82
83  ! initializing general variables
84  nmax=nMAxCcvectorsorII ! nmax initially will be equal to nmaxCcvectors or to II which
    ever is .... but there is a change the give Ccvectors till tn+1 maybe
85                     ! not to be enough for the DLSODA to calculate all AA,BB
    concetrations till time t=tn+1, and so more
86                     ! Ccvectors should be supplied To DLSODA. Thats why nmax  will be
    holding actually the time tn till we have
87                     ! Succefully calculated the AA,BB Concetrations
88   tmpCvceq=Cvceq
89   tmpCvcaeq=Cvcaeq
90   ir=0
91   CcAver=0
92   CcaAver=0
93   negSlope=.False.
94  !end of initialization
95
96 !Allocating vectors-Matrices
97  allocate (CcvectorT(2*nmax))      ! allocates the neccessary vector (2*nmax row the
    first nmax contain Cc and the rest contain Cca,
98                                    ! all values are for a speccific point.. and describe
     how it change thorugh time)
99  CcvectorT=0 ! a fast initialization
100 t_end=0
```

```
101 ! End of allocation
102
103
104 !!!<<<<<<<<<<<<<<<<<<<<<<<<<Starting initializing variables for the INTEL DODESOL approach DlSODA>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
105  NEQ = 2
106  Y(1) = 0.D0 ! Initial Values, for t=0,AA=BB=0
107  Y(2) = 0.D0 ! Initial Values, for t=0,AA=BB=0
108  T = 0.D0 ! initial value for the independant Value t (AA=BB=0 for T=0),always we start from t=0...
109           !but the next time AA_BB_Point_Over_Time_Calc is called the initial values AA0,BB0,Cc0 and Cca will have changed
110
111 ! global parameter settings
112 ! minimal step size for the methods
113          hm=1.d-20
114 ! relative tolerance. The code cannot guarantee the requested accuracy for ep<1.d-9
115          ep=1.d-8 ! absolute tolerance
116          tr=1.d-35
117
118 !!!*************************** dodesol ******************************
119 ! Please don't forget to initialize ipar array with zeros before the first call
120 ! to dodesol routines
121          DO i=1,128
122             ipar(i)=0
123          END DO
124          !ipar(4)=0 – the routine computes the numerical Jacobi matrix. In this case jacmat is a dummy parameter that will not be used.
125           !ipar(2)=0 !– the explicit or implicit scheme is chosen automatically,
126           ipar(2)=1 !– the Merson's method and the 1st order explicit up to 9-stage methods are used,
127           !ipar(2)=2 !– the implicit L-stable (5,2)-method of the 4th order is used.
128           !ipar(3)=Exit flag:
129           !ipar(3)=0 !ipar(3)=0 – exit at the end of the integration interval,1
130            ipar(3)=1 !ipar(3)=1 – exit after every successful step.
131           !ipar(6)=0 !– the explicit part works as the method with a variable order of accuracy and a variable number of stages in the 1st order method,
132           !ipar(6)=1 !– the explicit part works as the Merson's method,
133           !ipar(6)=7 !ipar(6)=k, 1<k<10 – the explicit part works as fixed k-stage 1st order method.
134
135
136 ! setting size of the system n, end of integration interval t_end, and initial
137 ! value y at t=0
138
139
140
141
142 !!!<<<<<<<<<<<<<<<<<<<<<<<<<end of initializing variables for the solver DlSODA>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
143
144
145
146  Do i=1,nx
147      do  j=1,ny
148         do k=1,nz
149
150         ir=ir+1
151
152
153
154         CcVectorT(1:nmax)=CcVectors(ir,1:nmax) ! we exctracted all time for a specific cell, Cc case
155         CcVectorT(nmax+1:2*nmax)=CcVectors(ir+nx*ny*nz,1:nmax) ! we exctracted all time for a specific cell,Cca case
156         Cc0=Ccn(ir)! extract initial values for a cell Cc case
157         Cca0=Ccn(ir+nx*ny*nz)! extract initial values for a cell, Cc case
158
159
160         !AA_BB_VecTrial(1:nmax)=AAVectors(ir,:) ! we fill the first half of Fvec vector, for each point we get its change through time,AAn case
161         !AA_BB_VecTrial(nmax+1:2*nmax)=BBvectors(ir,:) ! We fill the last half of Fvec Vector, for each point we get its change through time,AAn case
162         AA0=AAn(ir)! extract initial values for a cell AAn case
```

```
163         BB0=BBn(ir)! extract initial values for a cell, BBn case
164         Y(1)=AA0
165         Y(2)=BB0
166         h=1.d-7
167          T=0.d0 ! the initial value are for T=0, y(1)=AA0,y(2)==BB0
168          ipar=0 ! reinitializing parameters
169          ipar(3)=1 !ipar(3)=1 - exit after every successful step.
170          !ipar(2)=0 !- the explicit or implicit scheme is chosen automatically,
171          ipar(2)=1 !- the Merson's method and the 1st order explicit up to 9-stage
     methods are used,
172          !ipar(2)=2 !- the implicit L-stable (5,2)-method of the 4th order is used.

173          !ipar(8)=9 ! the maximal number of stages in the 1st order k-stage, k=9 now
174          ! start Temp just for debugging"
175
176           write(*,*)"we are currently on cell",ir
177
178          write(777,*)"<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<t=0 new cell
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>"
179          write(777,*)"For the cell with i=",i,"j= ",j,"k= ",k
180          write(777,*)"The AA and BB results are "
181          ! end temp sjust for debugging
182
183          DO it = 1,nmax
184           t_end=dt*it
185
186        99  if(CcvectorT(it)>1.d-28 .or.CcvectorT(it+nmax)>1.d-28) then
187
188              if(it==1)then
189
190                  !testing the slope
191                  if(CcvectorT(1)>=Cc0) then ! we have zero or positive slope (Cc slope)
192                    negSlope=.False.
193                    tmpCvceq=Cvceq
194                  else
195                    negSlope=.True.
196                    tmpCvceq=0.D0 ! the equilibrium is close to zero
197                  end if
198
199                  if(t+h >t_end) then
200                     h=T_end-t ! verifying that the region  we will check doesnt exceed
     T_end
201                  end if
202
203                  CcAver=((T+h)/dt)*CcvectorT(it)+(1.D0-(T+h)/dt)*Cc0
204                  CcaAver=((T+h)/dt)*CcvectorT(it+nmax)+(1.d0-(T+h)/dt)*Cca0
205
206                  if(y(1)<=CcAver*tmpCvceq+tr .and.y(1)>= CcAver*tmpCvceq-tr .and.y(2)<=
      CcaAver*tmpCvcaeq+tr .and.y(2)>= CcaAver*tmpCvcaeq-tr ) then ! results found
     previously seem good enough, and valid  for the next calculation step...no dodesol
     will be called, important to avoid numerical error due to roundoffs
207                   ! really usefull in case Cc doesnt change much and for that reason
     the dodesolver will probably fail to find fast enough a new better solution
208                     T=T+h  ! in case no calculations are needed the time being we
     move time forward by  h, the last valid time step
209                     h=2.D0*h
210                   ! we do nothing y(1),y(2) will remain the same, we only forward a
     little bit the time
211                  else
212                     ! we call intel_solver to calculate new y(1),y(2)
213                     CALL dodesol(ipar,NEQ,T,t_end,y,FEX,JDUM,h,hm,ep,tr,dpar,kd,ierr) !
      general solution
214                     if(ierr <0)then
215                      write(*,*)"intel solver failed to get results with the current
     step h=",h
216                      write(*,*)"error 1"
217                      pause
218                     end if
219                     !Y now holds the results on the tout time
220                  end if
221
222                  if(t<t_end) goto 99
223              else
224
```

```
225
226                 !testing the slope
227                 if(CcvectorT(it)>=CcvectorT(it-1)) then
228                   negSlope=.False. ! we have zero or positive slope (Cc slope)
229                   tmpCvceq=Cvceq
230                   tmpCvcaeq=Cvcaeq
231                 else
232                   negSlope=.True. ! we have zero or positive slope (Cc slope)
233                   tmpCvceq=0.D0 ! the equilibrium is close to zero
234                   tmpCvcaeq=0.D0
235                 end if
236
237                  if(t+h >t_end) then
238                    h=T_end-t ! verifying that the region  we will check doesnt exceed T_end
239                  end if
240
241
242
243               if(it>=3) then ! we will check the slope of y(1) and y(2)...
244                  if(AAVectors(ir,it-2)==0.d0 .or.BBVectors(ir,it-2)==0.D0 .or. CcvectorT(it-2)==0.D0 .or.CcvectorT(it-2+nmax)==0) goto 101 ! we avoid checking slopes division by zero detected
245                   slopeAA= (AAVectors(ir,it-1)-AAVectors(ir,it-2))/AAVectors(ir,it-2)
246                   slopeBB= (BBVectors(ir,it-1)-BBVectors(ir,it-2))/BBVectors(ir,it-2)
247                   slopeCc=  (CcvectorT(it-1)-CcvectorT(it-2))/CcvectorT(it-2)
248                   slopeCca= (CcvectorT(it-1+nmax)-CcvectorT(it-2+nmax))/CcvectorT(it-2+nmax)
249
250
251                   if ( Dabs(SlopeAA)<0.00001 .and.Dabs(SlopeBB)<0.00001.and.Dabs(SlopeCc)<0.000001.and.Dabs(SlopeCca)<0.000001 )  then ! results found  previously seem good enough, and valid  for the next calculation step...no dodesol will be called, important to avoid numerical error due to roundoffs.
252                     ! really usefull in case Cc doesnt change much and for that reason the dodesolver will probably fail to find fast enough a new better solution
253                       T=T+h  ! in case no calculations are needed the time being we move time forward by  h, the last valid time step
254                       h=2*h  ! we increase step in order not to do iterations with really small step without reason (later on h  will be checked so T+h<dt)
255                    ! we do nothing y(1),y(2) will remain the same, we only forward a little bit the time
256                   else
257                       ! we call intel_solver to calculate new y(1),y(2)
258                    101 CALL dodesol(ipar,NEQ,T,t_end,y,FEX,JDUM,h,hm,ep,tr,dpar,kd, ierr) ! general solution
259                      if(ierr<0 )then
260                       write(*,*)"intel solver failed to get results with the current step h=",h
261                       write(*,*)"error 2"
262                       pause
263                      end if
264                     ! Y now holds the results on the tout time
265
266                   end if
267
268                 else
269
270                  ! i<4 we cant check second derivative yet
271                  ! we call intel_solver to calculate new y(1),y(2)
272                  CALL dodesol(ipar,NEQ,T,t_end,y,FEX,JDUM,h,hm,ep,tr,dpar,kd,ierr) ! general solution
273                  if(ierr<0 )then
274                     write(*,*)"intel solver failed to get results with the current step h=",h
275                     write(*,*)"error 3"
276                     pause
277                   end if
278                  ! Y now holds the results on the tout time
279
280               end if
281
282               if(t<t_end) goto 99
283             end if
```

```
284
285
286             else
287              ! really small numbers were given from cc and thus its likely our attempt to solve for AA and BB will probably fail
288              ! also even  the calculattions for AA and BB werent to fail, it wouldnt matter since the resulting AA,BB  would have
289              ! the same magnitude which means AA=BB<10-34 =0
290              ! AA=BB<10-34 =0
291              Y(1)=0.0 ! AA is set to zero their value is so small and so there isnt any reason for us to calculate
292              Y(2)=0.0 ! BB is set to zero their value is so small and so there isnt any reason for us to calculate
293              t=T_end ! we move forward the time
294             end if
295
296
297
298              IF(ierr< 0) THEN
299                PRINT*,'======================='
300                PRINT*,'DODESOL FORTRAN  FAILED'
301               PRINT*,'dodesol routine exited with error code',ierr
302                Print*,"Please check manual of intel ode solvers for what the error means "
303                Print*,"You may change the input data and try again"
304                Print*,"The system will now pause"
305                pause 1
306             END IF
307            !start Temp just for debugging"
308             !write(777,54)t_end,y(1), y(2)
309             !54 format("  t=  ",Es14.3,"  AA=   ",En14.4, "  BB=   ",En14.4)
310             !54 format(Es14.3,"   ",En14.4, "   ",En14.4)
311           ! end temp sjust for debugging
312
313
314            AAVectors(ir,it)= Y(1)  ! AAvectors,BBvectors  is filled iteratively for all different cells
315            BBvectors(ir,it)= Y(2)  ! and for all different times (Based on the outCome of DLSODA,y(1)=AA,y(2)=BB)
316           End do
317
318
319
320
321
322          ! once AA_BB_VecTrial successfully aqcuires the correct data AA,BB it stores them to the AAVectors and BBvectors
323          ! but before that we need to undo the changes we did in order to reduce stiffness
324
325
326
327
328
329          End do
330       end do
331    End do
332
333
334
335    ! removes memory that wont be used by the main program
336     deallocate (CcvectorT)
337
338
339   END subroutine
340
341
342
343
344    End module
345
346
347
348
```

```
349
350     SUBROUTINE Jdum (n,t,y,a) ! we wont use jdum at all... since we numerically calculate the jacobi matrix...
351     IMPLICIT NONE
352     INTEGER n
353     DOUBLE PRECISION t,y(*),a(n,*)
354     ! though we need to create  a dummy subroutine Jdum since the Dodesol requires so
355     RETURN
356    END subroutine
357
358
359
360
361
362
363  Subroutine FEX(Neq,T,Y,YDot)
364   use AA_BB_Solver
365  implicit none !  FEX calculates the equation 8 and 13 based on AA0,BB0,CcVectorT,Cc0, Cca0,Cvceq,Cvcaeq,ScaleFvec,dt,nmax
366                !  where Neq= number of equations to be solved (2) in our case
367                !  T= initial value of our independant variable which is zero (t=0)
368                !  Y= array of initial values for our Ydot (t=0 => AA=0,BB=0),afterwards Y contains the point of which we want to calculate the derivatives YDot
369                !  Ydot time derivatives of each different equation
370                !  Ydot(1)   returns the time derivative from  equation 8, for the given input parameters. AA=CcCvc
371                !  Ydot(2)   returns the time derivative from  equation 13, for the given input parameters. BB=Cc*Cvc*
372                !  for more read odepack.f ,DLSODA Region
373
374   double Precision    FAA1 ! Functions FAA1  returns the value of equation 8, for the given input parameters. AA=CcCvc    , positive slope
375   double Precision    FBB1 ! Functions FBB1  returns the value of equation 13, for the given input parameters. BB=Cc*Cvc* , positive slope
376   double Precision    FAA2 ! Functions FAA1  returns the value of equation 8, for the given input parameters. AA=CcCvc    , negative slope
377   double Precision    FBB2 ! Functions FBB1  returns the value of equation 13, for the given input parameters. BB=Cc*Cvc* , negative slope
378
379   double Precision  Y(neq),YDOT(neq)
380   integer  Neq ! Neq = Number of differential equations to be solved together (only first order can be solved)
381   double Precision T   !the independent variable.  On input, T is used only on the
382              !first call, as the initial point of the integration.
383              !on output, after each call, T is the value at which a
384              !computed solution Y is evaluated (usually the same as TOUT).
385              !on an error return, T is the farthest point reached.
386
387
388   !integer   nmax ! nmax holds the number of time steps we will perform. Based on that nmax determines the size of the jacoby matrix ! already defined in module
389   double Precision    AAijktn, BBijktn    !usefull  variable holding concetrations on t=tn time (BB=Cc*Cvc*)
390   double Precision    Ccijktn,Ccijktnp1   !Variables holding the concetration of the colloids at the i,j,k cell, on  time t=tn and t=tnp1
391   double Precision    Ccaijktn,Ccaijktnp1 !Ccaijktnp1 holds the concetration of the sorbed colloids into the solid matrix on the next time step
392                                 !Ccaijktn  holds the concetration of the sorbed colloids into the solid matrix on the current time
393                                 !Cveq max holds the concetration of viruses attached onto clay collids  at equilibrium (M viruses)/(M clay)
394                                 !Cvcaeq max holds the concetration of viruses attached onto clay collids already attached onto glass beads at equilibrium (M viruses)/(M clay)
395   !double Precision  Cvceq,Cvcaeq  dt     !dt= the  (time) distance between two sequential time moments t1,t2  =dt=t2-t1 ! these 3 varaibles are already defined in the specification area of the module
396   !double Precision Cc0,Cca0 ! Cc0 and Cca0 are the initial conditions for the case of Solute colloids and Sorbed on the solid matrix colloids ! already defined in module
397   integer ir ! iterator
398   integer nRegion ! Y is   the point on which we want to calculate the ydot's. But even if Y seems to be continue its not since time is discretized and splited in parts of dt width
399                 ! nRegion1 indicates  in what region (part ) Y is located (needed for
```

```
      the AA variable)
400    double Precision wMM  ! Cc= Ccijktn*(1 - wMM1) + Ccijktnp1*wMM1. wMM1 indicates how
        much of Ccijktn and Ccijktnp1 we need to construct properly the Cc, here wMM1 isnt
        related to Crank-Nikolson, we got explicit shemes
401
402
403    !double Precision , allocatable :: CcvectorT(:) ! Vector that will hold Cc and Cca
        concetrations for a specific x,y,z cell over time...t=tn+1.....+n  (were n=nmax)  !
         already defined in module
404                                       ! The first nmax rows are Cc concetrations and
        the rest are Cca concetrations
405
406
407    ! Inialization
408    Neq=2 ! dont really needed ... but the DODESOL interface requires it...
409    !T.... The shame as above goes for T , The DLSODA requires it but... the subroutine
        dont...
410    ir=0
411    nregion=0
412
413    wmm=1
414    ! end of Initialization
415
416      nRegion=int(T/dt)
417      wMM=T/dt-nRegion ! Basicly the decimal part of the division is stored in wMM, so the
        correct analogy of Ccijktnp1 and Ccijktn .
418                       ! Cc= Ccijktn*(1 - wMM) + Ccijktnp1*wMM. If i have y=1.2 ,dt=1 then
        i need 0.2 value from Ccijktnp2 and 0.8 from Ccijktnp1
419
420
421      ! Based on given Y (point at which we want to calculate the Ydot's) we will
         calculate the needed parameterers  Ccijktn,Ccijktnp1,AAijktn and BBijktn
422
423      if(nRegion==0) then
424        ! here we need also to apply initial conditions
425        Ccijktn=Cc0 ! Cc0 =soluted concetration of colloids at t=0
426        Ccijktnp1=CcVectorT(1)  ! Calculating Ccijktn on the Ccn vector ( time t=tn+1)
427        AAijktn=Y(1)  !T(1)=AA0,T(2)=BB0
428        BBijktn=Y(2)
429        Ccaijktn=Cca0 ! Cca0 =sorbed concetration of colloids at t=0
430        Ccaijktnp1=CcVectorT(1+nmax)  ! Calculating Ccaijktn on the Ccn vector ( time t=tn
         +1)
431      Else if(nRegion<nmax) then ! not equal... since we need n+1...but if n=nmax , the
         nmax+1 doesnt exist
432        !Everything we need here comes from CcvectorT
433        Ccijktn=CcVectorT(nRegion) ! Calculating Ccijktn on the Ccn vector ( time t=tn)
434        Ccijktnp1=CcVectorT(nRegion+1)  ! Calculating Ccijktn on the Ccn vector ( time t=
         tn+1)
435        AAijktn=Y(1)  !Y(1)=AAijktn,T(2)=BBijktn
436        BBijktn=Y(2)
437        Ccaijktn=CcVectorT(nRegion+nmax) ! Cca0 =sorbed concetration of colloids at t=0
438        Ccaijktnp1=CcVectorT(nRegion+nmax+1)  ! Calculating Ccaijktn on the Ccn vector (
         time t=tn+1)
439      else if (nmax<nRegion<nmax+1) then ! we dont have values for Cc for the Y the DLSODA
         asked. So we cant calculate for the given y then YDot's
440        ! If  we are to close to the end of the given CcvectorT ... we can assume that the
         last value of the ccvectorT is representative and for the next value(although that
         doesnt even matter since we put wm=0 and we use only the previous ccijktn value)
441        wmm=0
442        Ccijktn=CcVectorT(nRegion) ! Calculating Ccijktn on the Ccn vector ( time t=tn)
443        Ccijktnp1=CcVectorT(nRegion)  ! Calculating Ccijktn on the Ccn vector ( time t=tn+
         1)
444        AAijktn=Y(1)  !Y(1)=AAijktn,T(2)=BBijktn
445        BBijktn=Y(2)
446        Ccaijktn=CcVectorT(nRegion+nmax) ! Cca0 =sorbed concetration of colloids at t=0
447        Ccaijktnp1=CcVectorT(nRegion+nmax)  ! Calculating Ccaijktn on the Ccn vector (
         time t=tn+1)
448      else
449        write(*,*)" we dont have values for Cc for the Y the DLSODA asked. So we cant
         calculate for the given y then YDot's"
450        Pause
451
452      end if
453
```

```fortran

   if (negSlope==.False.) then
     YDot(1)=FAA1(Ccijktn,Ccijktnp1,AAijktn,BBijktn,wMM)

     YDot(2)=FBB1(Ccaijktn,Ccaijktnp1,AAijktn,BBijktn,wMM)
   else
    YDot(1)=FAA2(Ccijktn,Ccijktnp1,AAijktn,BBijktn,wMM) ! functions for the negative slope

    YDot(2)=FBB2(Ccaijktn,Ccaijktnp1,AAijktn,BBijktn,wMM) ! functions for the negative slope
   end if


 End Subroutine










Function FAA1(Ccijktn,Ccijktnp1,AAijktn,BBijktn,wMM)
use  AA_BB_Solver , only: dt,Cvceq    ! uses the module to share globally some intresting variables
implicit none
  double Precision    FAA1         ! Functions FAA1  returns the value of equation 8, for the given input parameters. AA=CcCvc
  double Precision    AAijktn,BBijktn    !AAijktn usefull  variable holding concetrations on t=tn time (AA=CcCvc)
                              !BBijktn usefull  variable holding concetrations on t=tn time (BB=Cc*Cvc*)
  double Precision    Ccijktn,Ccijktnp1 !Variables holding the concetration of the colloids at the i,j,k cell, on  time t=tn and t=tnp1
  !double Precision    Cvceq             !Cveq max holds the concetration of viruses attached onto clay collids  at equilibrium (M viruses)/(M clay)! Already defined in the specification area of the module
  !double Precision    dt                !dt= the  (time) distance between two sequential time moments t1,t2  =dt=t2-t1  ! Already defined in the specification area of the module
  double Precision Lvc,Lvca ! decay rates coefficients  when virus is attached to the colloid, and virus is attached to the colloid and all together sorbed on the solid matrix
  double Precision Rvcdv,Rvdvc,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc  ! forward or reverse rates cooefficients  (Virus case)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

  double Precision th,rD              !th=thh=porosity (or th=th* the effective porosity can be used as well  page 509) ,
                               !rD= bulk density of the solid matrix(solid mass/aquifer volume)
  double Precision wMM  ! Cc= Ccijktn*(1 - wMM) + Ccijktnp1*wMM. wMM indicates how much of Ccijktn and Ccijktnp1 we need to construct properly the Cc, here wMM isnt related to Crank-Nikolson,we got explicit shemes



  Common/VarTran2/rD,th
  Common/vartrans8/Rvdvc,Rvcdv,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc
  Common/vartrans9/Lvc,Lvca


 If( (Ccijktn*wMM + (1.D0-wMM)* Ccijktnp1)*Cvceq-AAijktn>=0.D0 )then ! we test wether max concetration of AA has being reached in the system during our time step. Cvceq= max Cvc concetration

   FAA1=-(AAijktn*Lvc) - AAijktn*Rvcdv - AAijktn*Rvcdvca + (BBijktn*rd*Rvcadvc)/th + Rvdvc*(Ccijktn*(1 - wMM) + Ccijktnp1*wMM)*(Cvceq - AAijktn/(Ccijktn*(1 - wMM) + Ccijktnp1*wMM))**2
 else
   !FAA1=0.D0 ! since max cocnetration has being reached the derivative dAA/dt is set equal to zero to ensure we wont Cvceq is the max possible concetrations
  !FAA1=1.D-150
```

```
504   FAA1= ((Ccijktn*wMM + (1.D0-wMM)* Ccijktnp1)*Cvceq-AAijktn)/dt
505   !FAA1=-(AAijktn*Lvc) - AAijktn*Rvcdv - AAijktn*Rvcdvca + (BBijktn*rd*Rvcadvc)/th -
        Rvdvc*(Ccijktn*(1 - wMM) + Ccijktnp1*wMM)*(Cvceq - AAijktn/(Ccijktn*(1 - wMM) +
        Ccijktnp1*wMM))**2
506   end if
507
508
509
510 End function
511
512
513
514
515
516
517   Function FBB1(Ccaijktn,Ccaijktnp1,AAijktn,BBijktn,wMM)
518   use  AA_BB_Solver , only: dt,Cvcaeq   ! uses the module to share globally some
        intresting variables
519    implicit none
520    double Precision    FBB1 ! Functions FBB1  returns the value of equation 13, for the
        given input parameters. BB=Cc*Cvc*
521    double Precision    AAijktn,BBijktn !AAijktn usefull  variable holding concetrations
        on t=tn time (AA=CcCvc)
522                               !BBijktn usefull  variable holding concetrations on t=tn
        time (BB=Cc*Cvc*)
523    double Precision    Ccaijktn,Ccaijktnp1 !Ccaijktnp1 holds the concetration of the
        sorbed colloids into the solid matrix on the next time step
524                                !Ccaijktn  holds the concetration of the sorbed colloids
        into the solid matrix on the current time
525
526    !double Precision  Cvcaeq !Cvcaeq max holds the concetration of viruses attached onto
         clay collids already attached onto glass beads at equilibrium (M viruses)/(M clay)
        ! Already defined in the specification area of the module
527    !double Precision  dt     !dt= the  (time) distance between two sequential time
        moments t1,t2  =dt=t2-t1  ! Already defined in the specification area of the module
528    double Precision Lvc,Lvca ! decay rates coefficients  when virus is attached to the
        colloid, and virus is attached to the colloid and all together sorbed on the solid
        matrix
529    double Precision Rvcdv,Rvdvc,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc  ! forward or reverse rates
         cooefficients  (Virus case)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

530    double Precision th,rD              !th=thh=porosity (or th=th* the effective porosity
         can be used as well  page 509) ,
531                                !rD= bulk density of the solid matrix(solid mass/aquifer
         volume)
532    double Precision wMM  ! Cc= Ccijktn*(1 - wMM) + Ccijktnp1*wMM. wMM indicates how much
        of Ccijktn and Ccijktnp1 we need to construct properly the Cc, here wMM isnt
        related to Crank-Nikolson,we got explicit shemes
533
534
535
536    Common/VarTran2/rD,th
537    Common/vartrans8/Rvdvc,Rvcdv,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc
538    Common/vartrans9/Lvc,Lvca
539
540
541
542
543    If( (Ccaijktn*wMM + (1.D0-wMM)* Ccaijktnp1)*Cvcaeq-BBijktn>=0.D0 )then ! we test
        wether max concetration of BB has being reached in the system during our time step.
         Cvcaeq=max Cvca concetration
544      FBB1= -(BBijktn*Lvca) - BBijktn*Rvcadv - BBijktn*Rvcadvc + (AAijktn*Rvcdvca*th)/rd
        + Rvdvca*(Ccaijktn*(1 - wMM) + Ccaijktnp1*wMM)*(Cvcaeq - BBijktn/(Ccaijktn*(1 -
        wMM) + Ccaijktnp1*wMM))**2
545    else
546     !FBB1=0.D0 ! since max cocnetration has being reached the derivative dAA/dt is set
        equal to zero to ensure we wont Cvcaeq is the max possible concetrations
547     !FBB1=1.D-150
548      FBB1=((Ccaijktn*wMM + (1.D0-wMM)* Ccaijktnp1)*Cvcaeq-BBijktn)/dt
549      !FBB1= -(BBijktn*Lvca) - BBijktn*Rvcadv - BBijktn*Rvcadvc + (AAijktn*Rvcdvca*th)/rd
         -Rvdvca*(Ccaijktn*(1 - wMM) + Ccaijktnp1*wMM)*(Cvcaeq - BBijktn/(Ccaijktn*(1 -
        wMM) + Ccaijktnp1*wMM))**2
550
551    end if
```

```
552
553     !temp just for debugging
554   ! start Temp just for debugging"
555
556   !write(*,*)"Fbb1 was set to zero due to   AA>Cca Cvcaeq"
557
558  !write(*,*)"Dumping variables"
559  !write(*,*)"AAijktn given is ",AAijktn
560  !write(*,*)"BBijktn given is ",BBijktn
561  !write(*,*)"CCaijktn given is ",Ccaijktn
562  !write(*,*)"CCaijktnp1 given is ",Ccaijktnp1
563  !write(*,*)"wmmgiven is ",wmm
564  !write(*,*)"Result ydot is ",FBB1
565  ! end temp sjust for debugging
566
567 End function
568
569
570
571
572
573
574
575 Function FAA2(Ccijktn,Ccijktnp1,AAijktn,BBijktn,wMM)
576 use  AA_BB_Solver , only: dt,Cvceq   ! uses the module to share globally some
        intresting variables
577 implicit none
578   double Precision    FAA2        ! Functions FAA1  returns the value of equation 8, for
        the given input parameters. AA=CcCvc
579   double Precision  tmpFAA2 ! temp Varaible holding FAA2 value
580   double Precision    AAijktn,BBijktn   !AAijktn usefull  variable holding concetrations
        on t=tn time (AA=CcCvc)
581                             !BBijktn usefull  variable holding concetrations on t=tn
        time (BB=Cc*Cvc*)
582   double Precision    Ccijktn,Ccijktnp1 !Variables holding the concetration of the
        colloids at the i,j,k cell, on  time t=tn and t=tnp1
583   !double Precision    Cvceq           !Cveq max holds the concetration of viruses
        attached onto clay collids  at equilibrium (M viruses)/(M clay)! Already defined in
        the specification area of the module
584   !double Precision    dt              !dt= the  (time) distance between two sequential
        time moments t1,t2  =dt=t2-t1  ! Already defined in the specification area of the
        module
585   double Precision Lvc,Lvca ! decay rates coefficients  when virus is attached to the
        colloid, and virus is attached to the colloid and all together sorbed on the solid
        matrix
586   double Precision Rvcdv,Rvdvc,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc  ! forward or reverse rates
        cooefficients  (Virus case)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

587   double Precision th,rD              !th=thh=porosity (or th=th* the effective porosity
        can be used as well  page 509) ,
588                               !rD= bulk density of the solid matrix(solid mass/aquifer
        volume)
589   double Precision wMM  ! Cc= Ccijktn*(1 - wMM) + Ccijktnp1*wMM. wMM indicates how much
        of Ccijktn and Ccijktnp1 we need to construct properly the Cc, here wMM isnt
        related to Crank-Nikolson,we got explicit shemes
590
591
592
593
594   Common/VarTran2/rD,th
595   Common/vartrans8/Rvdvc,Rvcdv,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc
596   Common/vartrans9/Lvc,Lvca
597
598
599
600   If( (Ccijktn*wMM + (1.D0-wMM)* Ccijktnp1)*Cvceq-AAijktn<0.D0  )then ! we should never
        exceed the Cveq
601      !AAijktn=(Ccijktn*wMM + (1.D0-wMM)* Ccijktnp1)*Cvceq
602      !FAA2=-(AAijktn*Lvc) - AAijktn*Rvcdv - AAijktn*Rvcdvca + (BBijktn*rd*Rvcadvc)/th -
        Rvdvc*(Ccijktn*(1 - wMM) + Ccijktnp1*wMM)*(Cvceq )**2.D0
603      FAA2=((Ccijktn*wMM + (1.D0-wMM)* Ccijktnp1)*Cvceq-AAijktn)/dt
604      tmpFAA2=-(AAijktn*Lvc) - AAijktn*Rvcdv - AAijktn*Rvcdvca + (BBijktn*rd*Rvcadvc)/th
        - Rvdvc*(Ccijktn*(1 - wMM) + Ccijktnp1*wMM)*(-AAijktn/(Ccijktn*(1 - wMM) +
       Ccijktnp1*wMM))**2.D0
```

```fortran
     if( tmpFAA2<FAA2) FAA2=tmpFAA2 ! we select the absolute bigger slope
  else  If( AAijktn>=0.D0 )then ! we test wether max concetration of AA has being reached in the system during our time step. Cvceq=mac Cvc concetration
     FAA2=-(AAijktn*Lvc) - AAijktn*Rvcdv - AAijktn*Rvcdvca + (BBijktn*rd*Rvcadvc)/th - Rvdvc*(Ccijktn*(1 - wMM) + Ccijktnp1*wMM)*(-AAijktn/(Ccijktn*(1 - wMM) + Ccijktnp1*wMM))**2.D0
  else
   !FAA2=0.D0 ! since max cocnetration has being reached the derivative dAA/dt is set equal to zero to ensure we wont Cvceq is the max possible concetrations
   !AAijktn=(Ccijktn*wMM + (1.D0-wMM)* Ccijktnp1)*Cvceq
   FAA2=1.D-150

 end if

End function






 Function FBB2(Ccaijktn,Ccaijktnp1,AAijktn,BBijktn,wMM)
 use  AA_BB_Solver , only: dt,Cvcaeq  ! uses the module to share globally some intresting variables
  implicit none
  double Precision    FBB2  ! Functions FBB1  returns the value of equation 13, for the given input parameters. BB=Cc*Cvc*
  double Precision  tmpFBB2 ! temp Varaible holding Fbb2 value
  double Precision    AAijktn,BBijktn !AAijktn usefull  variable holding concetrations on t=tn time (AA=CcCvc)
                           !BBijktn usefull  variable holding concetrations on t=tn time (BB=Cc*Cvc*)
  double Precision    Ccaijktn,Ccaijktnp1 !Ccaijktnp1 holds the concetration of the sorbed colloids into the solid matrix on the next time step
                            !Ccaijktn  holds the concetration of the sorbed colloids into the solid matrix on the current time

  !double Precision  Cvcaeq !Cvcaeq max holds the concetration of viruses attached onto clay collids already attached onto glass beads at equilibrium (M viruses)/(M clay) ! Already defined in the specification area of the module
  !double Precision  dt     !dt= the  (time) distance between two sequential time moments t1,t2  =dt=t2-t1  ! Already defined in the specification area of the module
  double Precision Lvc,Lvca ! decay rates coefficients  when virus is attached to the colloid, and virus is attached to the colloid and all together sorbed on the solid matrix
  double Precision Rvcdv,Rvdvc,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc  ! forward or reverse rates cooefficients  (Virus case)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

  double Precision th,rD              !th=thh=porosity (or th=th* the effective porosity can be used as well  page 509) ,
                             !rD= bulk density of the solid matrix(solid mass/aquifer volume)
  double Precision wMM  ! Cc= Ccijktn*(1 - wMM) + Ccijktnp1*wMM. wMM indicates how much of Ccijktn and Ccijktnp1 we need to construct properly the Cc, here wMM isnt related to Crank-Nikolson,we got explicit shemes



  Common/VarTran2/rD,th
  Common/vartrans8/Rvdvc,Rvcdv,Rvdvca,Rvcdvca,Rvcadv,Rvcadvc
  Common/vartrans9/Lvc,Lvca



  If( (Ccaijktn*wMM + (1.D0-wMM)* Ccaijktnp1)*Cvcaeq-BBijktn<0.D0  )then ! we test wether max concetration of BB has being reached in the system during our time step. Cvcaeq=max Cvca concetration
    !BBijktn= (Ccaijktn*wMM + (1.D0-wMM)* Ccaijktnp1)*Cvcaeq ! max concetration reached ... We set the current concetration to max possible
    !FBB2= -(BBijktn*Lvca) - BBijktn*Rvcadv - BBijktn*Rvcadvc + (AAijktn*Rvcdvca*th)/rd - Rvdvca*(Ccaijktn*(1 - wMM) + Ccaijktnp1*wMM)*(Cvcaeq)**2.D0
    FBB2=((Ccaijktn*wMM + (1.D0-wMM)* Ccaijktnp1)*Cvcaeq-BBijktn)/dt
    tmpFBB2= -(BBijktn*Lvca) - BBijktn*Rvcadv - BBijktn*Rvcadvc + (AAijktn*Rvcdvca*th)/rd - Rvdvca*(Ccaijktn*(1 - wMM) + Ccaijktnp1*wMM)*(-BBijktn/(Ccaijktn*(1 - wMM) +
```

```
      Ccaijktnp1*wMM))**2.D0
653     if( tmpFBB2<FBB2) FBB2=tmpFBB2 ! we select the absolute bigger slope
654   else If( BBijktn>=0.D0 )then ! we test wether max concetration of AA has being reached
         in the system during our time step. Cvceq=mac Cvc concetration
655     FBB2= -(BBijktn*Lvca) - BBijktn*Rvcadv - BBijktn*Rvcadvc + (AAijktn*Rvcdvca*th)/rd
        - Rvdvca*(Ccaijktn*(1 - wMM) + Ccaijktnp1*wMM)*(-BBijktn/(Ccaijktn*(1 - wMM) +
        Ccaijktnp1*wMM))**2.D0
656    !FBB2= -(BBijktn*Lvca) - BBijktn*Rvcadv - BBijktn*Rvcadvc + (AAijktn*Rvcdvca*th)/rd
        - Rvdvca*(Ccaijktn*(1 - wMM) + Ccaijktnp1*wMM)*(Cvcaeq - BBijktn/(Ccaijktn*(1 -
        wMM) + Ccaijktnp1*wMM))**2
657   else
658    !FBB2=0.D0 ! since max cocnetration has being reached the derivative dAA/dt is set
        equal to zero to ensure we wont Cvcaeq is the max possible concetrations
659     FBB2=1.D-150
660
661   end if
662
663
664
665
666 End function
667
668
```