ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΜΗΧΑΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΔΙΑΔΙΚΤΥΑΚΕΣ & ΦΟΡΗΤΕΣ ΕΦΑΡΜΟΓΕΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΒΑΣΙΛΕΙΟΣ ΒΛΑΧΟΣ

Δημιουργία λογισμικού προσομοίωσης βιολογικών ιών

ΘΕΟΔΩΡΟΣ ΣΙΚΛΑΦΙΔΗΣ
Α.Μ.: 7419021

17 Φεβρουαρίου 2021

Ευχαριστίες

Ολοκληρώνοντας την παρούσα διπλωματική εργασία, θα ήθελα να ευχαριστήσω τον Δρ. Βασίλειο Βλάχο, Καθηγητή του Πανεπιστημίου Θεσσαλίας, για την σταθερή συνεργασία αλλά και για την συνεχής καθοδήγηση καθ'όλη την διάρκεια του έργου αυτού.

Θα ήθελα επίσης να ευχαριστήσω:

Την Μητέρα μου και τον Πατέρα μου για την αδιάκοπη στήριξη που μου παρέχουν, για την υπομονή τους, αλλά και για την ώθηση που μου έδωσαν μετά τις προπτυχιακές μου σπουδές, έτσι ώστε να μπορέσω να συνεχίσω αλλά και να αντεπεξέλθω στις προκλήσεις των μεταπτυχιακών σπουδών μου.

Περιεχόμενα

Περίληψη

Η ανάπτυξη και η πορεία μολυσματικών ασθενειών, βιολογικών ιών και παθογόνων ταχείας εξάπλωσης θεωρείται ένα από τα πιο δύσκολα πράγματα όσον αφορά την ανάλυση, την πρόβλεψη αλλά και την αποτροπή τους. Πολλά επιδημιολογικά μοντέλα έχουν αναπτυχθεί με το πέρασμα του χρόνου, με σκοπό την αντιμετώπιση του προβλήματος της πρόβλεψης της πορείας αλλά και της εξέλιξης ενός ιού με αναμενόμενο αποτέλεσμα την αποτροπή μιας πανδημίας. Τα περισσότερα όμως από αυτά τα μοντέλα, μπορούν να εφαρμοστούν μοναχά σε πλήρως ομοιογενή σύνολα, πράγμα το οποίο δεν αντικατοπτρίζει την πραγματικότητα της ανθρώπινης συμπεριφοράς, όσον αφορά το θέμα της δικτύωσης με άλλους ανθρώπους. Το παρόν έγγραφο παρουσιάζει το BVS, ένα εργαλείο εύκολο στην χρήση του, που συνδυάζει διάφορα μοντέλα από την Θεωρία των Γράφων με σύγχρονα Επιδημιολογικά Μοντέλα, με σκοπό να παρέχει μια αρχική εκτίμηση για το πως θα μπορούσε να εξελιχθεί μια μολυσματική ασθένεια σε διάφορες τροπολογίες οι οποίες προσεγγίζουν σε μεγάλο βαθμό τις κοινωνικές τοπολογίες που δημιουργούν οι άνθρωποι μεταξύ τους. Ένα σημαντικό χαρακτηριστικό του εργαλείου αυτού είναι ότι έχει κατασκευαστεί με τρόπο που επιτρέπει την αναπαραγωγή ορισμένων προσομοιώσεων με στόχο να βοηθήσει τους χρήστες να συγκρίνουν διαφορετικά αποτελέσματα που αφορούν την εξέλιξη μιας μολυσματικής ασθένειας.

Εισαγωγή

2.1 Ορισμός Επιδημιολογίας

Επιδημιολογία είναι η μελέτη της κατανομής και της εξέλιξης διαφόρων νοσημάτων στον ανθρώπινο πληθυσμό (περιγραφική επιδημιολογία) και των παραγόντων που τις διαμορφώνουν ή μπορούν να τις επηρεάσουν (αναλυτική επιδημιολογία). Ο ενεργός επιδημιολόγος που εργάζεται σε τομείς η θεματολογία των οποίων εκτείνεται από πρακτικά ζητήματα, όπως η διερεύνηση μίας επιδημίας, μία περιβαλλοντική έκθεση και η αγωγή υγείας, μέχρι πιο θεωρητικά, όπως η ανάπτυξη στατιστικών, μαθηματικών, φιλοσοφικών, βιολογικών και ψυχοκοινωνικών θεωριών. Έτσι, οι επιδημιολόγοι ασχολούνται σε μία σειρά τύπων επιστημονικής μελέτης, από μελέτες παρατήρησης έως πειραματικές μελέτες, με σκοπό τον προσδιορισμό αμερόληπτων σχέσεων μεταξύ εκθέσεων, όπως η διατροφή, οι βιολογικοί παράγοντες, το άγχος, και του αποτελέσματος, όπως παραδείγματος χάριν μία ασθένεια, η υγεία και λοιπά. Στη σύγχρονη επιδημιολογία, χρησιμοποιούνται μέθοδοι και τεχνικές από την Πληροφορική της Υγείας.

2.2 Ιστορική Αναδρομή

Επιδημιολογία είναι η μελέτη της κατανομής και της εξέλιξης διαφόρων νοσημάτων και ιών στον ευρύτερο πληθυσμό και των παραγόντων που τις διαμορφώνουν ή μπορούν να τις επηρεάσουν. Παρά το γεγονός του ότι η πλειοψηφία των ανθρώπων δεν έχει ακουστά τον όρο επιδημιολογία και το τι αντιπροσωπεύει, η επιδημιολογία δεν είναι κάτι το καινούργιο. Από το 400 π.χ. ο Ιπποκράτης ο οποίος είναι γνωστός και ως ο πατέρας της ιατρικής και πιστεύετε ότι είναι ο πρώτος επιδημιολόγος, επιχείρησε να αναζητήσει λογική στην ανθρώπινη ασθένεια. Έτσι λοιπόν έθεσε τα θεμέλια, γι αυτό που τώρα αποκαλούμε επιδημιολογία [1]. Οι μελέτες του Ιπποκράτη υπέδειξαν ότι οι αρρώστιες δημιουργούνται στο ανθρώπινο σώμα από την ανισορροπία μεταξύ τεσσάρων στοιχείων τα οποία είναι η μαύρη χολή, η κίτρινη χολή, το αίμα και το φλέγμα, καθώς και το ότι η θεραπεία για την ασθένεια που προέκυπτε ήταν να προστεθεί ή να αφαιρεθεί κάποιο από τα παραπάνω στοιχεία με σκοπό να διατηρηθεί η ισορροπία μεταξύ τους και έτσι να επιτευχθεί η πλήρης υγεία του ασθενή. Περαιτέρω, δύο από τους πιο συνηθισμένους όρους της επιδημιολογίας οι οποίοι είναι η ενδημία και η επιδημία, επινοήθηκαν επίσης από τον ίδιο τον Ιπποκράτη [2].

Στην επιδημιολογία, μία μολυσματική ασθένεια θεωρείται ότι είναι ενδημική όταν αυτή υπάρχει σε συγκεκριμένο αριθμό ανθρώπων, συγκεκριμένες ομάδες ή διατηρείτε συνεχώς σε μια συγκεκριμένη γεωγραφική περιοχή, χωρίς εξωτερικές παρεμβάσεις. Από την άλλη μεριά, μια ασθένεια θεωρείται ότι είναι επιδημική όταν αυτή εξαπλώνεται με πολύ μεγάλη ταχύτητα σε έναν μεγάλο αριθμό ανθρώπων, σε πολύ σύντομο χρονικό διάστημα. Παρά το γεγονός του ότι μια επιδημία μπορεί να περιοριστεί σε μια συγκεκριμένη γεωγραφική περιοχή, μπορεί επίσης να εξαπλωθεί και σε άλλες χώρες ή και ηπείρους επηρεάζοντας έναν ακόμη μεγαλύτερο αριθμό ανθρώπων όπου πλέον μπορεί να ορισθεί και ως πανδημία [3] [4] [5]. Αφού λοιπόν ο Ιπποκράτης χάραξε το μονοπάτι της επιδημιολογίας, ακολούθησε μια σειρά σημαντικών ανακαλύψεων αλλά και βελτιώσεων στον τομέα αυτόν.

Το 1676, ο Thomas Sydenham, ένας ΄Αγγλος φυσικός ο οποίος είναι γνωστός και ως η αγγλική έκδοση του Ιπποκράτη, δημοσίευσε τις παρατηρήσεις του σε ένα βιβλίο με τίτλο Observationes Medicae. Η ταξινομήσεις των πυρετών που μάστιζαν το Λονδίνο τις χρονιές 1660 μέχρι 1670 σε τρεις κατηγορίες (συνεχείς πυρετοί, διαλείπουσα πυρετό και ευλογιά), ήταν ένα από τα πιο σημαντικά έργα του Thomas Sydenham.

Κατά την διάρκεια του 1854, ο John Snow ο οποίος ήταν εξίσου ΄Αγγλος φυσικός και θεωρείται ένας από τους ιδρυτές της σύγχρονης επιδημιολογίας ξεκίνησε μια έρευνα κατά την διάρκεια της έξαρσης χολέρας στην περιοχή Σόχο στο Λονδίνο. Ερευνώντας λοιπόν τα μοτίβα των μολύνσεων που προέρχονταν από την χολέρα, ο John Snow κατέληξε στο συμπέρασμα του ότι η ασθένεια είχε πολύ συγκεκριμένο μοτίβο και ότι μεταδίδονταν μέσω του νερού το οποίο είχε ήδη έρθει σε επαφή και συνεπώς είχε μολυνθεί από ανθρώπους που έπασχαν ήδη από χολέρα. Τα ευρήματα του John Snow οδήγησαν στην ενδυνάμωση της μελέτης που αφορά την αναισθησία στις επεμβάσεις αλλά και σε δομικές αλλαγές σε αποχετευτικά και αρδευτικά συστήματα [6].

Δύο δεκαετίες αργότερα, ένας Βρετανός χειρουργός και πρωτεργάτης της χρήσης αντισηπτικού και απολύμανσης στα χειρουργεία που ονομάζονταν Joseph Lister, έθεσε σε εφαρμογή την ιδεολογία του αποστειρωμένου χειρουργικού περιβάλλοντος καθώς εργαζόταν στο Βασιλικό Ιατρείο της Γλασκόβης. Η χρήση του φαινικού οξέος γνωστό και ως φαινολικό οξύ για την αποστείρωση χειρουργικών εργαλείων αλλά και πληγών του σώματος, αποτέλεσε μια πρωτοποριακή εξέλιξη η οποία οδήγησε σε τεράστια μείωση της θνησιμότητας από σήψη στα χειρουργεία [7].

Το 1882, ο Robert Koch, ένας Γερμανός φυσικός και μικροβιολόγος ο οποίος θεωρείται ένας από του ιδρυτές της σύγχρονης μικροβιολογίας, κατάφερε να εντοπίσει τους συγκεκριμένους αιτιολογικούς παράγοντες της φυματίωσης, της χολέρας και του άνθρακα. Σε μεταγενέστερο στάδιο, ο Robert Koch έδωσε επίσης πειραματική υποστήριξη για την έννοια των μολυσματικών ασθενειών [8].

Ένα από τα βασικά έργα στην επιδημιολογία είναι το αυτό που του Anderson Gray McKendrick και του William Ogilvy Kermack που δημοσιεύτηκε το 1927. Ο McKendrick ήταν ένας Σκοτσέζος στρατιωτικός γιατρός και επιδημιολόγος που πρωτοστάτησε την χρήση μαθηματικών μεθόδων στην επιδημιολογία. Ο Kermack ήταν ένας Σκοτσέζος βιοχημικός ο οποίος διεξήγαγε μαθηματικές μελέτες σε επιδημιολογικές εξαπλώσεις μολυσματικών ασθενειών και καθιέρωσε συνδέσμους μεταξύ περιβαλλοντολογικών παραγόντων και συγκεκριμένων ασθενειών [9]. Μαζί μελέτησαν ένα ντετερμινιστικό επιδημιολογικό μοντέλο και δημιούργησαν μια εξίσωση για το τελικό επιδημικό μέγεθος, το οποίο δίνει έμφαση σε ένα ορισμένο όριο για την πυκνότητα ενός πληθυσμού. Μεγάλες επιδημίες μπορεί να εμφανιστούν παραπάνω αλλά όχι κάτω από αυτό το όριο [10]. Πιο συγκεκριμένα, ο McKendrick και ο Kermack έφτιαξαν ένα διαμερισματικό επιδημιολογικό μοντέλο όπου ένα ή περισσότερα μολυσμένα άτομα εισάγονται σε μια κοινότητα η οποία αποτελείται από άτομα τα οποία είναι ευπαθή στην εν λόγω ασθένεια. Η ασθένεια εξαπλώνεται από τους προσβεβλημένους στους μη επηρεασμένους μέσω μιας μολυσματικής επαφής. Έπειτα κάθε μολυσμένο άτομο διατρέχει την πορεία της ασθένειάς του, και τελικά αφαιρείται από τον αριθμό του μολυσμένου πληθυσμού μέσω ανάρρωσης ή θανάτου. Αυτό το μοντέλο ονομάζεται μοντέλο SIR και αποτελεί την βάση όλων τον υπόλοιπων μοντέλων που θα αναφερθούν στην ενότητα των διαμερισματικών μοντέλων [11].

Σχετικά Έργα

Στις προσομοιώσεις μολυσματικών ασθενειών χρησιμοποιούνται συνήθως μαθηματικά μοντέλα προκειμένου να εκτιμηθεί η πιθανότητα της λοίμωξης ή η ταχύτητα εξάπλωσης. Ο στόχος των προσομοιώσεων αυτών είναι η απεικόνιση της συνολικής πορείας μια μολυσματικής ασθένειας. Αυτή την στιγμή υπάρχουν αρκετά εργαλεία που μπορούν να χρησιμοποιηθούν προκειμένου να μελετηθεί η εξέλιξη μιας μολυσματικής ασθένειας και να γίνει κατανοητή η δυναμική των μαθηματικών μοντέλων που χρησιμοποιούνται στον τομέα της επιδημιολογίας.

3.1 Epidemix

Το Epidemix είναι μια διαδραστική εφαρμογή πολλαπλών μοντέλων για διδασκαλία και οπτικοποίηση της μετάδοσης μολυσματικών ασθενειών [12]. Το Epidemix αναπτύχθηκε για να είναι διαθέσιμο μέσω του διαδικτύου με ένα τυπικό πρόγραμμα περιήγησης χωρίς την εγκατάσταση ειδικού λογισμικού. Το RStudio Shiny [13] επιλέχθηκε ως το framework βασικής ανάπτυξης, το οποίο επέτρεψε τη δημοσίευση και την χρήση μέσω του διαδικτύου καθώς επίσης και την δυνατότητα χρήσης σε κατάσταση εκτός σύνδεσης (off-line mode). Τα μαθηματικά του μοντέλα μεταφράστηκαν σε κώδικα με την χρήση της γλώσσας προγραμματισμού R. Έπειτα κάθε μοντέλο συμπεριλήφθηκε σε συναρτήσεις της γλώσσας R και έγιναν διαθέσιμες μέσω του διακομιστή της RStudio Shiny εφαρμογής. Για την ενεργοποίηση κάποιας περαιτέρω λειτουργικότητας αλλά και οπτικών χαρακτηριστικών του πυρήνα του RStudio Shiny, ο πηγαίος κώδικας επεκτάθηκε χρησιμοποιώντας JavaScript και CSS.

3.2 GLEAMviz

Το GLEAMviz είναι ένα σύστημα λογισμικού δημόσια διαθέσιμο, το οποίο προσομοιώνει την εξάπλωση των αναδυόμενων από άνθρωπο σε άνθρωπο ασθένειες σε ολόκληρο τον κόσμο. Το GLEAMviz μπορεί να χρησιμοποιήθει ως desktop εφαρμογή με την οποία οι χρήστες αλληλεπιδρούν με τον διακομιστή GLEAM [14]. Παρέχει έναν απλό, διαισθητικό και οπτικό τρόπο για τη δημιουργία προσομοιώσεων, την ανάπτυξη μοντέλων μολυσματικών ασθενειών και την αξιολόγηση αποτελεσμάτων προσομοίωσης χρησιμοποιώντας μια ποικιλία χαρτών, διαγραμμάτων αλλά και εργαλείων ανάλυσης δεδομένων. Ο πυρήνας του είναι γραμμένος σε C++ και ο wrapper κώδικας του είναι γραμμένος με Python.

3.3 EpiSimS

Το EpiSimS είναι μια εφαρμογή C++ που τρέχει σε υψηλών επιδόσεων συστάδες υπολογιστών. Είναι βασισμένο σε στοχαστικό παράγοντα διακριτού μοντέλου συμβάντων που αντιπροσωπεύει ρητά κάθε άτομο σε μια πόλη, και κάθε μέρος της πόλης όπου αλληλεπιδρούν οι άνθρωποι. Το μοντέλο EpiSimS αναπτύχθηκε από το Los Alamos National Laboratory και χρησιμοποιεί πάνω από 18 εκατομμύρια συνθετικά άτομα σε 15 περιοχές των Ηνωμένων Πολιτειών [15]. Το μοντέλο δραστηριότητας κάθε ατόμου προέρχεται από το TRANSIMS σύστημα προσομοίωσης μεταφορών. Το TRANSIMS είναι ένα ολοκληρωμένο σύνολο εργαλείων για τη διενέργεια αναλύσεων περιφερειακών συστημάτων

μεταφοράς. Βασίζεται σε έναν προσομοιωτή αυτόματων κυψελοειδών και χρησιμοποιείται από το Υπουργείο Μεταφορών των Ηνωμένων Πολιτειών Αμερικής για ανάλυση της κυκλοφοριακής κίνησης [16].

3.4 FluPhone

Το FluPhone είναι μια εφαρμογή για κινητά που χρησιμοποιεί Bluetooth, δεδομένα συντονισμού GPS και αυτό-αναφερόμενα συμπτώματα γρίπης από το χρήστη, προκειμένου να εντοπίσει άλλες κοντινές συσκευές και να στείλει σχετικές πληροφορίες πίσω στον διακομιστή. Χρησιμοποιώντας τις πληροφορίες που ανακτήθηκαν στον διακομιστή, το FluPhone είναι επίσης σε θέση να προσομοιώσει την έξαρση μιας ασθένειας. Η κύρια λειτουργία του FluPhone είναι ένα απλό λογισμικό πελάτη-διακομιστή που αποτελείται από μια εφαρμογή κινητού τηλεφώνου στο τηλέφωνο και έναν δέκτη ως PHP script στον διακομιστή. Η εφαρμογή του τηλεφώνου είναι ανεπτυγμένη σε Java (J2ME). Ο μόνος περιορισμός που προκύπτει από τις τεχνολογίες που χρησιμοποιούνται είναι το μέγιστο φυσικό εύρος που υποστηρίζει το Bluetooth [17].

3.5 Sispread

Το Sispread είναι ένα λογισμικό που χρησιμοποιείται για την προσομοίωση της εξάπλωσης μολυσματικών ασθενειών στα δίκτυα επαφών. Το βασικό πρόγραμμα είναι εξ ολοκλήρου γραμμένο σε C, και διανέμεται ως ένα μικρό πακέτο λογισμικού. Το λογισμικό Sispread σχεδιάστηκε για την προσομοίωση της εξέλιξης των μολυσματικών ασθενειών που εξαπλώνονται στα δίκτυα επαφών, ανεξάρτητα από την τοπολογία στην οποία βασίζεται το κάθε δίκτυο. Αποτελείται από δύο κύρια συστατικά. Το πρώτο αναλαμβάνει την εγκατάσταση του δικτύου, φορτώνοντας ένα υπάρχον αρχείο ή φτιάχνοντας ένα καινούργιο. Το δεύτερο, τρέχει τις προσομοιώσεις επιδημίας [18]. Αξίζει να σημειωθεί ότι το Sispread είναι αρκετά παρόμοιο εργαλείο με το BVS και ότι και τα δύο συνδυάζουν την θεωρία των γράφων με την επιδημιολογία με σκοπό να προβλέψουν την πορεία μιας μολυσματικής ασθένειας.

Σε γενικές γραμμές, τα υπολογιστικά μοντέλα παίζουν έναν πολύ σημαντικό ρόλο στην επιδημιολογία. Αυτά τα μοντέλα επικεντρώνονται στα διαμερισματικά μοντέλα, όπου ο πληθυσμός χωρίζεται σε συγκεκριμένα υπό-τμήματα σύμφωνα με τα δημογραφικά στοιχεία, την κατάσταση υγείας των ανθρώπων και η δυναμική της επιδημίας διαμορφώνεται από διαφορικές εξισώσεις [11]. Το BVS κληρονομεί τα υπολογιστικά χαρακτηριστικά του και από τα επιδημιολογικά διαμερισματικά μοντέλα αλλά και από τα πιο γνωστά μοντέλα της θεωρίας των γράφων. Με αυτόν τον τρόπο παρέχει αξιόπιστους αλγορίθμους οι οποίοι μπορούν να χρησιμοποιηθούν με σκοπό την ανάλυση της πορείας μιας μολυσματικής ασθένειας.

Διαμερισματικά μοντέλα

Από το πρώτο ακόμη διαμερισματικό μοντέλο που δημιουργήθηκε το 1927, έχει μεσολαβήσει αρκετή ανάπτυξη και πρόοδος. Πιο συγκεκριμένα, καθώς οι μολυσματικές ασθένειες συνέχισαν να αναδύονται και να μεταλλάσσονται, τα διαμερισματικά αυτά μοντέλα έπρεπε κάπως να ακολουθήσουν τον αγώνα αυτόν και προσαρμοστούν με βάση την συμπεριφορά των ασθενειών. Συνεπώς ήταν αναμενόμενο, τα υπάρχοντα μοντέλα να προσαρμοστούν και οι διαφορικές τους εξισώσεις να αποκτήσουν αρκετές παραλλαγές, προκειμένου να συμμορφωθούν με τη δυναμική των μολυσματικών ασθενειών. Αξίζει επίσης να σημειωθεί, ότι πέρα από το να προσαρμοστούν υπάρχοντα διαμερισματικά μοντέλα, χρειάστηκε να δημιουργηθούν ακόμη περισσότερα τα οποία όλα τους προέρχονται από το μοντέλο SIR.

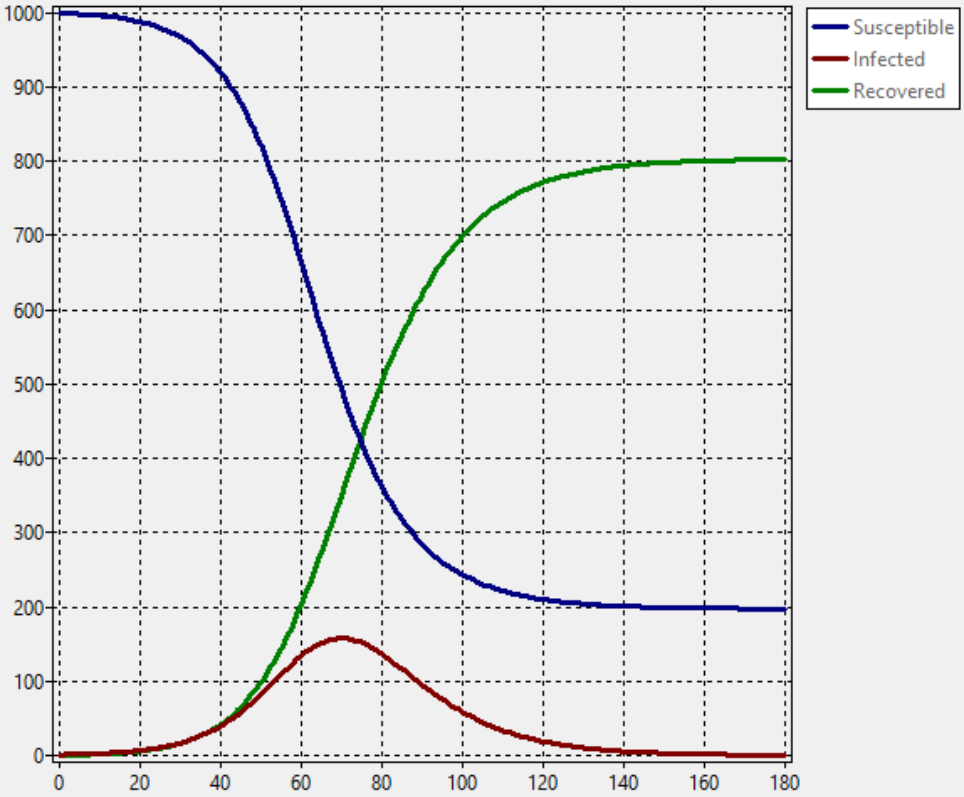
4.1 SIR

Το μοντέλο SIR παρουσιάστηκε αρχικά από τον McKendrick και τον Kermack στις αρχές του 20ου αιώνα. Είναι ένα από τα πιο απλά διαμερισματικά μοντέλα και πολλά άλλα κληρονομούν από αυτό. Το μοντέλο SIR αποτελείται από τρία διαμερίσματα (groups , ομάδες). Το πρώτο group είναι το group του ευαίσθητου πληθυσμού (Susceptible), όπου σε αυτό ανήκουν όσοι βρίσκονται στην αρχική τους ευάλωτη κατάσταση και δεν έχουν έρθει ακόμη σε μολυσματική επαφή με ένα μολυσματικό άτομο. Όταν τα ευάλωτα άτομα αποκτήσουν την ασθένεια, μεταφέρονται στο group αυτών που έχουν μολυνθεί (Infectious). Το group αυτό αποτελείτε από μολυσμένα άτομα τα οποία έχουν την δυνατότητα πλέον, να μολύνουν άλλα ευάλωτα άτομα. Τέλος το group αυτών που έχουν περάσει την ασθένεια αποτελείτε από άτομα τα οποία αφού μολύνθηκαν, ανάρρωσαν από την ασθένεια ή απεβίωσαν από αυτή (Recovered / Removed) [11]. Το μοντέλο SIR μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\frac{dS}{dt} = -\frac{\beta \cdot I \cdot S}{N}$$

$$\frac{dI}{dt} = \frac{\beta \cdot I \cdot S}{N} - \gamma \cdot I \tag{4.1}$$

$$\frac{dR}{dt} = \gamma \cdot I$$



Εικόνα 4.1: BVS | N: 1000, β: 0.2, γ: 0.1, Initial Infected: 1, Days: 180

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.

Μια από τις πιο σημαντικές παραμέτρους των διαμερισματικών μοντέλων είναι ο βασικός αριθμός αναπαραγωγής (basic reproductive number) ή αλλιώς R_0. Το R_0 είναι ο μέσος αριθμός των ατόμων που μολύνθηκαν από ένα περιστατικό σε έναν εντελώς ευάλωτο πληθυσμό με την απουσία παρεμβάσεων οι οποίες αποσκοπούν να ελέγξουν την πορεία της μόλυνσης. Όσον αφορά το μοντέλο SIR, η μαθηματική φόρμουλα του βασικού αριθμού αναπαραγωγής είναι $R_0 = \frac{\beta S}{\gamma}$. Το R_0 είναι ιδιαίτερα σημαντικό διότι υποδεικνύει το αν θα προκύψει μια επιδημία ή όχι. Εάν $R_0 > 1$, τότε θα προκύψει επιδημία, αλλά εάν $R_0 < 1$ τότε δεν θα προκύψει καμία επιδημία [19].

4.1.1 Αλγόριθμος SIR

```
unit utlSIR_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma: Double;
    S, I: Double;
    dSdt, dIdt, dRdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the SIR's (Susceptible - Infectious - Recovered) model
   differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' S I R
   states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                        0    1     2
                       [N, beta, gamma]

      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                      (Infection rate)
      Var gamma: The recovery rate.

      Var S: Number of Susceptible individuals.
      Var I: Number of Infected individuals.

      Var dSdt: Calculated differential equation result for
   Susceptibles.
      Var dIdt: Calculated differential equation result for Infected.
      Var dRdt: Calculated differential equation result for Recovered.

```

```
54        [*****************************************************************]
55
56      }
57
58      S := y[0];
59      I := y[1];
60
61      N := extraArgs[0];
62      beta := extraArgs[1];
63      gamma := extraArgs[2];
64
65      dSdt := -(beta * S * I / N);
66      dIdt := (beta * S * I / N) - (gamma * I);
67      dRdt := gamma * I;
68
69      Result := ArrayOfDouble.Create(dSdt, dIdt, dRdt);
70    end;
71
72  end.
```

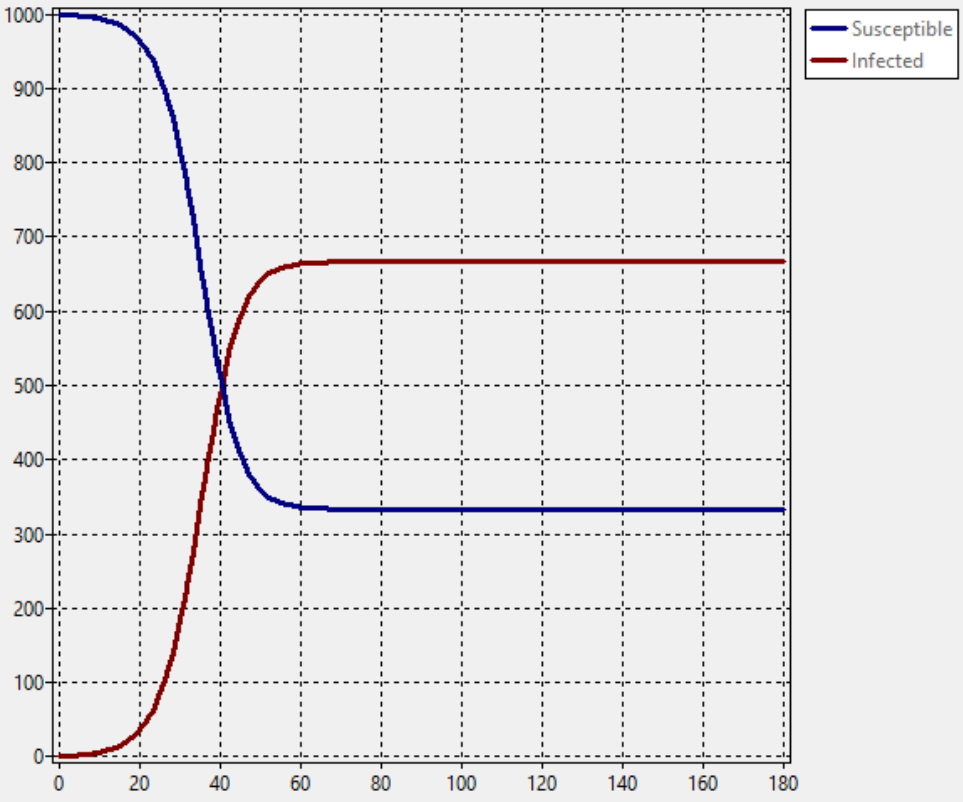
4.2 SIS

Παρόμοια με το μοντέλο SIR, το SIS αποτελείται από τα διαμερίσματα ευαίσθητων (Susceptible) και μολυσμένων (Infectious). Η μόνη διαφορά μεταξύ τους είναι ότι όταν ένα μολυσμένο άτομο ξεπεράσει την ασθένεια, αντί να μετακινηθεί στο διαμέρισμα οπού βρίσκονται αυτοί οι οποίοι έχουν ξεπεράσει την ασθένεια (Recovered), μεταφέρεται για άλλη μια φορά στο group με τα ευπαθή άτομα (Susceptible). Αυτό μπορεί να συμβεί λόγω του ότι ορισμένες μολυσματικές ασθένειες, όπως για παράδειγμα το κοινό κρυολόγημα και η γρίπη, δεν παρέχουν μακροχρόνια και διαρκή ανόσια [20] [21]. Το μοντέλο SIR μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\begin{aligned} \frac{dS}{dt} &= -\frac{\beta \cdot I \cdot S}{N} + \gamma \cdot I \\ \frac{dI}{dt} &= \frac{\beta \cdot I \cdot S}{N} - \gamma \cdot I \end{aligned} \tag{4.2}$$

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.



Εικόνα 4.2: BVS | N: 1000, β: 0.3, γ: 0.1, Initial Infected: 1, Days: 180

4.2.1 Αλγόριθμος SIS

```
unit utlSIS_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma: Double;
    S, I: Double;
    dSdt, dIdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the SIS's (Susceptible - Infectious - Susceptible)
   model differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' S I
   states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                        0    1     2
                       [N, beta, gamma]

      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                      (Infection rate)
      Var gamma: The recovery rate.

      Var S: Number of Susceptible individuals.
      Var I: Number of Infected individuals.

      Var dSdt: Calculated differential equation result for
   Susceptibles.
      Var dIdt: Calculated differential equation result for Infected.

      [*************************************************************]
```

```
54
55      }
56
57      S := y[0];
58      I := y[1];
59
60      N := extraArgs[0];
61      beta := extraArgs[1];
62      gamma := extraArgs[2];
63
64      dSdt := -(beta * S * I / N) + (gamma * I);
65      dIdt :=  (beta * S * I / N) - (gamma * I);
66
67      Result := ArrayOfDouble.Create(dSdt, dIdt);
68    end;
69
70  end.
```

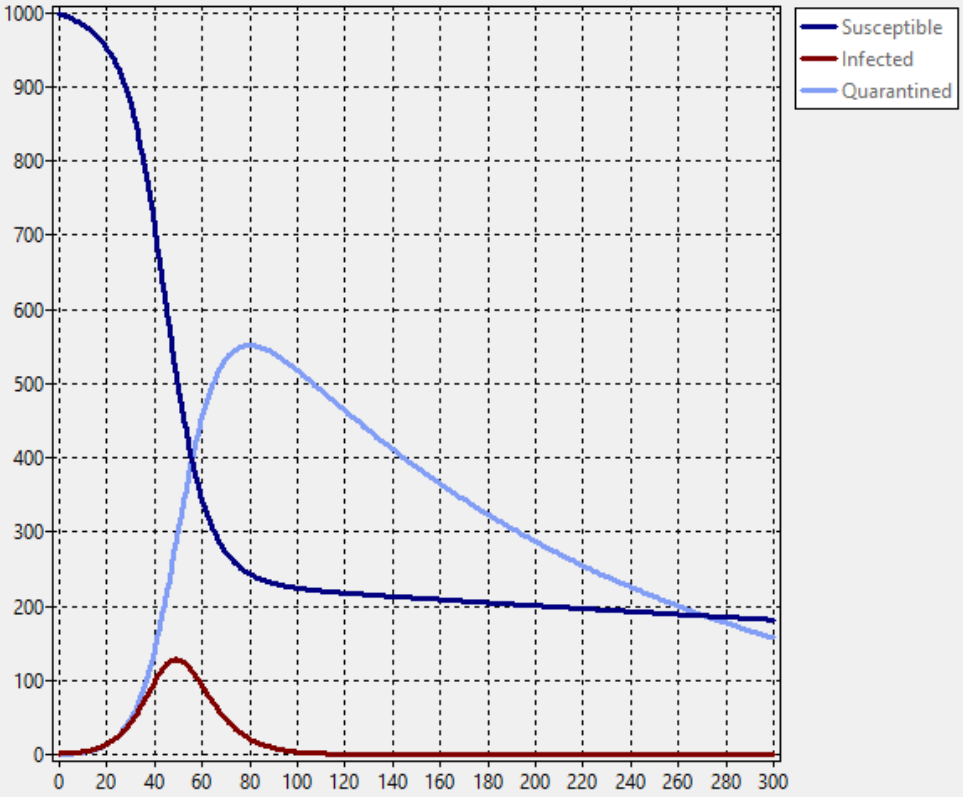
4.3 SIQ

Μία από τις πιο ώριμες αλλά αποτελεσματικές μεθόδους με σκοπό τον έλεγχο της εξάπλωσης των μολυσματικών ασθενειών είναι η απομόνωση ορισμένων μολυσμένων ατόμων που έχει συνήθως ως αποτέλεσμα την μείωση των μεταδόσεων μιας μολυσματικής ασθένειας στα ευάλωτα άτομα. Η απομόνωση θεωρείται ως η πρώτη μέθοδος ελέγχου μιας μολυσματικής ασθένειας. Η λέξη καραντίνα αρχικά αντιστοιχούσε σε μια περίοδο σαράντα ημερών, δηλαδή το χρονικό διάστημα κατά το οποίο έφταναν τα πλοία που ήταν ύποπτα για πανούκλα ή μόλυνση και περιορίζονταν από τη επαφή με λιμάνια της Μεσογείου τον 15-19ο αιώνα. Η λέξη καραντίνα έχει, σημαίνει αναγκαστική απομόνωση ή διακοπή αλληλεπιδράσεων με άλλους [22]. Το μοντέλο SIQ προέρχεται από το μοντέλο SIR με τη μόνη διαφορά του ότι αντί να αναρρώνει ένα άτομο από την ασθένεια ενώ είναι μολυσμένο, υφίσταται ένα ορισμένο χρονικό διάστημα σε μια κατάσταση όπου βρίσκονται σε καραντίνα. Τα άτομα που ανήκουν στο group καραντίνας (Quarantined) είναι εντελώς απομονωμένα και έχουν μηδενικές αλληλεπιδράσεις με άλλους [23]. Το μοντέλο SIQ μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\frac{dS}{dt} = \Lambda - \frac{\beta \cdot I \cdot S}{N} - \mu \cdot S$$

$$\frac{dI}{dt} = \Lambda + \frac{\beta \cdot I \cdot S}{N} - (\mu + \delta + \kappa) \cdot I \tag{4.3}$$

$$\frac{dQ}{dt} = \kappa \cdot I - (\mu + \delta) \cdot Q$$



Εικόνα 4.3: BVS | N: 1000, β: 0.3, μ: 0.001, Λ: 0, δ: 0.005, κ: 0.15, Initial Infected: 1, Days: 300

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- δ είναι ο ρυθμός θνησιμότητας της ασθένειας.
- κ είναι ο ρυθμός όπου τα μολυσμένα άτομα περνούν σε καραντίνα.

4.3.1 Αλγόριθμος SIQ

```
unit utlSIQ_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SIQDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SIQDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, mu, lambda, delta, kappa: Double;
    S, I, Q: Double;
    dSdt, dIdt, dQdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the SIQ's (Susceptible - Infectious - Quarantined)
   model differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' S I Q
   states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                        0    1   2     3       4      5
                       [N, beta, mu, lambda, delta, kappa]

      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                      (Infection rate)
      Var mu: Natural mortality rate.
      Var lambda: Recruitment of the susceptible individuals (birth etc
   .)
      Var delta: Disease mortality rate.
      Var Kappa: The rate where the infected individuals become
   quarantined.

      Var S: Number of Susceptible individuals.
      Var I: Number of Infected individuals.
      Var Q: Number of Quarantined individuals.
```

```
53
54      Var dSdt: Calculated differential equation result for
    Susceptibles.
55      Var dIdt: Calculated differential equation result for Infected.
56      Var dQdt: Calculated differential equation result for Quarantined
    .
57
58      [************************************************************]
59
60    }
61
62    S := y[0];
63    I := y[1];
64    Q := y[2];
65
66    N := extraArgs[0];
67    beta := extraArgs[1];
68    mu := extraArgs[2];
69    lambda := extraArgs[3];
70    delta := extraArgs[4];
71    kappa := extraArgs[5];
72
73    dSdt := lambda - (beta * S * I / N) - (mu * S);
74    dIdt := lambda + (beta * S * I / N) - ((mu + delta + kappa) * I);
75    dQdt := (kappa * I) - ((mu + delta) * Q);
76
77    Result := ArrayOfDouble.Create(dSdt, dIdt, dQdt);
78  end;
79
80 end.
```

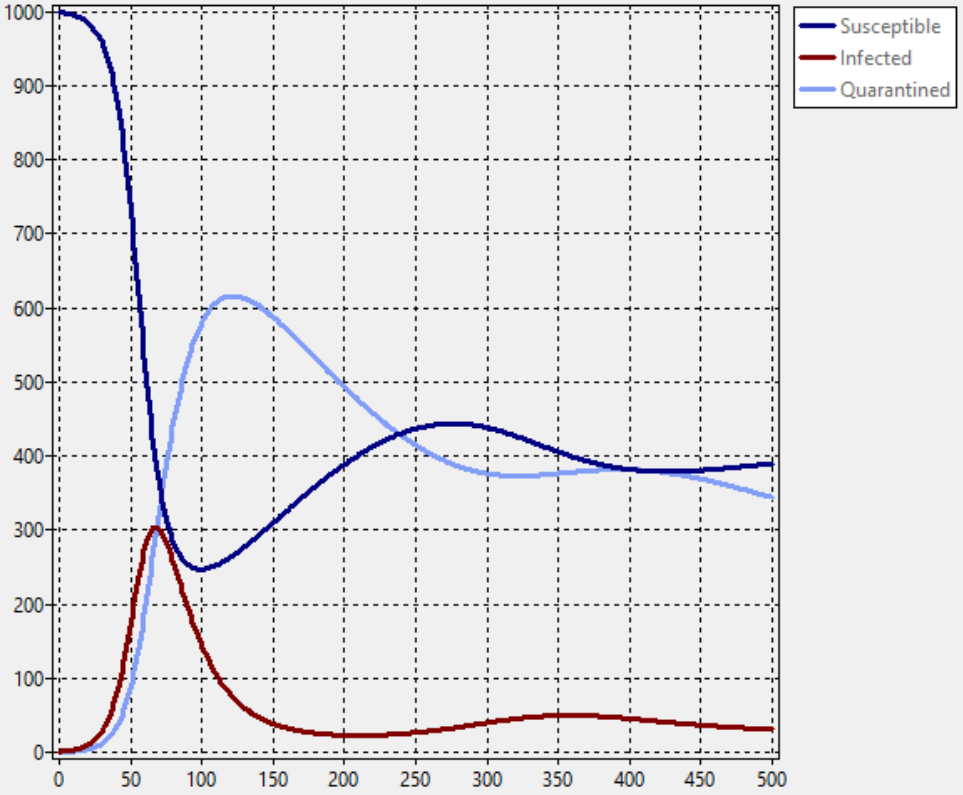
4.4 SIQS

Το μοντέλο SIQS είναι πολύ παρόμοιο με το SIQ, με την μόνη διαφορά όπου μετά από μια ορισμένη περίοδο καραντίνας, τα άτομα επιστρέφουν στην κατάσταση όπου είναι και πάλι ικανά να προσβληθούν από την ασθένεια (Susceptible) [24] [25]. Μια απλοποιημένη μορφή των διαφορικών εξισώσεων του μοντέλου SIQS είναι:

$$\frac{dS}{dt} = \Lambda - \frac{\beta \cdot I \cdot S}{N} - (\mu \cdot S) + (\gamma \cdot I) + (\zeta \cdot Q)$$

$$\frac{dI}{dt} = \frac{\beta \cdot I \cdot S}{N} - (\mu + \delta + \kappa + \gamma) \cdot I \tag{4.4}$$

$$\frac{dQ}{dt} = \kappa \cdot I - (\mu + \delta 1 + \zeta) \cdot Q$$



Εικόνα 4.4: BVS | N: 1000, β: 0.3, γ: 0.03, μ: 0.0001, Λ: 0.01, δ: 0.0001, $\delta 1$: 0.001, κ: 0.05, ζ: 0.005, Initial Infected: 1, Days: 500

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.

- γ είναι ο ρυθμός ανάρρωσης.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- δ είναι ο ρυθμός θνησιμότητας της ασθένειας.
- $\delta 1$ είναι ο ρυθμός θνησιμότητας της ασθένειας, για όσους βρίσκονται ήδη σε καραντίνα.
- κ είναι ο ρυθμός όπου τα μολυσμένα άτομα περνούν σε καραντίνα.
- ζ είναι ο ρυθμός με τον οποίο τα άτομα που βρίσκονται σε καραντίνα, επιστρέφουν σε κατάσταση όπου είναι και πάλι ευάλωτα στην ασθένεια.

4.4.1 Αλγόριθμος SIQS

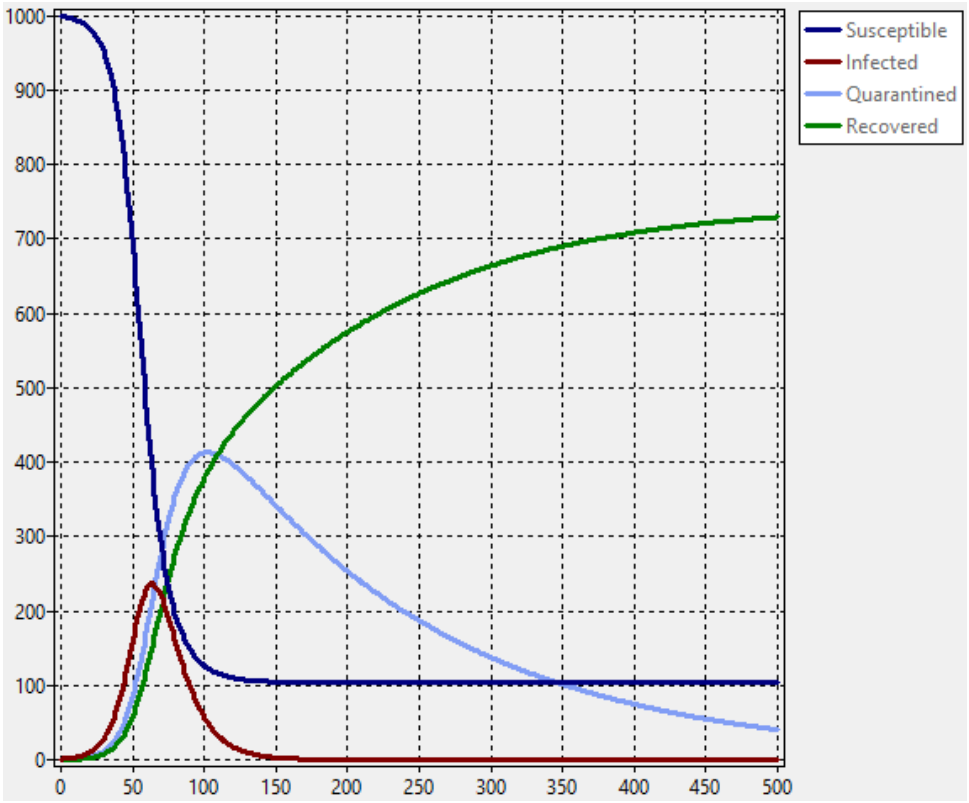
```
unit utlSIQS_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SIQSDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SIQSDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu, lambda, delta, delta1, kappa, zeta: Double;
    S, I, Q: Double;
    dSdt, dIdt, dQdt: Double;
  begin
    {
      [ Description ]
      [************************************************************]
      Calculate the SIQ's (Susceptible - Infectious - Quarantined -
   Susceptible) model differential equations given a y state.

      [************************************************************]

      [ Parameters ]
      [************************************************************]
      Param y: Array of double containing the 'to be calculated' S I Q
   states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:

                0    1     2    3     4       5      6      7     8
               [N, beta, gamma, mu, lambda, delta, delta1 kappa, zeta]
```

```
37
38      [*****************************************************************]
39
40      [ Variables ]
41      [*****************************************************************]
42      Var N: The total population.
43
44      Var beta: The average number of contacts per person per time.
45                    (Infection rate)
46      Var gamma: The recovery rate.
47      Var mu: Natural mortality rate.
48      Var lambda: Recruitment of the susceptible individuals (birth etc
    .)
49      Var delta: Disease mortality rate of infectious individuals.
50      Var delta1: Disease mortality rate of quarantined individuals.
51      Var Kappa: The rate where the infected individuals become
    quarantined.
52      Var Zeta: The rate where quarantined individuals become
    susceptible again.
53
54      Var S: Number of Susceptible individuals.
55      Var I: Number of Infected individuals.
56      Var Q: Number of Quarantined individuals.
57
58      Var dSdt: Calculated differential equation result for
    Susceptibles.
59      Var dIdt: Calculated differential equation result for Infected.
60      Var dQdt: Calculated differential equation result for Quarantined
    .
61
62      [*****************************************************************]
63
64     }
65
66     S := y[0];
67     I := y[1];
68     Q := y[2];
69
70     N := extraArgs[0];
71     beta := extraArgs[1];
72     gamma := extraArgs[2];
73     mu := extraArgs[3];
74     lambda := extraArgs[4];
75     delta := extraArgs[5];
76     delta1 := extraArgs[6];
77     kappa := extraArgs[7];
78     zeta := extraArgs[8];
79
80     dSdt := lambda - (beta * S * I / N) - (mu * S) + (gamma * I) + (
    zeta * Q);
81     dIdt := (beta * S * I / N) - ((mu + delta + kappa + gamma) * I);
82     dQdt := (kappa * I) - ((mu + delta1 + zeta) * Q);
83
84     Result := ArrayOfDouble.Create(dSdt, dIdt, dQdt);
85   end;
86
87 end.
```

4.5 SIQR

Το μοντέλο επιδημίας SIQR μοιάζει περισσότερο με το SIQS μόνο που στο τέλος, αντί να γίνουν ξανά τα άτομα ευάλωτα στην ασθένεια, αναπτύσσουν ανόσια σε αυτήν και έτσι μετακινούνται στο group με τα άτομα που έχουν περάσει την ασθένεια (Recovered) [26]. Μια απλοποιημένη μορφή των διαφορικών εξισώσεων του μοντέλου SIQR είναι:

$$\begin{aligned}
\frac{dS}{dt} &= \Lambda - \frac{\beta \cdot I \cdot S}{N} - (\mu \cdot S) \\
\frac{dI}{dt} &= \frac{\beta \cdot I \cdot S}{N} - (\mu + \delta + \kappa + \gamma) \cdot I \\
\frac{dQ}{dt} &= \kappa \cdot I - (\mu + \delta 1 + \zeta) \cdot Q \\
\frac{dR}{dt} &= (\gamma \cdot I) + (\zeta \cdot Q) - (\mu \cdot R)
\end{aligned} \tag{4.5}$$



Εικόνα 4.5: BVS | N: 1000, β: 0.3, γ: 0.03, μ: 0.0001, Λ: 0.01, δ: 0.0001, $\delta 1$: 0.001, κ: 0.05, ζ: 0.005, Initial Infected: 1, Days: 500

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- δ είναι ο ρυθμός θνησιμότητας της ασθένειας.
- $\delta 1$ είναι ο ρυθμός θνησιμότητας της ασθένειας, για όσους βρίσκονται ήδη σε καραντίνα.
- κ είναι ο ρυθμός όπου τα μολυσμένα άτομα περνούν σε καραντίνα.
- ζ είναι ο ρυθμός με τον οποίο τα άτομα που βρίσκονται σε καραντίνα, επιστρέφουν σε κατάσταση όπου είναι και πάλι ευάλωτα στην ασθένεια.

4.5.1 Αλγόριθμος SIQR

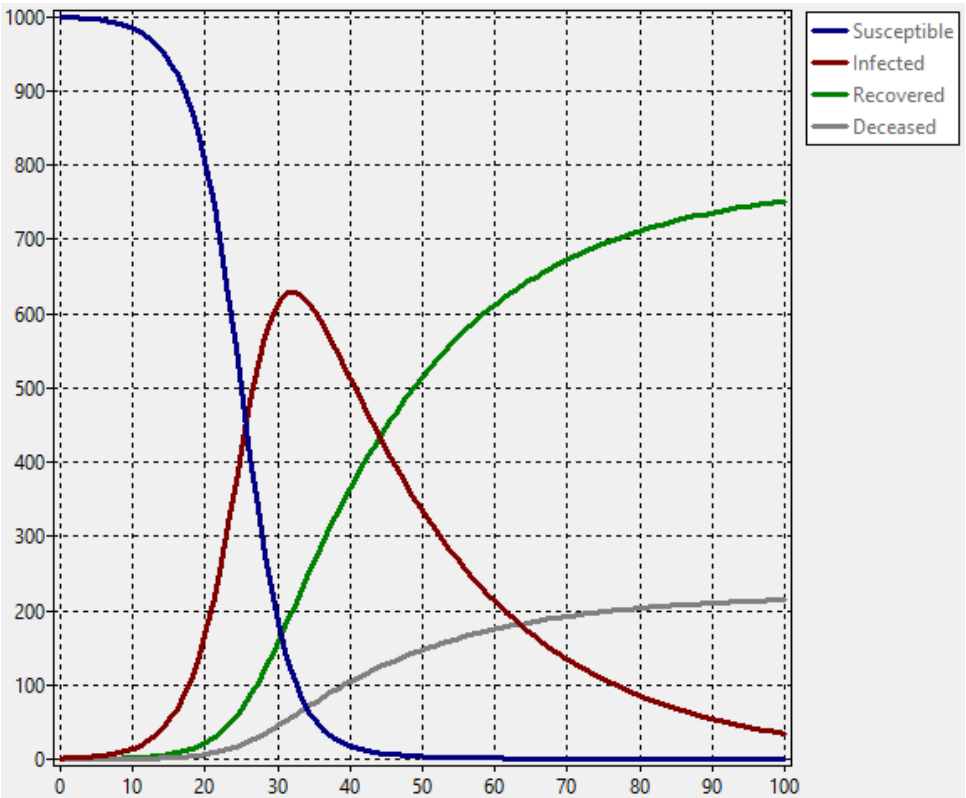
```
unit utlSIQR_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SIQRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SIQRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu, lambda, delta, delta1, kappa, zeta: Double;
    S, I, Q, R: Double;
    dSdt, dIdt, dQdt, dRdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the SIQR's (Susceptible - Infectious - Quarantined -
   Recovered) model differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' S I Q
   R states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                0    1     2    3      4      5      6       7     8
               [N, beta, gamma, mu, lambda, delta, delta1, kappa, zeta]

      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                     (Infection rate)
      Var gamma: The recovery rate.
      Var mu: Natural mortality rate.
      Var lambda: Recruitment of the susceptible individuals (birth etc
   .)
      Var delta: Disease mortality rate of infectious individuals.
      Var delta1: Disease mortality rate of quarantined individuals.
      Var Kappa: The rate where the infected individuals become
   quarantined.
      Var Zeta: The rate where quarantined individuals become
   susceptible again.
```

```
52
53      Var S: Number of Susceptible individuals.
54      Var I: Number of Infected individuals.
55      Var Q: Number of Quarantined individuals.
56      Var R: Number of Recovered individuals.
57
58      Var dSdt: Calculated differential equation result for
    Susceptibles.
59      Var dIdt: Calculated differential equation result for Infected.
60      Var dQdt: Calculated differential equation result for Quarantined
    .
61      Var dRdt: Calculated differential equation result for Recovered.
62
63      [************************************************************]
64
65    }
66
67    S := y[0];
68    I := y[1];
69    Q := y[2];
70    R := y[3];
71
72    N := extraArgs[0];
73    beta := extraArgs[1];
74    gamma := extraArgs[2];
75    mu := extraArgs[3];
76    lambda := extraArgs[4];
77    delta := extraArgs[5];
78    delta1 := extraArgs[6];
79    kappa := extraArgs[7];
80    zeta := extraArgs[8];
81
82    dSdt := lambda - (beta * S * I / N) - (mu * S);
83    dIdt := (beta * S * I / N) - ((gamma + kappa + mu + delta) * I);
84    dQdt := (kappa * I) - ((zeta + mu + delta1) * Q);
85    dRdt := (gamma * I) + (zeta * Q) - (mu * R);
86
87    Result := ArrayOfDouble.Create(dSdt, dIdt, dQdt, dRdt);
88  end;
89
90 end.
```

4.6 SIRD

Το μοντέλο επιδημίας SIRD κάνει διάκριση μεταξύ αυτών που έχουν αναρρώσει από την ασθένεια και αυτών που απεβίωσαν από αυτή [27]. Σχεδόν παρόμοια με τις διαφορικές εξισώσεις του μοντέλου SIR, έτσι και οι διαφορικές εξισώσεις του μοντέλου SIRD, έχουν επεξεργαστεί ελαφρώς με σκοπό να λαμβάνουν υπό όψιν τους τον ρυθμό θνησιμότητας που οδηγεί στο group με τα άτομα που απεβίωσαν από την ασθένεια. Το μοντέλο SIRD μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\begin{aligned}
\frac{dS}{dt} &= -\frac{\beta \cdot I \cdot S}{N} \\
\frac{dI}{dt} &= \frac{\beta \cdot I \cdot S}{N} - (\gamma \cdot I) - (\mu \cdot I) \\
\frac{dR}{dt} &= \gamma \cdot I \\
\frac{dD}{dt} &= \mu \cdot I
\end{aligned} \tag{4.6}$$



Εικόνα 4.6: BVS | N: 1000, β: 0.35, γ: 0.035, μ: 0.01, Initial Infected: 1, Days: 100

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.

4.6.1 Αλγόριθμος SIRD

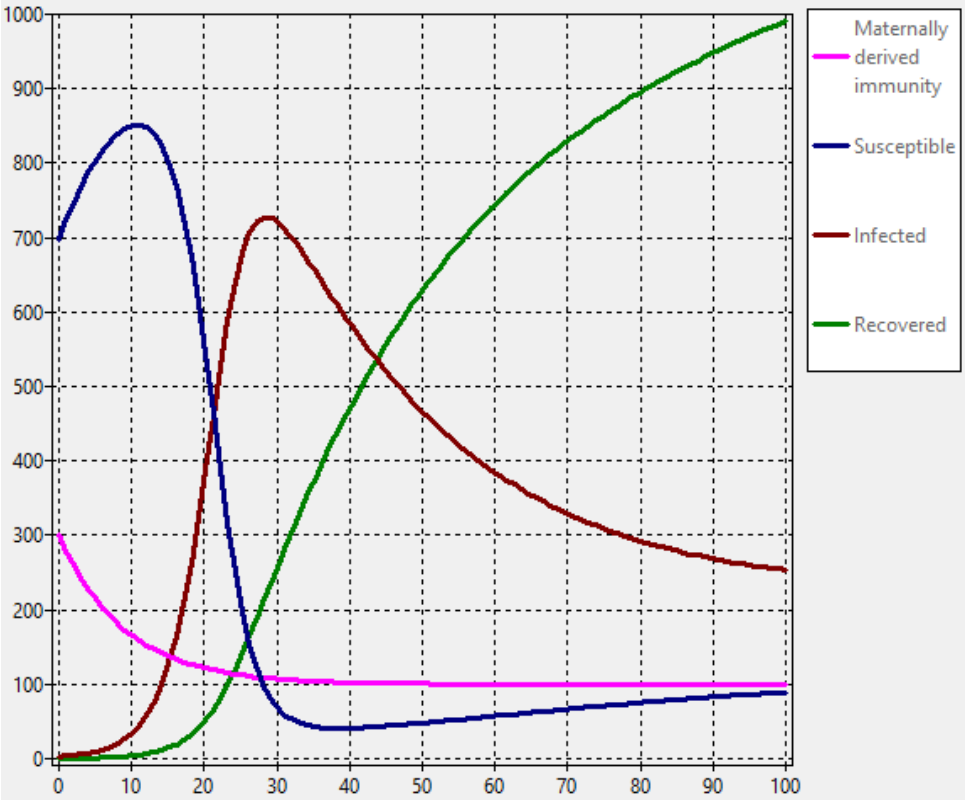
```
unit utlSIRD_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SIRDDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SIRDDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu: Double;
    S, I: Double;
    dSdt, dIdt, dRdt, dDdt: Double;
  begin
    {
      [ Description ]
      [************************************************************]
      Calculate the SIRD's (Susceptible - Infectious - Recovered -
   Deceased) model differential equations given a y state.

      [************************************************************]

      [ Parameters ]
      [************************************************************]
      Param y: Array of double containing the 'to be calculated' S I R
   D states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                        0    1     2    3
                       [N, beta, gamma, mu]

      [************************************************************]

      [ Variables ]
      [************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
```

```
                    (Infection rate)
      Var gamma: The recovery rate.
      Var mu: Natural mortality rate.

      Var S: Number of Susceptible individuals.
      Var I: Number of Infected individuals.

      Var dSdt: Calculated differential equation result for
   Susceptibles.
      Var dIdt: Calculated differential equation result for Infected.
      Var dRdt: Calculated differential equation result for Recovered.
      Var dDdt: Calculated differential equation result for Deceased.

      [************************************************************]

    }

    S := y[0];
    I := y[1];

    N := extraArgs[0];
    beta := extraArgs[1];
    gamma := extraArgs[2];
    mu := extraArgs[3];

    dSdt := -(beta * S * I / N);
    dIdt := (beta * S * I / N) - (gamma * I) - (mu * I);
    dRdt := gamma * I;
    dDdt := mu * I;

    Result := ArrayOfDouble.Create(dSdt, dIdt, dRdt, dDdt);
  end;

end.
```

4.7 MSIR

Για πολλές λοιμώξεις, συμπεριλαμβανομένης του ιλαρά, τα μωρά που γεννιούνται δεν ανήκουν στο ευαίσθητο group (Susceptible) αλλά έχουν ανοσία από την ασθένεια για τους πρώτους μήνες της ζωής λόγω προστασίας από μητρικά αντισώματα. Αυτό ονομάζεται παθητική ανοσία (passive immunity). Αυτή η προστιθέμενη λεπτομέρεια μπορεί να εκφραστεί συμπεριλαμβάνοντας μια τάξη M, η οποία αντιπροσωπεύει την μητρική ανοσία (maternally derived immunity) και βρίσκεται στην αρχή του επιδημιολογικού μοντέλου [28]. Το μοντέλο MSIR μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\begin{aligned}
\frac{dM}{dt} &= \Lambda - (\delta \cdot M) - (\mu \cdot M) \\
\frac{dS}{dt} &= (\delta \cdot M) - \frac{\beta \cdot I \cdot S}{N} - (\mu \cdot S) \\
\frac{dI}{dt} &= \frac{\beta \cdot I \cdot S}{N} - (\gamma \cdot I) - (\mu \cdot I) \\
\frac{dR}{dt} &= (\gamma \cdot I) - (\mu \cdot R)
\end{aligned} \tag{4.7}$$



Εικόνα 4.7: BVS | N: 1000, β: 0.4, γ: 0.035, μ: 0.005, Λ: 10.5, δ: 0.1, Maternally derived immunity: 300, Initial Infected: 3, Days: 100

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- δ είναι ο ρυθμός θνησιμότητας της ασθένειας.

4.7.1 Αλγόριθμος MSIR

```
unit utlMSIR_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function MSIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function MSIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu, lambda, delta: Double;
    M, S, I, R: Double;
    dMdt, dSdt, dIdt, dRdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the MSIR's
      (Maternally derived immunity - Susceptible - Infectious -
   Recovered) model differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' M S I
   R states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:

                          0    1     2    3      4      5
                         [N, beta, gamma, mu, lambda, delta]

```

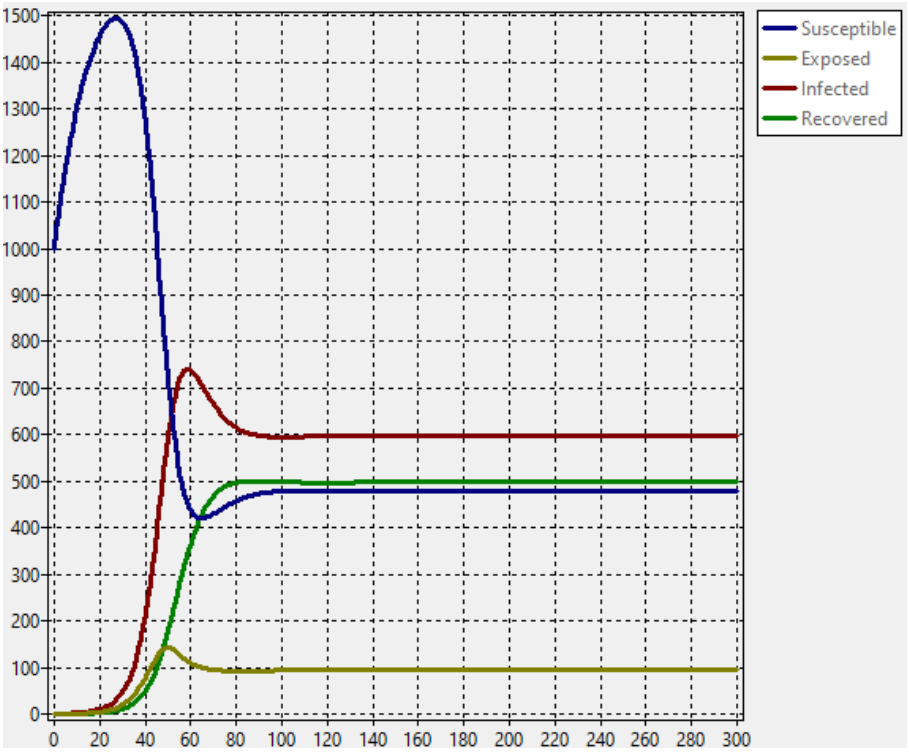
```
      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                    (Infection rate)
      Var gamma: The recovery rate.
      Var mu: Natural mortality rate.
      Var lambda: Recruitment of the susceptible individuals (birth etc
   .)
      Var delta: Disease mortality rate.

      Var M: Number of individuals with Maternally derived immunity.
      Var S: Number of Susceptible individuals.
      Var I: Number of Infected individuals.
      Var R: Number of Recovered individuals.

      Var dMdt: Calculated differential equation result for Maternally
                derived immunity.
      Var dSdt: Calculated differential equation result for
   Susceptibles.
      Var dIdt: Calculated differential equation result for Infected.
      Var dRdt: Calculated differential equation result for Recovered.

      [*************************************************************]

    }

    M := y[0];
    S := y[1];
    I := y[2];
    R := y[3];

    N := extraArgs[0];
    beta := extraArgs[1];
    gamma := extraArgs[2];
    mu := extraArgs[3];
    lambda := extraArgs[4];
    delta := extraArgs[5];

    dMdt := lambda - (delta * M) - (mu * M);
    dSdt := (delta * M) - (beta * S * I / N) - (mu * S);
    dIdt := (beta * S * I / N) - (gamma * I) - (mu * I);
    dRdt := (gamma * I) - (mu * R);

    Result := ArrayOfDouble.Create(dMdt, dSdt, dIdt, dRdt);
  end;

end.
```

4.8 SEIR

Για πολλές σημαντικές λοιμώξεις, υπάρχει σημαντική περίοδος επώασης (incubation period) κατά την οποία άτομα έχουν μολυνθεί αλλά είναι δεν είναι ακόμη μολυσματικοί. Κατά τη διάρκεια αυτής της περιόδου το άτομο βρίσκεται στο group E, το οποίο αντιπροσωπεύει τα εκτεθειμένα άτομα. Αν και το μεγαλύτερο μέρος της βιβλιογραφίας που

ασχολείται με επιδημιολογικά συστήματα θεωρεί ότι η επώαση της νόσου είναι αμελητέα, υπάρχουν πολλές ασθένειες όπως ο ιλαράς, σοβαρού οξέος αναπνευστικού συνδρόμου, επίσης γνωστό και ως SARS και ούτω καθεξής, που επωάζονται εντός των οργανισμών για ένα χρονικό διάστημα προτού αποκτήσουν την δυνατότητα εξάπλωσης της ασθένειας [29]. Το μοντέλο SEIR μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\begin{aligned}
\frac{dS}{dt} &= (\Lambda \cdot N) - (\mu \cdot S) - \frac{\beta \cdot I \cdot S}{N} \\
\frac{dE}{dt} &= \frac{\beta \cdot I \cdot S}{N} - (\mu + \alpha) \cdot E \\
\frac{dI}{dt} &= (\alpha \cdot E) - (\gamma + \mu) \cdot I \\
\frac{dR}{dt} &= (\gamma \cdot I) - (\mu \cdot R)
\end{aligned} \tag{4.8}$$



Εικόνα 4.8: BVS | N: 1000, β: 0.25, γ: 0.05, μ: 0.06, Λ: 0.1, α: 0.7, Initial Infected: 1, Days: 300

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.

- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- α είναι η μέση περίοδος επώασης (α^{-1}).

4.8.1 Αλγόριθμος SEIR

```
unit utlSEIR_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu, lambda, alpha: Double;
    S, E, I, R: Double;
    dSdt, dEdt, dIdt, dRdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the SEIR's (Susceptible - Exposed - Infectious -
   Recovered) model differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' S E I
   R states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                        0    1     2    3      4      5
                       [N, beta, gamma, mu, lambda, alpha]

      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                     (Infection rate)
      Var gamma: The recovery rate.
      Var mu: Natural mortality rate.
```

```
47        Var lambda: Recruitment of the susceptible individuals (birth etc
    .)
48        Var alpha: Average incubation period.
49
50        Var S: Number of Susceptible individuals.
51        Var E: Number of Exposed individuals.
52        Var I: Number of Infected individuals.
53        Var R: Number of Recovered individuals.
54
55        Var dSdt: Calculated differential equation result for
    Susceptibles.
56        Var dEdt: Calculated differential equation result for Exposed.
57        Var dIdt: Calculated differential equation result for Infected.
58        Var dRdt: Calculated differential equation result for Recovered.
59
60        [*************************************************************]
61
62     }
63
64     S := y[0];
65     E := y[1];
66     I := y[2];
67     R := y[3];
68
69     N := extraArgs[0];
70     beta := extraArgs[1];
71     gamma := extraArgs[2];
72     mu := extraArgs[3];
73     lambda := extraArgs[4];
74     alpha := extraArgs[5];
75
76     dSdt := (lambda * N) - (mu * S) - (beta * S * I / N);
77     dEdt := (beta * S * I / N) - ((mu + alpha) * E);
78     dIdt := (alpha * E) - ((gamma + mu) * I);
79     dRdt := (gamma * I) - (mu * R);
80
81     Result := ArrayOfDouble.Create(dSdt, dEdt, dIdt, dRdt);
82   end;
83
84 end.
```

4.9 SEIS

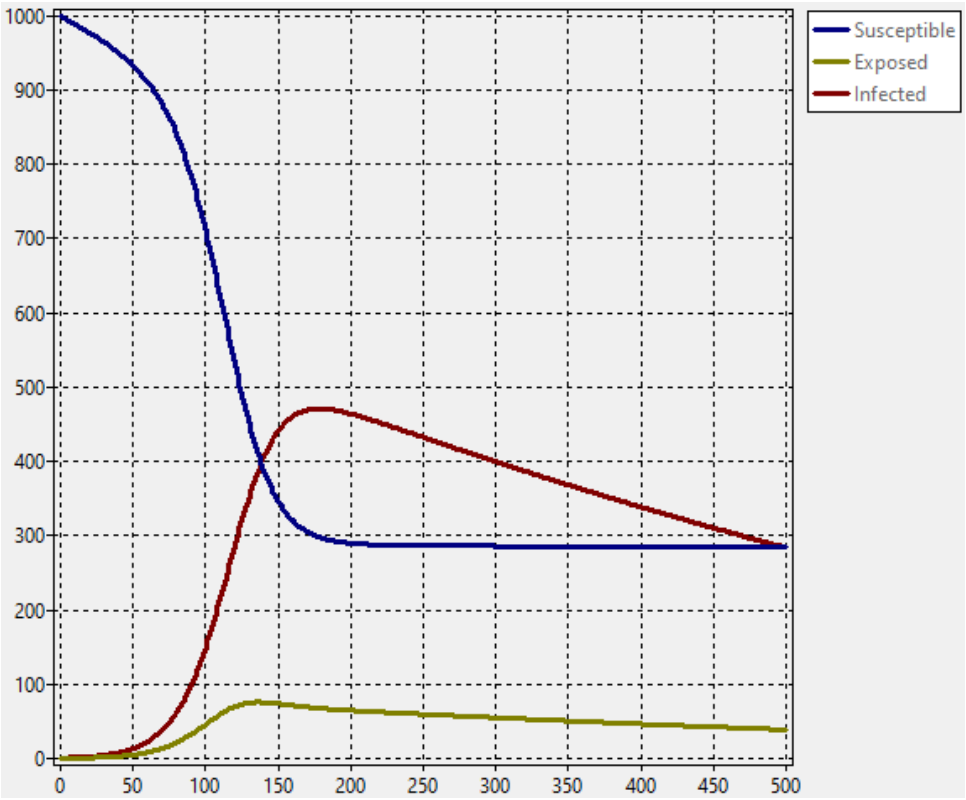
Το μοντέλο επιδημίας SEIS είναι σαν το μοντέλο SEIR με την διαφορά ότι στο τέλος δεν παρέχεται ανοσία. Αντί αυτού, τα μολυσμένα άτομα επιστρέφουν σε μια κατάσταση όπου είναι και πάλι ευπαθή στη νόσο [30]. Το μοντέλο SEIS μπορεί να περιγραφεί από τις

παρακάτω διαφορικές εξισώσεις:

$$\frac{dS}{dt} = \Lambda - \frac{\beta \cdot I \cdot S}{N} - (\mu \cdot S) + (\gamma \cdot I)$$

$$\frac{dE}{dt} = \frac{\beta \cdot I \cdot S}{N} - (\epsilon + \mu) \cdot E \tag{4.9}$$

$$\frac{dI}{dt} = (\epsilon \cdot E) - (\gamma + \mu) \cdot I$$



Εικόνα 4.9: BVS | N: 1000, β: 0.12, γ: 0.035, μ: 0.001, Λ: 0, ϵ: 0.25, Initial Infected: 1, Days: 500

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- ϵ είναι ο ρυθμός όπου τα εκτεθειμένα άτομα μετατρέπονται σε μολυσμένα άτομα.

4.9.1 Αλγόριθμος SEIS

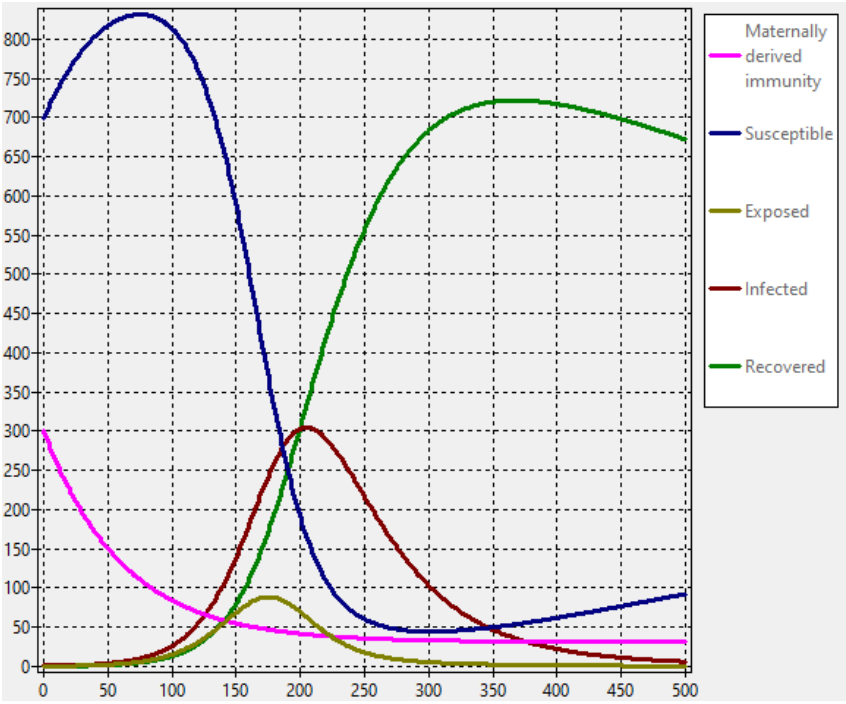
```
unit utlSEIS_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function SEISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function SEISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu, lambda, epsilon: Double;
    S, E, I: Double;
    dSdt, dEdt, dIdt: Double;
  begin
    {
      [ Description ]
      [*************************************************************]
      Calculate the SEIS's (Susceptible - Exposed - Infectious -
   Susceptible) model differential equations given a y state.

      [*************************************************************]

      [ Parameters ]
      [*************************************************************]
      Param y: Array of double containing the 'to be calculated' S E I
   S states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:
                        0    1     2    3      4       5
                       [N, beta, gamma, mu, lambda, epsilon]

      [*************************************************************]

      [ Variables ]
      [*************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                      (Infection rate)
      Var gamma: The recovery rate.
      Var mu: Natural mortality rate.
      Var lambda: Recruitment of the susceptible individuals (birth etc
   .)
      Var epsilon: The rate at which exposed individuals become
   infectious.

      Var S: Number of Susceptible individuals.
      Var E: Number of Exposed individuals.
      Var I: Number of Infected individuals.
```

```
53
54      Var dSdt: Calculated differential equation result for
    Susceptibles.
55      Var dEdt: Calculated differential equation result for Exposed.
56      Var dIdt: Calculated differential equation result for Infected.
57
58      [************************************************************]
59
60    }
61
62    S := y[0];
63    E := y[1];
64    I := y[2];
65
66    N := extraArgs[0];
67    beta := extraArgs[1];
68    gamma := extraArgs[2];
69    mu := extraArgs[3];
70    lambda := extraArgs[4];
71    epsilon := extraArgs[5];
72
73    dSdt := lambda - (beta * S * I / N) - (mu * S) + (gamma * I);
74    dEdt := (beta * S * I / N) - ((epsilon + mu) * E);
75    dIdt := (epsilon * E) - ((gamma + mu) * I);
76
77    Result := ArrayOfDouble.Create(dSdt, dEdt, dIdt);
78  end;
79
80 end.
```

4.10 MSEIR

Για την περίπτωση μιας ασθένειας με παράγοντες την παθητική ανοσία (passive immunity) αλλά και την περίοδο επώασης (incubation period) υπάρχει το μοντέλο MSEIR. Αυτό το μοντέλο είναι ιδανικό για ασθένειες που προσφέρουν στο τέλος διαρκής ανοσία όπως για παράδειγμα ο ιλαράς, ανεμοβλογιά, ερυθρά ή παρωτίτιδα [31]. Το μοντέλο MSEIR μπορεί να περιγραφεί από τις παρακάτω διαφορικές εξισώσεις:

$$\begin{aligned}
\frac{dM}{dt} &= \Lambda - (\delta + \mu) \cdot M \\
\frac{dS}{dt} &= (\delta \cdot M) - \frac{\beta \cdot I \cdot S}{N} - (\mu \cdot S) \\
\frac{dE}{dt} &= \frac{\beta \cdot I \cdot S}{N} - (\epsilon + \mu) \cdot E \\
\frac{dI}{dt} &= (\epsilon \cdot E) - (\gamma + \mu) \cdot I \\
\frac{dR}{dt} &= (\gamma \cdot I) - (\mu \cdot R)
\end{aligned} \tag{4.10}$$



Εικόνα 4.10: BVS | N: 1000, β: 0.1, γ: 0.02, μ: 0.001, Λ: 0.5, δ: 0.015, ϵ: 0.1, Maternally derived immunity: 300, Initial Infected: 1, Days: 500

Όπου:

- N είναι ο συνολικός πληθυσμός.
- β είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- γ είναι ο ρυθμός ανάρρωσης.
- μ είναι ο ρυθμός φυσικής θνησιμότητας.
- Λ είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- δ είναι ο ρυθμός θνησιμότητας της ασθένειας.
- ϵ είναι ο ρυθμός όπου τα εκτεθειμένα άτομα μετατρέπονται σε μολυσμένα άτομα.

4.10.1 Αλγόριθμος MSEIR

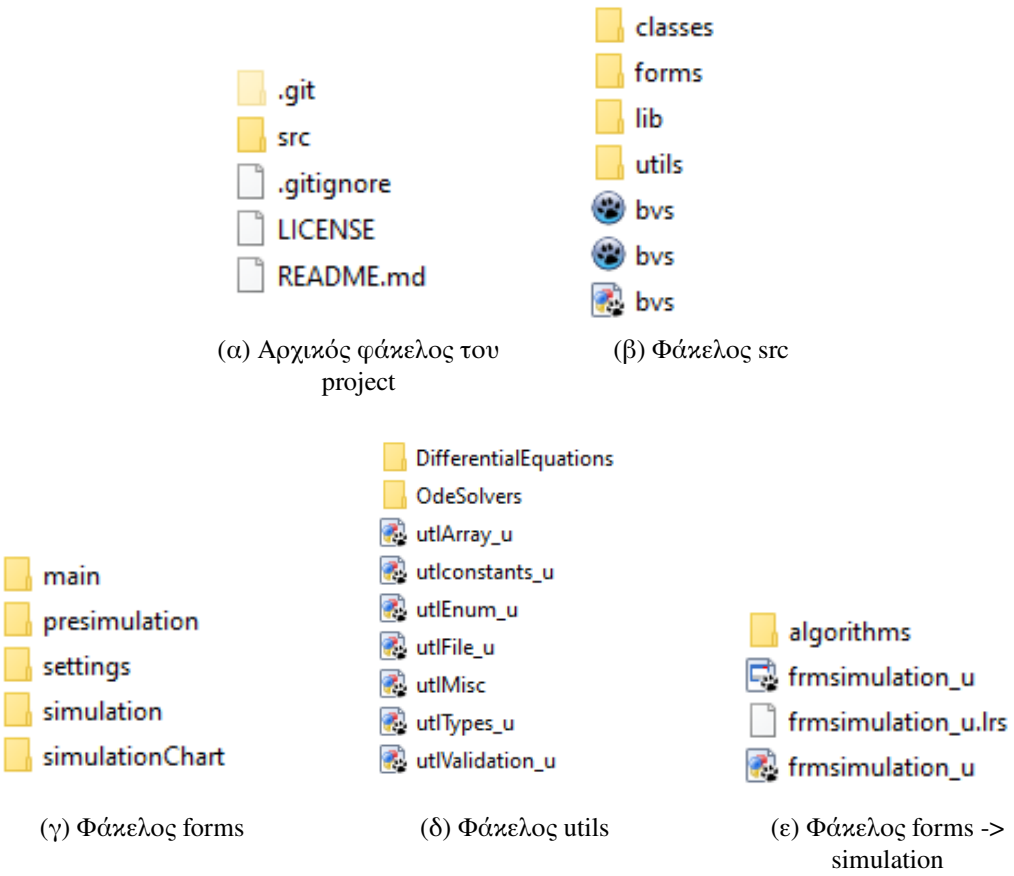
```
unit utlMSEIR_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Forms }
  { Classes }
  { Utilities }
  utlTypes_u;

function MSEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;

implementation
  function MSEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
  var
    N, beta, gamma, mu, lambda, delta, epsilon: Double;
    M, S, E, I, R: Double;
    dMdt, dSdt, dEdt, dIdt, dRdt: Double;
  begin
    {
      [ Description ]
      [************************************************************]
      Calculate the MSEIR's (Maternally derived immunity - Susceptible
   - Exposed - Infectious - Recovered) model differential equations
   given a y state.

      [************************************************************]

      [ Parameters ]
      [************************************************************]
      Param y: Array of double containing the 'to be calculated' M S E
   I R states.
      Param extraArgs: An array of double containing the extra
   arguments that are needed in order to perform calculations. The
   correct form of the array should be:

                        0    1     2    3      4      5       6
                       [N, beta, gamma, mu, lambda, delta, epsilon]

```

```
      [*****************************************************************]

      [ Variables ]
      [*****************************************************************]
      Var N: The total population.

      Var beta: The average number of contacts per person per time.
                    (Infection rate)
      Var gamma: The recovery rate.
      Var mu: Natural mortality rate.
      Var lambda: Recruitment of the susceptible individuals (birth etc
   .)
      Var delta: Disease mortality rate.
      Var epsilon: The rate at which exposed individuals become
   infectious.

      Var M: Number of individuals with Maternally derived immunity.
      Var S: Number of Susceptible individuals.
      Var E: Number of Exposed individuals.
      Var I: Number of Infected individuals.
      Var R: Number of Recovered individuals.

      Var dMdt: Calculated differential equation result for Maternally
                derived immunity.
      Var dSdt: Calculated differential equation result for
   Susceptibles.
      Var dEdt: Calculated differential equation result for Exposed.
      Var dIdt: Calculated differential equation result for Infected.
      Var dRdt: Calculated differential equation result for Recovered.

      [*****************************************************************]

    }

    M := y[0];
    S := y[1];
    E := y[2];
    I := y[3];
    R := y[4];

    N := extraArgs[0];
    beta := extraArgs[1];
    gamma := extraArgs[2];
    mu := extraArgs[3];
    lambda := extraArgs[4];
    delta := extraArgs[5];
    epsilon := extraArgs[6];

    dMdt := lambda - (delta * M) - (mu * M);
    dSdt := (delta * M) - (beta * S * I / N) - (mu * S);
    dEdt := (beta * S * I / N) - ((epsilon + mu) * E);
    dIdt := (epsilon * E) - ((gamma + mu) * I);
    dRdt := (gamma * I) - (mu * R);

    Result := ArrayOfDouble.Create(dMdt, dSdt, dEdt, dIdt, dRdt);
  end;

end.
```

Τα επιδημιολογικά μοντέλα έχουν εφαρμοστεί με τεράστια επιτυχία στην μελέτη αρκετών ασθενειών όπως HIV/AIDS [32] [33], Έμπολα [34] [35], Γρίπη [36], Καρκίνος [37], Δάγκειο [38] [39], Ελονοσία [40], μόλυνση από Σαλμονέλα [41], Zika [42] και COVID-19 [43]. Αξίζει να σημειωθεί ωστόσο ότι όλες οι διαφορικές εξισώσεις των επιδημιολογικών μοντέλων που αναφέρονται παραπάνω, προορίζονται να εφαρμοστούν σε ένα τέλεια ομογενή περιβάλλον. Αυτό σημαίνει ότι κάθε άτομο που είναι μέρος μιας προσομοίωσης που βασίζεται σε αυτά τα μοντέλα πρέπει να έχει έναν σύνδεσμο επαφής με όλα τα άλλα άτομα. Αυτό το γεγονός φυσικά δεν αντικατοπτρίζει την πραγματικότητα και σίγουρα δεν ταιριάζει με τη συμπεριφορά των ανθρώπων.

Αρχιτεκτονική λογισμικού

Το BVS είναι μια desktop εφαρμογή αφιερωμένη στη μελέτη μολυσματικών ασθενειών. Ο πυρήνας της εφαρμογής έχει αναπτυχθεί μέχρι στιγμής με Object Pascal, η οποία είναι μια γλώσσα προγραμματισμού πολύ χαμηλού επιπέδου. Ως αποτέλεσμα, το BVS είναι ένα εργαλείο με τεράστιες επιδόσεις ακόμη και σε καθημερινούς υπολογιστές. Το γραφικό του περιβάλλον έχει αναπτυχθεί μέχρι στιγμής χρησιμοποιώντας το λογισμικό Lazarus. Το Lazarus είναι ένα δωρεάν λογισμικό και ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για ταχεία ανάπτυξη εφαρμογών (RAD) χρησιμοποιώντας τον μεταγλωττιστή Free Pascal. Όσον αφορά τον κώδικα του εργαλείου αυτού, έχει γραφτεί με τρόπο ώστε να είναι ευανάγνωστος αλλά και εύκολα επεκτάσιμος ακολουθώντας κοινές αλλά καλές πρακτικές όπως αυτή της διάσπασης μεγάλων αρχείων και συναρτήσεων σε μικρότερα/ες. Η δομή των φακέλων χτίστηκε με τρόπο ο οποίος εξίσου βοηθάει και ενισχύει στην απλοϊκότητα της δομής του έργου.



(α) Αρχικός φάκελος του project

(β) Φάκελος src

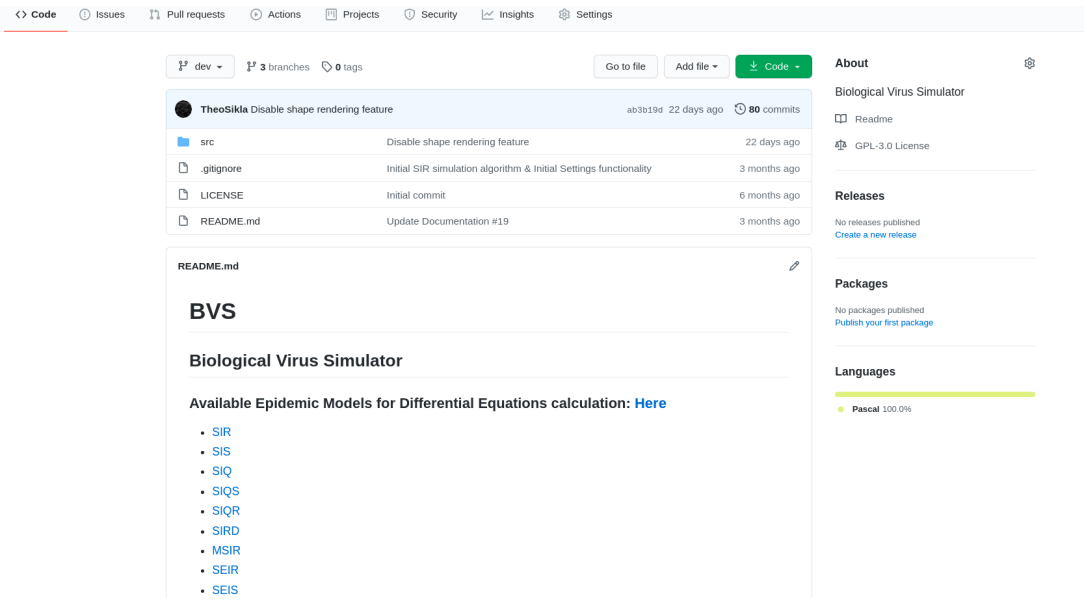
(γ) Φάκελος forms

(δ) Φάκελος utils

(ε) Φάκελος forms -> simulation

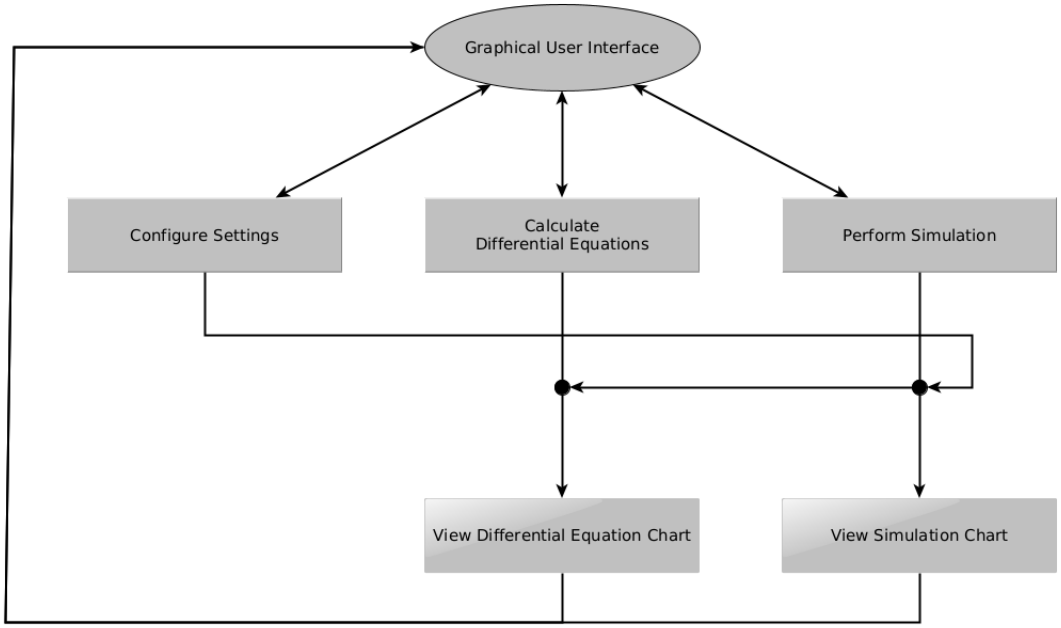
Εικόνα 5.1: BVS | Δομή φακέλων και αρχείων του έργου

Η ανάπτυξη και η διαχείριση του κώδικα έγινε με την χρήση ενός εργαλείου ελέγχου εκδόσεων (Version Control System - VCS), το οποίο ονομάζεται git. Το git δημιουργήθηκε το 2005 από τον Linus Torvalds, με σκοπό την διευκόλυνση της ανάπτυξης του πυρήνα του Linux. Από κει και έπειτα, το git έγινε ένα από τα σημαντικότερα εργαλεία ανάπτυξης λογισμικού καθώς αυτό παρακολουθεί αλλαγές αρχείων, επιτρέπει commit rollbacks ή ακόμη και παράλληλα branches ή αλλιώς εκδόσεις του κώδικα ενός λογισμικού, διατηρώντας ένα αρκετά οργανωμένο περιβάλλον όπως και ιστορικό ακόμη και σε πολύ μεγάλα έργα όπου ο κώδικας τους μπορεί να φτάσει ακόμη και μερικά εκατομμύρια γραμμές. Ο κώδικας του εργαλείου BVS είναι διαθέσιμος στην ιστοσελίδα του GitHub, η οποία παρέχει σε έναν χρήστη ένα φιλικό περιβάλλον για την διαχείριση των έργων που αναπτύσονται υπό την επίβλεψη του εργαλείου git.



Εικόνα 5.2: Διεπαφή χρήστη της ιστοσελίδας του GitHub

Ο σχεδιασμός της γραφικής διεπαφής χρήστη του BVS προσφέρει ένα πολύ ολοκληρωμένο και κατανοητό περιβάλλον για όλους τους χρήστες του. Πιο συγκεκριμένα η εφαρμογή μπορεί να χρησιμοποιηθεί από επιστήμονες, για να έχουν μια αρχική ανάλυση μιας μολυσματικής ασθένειας και πώς μπορεί να είναι η πορεία της. Το εργαλείο μπορεί επίσης να χρησιμοποιηθεί από μαθητές και τους καθημερινούς ανθρώπους με σκοπό την κατανόηση του θεμελιωδών εννοιών της επιδημιολογίας και των διαμερισματικών μοντέλων. Όσον αφορά τα διαμερισματικά μοντέλα, το BVS υποστηρίζει επί του παρόντος όλα τα μοντέλα που αναφέρθηκαν στην ενότητα των διαμερισματικών μοντέλων και προσφέρει μια επεξηγηματική γραφική παράσταση για το καθένα ξεχωριστά.



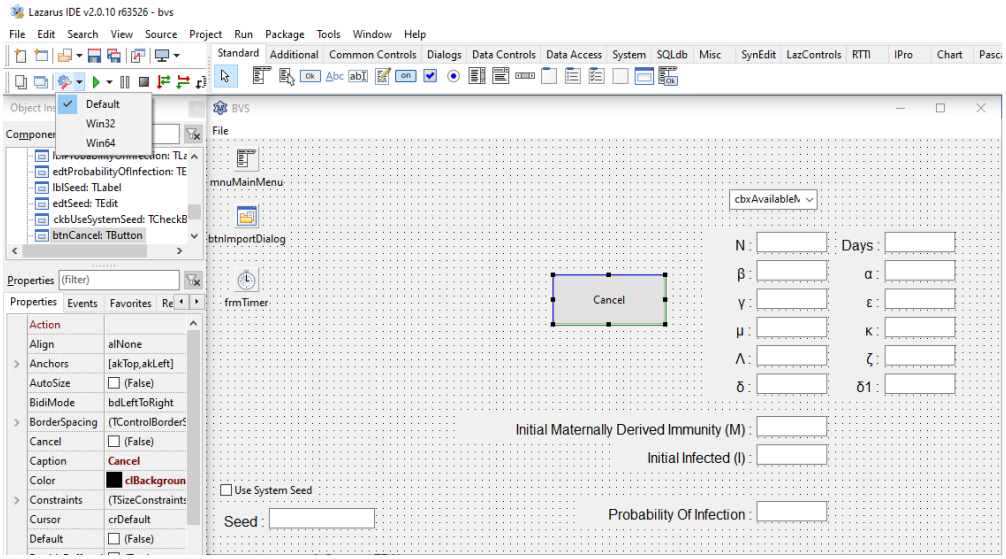
Εικόνα 5.3: Αρχιτεκτονική του εργαλείου BVS

Ωστόσο, όπως αναφέρθηκε προηγουμένως, οι διαφορικές εξισώσεις των διαμερισματικών μοντέλων, μπορούν να εφαρμοστούν μόνο σε ομοιογενές περιβάλλοντα. Επομένως, το BVS υποστηρίζει επίσης προσομοιώσεις που χρησιμοποιούν περιβάλλοντα, που είναι περισσότερο ρεαλιστικά και πιο κοντά στην πραγματική συμπεριφορά των ανθρώπων. Αυτά τα περιβάλλοντα μπορούν να αναπαρασταθούν ως γραφήματα όπου ένας κόμβος αντιπροσωπεύει ένα μεμονωμένο άτομο και μια ακμή μεταξύ ατόμων αντικατοπτρίζει μια επαφή που μπορεί να συμβεί μεταξύ τους. Τα πιο κοινά μοντέλα γραφημάτων που χρησιμοποιούνται με σκοπό να αντικατοπτρίσουν ένα δίκτυο επαφών πραγματικών ατόμων είναι τα τυχαία γραφήματα και τα γραφήματα ελεύθερης κλίμακας [44].

Αρκετά βολικά, ένα εργαλείο με το όνομα GRATIS μπορεί να χρησιμοποιηθεί για τη δημιουργία τέτοιων γραφημάτων. Το GRATIS είναι επίσης μια desktop εφαρμογή και μπορεί να χρησιμοποιηθεί για ανάλυση, οπτικοποίηση ή την δημιουργία ενός γραφήματος. Η ενότητα δημιουργίας του GRATIS επιτρέπει σε έναν χρήστη να δημιουργήσει ένα πολύ προσαρμοσμένο γράφημα, προσφέροντας έτσι την ικανότητα δημιουργίας τοπολογιών οι οποίες είναι πιο ¨ζωντανές¨ και πιο κοντά σε αυτές των ανθρώπων. Το GRATIS υποστηρίζει δύο μορφές γραφήματος ως έξοδο και αυτές είναι ως πίνακας γειτνίασης ή ως λίστα γειτνίασης [44]. Προχωρώντας περαιτέρω, με τον σκοπό της προσομοίωσης σε μια μη ομοιογενή τοπολογία, ένα αρχείο που περιέχει ένα γράφημα στην κατάλληλη μορφή πρέπει να εισαχθεί στο εργαλείο BVS. Επί του παρόντος, το BVS υποστηρίζει αρχεία σε μορφή πίνακα γειτνίασης αλλά και λίστα γειτνίασης.

Υλοποίηση

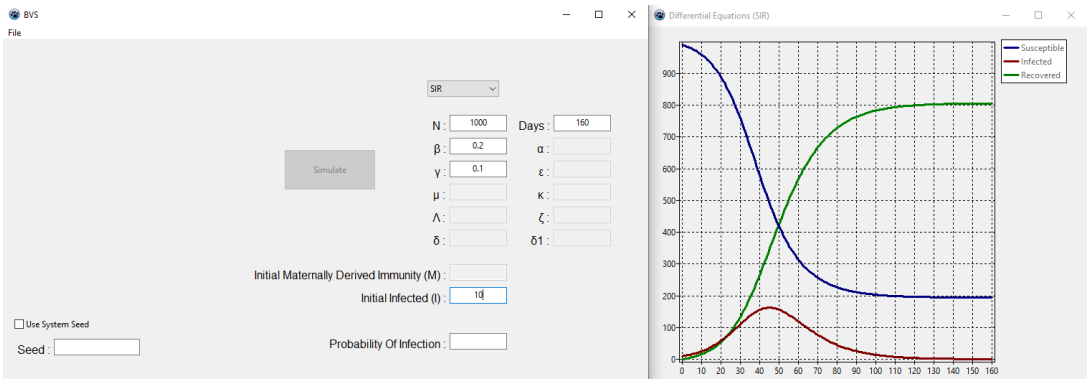
Όπως προαναφέρθηκε και στην ενότητα της αρχιτεκτονικής (Ενότητα 5), ως περιβάλλον ανάπτυξης της εφαρμογής BVS χρησιμοποιήθηκε το Lazarus IDE. Επειδή λοιπόν το Lazarus είναι φτιαγμένο για ταχεία ανάπτυξη εφαρμογών, παρέχει στους εκάστοτε developers αρκετά εργαλεία και λειτουργίες, καθιστώντας την ανάπτυξη desktop εφαρμογών μία ευχάριστη και αρκετά απλοϊκή διαδικασία. Παραδείγματος χάριν, το γραφικό περιβάλλον του Lazarus υποστηρίζει τον σχεδιασμό των σελίδων ή αλλιώς φορμών μιας εφαρμογής με την μέθοδο drag and drop, γεγονός το οποίο μειώνει τον χρόνο ανάπτυξης και δημιουργίας τους, αλλά και τυχόν λάθη σε κώδικα που έχει γραφτεί από έναν developer. Επιπλέον, το Lazarus δίνει την δυνατότητα δημιουργίας εκτελέσιμων αρχείων μιας εφαρμογής για παραπάνω από ένα λειτουργικά συστήματα (cross compilation) μονάχα με την προσαρμογή λίγων ρυθμίσεων και το πάτημα ενός κουμπιού. Αυτό το γεγονός είναι ύψιστης σημασίας καθώς πλέον οι χρήστες ηλεκτρονικών υπολογιστών είναι περισσότερο εξοικειωμένοι και συνεπακόλουθα χρησιμοποιούν ποικιλίες λειτουργικών συστημάτων όπως για παράδειγμα (Windows, Linux, MacOs) και λοιπά, πράγμα το οποίο εξαλείφει την ύπαρξή κάποιου μονοπωλίου που αφορά τα λειτουργικά συστήματα. Περαιτέρω, αξίζει να αναφερθεί και ένα σχετικά σημαντικό μειονέκτημα της βιβλιοθήκης συστατικών ή αλλιώς widgets του Lazarus γνωστή και ως Lazarus component library (LCL), το οποίο είναι ότι η βιβλιοθήκη είναι δομημένη με τρόπο ώστε να μην υποστηρίζεται η λεπτομερής επεξεργασία χρώματος και styling των αντικειμένων (widgets) της κάθε φόρμας. Αντ' αυτού η βιβλιοθήκη παρέχει ορισμένα themes τα οποία προσφέρουν αυστηρά συγκεκριμένο look and feel στις φόρμες και στα αντικείμενα (widgets) τους, τα οποία χρησιμοποιούνται με βάση το λογισμικό που χρησιμοποιείται για το τρέξιμο της εφαρμογής.



Εικόνα 6.1: Περιβάλλον ανάπτυξης εφαρμογών Lazarus

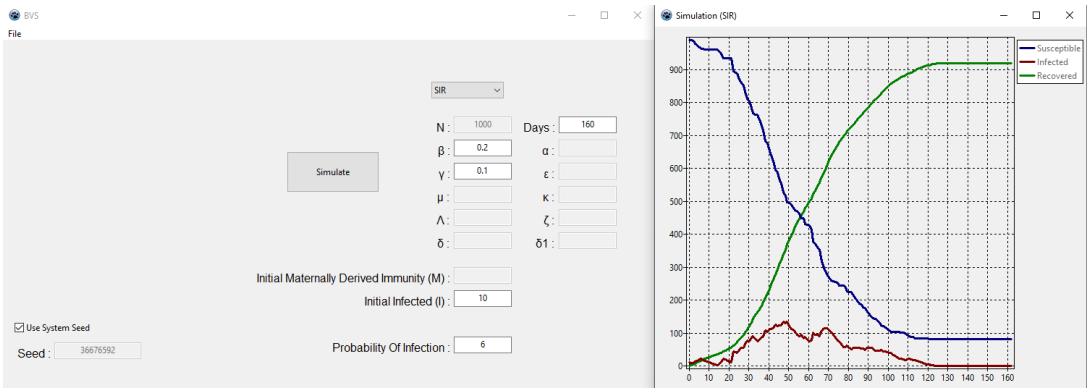
Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής BVS σε συνδυασμό με το Lazarus IDE, δεν ήταν άλλη από την Object Pascal. Η Object Pascal είναι μια αντικειμενοστραφή επέκταση της Pascal που αναπτύχθηκε από την Apple για πρώτη φορά το 1986. Κάποια από τα πιο σημαντικά πλεονεκτήματα της Object Pascal είναι ότι έχει πολύ εύκολο συντακτικό, καθιστώντας την μια πολύ καθαρή από άποψη ανάγνωσης κώδικα γλώσσα προγραμματισμού, χρησιμοποιεί πολύ λίγη μνήμη και είναι υπερβολικά γρήγορη σε σημείο όπου ανταγωνίζεται τις γλώσσες προγραμματισμού C και C++, οι οποίες θεωρούνται οι γρηγορότερες γλώσσες προγραμματισμού. Τέλος, έχει υψηλή διαθεσιμότητα, ειδικά σε συνδυασμό με τον μεταγλωττιστή Free pascal (FPC), που έχει ως αποτέλεσμα την δυνατότητα χρήσης σε πολλές πλατφόρμες και αρχιτεκτονικές συστημάτων.

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας παρέχει για την ώρα δύο αρκετά σημαντικές λειτουργίες στον χρήστη. Η πρώτη λειτουργία είναι ο υπολογισμός των διαφορικών εξισώσεων που υπάρχουν στο σύστημα σε συνδυασμό με την παραγωγή μιας αναλυτικής γραφικής παράστασης η οποία περιγράφει την πορεία μιας μολυσματικής ασθένειας για την χρονική περίοδο που ορίστηκε από τον χρήστη. Για να αξιοποιήσει λοιπόν αυτήν την λειτουργία, ένας χρήστης δεν έχει παρά μόνο να επιλέξει το μοντέλο διαφορικών εξισώσεων που επιθυμεί, να συμπληρώσει τα κατάλληλα πεδία που αντιστοιχούν στο κάθε μοντέλο και να πατήσει το πλήκτρο ENTER ενώ βρίσκεται μέσα σε ένα από αυτά τα πεδία (Σημείωση: Τα πεδία Use System Seed, Seed και Probability Of Infection δεν αφορούν τον υπολογισμό των διαφορικών εξισώσεων).



Εικόνα 6.2: Υπολογισμός διαφορικής εξίσωσης

Η δεύτερη λειτουργία αφορά την προσομοίωση της πορείας μιας μολυσματικής ασθένειας με βάση μια συγκεκριμένη τοπολογία (Γράφο). Για την αξιοποίηση αυτής της λειτουργίας, ένας χρήστης πρέπει να εισάγει στο σύστημα ένα αρχείο που περιέχει την κατάλληλη τοπολογία στην σωστή μορφή [44], πατώντας το κουμπί του μενού ¨file¨, και επιλέγοντας την λειτουργία ¨Open¨, έτσι ώστε να ανοίξει ο κατάλληλος κατάλογος και να επιλέξει το αρχείο της τοπολογίας. Έπειτα, αφού το αρχείο εισαχθεί στο σύστημα με επιτυχία θα πρέπει για ακόμη μία φορά να επιλεχθεί το μοντέλο του οποίου ο προσαρμοσμένος αλγόριθμος θα εφαρμοστεί στην εκάστοτε τοπολογία και να συμπληρωθούν τα πεδία που αντιστοιχούν στο μοντέλο που επιλέχθηκε καθώς επίσης και τα πεδία πεδία Use System Seed ή Seed και Probability Of Infection. Τέλος θα πρέπει να πατήσει το κουμπί Simulate και να περιμένει να ολοκληρωθεί η προσομοίωση με σκοπό να παραχθεί μια νέα γραφική παράσταση που αφορά την εξέλιξη της μετάδοσης μιας μολυσματικής ασθένειας στην τοπολογία που εισήχθηκε προηγουμένως στο σύστημα.



Εικόνα 6.3: Προσομοίωση μολυσματικής ασθένειας

Προσαρμοσμένοι αλγόριθμοι

Προς το παρόν υπάρχουν δύο προσαρμοσμένοι αλγόριθμοι επιδημίας στο BVS. Ένας για την προσομοίωση μιας μολυσματικής ασθένειας σε μη ομοιογενή περιβάλλοντα, ο οποίος βασίζεται σε διάφορα χαρακτηριστικά του επιδημιολογικού μοντέλου SIR και τις διαφορικές εξισώσεις του και ένας για το μοντέλο SIS αντίστοιχα. Αυτοί οι αλγόριθμοι κληρονομούν τα χαρακτηριστικά τους από μεταβλητές που υπάρχουν στις διαφορικές εξισώσεις όπως για παράδειγμα το β του μοντέλου SIR, το οποίο αντιπροσωπεύει τον μέσο αριθμό επαφών κάθε ατόμου. Συνεπώς, συνδυάζοντας αυτά τα χαρακτηριστικά και κάποια λογική, καταλήγουμε στους ακόλουθους αλγόριθμους:

7.1 SIR

Ο προσαρμοσμένος αλγόριθμος επιδημίας SIR που υπάρχει επί του παρόντος στο εργαλείο BVS, αποτελείται από τα ακόλουθα βήματα:

- Αρχικοποιήση των λιστών ευάλωτων (Susceptible), μολυσμένων (Infectious) και αυτών που έχουν αναρρώσει (Recovered).
- Μετατροπή όλων των κόμβων (ατόμων) σε ευάλωτους και τοποθέτηση αυτών στο group των ευάλωτων (Susceptible).
- Αρχική μόλυνση ενός η περισσοτέρων κόμβων.
- Κάθε μέρα της προσομοίωσης και για κάθε κόμβο επιλέγεται ένας τυχαίος αριθμός γειτόνων των κόμβων αυτών και μολύνονται με πιθανότητα pos. Όπου γείτονες είναι ίσο με: αριθμός γειτόνων του ενός κόμβου * β.
- Εάν ικανοποιηθεί ο μέγιστος αριθμός κόμβων που μπορούν να μολυνθούν σε μια μέρα, γίνεται διακοπή ελέγχου των υπολοίπων κόμβων και μετάβαση στην επόμενη μέρα.
- Στο τέλος της κάθε ημέρας προσομοίωσης αναρρώνουν από την ασθένεια $\gamma * I$ μολυσμένοι κόμβοι και μεταφέρονται στο group αυτών που ανάρρωσαν από την ασθένεια (Recovered).

7.1.1 Πηγαίος κώδικας SIR

```
unit AlgSIR_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Classes }
  clNode_u,
  utlTypes_u,
  utlMisc;

procedure SIRALG(days, NumberOfInitialInfectedNodes: Word;
                 beta, gamma: Double;
                 ProbabilityOfInfection: Byte;
                 var SamplingResult: TArrayOfArrayOfWord);

implementation
uses
  { Forms }
  frmMain_u,
  frmSimulation_u,
  frmSettings_u;

procedure SIRALG(days, NumberOfInitialInfectedNodes: Word;
                 beta, gamma: Double;
                 ProbabilityOfInfection: Byte;
                 var SamplingResult: TArrayOfArrayOfWord);
var
  i, j, k, day: Word;
  pos: Integer;
  exitedWhileSimulating: Boolean;

  InitialInfectedNodes, Neighbors, Susceptible, Infected, Recovered:
   TWordList;

  TestingNode, NodeToBeRecovered, NodesInfectedByNodePerDay,
  NumberOfNodesInfectedPerDay, NumOfMaxNeighborsToTest,
   NumOfMaxNodesPerDay: Word;

begin
  {
            <>=========================================================<>
            ||                                                         ||
            ||   Simulation algorithm for the SIR epidemic model.     ||
            ||                                                         ||
            <>=========================================================<>

    [ Algorithm ]

   [*************************************************************]

      The algorithm is constructed with the following steps:
      1) Initialize the Susceptible, Infected and Recovered lists.

      2) Make all nodes Susceptible add them to the Susceptible list.

```

```
55      3) Initial infect of one or more random node/s.
56
57      4) For each day of the simulation, for each node pick a random
    number of neighbors and infect them with probability pos. Where
    neighbors is equal to: number of neighbors of each node * beta.
58
59      5) If the maximum number of nodes that can be infected in one day
     is reached, go to the next day.
60
61      6) At the end of each day make sure that gamma*I nodes recover
    from the decease.
62
63
    [*****************************************************************]
64
65     [ Variables ]
66
    [*****************************************************************]
67          i, j, k, day: Counters.
68
69          days: Number of days that the simulation lasts.
70          firstInfected: Index of the first infected node.
71          TestingNode: Index of node to be tested.
72          NodeToBeRecovered: Index of node to be recovered.
73
74          ProbabilityOfInfection: The probability of infection specified
    by the user.
75          pos: Probability of Infection.
76
77          Neighbors: List with the contact indexes of each node.
78          Susceptible: List that holds the indexes of the susceptible
    nodes.
79          Infected: List that holds the indexes of the infected nodes.
80          Recovered: List that holds the indexes of the recovered nodes.
81
82          SamplingResult: List that holds the samplings taken from the
    Susceptible, Infected, Recovered lists, at the end of each day.
83
84          beta: The average number of contacts per person per time. (
    Infection rate)
85          gamma: The recovery rate.
86
87          NodesInfectedByNodePerDay: Number of nodes that were infected
    by a node in one day.
88          NumberOfNodesInfectedPerDay: Number of nodes that were infected
     in one day.
89
90          NumOfMaxNeighborsToTest: Number of maximum neighbors to be
    tested for each node.
91          NumOfMaxNodesPerDay: Number of maximum nodes that is allowed to
     be infected per day.
92
93
    [*****************************************************************]
94  }
95
96  frmSimulation.frmSmlInvoker.Enabled := False;
```

```
97  exitedWhileSimulating := False;
98
99  { Initialize Variable Lists }
100 InitialInfectedNodes := TWordList.Create;
101 Neighbors := TWordList.Create;
102 Susceptible := TWordList.Create;
103 Infected := TWordList.Create;
104 Recovered := TWordList.Create;
105
106 { Append all nodes to initial Susceptible state }
107 for i := 0 to frmMain.Nodes.Count - 1 do begin
108   frmMain.Nodes[i].MakeSusceptible;
109   Susceptible.Add(i);
110 end;
111
112 InitialInfectedNodes := frmSimulation.InfectRandomNodes(
     NumberOfInitialInfectedNodes);
113
114 for i := 0 to NumberOfInitialInfectedNodes - 1 do begin
115   Susceptible.Remove(InitialInfectedNodes[i]);
116   Infected.Add(InitialInfectedNodes[i]);
117 end;
118
119 { Take an initial sampling }
120 SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
     Count, Recovered.Count));
121
122 for Day := 0 to Days do begin
123
124   if not (Susceptible.Count = 0) and not (Infected.Count = 0) then
     begin
125     NumberOfNodesInfectedPerDay := 0;
126
127     //writeln('Day: ' + IntToStr(Day)); { Debug }
128
129     for i := 0 to frmMain.Nodes.Count - 1 do begin
130       NodesInfectedByNodePerDay := 0;
131       Neighbors := frmMain.Nodes[i].Neighbors;
132
133       if Neighbors.Count > 0 then begin
134         NumOfMaxNodesPerDay := Random(Round(Neighbors.Count * beta));
135         NumOfMaxNeighborsToTest := NumOfMaxNodesPerDay;
136         if NumOfMaxNeighborsToTest = 0 then NumOfMaxNeighborsToTest
     := 1;
137
138         //writeln('Number of neighbors to be tested by node ' +
     IntToStr(i) + ' at day ' + IntToStr(day) + ' are ' + IntToStr(
     NumOfMaxNeighborsToTest)); { Debug }
139
140         for j := 0 to NumOfMaxNeighborsToTest do begin
141           pos := Random(100);
142           if pos <= ProbabilityOfInfection then begin
143             { Pick a random Neighbor }
144             TestingNode := Neighbors[Random(Neighbors.Count)];
145
146             if not frmMain.Nodes[TestingNode].IsInfected and
147                not frmMain.Nodes[TestingNode].IsRecovered and
148                frmMain.Nodes[i].IsInfected then begin
149
150               frmSimulation.InfectNode(frmMain.Nodes[TestingNode],
```

```
  frmMain.Nodes[i].Id);

                Susceptible.Remove(TestingNode);
                Infected.Add(TestingNode);

                Inc(NodesInfectedByNodePerDay);
                //writeln('Node ' + IntToStr(frmMain.Nodes[TestingNode
].Id) + ' has been infected by node ' + IntToStr(frmMain.Nodes[i].Id
)); { Debug }
              end;
            end;
          end; { End j }

          //{ In case the simulation was forced closed before finishing
 set the
          //  exitedWhileSimulating flag to true. }
          //if not frmSimulation.Showing then begin
          //  exitedWhileSimulating := True;
          //  break;
          //end;

          { If there are no infected nodes stop the simulation. }
          if Infected.Count = 0 then break;

          NumberOfNodesInfectedPerDay += NodesInfectedByNodePerDay;
          //writeln('Node: ' + IntToStr(i) + ' infected: ' + IntToStr(
NodesInfectedByNodePerDay) + ' nodes, at day: ' + IntToStr(day)); {
Debug }

          if NumberOfNodesInfectedPerDay >= NumOfMaxNodesPerDay then
break;

        end;

        end; { End i }

      end;

     //writeln('At Day: ' + IntToStr(day) + ', { ' + IntToStr(
NumberOfNodesInfectedPerDay) + ' } where infected.'); { Debug }

     { Recover FIFO (First In First Out) Style }
     for k := 0 to Round(Infected.Count * gamma) do begin
       if (Infected.Count > 0) then begin
          NodeToBeRecovered := Infected[0];
          frmMain.Nodes[NodeToBeRecovered].Recover;

          Infected.Remove(NodeToBeRecovered);
          Recovered.Add(NodeToBeRecovered);
       end
       else break;
     end;

     SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
Count, Recovered.Count));

     if frmMain.CancelTriggered then break;

  end; { End Day }

```

```
202   { Repeat the simulation if needed }
203   if (Recovered.Count < StrToInt(frmSettings.
      edtReSimulateMinRecoveredNodeCount.Text)) and
204     frmMain.ckbUseSystemSeed.Checked and
205     frmSettings.cbxReSimulate.Checked and
206     not frmMain.CancelTriggered then begin
207
208       SamplingResult.Clear;
209       frmSimulation.RestoreNodes; // Restore the Nodes
210       RandomizeSystem; // Randomize System
211       SIRALG(days, NumberOfInitialInfectedNodes, beta, gamma,
      ProbabilityOfInfection, SamplingResult); // Repeat the simulation
212   end
213   else if not frmMain.CancelTriggered then begin
214     { Take missing samplings }
215     for i := 0 to days - SamplingResult.Count do
216       SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected
      .Count, Recovered.Count));
217
218     { Take a final sampling }
219     SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
      Count, Recovered.Count));
220
221     //{ If the exitedWhileSimulating is set to true invoke the
      FormClose action. }
222     //if exitedWhileSimulating then self.FormClose(self);
223
224     { Print the samples } { Debug }
225     //for i:=0 to SamplingResult.Count - 1 do begin
226     //  write(IntToStr(i) + ': ');
227     //  for j := 0 to 3 - 1 do begin
228     //    write(SamplingResult[i][j]:2);
229     //    write(', ');
230     //  end;
231     //  writeln();
232     //end;
233
234     SamplingResult.DeleteRange(days + 2, SamplingResult.Count - 1);
235
236   end;
237 end;
238
239 end.
```

7.2 SIS

Ο προσαρμοσμένος αλγόριθμος επιδημίας SIS που υπάρχει επί του παρόντος στο εργαλείο BVS, αποτελείται από τα ακόλουθα βήματα:

- Αρχικοποιήση των λιστών ευάλωτων (Susceptible) και μολυσμένων (Infectious).
- Μετατροπή όλων των κόμβων (ατόμων) σε ευάλωτους και τοποθέτηση αυτών στο group των ευάλωτων (Susceptible).
- Αρχική μόλυνση ενός η περισσοτέρων κόμβων.
- Κάθε μέρα της προσομοίωσης και για κάθε κόμβο επιλέγεται ένας τυχαίος αριθμός γειτόνων των κόμβων αυτών και μολύνονται με πιθανότητα pos. Όπου γείτονες είναι ίσο με: αριθμός γειτόνων του ενός κόμβου * β.
- Εάν ικανοποιηθεί ο μέγιστος αριθμός κόμβων που μπορούν να μολυνθούν σε μια μέρα, γίνεται διακοπή ελέγχου των υπολοίπων κόμβων και μετάβαση στην επόμενη μέρα.
- Στο τέλος της κάθε ημέρας προσομοίωσης αναρρώνουν από την ασθένεια $\gamma * I$ μολυσμένοι κόμβοι και μεταφέρονται ξανά στο group αυτών που είναι ευάλωτοι στην ασθένεια (Susceptible).

7.2.1 Πηγαίος κώδικας SIS

```
unit AlgSIS_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils,
  { Classes }
  clNode_u,
  utlTypes_u,
  utlMisc;

procedure SISALG(days, NumberOfInitialInfectedNodes: Word;
                 beta, gamma: Double;
                 ProbabilityOfInfection: Byte;
                 var SamplingResult: TArrayOfArrayOfWord);

implementation
uses
  { Forms }
  frmMain_u,
  frmSimulation_u,
  frmSettings_u;

procedure SISALG(days, NumberOfInitialInfectedNodes: Word;
                 beta, gamma: Double;
                 ProbabilityOfInfection: Byte;
                 var SamplingResult: TArrayOfArrayOfWord);
var
  i, j, k, day: Word;
```

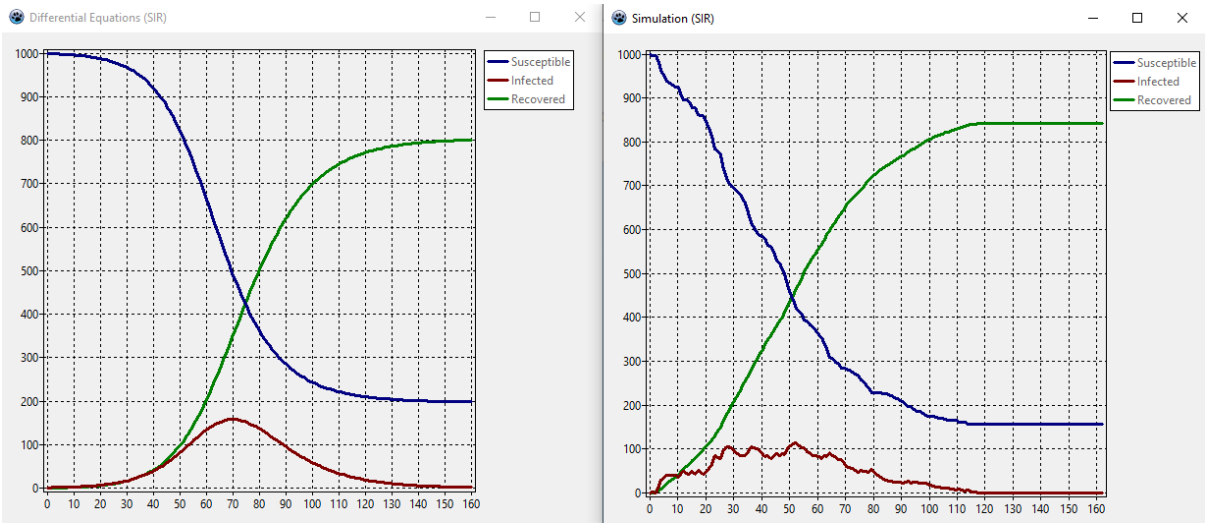
```
  pos: Integer;
  exitedWhileSimulating: Boolean;

  InitialInfectedNodes, Neighbors, Susceptible, Infected: TWordList;

  NodesInfectedByNodePerDay, NumberOfNodesInfectedPerDay,
    NumOfMaxNeighborsToTest, NumOfMaxNodesPerDay, TestingNode,
    NodesToBecomeSusceptible, NodeToBecomeSusceptible: Word;

begin
  {
            <>=========================================================<>
            ||                                                         ||
            ||   Simulation algorithm for the SIS epidemic model.   ||
            ||                                                         ||
            <>=========================================================<>

    [ Algorithm ]

   [*****************************************************************]

      The algorithm is constructed with the following steps:
      1) Initialize the Susceptible and Infected lists.

      2) Make all nodes Susceptible add them to the Susceptible list.

      3) Initial infect of one or more random node/s.

      4) For each day of the simulation, for each node pick a random
   number of neighbors and infect them with probability pos. Where
   neighbors is equal to: number of neighbors of each node * beta.

      5) If the maximum number of nodes that can be infected in one day
    is reached, go to the next day.

      6) At the end of each day make sure that gamma*I nodes recover
   from the decease and become susceptible once again.

   [*****************************************************************]

    [ Variables ]

   [*****************************************************************]

        i, j, k, day: Counters.

        days: Number of days that the simulation lasts.
        firstInfected: Index of the first infected node.
        TestingNode: Index of node to be tested.
        NodeToBecomeSusceptible: Index of node to be made susceptible
   again.

        ProbabilityOfInfection: The probability of infection specified
   by the user.
        pos: Probability of Infection.

        Neighbors: List with the contact indexes of each node.
        Susceptible: List that holds the indexes of the susceptible
   nodes.
        Infected: List that holds the indexes of the infected nodes.
```

```
79
80        SamplingResult: List that holds the samplings taken from the
    Susceptible, Infected lists, at the end of each day.
81
82        beta: The average number of contacts per person per time. (
    Infection rate)
83        gamma: The recovery rate.
84
85        NodesInfectedByNodePerDay: Number of nodes that were infected
    by a node in one day.
86        NumberOfNodesInfectedPerDay: Number of nodes that were infected
     in one day.
87
88        NumOfMaxNeighborsToTest: Number of maximum neighbors to be
    tested for each node.
89        NumOfMaxNodesPerDay: Number of maximum nodes that is allowed to
     be infected per day.
90
    [*****************************************************************]

91  }
92
93  frmSimulation.frmSmlInvoker.Enabled := False;
94  exitedWhileSimulating := False;
95
96  { Initialize Variable Lists }
97  Susceptible := TWordList.Create;
98  Infected := TWordList.Create;
99
100 { Append all nodes to initial Susceptible state }
101 for i := 0 to frmMain.Nodes.Count - 1 do begin
102   frmMain.Nodes[i].MakeSusceptible;
103   Susceptible.Add(i);
104 end;
105
106 InitialInfectedNodes := frmSimulation.InfectRandomNodes(
    NumberOfInitialInfectedNodes);
107
108 for i := 0 to NumberOfInitialInfectedNodes - 1 do begin
109   Susceptible.Remove(InitialInfectedNodes[i]);
110   Infected.Add(InitialInfectedNodes[i]);
111 end;
112
113 { Take an initial sampling }
114 SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
    Count));
115
116 for Day := 0 to Days do begin
117
118   if not (Susceptible.Count = 0) and not (Infected.Count = 0) then
    begin
119     NumberOfNodesInfectedPerDay := 0;
120
121     //writeln('Day: ' + IntToStr(Day)); { Debug }
122
123     for i := 0 to frmMain.Nodes.Count - 1 do begin
124       NodesInfectedByNodePerDay := 0;
125
126       Neighbors := frmMain.Nodes[i].Neighbors;
127
```

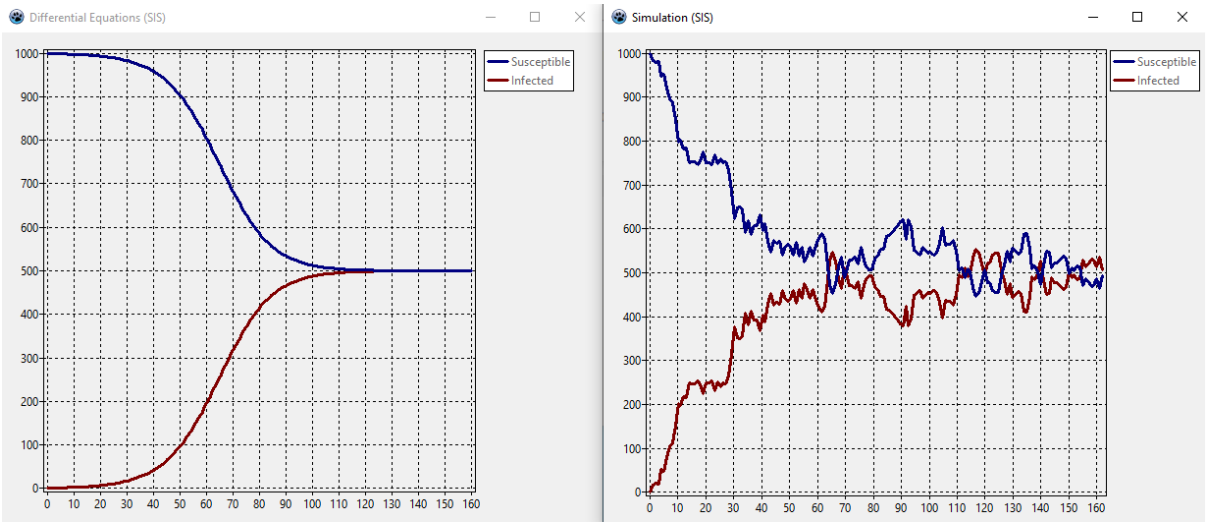
```
128         if Neighbors.Count > 0 then begin
129           NumOfMaxNodesPerDay := Random(Round(Neighbors.Count * beta));
130           NumOfMaxNeighborsToTest := NumOfMaxNodesPerDay;
131           if NumOfMaxNeighborsToTest = 0 then NumOfMaxNeighborsToTest
    := 1;
132
133           //writeln('Number of neighbors to be tested by node ' +
    IntToStr(i) + ' at day ' + IntToStr(day) + ' are ' + IntToStr(
    NumOfMaxNeighborsToTest)); { Debug }
134
135           for j := 0 to NumOfMaxNeighborsToTest do begin
136             pos := Random(100);
137             if pos <= ProbabilityOfInfection then begin
138               { Pick a random Neighbor }
139               TestingNode := Neighbors[Random(Neighbors.Count)];
140
141               if (not frmMain.Nodes[TestingNode].IsInfected and
142                  frmMain.Nodes[TestingNode].IsSusceptible) and
143                  frmMain.Nodes[i].IsInfected then begin
144
145                 frmSimulation.InfectNode(frmMain.Nodes[TestingNode],
    frmMain.Nodes[i].Id);
146
147                 Susceptible.Remove(TestingNode);
148                 Infected.Add(TestingNode);
149
150                 Inc(NodesInfectedByNodePerDay);
151                 //writeln('Node ' + IntToStr(frmMain.Nodes[TestingNode
    ].Id) + ' has been infected by node ' + IntToStr(frmMain.Nodes[i].Id
    )); { Debug }
152               end;
153             end;
154           end; { End j }
155
156           //{ In case the simulation was forced closed before finishing
     set the
157           //  exitedWhileSimulating flag to true. }
158           //if not frmSimulation.Showing then begin
159           //  exitedWhileSimulating := True;
160           //  break;
161           //end;
162
163           NumberOfNodesInfectedPerDay += NodesInfectedByNodePerDay;
164           //writeln('Node: ' + IntToStr(i) + ' infected: ' + IntToStr(
    NodesInfectedByNodePerDay) + ' nodes, at day: ' + IntToStr(day)); {
    Debug }
165
166           if NumberOfNodesInfectedPerDay >= NumOfMaxNodesPerDay then
    break;
167
168         end;
169
170       end; { End i }
171     end;
172
173     //writeln('At Day: ' + IntToStr(day) + ', { ' + IntToStr(
    NumberOfNodesInfectedPerDay) + ' } where infected.');
174
175     { Turn Nodes into Susceptible again }
176     NodesToBecomeSusceptible := Round(Infected.Count * gamma);
```

```
177     if (NodesToBecomeSusceptible > 1) then begin
178       for k := 0 to NodesToBecomeSusceptible do begin
179         if (Infected.Count > 0) then begin
180           NodeToBecomeSusceptible := Infected[0];
181           frmMain.Nodes[NodeToBecomeSusceptible].MakeSusceptible;
182
183           Infected.Remove(NodeToBecomeSusceptible);
184           Susceptible.Add(NodeToBecomeSusceptible);
185         end;
186       end;
187     end;
188
189     SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
    Count));
190
191     if frmMain.CancelTriggered then break;
192
193   end; { End Day }
194
195   { Repeat the simulation if needed }
196   if (Susceptible.Count >= frmMain.Nodes.Count - 1) and
197      frmMain.ckbUseSystemSeed.Checked and
198      frmSettings.cbxReSimulate.Checked and
199      not frmMain.CancelTriggered then begin
200
201       SamplingResult.Clear;
202       frmSimulation.RestoreNodes; // Restore the Nodes
203       RandomizeSystem; // Randomize System
204       SISALG(days, NumberOfInitialInfectedNodes, beta, gamma,
    ProbabilityOfInfection, SamplingResult); // Repeat the simulation
205   end
206   else if not frmMain.CancelTriggered then begin
207     { Take missing samplings }
208     for i := 0 to days - SamplingResult.Count do
209       SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected
    .Count));
210
211     { Take a final sampling }
212     SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
    Count));
213
214     //{ If the exitedWhileSimulating is set to true invoke the
    FormClose action. }
215     //if exitedWhileSimulating then self.FormClose(self);
216
217     { Print the samples } { Debug }
218     //for i:=0 to SamplingResult.Count - 1 do begin
219     //  write(IntToStr(i) + ': ');
220     //  for j := 0 to 3 - 1 do begin
221     //    write(SamplingResult[i][j]:2);
222     //    write(', ');
223     //  end;
224     //  writeln();
225     //end;
226
227     SamplingResult.DeleteRange(days + 2, SamplingResult.Count - 1);
228   end;
229 end;
230
231 end.
```

Για να ελεγχθεί εάν αυτοί οι προσαρμοσμένοι αλγόριθμοι είναι έγκυροι, πρέπει να εφαρμοστούν σε ένα ομοιογενές περιβάλλον και στη συνέχεια να συγκριθούν με τα αντίστοιχα αποτελέσματα των διαφορικών εξισώσεων που ανήκουν στα γνωστά διαμερισματικά μοντέλα. Όσο πιο όμοιες είναι οι δύο γραφικές παραστάσεις, τόσο πιο έγκυρος είναι ένας προσαρμοσμένος αλγόριθμος. [11] [45] [46]

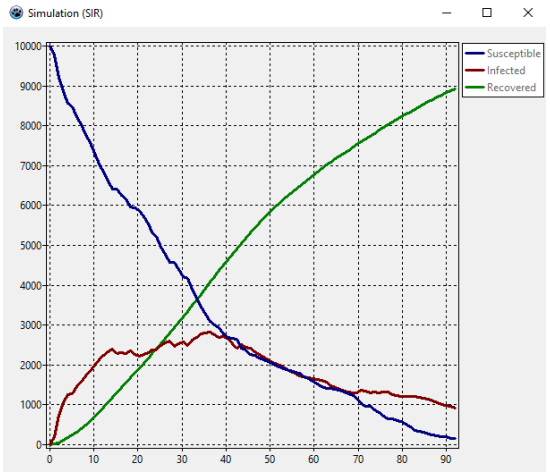


Εικόνα 7.1: BVS | SIR Custom Algorithm validity check, N: 1000, β: 0.2, γ: 0.1, Initial Infected: 1, Days: 160, Seed: 20300081, Probability of Infection: 6%

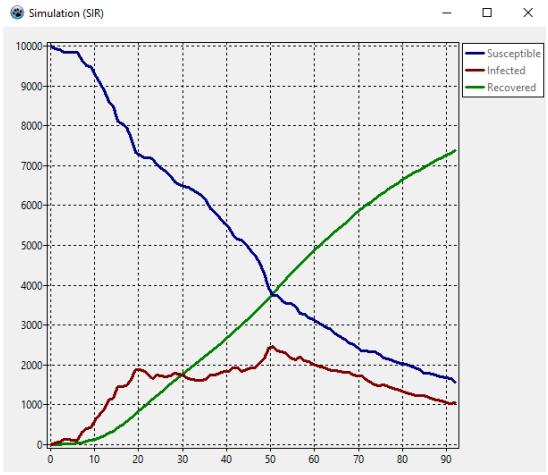


Εικόνα 7.2: BVS | SIS Custom Algorithm validity check, N: 1000, β: 0.2, γ: 0.1, Initial Infected: 1, Days: 160, Seed: 20923765, Probability of Infection: 25%

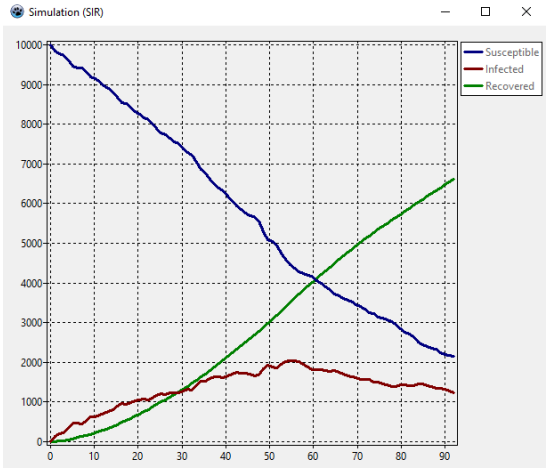
7.3 Προσομοίωση σε τυχαίο και ελεύθερης κλίμακας γράφο



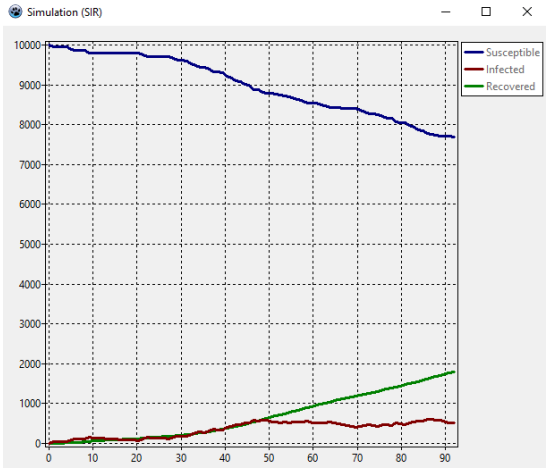
(α) Graph connection probability: 80%



(β) Graph connection probability: 50%



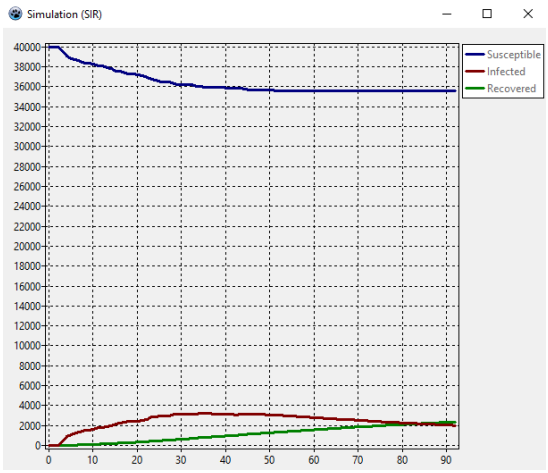
(γ) Graph connection probability: 30%



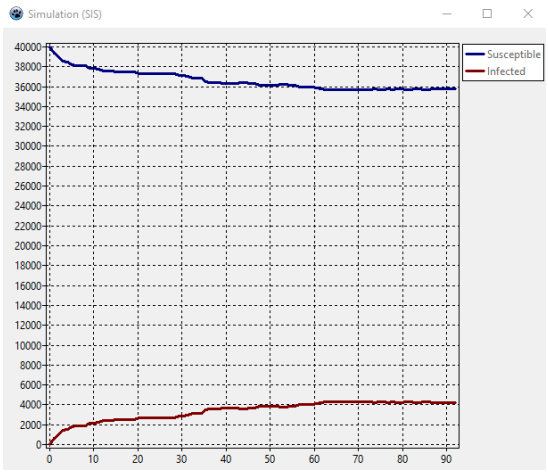
(δ) Graph connection probability: 10%

Εικόνα 7.3: BVS | SIR - ER Graph Simulation, N: 10000, β: 0.05, γ: 0.05, Initial Infected: 1, Days: 90, Probability of Infection: 70%

Παρατηρείται λοιπόν, καθώς προσομοιωθεί η εξάπλωση μιας μολυσματικής ασθένειας σε έναν τυχαίο γράφο ο οποίος πλέον αντικατοπτρίζει καλύτερα τις επαφές που μπορεί να υπάρχουν σε ένα σύνολο ατόμων, ότι πέραν της τοπολογίας, η πυκνότητα του γράφου παίζει έναν σημαντικό ρόλο στο εάν θα εξαπλωθεί μια ασθένεια και στο αν θα προκύψει κάποια επιδημία.

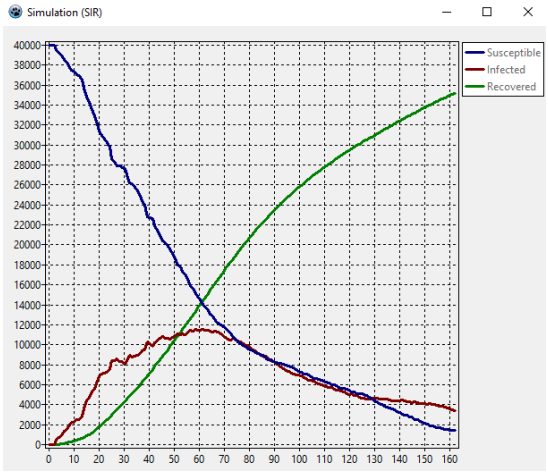


(α) SIR Simulation

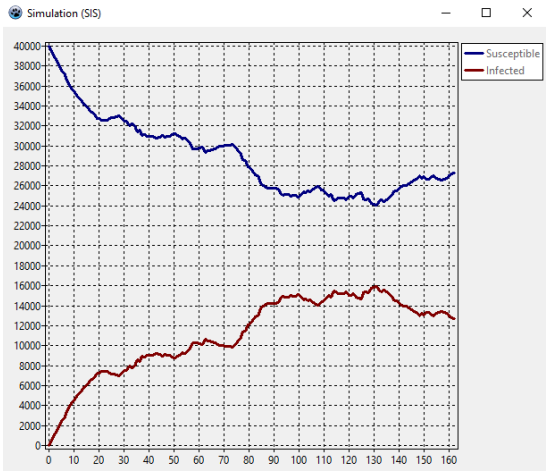


(β) SIS Simulation

Εικόνα 7.4: GRATIS | Scale Free, Adjacency matrix, 40000 Nodes, Initial Connections 1000
BVS | SIR, SIS - Scale Free Graph Simulation, N: 40000, β: 0.8, γ: 0.01, Initial Infected: 10, Days: 90, Probability of Infection: 70%



(α) SIR Simulation



(β) SIS Simulation

Εικόνα 7.5: GRATIS | ER Graph, Adjacency matrix, 40000 Nodes, Probability 50%
BVS | SIR, SIS - ER Graph Simulation, N: 40000, β: 0.031, γ: 0.0305, Initial Infected: 10, Days: 160, Probability of Infection: 50%

Συμπεράσματα

Ο πηγαίος κώδικας του εργαλείου BVS είναι διαθέσιμος στο GitHub (https://github.com/TheoSikla/BVS) υπό την άδεια ανοιχτού κώδικα, η οποία επιτρέπει περαιτέρω βελτιώσεις και προσθήκες. Η αρχιτεκτονική και η δομή του κώδικα του, επιτρέπει την ενσωμάτωση άλλων διαμερισματικών μοντέλων ή την τροποποίηση των ήδη υπαρχόντων. Το ίδιο ισχύει και για τους προσαρμοσμένους επιδημιολογικούς αλγόριθμους. Τα δύο αυτά χαρακτηριστικά της αρχιτεκτονικής του εργαλείου BVS καθώς και άλλες τεχνικές βελτιώσεις, αποτελούν τα θεμέλια των μελλοντικών επεκτάσεων της εφαρμογής.

Αν και η υποστήριξη της κοινότητας της γλώσσας προγραμματισμού Pascal έχει συρρικνωθεί όλα αυτά τα χρόνια, υπάρχει ακόμα ένα υγιές ποσοστό ατόμων που είναι πρόθυμα να βοηθήσουν και να μοιραστούν τις γνώσης τους. Το πιο σημαντικό πλεονέκτημα της Object Pascal είναι η τεράστια επιτάχυνση που προσφέρει λόγω του γεγονότος ότι είναι μια πολύ χαμηλού επιπέδου γλώσσα προγραμματισμού.

Στις μέρες μας οι υπολογιστές που υπάρχουν και είναι διαθέσιμοι στην αγορά, είναι αρκετά ισχυροί έτσι ώστε να μπορούν να υποστηρίξουν με σχετική ευκολία προσομοιώσεις οι οποίες μπορούν να αναπαραστήσουν ολόκληρες πόλεις από άποψη πληθυσμού. Οι τιμές των υπολογιστών αυτών είναι πολύ πιο προσιτές από ότι ήταν στα παλιότερα χρόνια. Το παραπάνω γεγονός αποτελεί μια σημαντική ένδειξη, του ότι πλέον ένας κλασσικός οικιακός υπολογιστής μπορεί κάλλιστα να χρησιμοποιηθεί για να τρέξει αρκετά μεγάλες σε κλίμακα προσομοιώσεις.

Το BVS έχει σχεδιαστεί με σκοπό τον εμπλουτισμό των γνώσεων ενός ατόμου στον τομέα της επιδημιολογίας. Μπορεί επίσης να βοηθήσει τους επιστήμονες να κάνουν μια αρχική ανάλυση μιας μολυσματικής ασθένειας και να προσπαθήσουν να προβλέψουν την πορεία της. Είναι εξαιρετικά σημαντικό, ένα εργαλείο αφιερωμένο στην επιδημιολογία όπως το BVS, να βρίσκεται σε συνεχή ανάπτυξη και βελτίωση καθώς θα αυξήσει την ευαισθητοποίηση του κοινού σχετικά με το πόσο εύκολο μπορεί να είναι για μια μεταδοτική ασθένεια να γίνει επιδημία.

Κλείνοντας, το BVS θα βοηθήσει το ιατρικό προσωπικό των νοσοκομείων να κατανοήσει καλύτερα την δυναμική ανάπτυξη μιας επιδημίας και συνεπώς να προετοιμαστεί έγκαιρα και με τον κατάλληλο τρόπο για την αντιμετώπιση της (ενίσχυση αποθέματος αντισηπτικών προϊόντων και υγιεινής, διοργάνωση πτερύγων και πρόωρη δημιουργία πτερύγων Μ.Ε.Θ.), καθώς επίσης και τους ερευνητές να αποκτήσουν μια πρώτη ματιά της εξέλιξης μιας μολυσματικής ασθένειας και ίσως έτσι να μπορέσουν να αποτρέψουν την διάδοση της.

Βιβλιογραφία

[1] N. Jr Forrest, "Understanding the Interrelationships Between Botanical, Human, and Veterinary Epidemiology: The Ys and Rs of It All" Ecosystem Health, 2001.

[2] A. Morabia, "A history of epidemiologic methods and concepts", pp. 93, Birkhäuser, 2004.

[3] P. Martin, E. Martin-Granel, "2,500-year Evolution of the Term Epidemic", 2006.

[4] Principles of Epidemiology, Third Edition. Atlanta, Georgia: Centers for Disease Control and Prevention. 2012.

[5] M. S. Green, T. Swartz, E. Mayshar, B. Lev, A. Leventhal, P. E. Slater, J. Shemer, "When is an epidemic an epidemic¿', The Israel Medical Association Journal, pp. 3–6, PMID 11802306, January 2002.

[6] P. Vinten-Johansen, H. Brody, N. Paneth, S. Rachman, M. Rip, D. Zuck, "Cholera, Chloroform, and the Science of Medicine: A Life of John Snow.", Oxford University Press, p. 30, ISBN 9780199747887, 2003.

[7] J. Galbraith Simmons, "Joseph Lister Antisepsis and Modern Surgery", Doctors and discoveries: lives that created today's medicine. Boston: Houghton Mifflin. pp. 94–99, ISBN 978-0618152766, 2002.

[8] B. W. Lerner, K. L. Lerner, "Robert Koch. World of Microbiology and Immunology", Detroit: Gale, 2006.

[9] G. Davey Smith, D. Kuh, "Commentary: William Ogilvy Kermack and the childhood origins of adult health and disease", International Journal of Epidemiolog, pp. 696–703, 1 August 2001.

[10] N. Bacaër, "McKendrick and Kermack on epidemic modelling (1926–1927). In: A Short History of Mathematical Population Dynamics.", Springer, London, 2011.

[11] Kermack, W. O.; McKendrick, A. G., "A Contribution to the Mathematical Theory of Epidemics.", Proceedings of the Royal Society A, pp. 700–721, 1927.

[12] U. Muellner, G. Fournié, P. Muellner, C. Ahlstrom, Dirk U. Pfeiffer, "Epidemix - An interactive multi-model application for teaching and visualizing infectious disease transmission", Epidemics, 2018.

[13] W. Chang, J. Cheng, J. Allaire, Y. Xie, J. McPherson "Shiny: Web Application Framework for R. R Packag.", 2015.

[14] W. Van den Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, A. Vespignani, "The GLEaMviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale", BMC Infectious Diseases, 2011.

[15] M. Mniszewski Susan, Y. Del Valle Sara, D. Stroud Phillip, M. Riese Jane, J. Sydoriak Stephen, "EpiSimS Simulation of a Multi-Component Strategy for Pandemic Influenza", EpiSimS Simulation of a Multi-Component Strategy for Pandemic Influenza, Ottawa, Canada, 2008.

[16] C. Barrett, R. Beckman, K. Berkbigler, "TRANSIMS: Transportation analysis simulation", 2000.

[17] Yoneki Eiko, “FluPhone study: virtual disease spread using haggle.“, 2011.

[18] F. Alvarez, P. Crépey, M. Barthelemy, A. Valleron, “sispread: A Software to Simulate Infectious Diseases Spreading on Contact Networks“, Methods of information in medicine, 2007.

[19] R. Beckley, Cametria Weatherspoon, M. Alexander, M. Chandler, A. Johnson, G. S. Bhatt, “Modeling epidemics with differential equations“, Tennessee State University Internal Report, 2013, Retrieved July 19, 2020.

[20] H. W. Hethcote, “Three Basic Epidemiological Models“, In Levin, Simon A., Hallam, Thomas G., Gross, Louis J, Applied Mathematical Ecology. Biomathematics. 18. Berlin: Springer. pp. 119–144, 1989.

[21] R. Parshani, S. Carmi, S. Havlin, “Epidemic Threshold for the Susceptible-Infectious-Susceptible Model on Random Networks“, Phys. Rev. Lett, 2010.

[22] H. Hethcote, Ma Zhien, Shengbing Liao, “Effects of quarantine in six endemic models for infectious diseases.“, Mathematical biosciences, 180. 141-60, 2002.

[23] S. Chinviriyasit, W. Chinviriyasit, “Global Stability of an SIQ Epidemic Model.“, pp. 225-228, 2007.

[24] Zhang Xiao-Bing, Huo Hai-Feng, Xiang Hong, Shi Qihong, Li Dungang, “The threshold of a stochastic SIQS epidemic model.“, Physica A: Statistical Mechanics and its Applications, 2017.

[25] Zhang Xiao-Bing, Shi Qihong, Ma Shuang-Hong, Huo Hai-Feng, Li Dungang, “Dynamic behavior of a stochastic SIQS epidemic model with Lévy jumps.“, Nonlinear Dynamics, 2018.

[26] Cao Zhongwei, Zhou Shengjuan, “Dynamical Behaviors of a Stochastic SIQR Epidemic Model with Quarantine-Adjusted Incidence.“, Discrete Dynamics in Nature and Society, 2018.

[27] T. J. Bailey Norman, “The mathematical theory of infectious diseases and its applications“, London: Griffin, ISBN 0-85264-231-8, 1975.

[28] Bichara Derdei, Iggidr Abderrahman, Sallet Gauthier, “Global analysis of multi-strains SIS, SIR and MSIR epidemic models.“, Journal of Applied Mathematics and Computing, 2014.

[29] Na Yi, Qingling Zhang, Kun Mao, Dongmei Yang, Qin Li, “Analysis and control of an SEIR epidemic system with nonlinear transmission rate“, Mathematical and Computer Modelling, Volume 50, Issues 9–10, 2009.

[30] Wang Jinghai, “Analysis of an SEIS Epidemic Model with a Changing Delitescence.“, Abstract and Applied Analysis, 2012.

[31] R. Almeida, A. Brito da Cruz, N. Martins, M. Monteiro, “An epidemiological MSEIR model described by the Caputo fractional derivative.“, International Journal of Dynamics and Control, International Journal of Dynamics and Control, 2019.

[32] C. Pinto, A. Carvalho, “A latency fractional order model for HIV dynamics.“, Journal of Computational and Applied Mathematics, 312, 2016.

[33] Z. Zainulabadin, R. Kashif, M. Mushtaq, “HIV/AIDS epidemic fractional-order model.“, Journal of Difference Equations and Applications, 2017.

[34] Rihan Fathalla, Venkattaraman Preethi, Rakkiyappan Rajan, Gandhi Velmurugan, “A fractional-Order Delay Differential Model for Ebola Infection and CD8 + T-cells Response: Stability analysis and Hopf bifurcation.“, International Journal of Biomathematics, 2017.

[35] Venkattaraman Preethi, Rihan Fathalla, Rakkiyappan Rajan, Gandhi Velmurugan, “A fractional-order model for Ebola virus infection with delayed immune response on heterogeneous complex networks.“, Journal of Computational and Applied Mathematics, 2018.

[36] M. El-shahed, A. Alsaedi, “The fractional SIRC model and influenza A.“, Mathematical Problems in Engineering, 2011.

[37] Rihan Fathalla, “On Fractional Order Cancer Model.“, Advances in Calculus of Variations, 2012.

[38] S. Pooseh, H. Rodrigues, F. M. Torres Delfim, “Fractional Derivatives in Dengue Epidemics.“, AIP Conference Proceedings, 2011.

[39] H. Al-Sulami, M. El-shahed, Nieto Juan, Shammakh Wafa, “On Fractional Order Dengue Epidemic Model.“, Mathematical Problems in Engineering, 2014.

[40] E. Okyere, F. Oduro, S. Amponsah Kwame, I. Dontwi, “Fractional Order Optimal Control Model For Malaria Infection.“, 2016.

[41] Rihan Fathalla, Baleanu Dumitru, S. Lakshmanan, Rakkiyappan Rajan, “On Fractional SIRC Model with Salmonella Bacterial Infection.“, Abstract and Applied Analysis, 2014.

[42] F. Ndairou, I. Area, J. Nieto, C. Silva, F. M. Torres Delfim, “Mathematical modeling of Zika disease in pregnant women and newborns with microcephaly in Brazil.“, Mathematical Methods in the Applied Sciences, 2017.

[43] He Shaobo, Peng Yuexi, Sun Kehui, “SEIR modeling of the COVID-19 and its dynamics.“, Nonlinear Dynamics, 2020.

[44] V. Vlachos, T. Siklafidis, K. Chantzi, “GRATIS: A GRaph Tool for Information Systems Scientists” 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), pp. 1–6, Piraeus, Greece, November 2019.

[45] D. Daley, J. Gani, “Epidemic Modelling: An Introduction“, Cambridge Studies in Mathematical Biology, Cambridge: Cambridge University Press, doi:10.1017/CBO9780511608834, 1999.

[46] V. Vlachos, “Security applications of peer to peer networks“, Economical University of Athens, Department of Management Science and Technology, PhD, doi:10.12681/eadd/17693, url: http://hdl.handle.net/10442/hedi/17693, pp. 106, 120-122, Greece, 2007.

Παράρτημα - Πηγαίος Κώδικας

Η ενότητα αυτή παρουσιάζει τα πιο σημαντικά αποσπάσματα κώδικα που απαρτίζουν την εφαρμογή BVS.

Α.1 Κλάση που αντιπροσωπεύει έναν κόμβο και τα χαρακτηριστικά του

```
unit clNode_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, ExtCtrls, Graphics, fgl;

type
  ArrayOfWord = Array of Word;
  PtrOfTShape = ^TShape;
  TWordList = specialize TFPGList<Word>;

type
  TNode = class(TObject)
    private
      FId: Word;
      FIsSusceptible: Boolean;
      FIsInfected: Boolean;
      FIsRecovered: Boolean;
      FInfectedByNode: Word;
      FNeighbors: TWordList;
      FPtrShape: PtrOfTShape;
    protected
      { protected declarations here }
    public
      constructor Create(
        AId: Word;
        ANeighbors: TWordList = Nil;
        AIsSusceptible: Boolean = false;
        AIsInfected: Boolean = false;
        AIsRecovered: Boolean = false);

      destructor Destroy; override;

      property Id: Word read FId write FId;
      property IsSusceptible: Boolean read FIsSusceptible write
   FIsSusceptible;
      property IsInfected: Boolean read FIsInfected write FIsInfected;
      property IsRecovered: Boolean read FIsRecovered write
   FIsRecovered;
      property InfectedByNode: Word read FInfectedByNode write
   FInfectedByNode;
      property Neighbors: TWordList read FNeighbors;
      property PtrShape: PtrOfTShape read FPtrShape write FPtrShape;

```

```
45        function GetNumberOfNeighbors(): Word;
46
47        procedure MakeSusceptible;
48        procedure Infect(InfectorNode: Word);
49        procedure Recover;
50        procedure Restore;
51
52      published
53        { published declarations here }
54    end;
55
56  implementation
57    constructor TNode.Create(
58      AId: Word;
59      ANeighbors: TWordList = Nil;
60      AIsSusceptible: Boolean = false;
61      AIsInfected: Boolean = false;
62      AIsRecovered: Boolean = false);
63    begin
64       Fid := AId;
65       FIsSusceptible := AIsSusceptible;
66       FIsInfected := AIsInfected;
67       FIsRecovered := AIsRecovered;
68       FNeighbors := TWordList.Create;
69       FNeighbors := ANeighbors;
70       FIsSusceptible := False;
71       FIsInfected := False;
72       FIsRecovered := False;
73    end;
74
75    destructor TNode.Destroy;
76    begin
77       inherited; // Also call parent class destroyer
78    end;
79
80    function TNode.GetNumberOfNeighbors(): Word;
81    begin
82      Result := self.Neighbors.Count;
83    end;
84
85    procedure TNode.MakeSusceptible;
86    begin
87      self.IsSusceptible := True;
88      self.IsInfected := False;
89      self.IsRecovered := False;
90    end;
91
92    procedure TNode.Infect(InfectorNode: Word);
93    begin
94      self.IsSusceptible := False;
95      self.IsInfected := True;
96      self.IsRecovered := False;
97
98      self.InfectedByNode := InfectorNode;
99      self.PtrShape^.Brush.Color := clRed;
100   end;
101
102   procedure TNode.Recover;
103   begin
104     self.IsSusceptible := False;
```

```
105     self.IsInfected := False;
106     self.IsRecovered := True;
107   end;
108
109   procedure TNode.Restore;
110   begin
111     self.IsSusceptible := True;
112     self.IsInfected := False;
113     self.IsRecovered := False;
114
115     self.InfectedByNode := 0;
116     self.PtrShape^.Brush.Color := clMedGray;
117   end;
118
119 end.
```

A.2 Επιλυτής διαφορικών εξισώσεων με μέθοδο Forward Euler

```
1  unit utlEuler_u;
2
3  {$mode objfpc}{$H+}
4
5  interface
6
7  uses
8    Classes, SysUtils,
9    { Forms }
10   { Classes }
11   { Utilities }
12   utlTypes_u,
13   utlEnum_u,
14   utlSIR_u,
15   utlSIS_u,
16   utlSIQ_u,
17   utlSIQS_u,
18   utlSIQR_u,
19   utlSIRD_u,
20   utlMSIR_u,
21   utlSEIR_u,
22   utlSEIS_u,
23   utlMSEIR_u;
24
25 function odeEuler(model: String; t, y0: ArrayOfDouble; extraArgs:
       ArrayOfDouble): ArrayOfArrayOfDouble;
26
27 implementation
28   function odeEuler(model: String; t, y0: ArrayOfDouble; extraArgs:
       ArrayOfDouble): ArrayOfArrayOfDouble;
29   var
30     i, j: Integer;
31     DiffEquations: ArrayOfArrayOfDouble;
32     dXdts, x: ArrayOfDouble;
33   begin
34     {
35
36
```

```
    [ Description ]
    [*************************************************************]
    Euler method (also called forward Euler method) a first-order
numerical procedure for solving ordinary differential equations (
ODEs) with a given initial value.

    [*************************************************************]

    [ Parameters ]
    [*************************************************************]
    Param t: An array of a specified duration with equal parts.
    Param y0: The initial system state.
    Param extraArgs: Array of double containing the appropriate
parameters needed in order to calculate the target differential
equations.

    [*************************************************************]

    [ Variables ]
    [*************************************************************]
    Var i, j: Counters.
    Var DiffEquations: Array of Arrays containing double numbers that
 holds the whole system's state.

    Var dXdts: Calculated differential equation result returned from
a differential equation calculating function.

    Var x: Temporary array that holds the next system's state to be
calculated.

    [*************************************************************]
  }

  SetLength(DiffEquations, Length(y0));
  SetLength(x, Length(y0));

  for i := 0 to Length(DiffEquations) - 1 do begin
    SetLength(DiffEquations[i], Length(t));
    DiffEquations[i][0] := y0[i];
  end;

  for i := 0 to Length(t) - 1 do begin
    for j := 0 to Length(DiffEquations) - 1 do begin
      x[j] := DiffEquations[j][i];
    end;

    { Invoke the appropriate differential equation calculator
function }
    case model of
      SIR: dXdts := SIRDE(x, extraArgs);
      SIS: dXdts := SISDE(x, extraArgs);
      SIQ: dXdts := SIQDE(x, extraArgs);
      SIQS: dXdts := SIQSDE(x, extraArgs);
      SIQR: dXdts := SIQRDE(x, extraArgs);
      SIRD: dXdts := SIRDDE(x, extraArgs);
      MSIR: dXdts := MSIRDE(x, extraArgs);
      SEIR: dXdts := SEIRDE(x, extraArgs);
      SEIS: dXdts := SEISDE(x, extraArgs);
      MSEIR: dXdts := MSEIRDE(x, extraArgs);
    end;
```

```
89
90        for j := 0 to Length(dXdts) - 1 do begin
91          DiffEquations[j][i + 1] := DiffEquations[j][i] + dXdts[j] * (t[
    i + 1] - t[i]);
92        end;
93      end;
94
95      { Print the calculated differential equations }
96      //for i:=0 to Length(DiffEquations) - 1 do begin
97      //  for j := 0 to Length(t) - 1 do begin
98      //    write(DiffEquations[i][j]:0:2);
99      //    write(', ');
100     //  end;
101     //  writeln();
102     //end;
103
104     Result := DiffEquations;
105   end;
106 end.
```

A.3 Κώδικας διαχείρισης αρχείων

```
1  unit utlFile_u;
2
3  {$mode objfpc}{$H+}
4
5  interface
6
7  uses
8    Classes, SysUtils, RegExpr, Dialogs, fpjson, jsonparser,
9    clNode_u, utlArray_u, utlTypes_u, utlConstants_u;
10
11 type
12   TAppenderType = specialize TAppender<TNode>;
13   ArrayOfTNodeType = Array of TNode;
14
15   TFileHandler = class(TObject)
16     public
17       function IsAdjacencyMaxtrixOrListFile(filename, Atype: String):
     Boolean;
18       function LoadAdjacencyMaxtrix(filename: String): TListOfTNode;
19       function LoadAdjacencyList(filename: String): TListOfTNode;
20       procedure WriteStringToFile(filename: string; data: String);
21       procedure WriteToJsonFile(filename: string; data: TJSONData);
22       function LoadJsonFile(filename: string): TJSONData;
23
24
25   end;
26
27 var
28   FileHandler: TFileHandler;
29
30 implementation
31   uses
32   { Forms }
33   frmMain_u;
34
```

```
function TFileHandler.IsAdjacencyMaxtrixOrListFile(filename, Atype:
    String): Boolean;
var
  tfIn: TextFile;
  s: String;
  RegexObj: TRegExpr;
begin
  Result := False;
  AssignFile(tfIn, filename);
  RegexObj := TRegExpr.Create;
  if Atype = 'matrix' then begin
    RegexObj.Expression := ADJACENCY_MATRIX_REGEX;
  end
  else
    RegexObj.Expression := ADJACENCY_LIST_REGEX;

  try
    try
      reset(tfIn);
      if not eof(tfIn) then
         begin
           readln(tfIn, s);
           if RegexObj.Exec(s) then Result := True;
         end;
    except
      on E: EInOutError do
       writeln('File handling error occurred. Details: ', E.Message);
    end;

  finally
    CloseFile(tfIn);
    RegexObj.Free;
  end;

end;

function TFileHandler.LoadAdjacencyMaxtrix(filename: String):
    TListOfTNode;
var
  tfIn: TextFile;
  s: String;
  i: Integer;
  lineLength: Integer;
  RegexObj: TRegExpr;
  Nodes: TListOfTNode;
  appender: TAppenderType;
  obj: TNode;
  Neighbors: TWordList;
  c: char;
  RowIndex, ColumnIndex: Word;
begin
  // Set the name of the file that will be read
  AssignFile(tfIn, filename);

  // Create a regex to validate file structure
  RegexObj := TRegExpr.Create;
  RegexObj.Expression := ADJACENCY_MATRIX_REGEX;

  Nodes := TListOfTNode.Create;     // Initialize list
  appender := TAppenderType.Create; // Initialize instance
```

```
93
94  frmMain.NumberOfEdges := 0;
95  frmMain.AvgNumberOfNeighbors := 0;
96  i := 0; RowIndex := 0; ColumnIndex := 0;
97  try
98    {
99      Embed the file handling in a try/except block to handle
100     errors gracefully
101   }
102   try
103     // Open the file for reading
104     reset(tfIn);
105
106     // Grab the first line s length
107     if not eof(tfIn) then
108        begin
109          readln(tfIn, s);
110          lineLength := Length(s);
111
112          Neighbors := TWordList.Create;
113
114          for c in s do begin
115            if c = '1' then
116               begin
117                 Neighbors.Add(ColumnIndex);
118                 if (ColumnIndex > RowIndex) then Inc(frmMain.
 NumberOfEdges);
119              end;
120            Inc(ColumnIndex);
121         end;
122         ColumnIndex := 0;
123         Inc(RowIndex);
124
125         frmMain.AvgNumberOfNeighbors += Neighbors.Count;
126
127         obj := TNode.Create(i, Neighbors);
128         Inc(i);
129         Nodes.Add(obj);
130       end;
131
132    // Keep reading lines until the end of the file is reached
133    while not eof(tfIn) do
134    begin
135      readln(tfIn, s);
136
137      // Validate each line of the file
138      if not RegexObj.Exec(s) or (Length(s) <> lineLength) then
139         begin
140           MessageDlg('Error', 'Invalid file format', mtError, [mbOK
 ], 0);
141           Break; // Jump to the 'finally' block
142         end;
143
144      Neighbors := TWordList.Create;
145
146      for c in s do begin
147        if c = '1' then
148           begin
149             Neighbors.Add(ColumnIndex);
150             if (ColumnIndex > RowIndex) then Inc(frmMain.
```

```
    NumberOfEdges);
151             end;
152           Inc(ColumnIndex);
153         end;
154         ColumnIndex := 0;
155         Inc(RowIndex);
156
157         frmMain.AvgNumberOfNeighbors += Neighbors.Count;
158
159         obj := TNode.Create(i, Neighbors);
160         Inc(i);
161         Nodes.Add(obj);
162         //writeln('Node: ' + IntToStr(i) + ' has {' + IntToStr(
    Neighbors.Count) + '} neighbors'); { Debug }
163       end;
164
165       frmMain.AvgNumberOfNeighbors := frmMain.AvgNumberOfNeighbors div
    Nodes.Count;
166       //writeln('Average number of neighbors: ' + IntToStr(frmMain.
    AvgNumberOfNeighbors)); { Debug }
167
168     except
169       on E: EInOutError do
170        writeln('File handling error occurred. Details: ', E.Message);
171     end;
172
173   finally
174     CloseFile(tfIn);      // Close the file
175     RegexObj.Free;        // Free the regex object
176     FreeAndNil(appender); // Free the generic appender instance
177
178     result := Nodes;
179   end;
180
181 end;
182
183 function TFileHandler.LoadAdjacencyList(filename: String): TListOfTNode
    ;
184 var
185   tfIn: TextFile;
186   s, neighbor: String;
187   RegexObj: TRegExpr;
188   Nodes: TListOfTNode;
189   obj: TNode;
190   Neighbors: TWordList;
191   colonSplit, commaSplit: TStringArray;
192 begin
193   AssignFile(tfIn, filename);
194
195   RegexObj := TRegExpr.Create;
196   RegexObj.Expression := ADJACENCY_LIST_REGEX;
197
198   Nodes := TListOfTNode.Create;
199
200   frmMain.NumberOfEdges := 0;
201   frmMain.AvgNumberOfNeighbors := 0;
202   try
203     try
204       reset(tfIn);
205
```

```
206       while not eof(tfIn) do
207       begin
208         readln(tfIn, s);
209
210         // Validate each line of the file
211         if not RegexObj.Exec(s) then
212            begin
213              MessageDlg('Error', 'Invalid file format', mtError, [mbOK
    ], 0);
214              Break; // Jump to the 'finally' block
215            end;
216
217         Neighbors := TWordList.Create;
218
219         colonSplit := s.Split(':');
220         commaSplit := colonSplit[1].Split(',');
221         if not (commaSplit[0] = '') then begin
222           for neighbor in commaSplit do begin
223             Neighbors.Add(StrToInt(neighbor));
224           end;
225         end;
226
227         frmMain.AvgNumberOfNeighbors += Neighbors.Count;
228
229         obj := TNode.Create(StrToInt(colonSplit[0]), Neighbors);
230         Nodes.Add(obj);
231         //writeln('Node: ' + colonSplit[0] + ' has {' + IntToStr(
    Neighbors.Count) + '} neighbors'); { Debug }
232       end;
233
234       frmMain.AvgNumberOfNeighbors := frmMain.AvgNumberOfNeighbors div
    Nodes.Count;
235       //writeln('Average number of neighbors: ' + IntToStr(frmMain.
    AvgNumberOfNeighbors)); { Debug }
236
237     except
238       on E: EInOutError do
239        writeln('File handling error occurred. Details: ', E.Message);
240     end;
241
242   finally
243     CloseFile(tfIn); // Close the file
244     RegexObj.Free;   // Free the regex object
245
246     result := Nodes;
247   end;
248
249 end;
250
251 procedure TFileHandler.WriteStringToFile(filename: string; data: String
    );
252 var
253   tfOut: TextFile;
254 begin
255   try
256     AssignFile(tfOut, filename);
257     Rewrite(tfOut);
258     writeln(tfOut, data);
259     closefile(tfOut);
260   except
```

```
    on E:Exception do
  end;
end;

procedure TFileHandler.WriteToJsonFile(filename: string; data:
   TJSONData);
begin
  self.WriteStringToFile(filename, data.FormatJSON);
end;

function TFileHandler.LoadJsonFile(filename: string): TJSONData;
var
  tfIn: TextFile;
  s, JsonString: String;
begin
  s := '';
  JsonString := '';

  try
    try
      AssignFile(tfIn, filename);
      reset(tfIn);

      while not eof(tfIn) do begin
        readln(tfIn, s);
        JsonString += s;
      end;

      Result := GetJSON(JsonString);
    except
      on E: EInOutError do
       writeln('File handling error occurred. Details: ', E.Message);
    end;
  finally
      CloseFile(tfIn);
  end;

end;

end.
```

A.4 Διαχείριση ρυθμίσεων εφαρμογής

```
unit frmSettings_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
   fpjson,
  { Utilities }
  utlFile_u,
  utlConstants_u,
  utlValidation_u;

type
```

```
15
16    { TfrmSettings }
17
18    TfrmSettings = class(TForm)
19      btnCancel: TButton;
20      btnApply: TButton;
21      cbxReSimulate: TCheckBox;
22      edtReSimulateMinRecoveredNodeCount: TEdit;
23      lblMinRecoveredSIR: TLabel;
24      lblSimulation: TLabel;
25
26      procedure btnCancelClick(Sender: TObject);
27      procedure btnApplyClick(Sender: TObject);
28      procedure cbxReSimulateChange(Sender: TObject);
29      procedure edtReSimulateMinRecoveredNodeCountKeyPress(Sender:
       TObject; var Key: char);
30      procedure FormClose(Sender: TObject);
31      procedure FormCreate(Sender: TObject);
32      procedure LoadSettings;
33    private
34
35    public
36
37    end;
38
39  var
40    frmSettings: TfrmSettings;
41
42  implementation
43
44  procedure TfrmSettings.LoadSettings;
45  var
46    i, j: Integer;
47    JsonData: TJSONData;
48    JsonObject, settings: TJSONObject;
49  begin
50    { Generate default settings if they do not exist }
51    if not FileExists(SETTINGS_FILE_NAME) then FileHandler.
       WriteToJsonFile(SETTINGS_FILE_NAME, GetJSON(DEFAULT_SETTINGS));
52    { Load the settings }
53    try
54      JsonData := FileHandler.LoadJsonFile(SETTINGS_FILE_NAME);
55      JsonObject := TJSONObject(JsonData);
56      settings := JsonObject.Find(SETTINGS_NAME) as TJSONObject;
57      for i := 0 to settings.Count - 1 do begin
58        for j := 0 to self.ComponentCount - 1 do begin
59          if self.Components[j].name = settings.Names[i] then begin
60            if self.Components[j] is TCheckBox then (self.Components[j]
       as TCheckbox).Checked := settings.FindPath(TJSONObject(settings).
       Names[i]).AsBoolean
61          else if self.Components[j] is TEdit then (self.Components[j]
       as TEdit).Text := settings.FindPath(TJSONObject(settings).Names[i]).
       AsString;
62        end;
63      end;
64      end;
65    except on E:Exception do begin end;
66    end;
67  end;
68
```

```
procedure TfrmSettings.FormCreate(Sender: TObject);
begin
  inherited;
  self.LoadSettings; // Load Application Settings
  if self.cbxReSimulate.Checked then self.
   edtReSimulateMinRecoveredNodeCount.Enabled := True
  else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
end;

procedure TfrmSettings.FormClose(Sender: TObject);
begin
  self.LoadSettings;
  inherited;
end;

procedure TfrmSettings.btnCancelClick(Sender: TObject);
begin
  self.Close;
end;

procedure TfrmSettings.btnApplyClick(Sender: TObject);
var
 JsonObject, settings: TJSONObject;
begin
  { Load the default settings }
  JsonObject := TJSONObject(GetJSON(DEFAULT_SETTINGS));

  { Get the new settings }
  settings := JsonObject.Find(SETTINGS_NAME) as TJSONObject;
  settings.Booleans[RE_SIMULATE_SETTING_NAME] := self.cbxReSimulate.
   Checked;
  if self.edtReSimulateMinRecoveredNodeCount.Text <> '' then settings.
   Strings[RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] :=
   self.edtReSimulateMinRecoveredNodeCount.Text
  else settings.Strings[
   RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] := '0';

  { Write the new settings }
  FileHandler.WriteToJsonFile(SETTINGS_FILE_NAME, TJSONData(JsonObject)
   );

  self.LoadSettings; // Load Application Settings
end;

procedure TfrmSettings.cbxReSimulateChange(Sender: TObject);
begin
  if self.cbxReSimulate.Checked then begin
    self.edtReSimulateMinRecoveredNodeCount.Enabled := True;
  end
  else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
end;

procedure TfrmSettings.edtReSimulateMinRecoveredNodeCountKeyPress(
   Sender: TObject; var Key: char
  );
begin
  ValidateInteger(Sender, Key);
end;

{$R *.lfm}
```

```
122
123 end.
```

A.5 Σταθερές μεταβλητές εφαρμογής

```
1  unit utlConstants_u;
2
3  {$mode objfpc}{$H+}
4
5  interface
6
7  uses
8    Classes, SysUtils;
9
10 const
11   sLineBreak = {$IFDEF LINUX} AnsiChar(#10) {$ENDIF}
12                {$IFDEF MSWINDOWS} AnsiString(#13#10) {$ENDIF};
13
14 { Filenames }
15 SETTINGS_FILE_NAME = 'settings.json';
16
17 { Settings }
18 SETTINGS_NAME = 'settings';
19 RE_SIMULATE_SETTING_NAME = 'cbxReSimulate';
20 RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME = '
     edtReSimulateMinRecoveredNodeCount';
21 DEFAULT_SETTINGS = '{' + sLineBreak +
22                    '  "' + SETTINGS_NAME + '": {' + sLineBreak +
23                    '    "' + RE_SIMULATE_SETTING_NAME + '": False,' +
24                    '    "' +
     RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME + '": 0' +
25                    '  }' + sLineBreak +
26                    '}';
27
28 { Regex }
29 ADJACENCY_MATRIX_REGEX = '^[0-1]*$';
30 ADJACENCY_LIST_REGEX = '(^\d+:\d+(,\d+)*$)|(^\d+:$)';
31
32 implementation
33
34 end.
```

A.6 Φόρμες

A.6.1 Αρχική

```
1  unit frmMain_u;
2
3  {$mode objfpc}{$H+}
4
5  interface
6
7  uses
8    Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
      ExtCtrls,
```

```
  Menus, TAGraph, typinfo, fpjson,
  { Forms }
  frmSettings_u,
  { Utilities }
  utlFile_u,
  utlValidation_u,
  utlEnum_u,
  utlTypes_u,
  utlMisc,
  { Epidemic Algorithms }
  AlgSIR_u,
  AlgSIS_u;

type

  { TfrmMain }

  TfrmMain = class(TForm)
    btnImportDialog: TOpenDialog;
    btnSimulate: TButton;
    btnCancel: TButton;
    cbxAvailableModels: TComboBox;
    ckbUseSystemSeed: TCheckBox;
    edtSeed: TEdit;
    edtProbabilityOfInfection: TEdit;
    edtN: TEdit;
    edtDelta1: TEdit;
    edtEpsilon: TEdit;
    edtDays: TEdit;
    edtBeta: TEdit;
    edtAlpha: TEdit;
    edtZeta: TEdit;
    edtKappa: TEdit;
    edtGamma: TEdit;
    edtInitialInfected: TEdit;
    edtMaternallyDerivedImmunity: TEdit;
    edtMu: TEdit;
    edtLambda: TEdit;
    edtDelta: TEdit;
    frmTimer: TTimer;
    lblSeed: TLabel;
    lblProbabilityOfInfection: TLabel;
    lblN: TLabel;
    lblDelta1: TLabel;
    lblEpsilon: TLabel;
    lblDays: TLabel;
    lblBeta: TLabel;
    lblAlpha: TLabel;
    lblZeta: TLabel;
    lblKappa: TLabel;
    lblGamma: TLabel;
    lblMaternallyDerivedImmunity: TLabel;
    lblMu: TLabel;
    lblInitialInfected: TLabel;
    lblLambda: TLabel;
    lblDelta: TLabel;
    mnuSettings: TMenuItem;
    mnuFileClose: TMenuItem;
    mnuMainMenu: TMainMenu;
    mnuFileOpen: TMenuItem;
```

```
69      mnuFile: TMenuItem;
70
71      procedure btnCancelClick(Sender: TObject);
72      procedure ckbUseSystemSeedChange(Sender: TObject);
73      procedure cbxAvailableModelsChange(Sender: TObject);
74      procedure mnuFileCloseClick(Sender: TObject);
75      procedure mnuSettingsClick(Sender: TObject);
76      procedure RefreshGUI;
77      procedure FormCreate(Sender: TObject);
78      procedure btnSimulateClick(Sender: TObject);
79      procedure mnuFileOpenClick(Sender: TObject);
80      procedure edtFloatKeyPress(Sender: TObject; var Key: char);
81      procedure edtIntegerKeyPress(Sender: TObject; var Key: char);
82      procedure edtKeyUpEnter(Sender: TObject; var Key: char);
83      procedure preparePreSimulationChart;
84      procedure prepareSimulationChart(SamplingResult:
       TArrayOfArrayOfWord);
85      procedure registerAvailableModels;
86      procedure AvailableSimulationCheck;
87      function validateSimulationFields: Boolean;
88      procedure InitiateSimulation(Sender: TObject);
89      function getN: Integer;
90    private
91
92    public
93      Nodes: TListOfTNode;
94      NumberOfEdges: Longword;
95      AvgNumberOfNeighbors: Longword;
96      CancelTriggered: Boolean;
97
98    end;
99
100 var
101   frmMain: TfrmMain;
102
103 implementation
104
105 uses
106   { Forms }
107   frmSimulation_u,
108   frmSimulationChart_u,
109   frmPreSimulationChart_u;
110
111 {$R *.lfm}
112
113 { TfrmMain }
114
115 procedure TfrmMain.mnuFileOpenClick(Sender: TObject);
116 var
117   filename: string;
118 begin
119   self.btnSimulate.Enabled := False;
120
121   if btnImportDialog.Execute then
122   begin
123     filename := btnImportDialog.Filename;
124   end;
125
126   if filename <> '' then begin
127     if self.Nodes.Count > 0 then frmSimulation.ResetShapes;
```

```
128     if FileHandler.IsAdjacencyMaxtrixOrListFile(filename, 'matrix')
        then begin
129       Nodes := FileHandler.LoadAdjacencyMaxtrix(filename);
130     end
131     else if FileHandler.IsAdjacencyMaxtrixOrListFile(filename, 'list')
        then begin
132       Nodes := FileHandler.LoadAdjacencyList(filename);
133     end
134     else MessageDlg('Error', 'Unsupported file type', mtError, [mbOK],
        0);
135
136     if self.Nodes.Count > 0 then begin
137       //frmSimulation.RenderShapes;
138       self.edtN.Enabled := False;
139       self.mnuFileClose.Enabled := True;
140     end;
141
142     if self.validateSimulationFields then self.
        preparePreSimulationChart;
143   end;
144
145   self.cbxAvailableModelsChange(self);
146
147 end;
148
149 procedure TfrmMain.btnSimulateClick(Sender: TObject);
150 begin
151   if self.validateSimulationFields and
152      (self.edtProbabilityOfInfection.Text <> '') then begin
153     self.btnSimulate.Enabled := False;
154     self.btnSimulate.Visible := False;
155     self.btnCancel.Enabled := True;
156     self.btnCancel.Visible := True;
157     self.CancelTriggered := False;
158
159     { Do not allow the close file function to be invokable while
        simulating }
160     self.mnuFileClose.Enabled := False;
161     //if self.btnSimulate.IsEnabled then begin
162     //   frmSimulation.Show;
163     //end;
164
165     { Randomize System }
166     RandomizeSystem;
167
168     frmSimulation.frmSmlInvoker.OnTimer := @self.InitiateSimulation;
169     frmSimulation.frmSmlInvoker.Enabled := True;
170
171   end;
172 end;
173
174 procedure TfrmMain.InitiateSimulation(Sender: TObject);
175 var
176   SamplingResult: TArrayOfArrayOfWord;
177 begin
178   SamplingResult := TArrayOfArrayOfWord.Create;
179   case self.cbxAvailableModels.Items[self.cbxAvailableModels.ItemIndex]
      of
180     SIR: begin
181       SIRALG(
```

```
182             StrToInt(frmMain.edtDays.Text),
183             StrToInt(frmMain.edtInitialInfected.Text),
184             StrToFloat(frmMain.edtBeta.Text),
185             StrToFloat(frmMain.edtGamma.Text),
186             StrToInt(frmMain.edtProbabilityOfInfection.Text),
187             SamplingResult
188       );
189       end;
190
191       SIS: begin
192          SISALG(
193             StrToInt(frmMain.edtDays.Text),
194             StrToInt(frmMain.edtInitialInfected.Text),
195             StrToFloat(frmMain.edtBeta.Text),
196             StrToFloat(frmMain.edtGamma.Text),
197             StrToInt(frmMain.edtProbabilityOfInfection.Text),
198             SamplingResult
199       );
200       end;
201     end;
202
203     if self.CancelTriggered then begin
204       self.CancelTriggered := False;
205     end
206     else begin
207       { Prepare Charts }
208       frmMain.preparePreSimulationChart;
209       frmMain.prepareSimulationChart(SamplingResult);
210     end;
211
212     frmSimulation.RestoreNodes; // Restore the Nodes
213
214     { Enable Simulate Button - Disable Cancel Button }
215     self.btnCancel.Enabled := False;
216     self.btnCancel.Visible := False;
217     self.btnSimulate.Enabled := True;
218     self.btnSimulate.Visible := True;
219     { Enable the file close functionality }
220     frmMain.mnuFileClose.Enabled := True;
221 end;
222
223 procedure TfrmMain.FormCreate(Sender: TObject);
224 begin
225     Nodes := TListOfTNode.Create;
226     self.registerAvailableModels;
227 end;
228
229 procedure TfrmMain.RefreshGUI;
230 begin
231     self.Update;
232 end;
233
234 procedure TfrmMain.cbxAvailableModelsChange(Sender: TObject);
235 begin
236     case self.cbxAvailableModels.Items[self.cbxAvailableModels.ItemIndex]
        of
237       SIR: begin
238          self.edtGamma.Enabled := True;
239
240          self.edtMu.Enabled := False;
```

```
        self.edtLambda.Enabled := False;
        self.edtDelta.Enabled := False;
        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtAlpha.Enabled := False;
        self.edtEpsilon.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    SIS: begin
        self.edtGamma.Enabled := True;

        self.edtMu.Enabled := False;
        self.edtLambda.Enabled := False;
        self.edtDelta.Enabled := False;
        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtAlpha.Enabled := False;
        self.edtEpsilon.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    SIQ: begin
        self.edtMu.Enabled := True;
        self.edtLambda.Enabled := True;
        self.edtDelta.Enabled := True;
        self.edtKappa.Enabled := True;

        self.edtGamma.Enabled := False;
        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtAlpha.Enabled := False;
        self.edtEpsilon.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    SIQS, SIQR: begin
        self.edtGamma.Enabled := True;
        self.edtMu.Enabled := True;
        self.edtLambda.Enabled := True;
        self.edtDelta.Enabled := True;
        self.edtDelta1.Enabled := True;
        self.edtKappa.Enabled := True;
        self.edtZeta.Enabled := True;

        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtAlpha.Enabled := False;
        self.edtEpsilon.Enabled := False;
    end;

    SIRD: begin
        self.edtGamma.Enabled := True;
        self.edtMu.Enabled := True;

        self.edtLambda.Enabled := False;
        self.edtDelta.Enabled := False;
        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtAlpha.Enabled := False;
```

```
        self.edtEpsilon.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    MSIR: begin
        self.edtGamma.Enabled := True;
        self.edtMu.Enabled := True;
        self.edtLambda.Enabled := True;
        self.edtDelta.Enabled := True;
        self.edtMaternallyDerivedImmunity.Enabled := True;

        self.edtAlpha.Enabled := False;
        self.edtEpsilon.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    SEIR: begin
        self.edtGamma.Enabled := True;
        self.edtMu.Enabled := True;
        self.edtLambda.Enabled := True;
        self.edtAlpha.Enabled := True;

        self.edtDelta.Enabled := False;
        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtEpsilon.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    SEIS: begin
        self.edtGamma.Enabled := True;
        self.edtMu.Enabled := True;
        self.edtLambda.Enabled := True;
        self.edtEpsilon.Enabled := True;

        self.edtDelta.Enabled := False;
        self.edtMaternallyDerivedImmunity.Enabled := False;
        self.edtAlpha.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
    end;

    MSEIR: begin
        self.edtGamma.Enabled := True;
        self.edtMu.Enabled := True;
        self.edtLambda.Enabled := True;
        self.edtEpsilon.Enabled := True;
        self.edtDelta.Enabled := True;
        self.edtMaternallyDerivedImmunity.Enabled := True;

        self.edtAlpha.Enabled := False;
        self.edtKappa.Enabled := False;
        self.edtDelta1.Enabled := False;
        self.edtZeta.Enabled := False;
```

```
361     end;
362   end;
363
364   self.AvailableSimulationCheck;
365 end;
366
367 procedure TfrmMain.AvailableSimulationCheck;
368 begin
369   if self.Nodes.Count > 0 then begin
370     case self.cbxAvailableModels.Items[self.cbxAvailableModels.
      ItemIndex] of
371       SIR: self.btnSimulate.Enabled := True;
372       SIS: self.btnSimulate.Enabled := True;
373     else
374       self.btnSimulate.Enabled := False;
375     end;
376   end
377   else
378     self.btnSimulate.Enabled := False;
379 end;
380
381 procedure TfrmMain.mnuFileCloseClick(Sender: TObject);
382 begin
383   if frmSimulation.Visible then frmSimulation.Close;
384   if Nodes.Count > 0 then frmSimulation.ResetShapes;
385   self.edtN.Enabled := True;
386   self.mnuFileClose.Enabled := False;
387   self.btnSimulate.Enabled := False;
388 end;
389
390 procedure TfrmMain.mnuSettingsClick(Sender: TObject);
391 begin
392   frmSettings.Show;
393 end;
394
395 procedure TfrmMain.ckbUseSystemSeedChange(Sender: TObject);
396 begin
397   if self.ckbUseSystemSeed.Checked then begin
398     self.edtSeed.Enabled := False;
399   end
400   else self.edtSeed.Enabled := True;
401 end;
402
403 procedure TfrmMain.btnCancelClick(Sender: TObject);
404 begin
405   self.CancelTriggered := True;
406 end;
407
408 procedure TfrmMain.preparePreSimulationChart;
409 begin
410   frmPreSimulationChart.Caption := 'Differential Equations (' + self.
      cbxAvailableModels.Items[self.cbxAvailableModels.ItemIndex] + ')';
411   frmPreSimulationChart.ClearPreSimulationChart;
412   frmPreSimulationChart.CalculatePreSimulation;
413   frmPreSimulationChart.Show;
414 end;
415
416 procedure TfrmMain.prepareSimulationChart(SamplingResult:
      TArrayOfArrayOfWord);
417 begin
```

```
418   frmSimulationChart.Caption := 'Simulation (' + self.
       cbxAvailableModels.Items[self.cbxAvailableModels.ItemIndex] + ')';
419   frmSimulationChart.ClearSimulationChart;
420   frmSimulationChart.CalculateSimulation(SamplingResult);
421   frmSimulationChart.Show;
422 end;
423
424 procedure TfrmMain.registerAvailableModels;
425 var
426   i: Integer;
427 begin
428   for i := Low(AvailableModels) to High(AvailableModels) do
429   self.cbxAvailableModels.Items.Add(AvailableModels[i]);
430
431   self.cbxAvailableModels.ItemIndex := 0;
432   self.cbxAvailableModelsChange(self);
433   self.cbxAvailableModels.Style := csDropDownList;
434 end;
435
436 procedure TfrmMain.edtIntegerKeyPress(Sender: TObject; var Key: char);
437 begin
438   ValidateInteger(Sender, Key);
439 end;
440
441 procedure TfrmMain.edtFloatKeyPress(Sender: TObject; var Key: char);
442 begin
443   ValidateFloat(Sender, Key);
444 end;
445
446 procedure TfrmMain.edtKeyUpEnter(Sender: TObject; var Key: char);
447 begin
448   { Prepare the pre simulation chart if all conditions are met. }
449   { Key #13 represents the Enter key }
450   if (Key = #13) AND self.validateSimulationFields then
451     self.preparePreSimulationChart;
452 end;
453
454 function TfrmMain.getN: Integer;
455 begin
456   Result := 0;
457   if self.edtN.Text <> '' then Result := StrToInt(self.edtN.Text);
458   if self.Nodes.Count > 0 then Result := self.Nodes.Count;
459 end;
460
461 function TfrmMain.validateSimulationFields: Boolean;
462 begin
463   { If all the required TEdits contain a value then return True. }
464   Result := False;
465   if self.getN <> 0.0 then
466   begin
467     case self.cbxAvailableModels.Items[self.cbxAvailableModels.
     ItemIndex] of
468     SIR, SIS: begin
469             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
     AND
470               (self.edtGamma.Text <> '') AND (self.edtInitialInfected.
     Text <> '')
471                 then Result := True;
472           end;
473
```

```
474     SIQ: begin
475             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
476                (self.edtMu.Text <> '') AND (self.edtLambda.Text <> '')
    AND
477                (self.edtDelta.Text <> '') AND (self.edtKappa.Text <> '')
     AND
478                (self.edtInitialInfected.Text <> '')
479                  then Result := True;
480           end;
481
482     SIQS, SIQR: begin
483             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
484                (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
    AND
485                (self.edtLambda.Text <> '') AND (self.edtDelta.Text <> ''
    ) AND
486                (self.edtDelta1.Text <> '') AND (self.edtKappa.Text <> ''
    ) AND
487                (self.edtZeta.Text <> '') AND (self.edtInitialInfected.
    Text <> '')
488                  then Result := True;
489           end;
490
491     SIRD: begin
492             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
493                (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
494                AND (self.edtInitialInfected.Text <> '')
495                  then Result := True;
496           end;
497
498     MSIR: begin
499             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
500                (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
    AND
501                (self.edtLambda.Text <> '') AND (self.edtDelta.Text <> ''
    ) AND
502                (self.edtMaternallyDerivedImmunity.Text <> '') AND
503                (self.edtInitialInfected.Text <> '')
504                  then Result := True;
505           end;
506
507     SEIR: begin
508             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
509                (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
    AND
510                (self.edtLambda.Text <> '') AND (self.edtAlpha.Text <> ''
    ) AND
511                (self.edtInitialInfected.Text <> '')
512                  then Result := True;
513           end;
514
515     SEIS: begin
516             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
517                (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
```

```
    AND
518             (self.edtLambda.Text <> '') AND (self.edtEpsilon.Text <>
    '') AND
519             (self.edtInitialInfected.Text <> '')
520               then Result := True;
521         end;
522
523     MSEIR: begin
524             if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
    AND
525             (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
    AND
526             (self.edtLambda.Text <> '') AND (self.edtEpsilon.Text <>
    '') AND
527             (self.edtDelta.Text <> '') AND
528             (self.edtMaternallyDerivedImmunity.Text <> '') AND
529             (self.edtInitialInfected.Text <> '')
530               then Result := True;
531         end;
532
533     end;
534   end;
535 end;
536
537 end.
```

A.6.2 Γραφική παράσταση διαφορικών εξισώσεων

```
1  unit frmPreSimulationChart_u;
2
3  {$mode objfpc}{$H+}
4
5  interface
6
7  uses
8    Classes, SysUtils, Forms, Controls, Graphics, Dialogs, TAGraph,
     TASeries,
9    { Forms }
10   frmMain_u,
11   { Classes }
12   { Utilities }
13   utlArray_u,
14   utlTypes_u,
15   utlEuler_u,
16   utlEnum_u,
17   utlConstants_u;
18
19 type
20
21   { TfrmPreSimulationChart }
22
23   TfrmPreSimulationChart = class(TForm)
24     chtPreSimulation: TChart;
25     chtPreSimulationI: TLineSeries;
26     chtPreSimulationD: TLineSeries;
27     chtPreSimulationE: TLineSeries;
28     chtPreSimulationQ: TLineSeries;
29     chtPreSimulationM: TLineSeries;
30     chtPreSimulationR: TLineSeries;
```

```
    chtPreSimulationS: TLineSeries;
    procedure CalculatePreSimulation;
    procedure FormClose(Sender: TObject);
    procedure ClearPreSimulationChart;
    procedure PrepareSIR(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSIS(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSIQ(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSIQS(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSIQR(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSIRD(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareMSIR(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSEIR(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareSEIS(var y0, extraArgs: ArrayOfDouble);
    procedure PrepareMSEIR(var y0, extraArgs: ArrayOfDouble);
  private

  public

  end;

var
  frmPreSimulationChart: TfrmPreSimulationChart;

implementation

procedure TfrmPreSimulationChart.CalculatePreSimulation;
var
  OdeEulerResult: ArrayOfArrayOfDouble;
  t, y0, extraArgs: ArrayOfDouble;
  i, days: Integer;
  model: String;
begin
  SetLength(y0, 0);
  SetLength(extraArgs, 0);

  days := StrToInt(frmMain.edtDays.Text);
  SetLength(t, days);
  t := linspace(0, days, days);

  model := frmMain.cbxAvailableModels.Items[frmMain.cbxAvailableModels.
   ItemIndex];
  { Initiate based on the model }
  case model of
    SIR: self.PrepareSIR(y0, extraArgs);
    SIS: self.PrepareSIS(y0, extraArgs);
    SIQ: self.PrepareSIQ(y0, extraArgs);
    SIQS: self.PrepareSIQS(y0, extraArgs);
    SIQR: self.PrepareSIQR(y0, extraArgs);
    SIRD: self.PrepareSIRD(y0, extraArgs);
    MSIR: self.PrepareMSIR(y0, extraArgs);
    SEIR: self.PrepareSEIR(y0, extraArgs);
    SEIS: self.PrepareSEIS(y0, extraArgs);
    MSEIR: self.PrepareMSEIR(y0, extraArgs);
  end;

  { Apply Euler to the model's differential equations }
  OdeEulerResult := odeEuler(model, t, y0, extraArgs);

  { Fill data to the appropriate axes }
  case model of
```

```
90      SIR: begin
91        for i := 0 to days - 1 do
92          begin
93            self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
94            self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
    I
95            self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[2][i]); //
    R
96          end;
97        end;
98
99      SIS: begin
100       for i := 0 to days - 1 do
101         begin
102           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
103           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
    I
104         end;
105       end;
106
107     SIQ, SIQS: begin
108       for i := 0 to days - 1 do
109         begin
110           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
111           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
    I
112           self.chtPreSimulationQ.AddXY(t[i], OdeEulerResult[2][i]); //
    Q
113         end;
114       end;
115
116     SIQR: begin
117       for i := 0 to days - 1 do
118         begin
119           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
120           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
    I
121           self.chtPreSimulationQ.AddXY(t[i], OdeEulerResult[2][i]); //
    Q
122           self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[3][i]); //
    R
123         end;
124       end;
125
126     SIRD: begin
127       for i := 0 to days - 1 do
128         begin
129           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
130           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
    I
131           self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[2][i]); //
    R
132           self.chtPreSimulationD.AddXY(t[i], OdeEulerResult[3][i]); //
    D
133         end;
```

```
134       end;
135
136     MSIR: begin
137       for i := 0 to days - 1 do
138         begin
139           self.chtPreSimulationM.AddXY(t[i], OdeEulerResult[0][i]); //
    M
140           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[1][i]); //
    S
141           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[2][i]); //
    I
142           self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[3][i]); //
    R
143         end;
144       end;
145
146     SEIR: begin
147       for i := 0 to days - 1 do
148         begin
149           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
150           self.chtPreSimulationE.AddXY(t[i], OdeEulerResult[1][i]); //
    E
151           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[2][i]); //
    I
152           self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[3][i]); //
    R
153         end;
154       end;
155
156     SEIS: begin
157       for i := 0 to days - 1 do
158         begin
159           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[0][i]); //
    S
160           self.chtPreSimulationE.AddXY(t[i], OdeEulerResult[1][i]); //
    E
161           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[2][i]); //
    I
162         end;
163       end;
164
165     MSEIR: begin
166       for i := 0 to days - 1 do
167         begin
168           self.chtPreSimulationM.AddXY(t[i], OdeEulerResult[0][i]); //
    M
169           self.chtPreSimulationS.AddXY(t[i], OdeEulerResult[1][i]); //
    S
170           self.chtPreSimulationE.AddXY(t[i], OdeEulerResult[2][i]); //
    E
171           self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[3][i]); //
    I
172           self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[4][i]); //
    R
173         end;
174       end;
175
176     end;
177
```

```
178   self.chtPreSimulation.Visible := true;
179 end;
180
181 procedure TfrmPreSimulationChart.ClearPreSimulationChart;
182 var
183   i: Integer;
184 begin
185   for i := 0 to self.chtPreSimulation.SeriesCount - 1 do
186   begin
187     if self.chtPreSimulation.Series[i] is TLineSeries then
188       begin
189         (self.chtPreSimulation.Series[i] as TLineSeries).Clear;
190         (self.chtPreSimulation.Series[i] as TLineSeries).Active :=
    False;
191       end;
192   end;
193 end;
194
195 procedure TfrmPreSimulationChart.PrepareSIR(var y0, extraArgs:
    ArrayOfDouble);
196 begin
197   SetLength(y0, 3);
198   y0[1] := StrToFloat(frmMain.edtInitialInfected.Text);       // I
199   y0[2] := 0;                                                 // R
200   y0[0] := frmMain.getN - y0[1] - y0[2];                      // S
201
202   SetLength(extraArgs, 3);
203   extraArgs[0] := frmMain.getN;                               // N
204   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);           // Beta
205   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);          // Gamma
206
207   { Arrange Line Series Color }
208   self.chtPreSimulationS.SeriesColor := clNavy;
209   self.chtPreSimulationI.SeriesColor := clMaroon;
210   self.chtPreSimulationR.SeriesColor := clGreen;
211
212   { Define Line Series Titles }
213   self.chtPreSimulationS.Title := 'Susceptible';
214   self.chtPreSimulationI.Title := 'Infected';
215   self.chtPreSimulationR.Title := 'Recovered';
216
217   { Activate Line Series }
218   self.chtPreSimulationS.Active := True;
219   self.chtPreSimulationI.Active := True;
220   self.chtPreSimulationR.Active := True;
221 end;
222
223 procedure TfrmPreSimulationChart.PrepareSIS(var y0, extraArgs:
    ArrayOfDouble);
224 begin
225   SetLength(y0, 2);
226   y0[1] := StrToFloat(frmMain.edtInitialInfected.Text);       // I
227   y0[0] := frmMain.getN - y0[1];                              // S
228
229   SetLength(extraArgs, 3);
230   extraArgs[0] := frmMain.getN;                               // N
231   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);           // Beta
232   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);          // Gamma
233
234   { Arrange Line Series Color }
```

```
235   self.chtPreSimulationS.SeriesColor := clNavy;
236   self.chtPreSimulationI.SeriesColor := clMaroon;
237
238   { Define Line Series Titles }
239   self.chtPreSimulationS.Title := 'Susceptible';
240   self.chtPreSimulationI.Title := 'Infected';
241
242   { Activate Line Series }
243   self.chtPreSimulationS.Active := True;
244   self.chtPreSimulationI.Active := True;
245 end;
246
247 procedure TfrmPreSimulationChart.PrepareSIQ(var y0, extraArgs:
        ArrayOfDouble);
248 begin
249   SetLength(y0, 3);
250   y0[1] := StrToFloat(frmMain.edtInitialInfected.Text);        // I
251   y0[2] := 0;                                                  // Q
252   y0[0] := frmMain.getN - y0[1] - y0[2];                       // S
253
254   SetLength(extraArgs, 6);
255   extraArgs[0] := frmMain.getN;                                // N
256   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);            // Beta
257   extraArgs[2] := StrToFloat(frmMain.edtMu.Text);              // Mu
258   extraArgs[3] := StrToFloat(frmMain.edtLambda.Text);          // Lambda
259   extraArgs[4] := StrToFloat(frmMain.edtDelta.Text);           // Delta
260   extraArgs[5] := StrToFloat(frmMain.edtKappa.Text);           // Kappa
261
262   { Arrange Line Series Color }
263   self.chtPreSimulationS.SeriesColor := clNavy;
264   self.chtPreSimulationI.SeriesColor := clMaroon;
265   self.chtPreSimulationQ.SeriesColor := TColor($F59D81);
266
267   { Define Line Series Titles }
268   self.chtPreSimulationS.Title := 'Susceptible';
269   self.chtPreSimulationI.Title := 'Infected';
270   self.chtPreSimulationQ.Title := 'Quarantined';
271
272   { Activate Line Series }
273   self.chtPreSimulationS.Active := True;
274   self.chtPreSimulationI.Active := True;
275   self.chtPreSimulationQ.Active := True;
276 end;
277
278 procedure TfrmPreSimulationChart.PrepareSIQS(var y0, extraArgs:
        ArrayOfDouble);
279 begin
280   SetLength(y0, 3);
281   y0[1] := StrToFloat(frmMain.edtInitialInfected.Text);        // I
282   y0[2] := 0;                                                  // Q
283   y0[0] := frmMain.getN - y0[1] - y0[2];                       // S
284
285   SetLength(extraArgs, 9);
286   extraArgs[0] := frmMain.getN;                                // N
287   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);            // Beta
288   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);           // Gamma
289   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);              // Mu
290   extraArgs[4] := StrToFloat(frmMain.edtLambda.Text);          // Lambda
291   extraArgs[5] := StrToFloat(frmMain.edtDelta.Text);           // Delta
292   extraArgs[6] := StrToFloat(frmMain.edtDelta1.Text);          // Delta1
```

```
293   extraArgs[7] := StrToFloat(frmMain.edtKappa.Text);          // Kappa
294   extraArgs[8] := StrToFloat(frmMain.edtZeta.Text);           // Zeta
295
296   { Arrange Line Series Color }
297   self.chtPreSimulationS.SeriesColor := clNavy;
298   self.chtPreSimulationI.SeriesColor := clMaroon;
299   self.chtPreSimulationQ.SeriesColor := TColor($F59D81);
300
301   { Define Line Series Titles }
302   self.chtPreSimulationS.Title := 'Susceptible';
303   self.chtPreSimulationI.Title := 'Infected';
304   self.chtPreSimulationQ.Title := 'Quarantined';
305
306   { Activate Line Series }
307   self.chtPreSimulationS.Active := True;
308   self.chtPreSimulationI.Active := True;
309   self.chtPreSimulationQ.Active := True;
310 end;
311
312 procedure TfrmPreSimulationChart.PrepareSIQR(var y0, extraArgs:
      ArrayOfDouble);
313 begin
314   SetLength(y0, 4);
315   y0[1] := StrToFloat(frmMain.edtInitialInfected.Text);       // I
316   y0[2] := 0;                                                 // Q
317   y0[0] := frmMain.getN - y0[1] - y0[2];                      // S
318   y0[3] := 0;                                                 // R
319
320   SetLength(extraArgs, 9);
321   extraArgs[0] := frmMain.getN;                               // N
322   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);           // Beta
323   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);          // Gamma
324   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);             // Mu
325   extraArgs[4] := StrToFloat(frmMain.edtLambda.Text);         // Lambda
326   extraArgs[5] := StrToFloat(frmMain.edtDelta.Text);          // Delta
327   extraArgs[6] := StrToFloat(frmMain.edtDelta1.Text);         // Delta1
328   extraArgs[7] := StrToFloat(frmMain.edtKappa.Text);          // Kappa
329   extraArgs[8] := StrToFloat(frmMain.edtZeta.Text);           // Zeta
330
331   { Arrange Line Series Color }
332   self.chtPreSimulationS.SeriesColor := clNavy;
333   self.chtPreSimulationI.SeriesColor := clMaroon;
334   self.chtPreSimulationQ.SeriesColor := TColor($F59D81);
335   self.chtPreSimulationR.SeriesColor := clGreen;
336
337   { Define Line Series Titles }
338   self.chtPreSimulationS.Title := 'Susceptible';
339   self.chtPreSimulationI.Title := 'Infected';
340   self.chtPreSimulationQ.Title := 'Quarantined';
341   self.chtPreSimulationR.Title := 'Recovered';
342
343   { Activate Line Series }
344   self.chtPreSimulationS.Active := True;
345   self.chtPreSimulationI.Active := True;
346   self.chtPreSimulationQ.Active := True;
347   self.chtPreSimulationR.Active := True;
348 end;
349
350 procedure TfrmPreSimulationChart.PrepareSIRD(var y0, extraArgs:
      ArrayOfDouble);
```

```
351 begin
352   SetLength(y0, 4);
353   y0[1] := StrToFloat(frmMain.edtInitialInfected.Text);        // I
354   y0[2] := 0;                                                  // R
355   y0[0] := frmMain.getN - y0[1] - y0[2];                       // S
356   y0[3] := 0;                                                  // D
357
358   SetLength(extraArgs, 4);
359   extraArgs[0] := frmMain.getN;                                // N
360   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);            // Beta
361   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);           // Gamma
362   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);              // Mu
363
364   { Arrange Line Series Color }
365   self.chtPreSimulationS.SeriesColor := clNavy;
366   self.chtPreSimulationI.SeriesColor := clMaroon;
367   self.chtPreSimulationR.SeriesColor := clGreen;
368   self.chtPreSimulationD.SeriesColor := clDkGray;
369
370   { Define Line Series Titles }
371   self.chtPreSimulationS.Title := 'Susceptible';
372   self.chtPreSimulationI.Title := 'Infected';
373   self.chtPreSimulationR.Title := 'Recovered';
374   self.chtPreSimulationD.Title := 'Deceased';
375
376   { Activate Line Series }
377   self.chtPreSimulationS.Active := True;
378   self.chtPreSimulationI.Active := True;
379   self.chtPreSimulationR.Active := True;
380   self.chtPreSimulationD.Active := True;
381 end;
382
383 procedure TfrmPreSimulationChart.PrepareMSIR(var y0, extraArgs:
      ArrayOfDouble);
384 begin
385   SetLength(y0, 4);
386   y0[0] := StrToFloat(frmMain.edtMaternallyDerivedImmunity.Text);  // M
387   y0[2] := StrToFloat(frmMain.edtInitialInfected.Text);            // I
388   y0[3] := 0;                                                      // R
389   { Susceptible (y0[1]) = N - Maternally derived immunity - Infected -
       Recovered }
390   y0[1] := frmMain.getN - y0[0] - y0[2] - y0[3];                   // S
391
392   SetLength(extraArgs, 6);
393   extraArgs[0] := frmMain.getN;                                // N
394   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);            // Beta
395   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);           // Gamma
396   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);              // Mu
397   extraArgs[4] := StrToFloat(frmMain.edtLambda.Text);          // Lambda
398   extraArgs[5] := StrToFloat(frmMain.edtDelta.Text);           // Delta
399
400   { Arrange Line Series Color }
401   self.chtPreSimulationM.SeriesColor := clFuchsia;
402   self.chtPreSimulationS.SeriesColor := clNavy;
403   self.chtPreSimulationI.SeriesColor := clMaroon;
404   self.chtPreSimulationR.SeriesColor := clGreen;
405
406   { Define Line Series Titles }
407   self.chtPreSimulationM.Title := 'Maternally' + sLineBreak + 'derived'
        + sLineBreak + 'immunity';
```

```
408   self.chtPreSimulationS.Title := ' ' + sLineBreak + 'Susceptible';
409   self.chtPreSimulationI.Title := ' ' + sLineBreak + 'Infected';
410   self.chtPreSimulationR.Title := ' ' + sLineBreak + 'Recovered';
411
412   { Activate Line Series }
413   self.chtPreSimulationM.Active := True;
414   self.chtPreSimulationS.Active := True;
415   self.chtPreSimulationI.Active := True;
416   self.chtPreSimulationR.Active := True;
417 end;
418
419 procedure TfrmPreSimulationChart.PrepareSEIR(var y0, extraArgs:
      ArrayOfDouble);
420 begin
421   SetLength(y0, 4);
422   y0[3] := 0;                                                    // R
423   y0[2] := StrToFloat(frmMain.edtInitialInfected.Text);          // I
424   y0[1] := 0;                                                    // E
425   y0[0] := frmMain.getN - y0[1] - y0[2] - y0[3];                 // S
426
427   SetLength(extraArgs, 6);
428   extraArgs[0] := frmMain.getN;                                  // N
429   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);              // Beta
430   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);             // Gamma
431   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);                // Mu
432   extraArgs[4] := StrToFloat(frmMain.edtLambda.Text);            // Lambda
433   extraArgs[5] := StrToFloat(frmMain.edtAlpha.Text);             // Alpha
434
435   { Arrange Line Series Color }
436   self.chtPreSimulationS.SeriesColor := clNavy;
437   self.chtPreSimulationE.SeriesColor := clOlive;
438   self.chtPreSimulationI.SeriesColor := clMaroon;
439   self.chtPreSimulationR.SeriesColor := clGreen;
440
441   { Define Line Series Titles }
442   self.chtPreSimulationS.Title := 'Susceptible';
443   self.chtPreSimulationE.Title := 'Exposed';
444   self.chtPreSimulationI.Title := 'Infected';
445   self.chtPreSimulationR.Title := 'Recovered';
446
447   { Activate Line Series }
448   self.chtPreSimulationS.Active := True;
449   self.chtPreSimulationE.Active := True;
450   self.chtPreSimulationI.Active := True;
451   self.chtPreSimulationR.Active := True;
452 end;
453
454 procedure TfrmPreSimulationChart.PrepareSEIS(var y0, extraArgs:
      ArrayOfDouble);
455 begin
456   SetLength(y0, 3);
457   y0[2] := StrToFloat(frmMain.edtInitialInfected.Text);          // I
458   y0[1] := 0;                                                    // E
459   y0[0] := frmMain.getN - y0[1] - y0[2];                         // S
460
461   SetLength(extraArgs, 6);
462   extraArgs[0] := frmMain.getN;                                  // N
463   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);              // Beta
464   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);             // Gamma
465   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);                // Mu
```

```
466   extraArgs[4] := StrToFloat(frmMain.edtLambda.Text);          // Lambda
467   extraArgs[5] := StrToFloat(frmMain.edtEpsilon.Text);         //
       Epsilon
468
469   { Arrange Line Series Color }
470   self.chtPreSimulationS.SeriesColor := clNavy;
471   self.chtPreSimulationE.SeriesColor := clOlive;
472   self.chtPreSimulationI.SeriesColor := clMaroon;
473
474   { Define Line Series Titles }
475   self.chtPreSimulationS.Title := 'Susceptible';
476   self.chtPreSimulationE.Title := 'Exposed';
477   self.chtPreSimulationI.Title := 'Infected';
478
479   { Activate Line Series }
480   self.chtPreSimulationS.Active := True;
481   self.chtPreSimulationE.Active := True;
482   self.chtPreSimulationI.Active := True;
483 end;
484
485 procedure TfrmPreSimulationChart.PrepareMSEIR(var y0, extraArgs:
       ArrayOfDouble);
486 begin
487   SetLength(y0, 5);
488   y0[0] := StrToFloat(frmMain.edtMaternallyDerivedImmunity.Text);  // M
489   y0[4] := 0;                                                      // R
490   y0[3] := StrToFloat(frmMain.edtInitialInfected.Text);            // I
491   y0[2] := 0;                                                      // E
492   { Susceptible (y0[1]) = N - Maternally derived immunity - Exposed -
       Infected - Recovered }
493   y0[1] := frmMain.getN - y0[0] - y0[2] - y0[3] - y0[4];           // S
494
495   SetLength(extraArgs, 7);
496   extraArgs[0] := frmMain.getN;                                    // N
497   extraArgs[1] := StrToFloat(frmMain.edtBeta.Text);                // Beta
498   extraArgs[2] := StrToFloat(frmMain.edtGamma.Text);               // Gamma
499   extraArgs[3] := StrToFloat(frmMain.edtMu.Text);                  // Mu
500   extraArgs[4] := StrToFloat(frmMain.edtLambda.Text);              // Lambda
501   extraArgs[5] := StrToFloat(frmMain.edtDelta.Text);               // Delta
502   extraArgs[6] := StrToFloat(frmMain.edtEpsilon.Text);             //
       Epsilon
503
504   { Arrange Line Series Color }
505   self.chtPreSimulationM.SeriesColor := clFuchsia;
506   self.chtPreSimulationS.SeriesColor := clNavy;
507   self.chtPreSimulationE.SeriesColor := clOlive;
508   self.chtPreSimulationI.SeriesColor := clMaroon;
509   self.chtPreSimulationR.SeriesColor := clGreen;
510
511   { Define Line Series Titles }
512   self.chtPreSimulationM.Title := 'Maternally' + sLineBreak + 'derived'
       + sLineBreak + 'immunity';
513   self.chtPreSimulationS.Title := ' ' + sLineBreak + 'Susceptible';
514   self.chtPreSimulationE.Title := ' ' + sLineBreak + 'Exposed';
515   self.chtPreSimulationI.Title := ' ' + sLineBreak + 'Infected';
516   self.chtPreSimulationR.Title := ' ' + sLineBreak + 'Recovered';
517
518   { Activate Line Series }
519   self.chtPreSimulationM.Active := True;
520   self.chtPreSimulationS.Active := True;
```

```
  self.chtPreSimulationE.Active := True;
  self.chtPreSimulationI.Active := True;
  self.chtPreSimulationR.Active := True;
end;

procedure TfrmPreSimulationChart.FormClose(Sender: TObject);
begin
  self.ClearPreSimulationChart;
end;

{$R *.lfm}

end.
```

A.6.3 Γραφική παράσταση προσομοίωσης

```
unit frmSimulationChart_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, TAGraph,
   TASeries,
  { Forms }
  frmMain_u,
  { Classes }
  { Utilities }
  utlArray_u,
  utlTypes_u,
  utlEnum_u,
  utlConstants_u;

type

  { TfrmSimulationChart }

  TfrmSimulationChart = class(TForm)
    chtSimulation: TChart;
    chtSimulationI: TLineSeries;
    chtSimulationD: TLineSeries;
    chtSimulationE: TLineSeries;
    chtSimulationQ: TLineSeries;
    chtSimulationM: TLineSeries;
    chtSimulationR: TLineSeries;
    chtSimulationS: TLineSeries;
    procedure CalculateSimulation(SamplingResult: TArrayOfArrayOfWord);
    procedure FormClose(Sender: TObject);
    procedure ClearSimulationChart;
    procedure FormResize(Sender: TObject);
    procedure PrepareSIR;
    procedure PrepareSIS;
    procedure PrepareSIQ;
    procedure PrepareSIQS;
    procedure PrepareSIQR;
    procedure PrepareSIRD;
    procedure PrepareMSIR;
    procedure PrepareSEIR;
```

```
43      procedure PrepareSEIS;
44      procedure PrepareMSEIR;
45    private
46
47    public
48
49    end;
50
51  var
52    frmSimulationChart: TfrmSimulationChart;
53
54  implementation
55
56  procedure TfrmSimulationChart.CalculateSimulation(SamplingResult:
        TArrayOfArrayOfWord);
57  var
58    t: ArrayOfDouble;
59    i, days: Integer;
60    model: String;
61  begin
62    days := StrToInt(frmMain.edtDays.Text) + 2;
63    SetLength(t, days);
64    t := linspace(0, days, days);
65
66    model := frmMain.cbxAvailableModels.Items[frmMain.cbxAvailableModels.
        ItemIndex];
67    { Initiate based on the model }
68    case model of
69      SIR: self.PrepareSIR;
70      SIS: self.PrepareSIS;
71      SIQ: self.PrepareSIQ;
72      SIQS: self.PrepareSIQS;
73      SIQR: self.PrepareSIQR;
74      SIRD: self.PrepareSIRD;
75      MSIR: self.PrepareMSIR;
76      SEIR: self.PrepareSEIR;
77      SEIS: self.PrepareSEIS;
78      MSEIR: self.PrepareMSEIR;
79    end;
80
81    { Fill data to the appropriate axes }
82    case model of
83      SIR: begin
84        for i := 0 to days - 1 do
85          begin
86            self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
87            self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
88            self.chtSimulationR.AddXY(t[i], SamplingResult[i][2]); // R
89          end;
90        end;
91
92      SIS: begin
93        for i := 0 to days - 1 do
94          begin
95            self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
96            self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
97          end;
98        end;
99
100     SIQ, SIQS: begin
```

```
      for i := 0 to days - 1 do
        begin
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
          self.chtSimulationQ.AddXY(t[i], SamplingResult[i][2]); // Q
        end;
      end;

    SIQR: begin
      for i := 0 to days - 1 do
        begin
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
          self.chtSimulationQ.AddXY(t[i], SamplingResult[i][2]); // Q
          self.chtSimulationR.AddXY(t[i], SamplingResult[i][3]); // R
        end;
      end;

    SIRD: begin
      for i := 0 to days - 1 do
        begin
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
          self.chtSimulationR.AddXY(t[i], SamplingResult[i][2]); // R
          self.chtSimulationD.AddXY(t[i], SamplingResult[i][3]); // D
        end;
      end;

    MSIR: begin
      for i := 0 to days - 1 do
        begin
          self.chtSimulationM.AddXY(t[i], SamplingResult[i][0]); // M
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][1]); // S
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][2]); // I
          self.chtSimulationR.AddXY(t[i], SamplingResult[i][3]); // R
        end;
      end;

    SEIR: begin
      for i := 0 to days - 1 do
        begin
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
          self.chtSimulationE.AddXY(t[i], SamplingResult[i][1]); // E
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][2]); // I
          self.chtSimulationR.AddXY(t[i], SamplingResult[i][3]); // R
        end;
      end;

    SEIS: begin
      for i := 0 to days - 1 do
        begin
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
          self.chtSimulationE.AddXY(t[i], SamplingResult[i][1]); // E
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][2]); // I
        end;
      end;

    MSEIR: begin
      for i := 0 to days - 1 do
        begin
```

```
          self.chtSimulationM.AddXY(t[i], SamplingResult[i][0]); // M
          self.chtSimulationS.AddXY(t[i], SamplingResult[i][1]); // S
          self.chtSimulationE.AddXY(t[i], SamplingResult[i][2]); // E
          self.chtSimulationI.AddXY(t[i], SamplingResult[i][3]); // I
          self.chtSimulationR.AddXY(t[i], SamplingResult[i][4]); // R
        end;
      end;

    end;

  self.chtSimulation.Visible := true;
end;

procedure TfrmSimulationChart.ClearSimulationChart;
var
  i: Integer;
begin
  for i := 0 to self.chtSimulation.SeriesCount - 1 do
  begin
    if self.chtSimulation.Series[i] is TLineSeries then
      begin
        (self.chtSimulation.Series[i] as TLineSeries).Clear;
        (self.chtSimulation.Series[i] as TLineSeries).Active := False;
      end;
  end;
end;

procedure TfrmSimulationChart.FormResize(Sender: TObject);
begin
  self.chtSimulation.Height := self.Height * 97 div 100;
  self.chtSimulation.Width := self.Width * 99 div 100;
end;

procedure TfrmSimulationChart.PrepareSIR;
begin
  { Arrange Line Series Color }
  self.chtSimulationS.SeriesColor := clNavy;
  self.chtSimulationI.SeriesColor := clMaroon;
  self.chtSimulationR.SeriesColor := clGreen;

  { Define Line Series Titles }
  self.chtSimulationS.Title := 'Susceptible';
  self.chtSimulationI.Title := 'Infected';
  self.chtSimulationR.Title := 'Recovered';

  { Activate Line Series }
  self.chtSimulationS.Active := True;
  self.chtSimulationI.Active := True;
  self.chtSimulationR.Active := True;
end;

procedure TfrmSimulationChart.PrepareSIS;
begin
  { Arrange Line Series Color }
  self.chtSimulationS.SeriesColor := clNavy;
  self.chtSimulationI.SeriesColor := clMaroon;

  { Define Line Series Titles }
  self.chtSimulationS.Title := 'Susceptible';
  self.chtSimulationI.Title := 'Infected';
```

```
221
222   { Activate Line Series }
223   self.chtSimulationS.Active := True;
224   self.chtSimulationI.Active := True;
225 end;
226
227 procedure TfrmSimulationChart.PrepareSIQ;
228 begin
229   { Arrange Line Series Color }
230   self.chtSimulationS.SeriesColor := clNavy;
231   self.chtSimulationI.SeriesColor := clMaroon;
232   self.chtSimulationQ.SeriesColor := TColor($F59D81);
233
234   { Define Line Series Titles }
235   self.chtSimulationS.Title := 'Susceptible';
236   self.chtSimulationI.Title := 'Infected';
237   self.chtSimulationQ.Title := 'Quarantined';
238
239   { Activate Line Series }
240   self.chtSimulationS.Active := True;
241   self.chtSimulationI.Active := True;
242   self.chtSimulationQ.Active := True;
243 end;
244
245 procedure TfrmSimulationChart.PrepareSIQS;
246 begin
247   { Arrange Line Series Color }
248   self.chtSimulationS.SeriesColor := clNavy;
249   self.chtSimulationI.SeriesColor := clMaroon;
250   self.chtSimulationQ.SeriesColor := TColor($F59D81);
251
252   { Define Line Series Titles }
253   self.chtSimulationS.Title := 'Susceptible';
254   self.chtSimulationI.Title := 'Infected';
255   self.chtSimulationQ.Title := 'Quarantined';
256
257   { Activate Line Series }
258   self.chtSimulationS.Active := True;
259   self.chtSimulationI.Active := True;
260   self.chtSimulationQ.Active := True;
261 end;
262
263 procedure TfrmSimulationChart.PrepareSIQR;
264 begin
265   { Arrange Line Series Color }
266   self.chtSimulationS.SeriesColor := clNavy;
267   self.chtSimulationI.SeriesColor := clMaroon;
268   self.chtSimulationQ.SeriesColor := TColor($F59D81);
269   self.chtSimulationR.SeriesColor := clGreen;
270
271   { Define Line Series Titles }
272   self.chtSimulationS.Title := 'Susceptible';
273   self.chtSimulationI.Title := 'Infected';
274   self.chtSimulationQ.Title := 'Quarantined';
275   self.chtSimulationR.Title := 'Recovered';
276
277   { Activate Line Series }
278   self.chtSimulationS.Active := True;
279   self.chtSimulationI.Active := True;
280   self.chtSimulationQ.Active := True;
```

```
  self.chtSimulationR.Active := True;
end;

procedure TfrmSimulationChart.PrepareSIRD;
begin
  { Arrange Line Series Color }
  self.chtSimulationS.SeriesColor := clNavy;
  self.chtSimulationI.SeriesColor := clOlive;
  self.chtSimulationR.SeriesColor := clGreen;
  self.chtSimulationD.SeriesColor := clMaroon;

  { Define Line Series Titles }
  self.chtSimulationS.Title := 'Susceptible';
  self.chtSimulationI.Title := 'Infected';
  self.chtSimulationR.Title := 'Recovered';
  self.chtSimulationD.Title := 'Deceased';

  { Activate Line Series }
  self.chtSimulationS.Active := True;
  self.chtSimulationI.Active := True;
  self.chtSimulationR.Active := True;
  self.chtSimulationD.Active := True;
end;

procedure TfrmSimulationChart.PrepareMSIR;
begin
  { Arrange Line Series Color }
  self.chtSimulationM.SeriesColor := clFuchsia;
  self.chtSimulationS.SeriesColor := clNavy;
  self.chtSimulationI.SeriesColor := clOlive;
  self.chtSimulationR.SeriesColor := clGreen;

  { Define Line Series Titles }
  self.chtSimulationM.Title := 'Maternally' + sLineBreak + 'derived' +
   sLineBreak + 'immunity';
  self.chtSimulationS.Title := ' ' + sLineBreak + 'Susceptible';
  self.chtSimulationI.Title := ' ' + sLineBreak + 'Infected';
  self.chtSimulationR.Title := ' ' + sLineBreak + 'Recovered';

  { Activate Line Series }
  self.chtSimulationM.Active := True;
  self.chtSimulationS.Active := True;
  self.chtSimulationI.Active := True;
  self.chtSimulationR.Active := True;
end;

procedure TfrmSimulationChart.PrepareSEIR;
begin
  { Arrange Line Series Color }
  self.chtSimulationS.SeriesColor := clNavy;
  self.chtSimulationE.SeriesColor := clOlive;
  self.chtSimulationI.SeriesColor := clMaroon;
  self.chtSimulationR.SeriesColor := clGreen;

  { Define Line Series Titles }
  self.chtSimulationS.Title := 'Susceptible';
  self.chtSimulationE.Title := 'Exposed';
  self.chtSimulationI.Title := 'Infected';
  self.chtSimulationR.Title := 'Recovered';

```

```
340   { Activate Line Series }
341   self.chtSimulationS.Active := True;
342   self.chtSimulationE.Active := True;
343   self.chtSimulationI.Active := True;
344   self.chtSimulationR.Active := True;
345 end;
346
347 procedure TfrmSimulationChart.PrepareSEIS;
348 begin
349   { Arrange Line Series Color }
350   self.chtSimulationS.SeriesColor := clNavy;
351   self.chtSimulationE.SeriesColor := clOlive;
352   self.chtSimulationI.SeriesColor := clMaroon;
353
354   { Define Line Series Titles }
355   self.chtSimulationS.Title := 'Susceptible';
356   self.chtSimulationE.Title := 'Exposed';
357   self.chtSimulationI.Title := 'Infected';
358
359   { Activate Line Series }
360   self.chtSimulationS.Active := True;
361   self.chtSimulationE.Active := True;
362   self.chtSimulationI.Active := True;
363 end;
364
365 procedure TfrmSimulationChart.PrepareMSEIR;
366 begin
367   { Arrange Line Series Color }
368   self.chtSimulationM.SeriesColor := clFuchsia;
369   self.chtSimulationS.SeriesColor := clNavy;
370   self.chtSimulationE.SeriesColor := clOlive;
371   self.chtSimulationI.SeriesColor := clMaroon;
372   self.chtSimulationR.SeriesColor := clGreen;
373
374   { Define Line Series Titles }
375   self.chtSimulationM.Title := 'Maternally' + sLineBreak + 'derived' +
      sLineBreak + 'immunity';
376   self.chtSimulationS.Title := ' ' + sLineBreak + 'Susceptible';
377   self.chtSimulationE.Title := ' ' + sLineBreak + 'Exposed';
378   self.chtSimulationI.Title := ' ' + sLineBreak + 'Infected';
379   self.chtSimulationR.Title := ' ' + sLineBreak + 'Recovered';
380
381   { Activate Line Series }
382   self.chtSimulationM.Active := True;
383   self.chtSimulationS.Active := True;
384   self.chtSimulationE.Active := True;
385   self.chtSimulationI.Active := True;
386   self.chtSimulationR.Active := True;
387 end;
388
389 procedure TfrmSimulationChart.FormClose(Sender: TObject);
390 begin
391   self.ClearSimulationChart;
392 end;
393
394 {$R *.lfm}
395
396 end.
```

A.6.4 Ρυθμίσεις εφαρμογής

```
unit frmSettings_u;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
   fpjson,
  { Utilities }
  utlFile_u,
  utlConstants_u,
  utlValidation_u;

type

  { TfrmSettings }

  TfrmSettings = class(TForm)
    btnCancel: TButton;
    btnApply: TButton;
    cbxReSimulate: TCheckBox;
    edtReSimulateMinRecoveredNodeCount: TEdit;
    lblMinRecoveredSIR: TLabel;
    lblSimulation: TLabel;

    procedure btnCancelClick(Sender: TObject);
    procedure btnApplyClick(Sender: TObject);
    procedure cbxReSimulateChange(Sender: TObject);
    procedure edtReSimulateMinRecoveredNodeCountKeyPress(Sender:
   TObject; var Key: char);
    procedure FormClose(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure LoadSettings;
  private

  public

  end;

var
  frmSettings: TfrmSettings;

implementation

procedure TfrmSettings.LoadSettings;
var
  i, j: Integer;
  JsonData: TJSONData;
  JsonObject, settings: TJSONObject;
begin
  { Generate default settings if they do not exist }
  if not FileExists(SETTINGS_FILE_NAME) then FileHandler.
   WriteToJsonFile(SETTINGS_FILE_NAME, GetJSON(DEFAULT_SETTINGS));
  { Load the settings }
  try
    JsonData := FileHandler.LoadJsonFile(SETTINGS_FILE_NAME);
    JsonObject := TJSONObject(JsonData);
```

```
    settings := JsonObject.Find(SETTINGS_NAME) as TJSONObject;
    for i := 0 to settings.Count - 1 do begin
      for j := 0 to self.ComponentCount - 1 do begin
        if self.Components[j].name = settings.Names[i] then begin
          if self.Components[j] is TCheckBox then (self.Components[j]
   as TCheckbox).Checked := settings.FindPath(TJSONObject(settings).
   Names[i]).AsBoolean
          else if self.Components[j] is TEdit then (self.Components[j]
   as TEdit).Text := settings.FindPath(TJSONObject(settings).Names[i]).
   AsString;
        end;
      end;
    end;
  except on E:Exception do begin end;
  end;
end;

procedure TfrmSettings.FormCreate(Sender: TObject);
begin
  inherited;
  self.LoadSettings; // Load Application Settings
  if self.cbxReSimulate.Checked then self.
   edtReSimulateMinRecoveredNodeCount.Enabled := True
  else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
end;

procedure TfrmSettings.FormClose(Sender: TObject);
begin
  self.LoadSettings;
  inherited;
end;

procedure TfrmSettings.btnCancelClick(Sender: TObject);
begin
  self.Close;
end;

procedure TfrmSettings.btnApplyClick(Sender: TObject);
var
 JsonObject, settings: TJSONObject;
begin
  { Load the default settings }
  JsonObject := TJSONObject(GetJSON(DEFAULT_SETTINGS));

  { Get the new settings }
  settings := JsonObject.Find(SETTINGS_NAME) as TJSONObject;
  settings.Booleans[RE_SIMULATE_SETTING_NAME] := self.cbxReSimulate.
   Checked;
  if self.edtReSimulateMinRecoveredNodeCount.Text <> '' then settings.
   Strings[RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] :=
   self.edtReSimulateMinRecoveredNodeCount.Text
  else settings.Strings[
   RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] := '0';

  { Write the new settings }
  FileHandler.WriteToJsonFile(SETTINGS_FILE_NAME, TJSONData(JsonObject)
   );

  self.LoadSettings; // Load Application Settings
end;
```

```
106
107 procedure TfrmSettings.cbxReSimulateChange(Sender: TObject);
108 begin
109   if self.cbxReSimulate.Checked then begin
110     self.edtReSimulateMinRecoveredNodeCount.Enabled := True;
111   end
112   else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
113 end;
114
115 procedure TfrmSettings.edtReSimulateMinRecoveredNodeCountKeyPress(
    Sender: TObject; var Key: char
116   );
117 begin
118   ValidateInteger(Sender, Key);
119 end;
120
121 {$R *.lfm}
122
123 end.
```