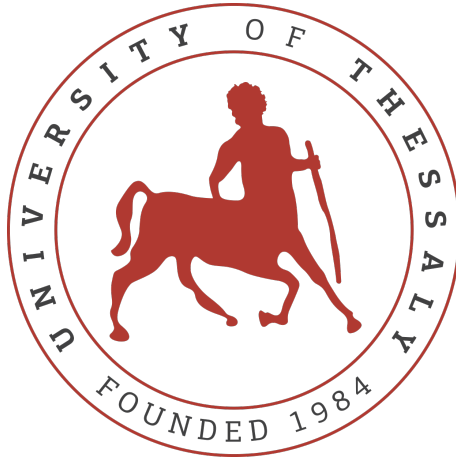


# ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ



ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΜΗΧΑΝΙΚΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΔΙΑΔΙΚΤΥΑΚΕΣ & ΦΟΡΗΤΕΣ  
ΕΦΑΡΜΟΓΕΣ

## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ  
ΒΑΣΙΛΕΙΟΣ ΒΛΑΧΟΣ

### Δημιουργία λογισμικού προσομοίωσης βιολογικών ιών

ΘΕΟΔΩΡΟΣ ΣΙΚΛΑΦΙΔΗΣ

A.M.: 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

Περιεχόμενα . . . . .	2
<b>1 Περίληψη . . . . .</b>	<b>4</b>
<b>2 Εισαγωγή . . . . .</b>	<b>5</b>
2.1 Ορισμός Επιδημιολογίας . . . . .	5
2.2 Ιστορική Αναδρομή . . . . .	5
<b>3 Σχετικά Έργα . . . . .</b>	<b>7</b>
3.1 Epidemix . . . . .	7
3.2 GLEAMviz . . . . .	7
3.3 EpiSimS . . . . .	7
3.4 FluPhone . . . . .	8
3.5 Sispread . . . . .	8
<b>4 Διαμερισματικά μοντέλα . . . . .</b>	<b>9</b>
4.1 SIR . . . . .	9
4.1.1 Αλγόριθμος SIR . . . . .	11
4.2 SIS . . . . .	12
4.2.1 Αλγόριθμος SIS . . . . .	14
4.3 SIQ . . . . .	15
4.3.1 Αλγόριθμος SIQ . . . . .	17
4.4 SIQS . . . . .	19
4.4.1 Αλγόριθμος SIQS . . . . .	20
4.5 SIQR . . . . .	22
4.5.1 Αλγόριθμος SIQR . . . . .	24
4.6 SIRD . . . . .	26
4.6.1 Αλγόριθμος SIRD . . . . .	27
4.7 MSIR . . . . .	29
4.7.1 Αλγόριθμος MSIR . . . . .	30
4.8 SEIR . . . . .	31
4.8.1 Αλγόριθμος SEIR . . . . .	33
4.9 SEIS . . . . .	34
4.9.1 Αλγόριθμος SEIS . . . . .	36
4.10 MSEIR . . . . .	38
4.10.1 Αλγόριθμος MSEIR . . . . .	39
<b>5 Αρχιτεκτονική λογισμικού . . . . .</b>	<b>42</b>
<b>6 Υλοποίηση . . . . .</b>	<b>45</b>
<b>7 Προσαρμοσμένοι αλγόριθμοι . . . . .</b>	<b>48</b>

7.1	SIR	48
7.1.1	Πηγαίος κώδικας SIR	49
7.2	SIS	54
7.2.1	Πηγαίος κώδικας SIS	54
7.3	Προσομοίωση σε τυχαίο και ελεύθερης κλίμακας γράφο	60
<b>8</b>	<b>Συμπεράσματα</b>	<b>62</b>
	<b>Βιβλιογραφία</b>	<b>63</b>
<b>A</b>	<b>Παράρτημα - Πηγαίος Κώδικας</b>	<b>66</b>
A.1	Κλάση που αντιπροσωπεύει έναν κόμβο και τα χαρακτηριστικά του	66
A.2	Επιλυτής διαφορικών εξισώσεων με μέθοδο Forward Euler	68
A.3	Κώδικας διαχείρισης αρχείων	70
A.4	Διαχείριση ρυθμίσεων εφαρμογής	75
A.5	Σταθερές μεταβλητές εφαρμογής	78
A.6	Φόρμες	78
A.6.1	Αρχική	78
A.6.2	Γραφική παράσταση διαφορικών εξισώσεων	88
A.6.3	Γραφική παράσταση προσομοίωσης	98
A.6.4	Ρυθμίσεις εφαρμογής	105

## Περίληψη

Η ανάπτυξη και η πορεία μολυσματικών ασθενειών, βιολογικών ιών και παθογόνων ταχέως 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 Άγγλος 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 μαστίζαν το Λονδίνο τις χρονιές 1660 μέχρι 1670 σε τρεις κατηγορίες (συνεχείς πυρετοί, διαλείπουσα πυρετό και ευλογία), ήταν ένα από τα πιο σημαντικά έργα του Thomas Sydenham.

Κατά την διάρκεια του 1854, ο John Snow ο οποίος ήταν εξίσου Άγγλος 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 Γλασκώβης. 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 τους 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 των 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 ασθενειών σε ολόκληρο τον κόσμο. Το GLEAMviz μπορεί να χρησιμοποιηθεί 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 διαφορετικές 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 αλγόριθμους οι οποίοι μπορούν να χρησιμοποιηθούν με σκοπό την ανάλυση της πορείας μιας μολυσματικής ασθένειας.

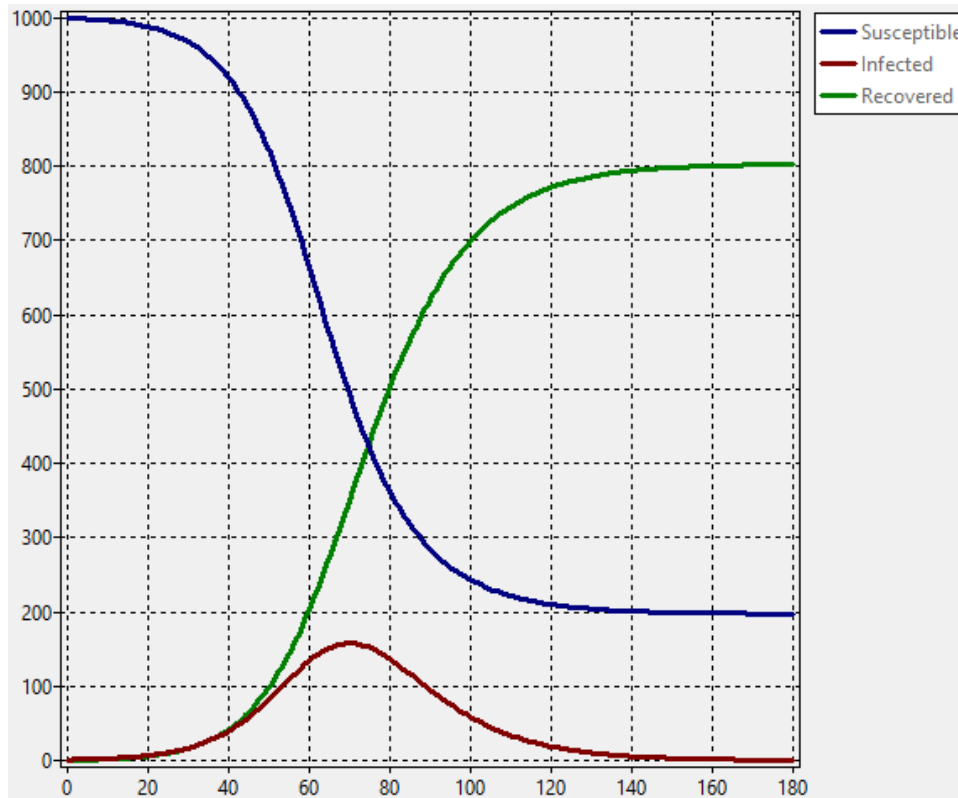
## Διαμερισματικά μοντέλα

Από το πρώτο ακόμη διαμερισματικό μοντέλο που δημιουργήθηκε το 1927, έχει μεσολαμβάνει 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 αποτελείται από μολυσμένα άτομα τα οποία έχουν την δυνατότητα πλέον, να μολύνουν άλλα ευάλωτα άτομα. Τέλος το group αυτών που έχουν περάσει την ασθένεια αποτελείται 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:

$$\begin{aligned}\frac{dS}{dt} &= -\frac{\beta \cdot I \cdot S}{N} \\ \frac{dI}{dt} &= \frac{\beta \cdot I \cdot S}{N} - \gamma \cdot I \\ \frac{dR}{dt} &= \gamma \cdot I\end{aligned}\tag{4.1}$$



Εικόνα 4.1: BVS |  $N$ : 1000,  $\beta$ : 0.2,  $\gamma$ : 0.1, Initial Infected: 1, Days: 180

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  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

```
1 unit utlSIR_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
14 function SIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17   function SIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18   var
19     N, beta, gamma: Double;
20     S, I: Double;
21     dSdt, dIdt, dRdt: Double;
22   begin
23     {
24       [ Description ]
25       [*****]
26       Calculate the SIR's (Susceptible - Infectious - Recovered) model
27       differential equations given a y state.
28
29       [*****]
30
31       [ Parameters ]
32       [*****]
33       Param y: Array of double containing the 'to be calculated' S I R
34       states.
35       Param extraArgs: An array of double containing the extra
36       arguments that are needed in order to perform calculations. The
37       correct form of the array should be:
38
39           0   1   2
40           [N, beta, gamma]
41
42       [*****]
43
44       [ Variables ]
45       [*****]
46       Var N: The total population.
47
48       Var beta: The average number of contacts per person per time.
49                 (Infection rate)
50       Var gamma: The recovery rate.
51
52       Var S: Number of Susceptible individuals.
53       Var I: Number of Infected individuals.
54
55       Var dSdt: Calculated differential equation result for
56       Susceptibles.
57       Var dIdt: Calculated differential equation result for Infected.
58       Var dRdt: Calculated differential equation result for Recovered.
59     }
```

```

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

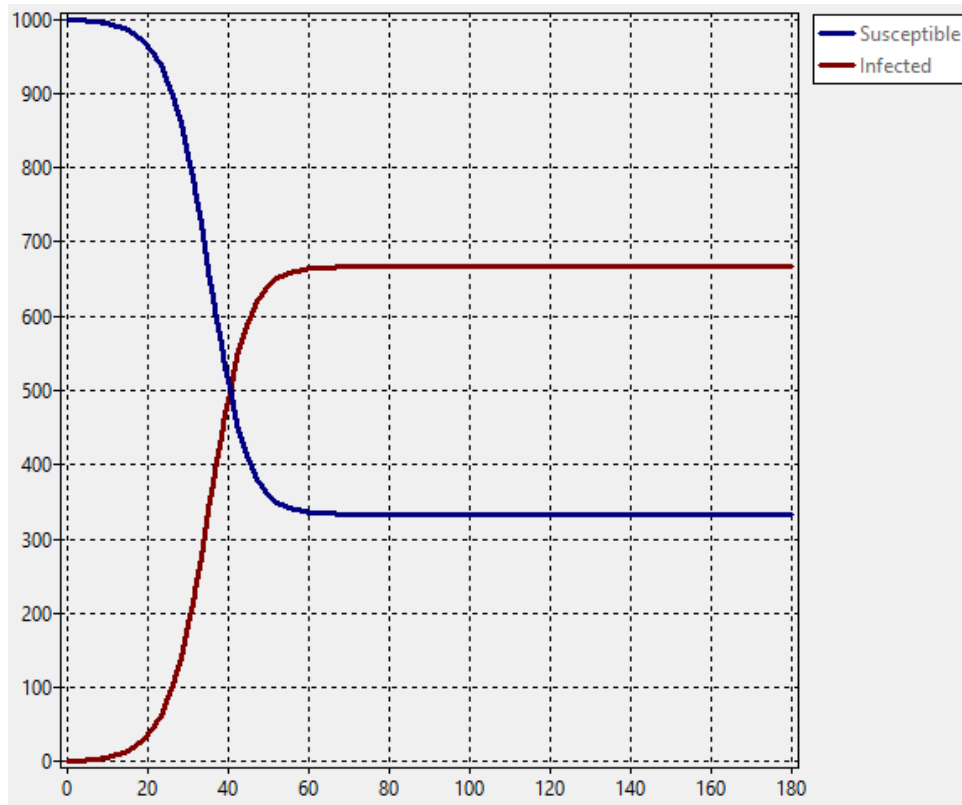
Παρόμοια με το μοντέλο SIR, το SIS αποτελείται από τα διαμερίσματα ευαίσθητων (Susceptible) και μολυσμένων (Infectious). Η μόνη διαφορά μεταξύ τους είναι ότι όταν ένα μολυσμένο άτομο ξεπεράσει την ασθένεια, αντί να μετακινηθεί στο διαμέρισμα όπου βρίσκονται αυτοί οι οποίοι έχουν ξεπεράσει την ασθένεια (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:

$$\frac{dS}{dt} = -\frac{\beta \cdot I \cdot S}{N} + \gamma \cdot I \quad (4.2)$$

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

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.



Εικόνα 4.2: BVS | N: 1000,  $\beta$ : 0.3,  $\gamma$ : 0.1, Initial Infected: 1, Days: 180

## 4.2.1 Αλγόριθμος SIS

```
1 unit utlSIS_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
14 function SISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17 function SISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18 var
19   N, beta, gamma: Double;
20   S, I: Double;
21   dSdt, dIdt: Double;
22 begin
23   {
24     [ Description ]
25     [*****]
26     Calculate the SIS's (Susceptible - Infectious - Susceptible)
27     model differential equations given a y state.
28     [*****]
29
30     [ Parameters ]
31     [*****]
32     Param y: Array of double containing the 'to be calculated' S I
33     states.
34     Param extraArgs: An array of double containing the extra
35     arguments that are needed in order to perform calculations. The
36     correct form of the array should be:
37           0   1   2
38           [N, beta, gamma]
39     [*****]
40
41     [ Variables ]
42     [*****]
43     Var N: The total population.
44
45     Var beta: The average number of contacts per person per time.
46               (Infection rate)
47
48     Var gamma: The recovery rate.
49
50     Var S: Number of Susceptible individuals.
51     Var I: Number of Infected individuals.
52
53     Var dSdt: Calculated differential equation result for
54     Susceptibles.
55     Var dIdt: Calculated differential equation result for Infected.
56     [*****]
```

```

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

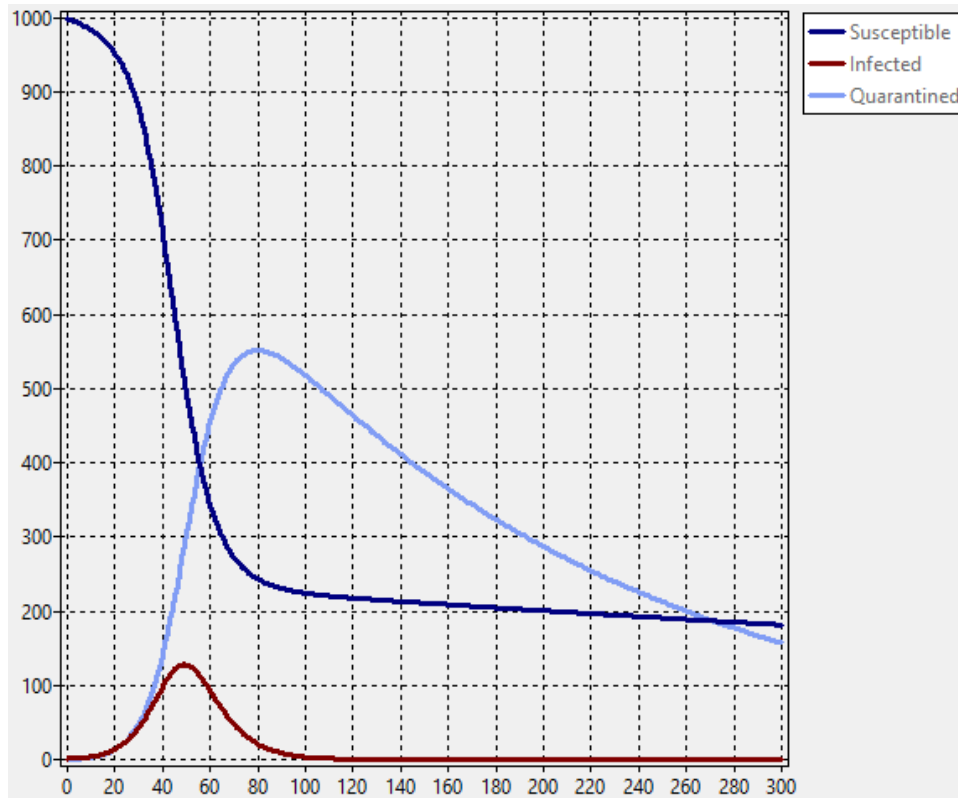
```

### 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 περιοριζόνταν 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:

$$\begin{aligned} \frac{dS}{dt} &= \Lambda - \frac{\beta \cdot I \cdot S}{N} - \mu \cdot S \\ \frac{dI}{dt} &= \Lambda + \frac{\beta \cdot I \cdot S}{N} - (\mu + \delta + \kappa) \cdot I \\ \frac{dQ}{dt} &= \kappa \cdot I - (\mu + \delta) \cdot Q \end{aligned} \quad (4.3)$$





Εικόνα 4.3: BVS |  $N$ : 1000,  $\beta$ : 0.3,  $\mu$ : 0.001,  $\Lambda$ : 0,  $\delta$ : 0.005,  $\kappa$ : 0.15, Initial Infected: 1, Days: 300

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\delta$  είναι ο ρυθμός θνησιμότητας της ασθένειας.
- $\kappa$  είναι ο ρυθμός όπου τα μολυσμένα άτομα περνούν σε καραντίνα.

### 4.3.1 Αλγόριθμος SIQ

```
1 unit utlSIQ_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
14 function SIQDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17 function SIQDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18 var
19   N, beta, mu, lambda, delta, kappa: Double;
20   S, I, Q: Double;
21   dSdt, dIdt, dQdt: Double;
22 begin
23   {
24     [ Description ]
25     [*****]
26     Calculate the SIQ's (Susceptible - Infectious - Quarantined)
27     model differential equations given a y state.
28     [*****]
29
30     [ Parameters ]
31     [*****]
32     Param y: Array of double containing the 'to be calculated' S I Q
33     states.
34     Param extraArgs: An array of double containing the extra
35     arguments that are needed in order to perform calculations. The
36     correct form of the array should be:
37
38         0   1   2   3   4   5
39         [N, beta, mu, lambda, delta, kappa]
40
41     [*****]
42
43     [ Variables ]
44     [*****]
45     Var N: The total population.
46
47     Var beta: The average number of contacts per person per time.
48               (Infection rate)
49     Var mu: Natural mortality rate.
50     Var lambda: Recruitment of the susceptible individuals (birth etc
51               .)
52     Var delta: Disease mortality rate.
53     Var Kappa: The rate where the infected individuals become
54               quarantined.
55
56     Var S: Number of Susceptible individuals.
57     Var I: Number of Infected individuals.
58     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.

```

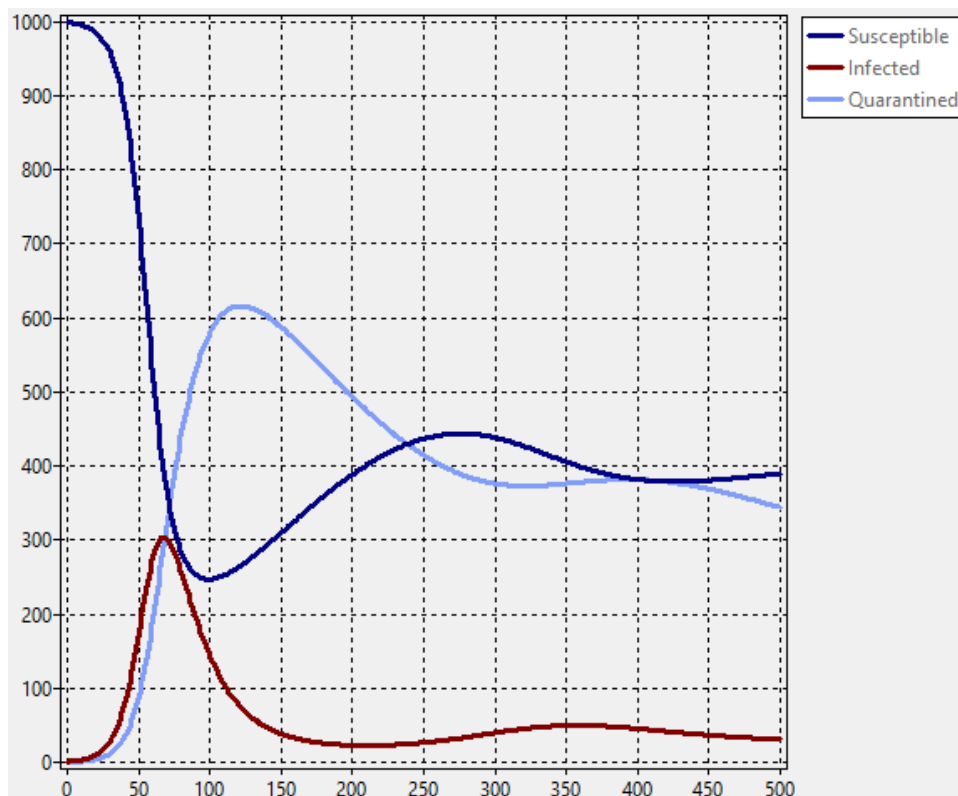
## 4.4 SIQS

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:

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

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

$$\frac{dQ}{dt} = \kappa \cdot I - (\mu + \delta_1 + \zeta) \cdot Q$$



Εικόνα 4.4: BVS | N: 1000,  $\beta$ : 0.3,  $\gamma$ : 0.03,  $\mu$ : 0.0001,  $\Lambda$ : 0.01,  $\delta$ : 0.0001,  $\delta_1$ : 0.001,  $\kappa$ : 0.05,  $\zeta$ : 0.005, Initial Infected: 1, Days: 500

Όπου:

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

- $\gamma$  είναι ο ρυθμός ανάρρωσης.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\delta$  είναι ο ρυθμός θνησιμότητας της ασθένειας.
- $\delta_1$  είναι ο ρυθμός θνησιμότητας της ασθένειας, για όσους βρίσκονται ήδη σε καραντίνα.
- $\kappa$  είναι ο ρυθμός όπου τα μολυσμένα άτομα περνούν σε καραντίνα.
- $\zeta$  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

```

1 unit utlSIQS_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
14 function SIQSDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17   function SIQSDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18   var
19     N, beta, gamma, mu, lambda, delta, delta1, kappa, zeta: Double;
20     S, I, Q: Double;
21     dSdt, dIdt, dQdt: Double;
22   begin
23     {
24     [ Description ]
25     [*****]
26     Calculate the SIQ's (Susceptible - Infectious - Quarantined -
27     Susceptible) model differential equations given a y state.
28     [*****]
29
30     [ Parameters ]
31     [*****]
32     Param y: Array of double containing the 'to be calculated' S I Q
33     states.
34     Param extraArgs: An array of double containing the extra
35     arguments that are needed in order to perform calculations. The
36     correct form of the array should be:
37
38         0      1      2      3      4      5      6      7      8
39         [N, beta, gamma, mu, lambda, delta, delta1 kappa, zeta]

```

```

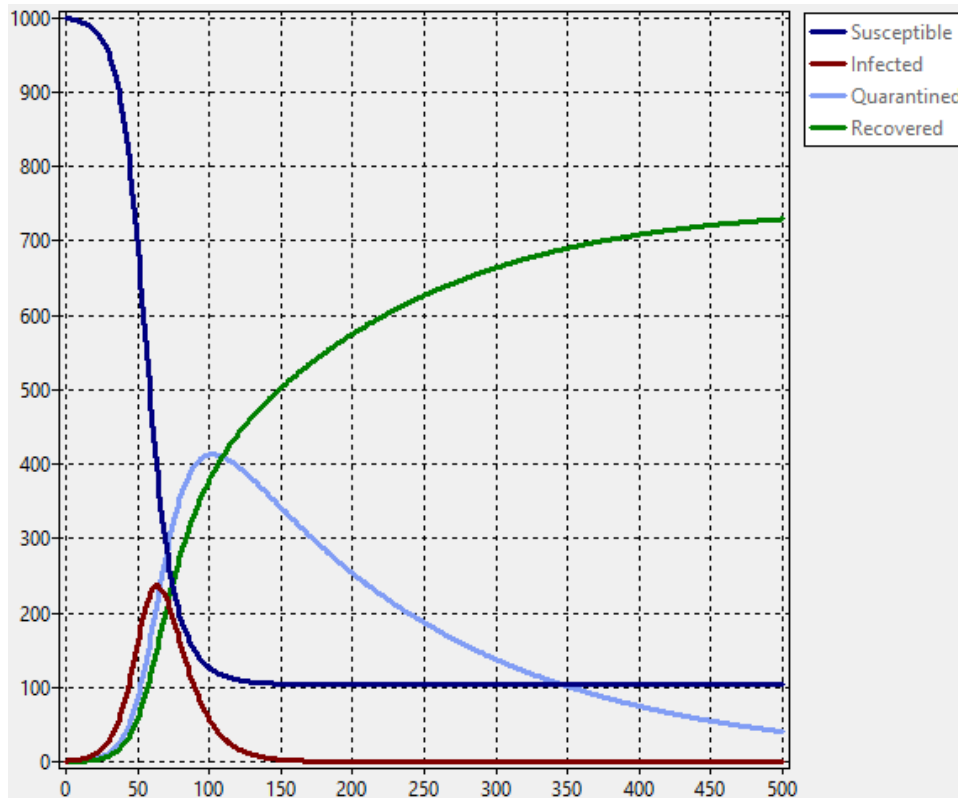
37
38     [*****]
39
40     [ Variables ]
41     [*****]
42     Var N: The total population.
43
44     Var beta: The average number of contacts per person per time.
45             (Infection rate)
46     Var gamma: The recovery rate.
47     Var mu: Natural mortality rate.
48     Var lambda: Recruitment of the susceptible individuals (birth etc
49     .)
50     Var delta: Disease mortality rate of infectious individuals.
51     Var delta1: Disease mortality rate of quarantined individuals.
52     Var Kappa: The rate where the infected individuals become
53     quarantined.
54     Var Zeta: The rate where quarantined individuals become
55     susceptible again.
56
57     Var S: Number of Susceptible individuals.
58     Var I: Number of Infected individuals.
59     Var Q: Number of Quarantined individuals.
60
61     Var dSdt: Calculated differential equation result for
62     Susceptibles.
63     Var dIdt: Calculated differential equation result for Infected.
64     Var dQdt: Calculated differential equation result for Quarantined
65     .
66
67     [*****]
68
69 }
70
71 S := y[0];
72 I := y[1];
73 Q := y[2];
74
75 N := extraArgs[0];
76 beta := extraArgs[1];
77 gamma := extraArgs[2];
78 mu := extraArgs[3];
79 lambda := extraArgs[4];
80 delta := extraArgs[5];
81 delta1 := extraArgs[6];
82 kappa := extraArgs[7];
83 zeta := extraArgs[8];
84
85 dSdt := lambda - (beta * S * I / N) - (mu * S) + (gamma * I) + (
86 zeta * Q);
87 dIdt := (beta * S * I / N) - ((mu + delta + kappa + gamma) * I);
88 dQdt := (kappa * I) - ((mu + delta1 + zeta) * Q);
89
90 Result := ArrayOfDouble.Create(dSdt, dIdt, dQdt);
91 end;
92
93 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 ανοσία 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:

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



Εικόνα 4.5: BVS |  $N$ : 1000,  $\beta$ : 0.3,  $\gamma$ : 0.03,  $\mu$ : 0.0001,  $\Lambda$ : 0.01,  $\delta$ : 0.0001,  $\delta_1$ : 0.001,  $\kappa$ : 0.05,  $\zeta$ : 0.005, Initial Infected: 1, Days: 500

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\delta$  είναι ο ρυθμός θνησιμότητας της ασθένειας.
- $\delta_1$  είναι ο ρυθμός θνησιμότητας της ασθένειας, για όσους βρίσκονται ήδη σε καραντίνα.
- $\kappa$  είναι ο ρυθμός όπου τα μολυσμένα άτομα περνούν σε καραντίνα.
- $\zeta$  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

```
1 unit utlSIQR_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
14 function SIQRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17 function SIQRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18 var
19   N, beta, gamma, mu, lambda, delta, delta1, kappa, zeta: Double;
20   S, I, Q, R: Double;
21   dSdt, dIdt, dQdt, dRdt: Double;
22 begin
23   {
24     [ Description ]
25     [*****]
26     Calculate the SIQR's (Susceptible - Infectious - Quarantined -
27     Recovered) model differential equations given a y state.
28     [*****]
29
30     [ Parameters ]
31     [*****]
32     Param y: Array of double containing the 'to be calculated' S I Q
33     R states.
34     Param extraArgs: An array of double containing the extra
35     arguments that are needed in order to perform calculations. The
36     correct form of the array should be:
37         0   1   2   3   4   5   6   7   8
38         [N, beta, gamma, mu, lambda, delta, delta1, kappa, zeta]
39     [*****]
40
41     [ Variables ]
42     [*****]
43     Var N: The total population.
44
45     Var beta: The average number of contacts per person per time.
46               (Infection rate)
47
48     Var gamma: The recovery rate.
49
50     Var mu: Natural mortality rate.
51
52     Var lambda: Recruitment of the susceptible individuals (birth etc
53     .)
54
55     Var delta: Disease mortality rate of infectious individuals.
56
57     Var delta1: Disease mortality rate of quarantined individuals.
58
59     Var Kappa: The rate where the infected individuals become
60     quarantined.
61
62     Var Zeta: The rate where quarantined individuals become
63     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:

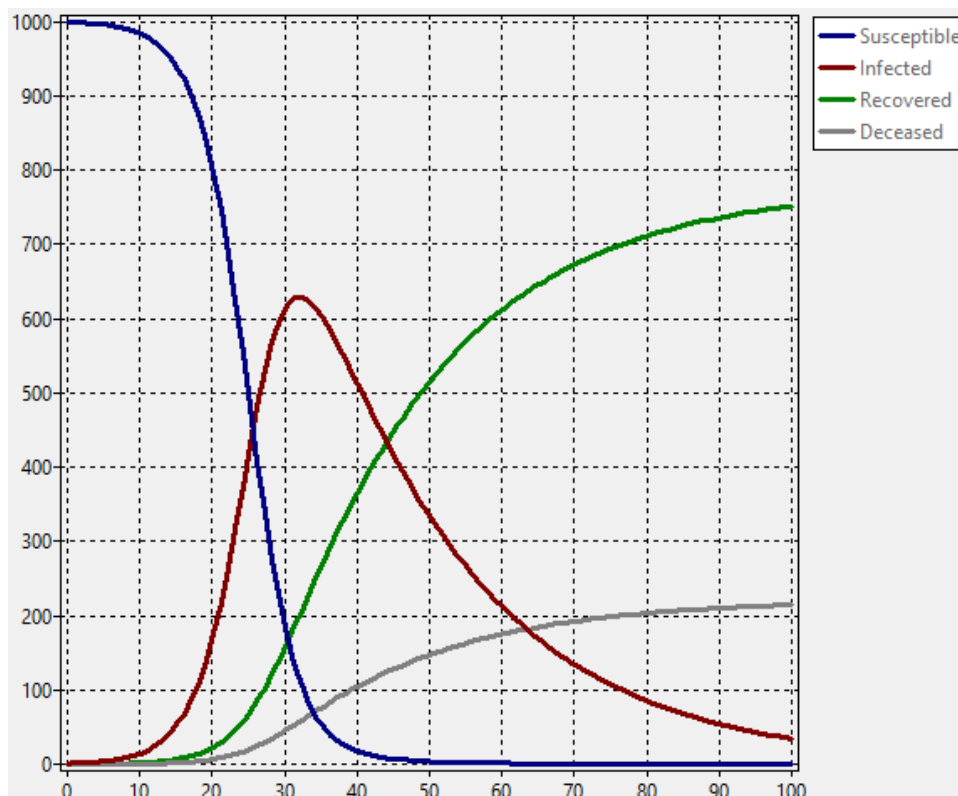
$$\frac{dS}{dt} = -\frac{\beta \cdot I \cdot S}{N}$$

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

(4.6)

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

$$\frac{dD}{dt} = \mu \cdot I$$



Εικόνα 4.6: BVS | N: 1000,  $\beta$ : 0.35,  $\gamma$ : 0.035,  $\mu$ : 0.01, Initial Infected: 1, Days: 100

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.

## 4.6.1 Αλγόριθμος SIRD

```
1 unit utlSIRD_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
14 function SIRDDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17   function SIRDDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18   var
19     N, beta, gamma, mu: Double;
20     S, I: Double;
21     dSdt, dIdt, dRdt, dDdt: Double;
22   begin
23     {
24       [ Description ]
25       [*****]
26       Calculate the SIRD's (Susceptible - Infectious - Recovered -
27       Deceased) model differential equations given a y state.
28       [*****]
29
30       [ Parameters ]
31       [*****]
32       Param y: Array of double containing the 'to be calculated' S I R
33       D states.
34       Param extraArgs: An array of double containing the extra
35       arguments that are needed in order to perform calculations. The
36       correct form of the array should be:
37
38           0   1   2   3
39           [N, beta, gamma, mu]
40       [*****]
41
42       [ Variables ]
43       [*****]
44       Var N: The total population.
45
46       Var beta: The average number of contacts per person per time.
```

```

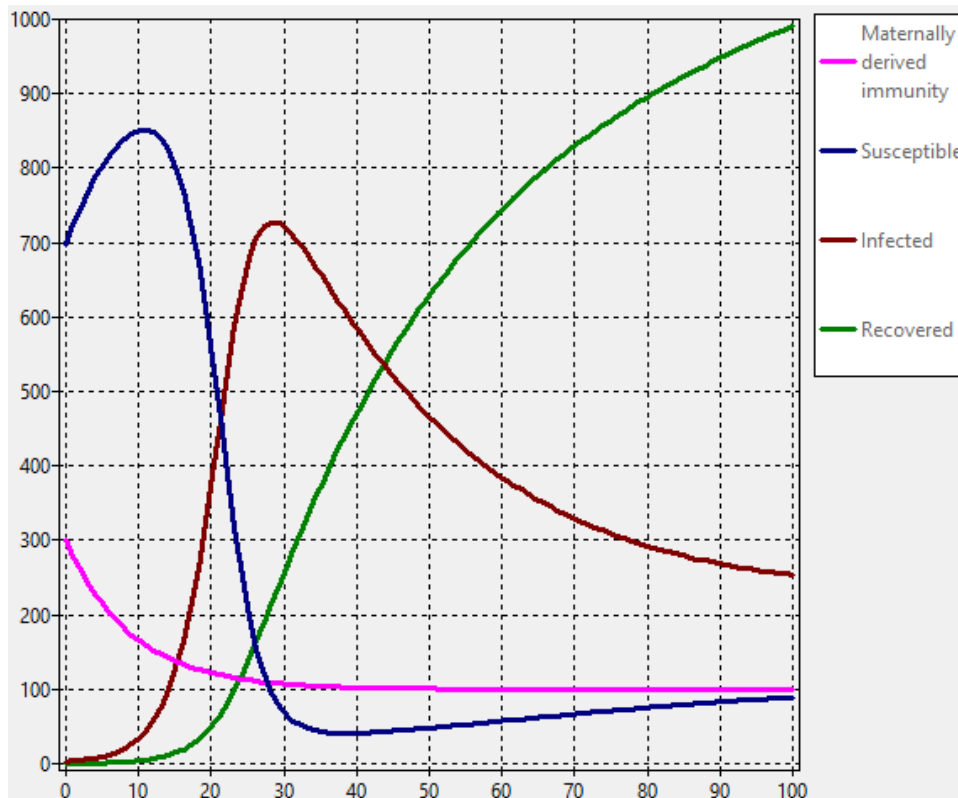
44         (Infection rate)
45     Var gamma: The recovery rate.
46     Var mu: Natural mortality rate.
47
48     Var S: Number of Susceptible individuals.
49     Var I: Number of Infected individuals.
50
51     Var dSdt: Calculated differential equation result for
Susceptibles.
52     Var dIdt: Calculated differential equation result for Infected.
53     Var dRdt: Calculated differential equation result for Recovered.
54     Var dDdt: Calculated differential equation result for Deceased.
55
56     [*****]
57
58 }
59
60 S := y[0];
61 I := y[1];
62
63 N := extraArgs[0];
64 beta := extraArgs[1];
65 gamma := extraArgs[2];
66 mu := extraArgs[3];
67
68 dSdt := -(beta * S * I / N);
69 dIdt := (beta * S * I / N) - (gamma * I) - (mu * I);
70 dRdt := gamma * I;
71 dDdt := mu * I;
72
73 Result := ArrayOfDouble.Create(dSdt, dIdt, dRdt, dDdt);
74 end;
75
76 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,  $\beta$ : 0.4,  $\gamma$ : 0.035,  $\mu$ : 0.005,  $\Lambda$ : 10.5,  $\delta$ : 0.1, Maternally derived immunity: 300, Initial Infected: 3, Days: 100

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\delta$  είναι ο ρυθμός θνησιμότητας της ασθένειας.

### 4.7.1 Αλγόριθμος MSIR

```
1 unit utlMSIR_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
14 function MSIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17   function MSIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18   var
19     N, beta, gamma, mu, lambda, delta: Double;
20     M, S, I, R: Double;
21     dMdt, dSdt, dIdt, dRdt: Double;
22   begin
23     {
24       [ Description ]
25       [*****]
26       Calculate the MSIR's
27       (Maternally derived immunity - Susceptible - Infectious -
28       Recovered) model differential equations given a y state.
29       [*****]
30
31       [ Parameters ]
32       [*****]
33       Param y: Array of double containing the 'to be calculated' M S I
34       R states.
35       Param extraArgs: An array of double containing the extra
36       arguments that are needed in order to perform calculations. The
37       correct form of the array should be:
38
39           0     1     2     3     4     5
40           [N, beta, gamma, mu, lambda, delta]
```

```

39      [*****]
40
41      [ Variables ]
42      [*****]
43      Var N: The total population.
44
45      Var beta: The average number of contacts per person per time.
46                (Infection rate)
47      Var gamma: The recovery rate.
48      Var mu: Natural mortality rate.
49      Var lambda: Recruitment of the susceptible individuals (birth etc
50      .)
51      Var delta: Disease mortality rate.
52
53      Var M: Number of individuals with Maternally derived immunity.
54      Var S: Number of Susceptible individuals.
55      Var I: Number of Infected individuals.
56      Var R: Number of Recovered individuals.
57
58      Var dMdt: Calculated differential equation result for Maternally
59                derived immunity.
60      Var dSdt: Calculated differential equation result for
61      Susceptibles.
62      Var dIdt: Calculated differential equation result for Infected.
63      Var dRdt: Calculated differential equation result for Recovered.
64
65      [*****]
66
67      }
68
69      M := y[0];
70      S := y[1];
71      I := y[2];
72      R := y[3];
73
74      N := extraArgs[0];
75      beta := extraArgs[1];
76      gamma := extraArgs[2];
77      mu := extraArgs[3];
78      lambda := extraArgs[4];
79      delta := extraArgs[5];
80
81      dMdt := lambda - (delta * M) - (mu * M);
82      dSdt := (delta * M) - (beta * S * I / N) - (mu * S);
83      dIdt := (beta * S * I / N) - (gamma * I) - (mu * I);
84      dRdt := (gamma * I) - (mu * R);
85
86      Result := ArrayOfDouble.Create(dMdt, dSdt, dIdt, dRdt);
87      end;
88
89      end.

```

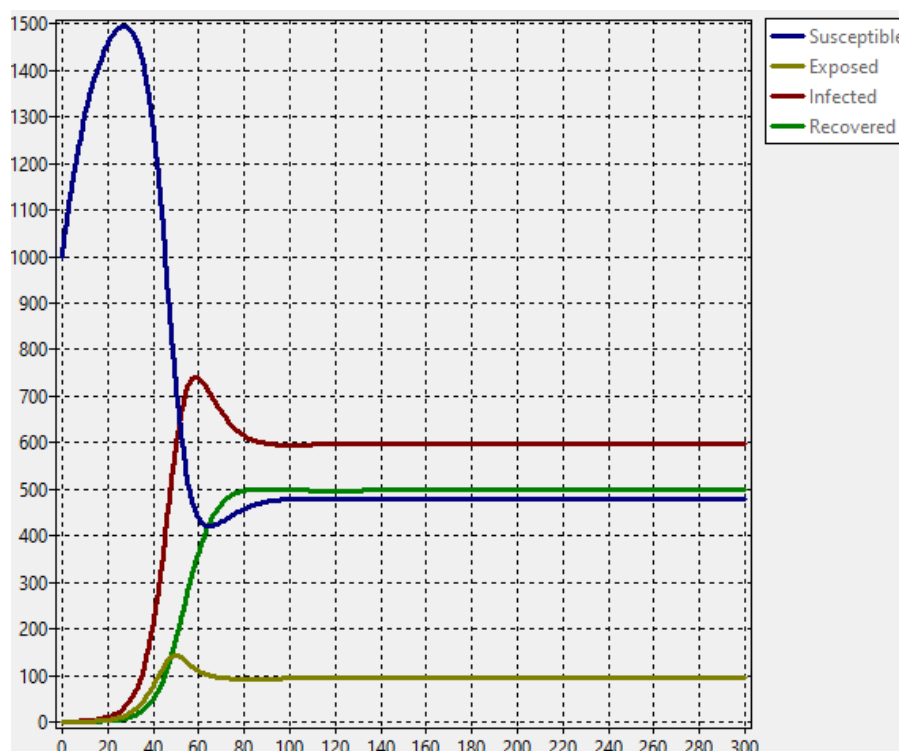
## 4.8 SEIR

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,  $\beta$ : 0.25,  $\gamma$ : 0.05,  $\mu$ : 0.06,  $\Lambda$ : 0.1,  $\alpha$ : 0.7, Initial Infected: 1, Days: 300

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.

- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\alpha$  είναι η μέση περίοδος επώασης ( $\alpha^{-1}$ ).

### 4.8.1 Αλγόριθμος SEIR

```

1 unit utlSEIR_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
14 function SEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17 function SEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18 var
19   N, beta, gamma, mu, lambda, alpha: Double;
20   S, E, I, R: Double;
21   dSdt, dEdt, dIdt, dRdt: Double;
22 begin
23   {
24     [ Description ]
25     [*****]
26     Calculate the SEIR's (Susceptible - Exposed - Infectious -
27     Recovered) model differential equations given a y state.
28     [*****]
29
30     [ Parameters ]
31     [*****]
32     Param y: Array of double containing the 'to be calculated' S E I
33     R states.
34     Param extraArgs: An array of double containing the extra
35     arguments that are needed in order to perform calculations. The
36     correct form of the array should be:
37
38         0      1      2      3      4      5
39         [N, beta, gamma, mu, lambda, alpha]
40     [*****]
41
42     [ Variables ]
43     [*****]
44     Var N: The total population.
45
46     Var beta: The average number of contacts per person per time.
47               (Infection rate)
48     Var gamma: The recovery rate.
49     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

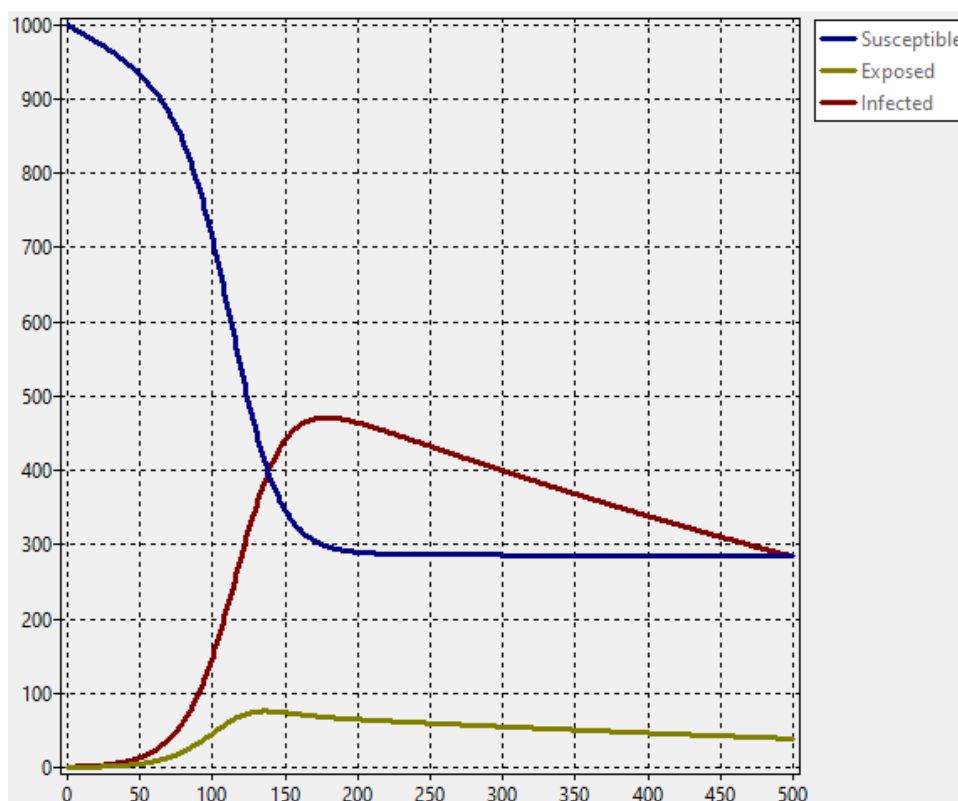
Το μοντέλο επιδημίας SEIS είναι σαν το μοντέλο SEIR με την διαφορά ότι στο τέλος δεν παρέχεται ανοσία. Αντί αυτού, τα μολυσμένα άτομα επιστρέφουν σε μια κατάσταση όπου είναι και πάλι ευπαθή στη νόσο [30]. Το μοντέλο SEIS μπορεί να περιγραφεί από τις

παρακάτω διαφορικές εξισώσεις:

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

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

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



Εικόνα 4.9: BVS |  $N$ : 1000,  $\beta$ : 0.12,  $\gamma$ : 0.035,  $\mu$ : 0.001,  $\Lambda$ : 0,  $\epsilon$ : 0.25, Initial Infected: 1, Days: 500

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\epsilon$  είναι ο ρυθμός όπου τα εκτεθειμένα άτομα μετατρέπονται σε μολυσμένα άτομα.

## 4.9.1 Αλγόριθμος SEIS

```
1 unit utlSEIS_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
14 function SEISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17 function SEISDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18 var
19   N, beta, gamma, mu, lambda, epsilon: Double;
20   S, E, I: Double;
21   dSdt, dEdt, dIdt: Double;
22 begin
23   {
24     [ Description ]
25     [*****]
26     Calculate the SEIS's (Susceptible - Exposed - Infectious -
27     Susceptible) model differential equations given a y state.
28     [*****]
29
30     [ Parameters ]
31     [*****]
32     Param y: Array of double containing the 'to be calculated' S E I
33     S states.
34     Param extraArgs: An array of double containing the extra
35     arguments that are needed in order to perform calculations. The
36     correct form of the array should be:
37
38         0   1   2   3   4   5
39         [N, beta, gamma, mu, lambda, epsilon]
40     [*****]
41
42     [ Variables ]
43     [*****]
44     Var N: The total population.
45
46     Var beta: The average number of contacts per person per time.
47               (Infection rate)
48     Var gamma: The recovery rate.
49     Var mu: Natural mortality rate.
50     Var lambda: Recruitment of the susceptible individuals (birth etc
51     .)
52     Var epsilon: The rate at which exposed individuals become
53     infectious.
54
55     Var S: Number of Susceptible individuals.
56     Var E: Number of Exposed individuals.
57     Var I: Number of Infected individuals.
```

```

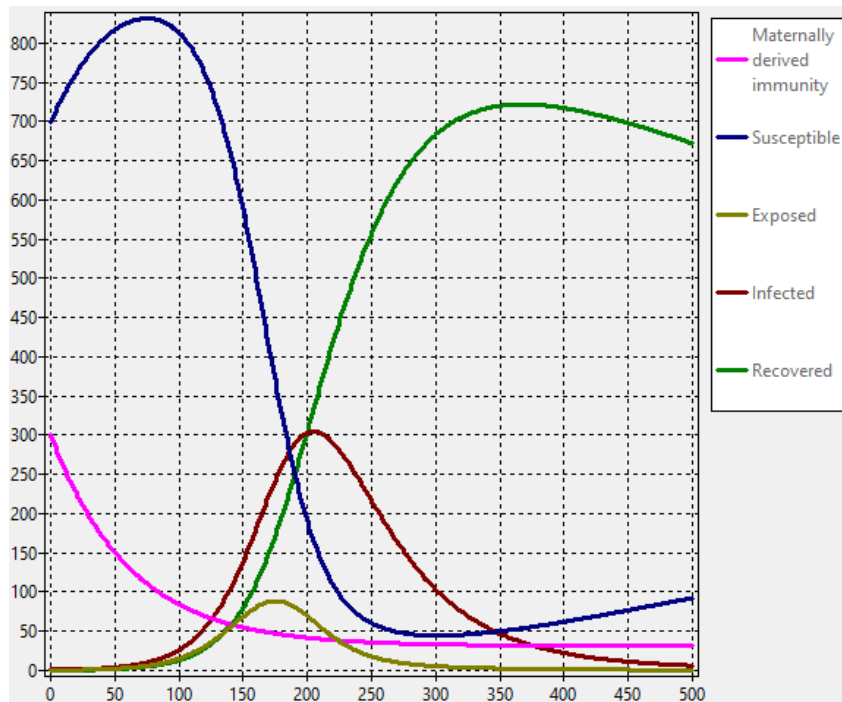
53
54     Var dSdt: Calculated differential equation result for
Susceptibles.
55     Var dEdt: Calculated differential equation result for Exposed.
56     Var dIdt: Calculated differential equation result for Infected.
57
58     [*****]
59
60 }
61
62 S := y[0];
63 E := y[1];
64 I := y[2];
65
66 N := 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,  $\beta$ : 0.1,  $\gamma$ : 0.02,  $\mu$ : 0.001,  $\Lambda$ : 0.5,  $\delta$ : 0.015,  $\epsilon$ : 0.1, Maternally derived immunity: 300, Initial Infected: 1, Days: 500

Όπου:

- $N$  είναι ο συνολικός πληθυσμός.
- $\beta$  είναι ο μέσος αριθμός επαφών κάθε ατόμου, γνωστός και ως *infection rate*.
- $\gamma$  είναι ο ρυθμός ανάρρωσης.
- $\mu$  είναι ο ρυθμός φυσικής θνησιμότητας.
- $\Lambda$  είναι η πρόσληψη των ευπαθών ατόμων (γεννήσεις, δυναμικό μέγεθος πληθυσμού).
- $\delta$  είναι ο ρυθμός θνησιμότητας της ασθένειας.
- $\epsilon$  είναι ο ρυθμός όπου τα εκτεθειμένα άτομα μετατρέπονται σε μολυσμένα άτομα.

### 4.10.1 Αλγόριθμος MSEIR

```

1 unit utlMSEIR_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
14 function MSEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
15
16 implementation
17   function MSEIRDE(y, extraArgs: ArrayOfDouble): ArrayOfDouble;
18   var
19     N, beta, gamma, mu, lambda, delta, epsilon: Double;
20     M, S, E, I, R: Double;
21     dMdt, dSdt, dEdt, dIdt, dRdt: Double;
22   begin
23     {
24       [ Description ]
25       [*****]
26       Calculate the MSEIR's (Maternally derived immunity - Susceptible
27       - Exposed - Infectious - Recovered) model differential equations
28       given a y state.
29
30       [ Parameters ]
31       [*****]
32       Param y: Array of double containing the 'to be calculated' M S E
33       I R states.
34       Param extraArgs: An array of double containing the extra
35       arguments that are needed in order to perform calculations. The
36       correct form of the array should be:
37
38           0     1     2     3     4     5     6
39           [N, beta, gamma, mu, lambda, delta, epsilon]
40     }
41   end;
42 end;

```



```

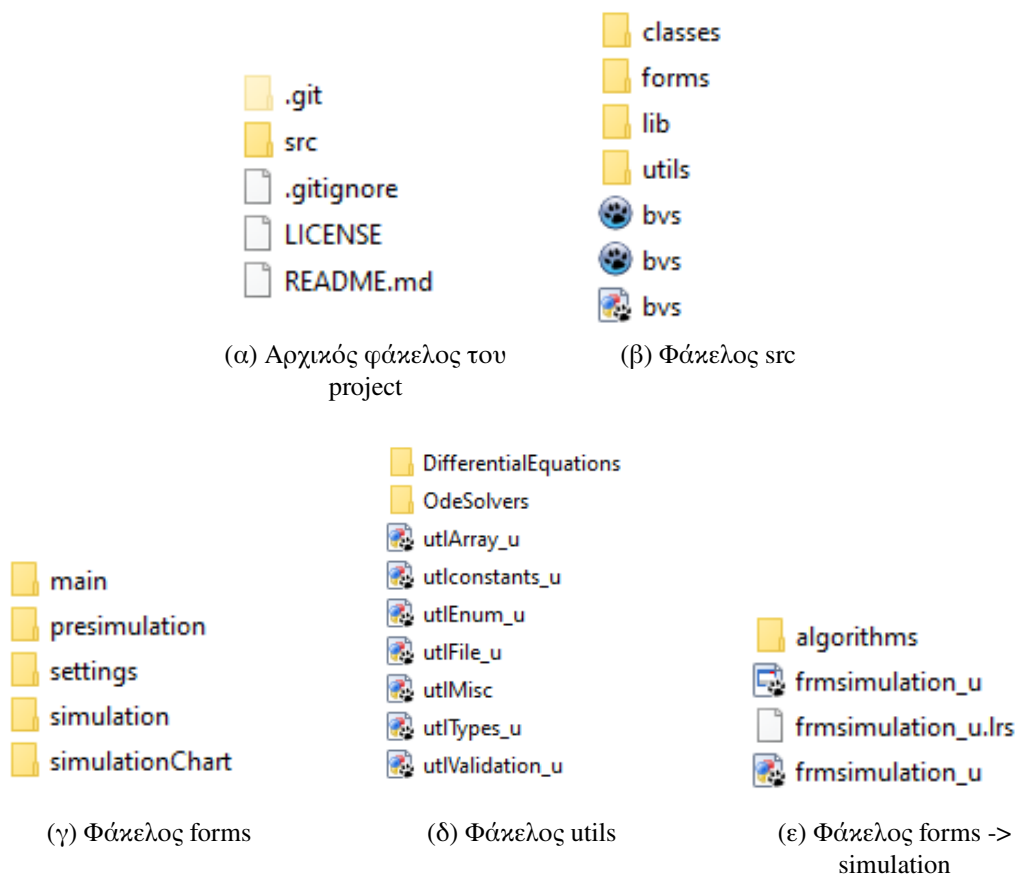
38     [*****]
39
40     [ Variables ]
41     [*****]
42     Var N: The total population.
43
44     Var beta: The average number of contacts per person per time.
45             (Infection rate)
46     Var gamma: The recovery rate.
47     Var mu: Natural mortality rate.
48     Var lambda: Recruitment of the susceptible individuals (birth etc
49     .)
50     Var delta: Disease mortality rate.
51     Var epsilon: The rate at which exposed individuals become
52     infectious.
53
54     Var M: Number of individuals with Maternally derived immunity.
55     Var S: Number of Susceptible individuals.
56     Var E: Number of Exposed individuals.
57     Var I: Number of Infected individuals.
58     Var R: Number of Recovered individuals.
59
60     Var dMdt: Calculated differential equation result for Maternally
61     derived immunity.
62     Var dSdt: Calculated differential equation result for
63     Susceptibles.
64     Var dEdt: Calculated differential equation result for Exposed.
65     Var dIdt: Calculated differential equation result for Infected.
66     Var dRdt: Calculated differential equation result for Recovered.
67
68     [*****]
69     }
70
71     M := y[0];
72     S := y[1];
73     E := y[2];
74     I := y[3];
75     R := y[4];
76
77     N := extraArgs[0];
78     beta := extraArgs[1];
79     gamma := extraArgs[2];
80     mu := extraArgs[3];
81     lambda := extraArgs[4];
82     delta := extraArgs[5];
83     epsilon := extraArgs[6];
84
85     dMdt := lambda - (delta * M) - (mu * M);
86     dSdt := (delta * M) - (beta * S * I / N) - (mu * S);
87     dEdt := (beta * S * I / N) - ((epsilon + mu) * E);
88     dIdt := (epsilon * E) - ((gamma + mu) * I);
89     dRdt := (gamma * I) - (mu * R);
90
91     Result := ArrayOfDouble.Create(dMdt, dSdt, dEdt, dIdt, dRdt);
92     end;
93
94 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 διαφορεικές 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.

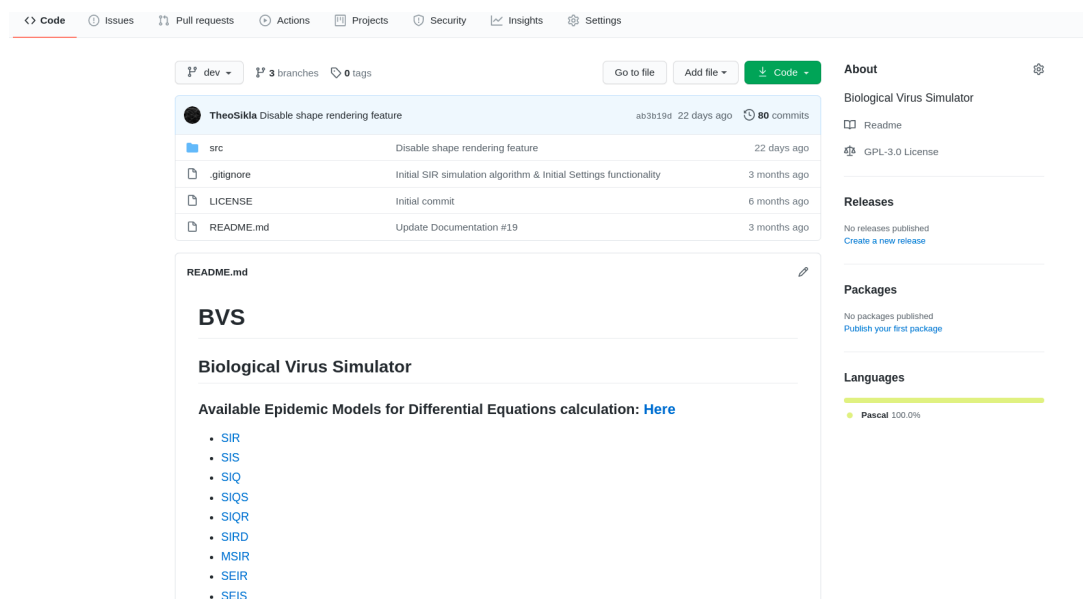
## Αρχιτεκτονική λογισμικού

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.



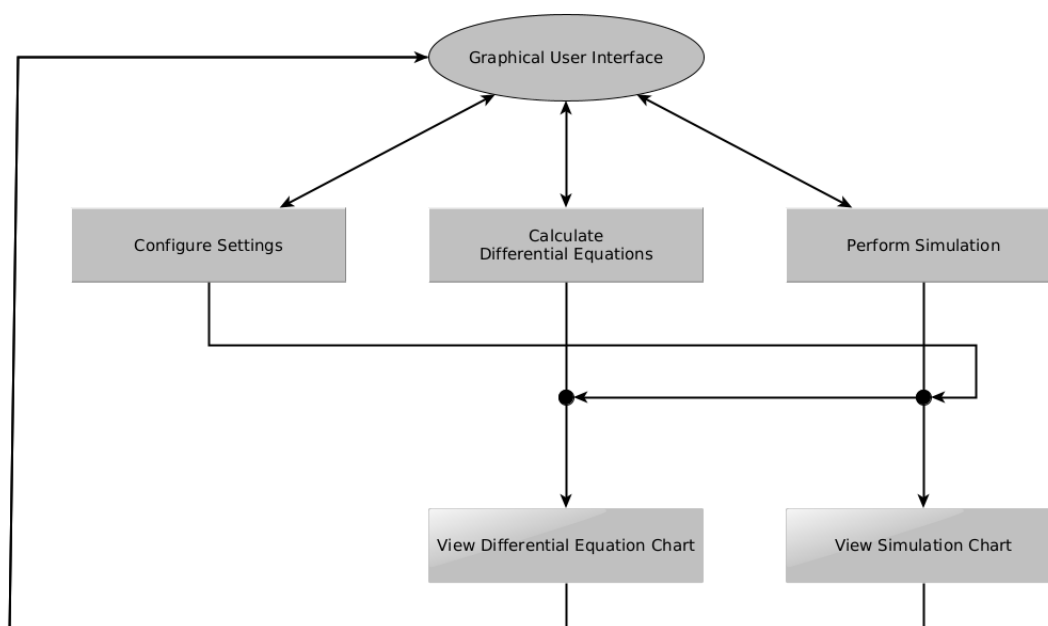
Εικόνα 5.1: BVS | Δομή φακέλων και αρχείων του έργου

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 αναπτύσσονται 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.



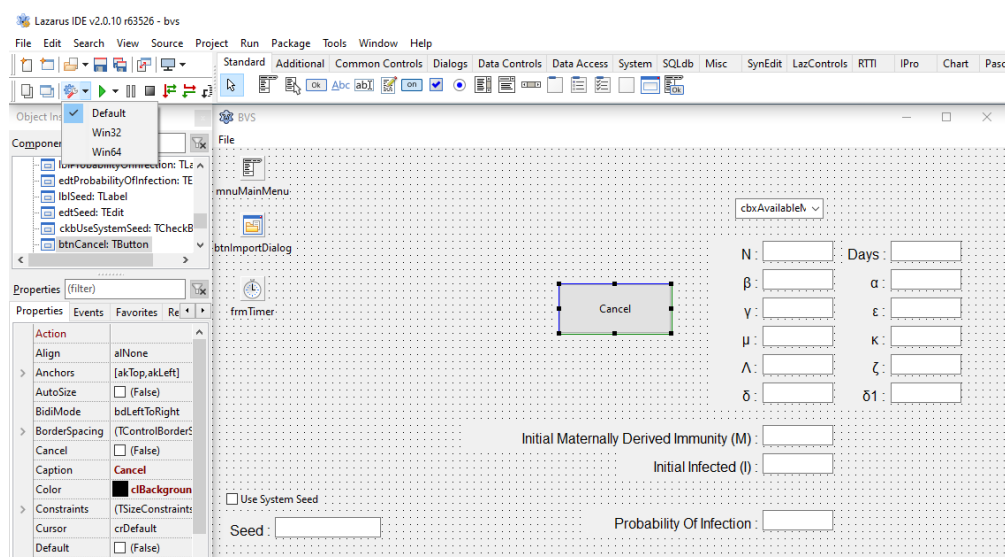
Εικόνα 5.3: Αρχιτεκτονική του εργαλείου BVS

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

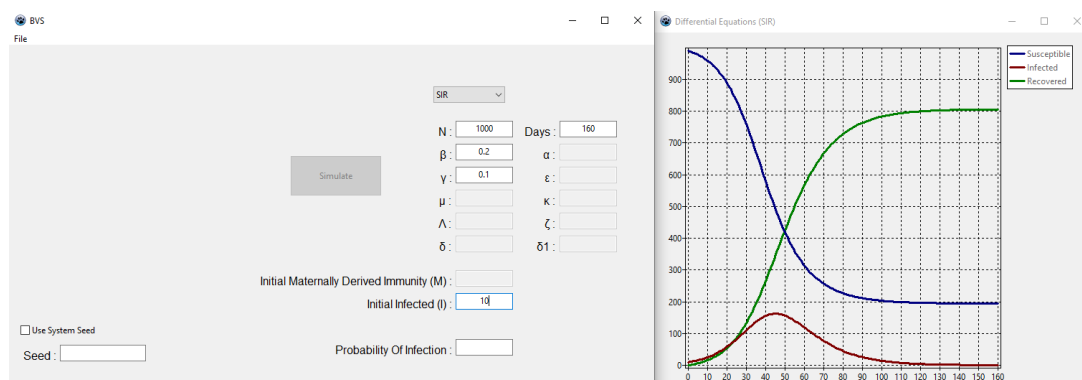
Όπως προαναφέρθηκε και στην ενότητα της αρχιτεκτονικής (Ενότητα 5), ως περιβάλλον ανάπτυξης της εφαρμογής BVS χρησιμοποιήθηκε το Lazarus IDE. Επειδή λοιπόν το Lazarus είναι φτιαγμένο για ταχεία ανάπτυξη εφαρμογών, παρέχει στους εκάστοτε developers αρκετά εργαλεία και λειτουργίες, καθιστώντας την ανάπτυξη 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 ύπαρξη 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

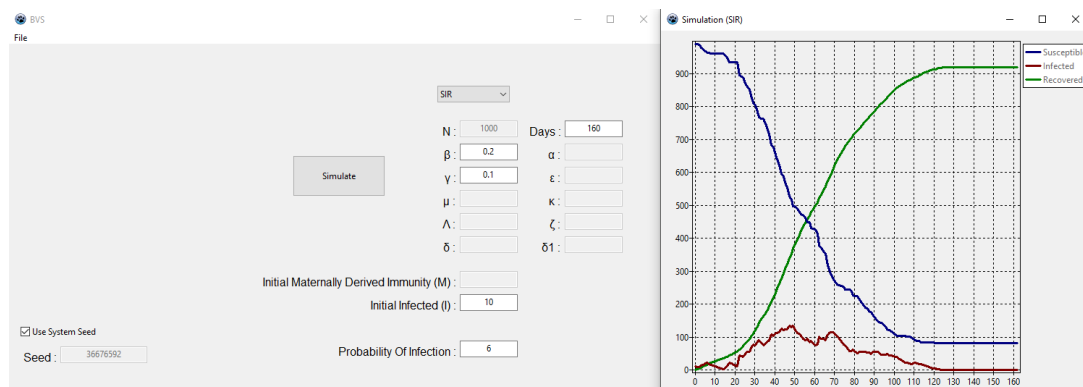
Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής BVS σε συνδυασμό με το Lazarus IDE, δεν ήταν άλλη από την Object Pascal. Η Object Pascal είναι μια αντικειμενοστραφή επέκταση της Pascal που αναπτύχθηκε από την Apple για πρώτη φορά το 1986. Κάποια από τα πιο σημαντικά πλεονεκτήματα 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 επιλεγεί 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 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 εισήχθη 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  $\beta$  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:

- Αρχικοποίηση 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 ή περισσότερων κόμβων.
- Κάθε μέρα της προσομοίωσης και για κάθε κόμβο επιλέγεται ένας τυχαίος αριθμός γειτόνων των κόμβων αυτών και μολύνονται με πιθανότητα  $\beta$ . Όπου γείτονες είναι ίσο με: αριθμός γειτόνων του ενός κόμβου \*  $\beta$ .
- 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

```
1 unit AlgSIR_u;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8   Classes, SysUtils,
9   { Classes }
10  clNode_u,
11  utlTypes_u,
12  utlMisc;
13
14 procedure SIRALG(days, NumberOfInitialInfectedNodes: Word;
15                 beta, gamma: Double;
16                 ProbabilityOfInfection: Byte;
17                 var SamplingResult: TArrayOfArrayOfWord);
18
19 implementation
20 uses
21   { Forms }
22   frmMain_u,
23   frmSimulation_u,
24   frmSettings_u;
25
26 procedure SIRALG(days, NumberOfInitialInfectedNodes: Word;
27                 beta, gamma: Double;
28                 ProbabilityOfInfection: Byte;
29                 var SamplingResult: TArrayOfArrayOfWord);
30 var
31   i, j, k, day: Word;
32   pos: Integer;
33   exitedWhileSimulating: Boolean;
34
35   InitialInfectedNodes, Neighbors, Susceptible, Infected, Recovered:
36     TWordList;
37
38   TestingNode, NodeToBeRecovered, NodesInfectedByNodePerDay,
39   NumberOfNodesInfectedPerDay, NumOfMaxNeighborsToTest,
40   NumOfMaxNodesPerDay: Word;
41
42 begin
43   {
44     <>===== <>
45     || Simulation algorithm for the SIR epidemic model. ||
46     <>===== <>
47
48   [ Algorithm ]
49
50   [*****]
51
52   The algorithm is constructed with the following steps:
53   1) Initialize the Susceptible, Infected and Recovered lists.
54
55   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
[ Variables ]
65
[*****]
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(
113      NumberOfInitialInfectedNodes);
114
115  for i := 0 to NumberOfInitialInfectedNodes - 1 do begin
116      Susceptible.Remove(InitialInfectedNodes[i]);
117      Infected.Add(InitialInfectedNodes[i]);
118  end;
119
120  { Take an initial sampling }
121  SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
122      Count, Recovered.Count));
123
124  for Day := 0 to Days do begin
125
126      if not (Susceptible.Count = 0) and not (Infected.Count = 0) then
127          begin
128              NumberOfNodesInfectedPerDay := 0;
129
130              //writeln('Day: ' + IntToStr(Day)); { Debug }
131
132              for i := 0 to frmMain.Nodes.Count - 1 do begin
133                  NodesInfectedByNodePerDay := 0;
134                  Neighbors := frmMain.Nodes[i].Neighbors;
135
136                  if Neighbors.Count > 0 then begin
137                      NumOfMaxNodesPerDay := Random(Round(Neighbors.Count * beta));
138                      NumOfMaxNeighborsToTest := NumOfMaxNodesPerDay;
139                      if NumOfMaxNeighborsToTest = 0 then NumOfMaxNeighborsToTest
140                          := 1;
141
142                      //writeln('Number of neighbors to be tested by node ' +
143                          IntToStr(i) + ' at day ' + IntToStr(day) + ' are ' + IntToStr(
144                          NumOfMaxNeighborsToTest)); { Debug }
145
146                      for j := 0 to NumOfMaxNeighborsToTest do begin
147                          pos := Random(100);
148                          if pos <= ProbabilityOfInfection then begin
149                              { Pick a random Neighbor }
150                              TestingNode := Neighbors[Random(Neighbors.Count)];
151
152                              if not frmMain.Nodes[TestingNode].IsInfected and
153                                  not frmMain.Nodes[TestingNode].IsRecovered and
154                                  frmMain.Nodes[i].IsInfected then begin
155
156                                  frmSimulation.InfectNode(frmMain.Nodes[TestingNode],

```

```

frmMain.Nodes[i].Id);
151
152     Susceptible.Remove(TestingNode);
153     Infected.Add(TestingNode);
154
155     Inc(NodesInfectedByNodePerDay);
156     //writeln('Node ' + IntToStr(frmMain.Nodes[TestingNode
].Id) + ' has been infected by node ' + IntToStr(frmMain.Nodes[i].Id
)); { Debug }
157         end;
158     end;
159     end; { End j }
160
161     //{ In case the simulation was forced closed before finishing
set the
162     // exitedWhileSimulating flag to true. }
163     //if not frmSimulation.Showing then begin
164     // exitedWhileSimulating := True;
165     // break;
166     //end;
167
168     { If there are no infected nodes stop the simulation. }
169     if Infected.Count = 0 then break;
170
171     NumberOfNodesInfectedPerDay += NodesInfectedByNodePerDay;
172     //writeln('Node: ' + IntToStr(i) + ' infected: ' + IntToStr(
NodesInfectedByNodePerDay) + ' nodes, at day: ' + IntToStr(day)); {
Debug }
173
174     if NumberOfNodesInfectedPerDay >= NumOfMaxNodesPerDay then
break;
175
176     end;
177
178     end; { End i }
179
180     end;
181
182     //writeln('At Day: ' + IntToStr(day) + ', { ' + IntToStr(
NumberOfNodesInfectedPerDay) + ' } where infected. '); { Debug }
183
184     { Recover FIFO (First In First Out) Style }
185     for k := 0 to Round(Infected.Count * gamma) do begin
186         if (Infected.Count > 0) then begin
187             NodeToBeRecovered := Infected[0];
188             frmMain.Nodes[NodeToBeRecovered].Recover;
189
190             Infected.Remove(NodeToBeRecovered);
191             Recovered.Add(NodeToBeRecovered);
192         end
193         else break;
194     end;
195
196     SamplingResult.Add(ArrayOfWord.Create(Susceptible.Count, Infected.
Count, Recovered.Count));
197
198     if frmMain.CancelTriggered then break;
199
200     end; { End Day }
201

```

```

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

```

## 7.2 SIS

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

- Αρχικοποίηση 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 περισσότερων κόμβων.
- Κάθε μέρα της προσομοίωσης και για κάθε κόμβο επιλέγεται ένας τυχαίος αριθμός γειτόνων των κόμβων αυτών και μολύνονται με πιθανότητα  $\rho$ . Όπου γείτονες είναι ίσο με: αριθμός γειτόνων του ενός κόμβου \*  $\beta$ .
- 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

```
1 unit AlgSIS_u;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8   Classes, SysUtils,
9   { Classes }
10  clNode_u,
11  utlTypes_u,
12  utlMisc;
13
14 procedure SISALG(days, NumberOfInitialInfectedNodes: Word;
15                 beta, gamma: Double;
16                 ProbabilityOfInfection: Byte;
17                 var SamplingResult: TArrayOfArrayOfWord);
18
19 implementation
20 uses
21   { Forms }
22   frmMain_u,
23   frmSimulation_u,
24   frmSettings_u;
25
26 procedure SISALG(days, NumberOfInitialInfectedNodes: Word;
27                 beta, gamma: Double;
28                 ProbabilityOfInfection: Byte;
29                 var SamplingResult: TArrayOfArrayOfWord);
30 var
31   i, j, k, day: Word;
```

```

32 pos: Integer;
33 exitedWhileSimulating: Boolean;
34
35 InitialInfectedNodes, Neighbors, Susceptible, Infected: TWordList;
36
37 NodesInfectedByNodePerDay, NumberOfNodesInfectedPerDay,
38   NumOfMaxNeighborsToTest, NumOfMaxNodesPerDay, TestingNode,
39   NodesToBecomeSusceptible, NodeToBecomeSusceptible: Word;
40
41 begin
42   {
43       <>===== <>
44       ||                                     ||
45       ||   Simulation algorithm for the SIS epidemic model.   ||
46       ||                                     ||
47       <>===== <>
48
49   [ Algorithm ]
50
51   [*****]
52   The algorithm is constructed with the following steps:
53   1) Initialize the Susceptible and Infected lists.
54
55   2) Make all nodes Susceptible add them to the Susceptible list.
56
57   3) Initial infect of one or more random node/s.
58
59   4) For each day of the simulation, for each node pick a random
60   number of neighbors and infect them with probability pos. Where
61   neighbors is equal to: number of neighbors of each node * beta.
62
63   5) If the maximum number of nodes that can be infected in one day
64   is reached, go to the next day.
65
66   6) At the end of each day make sure that gamma*I nodes recover
67   from the decease and become susceptible once again.
68
69   [*****]
70
71   [ Variables ]
72
73   [*****]
74
75   i, j, k, day: Counters.
76
77   days: Number of days that the simulation lasts.
78   firstInfected: Index of the first infected node.
79   TestingNode: Index of node to be tested.
80   NodeToBecomeSusceptible: Index of node to be made susceptible
81   again.
82
83   ProbabilityOfInfection: The probability of infection specified
84   by the user.
85   pos: Probability of Infection.
86
87   Neighbors: List with the contact indexes of each node.
88   Susceptible: List that holds the indexes of the susceptible
89   nodes.
90   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);

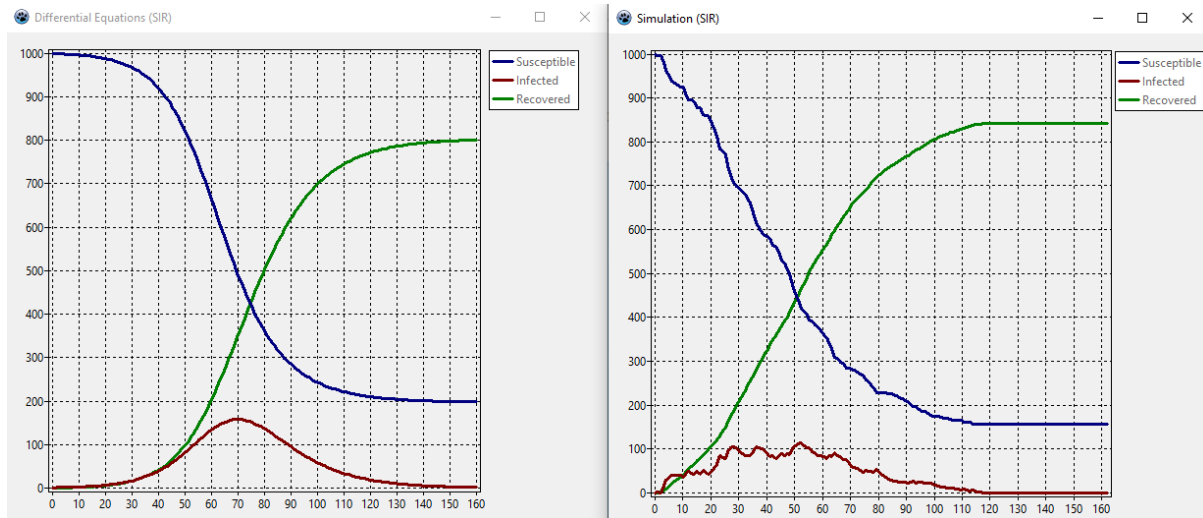
```

```

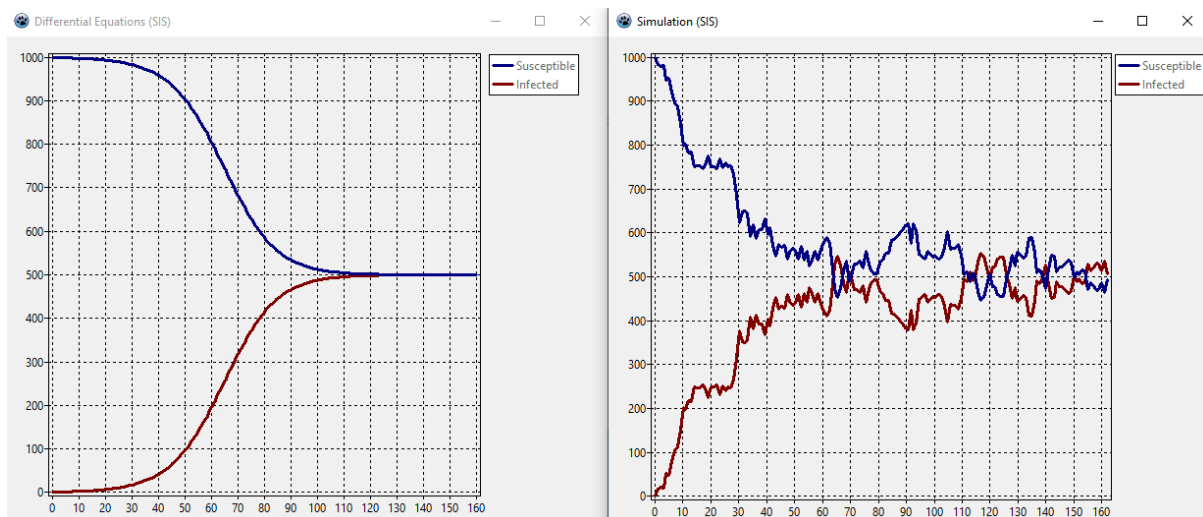
177     if (NodesToBecomeSusceptible > 1) then begin
178         for k := 0 to NodesToBecomeSusceptible do begin
179             if (Infected.Count > 0) then begin
180                 NodeToBecomeSusceptible := Infected[0];
181                 frmMain.Nodes[NodeToBecomeSusceptible].MakeSusceptible;
182
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 διακριτικά μοντέλα. Όσο πιο όμοιες είναι οι δύο γραφικές παραστάσεις, τόσο πιο έγκυρος είναι ένας προσαρμοσμένος αλγόριθμος. [11] [45] [46]

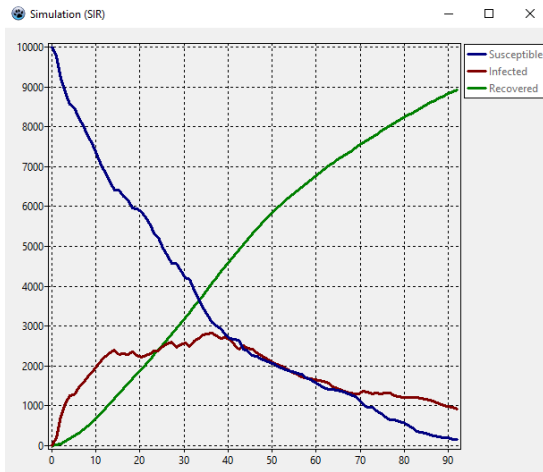


Εικόνα 7.1: BVS | SIR Custom Algorithm validity check, N: 1000,  $\beta$ : 0.2,  $\gamma$ : 0.1, Initial Infected: 1, Days: 160, Seed: 20300081, Probability of Infection: 6%

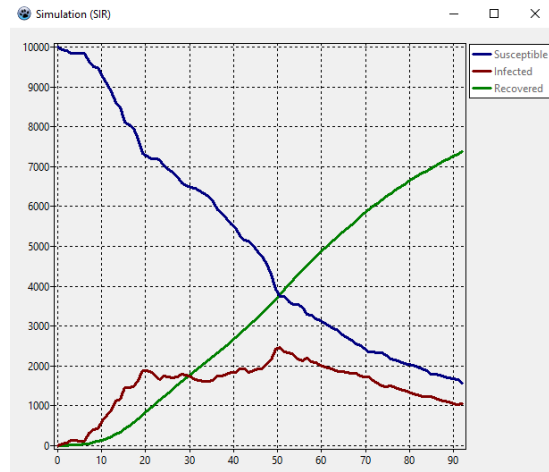


Εικόνα 7.2: BVS | SIS Custom Algorithm validity check, N: 1000,  $\beta$ : 0.2,  $\gamma$ : 0.1, Initial Infected: 1, Days: 160, Seed: 20923765, Probability of Infection: 25%

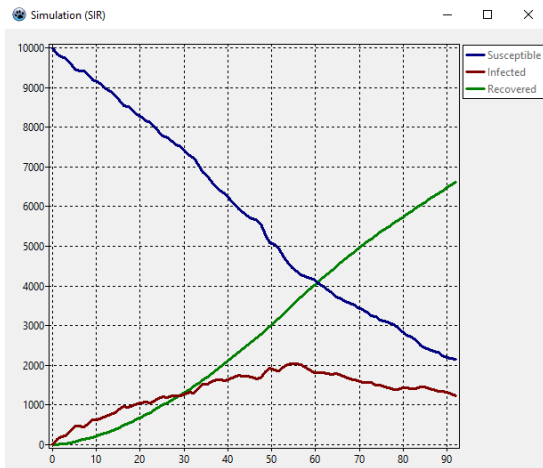
### 7.3 Προσομοίωση σε τυχαίο και ελεύθερης κλίμακας γράφο



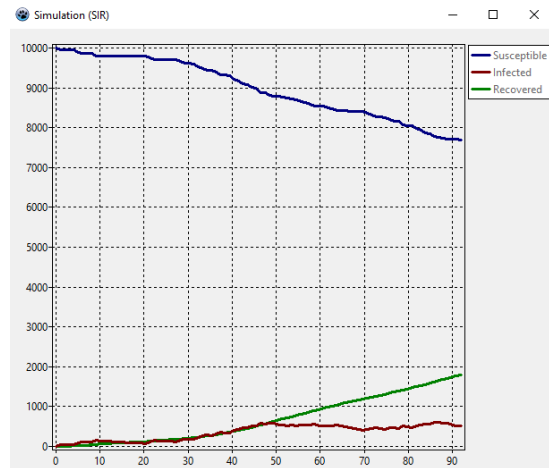
(α) Graph connection probability: 80%



(β) Graph connection probability: 50%



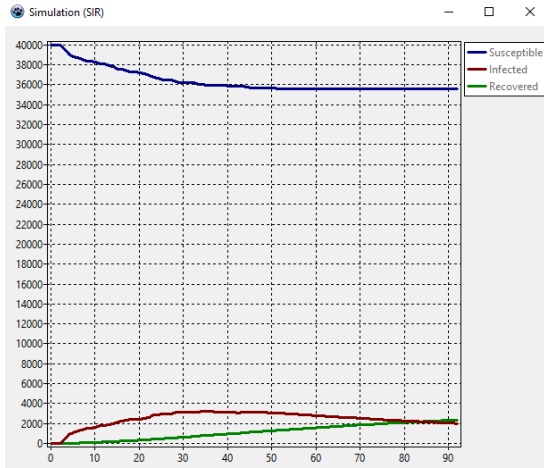
(γ) Graph connection probability: 30%



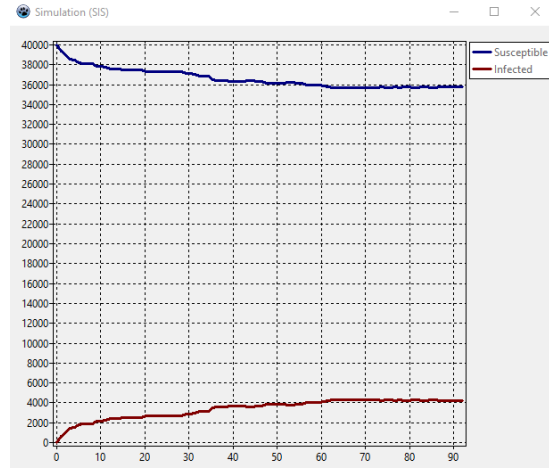
(δ) Graph connection probability: 10%

Εικόνα 7.3: BVS | SIR - ER Graph Simulation,  $N: 10000$ ,  $\beta: 0.05$ ,  $\gamma: 0.05$ , 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.

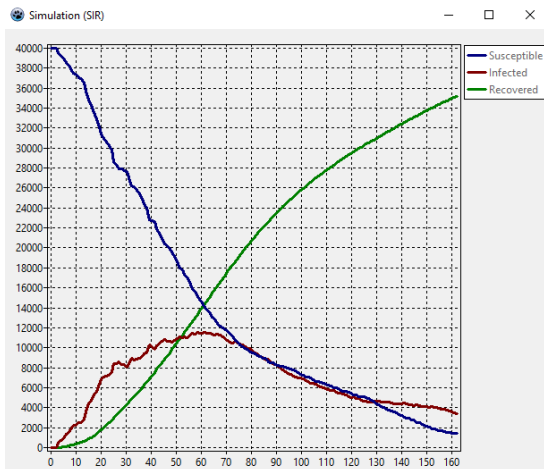


(α) SIR Simulation

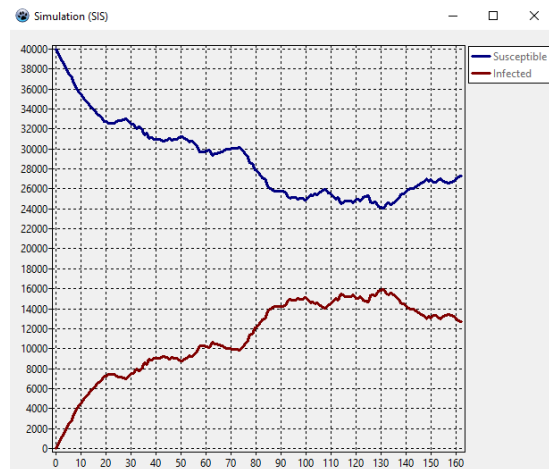


(β) SIS Simulation

Εικόνα 7.4: GRATIS | Scale Free, Adjacency matrix, 40000 Nodes, Initial Connections 1000  
 BVS | SIR, SIS - Scale Free Graph Simulation, N: 40000,  $\beta$ : 0.8,  $\gamma$ : 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,  $\beta$ : 0.031,  $\gamma$ : 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 γνώσεις 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 Epidemiology*, 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 hagggle.“, 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 Dungan, “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 Dungan, “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. Zainulabidin, 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. Siklaidis, 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.

### A.1 Κλάση που αντιπροσωπεύει έναν κόμβο και τα χαρακτηριστικά του

```
1 unit clNode_u;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8   Classes, SysUtils, ExtCtrls, Graphics, fgl;
9
10 type
11   ArrayOfWord = Array of Word;
12   PtrOfTShape = ^TShape;
13   TWordList = specialize TFPGList<Word>;
14
15 type
16   TNode = class(TObject)
17     private
18       FId: Word;
19       FIsSusceptible: Boolean;
20       FIsInfected: Boolean;
21       FIsRecovered: Boolean;
22       FInfectedByNode: Word;
23       FNeighbors: TWordList;
24       FPtrShape: PtrOfTShape;
25     protected
26       { protected declarations here }
27     public
28       constructor Create(
29         AId: Word;
30         ANeighbors: TWordList = Nil;
31         AIsSusceptible: Boolean = false;
32         AIsInfected: Boolean = false;
33         AIsRecovered: Boolean = false);
34
35       destructor Destroy; override;
36
37       property Id: Word read FId write FId;
38       property IsSusceptible: Boolean read FIsSusceptible write
39         FIsSusceptible;
40       property IsInfected: Boolean read FIsInfected write FIsInfected;
41       property IsRecovered: Boolean read FIsRecovered write
42         FIsRecovered;
43       property InfectedByNode: Word read FInfectedByNode write
44         FInfectedByNode;
45       property Neighbors: TWordList read FNeighbors;
46       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:
26     ArrayOfDouble): ArrayOfArrayOfDouble;
27
28 implementation
29 function odeEuler(model: String; t, y0: ArrayOfDouble; extraArgs:
30     ArrayOfDouble): ArrayOfArrayOfDouble;
31 var
32     i, j: Integer;
33     DiffEquations: ArrayOfArrayOfDouble;
34     dXdts, x: ArrayOfDouble;
35 begin
36     {

```

```

37     [ Description ]
38     [*****]
39     Euler method (also called forward Euler method) a first-order
numerical procedure for solving ordinary differential equations (
ODEs) with a given initial value.
40
41     [*****]
42
43     [ Parameters ]
44     [*****]
45     Param t: An array of a specified duration with equal parts.
46     Param y0: The initial system state.
47     Param extraArgs: Array of double containing the appropriate
parameters needed in order to calculate the target differential
equations.
48
49     [*****]
50
51     [ Variables ]
52     [*****]
53     Var i, j: Counters.
54     Var DiffEquations: Array of Arrays containing double numbers that
holds the whole system's state.
55
56     Var dXdts: Calculated differential equation result returned from
a differential equation calculating function.
57
58     Var x: Temporary array that holds the next system's state to be
calculated.
59
60     [*****]
61 }
62
63     SetLength(DiffEquations, Length(y0));
64     SetLength(x, Length(y0));
65
66     for i := 0 to Length(DiffEquations) - 1 do begin
67         SetLength(DiffEquations[i], Length(t));
68         DiffEquations[i][0] := y0[i];
69     end;
70
71     for i := 0 to Length(t) - 1 do begin
72         for j := 0 to Length(DiffEquations) - 1 do begin
73             x[j] := DiffEquations[j][i];
74         end;
75
76         { Invoke the appropriate differential equation calculator
function }
77         case model of
78             SIR: dXdts := SIRDE(x, extraArgs);
79             SIS: dXdts := SISDE(x, extraArgs);
80             SIQ: dXdts := SIQDE(x, extraArgs);
81             SIQS: dXdts := SIQSDE(x, extraArgs);
82             SIQR: dXdts := SIQRDE(x, extraArgs);
83             SIRD: dXdts := SIRDDE(x, extraArgs);
84             MSIR: dXdts := MSIRDE(x, extraArgs);
85             SEIR: dXdts := SEIRDE(x, extraArgs);
86             SEIS: dXdts := SEISDE(x, extraArgs);
87             MSEIR: dXdts := MSEIRDE(x, extraArgs);
88         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     TArrayOfType = Array of TNode;
14
15     TFileHandler = class(TObject)
16     public
17         function IsAdjacencyMaxtrixOrListFile(filename, Atype: String):
Boolean;
18         function LoadAdjacencyMaxtrix(filename: String): TListOfType;
19         function LoadAdjacencyList(filename: String): TListOfType;
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

```

```

35 function TFileHandler.IsAdjacencyMaxtrixOrListFile(filename, Atype:
    String): Boolean;
36 var
37     tfIn: TextFile;
38     s: String;
39     RegexObj: TRegExpr;
40 begin
41     Result := False;
42     AssignFile(tfIn, filename);
43     RegexObj := TRegExpr.Create;
44     if Atype = 'matrix' then begin
45         RegexObj.Expression := ADJACENCY_MATRIX_REGEX;
46     end
47     else
48         RegexObj.Expression := ADJACENCY_LIST_REGEX;
49
50     try
51         try
52             reset(tfIn);
53             if not eof(tfIn) then
54                 begin
55                     readln(tfIn, s);
56                     if RegexObj.Exec(s) then Result := True;
57                 end;
58             except
59                 on E: EInOutError do
60                     writeln('File handling error occurred. Details: ', E.Message);
61                 end;
62
63             finally
64                 CloseFile(tfIn);
65                 RegexObj.Free;
66             end;
67
68     end;
69
70 function TFileHandler.LoadAdjacencyMaxtrix(filename: String):
    TListofTNode;
71 var
72     tfIn: TextFile;
73     s: String;
74     i: Integer;
75     lineLength: Integer;
76     RegexObj: TRegExpr;
77     Nodes: TListofTNode;
78     appender: TAppenderType;
79     obj: TNode;
80     Neighbors: TWordList;
81     c: char;
82    RowIndex, ColumnIndex: Word;
83 begin
84     // Set the name of the file that will be read
85     AssignFile(tfIn, filename);
86
87     // Create a regex to validate file structure
88     RegexObj := TRegExpr.Create;
89     RegexObj.Expression := ADJACENCY_MATRIX_REGEX;
90
91     Nodes := TListofTNode.Create; // Initialize list
92     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.
119               NumberOfEdges);
120               end;
121               Inc(ColumnIndex);
122             end;
123             ColumnIndex := 0;
124             Inc(RowIndex);
125
126             frmMain.AvgNumberOfNeighbors += Neighbors.Count;
127
128             obj := TNode.Create(i, Neighbors);
129             Inc(i);
130             Nodes.Add(obj);
131           end;
132
133           // Keep reading lines until the end of the file is reached
134           while not eof(tfIn) do
135             begin
136               readln(tfIn, s);
137
138               // Validate each line of the file
139               if not RegexObj.Exec(s) or (Length(s) <> lineLength) then
140                 begin
141                   MessageDlg('Error', 'Invalid file format', mtError, [mbOK
142                   ], 0);
143                   Break; // Jump to the 'finally' block
144                 end;
145
146                 Neighbors := TWordList.Create;
147
148                 for c in s do begin
149                   if c = '1' then
150                     begin
151                       Neighbors.Add(ColumnIndex);
152                       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
214 ], 0);
215                 Break; // Jump to the 'finally' block
216             end;
217
218             Neighbors := TWordList.Create;
219
220             colonSplit := s.Split(':');
221             commaSplit := colonSplit[1].Split(',');
222             if not (commaSplit[0] = '') then begin
223                 for neighbor in commaSplit do begin
224                     Neighbors.Add(StrToInt(neighbor));
225                 end;
226             end;
227
228             frmMain.AvgNumberOfNeighbors += Neighbors.Count;
229
230             obj := TNode.Create(StrToInt(colonSplit[0]), Neighbors);
231             Nodes.Add(obj);
232             //writeln('Node: ' + colonSplit[0] + ' has {' + IntToStr(
233 Neighbors.Count) + '} neighbors'); { Debug }
234             end;
235
236             frmMain.AvgNumberOfNeighbors := frmMain.AvgNumberOfNeighbors div
237 Nodes.Count;
238             //writeln('Average number of neighbors: ' + IntToStr(frmMain.
239 AvgNumberOfNeighbors)); { Debug }
240
241         except
242             on E: EInOutError do
243                 writeln('File handling error occurred. Details: ', E.Message);
244             end;
245
246     finally
247         CloseFile(tfIn); // Close the file
248         RegexObj.Free; // Free the regex object
249
250         result := Nodes;
251     end;
252 end;
253
254 procedure TFileHandler.WriteStringToFile(filename: string; data: String
255 );
256 var
257     tfOut: TextFile;
258 begin
259     try
260         AssignFile(tfOut, filename);
261         Rewrite(tfOut);
262         writeln(tfOut, data);
263         closefile(tfOut);
264     except

```

```

261     on E:Exception do
262     end;
263 end;
264
265 procedure TFileHandler.WriteToJsonFile(filename: string; data:
    TJSONData);
266 begin
267     self.WriteStringToFile(filename, data.FormatJSON);
268 end;
269
270 function TFileHandler.LoadJsonFile(filename: string): TJSONData;
271 var
272     tfIn: TextFile;
273     s, JsonString: String;
274 begin
275     s := '';
276     JsonString := '';
277
278     try
279         try
280             AssignFile(tfIn, filename);
281             reset(tfIn);
282
283             while not eof(tfIn) do begin
284                 readln(tfIn, s);
285                 JsonString += s;
286             end;
287
288             Result := GetJSON(JsonString);
289         except
290             on E: EInOutError do
291                 writeln('File handling error occurred. Details: ', E.Message);
292             end;
293         finally
294             CloseFile(tfIn);
295         end;
296
297 end;
298
299 end.

```

## A.4 Διαχείριση ρυθμίσεων εφαρμογής

```

1 unit frmSettings_u;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8     Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
9     fpjson,
10    { Utilities }
11    utlFile_u,
12    utlConstants_u,
13    utlValidation_u;
14
15 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

```

```

69 procedure TfrmSettings.FormCreate(Sender: TObject);
70 begin
71     inherited;
72     self.LoadSettings; // Load Application Settings
73     if self.cbxReSimulate.Checked then self.
74         edtReSimulateMinRecoveredNodeCount.Enabled := True
75     else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
76 end;
77 procedure TfrmSettings.FormClose(Sender: TObject);
78 begin
79     self.LoadSettings;
80     inherited;
81 end;
82
83 procedure TfrmSettings.btnCancelClick(Sender: TObject);
84 begin
85     self.Close;
86 end;
87
88 procedure TfrmSettings.btnApplyClick(Sender: TObject);
89 var
90     JsonObject, settings: TJSONObject;
91 begin
92     { Load the default settings }
93     JsonObject := TJSONObject(GetJSON(DEFAULT_SETTINGS));
94
95     { Get the new settings }
96     settings := JsonObject.Find(SETTINGS_NAME) as TJSONObject;
97     settings.Booleans[RE_SIMULATE_SETTING_NAME] := self.cbxReSimulate.
98         Checked;
99     if self.edtReSimulateMinRecoveredNodeCount.Text <> '' then settings.
100         Strings[RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] :=
101             self.edtReSimulateMinRecoveredNodeCount.Text
102     else settings.Strings[
103         RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] := '0';
104
105     { Write the new settings }
106     FileHandler.WriteToJsonFile(SETTINGS_FILE_NAME, TJSONData(JsonObject)
107         );
108
109     self.LoadSettings; // Load Application Settings
110 end;
111
112 procedure TfrmSettings.cbxReSimulateChange(Sender: TObject);
113 begin
114     if self.cbxReSimulate.Checked then begin
115         self.edtReSimulateMinRecoveredNodeCount.Enabled := True;
116     end
117     else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
118 end;
119
120 procedure TfrmSettings.edtReSimulateMinRecoveredNodeCountKeyPress(
121     Sender: TObject; var Key: char
122 );
123 begin
124     ValidateInteger(Sender, Key);
125 end;
126
127 {$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 = '
21   edtReSimulateMinRecoveredNodeCount';
22 DEFAULT_SETTINGS = '{' + sLineBreak +
23                   '  "' + SETTINGS_NAME + '": {' + sLineBreak +
24                   '    "' + RE_SIMULATE_SETTING_NAME + '": False,' +
25                   '    "' +
26                   RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME + '": 0' +
27                   '  }' + sLineBreak +
28                   '}'';
29
30 { Regex }
31 ADJACENCY_MATRIX_REGEX = '^([0-1]*)$';
32 ADJACENCY_LIST_REGEX = '^(\d+:\d+(,\d+)*$)|(\d+)$';
33
34 implementation
35
36 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,
9   ExtCtrls,
```

```

9  Menus, TAGraph, typinfo, fpjson,
10 { Forms }
11 frmSettings_u,
12 { Utilities }
13 utlFile_u,
14 utlValidation_u,
15 utlEnum_u,
16 utlTypes_u,
17 utlMisc,
18 { Epidemic Algorithms }
19 AlgSIR_u,
20 AlgSIS_u;
21
22 type
23
24 { TfrmMain }
25
26 TfrmMain = class(TForm)
27   btnImportDialog: TOpenDialog;
28   btnSimulate: TButton;
29   btnCancel: TButton;
30   cbxAvailableModels: TComboBox;
31   ckbUseSystemSeed: TCheckBox;
32   edtSeed: TEdit;
33   edtProbabilityOfInfection: TEdit;
34   edtN: TEdit;
35   edtDelta1: TEdit;
36   edtEpsilon: TEdit;
37   edtDays: TEdit;
38   edtBeta: TEdit;
39   edtAlpha: TEdit;
40   edtZeta: TEdit;
41   edtKappa: TEdit;
42   edtGamma: TEdit;
43   edtInitialInfected: TEdit;
44   edtMaternallyDerivedImmunity: TEdit;
45   edtMu: TEdit;
46   edtLambda: TEdit;
47   edtDelta: TEdit;
48   frmTimer: TTimer;
49   lblSeed: TLabel;
50   lblProbabilityOfInfection: TLabel;
51   lblN: TLabel;
52   lblDelta1: TLabel;
53   lblEpsilon: TLabel;
54   lblDays: TLabel;
55   lblBeta: TLabel;
56   lblAlpha: TLabel;
57   lblZeta: TLabel;
58   lblKappa: TLabel;
59   lblGamma: TLabel;
60   lblMaternallyDerivedImmunity: TLabel;
61   lblMu: TLabel;
62   lblInitialInfected: TLabel;
63   lblLambda: TLabel;
64   lblDelta: TLabel;
65   mnuSettings: TMenuItem;
66   mnuFileClose: TMenuItem;
67   mnuMainMenu: TMainMenu;
68   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')
129     then begin
130         Nodes := FileHandler.LoadAdjacencyMaxtrix(filename);
131     else if FileHandler.IsAdjacencyMaxtrixOrListFile(filename, 'list')
132     then begin
133         Nodes := FileHandler.LoadAdjacencyList(filename);
134     else MessageDlg('Error', 'Unsupported file type', mtError, [mbOK],
135     0);
136
137     if self.Nodes.Count > 0 then begin
138         //frmSimulation.RenderShapes;
139         self.edtN.Enabled := False;
140         self.mnuFileClose.Enabled := True;
141     end;
142
143     if self.validateSimulationFields then self.
144     preparePreSimulationChart;
145     end;
146
147     self.cbxAvalableModelsChange(self);
148
149 end;
150
151 procedure TfrmMain.btnSimulateClick(Sender: TObject);
152 begin
153     if self.validateSimulationFields and
154     (self.edtProbabilityOfInfection.Text <> '') then begin
155         self.btnSimulate.Enabled := False;
156         self.btnSimulate.Visible := False;
157         self.btnCancel.Enabled := True;
158         self.btnCancel.Visible := True;
159         self.CancelTriggered := False;
160
161         { Do not allow the close file function to be invocable while
162         simulating }
163         self.mnuFileClose.Enabled := False;
164         //if self.btnSimulate.IsEnabled then begin
165         //    frmSimulation.Show;
166         //end;
167
168         { Randomize System }
169         RandomizeSystem;
170
171         frmSimulation.frmSmlInvoker.OnTimer := @self.InitiateSimulation;
172         frmSimulation.frmSmlInvoker.Enabled := True;
173
174     end;
175 end;
176
177 procedure TfrmMain.InitiateSimulation(Sender: TObject);
178 var
179     SamplingResult: TArrayOfArrayOfWord;
180 begin
181     SamplingResult := TArrayOfArrayOfWord.Create;
182     case self.cbxAvalableModels.Items[self.cbxAvalableModels.ItemIndex]
183     of
184         SIR: begin
185             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.cbxAvaliableModelsChange(Sender: TObject);
235 begin
236     case self.cbxAvaliableModels.Items[self.cbxAvaliableModels.ItemIndex]
237     of
238         SIR: begin
239             self.edtGamma.Enabled := True;
240
241             self.edtMu.Enabled := False;

```

```

241     self.edtLambda.Enabled := False;
242     self.edtDelta.Enabled := False;
243     self.edtMaternallyDerivedImmunity.Enabled := False;
244     self.edtAlpha.Enabled := False;
245     self.edtEpsilon.Enabled := False;
246     self.edtKappa.Enabled := False;
247     self.edtDelta1.Enabled := False;
248     self.edtZeta.Enabled := False;
249 end;
250
251 SIS: begin
252     self.edtGamma.Enabled := True;
253
254     self.edtMu.Enabled := False;
255     self.edtLambda.Enabled := False;
256     self.edtDelta.Enabled := False;
257     self.edtMaternallyDerivedImmunity.Enabled := False;
258     self.edtAlpha.Enabled := False;
259     self.edtEpsilon.Enabled := False;
260     self.edtKappa.Enabled := False;
261     self.edtDelta1.Enabled := False;
262     self.edtZeta.Enabled := False;
263 end;
264
265 SIQ: begin
266     self.edtMu.Enabled := True;
267     self.edtLambda.Enabled := True;
268     self.edtDelta.Enabled := True;
269     self.edtKappa.Enabled := True;
270
271     self.edtGamma.Enabled := False;
272     self.edtMaternallyDerivedImmunity.Enabled := False;
273     self.edtAlpha.Enabled := False;
274     self.edtEpsilon.Enabled := False;
275     self.edtDelta1.Enabled := False;
276     self.edtZeta.Enabled := False;
277 end;
278
279 SIQS, SIQR: begin
280     self.edtGamma.Enabled := True;
281     self.edtMu.Enabled := True;
282     self.edtLambda.Enabled := True;
283     self.edtDelta.Enabled := True;
284     self.edtDelta1.Enabled := True;
285     self.edtKappa.Enabled := True;
286     self.edtZeta.Enabled := True;
287
288     self.edtMaternallyDerivedImmunity.Enabled := False;
289     self.edtAlpha.Enabled := False;
290     self.edtEpsilon.Enabled := False;
291 end;
292
293 SIRD: begin
294     self.edtGamma.Enabled := True;
295     self.edtMu.Enabled := True;
296
297     self.edtLambda.Enabled := False;
298     self.edtDelta.Enabled := False;
299     self.edtMaternallyDerivedImmunity.Enabled := False;
300     self.edtAlpha.Enabled := False;

```

```

301     self.edtEpsilon.Enabled := False;
302     self.edtKappa.Enabled := False;
303     self.edtDelta1.Enabled := False;
304     self.edtZeta.Enabled := False;
305 end;
306
307 MSIR: begin
308     self.edtGamma.Enabled := True;
309     self.edtMu.Enabled := True;
310     self.edtLambda.Enabled := True;
311     self.edtDelta.Enabled := True;
312     self.edtMaternallyDerivedImmunity.Enabled := True;
313
314     self.edtAlpha.Enabled := False;
315     self.edtEpsilon.Enabled := False;
316     self.edtKappa.Enabled := False;
317     self.edtDelta1.Enabled := False;
318     self.edtZeta.Enabled := False;
319 end;
320
321 SEIR: begin
322     self.edtGamma.Enabled := True;
323     self.edtMu.Enabled := True;
324     self.edtLambda.Enabled := True;
325     self.edtAlpha.Enabled := True;
326
327     self.edtDelta.Enabled := False;
328     self.edtMaternallyDerivedImmunity.Enabled := False;
329     self.edtEpsilon.Enabled := False;
330     self.edtKappa.Enabled := False;
331     self.edtDelta1.Enabled := False;
332     self.edtZeta.Enabled := False;
333 end;
334
335 SEIS: begin
336     self.edtGamma.Enabled := True;
337     self.edtMu.Enabled := True;
338     self.edtLambda.Enabled := True;
339     self.edtEpsilon.Enabled := True;
340
341     self.edtDelta.Enabled := False;
342     self.edtMaternallyDerivedImmunity.Enabled := False;
343     self.edtAlpha.Enabled := False;
344     self.edtKappa.Enabled := False;
345     self.edtDelta1.Enabled := False;
346     self.edtZeta.Enabled := False;
347 end;
348
349 MSEIR: begin
350     self.edtGamma.Enabled := True;
351     self.edtMu.Enabled := True;
352     self.edtLambda.Enabled := True;
353     self.edtEpsilon.Enabled := True;
354     self.edtDelta.Enabled := True;
355     self.edtMaternallyDerivedImmunity.Enabled := True;
356
357     self.edtAlpha.Enabled := False;
358     self.edtKappa.Enabled := False;
359     self.edtDelta1.Enabled := False;
360     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.cbxAvalableModels.Items[self.cbxAvalableModels.
371             ItemIndex] of
372             SIR: self.btnSimulate.Enabled := True;
373             SIS: self.btnSimulate.Enabled := True;
374             else
375                 self.btnSimulate.Enabled := False;
376             end;
377         end
378     else
379         self.btnSimulate.Enabled := False;
380     end;
381 end;
382
383 procedure TfrmMain.mnuFileCloseClick(Sender: TObject);
384 begin
385     if frmSimulation.Visible then frmSimulation.Close;
386     if Nodes.Count > 0 then frmSimulation.ResetShapes;
387     self.edtN.Enabled := True;
388     self.mnuFileClose.Enabled := False;
389     self.btnSimulate.Enabled := False;
390 end;
391
392 procedure TfrmMain.mnuSettingsClick(Sender: TObject);
393 begin
394     frmSettings.Show;
395 end;
396
397 procedure TfrmMain.ckbUseSystemSeedChange(Sender: TObject);
398 begin
399     if self.ckbUseSystemSeed.Checked then begin
400         self.edtSeed.Enabled := False;
401     end
402     else self.edtSeed.Enabled := True;
403 end;
404
405 procedure TfrmMain.btnCancelClick(Sender: TObject);
406 begin
407     self.CancelTriggered := True;
408 end;
409
410 procedure TfrmMain.preparePreSimulationChart;
411 begin
412     frmPreSimulationChart.Caption := 'Differential Equations (' + self.
413         cbxAvalableModels.Items[self.cbxAvalableModels.ItemIndex] + ')';
414     frmPreSimulationChart.ClearPreSimulationChart;
415     frmPreSimulationChart.CalculatePreSimulation;
416     frmPreSimulationChart.Show;
417 end;
418
419 procedure TfrmMain.prepareSimulationChart(SamplingResult:
420     TArrayOfArrayOfWork);
421 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.
468         ItemIndex] of
469         SIR, SIS: begin
470           if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
471             AND
472               (self.edtGamma.Text <> '') AND (self.edtInitialInfected.
473               Text <> '')
474             then Result := True;
475         end;
476     end;
477 end;

```

```

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 <> '')

```



```

518 AND
    (self.edtLambda.Text <> '') AND (self.edtEpsilon.Text <>
519 '' ) AND
    (self.edtInitialInfected.Text <> '')
520     then Result := True;
521     end;
522
523 MSEIR: begin
524     if (self.edtDays.Text <> '') AND (self.edtBeta.Text <> '')
525 AND
    (self.edtGamma.Text <> '') AND (self.edtMu.Text <> '')
526 AND
    (self.edtLambda.Text <> '') AND (self.edtEpsilon.Text <>
527 '' ) AND
    (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,
9     TAsSeries,
10    { Forms }
11    frmMain_u,
12    { Classes }
13    { Utilities }
14    utlArray_u,
15    utlTypes_u,
16    utlEuler_u,
17    utlEnum_u,
18    utlConstants_u;
19
20 type
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;

```

```

31     chtPreSimulationS: TLineSeries;
32     procedure CalculatePreSimulation;
33     procedure FormClose(Sender: TObject);
34     procedure ClearPreSimulationChart;
35     procedure PrepareSIR(var y0, extraArgs: ArrayOfDouble);
36     procedure PrepareSIS(var y0, extraArgs: ArrayOfDouble);
37     procedure PrepareSIQ(var y0, extraArgs: ArrayOfDouble);
38     procedure PrepareSIQS(var y0, extraArgs: ArrayOfDouble);
39     procedure PrepareSIQR(var y0, extraArgs: ArrayOfDouble);
40     procedure PrepareSIRD(var y0, extraArgs: ArrayOfDouble);
41     procedure PrepareMSIR(var y0, extraArgs: ArrayOfDouble);
42     procedure PrepareSEIR(var y0, extraArgs: ArrayOfDouble);
43     procedure PrepareSEIS(var y0, extraArgs: ArrayOfDouble);
44     procedure PrepareMSEIR(var y0, extraArgs: ArrayOfDouble);
45 private
46
47 public
48
49     end;
50
51 var
52     frmPreSimulationChart: TfrmPreSimulationChart;
53
54 implementation
55
56 procedure TfrmPreSimulationChart.CalculatePreSimulation;
57 var
58     OdeEulerResult: ArrayOfArrayOfDouble;
59     t, y0, extraArgs: ArrayOfDouble;
60     i, days: Integer;
61     model: String;
62 begin
63     SetLength(y0, 0);
64     SetLength(extraArgs, 0);
65
66     days := StrToInt(frmMain.edtDays.Text);
67     SetLength(t, days);
68     t := linspace(0, days, days);
69
70     model := frmMain.cbxAvalableModels.Items[frmMain.cbxAvalableModels.
71         ItemIndex];
72     { Initiate based on the model }
73     case model of
74         SIR: self.PrepareSIR(y0, extraArgs);
75         SIS: self.PrepareSIS(y0, extraArgs);
76         SIQ: self.PrepareSIQ(y0, extraArgs);
77         SIQS: self.PrepareSIQS(y0, extraArgs);
78         SIQR: self.PrepareSIQR(y0, extraArgs);
79         SIRD: self.PrepareSIRD(y0, extraArgs);
80         MSIR: self.PrepareMSIR(y0, extraArgs);
81         SEIR: self.PrepareSEIR(y0, extraArgs);
82         SEIS: self.PrepareSEIS(y0, extraArgs);
83         MSEIR: self.PrepareMSEIR(y0, extraArgs);
84     end;
85     { Apply Euler to the model's differential equations }
86     OdeEulerResult := odeEuler(model, t, y0, extraArgs);
87
88     { Fill data to the appropriate axes }
89     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       self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
94       I       self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[2][i]); //
95       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      self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
103      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      self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
111      I      self.chtPreSimulationQ.AddXY(t[i], OdeEulerResult[2][i]); //
112      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      self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
120      I      self.chtPreSimulationQ.AddXY(t[i], OdeEulerResult[2][i]); //
121      Q      self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[3][i]); //
122      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      self.chtPreSimulationI.AddXY(t[i], OdeEulerResult[1][i]); //
130      I      self.chtPreSimulationR.AddXY(t[i], OdeEulerResult[2][i]); //
131      R      self.chtPreSimulationD.AddXY(t[i], OdeEulerResult[3][i]); //
132      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 :=
191             False;
192         end;
193     end;
194 end;
195 procedure TfrmPreSimulationChart.PrepareSIR(var y0, extraArgs:
196     ArrayOfDouble);
197 begin
198     SetLength(y0, 3);
199     y0[1] := StrToFloat(frmMain.edtInitialInfected.Text); // I
200     y0[2] := 0; // R
201     y0[0] := frmMain.getN - y0[1] - y0[2]; // S
202
203     SetLength(extraArgs, 3);
204     extraArgs[0] := frmMain.getN; // N
205     extraArgs[1] := StrToFloat(frmMain.edtBeta.Text); // Beta
206     extraArgs[2] := StrToFloat(frmMain.edtGamma.Text); // Gamma
207
208     { Arrange Line Series Color }
209     self.chtPreSimulationS.SeriesColor := clNavy;
210     self.chtPreSimulationI.SeriesColor := clMaroon;
211     self.chtPreSimulationR.SeriesColor := clGreen;
212
213     { Define Line Series Titles }
214     self.chtPreSimulationS.Title := 'Susceptible';
215     self.chtPreSimulationI.Title := 'Infected';
216     self.chtPreSimulationR.Title := 'Recovered';
217
218     { Activate Line Series }
219     self.chtPreSimulationS.Active := True;
220     self.chtPreSimulationI.Active := True;
221     self.chtPreSimulationR.Active := True;
222 end;
223 procedure TfrmPreSimulationChart.PrepareSIS(var y0, extraArgs:
224     ArrayOfDouble);
225 begin
226     SetLength(y0, 2);
227     y0[1] := StrToFloat(frmMain.edtInitialInfected.Text); // I
228     y0[0] := frmMain.getN - y0[1]; // S
229
230     SetLength(extraArgs, 3);
231     extraArgs[0] := frmMain.getN; // N
232     extraArgs[1] := StrToFloat(frmMain.edtBeta.Text); // Beta
233     extraArgs[2] := StrToFloat(frmMain.edtGamma.Text); // Gamma
234
235     { 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;

```

```

521 self.chtPreSimulationE.Active := True;
522 self.chtPreSimulationI.Active := True;
523 self.chtPreSimulationR.Active := True;
524 end;
525
526 procedure TfrmPreSimulationChart.FormClose(Sender: TObject);
527 begin
528     self.ClearPreSimulationChart;
529 end;
530
531 {$R *.lfm}
532
533 end.

```

### A.6.3 Γραφική παράσταση προσομοίωσης

```

1 unit frmSimulationChart_u;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8     Classes, SysUtils, Forms, Controls, Graphics, Dialogs, TAGraph,
9     TAsSeries,
10    { Forms }
11    frmMain_u,
12    { Classes }
13    { Utilities }
14    utlArray_u,
15    utlTypes_u,
16    utlEnum_u,
17    utlConstants_u;
18
19 type
20     { TfrmSimulationChart }
21
22     TfrmSimulationChart = class(TForm)
23     chtSimulation: TChart;
24     chtSimulationI: TLineSeries;
25     chtSimulationD: TLineSeries;
26     chtSimulationE: TLineSeries;
27     chtSimulationQ: TLineSeries;
28     chtSimulationM: TLineSeries;
29     chtSimulationR: TLineSeries;
30     chtSimulationS: TLineSeries;
31     procedure CalculateSimulation(SamplingResult: TArrayOfArrayOfWord);
32     procedure FormClose(Sender: TObject);
33     procedure ClearSimulationChart;
34     procedure FormResize(Sender: TObject);
35     procedure PrepareSIR;
36     procedure PrepareSIS;
37     procedure PrepareSIQ;
38     procedure PrepareSIQS;
39     procedure PrepareSIQR;
40     procedure PrepareSIRD;
41     procedure PrepareMSIR;
42     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:
57     TArrayofArrayofWord);
58 var
59     t: ArrayofDouble;
60     i, days: Integer;
61     model: String;
62 begin
63     days := StrToInt(frmMain.edtDays.Text) + 2;
64     SetLength(t, days);
65     t := linspace(0, days, days);
66
67     model := frmMain.cbxAvalableModels.Items[frmMain.cbxAvalableModels.
68         ItemIndex];
69     { Initiate based on the model }
70     case model of
71         SIR: self.PrepareSIR;
72         SIS: self.PrepareSIS;
73         SIQ: self.PrepareSIQ;
74         SIQS: self.PrepareSIQS;
75         SIQR: self.PrepareSIQR;
76         SIRD: self.PrepareSIRD;
77         MSIR: self.PrepareMSIR;
78         SEIR: self.PrepareSEIR;
79         SEIS: self.PrepareSEIS;
80         MSEIR: self.PrepareMSEIR;
81     end;
82
83     { Fill data to the appropriate axes }
84     case model of
85         SIR: begin
86             for i := 0 to days - 1 do
87                 begin
88                     self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
89                     self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
90                     self.chtSimulationR.AddXY(t[i], SamplingResult[i][2]); // R
91                 end;
92             end;
93
94         SIS: begin
95             for i := 0 to days - 1 do
96                 begin
97                     self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
98                     self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
99                 end;
100            end;
101
102         SIQ, SIQS: begin

```

```

101     for i := 0 to days - 1 do
102         begin
103             self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
104             self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
105             self.chtSimulationQ.AddXY(t[i], SamplingResult[i][2]); // Q
106         end;
107     end;
108
109 SIQR: begin
110     for i := 0 to days - 1 do
111         begin
112             self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
113             self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
114             self.chtSimulationQ.AddXY(t[i], SamplingResult[i][2]); // Q
115             self.chtSimulationR.AddXY(t[i], SamplingResult[i][3]); // R
116         end;
117     end;
118
119 SIRD: begin
120     for i := 0 to days - 1 do
121         begin
122             self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
123             self.chtSimulationI.AddXY(t[i], SamplingResult[i][1]); // I
124             self.chtSimulationR.AddXY(t[i], SamplingResult[i][2]); // R
125             self.chtSimulationD.AddXY(t[i], SamplingResult[i][3]); // D
126         end;
127     end;
128
129 MSIR: begin
130     for i := 0 to days - 1 do
131         begin
132             self.chtSimulationM.AddXY(t[i], SamplingResult[i][0]); // M
133             self.chtSimulationS.AddXY(t[i], SamplingResult[i][1]); // S
134             self.chtSimulationI.AddXY(t[i], SamplingResult[i][2]); // I
135             self.chtSimulationR.AddXY(t[i], SamplingResult[i][3]); // R
136         end;
137     end;
138
139 SEIR: begin
140     for i := 0 to days - 1 do
141         begin
142             self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
143             self.chtSimulationE.AddXY(t[i], SamplingResult[i][1]); // E
144             self.chtSimulationI.AddXY(t[i], SamplingResult[i][2]); // I
145             self.chtSimulationR.AddXY(t[i], SamplingResult[i][3]); // R
146         end;
147     end;
148
149 SEIS: begin
150     for i := 0 to days - 1 do
151         begin
152             self.chtSimulationS.AddXY(t[i], SamplingResult[i][0]); // S
153             self.chtSimulationE.AddXY(t[i], SamplingResult[i][1]); // E
154             self.chtSimulationI.AddXY(t[i], SamplingResult[i][2]); // I
155         end;
156     end;
157
158 MSEIR: begin
159     for i := 0 to days - 1 do
160         begin

```

```

161         self.chtSimulationM.AddXY(t[i], SamplingResult[i][0]); // M
162         self.chtSimulationS.AddXY(t[i], SamplingResult[i][1]); // S
163         self.chtSimulationE.AddXY(t[i], SamplingResult[i][2]); // E
164         self.chtSimulationI.AddXY(t[i], SamplingResult[i][3]); // I
165         self.chtSimulationR.AddXY(t[i], SamplingResult[i][4]); // R
166     end;
167 end;
168
169 end;
170
171 self.chtSimulation.Visible := true;
172 end;
173
174 procedure TfrmSimulationChart.ClearSimulationChart;
175 var
176     i: Integer;
177 begin
178     for i := 0 to self.chtSimulation.SeriesCount - 1 do
179         begin
180             if self.chtSimulation.Series[i] is TLineSeries then
181                 begin
182                     (self.chtSimulation.Series[i] as TLineSeries).Clear;
183                     (self.chtSimulation.Series[i] as TLineSeries).Active := False;
184                 end;
185             end;
186         end;
187
188 procedure TfrmSimulationChart.FormResize(Sender: TObject);
189 begin
190     self.chtSimulation.Height := self.Height * 97 div 100;
191     self.chtSimulation.Width := self.Width * 99 div 100;
192 end;
193
194 procedure TfrmSimulationChart.PrepareSIR;
195 begin
196     { Arrange Line Series Color }
197     self.chtSimulationS.SeriesColor := clNavy;
198     self.chtSimulationI.SeriesColor := clMaroon;
199     self.chtSimulationR.SeriesColor := clGreen;
200
201     { Define Line Series Titles }
202     self.chtSimulationS.Title := 'Susceptible';
203     self.chtSimulationI.Title := 'Infected';
204     self.chtSimulationR.Title := 'Recovered';
205
206     { Activate Line Series }
207     self.chtSimulationS.Active := True;
208     self.chtSimulationI.Active := True;
209     self.chtSimulationR.Active := True;
210 end;
211
212 procedure TfrmSimulationChart.PrepareSIS;
213 begin
214     { Arrange Line Series Color }
215     self.chtSimulationS.SeriesColor := clNavy;
216     self.chtSimulationI.SeriesColor := clMaroon;
217
218     { Define Line Series Titles }
219     self.chtSimulationS.Title := 'Susceptible';
220     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;

```

```

281     self.chtSimulationR.Active := True;
282 end;
283
284 procedure TfrmSimulationChart.PrepareSIRD;
285 begin
286     { Arrange Line Series Color }
287     self.chtSimulationS.SeriesColor := clNavy;
288     self.chtSimulationI.SeriesColor := clOlive;
289     self.chtSimulationR.SeriesColor := clGreen;
290     self.chtSimulationD.SeriesColor := clMaroon;
291
292     { Define Line Series Titles }
293     self.chtSimulationS.Title := 'Susceptible';
294     self.chtSimulationI.Title := 'Infected';
295     self.chtSimulationR.Title := 'Recovered';
296     self.chtSimulationD.Title := 'Deceased';
297
298     { Activate Line Series }
299     self.chtSimulationS.Active := True;
300     self.chtSimulationI.Active := True;
301     self.chtSimulationR.Active := True;
302     self.chtSimulationD.Active := True;
303 end;
304
305 procedure TfrmSimulationChart.PrepareMSIR;
306 begin
307     { Arrange Line Series Color }
308     self.chtSimulationM.SeriesColor := clFuchsia;
309     self.chtSimulationS.SeriesColor := clNavy;
310     self.chtSimulationI.SeriesColor := clOlive;
311     self.chtSimulationR.SeriesColor := clGreen;
312
313     { Define Line Series Titles }
314     self.chtSimulationM.Title := 'Maternally' + sLineBreak + 'derived' +
315         sLineBreak + 'immunity';
316     self.chtSimulationS.Title := ' ' + sLineBreak + 'Susceptible';
317     self.chtSimulationI.Title := ' ' + sLineBreak + 'Infected';
318     self.chtSimulationR.Title := ' ' + sLineBreak + 'Recovered';
319
320     { Activate Line Series }
321     self.chtSimulationM.Active := True;
322     self.chtSimulationS.Active := True;
323     self.chtSimulationI.Active := True;
324     self.chtSimulationR.Active := True;
325 end;
326
327 procedure TfrmSimulationChart.PrepareSEIR;
328 begin
329     { Arrange Line Series Color }
330     self.chtSimulationS.SeriesColor := clNavy;
331     self.chtSimulationE.SeriesColor := clOlive;
332     self.chtSimulationI.SeriesColor := clMaroon;
333     self.chtSimulationR.SeriesColor := clGreen;
334
335     { Define Line Series Titles }
336     self.chtSimulationS.Title := 'Susceptible';
337     self.chtSimulationE.Title := 'Exposed';
338     self.chtSimulationI.Title := 'Infected';
339     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

```
1 unit frmSettings_u;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8   Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,
9   fpjson,
10  { Utilities }
11  utlFile_u,
12  utlConstants_u,
13  utlValidation_u;
14
15 type
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:
30     TObject; var Key: char);
31     procedure FormClose(Sender: TObject);
32     procedure FormCreate(Sender: TObject);
33     procedure LoadSettings;
34 private
35
36 public
37
38   end;
39
40 var
41   frmSettings: TfrmSettings;
42
43 implementation
44
45 procedure TfrmSettings.LoadSettings;
46 var
47   i, j: Integer;
48   JsonData: TJSONData;
49   JsonObject, settings: TJSONObject;
50 begin
51   { Generate default settings if they do not exist }
52   if not FileExists(SETTINGS_FILE_NAME) then FileHandler.
53     WriteToJsonFile(SETTINGS_FILE_NAME, GetJSON(DEFAULT_SETTINGS));
54   { Load the settings }
55   try
56     JsonData := FileHandler.LoadJsonFile(SETTINGS_FILE_NAME);
57     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
69 procedure TfrmSettings.FormCreate(Sender: TObject);
70 begin
71     inherited;
72     self.LoadSettings; // Load Application Settings
73     if self.cbxReSimulate.Checked then self.
edtReSimulateMinRecoveredNodeCount.Enabled := True
74     else self.edtReSimulateMinRecoveredNodeCount.Enabled := False;
75 end;
76
77 procedure TfrmSettings.FormClose(Sender: TObject);
78 begin
79     self.LoadSettings;
80     inherited;
81 end;
82
83 procedure TfrmSettings.btnCancelClick(Sender: TObject);
84 begin
85     self.Close;
86 end;
87
88 procedure TfrmSettings.btnApplyClick(Sender: TObject);
89 var
90     JsonObject, settings: TJSONObject;
91 begin
92     { Load the default settings }
93     JsonObject := TJSONObject(GetJSON(DEFAULT_SETTINGS));
94
95     { Get the new settings }
96     settings := JsonObject.Find(SETTINGS_NAME) as TJSONObject;
97     settings.Booleans[RE_SIMULATE_SETTING_NAME] := self.cbxReSimulate.
Checked;
98     if self.edtReSimulateMinRecoveredNodeCount.Text <> '' then settings.
Strings[RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] :=
self.edtReSimulateMinRecoveredNodeCount.Text
99     else settings.Strings[
RE_SIMULATE_MINIMUM_RECOVERED_NODE_COUNT_SETTING_NAME] := '0';
100
101     { Write the new settings }
102     FileHandler.WriteToJsonFile(SETTINGS_FILE_NAME, TJSONData(JsonObject)
);
103
104     self.LoadSettings; // Load Application Settings
105 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(
116     Sender: TObject; var Key: char
117 );
118 begin
119     ValidateInteger(Sender, Key);
120 end;
121 {$R *.lfm}
122
123 end.
```