ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ Κ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Προσομοίωση Λειτουργίας Αεροσταθμού με Μοντέλο 2-1-2»

Φοιτήτρια: Κονταξάκη Ειρήνη

Επ.Καθηγητής: Χριστοδούλου Εμμανουήλ

Χανιά, Σεπτέμβριος 2004

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή κ.Μανόλη Χριστοδούλου για τη συνεχή του καθοδήγηση και επίβλεψη κατά τη διάρκεια εκπόνησης της Διπλωματικής μου Εργασίας.

Νιώθω επίσης την ανάγκη να εκφράσω τις ευχαριστίες μου σε όλους τους φίλους μου και την οικογένεια μου για την υποστήριξη τους, πρακτικά και ψυχολογικά. Ξεχωριστά θέλω να ευχαριστήσω την καλή μου φίλη Μαρία, που είναι δίπλα μου σε δύσκολες και μη στιγμές μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ **4**

ΚΕΦΑΛΑΙΟ 2. ΠΡΟΣΟΜΟΙΩΣΗ ΣΥΣΤΗΜΑΤΩΝ **7**

2.1 Γενικά *7*

2.2 Η έννοια του συστήματος *7*

2.3 Μοντέλο προσομοίωσης *10*

2.4 Μηχανισμός εξέλιξης χρόνου σε προσομοίωση διακριτών γεγονότων *14*

2.5 Βήματα μελέτης προσομοίωσης *17*

2.6 Πλεονεκτήματα και μειονεκτήματα της προσομοίωσης *18*

2.7 Προσομοίωση ενός συστήματος αναμονής – Ουρές αναμονής *20*

2.8 Κατανομή Poisson *23*

2.9 Ανάπτυξη προγραμμάτων προσομοίωσης *25*

ΚΕΦΑΛΑΙΟ 3. ΦΥΣΙΚΟ ΠΡΟΒΛΗΜΑ **30**

3.1 Στοιχεία λειτουργίας αεροδρομίων *30*

3.2 Περιγραφή προβλήματος *40*

3.3 Ανασκόπηση μεθόδων-μοντέλων προσέγγισης *45*

ΚΕΦΑΛΑΙΟ 4. ΜΟΝΤΕΛΟΠΟΙΗΣΗ **48**

4.1 Περιγραφή μοντέλου *48*

4.2 Υλοποίηση προσομοίωσης *50*

4.3 Αρχικές τιμές & δεδομένα *61*

4.4 Αποτελέσματα *62*

ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ **80**

ΠΑΡΑΡΤΗΜΑ Α. ΚΩΔΙΚΕΣ ΕΠΙΛΥΣΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ **81**

Α.1 Κώδικας μοντέλου με σειριακό έλεγχο (ΚΩΔΙΚΑΣ Α) *82*

Α.2 Κώδικας μοντέλου με έλεγχο βάσει κριτήριο κόστους (ΚΩΔΙΚΑ Β) *96*

ΒΙΒΛΙΟΓΡΑΦΙΑ **112**

ΚΕΦΑΛΑΙΟ 1. Εισαγωγή

Η μεταφορά των ανθρώπων και των εμπορευμάτων έχει εξελιχθεί σε μια επιτακτική ανάγκη για κάθε κοινωνία ανθρώπων, ανεξάρτητα από το μέγεθος ή το βαθμό της αναπτύξεως. Όσο ανεβαίνει το τεχνολογικό και οικονομικό επίπεδο μιας κοινωνίας τόσο μεγαλύτερη είναι η ανάγκη σε μεταφορές. Η μηχανοποίηση στον τομέα μεταφορών ήταν αναπόφευκτη εξέλιξη και όπως είναι αυτονόητο βρίσκεται σε άμεση σχέση με το τεχνολογικό επίπεδο.

Η μεταφορά με μηχανικά μέσα έχει αναπτυχθεί κατά τρόπο που να επιτρέπει τη κίνηση των μεταφορικών μέσων στη γη, στη θάλασσα και στον αέρα. Σε αντιδιαστολή με τις αεροπορικές μεταφορές οι λοιπές χαρακτηρίζονται ως επιφανειακές.

Η αεροπορική μεταφορά έχει γίνει το κύριο σύγχρονο μέσο μεταφοράς. Η μέγιστη απόδοση και αξία επιτυγχάνονται όταν διανύονται μεγάλες αποστάσεις, μετακινούνται εμπορεύματα υψηλής αξίας, όταν πρέπει να αντιμετωπισθούν άμεσες ανάγκες ή όταν η τοπολογία του εδάφους εμποδίζει την εύκολη μετακίνηση ή αυξάνει σημαντικά το κόστος μεταφοράς με επίγεια μέσα μεταφοράς.

Αν και η αερομεταφορά έχει μεγαλύτερη απόδοση σε χρόνο και κόστος όσο μεγαλύτερη είναι η διανυόμενη απόσταση, ωστόσο, αξίζει ακόμα και για σχετικά μικρές αποστάσεις. Επίσης, η αεροπορική μεταφορά εξασφαλίζει την επικοινωνία ή την ιατρική διασύνδεση μεταξύ ομάδών ατόμων σε απομακρυσμένες και δύσβατες περιοχές, γεγονός το οποίο μερικές φορές αποδεικνύεται ιδιαίτερα κρίσιμο.

Οι αεροπορικές μεταφορές αποτελούν οικονομικά αναπτυσσόμενη βιομηχανία. Η επιβατηγός κυκλοφορία (εκφρασμένη ως προσοδοφόρα επιβατο-χιλιόμετρα) έχει αναπτυχθεί από το 1960 σε ποσοστό περίπου 9% ανά έτος. Οι μεταφορές φορτίων, 80% των οποίων πραγματοποιούνται από τα επιβατικά αεροσκάφη, έχουν επίσης αναπτυχθεί στην ίδια χρονική περίοδο [10]. Συνέπεια αυτής

της οικονομικής δυναμικής είναι ο βιομηχανικός αυτός κλάδος να έχει μεγάλη σημασία για την παγκόσμια οικονομία.

Ωστόσο, η βιομηχανία των αεροπορικών μεταφορών αναπτύσσεται με ρυθμούς ταχύτερους από τους τρέχοντες ρυθμούς με τους οποίους παράγονται και καθιερώνονται τα τεχνολογικά και επιχειρησιακά βήματα αλματώδους προόδου που θα βελτιστοποιήσουν τις διεργασίες και τους χρόνους εξυπηρέτησης των αεροσκαφών εντός των αεροδρομίων.[9]

Εκτιμάται ότι ο αριθμός των πτήσεων στον εναέριο χώρο της Ευρωπαϊκής Ένωσης θα υπερδιπλασιαστεί μέχρι το 2015. Ο υπάρχων οργανισμός Διαχείρισης Εναέριας Κυκλοφορίας (Air Traffic Management) δεν θα είναι σε θέση να αντεπεξέλθει στις ολοένα αυξανόμενες απαιτήσεις.

Οι μελλοντικές μέθοδοι και επιχειρήσεις Διαχείρισης Εναέριας Κυκλοφορίας θα πρέπει να διαθέτουν μεγαλύτερη ευελιξία προκειμένου να αυξήσουν την ικανότητα δρομολογίων, να μειώσουν τις καθυστερήσεις με στόχο την μεγαλύτερη ικανότητα υιοθετούν πορείες προσέγγισης που προτιμούνται από τους χρήστες.

Χρειάζονται νέες δυνατότητες να υποστηρίξουν ευέλικτες αλλαγές πορειών, καθώς επίσης δυναμικές τροποποιήσεις στις εναέριες διαδρομές ούτως ώστε να ανταποκρίνονται στις μεταβολές τόσο των καιρικών όσο και των κυκλοφοριακών συνθηκών. Θα χρειαστεί η ενοποίηση των συναρτήσεων Διαχείρισης Εναέριας Κυκλοφορίας (ATM functions) που αφορούν στον τερματικό σταθμό με αυτές που αφορούν στο δρομολόγιο, προκειμένου να επιτευχθεί ομαλή κυκλοφοριακή ροή εντός και εκτός των περιοχών των αεροσταθμών.

Η χρήση Μεθόδων Προηγμένης Τεχνολογίας Πληροφορίας θεωρείται θα αποτελέσει βασικό κλειδί στην εξέλιξη των μελλοντικών επιχειρήσεων Διαχείρισης Εναέριας Κυκλοφορίας. Η χρήση αυτών των τεχνολογιών θα αποτελέσει τη βάση για την ανάπτυξη εργαλείων που θα παρέχουν αποτελεσματική βοήθεια στην (multilevel) διαδικασία αποφάσεων και που θα συμπεριλαμβάνουν πιλότους, ελεγκτές εδάφους, αερογραμμές κ.α. στο μελλοντικό περιβάλλον Διαχείρισης Εναέριας Κυκλοφορίας. Πιο συγκεκριμένα θα χρησιμοποιηθούν οι πιο εξελιγμένες επιστημονικά τεχνολογίες

πληροφορικής από το πεδίο των επικοινωνιών, των δικτύων, της βελτιστοποίησης και του βέλτιστου ελέγχου.

Η εφαρμογή αυτών των τεχνολογιών σε προβλήματα Διαχείρισης Εναέριας Κυκλοφορίας, αναπτύχθηκε πολύ πρόσφατα, με αφετηρία τις Η.Π.Α. Μπορούμε να αναφέρουμε την εφαρμογή τεχνολογιών βέλτιστου ελέγχου προκειμένου να αντιμετωπιστούν προβλήματα όπως: ανάλυση κυκλοφορίας πολλών αεροσκαφών, αποφυγή σύγκρουσης αεροσκαφών με διασταυρούμενες πορείες πτήσης και μοντελοποίηση και βέλτιστο έλεγχο των διεργασιών εξυπηρέτησης εδάφους. Τεχνικές ανάλυσης υβριδικών συστημάτων και θεωρίας παιγνίων πρόσφατα εφαρμόστηκαν σε προβλήματα ελαχιστοποίησης της απόστασης των κοντινών αεροσκαφών. [3]

Στην παρούσα εργασία δίνεται στην αρχή μια περιγραφή του τρόπου μοντελοποίησης και προσομοίωσης συστημάτων καθώς και τις ιδιαιτερότητες του κάθε τρόπου. Έπειτα δίνεται μια σύντομη περιγραφή των αεροδρομίων και των στοιχείων λειτουργίας τους για να εξοικειωθεί ο αναγνώστης με την ορολογία, την χωροταξική τοποθέτηση των εγκαταστάσεων καθώς και τις λειτουργίες που λαμβάνουν χώρα σε ένα αεροδρόμιο αναφορικά με την εξυπηρέτηση των αεροσκαφών. Κατόπιν, περιγράφεται το πρόβλημα της αυξανόμενης κυκλοφορίας των αεροσκαφών που μας οδηγεί στην παρουσίαση των κυριοτέρων μοντέλων ελέγχου και βελτιστοποίησης των διεργασιών εξυπηρέτησης των αεροσκαφών σε ένα αεροδρόμιο.

Στην συνέχεια αναπτύσσουμε ένα αριθμητικό μοντέλο προσομοίωσης της εξυπηρέτησης αεροσκαφών σε ένα αεροδρόμιο, που είναι και ο αντικειμενικός σκοπός της παρούσας διπλωματικής. Συγκεκριμένα, τα στοιχεία του αεροδρομίου μοντελοποιούνται ως μηχανές (machines), υπομηχανές (submachines) και ενδιάμεσοι χώροι αποθήκευσης ή στάσης (buffers) με αλληλεπιδράσεις.

Τα αριθμητικά αποτελέσματα για δύο διαφορετικούς τρόπους ελέγχου του μοντέλου καθώς και για διαφορετικές τιμές ωριαίου ρυθμού προσγειώσεων / απογειώσεων καταδεικνύουν τις περιπτώσεις που η συγκεκριμένη διάταξη του αεροδρομίου εμφανίζει μεγάλες καθυστερήσεις στην εξυπηρέτηση των αεροσκαφών και την δημιουργία του αδιαχώρητου.

ΚΕΦΑΛΑΙΟ 2. Προσομοίωση Συστημάτων

2.1 Γενικά

Η έννοια της μοντελοποίησης, στην οποία βασίζεται η προσομοίωση, χρησιμοποιείται εδώ και πολλά χρόνια. Τα τελευταία όμως χρόνια παράλληλα με την ανάπτυξη των υπολογιστών είχαμε και την χρήση της προσομοίωσης σε πάρα πολλούς τομείς. Σε γενικές γραμμές προσομοίωση είναι η διαδικασία σχεδιασμού του μοντέλου ενός συστήματος και η διεξαγωγή πειραμάτων με το μοντέλο αυτό που σκοπό έχουν είτε την κατανόηση της συμπεριφοράς του συστήματος ή την δοκιμή διάφορων στρατηγικών για την λειτουργία του συστήματος. Προσφέρει στον αναλυτή, χειριστή, σχεδιαστή, ή μελετητή τη δυνατότητα να ελέγχει ιδέες και συλλήψεις σχεδιασμού ή να αλληλεπιδρά με ολόκληρα συστήματα όσον αφορά τη λειτουργία, την απόδοση και τα χαρακτηριστικά τους χωρίς να χρειαστεί να επέμβει στο πραγματικό σύστημα.

Στις επόμενες παραγράφους, γίνεται εκτενής αναφορά σε βασικές έννοιες και ορολογία για την προσομοίωση συστημάτων βάση και της βιβλιογραφίας [12], [13], [14], [15] και [16].

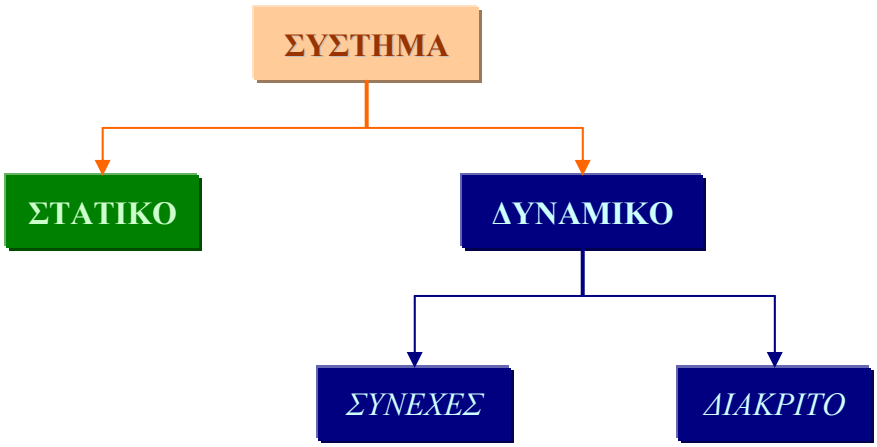
2.2 Η έννοια του συστήματος

Κεντρική ιδέα για κάθε μελέτη προσομοίωσης είναι αυτή του συστήματος. Ένα σύστημα ορίζεται ως μία συλλογή οντοτήτων (π.χ. αεροπλάνα και οι χώροι ενός αεροδρομίου στο οποίο κινούνται) που ενεργούν και αλληλεπιδρούν, με στόχο κάποιο λογικό τερματισμό. Στην πράξη, η έννοια του συστήματος εξαρτάται κάθε φορά από τους στόχους της συγκεκριμένης μελέτης. Η συλλογή οντοτήτων που συγκροτούν ένα σύστημα για μία μελέτη, μπορεί να είναι μόνο ένα υποσύνολο του συνολικού συστήματος για μία άλλη μελέτη.

Ορίζουμε την κατάσταση ενός συστήματος ως τη συλλογή των μεταβλητών που είναι απαραίτητες για την περιγραφή του συστήματος σε μια χρονική στιγμή, αναφορικά με τις απαιτήσεις της μελέτης.

Τα συστήματα μπορούν να ταξινομηθούν σε στατικά και δυναμικά. Στα στατικά δεν έχουμε αλλαγή στην κατάστασή τους με τον χρόνο ενώ στα δυναμικά ισχύει το αντίθετο. Αν ένα σύστημα είναι δυναμικό μπορεί να ταξινομηθεί ανάλογα με το πώς οι μεταβλητές οι οποίες το περιγράφουν αλλάζουν στην διάρκεια του χρόνου. Έτσι μπορεί να είναι διακριτό ή συνεχές. Διακριτό σύστημα είναι αυτό στο οποίο οι μεταβλητές κατάστασης αλλάζουν σε διακεκριμένες στιγμές του χρόνου, ενώ συνεχές είναι το σύστημα του οποίου οι μεταβλητές κατάστασης αλλάζουν συνεχώς στο χρόνο.

Στην πράξη, λίγα συστήματα είναι εξ' ολοκλήρου διακριτά ή συνεχή, αλλά επειδή συνήθως μία από τις δύο ιδιότητες κυριαρχεί, μπορούμε τις περισσότερες φορές να χαρακτηρίσουμε το σύστημα ως διακριτό ή συνεχές. Οι διαχωρισμοί των συστημάτων φαίνονται στο Σχήμα 2.1:



Σχήμα 2.1 Διαχωρισμοί συστημάτων.

Κατά τη διάρκεια της λειτουργίας ενός συστήματος, παρουσιάζεται η ανάγκη μελέτης του, ώστε να αποκτήσουμε γνώση για τις σχέσεις μεταξύ των διαφόρων τμημάτων του, ή για να προβλέψουμε την συμπεριφορά του κάτω από νέες, πιθανές συνθήκες.

Αν είναι δυνατόν να έχουμε φυσική πρόσβαση στο πραγματικό σύστημα και να το λειτουργήσουμε κάτω από τις νέες συνθήκες, τότε δεν υπάρχει αμφισβήτηση για την ακρίβεια της μελέτης. Σπάνια όμως μας δίνεται αυτή η ευκαιρία, είτε γιατί

κοστίζει, ή γιατί είναι πολύ πιθανό η διείσδυσή μας στο σύστημα να προκαλέσει προβλήματα στην κανονική λειτουργία του και να αλλοιώσει τα αποτελέσματα της μελέτης.

Όταν ένα σύστημα είναι στη φάση σχεδιασμού του ή δεν ενδείκνυται η φυσική πρόσβαση σε αυτό, ο μόνος εναλλακτικός τρόπος μελέτης του είναι η μοντελοποίησή του. Έτσι λοιπόν, κατασκευάζουμε ένα μοντέλο, το οποίο θα αναπαριστά το σύστημα και θα το αντικαθιστά κατά τη μελέτη. Όταν χρησιμοποιούμε ένα μοντέλο, η εγκυρότητά του, δηλαδή ο βαθμός ακρίβειας με τον οποίο αναπαριστά το φυσικό σύστημα, είναι πάντα ο πιο κρίσιμος παράγοντας.

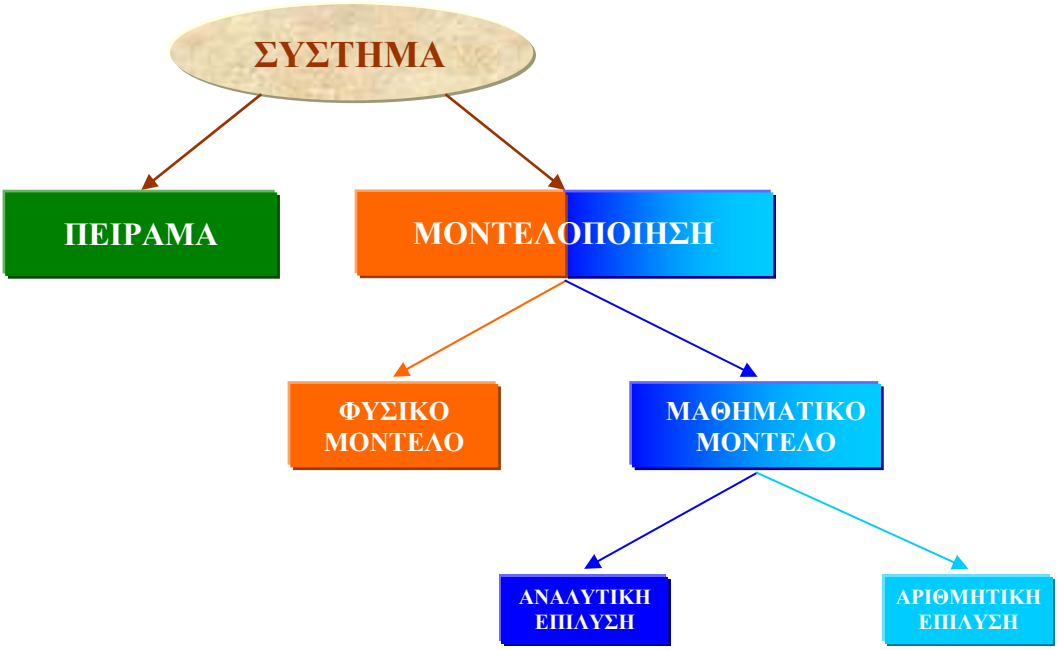
Ένα μοντέλο μπορεί να είναι είτε φυσικό είτε μαθηματικό. Το φυσικό μοντέλο είναι η φυσική αναπαράσταση του αντικειμένου που αντιπροσωπεύει. Για παράδειγμα φυσικά μοντέλα αποτελούν οι μακέτες. Συνήθως, όμως τα φυσικά μοντέλα προσφέρουν περιορισμένες πληροφορίες για την απόκριση του συστήματος σε διάφορες συνθήκες λειτουργίας του. Για το λόγο αυτό, η μεγάλη πλειοψηφία των μοντέλων είναι μαθηματικά. Ένα μαθηματικό μοντέλο αναπαριστά το σύστημα με ένα σύνολο μεταβλητών που συνδέονται μέσω εξισώσεων και λογικών σχέσεων, οι οποίες χειριζόμενες και μεταβαλλόμενες κατάλληλα, επιτρέπουν να δούμε πώς αντιδρά το μοντέλο και κατά συνέπεια πώς θα αντιδρούσε το πραγματικό σύστημα, με την προϋπόθεση βέβαια ότι το μαθηματικό μοντέλο είναι έγκυρο.

Από τη στιγμή που έχει δημιουργηθεί ένα έγκυρο μαθηματικό μοντέλο, θα πρέπει να εξετασθεί ο τρόπος χρήσης του ώστε να μπορεί να απαντήσει στα ερωτήματα που μας ενδιαφέρουν για το σύστημα που αντιπροσωπεύει. Αν το μοντέλο είναι αρκετά απλό, είναι δυνατό να πάρουμε μία ακριβή, αναλυτική λύση μέσω ενός συνόλου εξισώσεων οι οποίες περιγράφουν το σύστημα. Συνήθως, αυτό συμβαίνει μόνο με τα στατικά συστήματα. Στις περισσότερες περιπτώσεις η αναλυτική λύση μπορεί να είναι αρκετά πολύπλοκη και να απαιτεί μεγάλη υπολογιστική ισχύ. Πάντως, αν υπάρχει αναλυτική λύση στο μαθηματικό μοντέλο και είναι υπολογιστικά αποδοτική, συνήθως την προτιμούμε από την προσομοίωση.

Στην περίπτωση που το εξεταζόμενο σύστημα είναι πολύπλοκο, και είναι πολύ δύσκολο ως αδύνατο να εξευρεθούν ή να επιλυθούν μαθηματικές εξισώσεις που

να το περιγράφουν, κάθε πιθανότητα αναλυτικής λύσης αποκλείεται. Τότε το μοντέλο πρέπει να μελετηθεί με αριθμητική επίλυση (προσομοίωση), δηλαδή με την εκτέλεση αριθμητικών πειραμάτων στο μοντέλο για τις εισόδους που μας ενδιαφέρουν, για να δούμε πως αυτά επηρεάζουν τις εξόδους του συστήματος. Η προσομοίωση είναι επίσης πολύ χρήσιμη στις περιπτώσεις που μπορούμε να πάρουμε μόνο προσεγγιστική αναλυτική λύση, οπότε θέλουμε να επιβεβαιώσουμε την προσέγγιση, να βρούμε το σχετικό λάθος κ.λ.π. Συνήθως, τα δυναμικά συστήματα μελετώνται με χρήση κατάλληλης προσομοίωσης.

Στο παρακάτω σχήμα 2.2, δίνεται διαγραμματικά ο τρόπος ανάλυσης και μελέτης ενός συστήματος.



Σχήμα 2.2 Τρόποι μελέτης ενός συστήματος.

2.3 Μοντέλο προσομοίωσης

Έχοντας ένα μαθηματικό μοντέλο που πρέπει να μελετήσουμε με προσομοίωση, θα πρέπει να αναζητήσουμε κατάλληλα εργαλεία για το σκοπό αυτό. Στην προσπάθεια αυτή, είναι χρήσιμο να ταξινομήσουμε τα μοντέλα προσομοίωσης με βάση τέσσερις διαφορετικές έννοιες:

Στατικά ή Δυναμικά Μοντέλα Προσομοίωσης

Ένα στατικό μοντέλο προσομοίωσης, αναπαριστά ένα σύστημα σε μία συγκεκριμένη χρονική στιγμή, ή αναπαριστά ένα σύστημα στο οποίο ο χρόνος δεν έχει σημασία. Αντίθετα, ένα δυναμικό μοντέλο προσομοίωσης αναπαριστά ένα σύστημα, όπως αυτό εξελίσσεται με την πάροδο του χρόνου.

Ντετερμινιστικά ή Στοχαστικά Μοντέλα Προσομοίωσης

Αν ένα μοντέλο προσομοίωσης δεν περιλαμβάνει "τυχαία" τμήματα, ονομάζεται ντετερμινιστικό. Στα ντετερμινιστικά μοντέλα, η έξοδος είναι καθορισμένη, με δεδομένο το σύνολο των ποσοτήτων και σχέσεων εισόδου του μοντέλου. Όμως, πολλά συστήματα χρησιμοποιούν στοχαστικά μοντέλα προσομοίωσης, δηλαδή μοντέλα που θα έχουν τουλάχιστον ορισμένα τμήματα με "τυχαία" είσοδο. Τα περισσότερα υπολογιστικά συστήματα, που βασίζονται στα συστήματα αναμονής (queueing systems), χρησιμοποιούν στοχαστικά μοντέλα προσομοίωσης.

Αυτο-οδηγούμενα ή Ιχνο-οδηγούμενα Μοντέλα Προσομοίωσης

Σε ένα αυτο-οδηγούμενο μοντέλο, υπάρχει μία εσωτερική πηγή τυχαίων αριθμών. Οι τυχαίοι αριθμοί οδηγούν τα τμήματα του μοντέλου, δηλαδή χρησιμοποιούνται για τον προσδιορισμό των στιγμών εμφανίσεων των γεγονότων του συστήματος. Το βασικό χαρακτηριστικό του αυτο-οδηγούμενου μοντέλου είναι ότι αποτελεί ένα αυτάρκες μοντέλο το οποίο δεν χρειάζεται εξωτερικές εισόδους (inputs) για να λειτουργήσει. Αντίθετα, ένα ιχνο-οδηγούμενο (trace-driven) μοντέλο καθοδηγείται από ακολουθίες εισόδου που προέρχονται από δεδομένα που έχουν δημιουργηθεί από τη λειτουργία του πραγματικού συστήματος το οποίο προσομοιώνουμε.

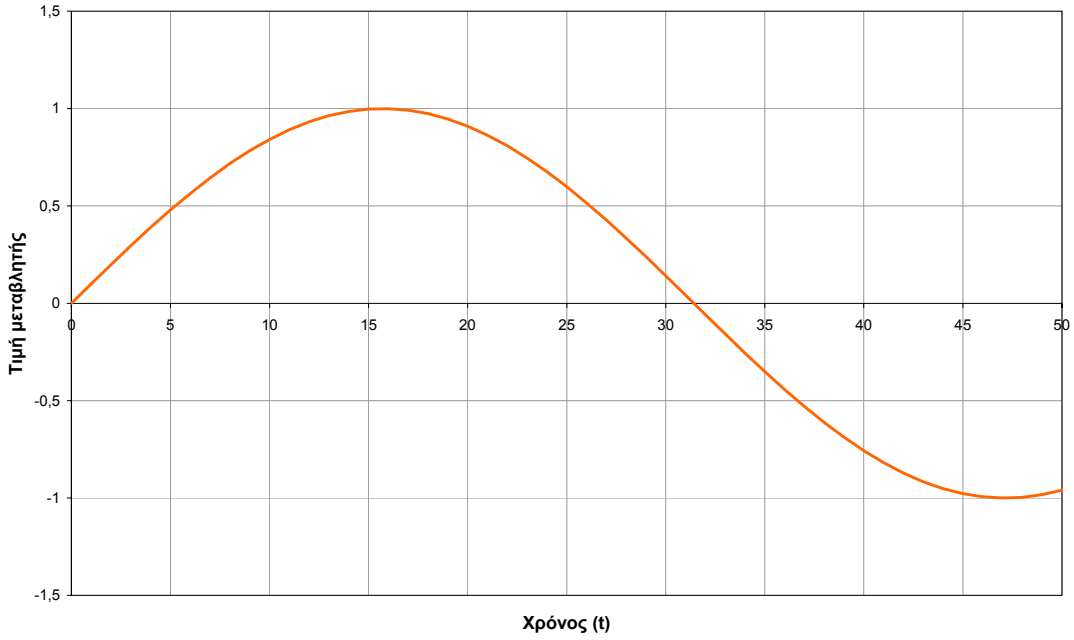
Τα ιχνο-οδηγούμενα μοντέλα έχουν ορισμένα πλεονεκτήματα, όπως το γεγονός ότι αποφεύγονται οι δυσκολίες της πιθανοτικής ανάλυσης που χρειάζεται για τη χρήση κατανομών στην περιγραφή των εισόδων του μοντέλου και επίσης το γεγονός ότι τα μοντέλα αυτά είναι εύκολο να επιβεβαιωθούν. Το πρόβλημα με τα ιχνο-οδηγούμενα μοντέλα είναι το μικρό εύρος εφαρμογών που μπορούν να αντιμετωπίσουν. Οι εφαρμογές αυτές πρακτικά περιορίζονται σε υπολογιστικά

συστήματα και μάλιστα μόνο για τη μελέτη μετατροπών σε ένα σύστημα που ήδη λειτουργεί.

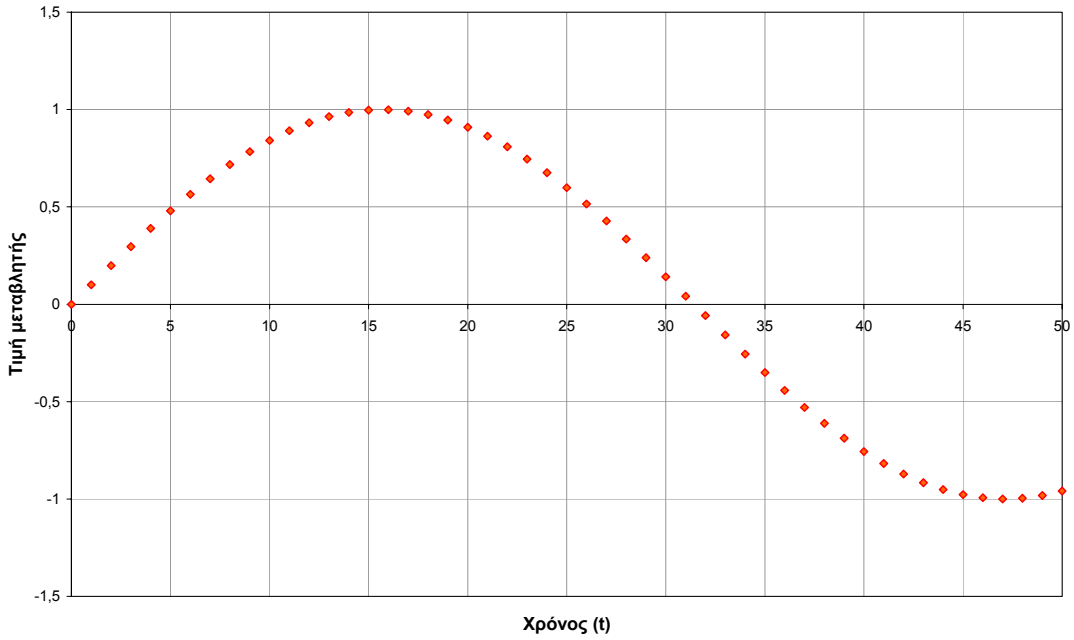
Συνεχή ή Διακριτά Μοντέλα Προσομοίωσης

Οι ορισμοί των συνεχών και διακριτών μοντέλων προσομοίωσης, είναι ανάλογοι με τους ορισμούς των συνεχών και διακριτών συστημάτων που αναφέρθηκαν στην προηγούμενη παράγραφο. Πάντως, πρέπει να σημειωθεί ότι ένα διακριτό μοντέλο δεν χρησιμοποιείται μόνο για την αναπαράσταση ενός διακριτού συστήματος και ένα διακριτό σύστημα δεν αναπαριστάται μόνο από ένα διακριτό μοντέλο προσομοίωσης. Η απόφαση για τη χρήση ενός διακριτού ή ενός συνεχούς μοντέλου για ένα συγκεκριμένο σύστημα, εξαρτάται από τους ιδιαίτερους στόχους της μελέτης. Στο Σχήμα 2.3 έχουμε ένα παράδειγμα συνεχούς και διακριτού μοντέλου προσομοίωσης.

Το μοντέλο προσομοίωσης χρησιμοποιούμε στην παρούσα διπλωματική εργασία, θα είναι διακριτό, δυναμικό, στοχαστικό και αυτο-οδηγούμενο δηλαδή είναι ένα μοντέλο προσομοίωσης διακριτών γεγονότων (discrete event simulation model). Μάλιστα, αφού τα ντετερμινιστικά μοντέλα μπορούν να θεωρηθούν ειδικές περιπτώσεις των στοχαστικών μοντέλων, δεν θα έχουμε απώλεια της γενικότητας στη μελέτη των μοντέλων προσομοίωσης.



(α)

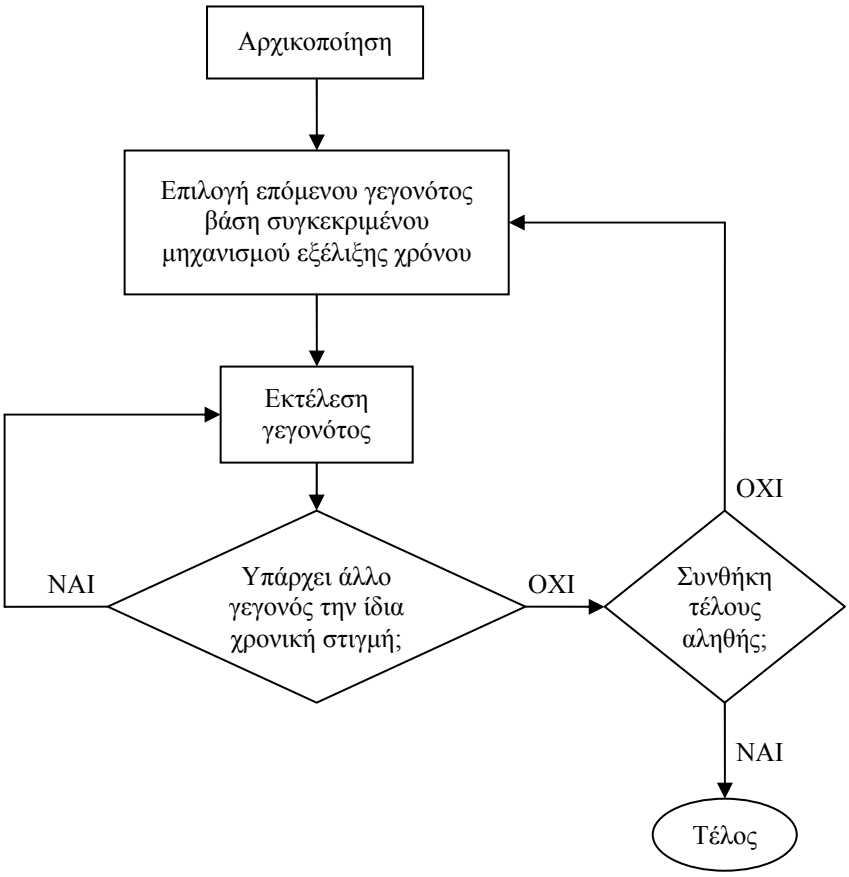


(β)

Σχήμα 2.3 α)Συνεχής προσομοίωση, β)Διακριτή προσομοίωση.

2.4 Μηχανισμός εξέλιξης χρόνου σε προσομοίωση διακριτών γεγονότων

Λόγω του δυναμικού χαρακτήρα των μοντέλων προσομοίωσης διακριτών γεγονότων, πρέπει να έχουμε τη δυνατότητα αποθήκευσης της τρέχουσας τιμής του προσομοιωμένου χρόνου, ενώ χρειαζόμαστε και ένα μηχανισμό αύξησής του από μία τιμή σε μία άλλη. Η μεταβλητή του μοντέλου προσομοίωσης που μας δίνει την τρέχουσα τιμή του χρόνου, ονομάζεται ρολόι προσομοίωσης (simulation clock). Η μονάδα χρόνου που χρησιμοποιεί το ρολόι είναι συνήθως η ίδια με αυτή που χρησιμοποιούν οι παράμετροι εισόδου, ενώ γενικά δεν υπάρχει σχέση του χρόνου που καταγράφει το ρολόι, με το χρόνο που απαιτείται για την εκτέλεση του προσομοιωτή στον υπολογιστή. Κατά την προσομοίωση λοιπόν είναι αναγκαία η χρήση κάποιου μηχανισμού καταγραφής του χρόνου για να μπορούμε να ελέγχουμε τις αλλαγές κατάστασης στο σύστημα που μελετάμε (σχ. 2.4).



Σχήμα 2.4 Σημασία μηχανισμού εξέλιξης του χρόνου.

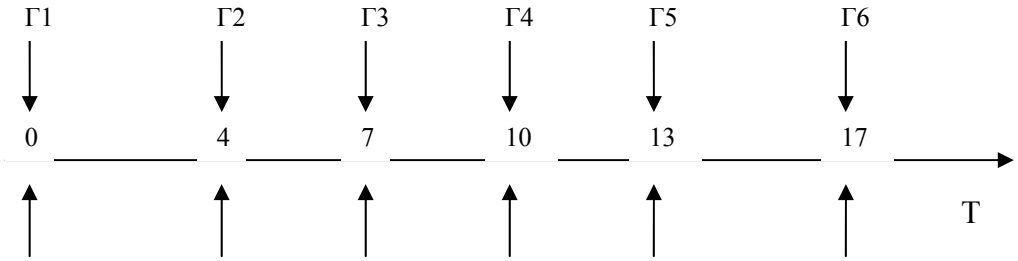
Έχουν επικρατήσει δύο βασικές μέθοδοι για την εξέλιξη του ρολογιού προσομοίωσης οι οποίοι αναφέρονται παρακάτω.

2.4.1 Εξέλιξη με βάση το Χρόνο του Επομένου Γεγονότος

Η Εξέλιξη με βάση το Χρόνο του Επομένου Γεγονότος ή Ασύγχρονη Προσομοίωση (next-event time advance, asynchronous) είναι η μέθοδος κατά την οποία το ρολόι προσομοίωσης αρχικοποιείται στο μηδέν και καθορίζονται εκ των προτέρων οι στιγμές εμφάνισης των μελλοντικών γεγονότων. Το ρολόι τότε αυξάνει στο χρόνο εμφάνισης του πιο κοντινού στο μέλλον, από τα γεγονότα αυτά. Τη στιγμή αυτή η κατάσταση του συστήματος ενημερώνεται ώστε να πάρει υπόψη της το γεγονός που εμφανίστηκε, ενώ ενημερώνεται επίσης η γνώση μας για τις χρονικές στιγμές εμφάνισης των μελλοντικών γεγονότων.

Στη συνέχεια, το ρολόι αυξάνει ώστε να δείχνει τη στιγμή εμφάνισης του νέου πιο κοντινού στο μέλλον γεγονότος, η κατάσταση του συστήματος ενημερώνεται, καθορίζονται οι χρονικές στιγμές εμφάνισης των μελλοντικών γεγονότων κ.ο.κ. Η διαδικασία αυτή εξέλιξης του ρολογιού προσομοίωσης από το ένα γεγονός στο άλλο, συνεχίζεται μέχρι να ικανοποιηθεί κάποια προκαθορισμένη συνθήκη τερματισμού της προσομοίωσης. Αφού όλες οι αλλαγές κατάστασης γίνονται μόνο στις χρονικές στιγμές εμφάνισης των γεγονότων, οι ενδιάμεσες ανενεργοί περίοδοι δεν λαμβάνονται υπόψη και το ρολόι μετακινείται αυτόματα στη στιγμή εμφάνισης του επομένου γεγονότος.

Στο Σχήμα 2.5 έχουμε εμφάνιση γεγονότων Γ1, Γ2, Γ3, Γ4 τις χρονικές στιγμές 0, 4, 7, 10, 13 και 17 αντίστοιχα . Σύμφωνα με την εξέλιξη του χρόνου με βάση το επόμενο γεγονός το ρολόι θα σταματήσει μόνο στις στιγμές εμφάνισης των γεγονότων. Τα βέλη πάνω από τον άξονα του χρόνου Τ δηλώνουν τις χρονικές στιγμές που συμβαίνουν τα γεγονότα ενώ τα βέλη κάτω από τον άξονα του χρόνου δηλώνουν τις χρονικές στιγμές που το ρολόι σταματά για έλεγχο του συστήματος.



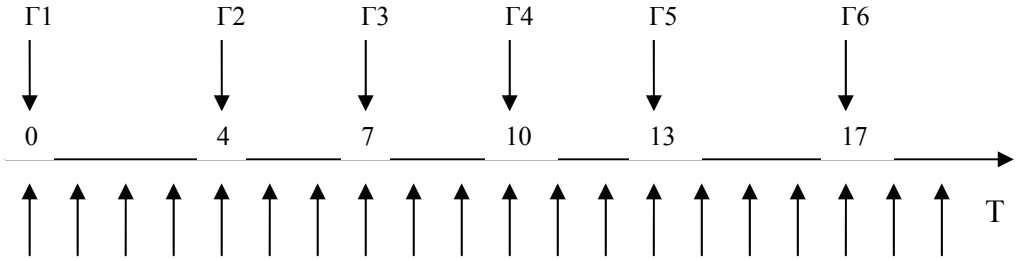
Σχήμα 2.5 Εξέλιξη με βάση το Χρόνο του Επομένου Γεγονότος

2.4.2 Εξέλιξη Σταθερής Αύξησης του Χρόνου

Η Εξέλιξη Σταθερής Αύξησης του Χρόνου ή Σύγχρονη Προσομοίωση (fixed-increment time advance,synchronous) είναι η μέθοδος κατά την οποία το ρολόι προσομοίωσης εξελίσσεται με σταθερές αυξήσεις ακριβώς Δt μονάδων χρόνου κάθε φορά. Μετά από κάθε ενημέρωση του ρολογιού, γίνεται ένας έλεγχος για να εξακριβωθεί εάν θα έπρεπε να έχουν εμφανισθεί κάποια γεγονότα κατά το προηγούμενο χρονικό διάστημα Δt. Αν εμφανίσθηκαν γεγονότα στο διάστημα αυτό, θεωρούμε ότι αυτά εμφανίζονται στο τέλος του χρονικού διαστήματος και η κατάσταση του συστήματος ενημερώνεται κατάλληλα.

Η τιμή του Δt θα πρέπει να είναι μικρή για να έχουμε μεγαλύτερη ακρίβεια. Όσο πιο μικρό όμως είναι το Δt τόσο πιο αργή γίνεται η προσομοίωση αφού ελέγχουμε περισσότερα διαστήματα διάρκειας Δt κατά τα οποία πιθανότατα να μην έχουμε κάποια αλλαγή στο σύστημα. Χρήσιμη σε μοντελοποίηση συστημάτων στα οποία έχουμε εμφάνιση γεγονότων κατά διαστήματα σταθερού μεγέθους.

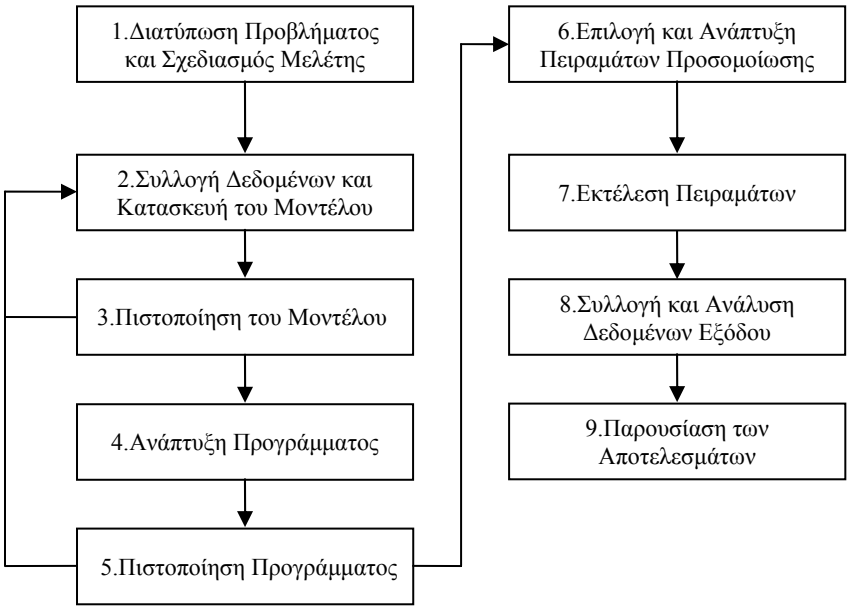
Στο Σχήμα 2.6 έχουμε εμφάνιση γεγονότων Γ1, Γ2, Γ3, Γ4 τις χρονικές στιγμές 0, 4, 7, 10, 13 και 17 αντίστοιχα. Σύμφωνα με την εξέλιξη σταθερής αύξησης του χρόνου το ρολόι της προσομοίωσης σταματά σε όλες τις χρονικές στιγμές που καθορίζονται από το μέγεθος του σταθερού διαστήματος (στην περίπτωση αυτή Δt=1) ανεξάρτητα από το αν συμβαίνει κάποιο γεγονός ή όχι. Τα βέλη πάνω από τον άξονα του χρόνου Τ δηλώνουν τις χρονικές στιγμές που συμβαίνουν τα γεγονότα ενώ τα βέλη κάτω από τον άξονα του χρόνου δηλώνουν τις χρονικές στιγμές που το ρολόι σταματά για έλεγχο του συστήματος.



Σχήμα 2.6 Εξέλιξη Σταθερής Αύξησης του Χρόνου

2.5 Βήματα μελέτης προσομοίωσης

Στο Σχήμα 2.7 φαίνονται τα βήματα που ακολουθεί μία τυπική μελέτη ενός συστήματος με χρήση προσομοίωσης. Τα βήματα 1 έως και 6 αφορούν την κατανόηση και δημιουργία του μοντέλου σε θεωρητικό και μετά σε προγραμματιστικό επίπεδο. Αν αυτό είναι έγκυρο προχωράμε στα βήματα 6 και 7 για την διεξαγωγή των πειραμάτων που μας ενδιαφέρουν. Τέλος, τα αποτελέσματα της προσομοίωσης αναλύονται κατά τα βήματα 8 και 9.



Σχήμα 2.7 Βήματα προσομοίωσης.

2.6 Πλεονεκτήματα και μειονεκτήματα της προσομοίωσης

Τα τελευταία χρόνια η προσομοίωση εφαρμόζεται σχεδόν σε κάθε πεδίο των ανθρώπινων δραστηριοτήτων και το ερώτημα αν όντως είναι τόσο αποτελεσματική για την χρήση που τυγχάνει είναι εύλογο. Είναι γεγονός ότι σε πολλές περιπτώσεις η προσομοίωση δεν είναι εφαρμόσιμη ή ακόμα και αν είναι μπορεί να υπάρχουν άλλοι ποιο γρήγοροι και φθηνοί τρόποι επίλυσης του προβλήματος. Μπορεί η προσομοίωση να είναι ένα από τα πιο εύχρηστα εργαλεία για χρήση στην επιστήμη της διαχείρισης συστημάτων αλλά πιθανόν και από τα πιο δύσκολα ως προς την σωστή εφαρμογή τους και ως προς την απεικόνιση εύστοχων συμπερασμάτων. Γενικά θα πρέπει να αναγνωρίσουμε ορισμένα πλεονεκτήματα και μειονεκτήματα για την όλη διαδικασία μιας προσομοίωσης .

Ορισμένα πλεονεκτήματα είναι τα εξής:

- Η προσομοίωση επιτρέπει να πραγματοποιούνται ελεγχόμενοι πειραματισμοί πάνω σε ένα σύστημα. Ένα πείραμα προσομοίωσης μπορεί να εκτελεστεί πολλές φορές αλλά με διαφορετικές τιμές εισόδου για να εξεταστεί η συμπεριφορά του υπό εξέταση συστήματος κάτω από διαφορετικές κάθε φορά καταστάσεις και συνθήκες.

- Ένα άλλο πλεονέκτημα είναι ότι έχουμε συμπίεση του χρόνου. Αυτό σημαίνει ότι η λειτουργία ενός συστήματος η οποία διαρκεί για κάποιο εκτεταμένο χρονικό διάστημα μπορεί να προσομοιωθεί μέσα σε λίγα λεπτά με την χρήση ενός σύγχρονου υπολογιστή.

- Τα περισσότερα σύνθετα συστήματα του πραγματικού κόσμου με στοχαστικές παραμέτρους, δεν μπορούν να περιγραφούν ικανοποιητικά με κάποιο μαθηματικό μοντέλο που μπορεί να λυθεί αναλυτικά. Έτσι, η προσομοίωση είναι συχνά η μόνη διαθέσιμη μέθοδος μελέτης.

- Επιτρέπεται ανάλυση της ευαισθησίας του συστήματος η οποία απορρέει από διαφοροποιήσεις των μεταβλητών εισόδου.

- Δεν έχουμε διατάραξη του πραγματικού συστήματος. Αυτό είναι ένα μεγάλο πλεονέκτημα αφού πολλοί διαχειριστές, αναλυτές δεν είναι πρόθυμοι να εφαρμόσουν πειραματικές στρατηγικές σε ένα πραγματικό σύστημα το οποίο βρίσκεται σε λειτουργία.

- Η προσομοίωση είναι ένα αποδοτικό εργαλείο εκπαίδευσης.

- Είναι οικονομική μέθοδος, αφού είναι δυνατό να υλοποιηθεί σε μικρούς υπολογιστές με τη χρήση γλωσσών προγραμματισμού γενικού σκοπού όπως η C, η C++ και η MATLAB.

- Μπορούν μέσο της προσομοίωσης να συγκριθούν εναλλακτικές προτάσεις τόσο σε επίπεδο σχεδιασμού όσο και σε επίπεδο λειτουργίας του συστήματος, ώστε να προσδιορισθεί η βέλτιστη λύση που ικανοποιεί τις προδιαγραφές που έχουν ορισθεί.

Κάποια από τα μειονεκτήματα είναι τα εξής:

- Μπορεί να υπάρξουν μη εμφανής κρίσιμες υποθέσεις οι οποίες να παραληφθούν με αποτέλεσμα να διαφοροποιήσουν το πρόβλημα από όπως έχει στην πραγματικότητα. Κάτι τέτοιο θα πρέπει να αποφθεχθεί κατά την διαδικασία πιστοποίησης της διαδικασίας προσομοίωσης. Όμως και πάλι η ανίχνευση μιας τέτοιας κατάστασης εξαρτάται από την αυστηρότητα και την επιμέλεια με την οποία πιστοποιούμε το μοντέλο.

- Οι παράμετροι ενός μοντέλου ίσως είναι δύσκολο να αρχικοποιηθούν. Μπορεί να απαιτούν χρονοβόρες διαδικασίες για την συλλογή, ανάλυση και αποσαφήνιση τους.

- Τα μοντέλα προσομοίωσης συχνά απαιτούν πολύ χρόνο και πόρους για να αναπτυχθούν. Το κόστος μιας προσομοίωσης μπορεί να ελαττωθεί μέσο μιας εις βάθος κατανόησης του συστήματος που προσομοιώνεται προτού αναπτυχθεί το μοντέλο και μέσο ενός προσεκτικού σχεδιασμού του πειράματος της προσομοίωσης.

- Ο μεγάλος όγκος αριθμών που παράγονται από μία μελέτη προσομοίωσης ή η εντύπωση που δημιουργούν οι τυχόν γραφικές παραστάσεις των αποτελεσμάτων της, συχνά ενισχύουν μία τάση να δίνεται μεγαλύτερη εμπιστοσύνη στα αποτελέσματα αυτά από όσο πρέπει. Αν το μοντέλο δεν είναι μία αρκετά έγκυρη αναπαράσταση του συστήματος, τα αποτελέσματα της προσομοίωσης, ανεξάρτητα του πόσο εντυπωσιακά είναι, θα προσθέσουν λίγη χρήσιμη πληροφορία για το πραγματικό σύστημα.

2.7 Προσομοίωση ενός συστήματος αναμονής - Ουρές αναμονής

Οι ουρές αναμονής είναι από τα πιο κοινά προβλήματα που εμφανίζονται στην μοντελοποίηση ή προσομοίωση ενός συστήματος και οφείλουν την ύπαρξή τους σε έλλειψη πόρων ενός συστήματος. Η απαίτηση για να αντιμετωπιστούν οι ουρές αναμονής ήταν αυτό που πραγματικά οδήγησε στην ανάπτυξη των γλωσσών προγραμματισμού.

Μία ουρά αναμονής αποτελείται από οντότητες οι οποίες περιμένουν να εξυπηρετηθούν από έναν ή περισσότερους εξυπηρετητές. Στην περίπτωση ενός αεροδρομίου μπορούμε να θεωρήσουμε ως οντότητες τα αεροπλάνα και ως εξυπηρετητές τις διάφορες εγκαταστάσεις του αεροδρομίου που χρησιμοποιούν τα αεροπλάνα. Έτσι σε ένα αεροδρόμιο έχουμε εμφάνιση συστημάτων αναμονής όταν εμφανίζονται καθυστερήσεις λόγω της απαίτησης χρήσης ενός εξυπηρετητή από περισσότερα του ενός αεροπλάνων.

Η όλη διαδικασία έχει ως εξής: έχουμε αφίξεις αεροπλάνων τα οποία ζητούν άμεση εξυπηρέτηση από κάποιον χώρο ενός αεροδρομίου (π.χ. διάδρομο προσγείωσης), όταν ο εξυπηρετητής είναι ελεύθερος δεν έχουμε αναμονή, το αεροπλάνο εξυπηρετείται και μετά αναχωρεί από τον συγκεκριμένο χώρο του αεροδρομίου (εξυπηρετητή). Όταν ο εξυπηρετητής είναι κατειλημμένος από ένα άλλο αεροπλάνο τότε έχουμε αναμονή σε ουρά (queue) μέχρι αυτός να ελευθερωθεί. Όταν ολοκληρωθεί μια εξυπηρέτηση τότε το αεροπλάνο αποχωρεί από το σύστημα αναμονής. Ένα απλό σύστημα αναμονής με έναν εξυπηρετητή παρουσιάζεται στο Σχήμα 2.8:

Σχήμα 2.8 Σύστημα αναμονής ενός εξυπηρετητή

Όπως φαίνεται και από το σχήμα, τα συστήματα αναμονής αποτελούνται από στατικά αντικείμενα τα οποία είναι οι πόροι (server, queue), από δυναμικά αντικείμενα που είναι οι χρήστες (π.χ. αεροπλάνα), και από τα μέρει πηγές όπου δημιουργούνται οι αφίξεις και καταβόθρες όπου καταλήγουν οι χρήστες όταν εξυπηρετηθούν.

Τα συστήματα αναμονής έχουν ορισμένα χαρακτηριστικά τα οποία και αναφέρουμε παρακάτω:

Συνάρτηση πυκνότητας πιθανότητας των χρόνων ανάμεσα στις αφίξεις

Ο χρόνος που μεσολαβεί ανάμεσα σε δύο αφίξεις μπορεί να λαμβάνει μια προκαθορισμένη γνωστή τιμή (ντετερμινιστική προσέγγιση) ή μπορεί να είναι μια τυχαία μεταβλητή που ακολουθεί κάποια κατανομή όπως αυτή της Poisson (στοχαστική προσέγγιση). Εδώ θα πρέπει επίσης να επιλέξουμε το αν θα έχουμε άπειρο ή πεπερασμένο αριθμό αφίξεων.

Συνάρτηση πυκνότητας πιθανότητας των χρόνων εξυπηρέτησης

Και εδώ, ο χρόνος εξυπηρέτησης μπορεί να λαμβάνει μια προκαθορισμένη γνωστή τιμή (ντετερμινιστική προσέγγιση) ή μπορεί να είναι μια τυχαία μεταβλητή που ακολουθεί κάποια κατανομή όπως αυτή της Poisson (στοχαστική προσέγγιση).

Αριθμός παράλληλων εξυπηρετητών

Σε μερικά συστήματα αναμονής υπάρχουν περισσότεροι του ενός εξυπηρετητές που εξυπηρετούν ταυτόχρονα τα αεροπλάνα. Οι εξυπηρετητές αυτοί

μπορούν να έχουν όλοι την ίδια μία ουρά αναμονής ή να έχει ο καθένας την δικιά του.

Προτεραιότητα στην ουρά

Εδώ καθορίζεται ο τρόπος επιλογής του επόμενου αεροπλάνου που πρόκειται να εξυπηρετηθεί. Η πιο συνηθισμένη νόρμα είναι η 'Πρώτος ερχόμενος πρώτος εξυπηρετούμενος' (FCFS, first come first served). Υπάρχουν όμως και άλλες νόρμες όπως η 'Τελευταίος ερχόμενος πρώτος εξυπηρετούμενος' (LCFS, last come first served), ή η τυχαία επιλογή του πελάτη προς εξυπηρέτηση, καθώς και αυτή, στην οποία υπάρχουν διαφορετικές προτεραιότητες για κάθε αεροπλάνο (π.χ. αν προέρχονται από διαφορετικές ουρές αναμονής και καταλήγουν στον ίδιο εξυπηρετητή) και το καθένα εξυπηρετείται ανάλογα με την προτεραιότητά του.

Μέγιστο επιτρεπόμενο μήκος ουράς

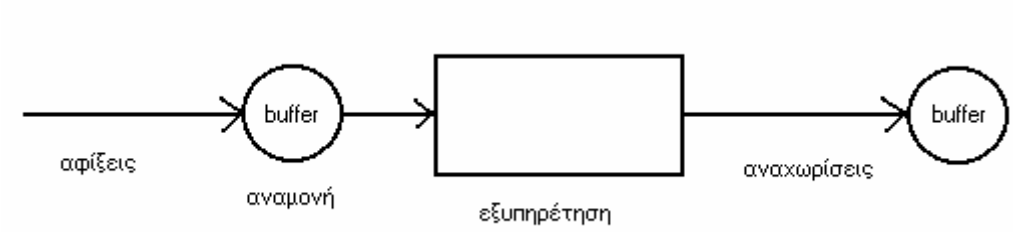
Στις περισσότερες των περιπτώσεων θεωρείται ότι η ουρά αναμονής μπορεί να έχει άπειρο μήκος .Υπάρχουν όμως και συστήματα αναμονής στα οποία ο χώρος που διατίθεται για την ουρά είναι προκαθορισμένος πράγμα που πολλές φορές είναι απαραίτητο για μια ρεαλιστική αντίληψή τους.

Για τη περιγραφή των χαρακτηριστικών κάθε ουράς αναμονής χρησιμοποιείται μια συντομογραφία που αποτελείται από πέντε σύμβολα A/B/m/K/M. Σε αυτή τη συντομογραφία, τα A και B καθορίζουν την κατανομή των χρόνων άφιξής και εξυπηρέτησης αντίστοιχα, το m καθορίζει τον αριθμό των παράλληλων εξυπηρετητών, το K είναι η χωρητικότητα της ουράς αναμονής και τέλος το M είναι ο συνολικός αριθμός των αφίξεων. Τα A, B όταν καθορίζονται από εκθετικές κατανομές (π.χ. Poisson) παίρνουν την τιμή M, ενώ όταν καθορίζονται ντετερμινιστικά παίρνουν την τιμή D. Υποθέτουμε ότι για όλες τις ουρές αναμονής ισχύει η πειθαρχία FCFS.

Θα μας απασχολήσουν τα παρακάτω συγκεκριμένα συστήματα αναμονής:

- *M/D/1/∞/M*

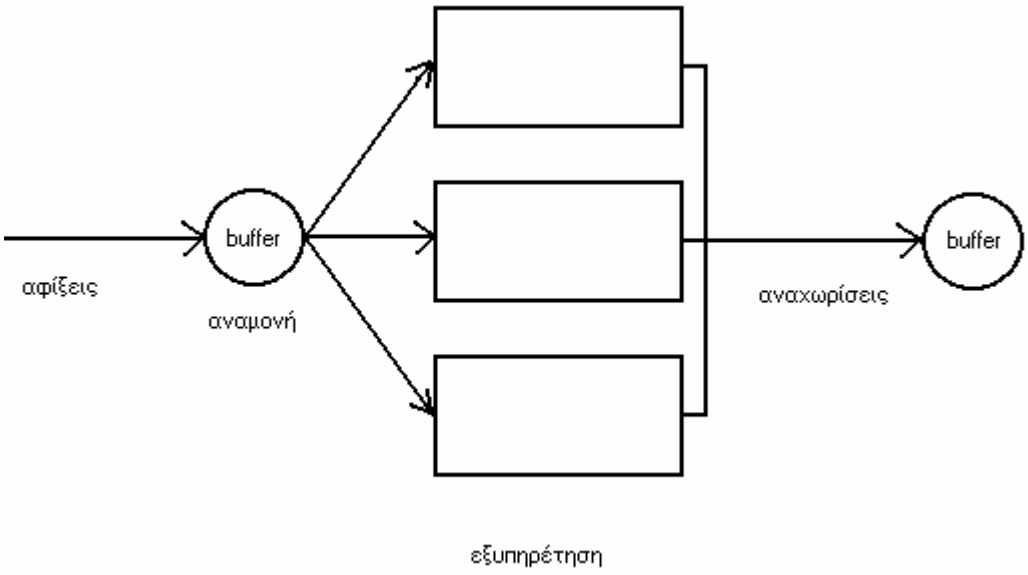
  Είναι η περίπτωση όπου έχουμε αφίξεις βάση της κατανομής Poisson ενώ ο χρόνος εξυπηρέτησης είναι σταθερός. Επιπλέον έχουμε έναν εξυπηρετητή άπειρο χώρο για αναμονή και συγκεκριμένο αριθμό αφίξεων. Η περίπτωση αυτή παρουσιάζεται στο Σχήμα 2.9.



Σχήμα 2.9 M/D/1/∞/M

- *M/D/m/∞/*

  Είναι η περίπτωση όπου έχουμε αφίξεις βάση της κατανομής Poisson ενώ ο χρόνος εξυπηρέτησης είναι σταθερός . Επιπλέον έχουμε περισσότερους του ενός εξυπηρετητές με έστω m=3 (τρεις εξυπηρετητές), άπειρο χώρο για αναμονή και συγκεκριμένο αριθμό αφίξεων. Η περίπτωση αυτή παρουσιάζεται στο Σχήμα 2.10.



Σχήμα 2.10 M/D/m

2.8 Κατανομή Poisson

Θα πρέπει να δοθεί ιδιαίτερη σημασία στην κατανομή των τυχαίων αφίξεων η οποία περιγράφει ότι σε οποιοδήποτε δεδομένο χρονικό διάστημα t ο αριθμός των αφίξεων είναι εντελώς τυχαίος. Για την περίπτωση των αφίξεων αεροπλάνων σε ένα αεροδρόμιο η προσέγγιση με μια κατανομή Poisson αντιπροσωπεύει γενικά την πραγματικότητα.

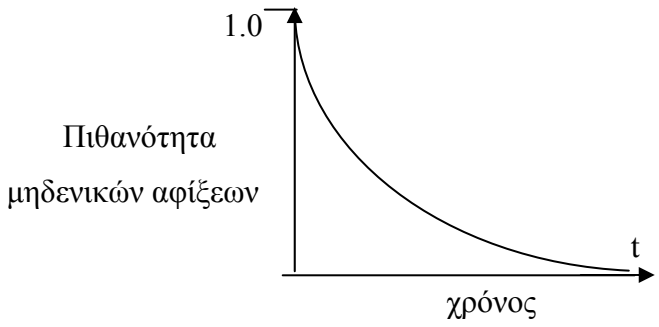
Έτσι η πιθανότητα άφιξης x αεροπλάνων την χρονική περίοδο t μπορεί να δοθεί από τον τύπο της κατανομής Poisson, ο οποίος είναι:

$$P(x) = \frac{(\lambda t)^X e^{-\lambda t}}{x!} \qquad (1)$$

με λ τον μέσο ρυθμό αφίξεων και την απόκλιση της κατανομής. Έτσι όσο μεγαλώνουν οι αριθμοί σε ένα συγκεκριμένο δείγμα τυχαίων αριθμών Poisson τόσο μεγαλώνει και η διασπορά τους.. Από την εξίσωση (1) θέτοντας x = 0 έχουμε:

$$P(0) = e^{-\lambda t} \qquad (2)$$

Αυτή είναι η πιθανότητα να έχουμε μηδέν αφίξεις το χρονικό διάστημα t. Αυτή είναι μια αρνητική εκθετική συνάρτηση η οποία φαίνεται στο Σχήμα 2.12.



Σχήμα 2.12 Πιθανότητα μηδενικών αφίξεων το χρονικό διάστημα t

Αν αφαιρέσουμε την συνάρτηση (2) από την μονάδα, έχουμε την αθροιστική πιθανότητα (cumulative probability) για μία ή περισσότερες αφίξεις κατά το χρονικό διάστημα t:

$$F(t) = 1 - e^{-\lambda t} \qquad (3)$$

Από την θεωρία πιθανοτήτων έχουμε:

$$f(t) = \frac{d}{dt} F(t)$$

η οποία γίνεται:

$$\underline{f(t)} = \frac{d}{dt}\left(1 - e^{\lambda t}\right) = \lambda\, e^{-\lambda t} \qquad (4)$$

Ο τύπος αυτός καθορίζει την κατανομή του χρόνου μεταξύ των αφίξεων. Είναι η συνάρτηση πυκνότητας της γνωστής εκθετικής κατανομής με μέση τιμή $1/\lambda$. Έτσι μπορούμε να ορίσουμε την κατανομή των αφίξεων από τον χρόνο μεταξύ των αφίξεων ως:

$$a(t) = \lambda\, e^{-\lambda t} \quad (5)$$

Αυτή είναι μια πολύ σημαντική περίπτωση για τους χρόνους αφίξεων. Η κατανομή Poisson είναι μια διακριτή κατανομή μίας παραμέτρου που λαμβάνει θετικές ακέραιες τιμές, γεγονός που την κάνει απλή σε επίπεδο μαθηματικών.

2.9 Ανάπτυξη προγραμμάτων προσομοίωσης

Η ανάπτυξη προγραμμάτων προσομοίωσης μπορεί να γίνει με δυο διαφορετικούς τρόπους. Ο πρώτος είναι με την χρήση γλωσσών που είναι εξειδικευμένες για προσομοίωση και ο δεύτερος με την χρήση γλωσσών προγραμματισμού γενικής χρήσης όπως C, C++, MATLAB.

Στον πρώτο τρόπο έχουμε αυτοματοποίηση των βασικών λειτουργιών της προσομοίωσης όπως του μηχανισμού του χρόνου, της παραγωγής τυχαίων αριθμών, του ελέγχου συνθηκών τερματισμού κλπ. Δεν προσφέρουν όμως την ευελιξία που έχουμε από τον δεύτερο τρόπο, όπου μπορεί κανείς να προγραμματίσει κάθε είδους σύστημα όσο πολύπλοκο και να είναι.

Αντιθέτως, πολλές εξειδικευμένες γλώσσες έχουν την δυνατότητα ενσωμάτωσης σε αυτές ρουτινών και υποπρογραμμάτων γραμμένες σε άλλη γλώσσα, γεγονός που δίνει και σε αυτές κάποια ευελιξία. Όταν όμως ο αριθμός των ρουτινών αυτών είναι μεγάλος, η τροποποίηση του προγράμματος και η ανίχνευση των λαθών γίνεται ιδιαίτερα δύσκολη.

Τα λάθη αυτά εμφανίζονται συνήθως στην επικοινωνία της εξειδικευμένης γλώσσας με τις ρουτίνες του χρήστη. Έτσι είναι προτιμότερο να αναπτυχθεί η εφαρμογή εξολοκλήρου σε μια γλώσσα γενικής χρήσεως, αφού έτσι ο προγραμματιστής έχει πλήρη έλεγχο της ροής του προγράμματος και της διασύνδεσης των τμημάτων του. Στην παρούσα διπλωματική έγινε η επιλογή της χρήσης γλώσσας προγραμματισμού γενικής χρήσης και συγκεκριμένα αυτή του MATLAB λόγω του πλεονεκτήματος της ευελιξίας.

Παρακάτω θα κάνουμε μια αναφορά για τις εξειδικευμένες γλώσσες προσομοίωσης και θα αναφερθούμε ειδικά σε κάποιες από αυτές. Το ευρύ πεδίο εφαρμογών της προσομοίωσης δημιούργησε την ανάγκη για την ανάπτυξη εξειδικευμένων γλωσσών προσομοίωσης που επιτρέπουν στο χρήστη να προσομοιώνει το μοντέλο του συστήματός του με μερικές απλές εντολές.

Το πλεονέκτημα λοιπόν που παρέχουν είναι η δραματική μείωση του χρόνου που απαιτείται για τον προγραμματισμό ενός μοντέλου. Αυτό είναι αρκετά σημαντικό γιατί επιτρέπει στον αναλυτή, που κάνει χρήση της προσομοίωσης στη μελέτη του, να αφιερώσει περισσότερο χρόνο στις άλλες φάσεις της μελέτης του όπως είναι η ανάλυση των αποτελεσμάτων και η επαλήθευση του μοντέλου που χρησιμοποιεί. Εκτός από την βοήθεια στο στάδιο του προγραμματισμού, οι εξειδικευμένες γλώσσες προσομοίωσης προσφέρουν δυνατότητες μοντελοποίησης με παραστατική καθοδήγηση στην ανάπτυξη του μοντέλου και βοήθεια στην τεκμηρίωση και παρουσίαση των αποτελεσμάτων.

Υπάρχουν όμως και μειονεκτήματα όπως το υψηλό κόστος σε χρήματα της απόκτησης και συντήρησης τους. Επίσης οι αναλυτές θα πρέπει να αφιερώσουν αρκετό χρόνο στην εκμάθηση της εξειδικευμένης γλώσσας προσομοίωσης που πρόκειται να χρησιμοποιήσουν η ευελιξία της οποίας μπορεί να είναι μειωμένη και αυτό να δημιουργεί και περαιτέρω προβλήματα. Ακόμα οι εξειδικευμένες γλώσσες προσομοίωσης έχουν υψηλές απαιτήσεις σε υπολογιστική ισχύ και μνήμη του υπολογιστή σε σχέση με γλώσσες προσομοίωσης που αναπτύσσονται πάνω σε γενικές γλώσσες προγραμματισμού.

Οι εξειδικευμένες γλώσσες προσφέρουν έτοιμα αρκετά στοιχεία που χρειάζονται για την μοντελοποίηση και εκτέλεση ενός προγράμματος προσομοίωσης. τα πιο συνηθισμένα από τα οποία είναι:

- Άποψη του κόσμου με προσανατολισμό σε γεγονότα, δραστηριότητες, διεργασίες.
- Μηχανισμό ροής του χρόνου.
- Γεννήτρια τυχαίων που να ακολουθούν την κατανομή που προσδιορίζει ο χρήστης.
- Μηχανισμούς συλλογής δεδομένων του μοντέλου.
- Παροχή στοιχειώδους στατιστικής ανάλυσης των δεδομένων που έχουν συλλέξει.
- Παροχή διαγνωστικών μηχανισμών και ανίχνευσης λαθών.

Κάποιες από τις πιο σημαντικές εξειδικευμένες γλώσσες προσομοίωσης είναι και οι παρακάτω:

SIMSCRIPT

Δημιουργήθηκε από τον Markowitz και τους συνεργάτες του την δεκαετία του 1960 και σήμερα υπάρχουν εκδόσεις σχεδόν για κάθε είδους υπολογιστικό σύστημα. Υλοποιείται στις γλώσσες assembly και C. Οι πλατφόρμες για τις οποίες είναι διαθέσιμη είναι PC, πολλές πλατφόρμες UNIX, και πολλά μεγάλα συστήματα διαμοιρασμού χρόνου. Μέθοδος του προγραμματισμού είναι με εντολές και η διαχείριση μνήμης είναι δυναμική. Έχει άποψη του κόσμου ως γεγονότος ή διεργασίας και το κόστος της είναι υψηλό.

MODSIM

Είναι μια γλώσσα προσομοίωσης νέας γενιάς. Αν και είναι προσανατολισμένη στην προσομοίωση, μπορεί να χρησιμοποιηθεί και ως γλώσσα γενικής χρήσεως. Συνδυάζει τα πλεονεκτήματα των αντικειμενοστραφών γλωσσών με δυνατότητες προσομοίωσης διακριτών γεγονότων. Λόγω της δυνατότητας επαναχρησιμοποίησης κώδικα που προσφέρει, χρησιμοποιείται περισσότερο για την κατασκευή μεγάλων διεργασιακών μοντέλων. Ακόμα έχει ενδιάμεσο μεταγλωττιστή, δηλαδή δεν παράγει κώδικα σε γλώσσα μηχανής αλλά σε πηγαίο κώδικα άλλης γλώσσας η οποία είναι η C++. Είναι διαθέσιμη για πλατφόρμες PC και UNIX, και η μέθοδος προγραμ-

ματισμού είναι με εντολές. Διαχειρίζεται την μνήμη δυναμικά και το κόστος είναι υψηλό.

GPSS

Αναπτύχθηκε την δεκαετία του 1960 αλλά ακόμα και σήμερα έχει πολλούς υποστηρικτές και είναι διαθέσιμη για κάθε πλατφόρμα υπολογιστικού συστήματος. Είναι ίσως η μόνη γλώσσα προσομοίωσης που σχεδιάσθηκε από την αρχή για αναλυτές οι οποίοι δεν είχαν καμία σχέση με τον προγραμματισμό. Η ανάπτυξη προγραμμάτων στην GPSS γίνεται με την παράθεση μερικών απλών εντολών που καθορίζουν την κίνηση των οντοτήτων μέσα στο μοντέλο. Οι βασικές της έννοιες είναι η δοσοληψία, η εγκατάσταση και το μπλοκ. Οι δοσοληψίες αντιστοιχούν σε ένα μέρος των οντοτήτων του μοντέλου και έχουν τα ακόλουθα χαρακτηριστικά:

- Είναι δυναμικές οντότητες που κινούνται μέσα στο μοντέλο όπως για παράδειγμα τα αεροπλάνα σε ένα αεροδρόμιο.
- Κατά την διάρκεια της προσομοίωσης, οι δοσοληψίες έρχονται στο μοντέλο και φεύγουν από αυτό.
- Είναι δυνατό να υπάρχουν ταυτόχρονα πολλές παρόμοιες ενεργές δοσοληψίες στο μοντέλο.
- Η κατάσταση των δοσοληψιών, δηλαδή η θέση τους στο μοντέλο και οι αριθμητικές τους ιδιότητες, ενημερώνονται σειριακά από τον επεξεργαστή της GPSS.
- Όταν μια δοσοληψία έχει αρχίσει να κινείται μέσα στο μοντέλο, η κίνησή της από μπλοκ σε μπλοκ συνεχίζεται μέχρι να συμβεί ένα από τρία γεγονότα. Πρώτον όταν η δοσοληψία εισέρχεται σε ένα μπλοκ που έχει ως σκοπό να την κρατήσει εκεί. Δεύτερον, όταν η δοσοληψία εισέρχεται σε ένα μπλοκ που έχει ως σκοπό να την αφαιρέσει από το μοντέλο. Τρίτον όταν το επόμενο μπλοκ στην πορεία της δοσοληψίας αρνείται την είσοδό της σε αυτό.

Τα μπλοκ είναι περίπου αντίστοιχα με τις εντολές σε μια γλώσσα προγραμματισμού με την έννοια ότι ο αναλυτής αναπτύσσει το πρόγραμμα της προσομοίωσης του μοντέλου παραθέτοντας μια σειρά από μπλοκ. Ο σκοπός του κάθε μπλοκ είναι είτε ο έλεγχος της κίνησης των δοσοληψιών, είτε η συλλογή στατιστικών στοιχείων της προσομοίωσης. Εκτός από τα μπλοκ υπάρχουν και εντολές ελέγχου με

τις οποίες καθορίζεται η συνολική πορεία της προσομοίωσης. Οι εγκαταστάσεις είναι μια κατηγορία μπλοκ που σκοπό έχουν την παροχή υπηρεσιών προς τις δοσοληψίες. Μπορούμε να τις αντιστοιχίσουμε με τους εξυπηρετητές ενός συστήματος ουράς.

Η διαχείριση του χρόνου γίνεται από τον επεξεργαστή της γλώσσας GPSS χωρίς να απαιτείται καμία προγραμματιστική παρέμβαση από τον αναλυτή και έχει τις παρακάτω ιδιότητες:

- Καταγράφει μόνο ακέραιες τιμές.
- Η μονάδα του χρόνου είναι αυθαίρετη και καθορίζεται από τον αναλυτή ανάλογα με τις απαιτήσεις του μοντέλου. Για τον λόγο αυτό το ότι το ρολόι καταγράφει μόνο ακέραιες τιμές δεν αποτελεί πρόβλημα γιατί ο αναλυτής μπορεί να χρησιμοποιήσει οποιοδήποτε ακέραιο πολλαπλάσιο της χρονικής του μονάδας.
- Αρχικοποιείται στην τιμή μηδέν.
- Καθώς γίνεται η ενημέρωση του συστήματος, το ρολόι παραμένει ακίνητο αν και παρέρχεται πραγματικός χρόνος.
- Αυξάνεται κατά χρονικά διαστήματα που είναι συνάρτηση των συνθηκών του συστήματος. Άρα ο μηχανισμός χρόνου αντιστοιχεί στον μηχανισμό ροής χρόνου επόμενου γεγονότος.

Η συλλογή στατιστικών στοιχείων και η παρουσίαση των αποτελεσμάτων της προσομοίωσης γίνεται αυτόματα από την GPSS. Και ο αναλυτής όμως αν επιθυμεί μπορεί να εισάγει ειδικές εντολές που αφορούν αυτές τις ενέργειες. Τέλος, υλοποιείται σε γλώσσα assembly και έχει μέσο κόστος.

ΚΕΦΑΛΑΙΟ 3. Φυσικό Πρόβλημα

3.1 Στοιχεία λειτουργίας αεροδρομίων

Ως αεροδρόμιο ή αερολιμένας ορίζεται "*οποιαδήποτε χερσαία ή υδάτινη περιοχή, καθορισμένη με πράξη της αρμόδιας Αρχής, η οποία χρησιμεύει ολικά ή μερικά για την προσγείωση, απογείωση, κίνηση και εξυπηρέτηση των αεροσκαφών.*"

Τα αεροδρόμια, βάση της αποστολής των αεροσκαφών που τα χρησιμοποιούν, χωρίζονται σε τρεις κύριες κατηγορίες:

- **Πολιτικά αεροδρόμια,** που αποσκοπούν στην επιβίβαση, αποβίβαση και διακίνηση ανθρώπων ή και εμπορευμάτων.

- **Στρατιωτικά αεροδρόμια,** που χρησιμοποιούνται αποκλειστικά για στρατιωτικούς σκοπούς.

- **Ιδιωτικά αεροδρόμια,** που εξυπηρετούν σκοπούς αναψυχής ή άλλους σκοπούς της γενικής αεροπορίας, όπως ερευνητικές εργασίες, εξυπηρέτηση ασθενών, κτλ.

Το πλήθος των πολιτικών αεροδρομίων παγκοσμίως είναι σημαντικό, υπερβαίνει τις 30000. Από αυτά περίπου το 55% είναι Δημόσια και τα υπόλοιπα Ιδιωτικά. Μόνον όμως 1000 περίπου αεροδρόμια προσφέρονται για διεθνείς πτήσεις. Στην παρούσα διπλωματική εργασία, θα επικεντρωθούμε στα πολιτικά αεροδρόμια.

Ειδικότερα, αεροδρόμιο είναι ο χώρος και οι κατασκευές πού εξυπηρετούν τις ακόλουθες ανάγκες και διεργασίες:

Σε σχέση με τα αεροσκάφη:

- Προσγειοαπογειώσεις
- Στάθμευση, κατά τον χρόνο πού δεν χρησιμοποιούνται
- Στάθμευση, κατά τον χρόνο που πραγματοποιείται ή επιβίβαση- αποβίβαση επιβατών και αποσκευών
- Φορτοεκφόρτωση εμπορευμάτων, κλπ

- Ανεφοδιασμός, καθαρισμός, κλπ
- Έλεγχος, συντήρηση και επισκευές

Σε σχέση με τους επιβάτες πού θα ταξιδεύσουν:

- Σύντομη παραμονή των επιβατών και φίλων τους πού τους συνοδεύουν
- Έλεγχος εισιτηρίων, βάρους αποσκευών καθώς και παραλαβή από τους επιβάτες των αποσκευών τους
- Μεταφορά των αποσκευών και φόρτωση στα αεροσκάφη.
- Έλεγχοι ασφάλειας
- Πρόσθετοι έλεγχοι για τους επιβάτες του εξωτερικού
- Αναμονή των επιβατών μέχρι της επιβιβάσεως
- Μεταφορά των επιβατών από τους χώρους αναμονής στο αεροσκάφος
- 'Εξυπηρέτηση των μέσων μεταφοράς που χρησιμοποιούνται για την μεταφορά στο αεροδρόμιο.

Σε σχέση με τους επιβάτες που ταξίδευσαν:

- Μεταφορά τους από το αεροσκάφος στο χώρο παραλαβής των αποσκευών
- Παραλαβή των αποσκευών
- Έλεγχοι για τους επιβάτες του εξωτερικού
- Χώροι αναμονής για τους επιβάτες πού θα συνεχίσουν το ταξίδι τους με το ίδιο ή άλλο αεροσκάφος
- Διακίνηση των επιβατών προς τα μέσα μεταφοράς τους για και ευκολίες για τα μέσα αυτά

Σε σχέση με τα εμπορεύματα, ταχυδρομείο κλπ, ανάλογες (όπως για τους επιβάτες) ανάγκες.

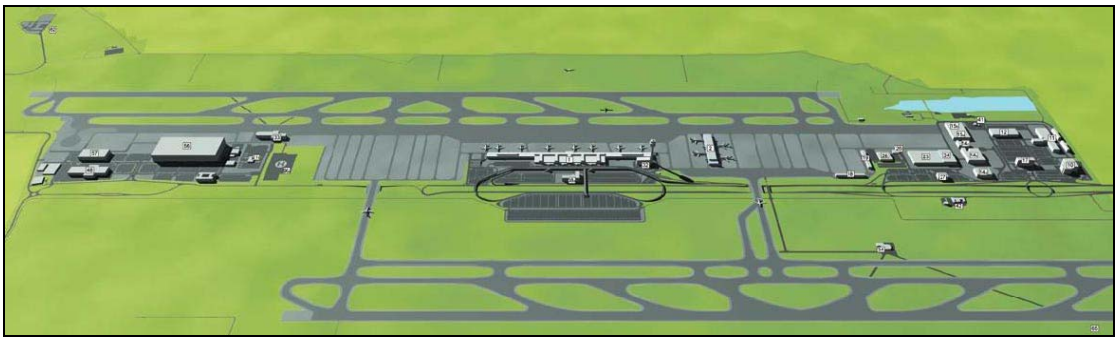
Σε σχέση με τις επιχειρήσεις πού με οποιοδήποτε τρόπο συμμετέχουν στην κάλυψη των παραπάνω αναγκών:

- 'Εξυπηρέτηση του προσωπικού των (διακίνηση, εργασία κλπ)
- 'Εξυπηρέτηση των ίδιων των επιχειρήσεων, με χώρους γραφείων κλπ.

Οι κύριες κατασκευές ενός αεροδρομίου, ή αλλιώς τα βασικά στοιχεία του πού καλύπτουν τις πιο πάνω ανάγκες είναι:

- Ο διάδρομος ή το σύστημα διαδρόμων πού εξυπηρετεί την προσγείωση και απογείωση των αεροσκαφών
- Τα δάπεδα σταθμεύσεως, όπου τα αεροσκάφη σταθμεύουν για φορτοεκφόρτωση, ανεφοδιασμό κλπ
- Το τροχοδρομικό σύστημα πού συνδέει τον διάδρομο ή το σύστημα διαδρόμων με τα δάπεδα σταθμεύσεως
- Ο επιβατικός αεροσταθμός πού εξυπηρετεί την διακίνηση των επιβατών μεταξύ των επιφανειακών δικτύων μεταφοράς και των αεροσκαφών - Ο εμπορευματικός αεροσταθμός για την διακίνηση των εμπορευμάτων
- Οι συνδέσεις του (των) αεροσταθμών, τα επιφανειακά δίκτυα μεταφοράς και οι σχετικές τερματικές εγκαταστάσεις εξυπηρετήσεως των επιφανειακών μέσων μεταφοράς
- Ο Πύργος Έλεγχου
- Λοιπές εγκαταστάσεις πού εξυπηρετούν την ασφάλεια γενικά (Πυροσβεστικός σταθμός, κλπ), την διακίνηση των επιβατών, ταχυδρομείου και εμπορευμάτων και τον ανεφοδιασμό, έλεγχους, και συντήρηση των αεροσκαφών καθώς και του υπόλοιπου μηχανικού εξοπλισμού του αεροδρομίου.

Τα παραπάνω στοιχεία αναγνωρίζονται στο σχήμα 3.1.



Σχήμα 3.1. Κάτοψη Διεθνούς Αερολιμένα Αθηνών "Ελ. Βενιζέλος"

Τα διάφορα χαρακτηριστικά του αερομεταφορικού έργου που θα εξυπηρετήσει το αεροδρόμιο, επηρεάζουν καθοριστικά τη μορφή και το μέγεθος των εγκαταστάσεων.

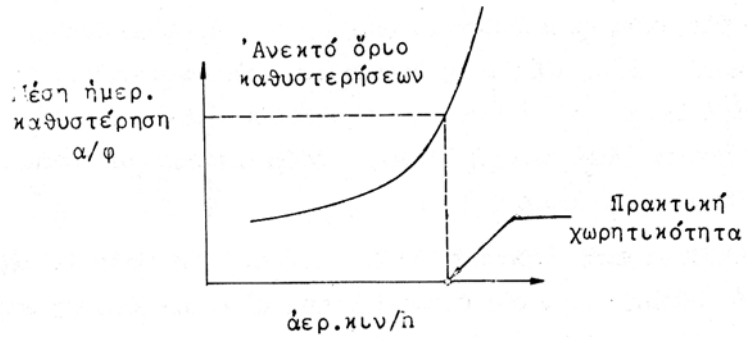
Όχι όμως μόνο το αερομεταφορικό έργο, αλλά και ο όγκος κυκλοφορίας των αεροσκαφών, δηλαδή ο αριθμός προσγειοαπογειώσεων στη μονάδα του χρόνου, αποτελεί πολύ βασική παράμετρο. Από αυτή καθώς και την κατανομή της στο χρόνο (στις ώρες της ημέρας, στους μήνες του έτους) την διαχρονική εξέλιξη της καθώς και από το επιδιωκόμενο (για το έτος-στόχο) επίπεδο εξυπηρετήσεως εξαρτιέται το πλήθος των διαδρόμων, η διάταξή τους, το τροχοδρομικό σύστημα, κλπ.

Πρέπει ακόμα να σημειωθεί ότι το αερομεταφορικό έργο αυξάνει, με τα χρόνια, ταχύτερα από ότι ο όγκος της κυκλοφορίας των αεροσκαφών (χωρητικότητα αεροδρομίου). Ο βασικός λόγος είναι η αύξηση του μεγέθους των αεροσκαφών και η επιχειρούμενη βελτίωση του συντελεστή πληρότητας.

Με τον όρο χωρητικότητα αεροδρομίου εννοούμε τον αριθμό των αεροπορικών κινήσεων που μπορεί να εξυπηρετήσει ένα αεροδρόμιο με μια μέση καθυστέρηση για τα αεροπλάνα, μικρότερη από ένα καθορισμένο όριο.

Μια αεροπορική κίνηση είναι μια προσγείωση ή μια απογείωση. Εννοείται ότι όταν οι αριθμοί των κινήσεων αυξάνονται, αυξάνεται και η καθυστέρηση που προκαλείται από τα αεροπλάνα.

Έτσι ορισμένη, η χωρητικότητα χαρακτηρίζεται και ως «πρακτική χωρητικότητα». Διαγραμματικά η έννοια περιγράφεται στο σχ.3.2. Η καμπύλη δείχνει ότι στη περιοχή της χωρητικότητας μικρή αύξηση των αεροπορικών κινήσεων συνεπάγεται ραγδαία αύξηση των καθυστερήσεων.



Σχ. 3.2. Ωριαία χωρητικότητα. [1]

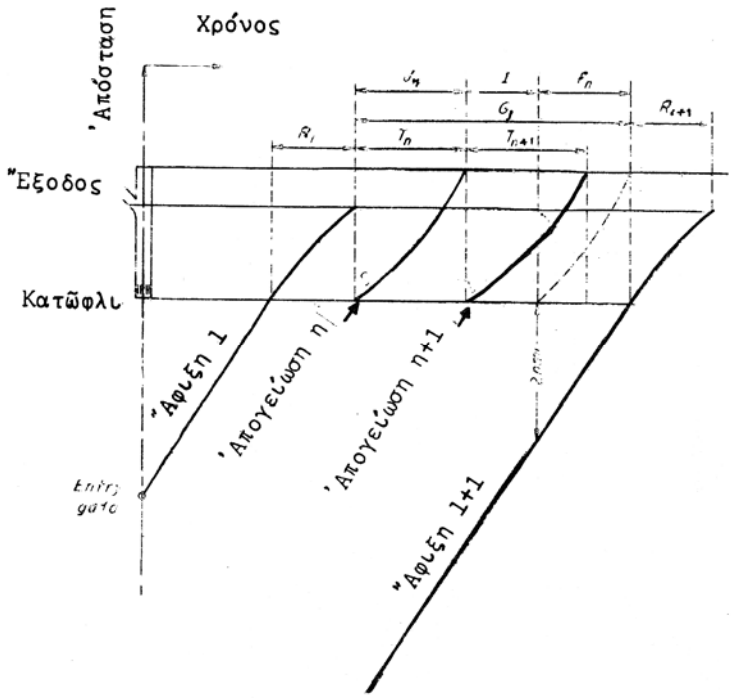
Ένας νεώτερος ορισμός, που αποχωρίζει την έννοια της καθυστερήσεως είναι η εξίσωση της χωρητικότητας προς το μέγιστο αριθμό αεροπορικών κινήσεων που μπορεί το αεροδρόμιο να εξυπηρετήσει στη μονάδα του χρόνου, υπό συνθήκες συνεχούς ζήτησης.

Συνεχής ζήτηση σημαίνει ότι αναμένει τουλάχιστον ένα αεροσκάφος που θέλει να προσγειωθεί ή απογειωθεί. Με τον ορισμό αυτό αποδίδεται καλύτερα η έννοια της «μέγιστης χωρητικότητας», ή «χωρητικότητας κορεσμού» που είναι αντίστοιχη προς την έννοια χωρητικότητας οδού.

Η χωρητικότητα εξαρτάται από το είδος, το πλήθος και τη διάταξη των διαδρόμων και επηρεάζεται από πλήθος παραμέτρων που ενδεικτικά σημειώνονται:
- η σχέση προσγειώσεων-απογειώσεων
- το σύστημα και διάταξη τροχοδρόμων που μπορεί να ελαχιστοποιεί ή όχι τον χρόνο παραμονής του αεροσκάφους στο διάδρομο
- η επιβαλλόμενη απόσταση ασφαλείας μεταξύ δύο διαδοχικών αεροσκαφών
- οι καιρικές συνθήκες γενικά
- η δυνατότητα πτήσεων όψεως (VFR) σε σχέση με τις καιρικές συνθήκες
- τα βοηθήματα αεροπλοΐας που διατίθενται και μάλιστα για πτήσεις με όργανα (IFR)
- η πείρα των ελεγκτών να καθοδηγούν μεγάλο όγκο κυκλοφορίας
- το είδος των αεροσκαφών και η μίξη τους (ποσοστιαία και διαδοχής)
- το είδος και η έκταση των υπηρεσιών του πύργου ελέγχου
- ενδεχόμενοι περιορισμοί περιβάλλοντος (θορύβου, κλπ) για τυχόν άλλους διαθέσιμους διαδρόμους.

Η πρακτική χωρητικότητα ενός αεροδρομίου υπολογίζεται με μαθηματικές μεθόδους προσομοιώσεως (θεωρία ουρών). Με αυτές υπολογίζονται οι αναμενόμενες καθυστερήσεις για διάφορες τιμές πλήθους αεροσκαφών που θέλουν να απογειωθούν ή να προσγειωθούν. Η μέση αποδεκτή καθυστέρηση που θα προσδιορίσει την πρακτική χωρητικότητα είναι 1-4min για διάφορες περιπτώσεις πτήσεως (IFR ή VFR), μίξεως, κλπ.

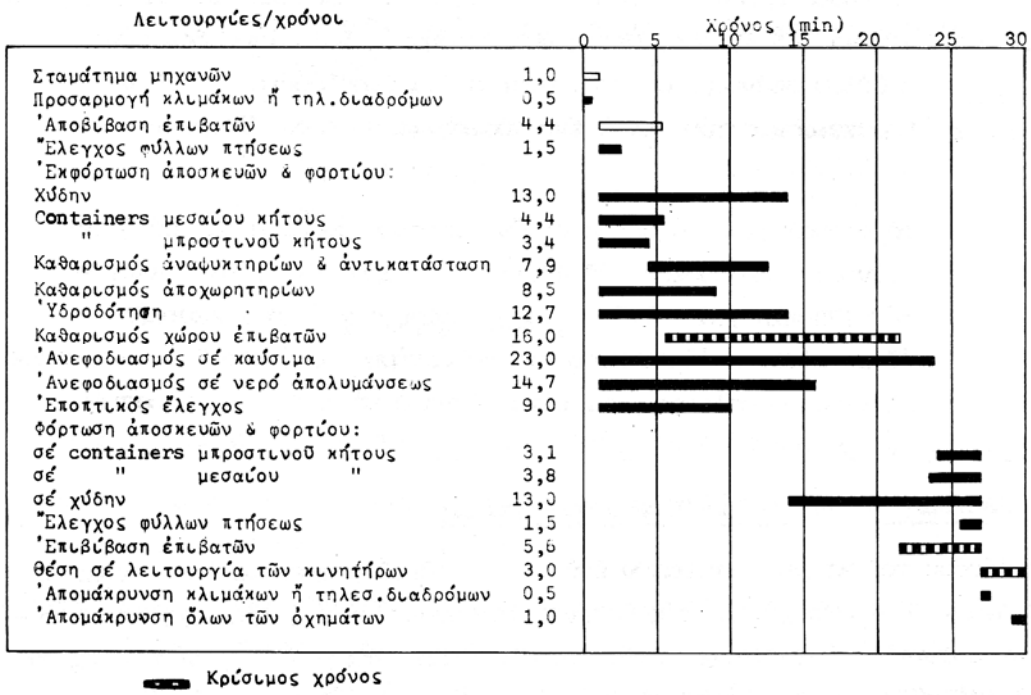
Στο διάγραμμα του σχήματος 3.3, δείχνεται γραφικά μια βαθμίδα οικοδομήσεως του μαθηματικού προτύπου στην περίπτωση αεροδρομίου ενός διαδρόμου με μία έξοδο.



Σχήμα 3.3. Τυπικό διάγραμμα χρόνου διανυόμενων αποστάσεων των αεροπορικών κινήσεων σε ένα διάδρομο. [1]

Ο χρόνος στάθμευσης ενός αεροσκάφους στον αεροσταθμό, εξαρτιέται από:
- το μέγεθος του αεροσκάφους
- το είδος της στάσεως του αεροσκάφους (ενδιάμεση ή τερματική)
- τον εξοπλισμό και τις εγκαταστάσεις για την εξυπηρέτηση του αεροσκάφους και των επιβατών που προσφέρει ο αεροσταθμός και γενικότερα το αεροδρόμιο.

Ο χρόνος στάθμευσης, προσδιορίζεται αναλυτικά με χρονοδιαγράμματα του είδους του σχήματος 3.4. Μέσα στον χρόνο αυτό πρέπει να πραγματοποιηθούν σειρά εργασιών που αφορά το αεροσκάφος (έλεγχοι, καθαρισμός, ανεφοδιασμοί, κτλ) και ακόμα η διαδικασία αποβιβάσεως των επιβατών που αφίχθηκαν και η επιβίβαση των επιβατών που θα αναχωρήσουν. Αεροσκάφη που αρχίζουν ένα ταξίδι καταλαμβάνουν τη θέση σταθμεύσεως για περισσότερο, κατά κανόνα, χρόνο από ότι για μια ενδιάμεση στάση.



Σχήμα 3.4. Τυπικό διάγραμμα χρόνου σταθμεύσεως για επιβατικά αεροσκάφη μέσου μεγέθους (~ 120 επιβ.) που εκτελεί ενδιάμεση στάση στο δρομολόγιό του. [1]

Αναφερόμενοι τώρα ειδικότερα στην διάταξη των αεροδρομίων αναφορικά με την εξυπηρέτηση των αεροσκαφών διακρίνονται τρεις ομάδες στοιχείων:

Α. Ο χώρος προσγειοαπογειώσεων (διάδρομοι, τροχόδρομοι)

Β. Ο χώρος σταθμών εξυπηρετήσεως (terminal)

Γ. Οι εγκαταστάσεις του ελέγχου της εναέριας κυκλοφορίας στο χώρο που περιβάλλει το αεροδρόμιο.

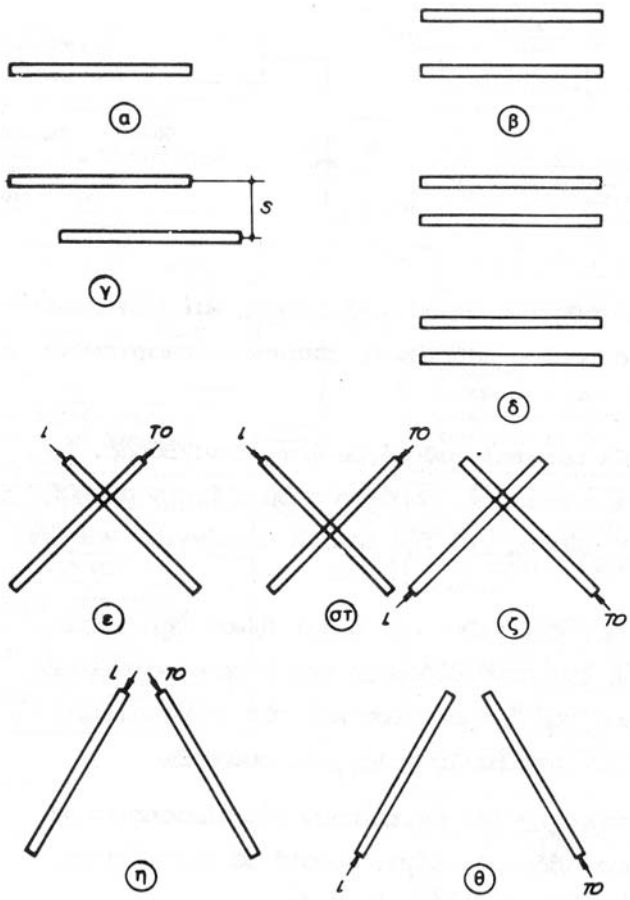
Αυτές οι τρεις ομάδες στοιχείων αποτελούν το σύστημα του αεροδρομίου. Κάθε μια μπορεί να επηρεάζει την χωρητικότητα του συστήματος και εξαρτάται από τις άλλες, επειδή πρέπει να λειτουργούν αρμονικά μεταξύ τους. Είναι, για παράδειγμα, προφανώς άχρηστο, να υπάρχει ένας χώρος υποδομής με χωρητικότητα τριπλάσια από την ικανότητα του αεροδρομίου σε προσγειώσεις-απογειώσεις.

Στο σχεδιασμό ενός αεροδρομίου, δίνεται μια προτεραιότητα στη τοποθέτηση των διαδρόμων που ελέγχεται από πλήθος παραμέτρων. Ο αριθμός των διαδρόμων

είναι συνάρτηση του όγκου κυκλοφορίας και περαιτέρω του ανεμολογίου της περιοχής του αεροδρομίου και των υφιστάμενων τοπογραφικών δυνατοτήτων.

Πολλά αεροδρόμια (με αρ. επιβατών που φθάνει ή/και υπερβαίνει τα 5 000 000 κατ'έτος) λειτουργούν με ένα μόνο διάδρομο που ο προσανατολισμός του είναι αποτέλεσμα της συνεκτιμήσεων του ανεμολογίου και των τοπογραφικών συνθηκών της περιοχής, ακόμη δε και των περιβαλλοντικών συνθηκών. Όταν η κυκλοφορία είναι πολύ μεγάλη τότε χρησιμοποιείται ζεύγος παραλλήλων διαδρόμων που μπορεί να επαυξηθεί και με $3^{ο}$ ή και $4^{ο}$ διάδρομο. Χαρακτηριστικές διατάξεις δίνονται στο σχήμα 3.5.

Σ'αυτό, οι διατάξεις α..δ αντιστοιχούν σε αεροδρόμια που η σύνθεση της κυκλοφορίας τους και οι συνθήκες ανέμων επιτρέπουν την διάταξη διαδρόμων μιας κατεύθυνσης. Η πολλαπλότητα είναι αποτέλεσμα της επιθυμίας αυξήσεως της χωρητικότητας (β,γ,δ). Οι διατάξεις ε, στ, ζ αντιστοιχούν στη περίπτωση που οι συνθήκες ανέμων επιβάλλουν διαδρόμους κατά διαφορετικές διευθύνσεις. Τότε κι όταν επικρατούν ευνοϊκές καιρικές συνθήκες μπορεί η χρησιμοποίηση των διαδρόμων να γίνεται με κατάλληλο συντονισμό των κινήσεων και χωρισμό των προσγειώσεων και απογειώσεων σε κάθε διάδρομο, ώστε τελικά η χωρητικότητα του συστήματος να αυξάνει σε σχέση με την χωρητικότητα του κάθε διαδρόμου μόνου του. Οι διατάξεις η και θ αντιστοιχούν σε ανάλογη περίπτωση, προϋποθέτουν όμως κατά κανόνα μεγαλύτερη επιφάνεια και έχουν βελτιωμένη χωρητικότητα σε σχέση με τις διατάξεις ε, στ και ζ.



Σχήμα 3.5. Χαρακτηριστικές διατάξεις αεροδρομίων (L: προσγείωση, ΤΟ: απογείωση) [1]

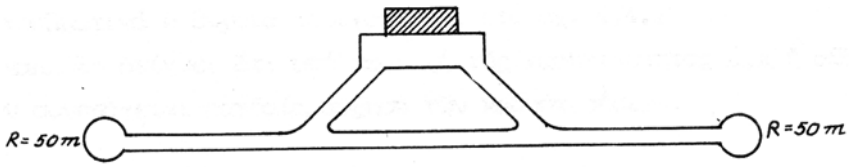
Ο αεροσταθμός και οι συναφείς εγκαταστάσεις διατάσσονται στη συνέχεια, έτσι ώστε εκτός των άλλων, να τηρούνται οι κανόνες ασφαλείας αλλά και να μειώνεται στο ελάχιστο η διαδρομή του αεροσκάφους.

Προ του αεροσταθμού θα πρέπει να υπάρχει το δάπεδο σταθμεύσεως των αεροσκαφών όπου διεξάγονται η αποβίβαση των επιβατών, η επιβίβαση των επιβατών, ο έλεγχος και ο πάσης φύσεως ανεφοδιασμός του αεροσκάφους.

Η σύνδεση των διαδρόμων με το δάπεδο σταθμεύσεως γίνεται με την βοήθεια των τροχοδρόμων. Η κυριότερη λειτουργία των τροχοδρόμων είναι να εξυπηρετούν την κυκλοφορία μεταξύ των διαδρόμων και του χώρου του αεροσταθμού ή των υπόστεγων επισκευών.

Οι τροχόδρομοι σχεδιάζονται έτσι ώστε να ελαχιστοποιείται ο χρόνος χρησιμοποιήσεως του ή των διαδρόμων. Συνήθως, τροχόδρομοι σε μήκος ίσο και παράλληλο με τον διάδρομο αποτελούν την λύση σε αεροδρόμια με μεγάλη κυκλοφορία.

Σε αεροδρόμια με μικρή κυκλοφορία οι τροχόδρομοι είναι δυνατόν να παραλείπονται (π.χ. αεροδρόμια πολλών μικρών νησιών και επαρχιακών πόλεων). Στη περίπτωση αυτή τα άκρα του διαδρόμου μορφώνονται με κυκλική (ή ημικυκλική) διαπλάτυνση όπου το αεροσκάφος μπορεί να στραφεί κατά 180^{o} και να κινηθεί επί του διαδρόμου (μετά τη προσγείωση – και αντίστροφα προ της απογειώσεως) σαν ο διάδρομος να ήταν τροχόδρομος (σχ. 3.6). Το σύστημα τροχόδρομων περιορίζεται σε μικρούς και συνδετήριους κλάδους.



Σχήμα 3.6. Σχηματική διάταξη αεροδρομίου χωρίς τροχοδρόμους. [1]

Στη διάρκεια των ωρών αιχμής, σημαντικό στοιχείο για την χωρητικότητα του αεροδρομίου, είναι το πόση ώρα μπορεί ένα αεροπλάνο που προσγειώνεται να εγκαταλείψει τον διάδρομο. Γι' αυτό πρέπει να προβλέπονται πολλές έξοδοι και μάλιστα σε γωνία, ως προς τον διάδρομο, που να επιτρέπει την έξοδο του αεροσκάφους με μεγάλη ταχύτητα.

Εξ' άλλου οι τροχόδρομοι πρέπει να βραχύνουν την διάνυση του αεροσκάφους από το σημείο που θα εγκαταλείψει τον διάδρομο μέχρι το δάπεδο σταθμεύσεως.

Για τα απογειούμενα αεροσκάφη το σύστημα των τροχοδρόμων πρέπει να τα εξυπηρετεί κατά τρόπο που με τη μικρότερη διανυόμενη απόσταση και χωρίς να διασταυρώνουν διάδρομο να προσεγγίζουν το άκρο του διαδρόμου που θα χρησιμοποιήσουν.

Έτσι πολλές φορές κάθε διάδρομος συνοδεύεται από 2 ή και 3 τροχοδρόμους παράλληλους. Επιπλέον, και αυτό κατά κανόνα, στη σύνδεση του τροχόδρομου με το άκρο του διαδρόμου παρεμβάλλεται δάπεδο στάσεως αναμονής του αεροσκάφους από όπου το αεροσκάφος μπορεί να ξεκινήσει υπό γωνία ως προς τον άξονα του διαδρόμου και επιταχύνοντας να εγγραφεί στον άξονα για την απογείωση. Τέτοια διαμόρφωση μειώνει το χρόνο καταλήψεως του διαδρόμου από το αεροσκάφος.

Οι τροχοδρόμοι διατάσσονται κατά τρόπο που να επιτρέπει:
- τη μείωση της διαδρομής και του χρόνου μεταβάσεως του αεροσκάφους από τον διάδρομο στο δάπεδο σταθμεύσεως (και αντιστρόφως)
- τη μετακίνηση των αεροσκαφών επ' αυτών χωρίς καμία παρενόχληση των επί των διαδρόμων προσγειοαπογειώσεων
- την ελαχιστοποίηση του χρόνου χρησιμοποιήσεως του διαδρόμου από κάθε αεροσκάφος.[1]

3.2 Περιγραφή προβλήματος

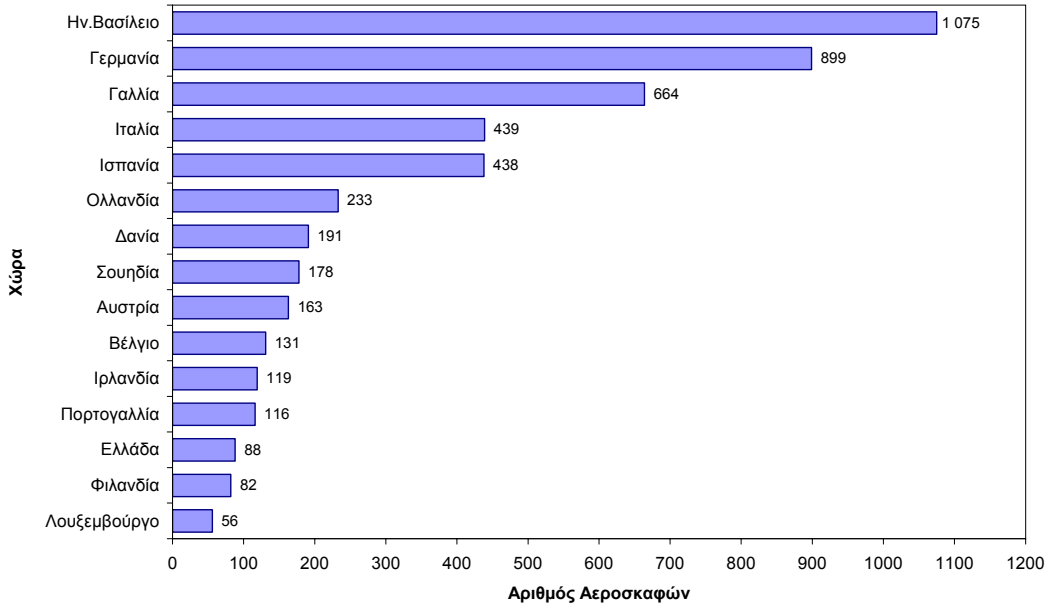
Η αεροπορική βιομηχανία έχει ήδη αναπτυχθεί με έναν πολύ ταχύ ρυθμό, υψηλότερο από όλες τις άλλες μορφές μεταφορών. Διαφαίνεται, δε, ότι η μελλοντική ανάπτυξή της θα είναι ακόμα ταχύτερη. Είναι αυτονόητο, ότι όσο μεγαλύτερη είναι η αεροπορική βιομηχανία τόσο σοβαρότερο και το αντίκτυπό της στο περιβάλλον.

Μεταξύ του 1970 και του 1995 ο αριθμός των χιλιομέτρων που διανύθηκαν παγκοσμίως από επιβάτες αυξήθηκε από 551 σε 2537 δισεκατομμύρια, αύξηση της τάξης του 360%. Η μεταφορά φορτίων αυξήθηκε ακόμα περισσότερο. Μεταξύ του 1960 και 1995, η αύξηση στα τονοχιλιόμετρα παγκοσμίως ήταν 2200%. [4]

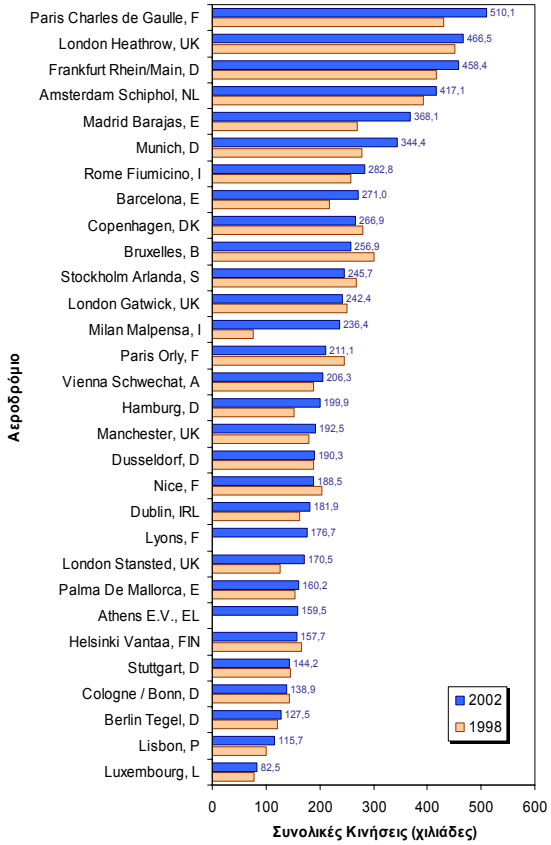
Αναφορικά τώρα με την αεροπορική δραστηριότητα στην Ευρωπαϊκή ένωση, δίνεται στο σχήμα 3.7 ο αριθμός των αεροσκαφών που επιχειρούν στον χώρο της, ενώ στο σχήμα 3.8 παρουσιάζεται η συνολική κίνηση των αεροσκαφών ανά ευρωπαϊκό αεροδρόμιο.

Διαφαίνεται ότι η πρώτη χώρα σε αριθμό αεροσκαφών είναι το Ηνωμένο Βασίλειο, με την Γερμανία και την Γαλλία να ακολουθούν με αισθητή διαφορά.

Ωστόσο, από πλευράς κίνησης αεροσκαφών στα αεροδρόμια των χωρών αυτών, οι διαφορές μεταξύ τους δεν είναι μεγάλες.

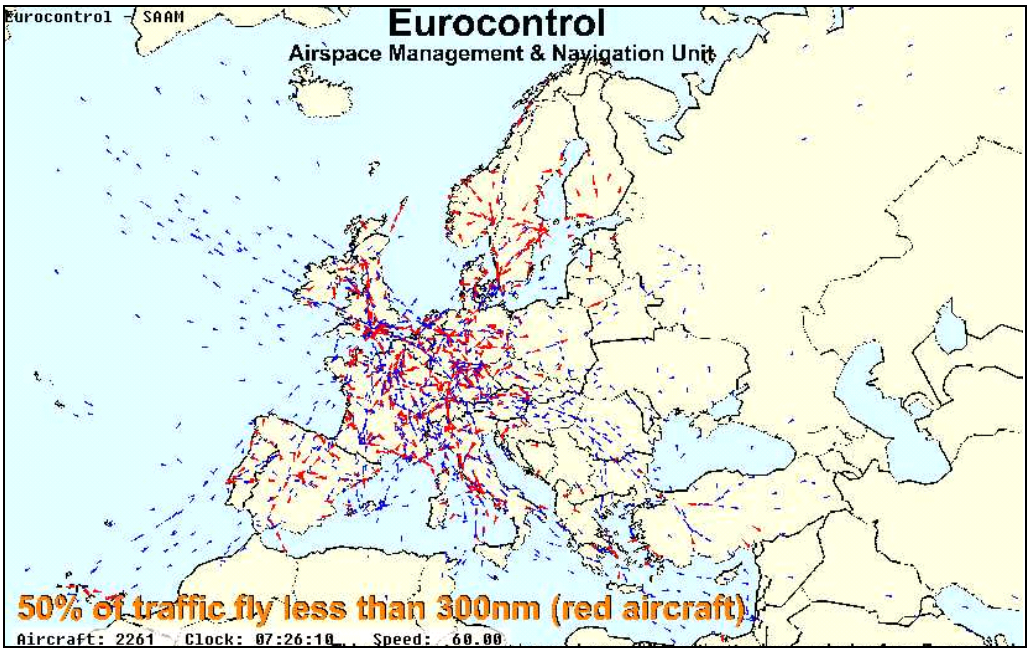


Σχήμα 3.7. Ευρωπαϊκός στόλος αεροσκαφών, έτος 2002.



Σχήμα 3.8. Συνολική κίνηση αεροσκαφών ανά ευρωπαϊκό αεροδρόμιο.[7]

Στο σχήμα 3.9 δίνεται μια απεικόνιση της τυπικής κυκλοφορίας, στον Ευρωπαϊκό εναέριο χώρο, όπως αυτή παρουσιάζεται στο Ευρωπαϊκό Κέντρο Ελέγχου Εναέριας Κυκλοφορίας (Eurocontrol). Παρουσιάζονται συνολικά 2261 αεροσκάφη όπου με κόκκινο χρώμα δηλώνονται τα αεροσκάφη που εκτελούν πτήση απόστασης κάτω των 300nm.



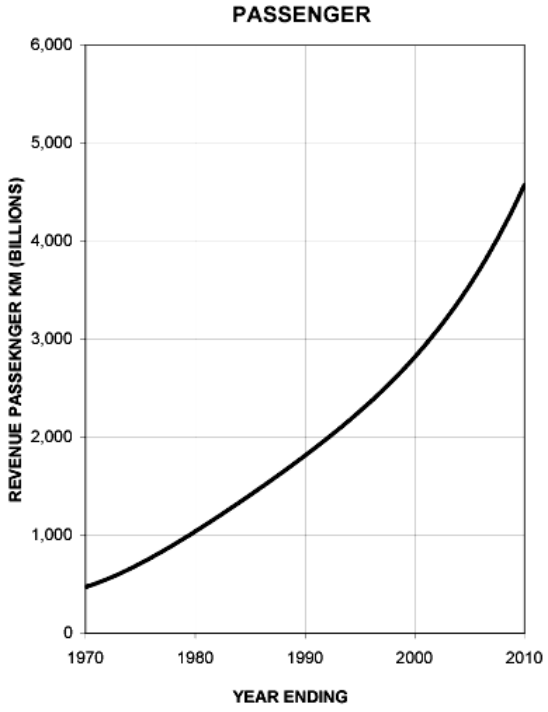
Σχήμα 3.9. Απεικόνιση τυπικής αεροπορικής κυκλοφορίας, στον Ευρωπαϊκό εναέριο χώρο.[5]

Όσον αφορά στην μελλοντική ανάπτυξη των αερομεταφορών, η Boeing, μία από τις μεγαλύτερες εταιρίες κατασκευής αεροσκαφών παγκοσμίως, έδωσε δύο ανάλογα διαγράμματα. Έτσι, στο σχήμα 3.10, δίνονται οι τυπικές προβλέψεις των προσοδοφόρων επιβατοχιλιομέτρων, μέχρι το έτος 2010. Ακολουθώντας μια περίοδο ύφεσης το 2001 και δύο διαδοχικά έτη σταθερότητας, η παγκόσμια επιβατική αερομεταφορική κυκλοφορία προβλέπεται να ανακάμψει με ένα ετήσιο ρυθμό περίπου 4%. Πιο συγκεκριμένα, η μεγαλύτερη ανάπτυξη αναμένεται να πραγματοποιηθεί εκεί όπου οι ρυθμοί οικονομικής ανάπτυξης είναι ταχείς, όπως η Ασία, η ανατολική Ευρώπη και σε κάποιο βαθμό η Λατινική Αμερική.

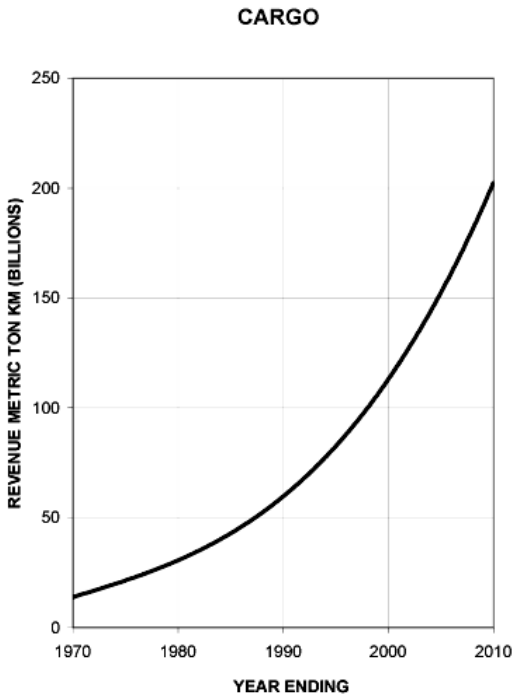
Όσον αφορά στη μεταφορά φορτίων, σχήμα 3.11, ακόμα και με την ύφεση που παρουσιάστηκε το 2001, η τυπική πρόβλεψη υποδεικνύει αύξηση της τάξης του 50%, στα παγκοσμίως προσοδοφόρα ICAO τονοχιλιόμετρα, μεταξύ του 2003 και

2010. Διαφαίνεται δηλαδή ότι η μεταφορά φορτίων μέχρι το 2010, θα παρουσιάσει ταχύτερους ρυθμούς αύξησης από την αντίστοιχη επιβατική μεταφορά. Ένας λόγος είναι ότι οι παραδοσιακές μορφές τροφοδοσίας, όπου η τοπική ζήτηση ικανοποιείται από την τοπική προσφορά, θα δώσουν την θέση τους σε παγκόσμιες γραμμές τροφοδοσίας. Ένας επιπλέον λόγος είναι ότι δύο από τις πιο πυκνοκατοικημένες χώρες, η Κίνα και η Ινδία, οδηγούνται προς ισχυρά απελευθερωμένες μορφές οικονομικής δραστηριότητας.

Θα πρέπει να σημειωθεί ότι οι συγκεκριμένες προβλέψεις πραγματοποιήθηκαν το 2002, λαμβάνοντας υπόψη τις αρνητικές επιδράσεις του τρομοκρατικού χτυπήματος κατά των ΗΠΑ, την 11^{η} Σεπτεμβρίου.



Σχήμα 3.10. Πρόβλεψη αύξησης επιβατοχιλιομέτρων παγκοσμίως. [8]



Σχήμα 3.11. Πρόβλεψη αύξησης τονοχιλιομέτρων παγκοσμίως. [8]

Η συνεχώς αυξανόμενη εναέρια κυκλοφορία σε παγκόσμιο επίπεδο έχει ως αποτέλεσμα την αύξηση της συμφόρησης και των καθυστερήσεων στα αεροδρόμια. Ο μέσος χρόνος ταξιδιού (από στάθμευση σε στάθμευση) μεταξύ μεγάλων πόλεων είναι ολοένα αυξανόμενος. Για παράδειγμα ο κατά μέσο όρο χρόνος από πύλη σε πύλη από το αεροδρόμιο της Βοστόνης στο εθνικό αεροδρόμιο της Ουάσιγκτον αυξήθηκε κατά 20% από το 1973 ως το 1994.

Η μεγαλύτερη κυκλοφοριακή συμφόρηση του U.S. National Airspace System (NAS) εμφανίζεται στα αεροδρόμια. Ακόμα και στην περίπτωση ιδανικών καιρικών συνθηκών η δυναμικότητα των αφίξεων και αναχωρήσεων μειώνεται δραματικά, ενώ ταυτόχρονα οι αερογραμμές, αδυνατώντας να μειώσουν τη ζήτηση, προβαίνουν στην ακύρωση πτήσεων. Η μειωμένη ικανότητα αναχωρήσεων μπορεί να οδηγήσει σε μεγάλους χρόνους τροχοδρόμησης εξόδου στις ώρες αιχμής, καθώς το αεροσκάφος περιμένει αρκετό χρόνο στην ουρά πριν την απογείωσή του. Αυτοί οι μεγάλοι χρόνοι τροχοδρόμησης όχι μόνο αυξάνουν τα άμεσα λειτουργικά κόστη, αλλά αυξάνουν ταυτόχρονα το θόρυβο και τις ρυπογόνες εκπομπές στην περιοχή του αεροδρομίου.

Είναι εμφανές ότι υπάρχει έντονη επιθυμία να αναπτυχθεί ένας μηχανισμός που θα μειώνει τη συμφόρηση και τις ουρές αναχωρήσεων. Η αύξηση της ικανότητας των αεροδρομίων προσθέτοντας νέους διαδρόμους προσγείωσης - απογείωσης θα απαιτούσε υψηλό οικονομικό και πολιτικό κόστος. Για το λόγο αυτό η έρευνα στρέφεται στις επιχειρησιακές βελτιώσεις του υπάρχοντος συστήματος με την ανάπτυξη μοντέλων προσομοίωσης και ελέγχου της λειτουργίας των αεροδρομίων.[6]

Οι περισσότερες μελέτες για το σύστημα αερομεταφορών δεν λαμβάνουν υπ' όψιν διεργασίες εδάφους. Πράγματι, σε πολλά μοντέλα ο χρόνος εδάφους, ο οποίος περιλαμβάνει όλες τις διαδικασίες και τις δραστηριότητες από στάθμευση σε στάθμευση, θεωρείται σταθερός. Η υπόθεση αυτή αγνοεί τα φαινόμενα ουρών που εμφανίζονται στα αεροδρόμια και που οδηγούν τις αερογραμμές στις πολλαπλασιαζόμενες καθυστερήσεις κατά τη διάρκεια που τα αεροσκάφη είναι στο έδαφος. Ωστόσο, στην πράξη οι αερογραμμές συχνά προσπαθούν να μειώσουν τους χρόνους εδάφους των καθυστερημένων αεροσκαφών, προκειμένου να ελέγχουν τις αρνητικές συνέπειες των καθυστερήσεων.

3.3 Ανασκόπηση μεθόδων-μοντέλων προσέγγισης

Η κατασκευή απλών και ορθών μοντέλων για τα αεροδρόμια με τερματικούς σταθμούς (hub airports) μπορεί να αποτελέσει σημαντικό εφόδιο για την κατανόηση της δυναμικής των αεροδρομίων και να παρέχει σημαντικές εκτιμήσεις για την αναβάθμιση των επιχειρήσεων των αεροδρομίων.

Τα κυριότερα μοντέλα προσομοίωσης της λειτουργίας των αεροδρομίων όσον αφορά την εξυπηρέτηση των προσγειοαπογειούμενων αεροσκαφών είναι τα ακόλουθα:

- Μοντέλα Υψηλής Λεπτομέρειας ή μικροσκοπικά
- Μοντέλα Προσέγγισης Καθυστερήσεων Δικτύου
- Μεσοσκοπικά Μοντέλα Αεροδρομίων

Μοντέλα Υψηλής Λεπτομέρειας ή μικροσκοπικά

Τέτοια μοντέλα είναι το SIMMOD ή το TAAM, που αναπαράγουν με μεγάλη λεπτομέρεια τη δομή του αεροδρομίου, τους επιχειρησιακούς κανόνες και τη δυναμικότητα κάθε πύλης, τροχοδρόμου και διαδρόμου απογείωσης-προσγείωσης για κάθε τύπο αεροσκάφους.

Τα μοντέλα αυτά είναι χρήσιμα για τον έλεγχο των διαδικαστικών αλλαγών στα δρομολόγια των αεροσκαφών στα συστήματα τροχοδρόμησης. Το μειονέκτημα αυτών των μοντέλων είναι η δυσκολία και το υψηλό κόστος προκειμένου να έχουμε ικανοποιητικά και αξιόπιστα δεδομένα για όλα τα επίπεδα και τη σχετική διάταξη τμημάτων του αεροδρομίου.

Είναι λοιπόν αρκετά δύσκολο να έχουμε από αυτά τα μοντέλα γρήγορους και αξιόπιστους υπολογισμούς από τα πλεονεκτήματα των νέων επιχειρησιακών αρχών στην κλίμακα του αεροδρομίου για πολύ μεγάλο χρονικό διάστημα.

Μοντέλα Προσέγγισης Καθυστερήσεων Δικτύου

Τα μοντέλα αυτά λαμβάνουν μια μακροσκοπική προοπτική της ικανότητας και των απαιτήσεων του αεροδρομίου κατά τη διάρκεια μιας ημέρας και παρέχουν υπολογισμούς των καθυστερήσεων.

Αυτά τα μοντέλα μας επιτρέπουν τη μελέτη της διάδοσης των καθυστερήσεων σε επίπεδο εθνικού εναέριου συστήματος, αλλά η μακροσκοπική μελέτη δεν παρέχει αρκετές λεπτομέρειες για τις επιμέρους επιχειρήσεις των αεροδρομίων προκειμένου να μελετήσουμε ένα πλάνο ενεργειών για μείωση των χρόνων τροχοδρόμησης εξόδου.

Μεσοσκοπικά Μοντέλα Αεροδρομίων

Σε αυτά, τα μοντέλα εισόδου-εξόδου από τα τερματικούς σταθμούς του αεροδρομίου, τα συστήματα τροχοδρόμησης και τα συστήματα των διαδρόμων προσγείωσης-απογείωσης συνυπολογίζονται και δημιουργούν ένα μεσοσκοπικό μοντέλο αεροδρομίων.

Αυτά τα μοντέλα αναλύουν τη διαδικασία αναχώρησης με μεγάλη λεπτομέρεια και υπολογίζουν την αποτελεσματικότητα των πλάνων ελέγχου αναχωρήσεων μειώνοντας έτσι τους χρόνους τροχοδρόμησης εξόδου. Ταυτόχρονα η απλότητα των μοντέλων αυτών μας επιτρέπουν γρήγορη βαθμονόμηση και επικύρωση σε κάθε επιμέρους τμήμα των διαδρόμων προσγείωσης-απογείωσης. Δύο από τα μοντέλα αυτής της κατηγορίας είναι τα μοντέλα του Shumsky και τα μοντέλα του Hebert.

Τα μοντέλα του Shumsky τα οποία αποτελούν ντετερμινιστικά μοντέλα που προβλέπουν τους χρόνους απογείωσης των πτήσεων για την πλειοψηφία των αεροδρομίων. Μερικά από τα μοντέλα αυτά αναπαριστούν το σύστημα διαδρόμων προσγείωσης-απογείωσης υπολογιστή εξυπηρέτησης δικτύων ουρών, του οποίου η ικανότητα είναι σταθερή για διαστήματα 10 λεπτών της ώρας. Σε αυτά τα μοντέλα το αεροσκάφος προσεγγίζει την ουρά του διαδρόμου προσγείωσης-απογείωσης στο τέλος μιας ονομαστικής διαδρομής στο σύστημα τροχοδρόμησης. Ο Shumsky παρατήρησε επίσης μία σχέση ανάμεσα στη συμφόρηση του αεροδρομίου και στην εκτίμηση αναχωρήσεων του αεροδρομίου, το οποίο αποτελεί τη βάση μιας απλής στρατηγικής ελέγχου αναχωρήσεων.

Τα μοντέλα του Hebert που αναφέρονται στη διαδικασία αναχωρήσεων στο αεροδρόμιο La Guardia και βασίζονται σε δεδομένα 5 ημερών προκειμένου να προβλέπονται οι καθυστερήσεις αναχωρήσεων. Σε αυτά τα μοντέλα η ζήτηση αναχωρήσεων είναι μία μη-ομογενής διαδικασία Poisson και οι χρόνοι τροχοδρόμησης εξόδου μοντελοποιούνται ως το άθροισμα του ονομαστικού χρόνου στην ουρά του διαδρόμου προσγείωσης-απογείωσης και του χρόνου συντήρησης στο διάδρομο προσγείωσης-απογείωσης. [6]

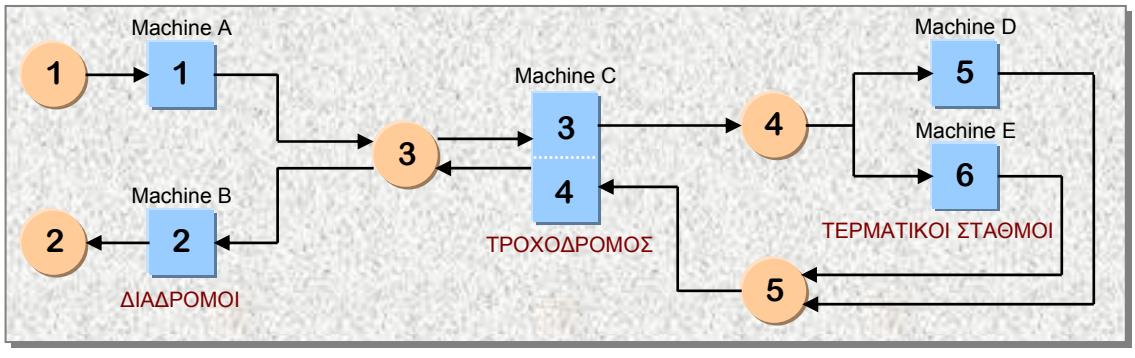
ΚΕΦΑΛΑΙΟ 4. Μοντελοποίηση

4.1 Περιγραφή μοντέλου

Ο αερολιμένας που μοντελοποιείται στην παρούσα εργασία αποτελείται από έναν διάδρομο προσγείωσης, έναν διάδρομο απογείωσης, έναν τροχόδρομο που εξυπηρετεί από κοινού τα προσγειωθέντα και τα προς απογείωση αεροσκάφη καθώς και από δύο ξεχωριστές θέσεις εξυπηρέτησης αεροσκαφών.

Ο αερολιμένας αυτός μοντελοποιείται με την χρήση μηχανών (machines) και προσωρινών θέσεων αποθήκευσης/αναμονής (buffers), σχήμα 4.1. Η κάθε machine αντιπροσωπεύει και από μία διαδικασία εξυπηρέτησης του αεροσκάφους (π.χ. προσγείωση, απογείωση, τροχοδρόμηση, επίγεια εξυπηρέτηση, κτλ.). Είναι δυνατό μία μηχανή να μπορεί να εκτελεί δύο διαφορετικές διεργασίες (submachines), ωστόσο μια κάθε φορά εκτελείται.

Η κάθε μηχανή μπορεί να εξυπηρετήσει ένα αεροσκάφος την φορά και χαρακτηρίζεται από τον χρόνο που απαιτείται για να εκτελέσει την διαδικασία της. Ο κάθε buffer αντιπροσωπεύει τους ενδιάμεσους χώρους αναμονής του αεροσκάφους πριν η μετά την εκτέλεση κάποιας διαδικασίας. Ο κάθε buffer μπορεί να χωρέσει περισσότερα από ένα αεροσκάφη για όσο χρόνο χρειάζεται και χαρακτηρίζεται από την χωρητικότητά του.



Σχήμα 4.1. Μοντελοποίηση αερολιμένα με την χρήση μηχανών και προσωρινών θέσεων αποθήκευσης/αναμονής.

Ακολουθώντας την λογική ροή του σχήματος 4.1, μπορούμε να περιγράψουμε την διαδικασία που αναπαριστά η κάθε machine και ο κάθε buffer. Τα αεροσκάφη αφικνούνται στον εναέριο χώρο του αεροδρομίου (buffer 1) με ένα δεδομένο ωριαίο ρυθμό αφίξεων. Η χωρητικότητα του buffer 1 είναι άπειρη καθώς θεωρείται ότι ο εναέριος χώρος του αεροδρομίου μπορεί να δεχθεί άπειρα αεροσκάφη. Έπειτα, τα αφιχθέντα αεροσκάφη προσγειώνονται στον αερολιμένα (machine A, submachine 1). Καθώς εξέρχονται του διαδρόμου προσγείωσης εισέρχονται στον προσωρινό χώρο αναμονής (buffer 3) πριν εισέλθουν στον τροχόδρομο (machine C, submachine 3). Ο τροχόδρομος είναι κοινός τόσο για τα προσγειωθέντα (submachine 3) όσο και για τα προς απογείωση αεροσκάφη (submachine 4) και έτσι μπορεί να εξυπηρετηθεί ένα αεροσκάφος ανά κατεύθυνση.

Τα προσγειωθέντα αεροσκάφη μετά την έξοδό τους από τον τροχόδρομο εισέρχονται πάλι σε ένα προσωρινό χώρο αναμονής (buffer 4) και παραμένουν εκεί μέχρι να ελευθερωθεί μία από τις δύο διαφορετικές θέσεις εξυπηρέτησης (machine D, machine E) όπου και προωθούνται. Μετά την πάροδο του χρόνου εξυπηρέτησης, τα αεροσκάφη οδηγούνται στον χώρο αναμονής (buffer 5) όπου και αναμένουν μέχρι να ελευθερωθεί ο τροχόδρομος (machine C, submachine 4) που οδηγεί στον χώρο αναμονής για απογείωση (buffer 3). Εκεί αναμένουν μέχρι να ελευθερωθεί ο διάδρομος απογείωσης (machine B). Μετά την απογείωση τα αεροσκάφη εισέρχονται πάλι στον εναέριο χώρο του αεροδρομίου (buffer 2), ο οποίος έχει πάλι άπειρη χωρητικότητα.

Η συγκεκριμένη διαμόρφωση του αερολιμένα έχει τα χαρακτηριστικά που δίνονται στον πίνακα 4.1. Ο Buffer 1 είναι ο buffer εισόδου ενώ ο Buffer 2 είναι ο buffer εξόδου. Η διαδρομή ακολουθεί τους buffers 1→3→4→5→3→2. Η αλληλοσύνδεση των κελιών δίνεται από τον πίνακα 4.2.

Πίνακας 4.1. Διαμόρφωση αερολιμένα και αλληλοσύνδεση machines-buffers

Πλήθος Machines	Πλήθος Buffers
M=5	B=5
Πλήθος submachines ανά machine	**Αριθμός submachine ανά machine**
N(A)=1 N(B)=1 N(C)=2 N(D)=1 N(E)=1	O(A)={1} O(B)={2} O(C)={3,4} O(D)={5} O(E)={6}
Αριθμός buffer εισόδου ανά submachine	**Αριθμός buffer εξόδου ανά submachine**
$B_+(1)=\{1\}$ $B_+(2)=\{3\}$ $B_+(3)=\{3\}$ $B_+(4)=\{5\}$ $B_+(5)=\{4\}$ $B_+(6)=\{4\}$	$B_-(1)=\{3\}$ $B_-(2)=\{2\}$ $B_-(3)=\{4\}$ $B_-(4)=\{3\}$ $B_-(5)=\{5\}$ $B_-(6)=\{5\}$
Αριθμός submachine εξόδου ανά buffer	**Πλήθος submachines εξόδου ανά buffer**
$M_-(1)=\{1\}$ $M_-(2)=\{\emptyset\}$ $M_-(3)=\{3,2\}$ $M_-(4)=\{5,6\}$ $M_-(5)=\{4\}$	$N_M(1)=1$ $N_M(2)=0$ $N_M(3)=2$ $N_M(4)=2$ $N_M(5)=1$

Πίνακας 4.2. Αλληλοσύνδεση κελιών.

BUFFER	PRECEDING BUFFER	INPUT SUBMACHINE	OUTPUT SUBMACHINE
1	-	-	1
2	3	2	-
3	1,2	1,4	3,2
4	3	3	5,6
5	4	5,6	4

4.2 Υλοποίηση προσομοίωσης

Η προσομοίωση της λειτουργίας του αερολιμένα πραγματοποιήθηκε για δύο διαφορετικούς τρόπους ελέγχου, έτσι ώστε να είναι δυνατή η μετέπειτα σύγκριση της αποτελεσματικότητάς τους.

Ο πρώτος τρόπος είναι με σειριακό έλεγχο, όπου σε κάθε χρονικό βήμα ελέγχεται η κατάσταση και γίνεται η κατάλληλη ενέργεια σε όλες τις μηχανές και τους buffers από την είσοδο έως την έξοδο του συστήματος. Ο έλεγχος δηλαδή ακολουθεί την λογική ροή επεξεργασίας ενός αεροσκάφους από τον buffer 1 στον buffer 2. Έτσι, δίνεται προτεραιότητα στην λειτουργία της υπομηχανής 1 έναντι της 2 και της υπομηχανής 3 έναντι της 4.

Ο δεύτερος τρόπος ελέγχου έγκειται στην χρήση κατάλληλου κριτηρίου κόστους [11], για την επιλογή εκείνης της υπομηχανής στη μηχανή C (3 ή 4) που θα πρέπει να λειτουργήσει στην περίπτωση που και οι δύο υπομηχανές στην μηχανή πληρούν τις προϋποθέσεις λειτουργίας.

Για μία συγκεκριμένη μηχανή *m* που αποτελείται από *N(m)* υπομηχανές, το κριτήριο υπολογίζεται για κάθε διαδρομή που χρησιμοποιεί οποιαδήποτε υπομηχανή $s \in O(m)$. Ο υπολογισμός της τιμής του κριτηρίου βασίζεται στην διαθεσιμότητα του ημιτελούς προϊόντος, (Work-In-Progress, WIP) καθώς και στο σφάλμα ελέγχου του buffer εξόδου της διαδρομής. Σύμφωνα με αυτή την προσέγγιση ελέγχου, η διαδρομή που πρόκειται να εξυπηρετηθεί είναι αυτή με την μεγαλύτερη τιμή του κριτηρίου. Η τιμή του κριτηρίου για κάθε υπομηχανή *s* ενός μιας συγκεκριμένης μηχανής *m* με $s \in O(m)$ δίνεται από την εξής σχέση:

$$J_s = \lambda_1^- f\left(x_1^-\right) + \lambda_2^- f\left(x_2^-\right) + \ldots + \lambda_{N_B(s)}^- f\left(x_{N_B(s)}^-\right) + \lambda_1^+ g\left(x_1^+\right) + \lambda_2^+ g\left(x_2^+\right) + \ldots + \lambda_N^+ \frac{e^2}{1+e^2} \quad (1)$$

όπου x_i^-, $i=1,2,\ldots,N_B(s)$ είναι τα επίπεδα των buffer που προηγούνται της υπομηχανής *s* και x_i^+, $i=1,2,\ldots,N$ είναι τα επίπεδα των buffer που ακολουθούν στην ίδια διαδρομή, ενώ το *N* υποδηλώνει το πλήθος των υπομηχανών που ακολουθούν την υπομηχανή *s* κατά την διαδρομή της, συμπεριλαμβάνοντας τον buffer εξόδου. Το επίπεδο του τελευταίου υποδηλώνεται με $\in$. Οι παράμετροι λ_i^- και λ_i^+ είναι συντελεστές βαρύτητας που πρέπει να προσδιορισθούν. Η εξάρτηση του κάθε συντελεστή από το *s* δεν λαμβάνεται υπόψη για λόγους απλότητας.

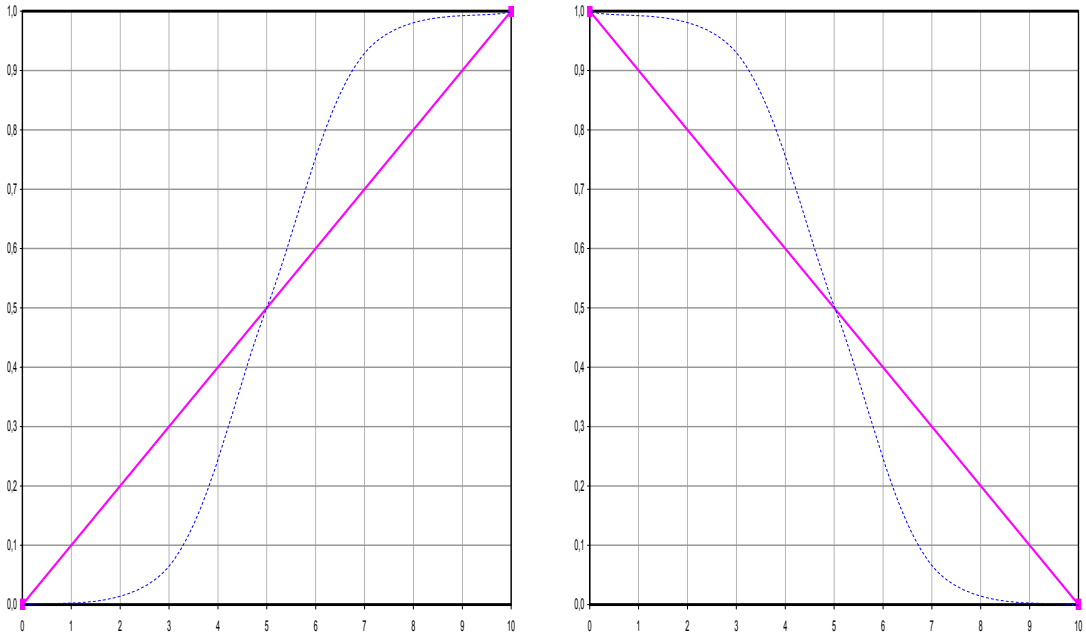
Στον παραπάνω ορισμό, η *f(.)* είναι μια θετική, μονοτονικά αύξουσα μη-γραμμική συνάρτηση με *f(0)=0* και *f(c)=1*, όπου το *c* αντιπροσωπεύει την αντίστοιχη χωρητικότητα του buffer. Είναι προφανές ότι αυτή η συνάρτηση παίρνει την μεγαλύτερη τιμή της για την περίπτωση μεγάλης συσσώρευσης αντικειμένων στον buffer που τροφοδοτεί. Ομοίως, η *g(.)* είναι μια κατοπτρική εκδοχή της *f(.)*, με *g(0)=1* και *g(c)=0*. Αυτή η συνάρτηση παίρνει την μέγιστη τιμής της εάν οι ακολουθούντες buffers είναι σχεδόν άδειοι. Με αυτό τον τρόπο, συνθήκες όπως όταν ο τροφοδοτών buffer είναι σχεδόν άδειος και/ή ο ακολουθούντες buffers είναι κοντά στην μέγιστη χωρητικότητά τους οδηγούν σε μικρό *J*, το οποίό με την σειρά του σημαίνει ότι η αντίστοιχη διαδρομή δεν θα εξυπηρετηθεί.

Οι συναρτήσεις *f(.)* και *g(.)* μπορούν να προσεγγισθούν σε ικανοποιητικό βαθμό από μετατοπισμένες σιγμοειδείς συναρτήσεις, δηλαδή $f(y) = \frac{1}{1+e^{-\mu_1(\mu+\nu_1)}}$ και $g(y) = 1 - \frac{1}{1+e^{-\mu_2(\mu+\nu_2)}}$, όπου οι σταθερές μ_i και ν_i επιλέγονται έτσι ώστε να ισχύει η σχέση $f(c/2) = g(c/2) = 0.5$.

Το κριτήριο J_s δίνει ένα μέτρο της ανάγκης για δόσιμο προτεραιότητας στην τρέχουσα διαδρομή. Τα βάρη λ_i^- και λ_i^+ μπορούν όλα να επιλεχθούν ίσα με την μονάδα, επιτρέποντας έτσι σε όλα τα ημιτελή αντικείμενα να ληφθούν υπόψη εξίσου. Μία σειρά από ελαττούμενα βάρη, οδηγεί σε ένα κριτήριο με τοπικό χαρακτήρα, λαμβάνοντας υπόψη τα buffers σε μια μικρή περιοχή γύρω από την υπομηχανή *s*.

Επιπλέον, ένας εναλλακτικός ορισμός του J_s μπορεί να επιτευχθεί θέτοντας $f(y) = \frac{y}{c}$ και $g(y) = 1 - \frac{y}{c}$ λαμβάνοντας $\lambda_N = 0$ και λ_i^-, λ_i^+ ίσα με το ολικό WIP όταν ο αντίστοιχος buffer έχει φθάσει στην χωρητικότητά του. Σε αυτή την περίπτωση, δίνοντας προτεραιότητα στην διαδρομή που έχει την μεγαλύτερη τιμή του J_s προκαλείται μείωση του κόστους WIP της πιο επιβαρημένης διαδρομής.

Αυτοί οι εναλλακτικοί ορισμοί των *f(.), g(.)* παρουσιάζονται στο σχήμα 4.2.



(α) (β)

Σχήμα 4.2. εναλλακτικοί ορισμοί των α) *f(.)* και β) *g(.)*.

Οι κώδικες και για τις δύο μεθόδους ελέγχου υλοποιήθηκαν στο πακέτο MATLAB, έκδοση 6.1 της εταιρίας THE MATHWORKS Inc. και επισυνάπτονται στο παράρτημα Α της παρούσης εργασίας. Στις επόμενες παραγράφους περιγράφεται αναλυτικά η φιλοσοφία υλοποίησης και δόμησης του κάθε κώδικα καθώς και των υπορουτίνων που τον απαρτίζουν.

4.2.1 Γενικές παραδοχές – κοινά χαρακτηριστικά υλοποίησης

Οι buffers εισόδου (1) και εξόδου (2) έχουν άπειρη χωρητικότητα, ενώ οι εσωτερικοί buffers έχουν χωρητικότητα που καθορίζεται από το χρήστη. Η χωρητικότητα αποθηκεύεται στην μεταβλητή *buf_cap(i)*. Κάθε ένας buffer λειτουργεί ως μια ουρά FCFS (First Come First Served) και υλοποιείται ως διάνυσμα με διάσταση ίση με την χωρητικότητά του (μεταβλητή *bufi*). Οι buffers 1 και 2, λόγω της άπειρης χωρητικότητάς τους έχουν δυναμικά καθοριζόμενη διάσταση.

Σε μια χρονική στιγμή ο buffer *i* χαρακτηρίζεται από το πλήθος των κατειλημμένων θέσεων μέσω της μεταβλητής *buf_log(i)* και από την κατάσταση του (διαθέσιμες θέσεις ή γεμάτος) μέσω της μεταβλητής *buf_flag(i)*. Η τιμή της μεταβλητής *buf_flag(i)* είναι 0 για μη γεμάτο buffer, ενώ 1 στην περίπτωση του

γεμάτου buffer. Η είσοδος ενός αντικειμένου στον buffer *i* υλοποιείται με την συνάρτηση **push_buf.m**, ενώ η έξοδος με την συνάρτηση pop_buf.m που περιγράφεται παρακάτω.

Η υπομηχανή *j* υλοποιείται ως μεταβλητή *submj*. Κάθε υπομηχανή έχει την δυνατότητα επεξεργασίας ενός αντικειμένου μια δεδομένη χρονική στιγμή. Μεταξύ των υπομηχανών μιας δεδομένης μηχανής μόνο μια επιτρέπεται να είναι σε κατάσταση επεξεργασίας σε μια χρονική στιγμή. Η κατάσταση της υπομηχανής *j* δίνεται από τη μεταβλητή *subm_flag(j)*, όπου τιμή ίση με 0 δηλώνει την μηχανή σε κατάσταση ηρεμίας (idle) ενώ τιμή ίση με 1 δηλώνει μηχανή σε λειτουργία.

Ο χρόνος επεξεργασίας κάθε υπομηχανής *j* δίνεται από το χρήστη σε sec και η τιμή του αποθηκεύεται στην μεταβλητή *proc_time(j)*. Για την κάθε υπομηχανή *j* ορίζεται η έξοδός της μέσω της μεταβλητής *subm_out(j)*, όπου αποθηκεύεται το αντικείμενο του οποίου μόλις ολοκληρωθεί η επεξεργασία του. Η υπομηχανή παραμένει σε κατάσταση επεξεργασίας μέχρις ότου το αντικείμενο απομακρυνθεί από την έξοδό της, δηλαδή προωθηθεί στον buffer εξόδου της. Ο έλεγχος της επεξεργασίας μιας υπομηχανής υλοποιείται με τη συνάρτηση **subm_proc.m** που περιγράφεται παρακάτω.

Για τις υπομηχανές που μοντελοποιούν τους διαδρόμους προσγείωσης/απογείωσης και τους τροχόδρομους δεν επιτρέπεται καθυστέρηση, δηλαδή με την ολοκλήρωση της επεξεργασίας το περιεχόμενό τους πρέπει να προωθηθεί στον buffer εξόδου τους. Για να διασφαλιστεί η διαθεσιμότητα των buffer εξόδου, οι υπομηχανές αυτές δεν αρχίζουν επεξεργασία αν ο buffer εξόδου τους είναι γεμάτος. Η παραδοχή αυτή αντικατοπτρίζει τις πραγματικές συνθήκες λειτουργίας ενός αεροδρομίου, όπου κυρίως για λόγους ασφαλείας, τα αεροσκάφη δεν επιτρέπεται να παραμένουν στους διαδρόμους προσγείωσης και στους τροχόδρομους μετά την ολοκλήρωση της προσγείωσης και τροχοδρόμησης.

Ωστόσο, για τις υπομηχανές που μοντελοποιούν τους τερματικούς σταθμούς επίγειας εξυπηρέτησης επιτρέπεται καθυστέρηση, είναι δηλαδή δυνατή η παραμονή του περιεχομένου της υπομηχανής σε αυτή ακόμα και μετά το τέλος της

επεξεργασίας, έως ότου γίνει διαθέσιμος ο buffer εξόδου της. Η παραδοχή αυτή πάλι συνάδει με τις πραγματικές συνθήκες λειτουργίας ενός αερολιμένα.

Ο κώδικας στην εκκίνηση της εκτέλεσής του αρχικοποιεί τους buffers και τις υπομηχανές. Έτσι, στην αρχή όλοι οι buffers είναι άδειοι και όλες οι υπομηχανές σε κατάσταση idle (οι μεταβλητές *bufi, buf_log(i)*, *buf_flag(i)* και *submj*, *subm_out(j)*, *subm_flag(j)* είναι ίσες με 0).

Ο χρόνος προσομοίωσης δίνεται από τον χρήστη σε sec ενώ η βάση χρόνου της προσομοίωσης είναι το 1 sec με αρχικό χρόνο t=1.

Οι αφίξεις παράγονται από κατανομή Poisson ανά 60 sec. Η παράμετρος λ της κατανομής καθορίζεται από τον μέσο αριθμό αφίξεων/ώρα, που εισάγεται από τον χρήστη. Επιτρέπεται η άφιξη ενός αεροπλάνου/λεπτό. Η έξοδος της κατανομής περιορίζεται με άνω όριο το 1 και ο αριθμός αφίξεων στις 60 αφίξεις/ώρα.

Κάθε αεροπλάνο χαρακτηρίζεται από την χρονική στιγμή εισόδου του στο σύστημα (π.χ. αυτός που εισάγεται στο σύστημα το 163 λεπτό χαρακτηρίζεται με τον αριθμό 9780). Η στιγμή αυτή είναι διαφορετική για κάθε αεροπλάνο, καθώς επιτρέπεται 1 άφιξη/λεπτό. Η τιμή του χρόνου άφιξης για ένα αεροπλάνο είναι η πληροφορία που διαδίδεται μέσα στο σύστημα μέσω των buffers και των υπομηχανών.

Ο χρόνος εισόδου στο σύστημα είναι η χρονική στιγμή κατά την οποία η νέα άφιξη εισάγεται στον buffer 1 ενώ ο χρόνος εξόδου είναι η χρονική στιγμή κατά την οποία το αεροπλάνο εισάγεται στον buffer 2. Ο χρόνος εξυπηρέτησης ενός αεροσκάφους είναι η διαφορά του χρόνου εξόδου από τον χρόνο εισόδου του στο σύστημα και υπολογίζεται για κάθε αεροπλάνο. Οι χρόνοι εισόδου, εξόδου και εξυπηρέτησης ανά αεροσκάφος αποθηκεύονται σε αρχείο κειμένου (.txt) μετά το τέλος της προσομοίωσης, ενώ σε άλλο αρχείο κειμένου αποθηκεύεται το ιστορικό όλης της προσομοίωσης ανά 15 sec. Συγκεκριμένα αποθηκεύονται:

- Οι χρονικές στιγμές αποθήκευσης.
- Οι εσωτερικοί buffers του συστήματος.

- Οι μεταβλητές *buf_log(i)* για όλους τους buffers (κατειλημμένες θέσεις).
- Οι μεταβλητές *buf_flag(i)* για όλους τους buffers (διαθεσιμότητα θέσεων).
- Τα περιεχόμενα των υπομηχανών.
- Οι μεταβλητές *subm_flag(j)* για όλες τις υπομηχανές (κατάσταση υπομηχανής).
- Ο συνολικός αριθμός αφίξεων ως την δεδομένη χρονική στιγμή.
- Ο συνολικός αριθμός αναχωρήσεων ως την δεδομένη χρονική στιγμή.
- Ο πληθυσμός του συστήματος (ο αριθμός των αεροπλάνων που έχουν εισαχθεί αλλά δεν έχουν αναχωρήσει) ως την δεδομένη χρονική στιγμή αποθήκευσης.

Σε ξεχωριστά αρχεία κειμένου αποθηκεύονται για κάθε υπομηχανή οι χρονικές στιγμές έναρξης και λήξης επεξεργασίας της, για όλη τη διάρκεια της προσομοίωσης.

4.2.2 Προσομοίωση με σειριακό έλεγχο (Κώδικας Α)

Κάθε 1 sec (ένας κύκλος του βρόγχου του χρόνου προσομοίωσης), εξετάζονται διαδοχικά οι υπομηχανές 1, 3, 5, 6, 4 και 2. Αυτό σημαίνει ότι για τις μηχανές Α και Β που απαρτίζονται από τις υπομηχανές 1 και 2 αντίστοιχα, δίνεται προτεραιότητα στην μηχανή Α, σε εκείνες τις περιπτώσεις όπου είναι δυνατή η λειτουργία και των δύο μηχανών. Η φυσική σημασία αυτού του τρόπου ελέγχου είναι ότι σε συνθήκες που είναι δυνατή η εξυπηρέτηση αεροσκάφους και προς τις δύο κατευθύνσεις, δίνεται προτεραιότητα στις αφίξεις έναντι των αναχωρήσεων.

Οι υπομηχανές 5, 6 (μηχανές D και E) χρησιμοποιούνται με σειρά προτεραιότητας από την 5 προς την 6, δηλαδή την χρονική στιγμή *t* ο buffer 4 τροφοδοτεί την πρώτη από τις υπομηχανές 5, 6 που είναι ανενεργή. Αν σε μια χρονική στιγμή έχουν και οι δύο υπομηχανές 5 και 6 τελειώσει την επεξεργασία τους, τότε στον buffer 5 προωθείται το αεροπλάνο με τον παλαιότερο χρόνο εισόδου στο σύστημα (φιλοσοφία FCFS).

4.2.3 Προσομοίωση με έλεγχο βάσει του κριτηρίου κόστους (Κώδικας Β)

Εντός του κύκλου προσομοίωσης με χρόνο *t* εξετάζονται διαδοχικά οι μηχανές Α, Β και C-E. Στις περιπτώσεις όπου είναι δυνατή η λειτουργία και των δύο υπομηχανών της μηχανής C, επιλέγεται η υπομηχανή εκείνη με την μεγαλύτερη τιμή

του κριτηρίου κόστους, όπως αυτό περιγράφηκε προηγουμένως. Αν η τιμή του κριτηρίου είναι ίδια και για τις δύο υπομηχανές τότε τίθεται σε λειτουργία η υπομηχανή 4, που βρίσκεται στην πορεία εξόδου. Αυτό σημαίνει ότι δίνεται προτεραιότητα στα αντικείμενα των οποίων η επεξεργασία από το σύστημα πλησιάζει στην ολοκλήρωση.

Για την λειτουργία των υπομηχανών 5, 6 ισχύουν τα ίδια με την περίπτωση προσομοίωσης με σειριακό έλεγχο.

Ο buffer 3 παρουσιάζει και στις δύο περιπτώσεις μια ιδιαιτερότητα η οποία έγκειται στην κοινή χρήση του τόσο για τα εισερχόμενα όσο και για τα εξερχόμενα αεροσκάφη. Η υλοποίηση του buffer 3 τροποποιείται ως εξής:

- Διατηρούνται οι μεταβλητές για την χωρητικότητα, το πλήθος κατειλημμένων θέσεων και τη διαθεσιμότητα του buffer 3 και ορίζονται δύο νέες μεταβλητές, οι *buf3_iq* και *buf3_oq* που αντιστοιχούν στις ουρές των εισερχομένων και εξερχόμενων αεροσκαφών που εξυπηρετεί ο buffer.
- Για κάθε μια από τις *buf3_iq* και *buf3_oq* ορίζεται η χωρητικότητα, το πλήθος κατειλημμένων θέσεων και η διαθεσιμότητα (*buf3_xq_cap*, *buf3_xq_log*, *buf3_xq_flag*, *x=i,o* αντίστοιχα).
- Σε κάθε κύκλο χρόνου γίνεται δυναμικός καθορισμός της χωρητικότητας μεταξύ των δύο ουρών. Αρχικά, *buf3_iq_cap=buf_cap(3)* και *buf3_oq_cap=0*, δηλαδή όλες οι θέσεις δίνονται στην ουρά εισερχομένων.
- Η υλοποίηση του καθορισμού της χωρητικότητας γίνεται από την συνάρτηση **cap_alloc.m**, που περιγράφεται παρακάτω.

Η ουρά *buf3_iq* χρησιμοποιείται από τις υπομηχανές 1 και 3 ενώ η ουρά *buf3_oq* από τις υπομηχανές 2 και 4. Στην αποθήκευση του ιστορικού προσομοίωσης αποθηκεύονται τα περιεχόμενα των ουρών *buf3_iq* και *buf3_oq* αντί του συνολικού περιεχομένου του buffer 3. Για την αποφυγή πιθανού μπλοκαρίσματος της λειτουργίας του συστήματος σε συνθήκες μεγάλου φόρτου (bottleneck), όπου ο buffer 3 είναι πιθανό να χρησιμοποιηθεί εξ' ολοκλήρου από την εισερχόμενη κυκλοφορία μην επιτρέποντας την έξοδο από το σύστημα στα αντικείμενα που βρίσκονται στον buffer 5, εισάγεται μια επιπλέον συνθήκη ελέγχου για την λειτουργία της μηχανής Α.

Έτσι, δεν επιτρέπεται η λειτουργία της μηχανής αν ο buffer 5 είναι γεμάτος. Σε πραγματικές συνθήκες, αυτό σημαίνει ότι δεν επιτρέπεται προσγείωση όταν η αναμονή για τροχοδρόμηση προς απογείωση είναι πλήρης.

4.2.4 Συνάρτηση **push_buf.m** (*είσοδος αντικειμένου στον buffer*)

Η συνάρτηση δέχεται ως είσοδο το αντικείμενο προς είσοδο (μεταβλητή *input*), το πλήθος κατειλημμένων θέσεων (*buf_log*), τον buffer (*buf*) καθώς και την χωρητικότητα του buffer (*buf_size*). Η έξοδός της είναι το πλήθος κατειλημμένων θέσεων μετά την είσοδο του αντικειμένου (*new_buf_log*), η διαθεσιμότητα του buffer (*new_buf_flag*) και τα περιεχόμενα του buffer μετά την είσοδο (*new_buf*).

Η συνάρτηση ελέγχει αν ο buffer είναι γεμάτος. Στην περίπτωση αυτή παράγει ένα μήνυμα warning και επιστρέφει τον buffer αναλλοίωτο δίνοντας *new_buf_flag=1* (buffer πλήρης). Διαφορετικά, τοποθετεί το νέο αντικείμενο στο τέλος της ουράς (*θέση buf_log+1*), αυξάνει τον αριθμό κατειλημμένων θέσεων κατά 1 και επιστρέφει, έχοντας ελέγξει και την διαθεσιμότητα του buffer μετά την είσοδο του αντικειμένου.

4.2.5 Συνάρτηση **pop_buf.m** (*έξοδος αντικειμένου από τον buffer*)

Η συνάρτηση δέχεται ως είσοδο το πλήθος κατειλημμένων θέσεων (*buf_log*) και τα περιεχόμενα του buffer (*buf*) ενώ έχει ως έξοδο το πλήθος κατειλημμένων θέσεων (*new_buf_log*), τη διαθεσιμότητα του buffer (*new_buf_flag*), τα περιεχόμενα του buffer (*new_buf*) και το εξερχόμενο αντικείμενο (*output*).

Η συνάρτηση ελέγχει αν ο buffer είναι άδειος. Στην περίπτωση αυτή παράγει ένα μήνυμα warning και επιστρέφει *new_buf_log=0*, *new_buf_flag=0*, *output=“ ”* και τον buffer αναλλοίωτο. Διαφορετικά, η έξοδος παίρνει το πρώτο στοιχείο της ουράς (*output=buf(1)*), τα περιεχόμενα του buffer ολισθαίνουν κατά 1 θέση προς τα αριστερά και το *buf_log* μειώνεται κατά 1. Επιπλέον, το *new_buf_flag* γίνεται 0 καθώς είναι λογικό μετά την έξοδο ενός αντικειμένου, ο buffer να μην είναι πλέον πλήρης.

4.2.6 Συνάρτηση **subm_proc.m** (*έλεγχος κατάστασης λειτουργίας υπομηχανής*)

Η συνάρτηση έχει ως είσοδο το περιεχόμενο υπομηχανής (*subm*), την χρονική στιγμή έναρξης επεξεργασίας (*proc_start_time*), τον χρόνο του συστήματος (*t*) καθώς και τον χρόνο επεξεργασίας της υπομηχανής (*subm_proc_time*). Η μοναδική έξοδος της συνάρτησης είναι η έξοδος της υπομηχανής (*subm_out*).

Η συνάρτηση ελέγχει αν το χρονικό διάστημα από την χρονική στιγμή έναρξης επεξεργασίας για την υπομηχανή είναι μικρότερο από το χρόνο επεξεργασίας της. Αν αυτό συμβαίνει, η έξοδος της υπομηχανής επιστρέφεται ως 0 (δεν έχει ολοκληρωθεί η επεξεργασία) ενώ σε διαφορετική περίπτωση (έχει ολοκληρωθεί η επεξεργασία), το περιεχόμενο της υπομηχανής προωθείται στην έξοδό της. Δηλαδή, ελέγχει σε κάθε χρονική στιγμή την κατάσταση της εξόδου της εκάστοτε ενεργής υπομηχανής.

4.2.7 Συνάρτηση **start_proc.m** *(έναρξη επεξεργασίας από υπομηχανή)*

Η συνάρτηση έχει ως είσοδο τον buffer εισόδου της υπομηχανής (*ibuf)*, τον αριθμό κατειλημμένων θέσεων του συγκεκριμένου buffer *(ibuf_log)* και την τιμή του τρέχοντα χρόνου *(t)*. Οι έξοδοί της είναι το περιεχόμενο της υπομηχανής *(subm)*, η κατάσταση της υπομηχανής *(subm_flag)*, την χρονική στιγμή έναρξης της επεξεργασίας *(start_time)*, τον ενημερωμένο buffer εισόδου της υπομηχανής (*ibuf)*, τον ενημερωμένο αριθμό κατειλημμένων θέσεων του συγκεκριμένου buffer *(ibuf_log)* και την ενημερωμένη κατάσταση του buffer *(ibuf_flag)*.

Η συνάρτηση αυτή καλείται εφόσον η υπομηχανή είναι ανενεργή (ο έλεγχος αυτός γίνεται από το κυρίως πρόγραμμα) και υλοποιεί την αλλαγή κατάστασης της υπομηχανής από ανενεργή (idle, *subm_flag=0*) σε ενεργή (working, *subm_flag=1*), δηλαδή γίνεται τροφοδοσία της ανενεργής υπομηχανής από τον buffer εισόδου της, ο οποίος πλέον θα έχει *ibuf_flag=0 (τουλάχιστον μια κενή θέση)*, και αυτή αλλάζει κατάσταση σε ενεργή. Η συνάρτηση καλεί την συνάρτηση **pop_buf**.

4.2.8 Συνάρτηση **end_proc.m** *(τερματισμός επεξεργασίας από υπομηχανή)*

Η συνάρτηση αυτή έχει ως είσοδο την έξοδο της υπομηχανής *(subm_out)*, τον buffer εξόδου *(obuf)*, τον αριθμό των κατειλημμένων θέσεων αυτού *(obuf_log)*, την χωρητικότητά του *(obuf_cap)*, και την τιμή του τρέχοντα χρόνου *(t)*. Οι έξοδοί της

είναι το περιεχόμενο της υπομηχανής *(subm)*, την κατάσταση της υπομηχανής *(subm_flag)*, την ενημερωμένη έξοδο της υπομηχανής *(subm_out)*, την χρονική στιγμή τέλους της επεξεργασίας *(end_time)*, τον ενημερωμένο buffer εξόδου *(obuf)*, τον ενημερωμένο αριθμό των κατειλημμένων θέσεων αυτού *(obuf_log)* και την ενημερωμένη κατάστασή του *(obuf_flag)*.

Η συνάρτηση αυτή καλείται εφόσον η υπομηχανή είναι ενεργή και έχει τελειώσει την επεξεργασία της (ο έλεγχος αυτός γίνεται από το κυρίως πρόγραμμα) και υλοποιεί την αλλαγή κατάστασης της υπομηχανής από ενεργή (working, *subm_flag=1*) σε ανενεργή (idle, *subm_flag=0*), δηλαδή η έξοδος της ενεργής υπομηχανής προωθείται στον buffer εξόδου της και αυτή αλλάζει κατάσταση σε ανενεργή ενώ το περιεχόμενο και η έξοδός της μηδενίζονται. Η συνάρτηση καλεί την συνάρτηση **push_buf**.

4.2.9 Συνάρτηση **find_hub2.m** *(επιλογή τερματικού σταθμού προς εξυπηρέτηση)*

Η συνάρτηση αυτή έχει ως είσοδο τις εξόδους των τερματικών σταθμών *(out5, out6)* και ως έξοδο τον αριθμό του επιλεχθέντος τερματικού σταθμού προς εξυπηρέτηση.

Η συγκεκριμένη συνάρτηση υλοποιεί την εύρεση του αεροσκάφους με τον μεγαλύτερο χρόνο παραμονής στο σύστημα μεταξύ αυτών που έχουν ολοκληρώσει την εξυπηρέτησή τους στους τερματικούς σταθμούς. Η συνάρτηση επιστρέφει τον αριθμό της υπομηχανής που θα τροφοδοτήσει τον buffer 5. Στην περίπτωση που καμία υπομηχανή εξυπηρέτησης (τερματικός σταθμός) δεν είναι έτοιμη, η συνάρτηση επιστρέφει μηδενική τιμή.

4.2.10 Συνάρτηση **cap_alloc.m** (*χωρητικότητες ουρών εισόδου-εξόδου του buffer 3*)

Οι είσοδοι της συνάρτησης αυτής είναι η χωρητικότητα του buffer (*buf_cap*), οι κατειλημμένες θέσεις της ουράς 1 (*b1_log*), οι κατειλημμένες θέσεις της ουράς 2 (*b2_log*) και η κατάσταση της υπομηχανής που τροφοδοτεί την ουρά 1. Οι έξοδοι είναι η νέα χωρητικότητα της ουράς 2 και η διαθεσιμότητα της ουράς 2.

Ως ουρά 1 νοείται η ουρά εισερχομένων ενώ ως ουρά 2 η ουρά εξερχόμενων ή και αντίστροφα. Η χωρητικότητα της ουράς 2 καθορίζεται μέσω της μεταβλητής

buf_cap-b1_log, δηλαδή της αποδίδονται οι θέσεις που δεν είναι κατειλημμένες από την ουρά 1. Αν η υπομηχανή *subm* είναι σε κατάσταση λειτουργίας, η ουρά 1 δεσμεύει 1 επιπλέον θέση. Στη συνέχεια ελέγχεται αν η ουρά 2 είναι πλήρης μετά τον καθορισμό της νέας χωρητικότητας λαμβάνοντας υπόψη τον τρέχοντα αριθμό κατειλημμένων θέσεων (*b2_log*).

Ο παραπάνω αλγόριθμος έχει ως αποτέλεσμα οι κενές θέσεις του buffer 3 να είναι σε κάθε στιγμή διαθέσιμες και στις δύο ουρές. Η ουρά που πρώτη χρειάζεται θέση (η υπομηχανή που την τροφοδοτεί ξεκινά επεξεργασία) την δεσμεύει από τις κενές θέσεις. Η δέσμευση της θέσης διατηρείται μέχρις ότου το αντικείμενο εξέλθει της ουράς.

4.3 Αρχικές τιμές & δεδομένα

Οι τιμές που τέθηκαν τόσο για τους χρόνους επεξεργασίας των μηχανών (και υπομηχανών) όσο για την χωρητικότητα των buffers είναι αντιπροσωπευτικές ενός αερολιμένα μικρής χωρητικότητας, όπως περιγράφηκε στα προηγούμενα κεφάλαια. Στους πίνακες 4.3 και 4.4 δίνονται οι σταθερές που χρησιμοποιήθηκαν στο μοντέλο μας για την εν λόγω προσομοίωση λειτουργίας του αερολιμένα.

Πίνακας 4.3. Χρόνος επεξεργασίας για κάθε machine και submachine.

Machine	Submachine	Περιγραφή	Χρόνος (s)
A	1	Προσγείωση	120
B	2	Απογείωση	75
C	3	Τροχοδρόμηση (μετά την προσγείωση)	180
	4	Τροχοδρόμηση (πριν την απογείωση)	180
D	5	Επίγεια εξυπηρέτηση	1500
E	6	Επίγεια εξυπηρέτηση	1500
Ελάχιστος χρόνος παραμονής αεροσκάφους στον αερολιμένα			*2055*

Πίνακας 4.4. Χωρητικότητα του κάθε buffer.

Buffer	Περιγραφή	Χωρητικότητα (τεμ.)
1	Εναέριος χώρος αφίξεων	∞
2	Εναέριος χώρος αναχωρήσεων	∞
3	Αναμονή μετά την προσγείωση ή πριν την απογείωση	4
4	Αναμονή για θέση εξυπηρέτησης	2
5	Αναμονή για τροχοδρόμηση προς απογείωση	2

Η λειτουργία του αερολιμένα εξετάσθηκε για διάφορους ωριαίους ρυθμούς αφίξεων (2.5 ~ 20 αφίξεις/h, με βήμα 2.5) για να διαπιστωθεί ποιες από τις δύο προαναφερθέντες μεθόδους ελέγχου συντελούσαν στην μεγαλύτερη απόδοση του αερολιμένα δηλαδή μικρότερος μέσος χρόνος εξυπηρέτησης ανά αεροσκάφος και περισσότερα εξυπηρετηθέντα αεροσκάφη στη διάρκεια ενός 24-ώρου (86400 s). Εδώ, θα πρέπει να σημειώσουμε ότι στην πράξη συνήθως ένας αερολιμένας δεν λειτουργεί όλο το 24ωρο με τον ίδιο ωριαίο αριθμός αφικνούμενων αεροσκαφών, καθώς κατά την διάρκεια της νύχτας ο ρυθμός αυτός μειώνεται και σε κάποιες περιπτώσεις μηδενίζεται (λόγω επιπτώσεων θορύβου). Ο τακτικός χρόνος αποθήκευσης της κατάστασης του συστήματος ορίσθηκε στα 15 s.

4.4 Αποτελέσματα

Τα αποτελέσματα μετά την εκτέλεση των κωδίκων για κάθε τρόπο ελέγχου και για κάθε ωριαίο ρυθμό αφίξεων παρατίθενται παρακάτω σε διαγραμματική μορφή, (σχ.4.3 – σχ.4.16).

Σχολιάζοντας τα διαγράμματα αυτά μπορούμε να υποστηρίξουμε ότι και οι δύο τρόποι ελέγχου της λειτουργίας του αερολιμένα λειτούργησαν σε ικανοποιητικό βαθμό.

Αρχικά, στα διαγράμματα του σχήματος 4.3, παρατηρούμε ότι σε γενικές γραμμές οι κώδικες δίνουν αποδεκτό μέσο ωριαίο αριθμό αφίξεων. Οι όποιες διαφορές δικαιολογούνται από την στοχαστική φύση της κατανομής Poisson. Αυτό σημαίνει ότι όσο μεγαλύτερος είναι ο χρόνος προσομοίωσης τόσο θα πλησιάζει ο τελικός μέσος ωριαίος αριθμός αφίξεων προς την τιμή που θέσαμε σε κάθε εκτέλεση των κωδίκων.

Στο σχήμα 4.4, παρουσιάζονται πολύ χρήσιμες και κρίσιμες πληροφορίες για την δυναμικότητα του αερολιμένα. Συγκεκριμένα, και οι δύο τρόποι ελέγχου κατέληξαν σε ένα όριο από 111 έως 112 αναχωρούντων αεροσκαφών σε ένα 24ωρο, διαφορά που δεν μπορεί να θεωρηθεί σημαντική και οφείλεται στις τυχαίες χρονικές στιγμές αφίξεων των αεροσκαφών από την κατανομή Poisson.

Ένα άλλο στοιχείο που προκύπτει από την ανάλυση αυτών των διαγραμμάτων είναι ότι ο κρίσιμος ωριαίος αριθμός αφίξεων του αερολιμένα είναι περίπου στα 5 αεροσκάφη ανά ώρα. Αυτό, συνάδει με τον θεωρητικό υπολογισμό της «παραγωγικότητας» του αερολιμένα, θεωρώντας ότι οι μηχανές D, E δουλεύουν συνεχώς και τα αεροσκάφη δεν καθυστερούν σε καμία από τις υπόλοιπες μηχανές και buffers. Έτσι, θεωρητικά ο αερολιμένας μπορεί να εξυπηρετεί, στην καλύτερη περίπτωση, 2 αεροσκάφη σε χρόνο 1500 sec, δηλαδή 4.8 αεροσκάφη ανά ώρα.

Για ρυθμούς αφίξεων κάτω από 5 αεροσκάφη ανά ώρα, και οι δύο τρόποι ελέγχου επέδειξαν ικανότητα επαρκούς ελέγχου της λειτουργίας του αερολιμένα. Εδώ, θα πρέπει να υπενθυμίσουμε, ότι στην πραγματικότητα ένας αερολιμένας δεν λειτουργεί όλο το 24ωρο με τον ίδιο ωριαίο αριθμό αφίξεων. Έτσι, υπάρχουν χρονικά διαστήματα που ο αερολιμένας έχει τον χρόνο να εξυπηρετήσει και να αποδεσμεύσει τα αεροσκάφη χωρίς να ανεβαίνει ο πληθυσμός των εισερχομένων αεροσκαφών προς προσγείωση και εξυπηρέτηση.

Παρατηρώντας το σχήμα 4.5 που αφορά στον μέσο χρόνο εξυπηρέτησης ανά αεροσκάφος, φαίνεται ότι για κάτω από 5 αφίξ/ώρα, οι δύο τρόποι ελέγχου οδηγούν στον ίδιο μέσο χρόνο. Αξιοσημείωτη είναι η συμπεριφορά των ελέγχων για 7.5 αφίξ/ώρα, όπου είναι φανερό ότι ο σειριακός έλεγχος υπερτερεί του ελέγχου με βάση το κριτήριο κόστους κατά 2 ώρες ταχύτερης εξυπηρέτησης. Ακόμα και για μεγαλύτερους ρυθμούς αφίξεων, ο έλεγχος βάσει του κριτηρίου κόστους διαφαίνεται να είναι λιγότερο αποδοτικότερος έναντι του σειριακού, αν και πλέον ο χρόνος εξυπηρέτησης σε απόλυτες τιμές (16 ώρες ανά αεροσκάφος) είναι απαγορευτικός για επίγεια εξυπηρέτηση αεροσκάφους σε αερολιμένα.

Όσον αφορά τώρα στα διαγράμματα συχνότητας λειτουργίας των υπομηχανών και συγκεκριμένα της μηχανής Α, διαπιστώνουμε ότι ο σειριακός έλεγχος την ελέγχει αποτελεσματικότερα. Στο διάγραμμα 4.6α, φαίνεται ότι η υπομηχανή 1 (προσγείωση) παρουσιάζει μικρότερα διαστήματα σε κατάσταση αναμονής, μεταξύ δύο διαδοχικών εκκινήσεων, με σειριακό έλεγχο. Επίσης, παρατηρείται η ομαλοποίηση της συχνότητας λειτουργίας μετά από κάποιο χρόνο προσομοίωσης ($t > 7000$ sec). Αυτό εξηγείται από το γεγονός ότι έχοντας πλέον συγκεντρωθεί πολλά αεροσκάφη στον

buffer 1 και με δεδομένη της προτεραιότητα λειτουργίας της υπομηχανής 1 έναντι της υπομηχανής 2, η πρώτη αναμένει να αδειάσει ο buffer 3 για να αφήσει άλλο αεροσκάφος να προσγειωθεί.

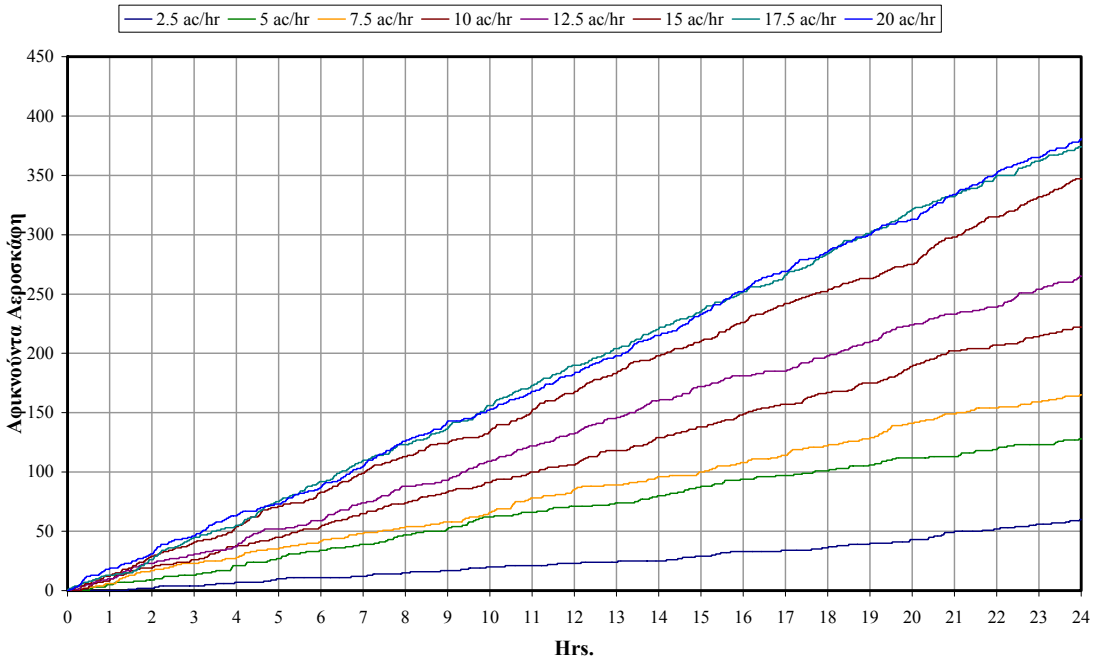
Η λειτουργία των υπομηχανών 2 και 4 φαίνεται να είναι πιο αποδοτική με βάση το κριτήριο κόστους, καθώς παρατηρείται να έχουν διπλάσια συχνότητα λειτουργίας από ότι με σειριακό έλεγχο. Το κριτήριο ελέγχου με βάση το κόστος δεν δίνει πάντα προτεραιότητα στα εισερχόμενα αεροσκάφη και έτσι φαίνεται ότι το κριτήριο συνήθως δίνει μεγαλύτερη τιμή J_s για τα εξερχόμενα αεροσκάφη με αποτέλεσμα να λειτουργούν συχνότερα οι υπομηχανές 2 και 4. Φαίνεται επίσης ότι μετά από κάποια χρονική στιγμή παρατηρείται η συνεχής λειτουργία των υπομηχανών 5 και 6, και με τους δύο τρόπους ελέγχου.

Εξετάζοντας την συμπεριφορά των buffers, ενδιαφέρον παρουσιάζει η χρονική μεταβολή του buffer 1. Είναι προφανής εδώ η επίδραση των διαφορετικών ελέγχων του συστήματος. Ο buffer 1 στον σειριακό έλεγχο (σχ. 4.12α) για κάθε χρονική στιγμή προσομοίωσης συγκεντρώνει περισσότερα αναμένοντα αεροσκάφη στον εναέριο χώρο, ενώ στον έλεγχο με βάση το κριτήριο κόστους ο αριθμός έχει ελαττωθεί.

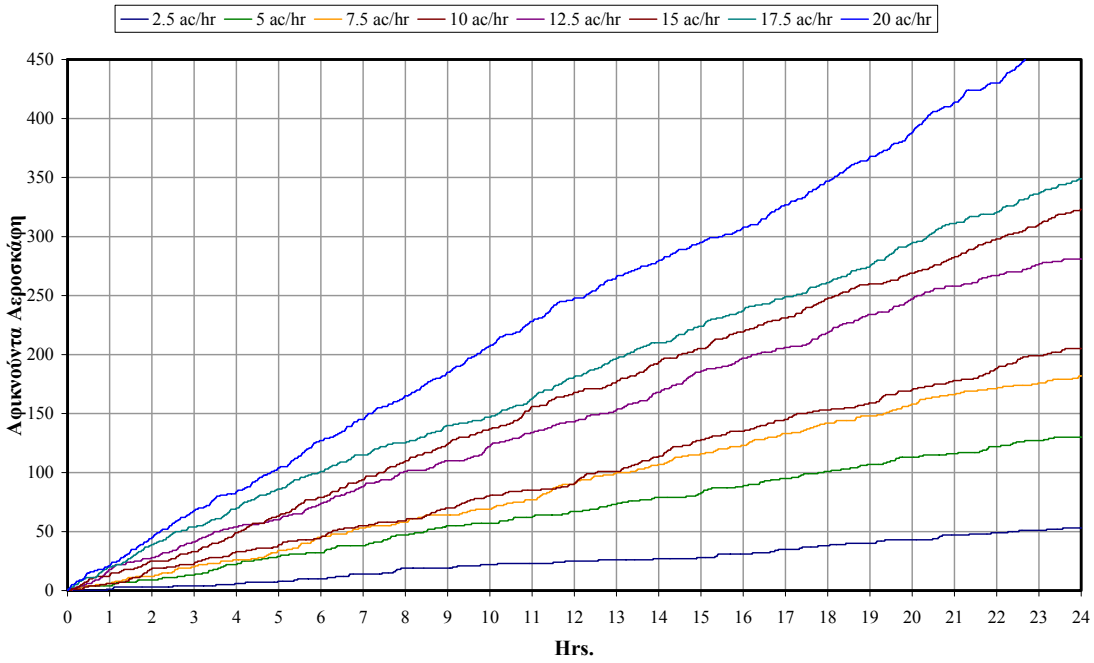
Η συμπεριφορά αυτή δικαιολογείται από τον τρόπο επιλογής της προς λειτουργίας μηχανής Α ή μηχανής Β, ο οποίος δεν δίνει πάντα προτεραιότητα στην μηχανή Α και έτσι το σύστημα αποσυμφορίζεται με ομαλότερο τρόπο. Ο έλεγχος με βάση το κριτήριο κόστους διαφαίνεται να κάνει καλύτερη διαχείριση της λειτουργίας της μηχανής C και του buffer 3 και κατ΄επέκταση και του buffer 1, παρατήρηση που ενισχύεται από τα διαγράμματα του buffer 2 (σχ. 4.13) όπου βλέπουμε ότι και οι αναχωρήσεις ακολουθούν περίπου το ίδιο μοτίβο.

Παρατηρώντας τα διαγράμματα των υπολοίπων buffers διαπιστώνουμε μια εν γένει καλύτερη διαχείρισή τους από τον έλεγχο βάσει του κριτηρίου κόστους. Διαπιστώνεται ότι όλοι σχεδόν οι buffers είναι βεβαρημένοι καθώς το πλήθος των αεροσκαφών σε αυτούς είναι συνεχώς κοντά στην χωρητικότητά τους. Η κύρια αιτία που το προκαλεί αυτό είναι ότι ο ωριαίος ρυθμός αφίξεων (10 αφη/hr) έχει υπερβεί την δυναμικότητα του συστήματος.

Ενδιαφέρον παρουσιάζει η συμπεριφορά του buffer 5, στο σχήμα 4.16. Ο έλεγχος βάσει του κριτηρίου κόστους κάνει σαφώς καλύτερη διαχείριση του καθώς ελάχιστες φορές αγγίζει το μέγιστο της χωρητικότητάς του, ενώ σχεδόν τον μισό χρόνο λειτουργίας του είναι άδειος. Αυτό βέβαια έχει να κάνει και με την διαχείριση των υπομηχανών 3 και 4 που δεν δίνεται πάντα προτεραιότητα στην 3, όπως συμβαίνει στον σειριακό έλεγχο.

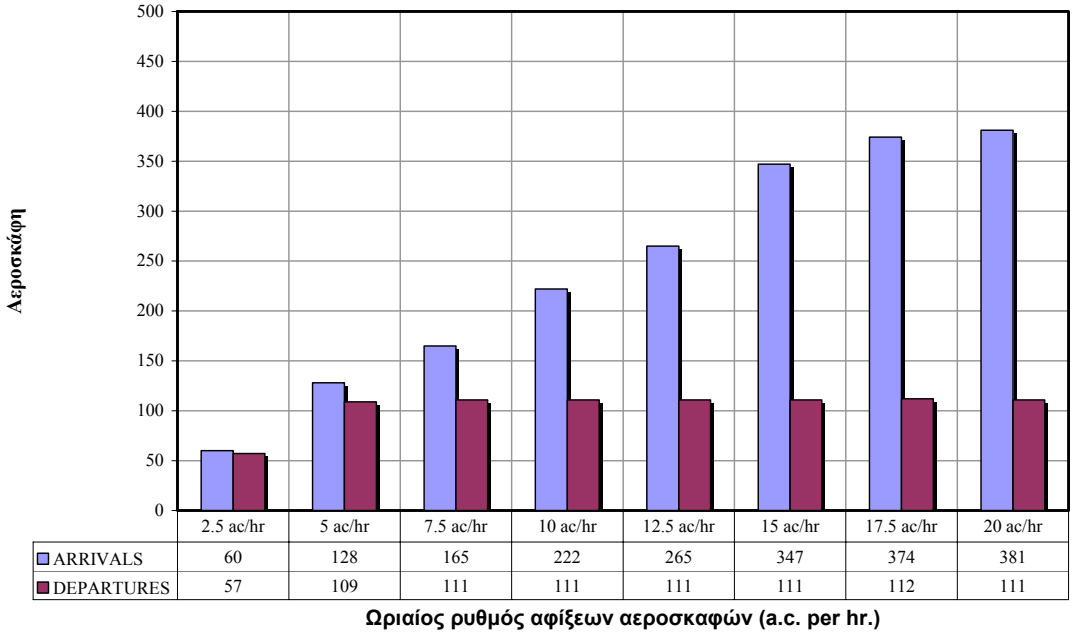


(α)

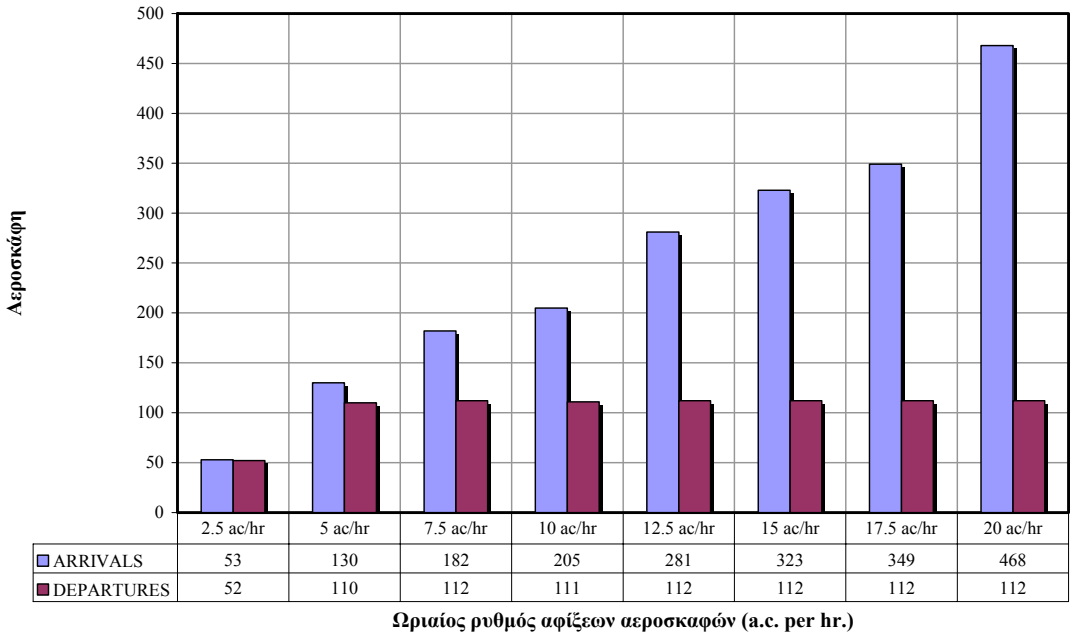


(β)

Σχήμα 4.3. Αθροιστικός αριθμός αφίξεων σε ένα 24-ωρο για διάφορους ωριαίους ρυθμούς αφίξεων: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

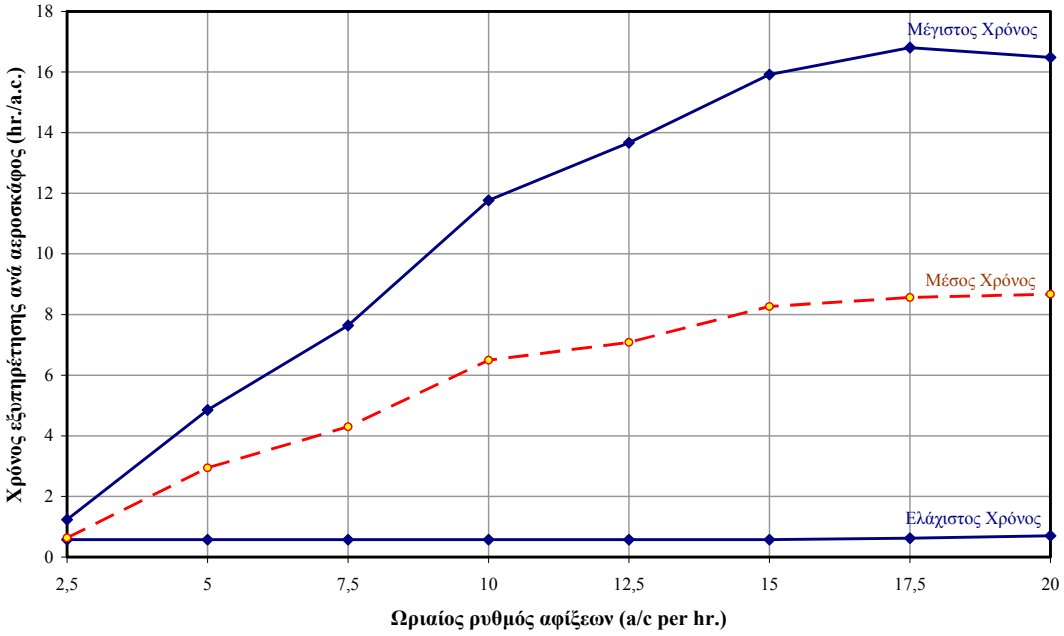


(α)

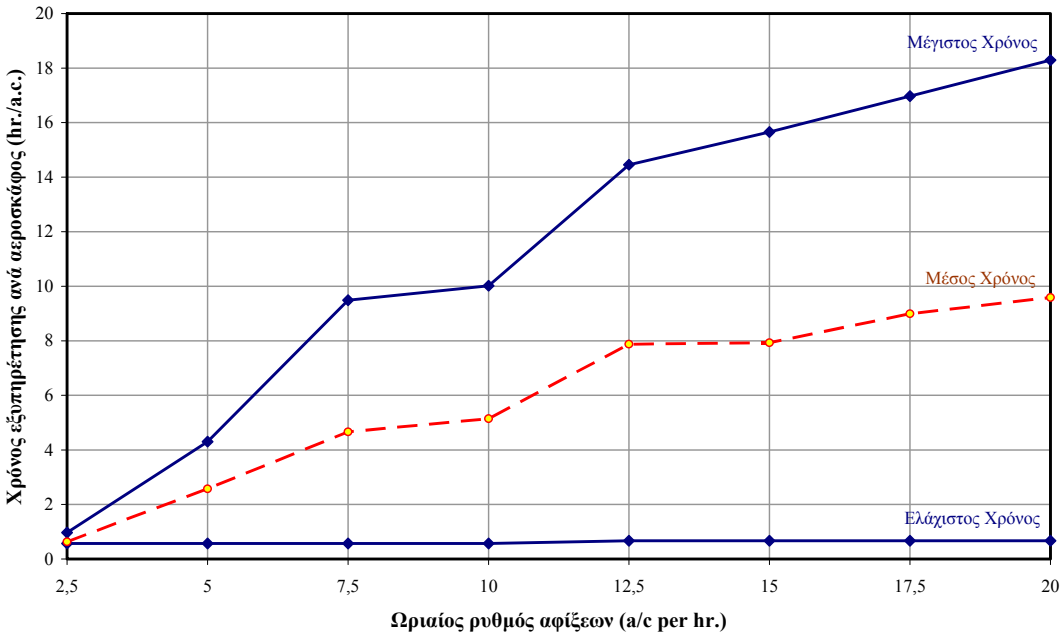


(β)

Σχήμα 4.4. Σύγκριση αριθμού αφίξεων & αναχωρήσεων σε ένα 24-ωρο για διάφορους ωριαίους ρυθμούς αφίξεων: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

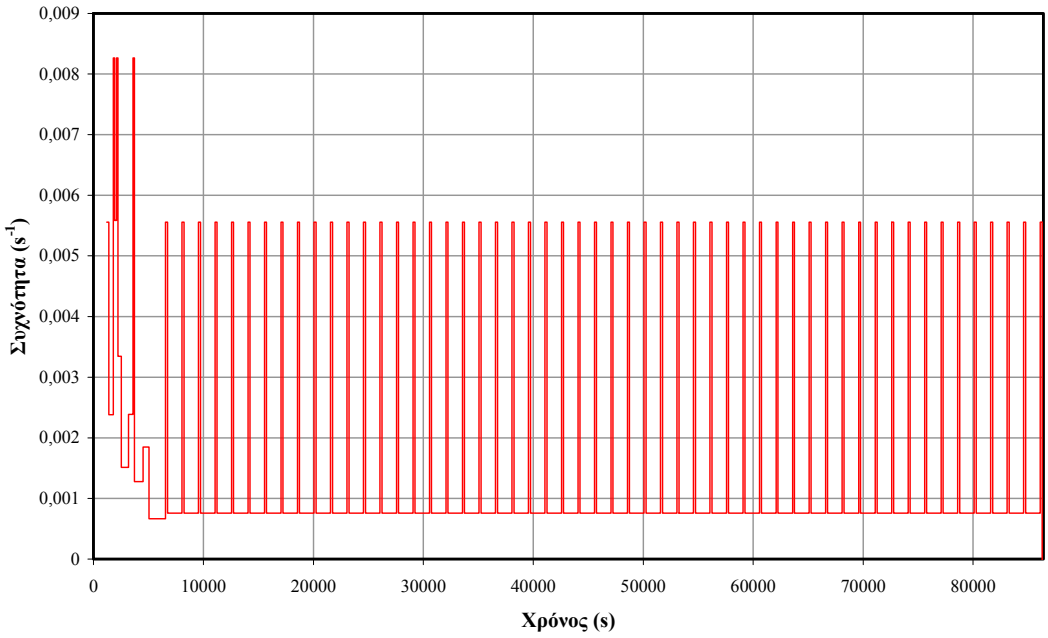


(α)

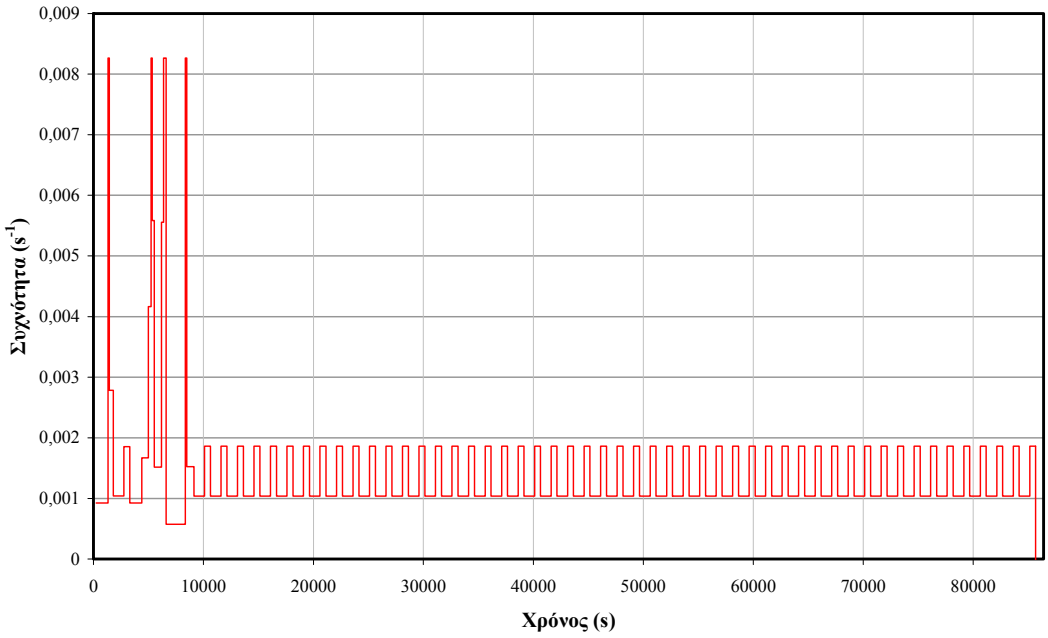


(β)

Σχήμα 4.5. Σύγκριση ελάχιστου, μέγιστου και μέσου χρόνου εξυπηρέτησης ανά αεροσκάφος, σε ένα 24-ωρο για διάφορους ωριαίους ρυθμούς αφίξεων: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

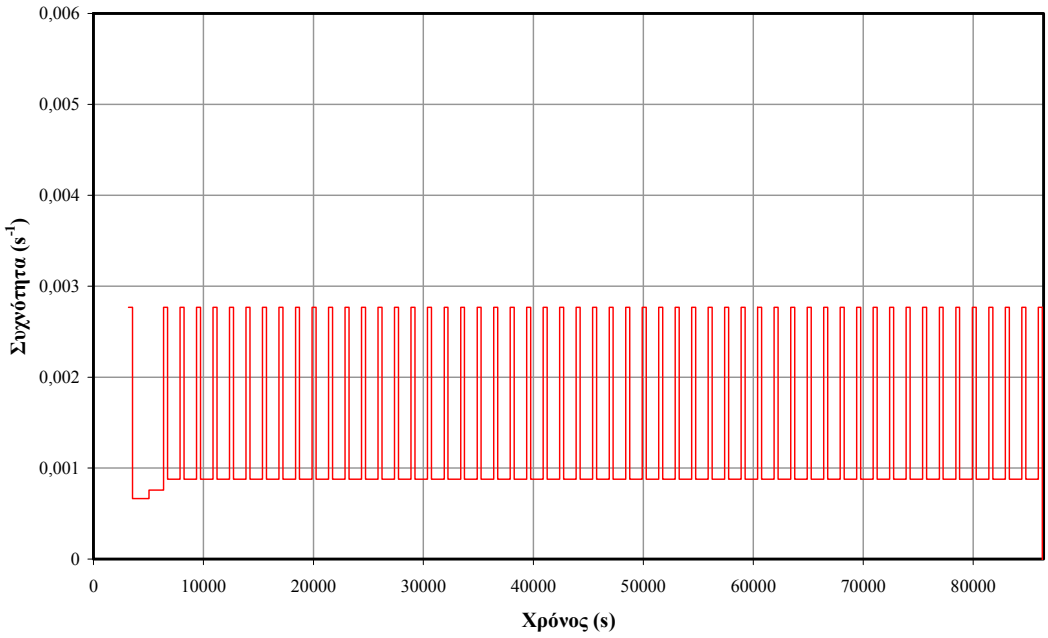


(α)

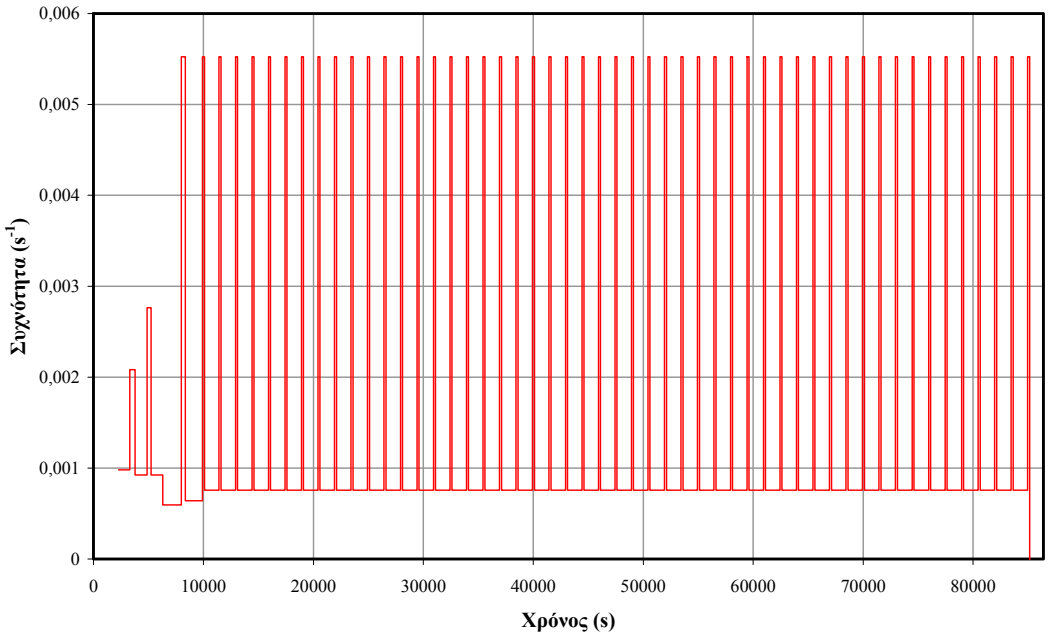


(β)

Σχήμα 4.6. Σύγκριση συχνότητας λειτουργίας υπομηχανής 1, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

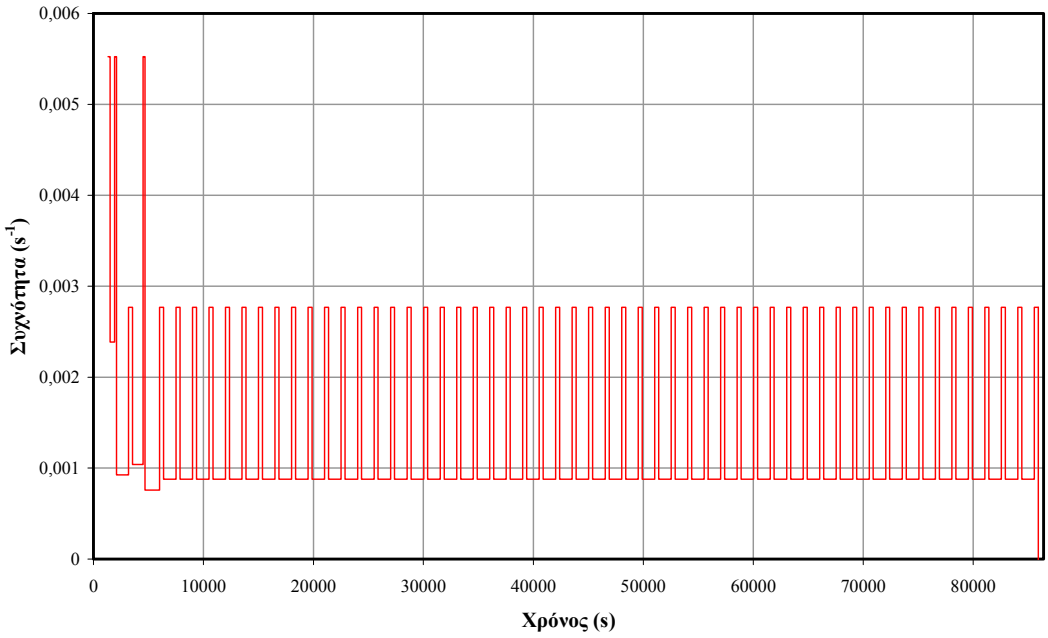


(α)

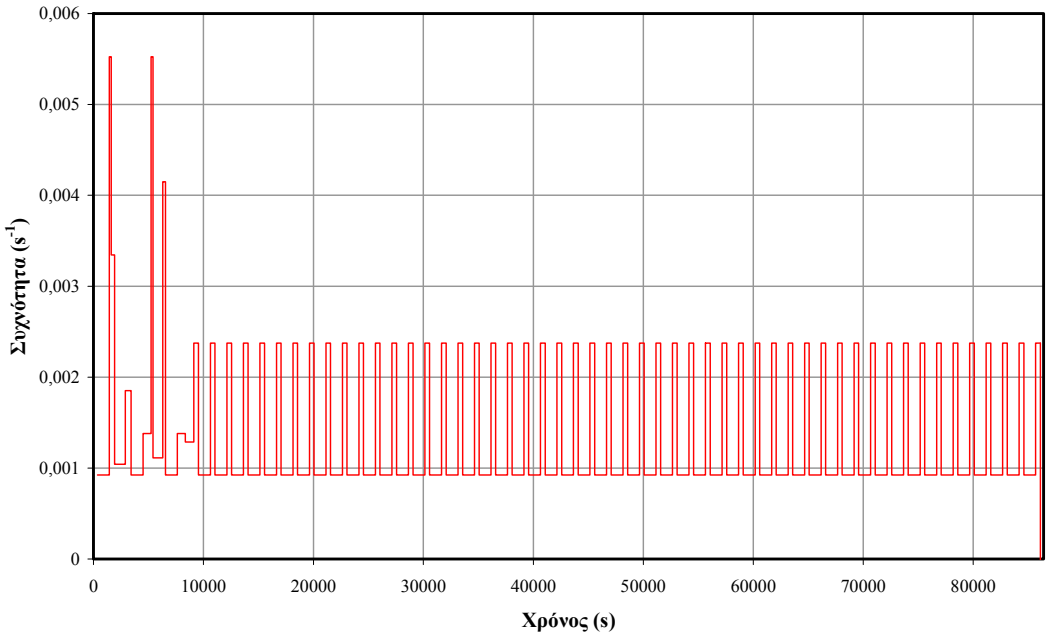


(β)

Σχήμα 4.7. Σύγκριση συχνότητας λειτουργίας υπομηχανής 2, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

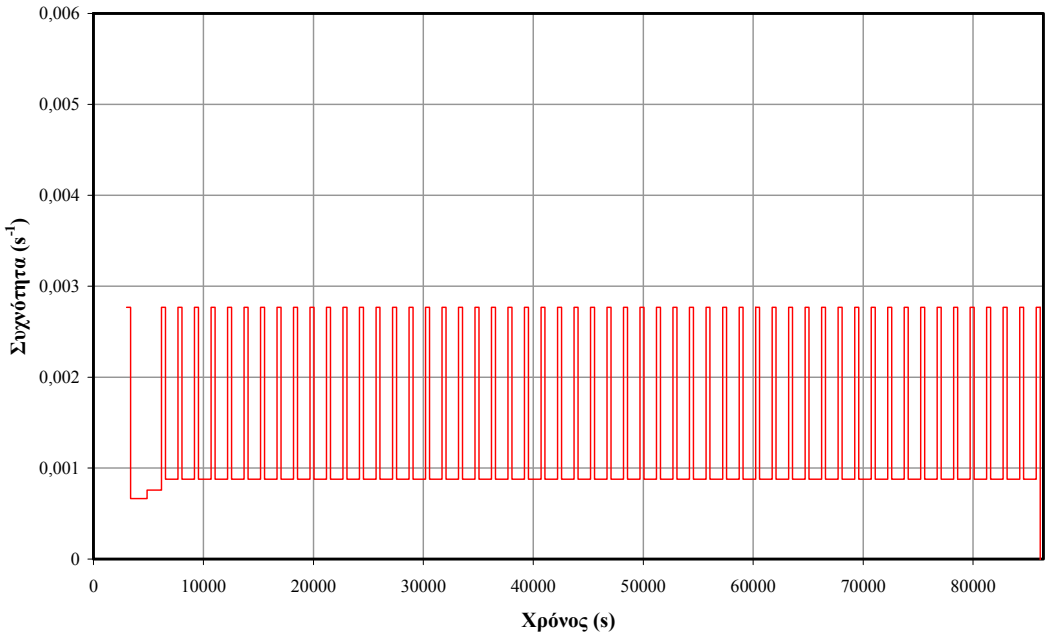


(α)

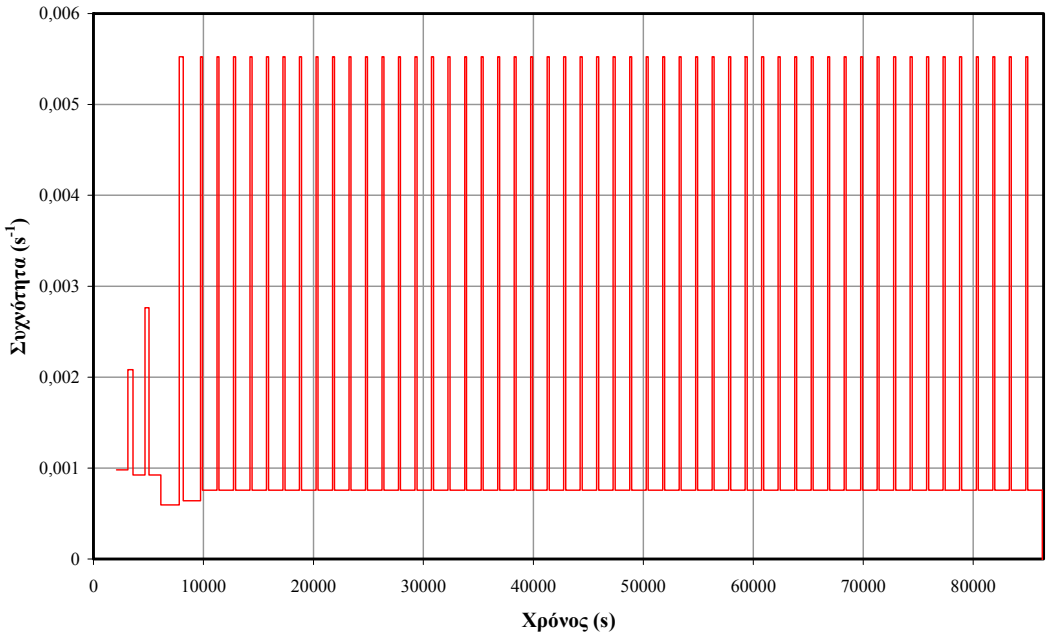


(β)

Σχήμα 4.8. Σύγκριση συχνότητας λειτουργίας υπομηχανής 3, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

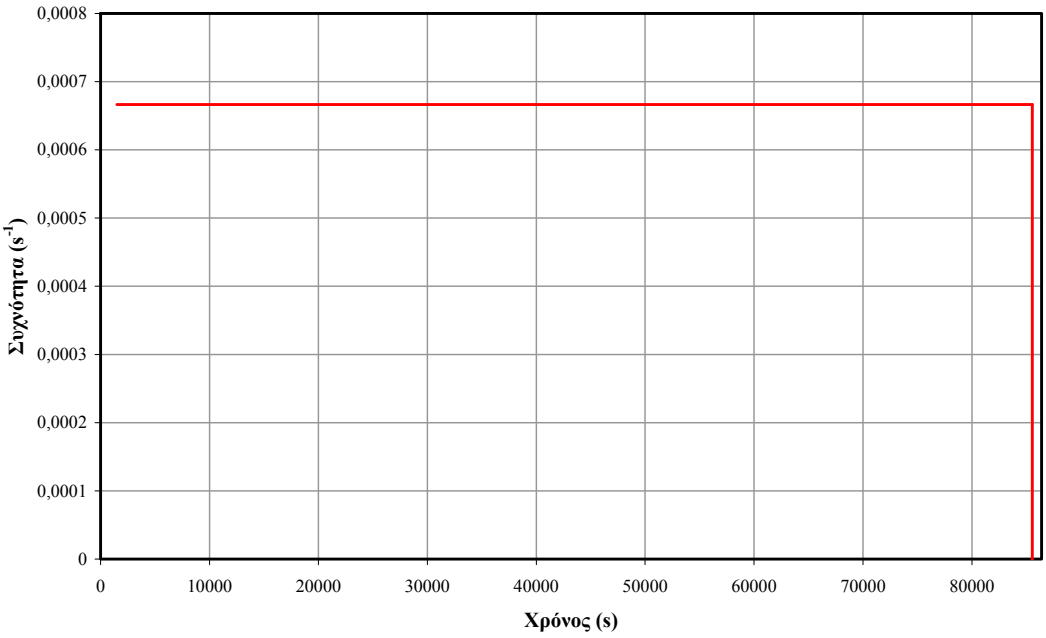


(α)

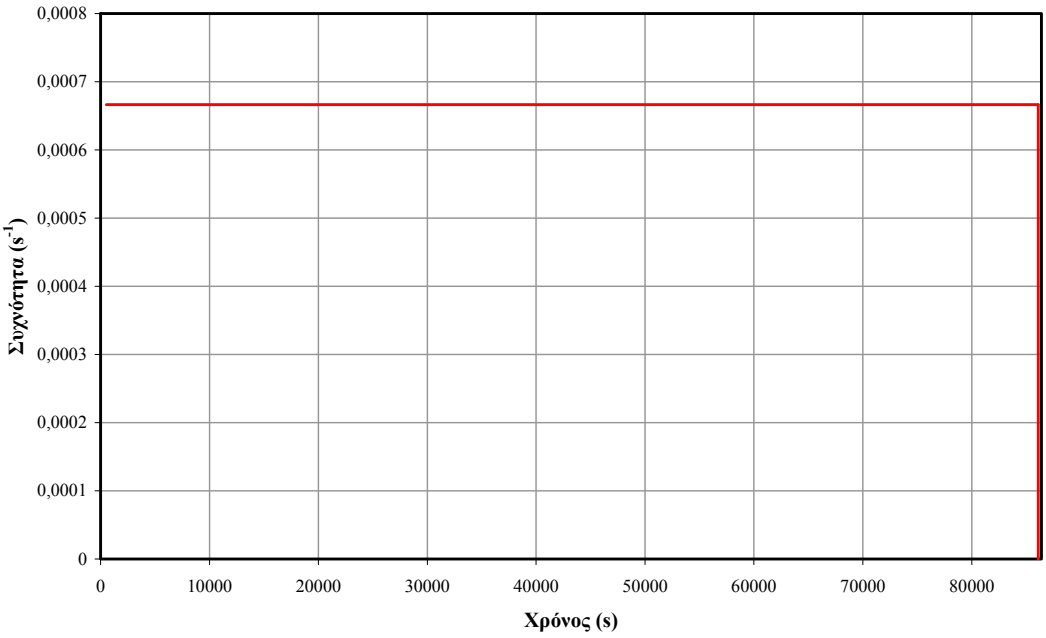


(β)

Σχήμα 4.9. Σύγκριση συχνότητας λειτουργίας υπομηχανής 4, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

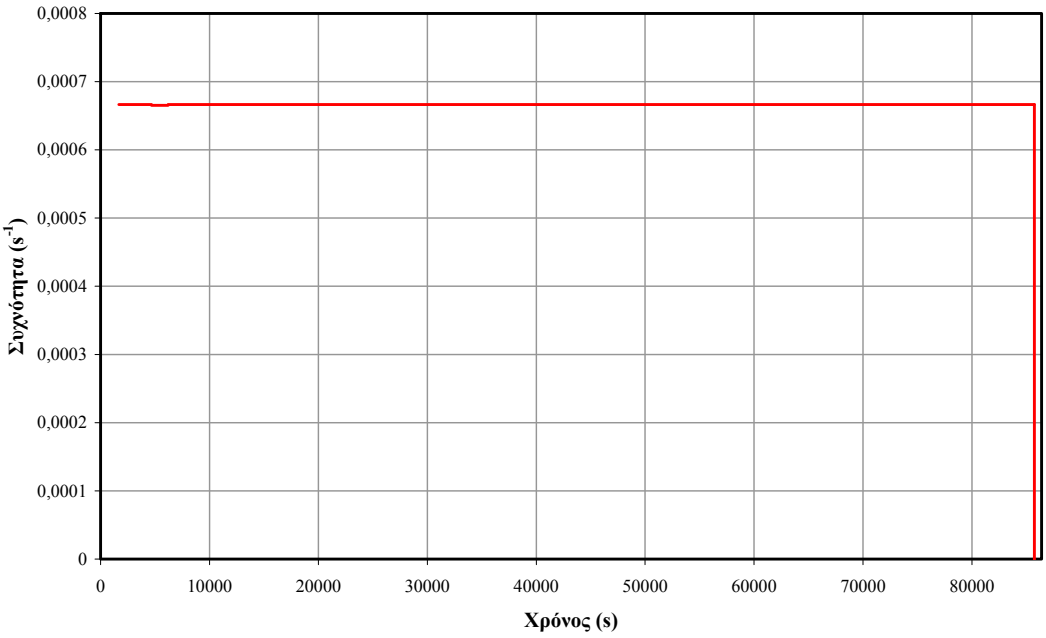


(α)

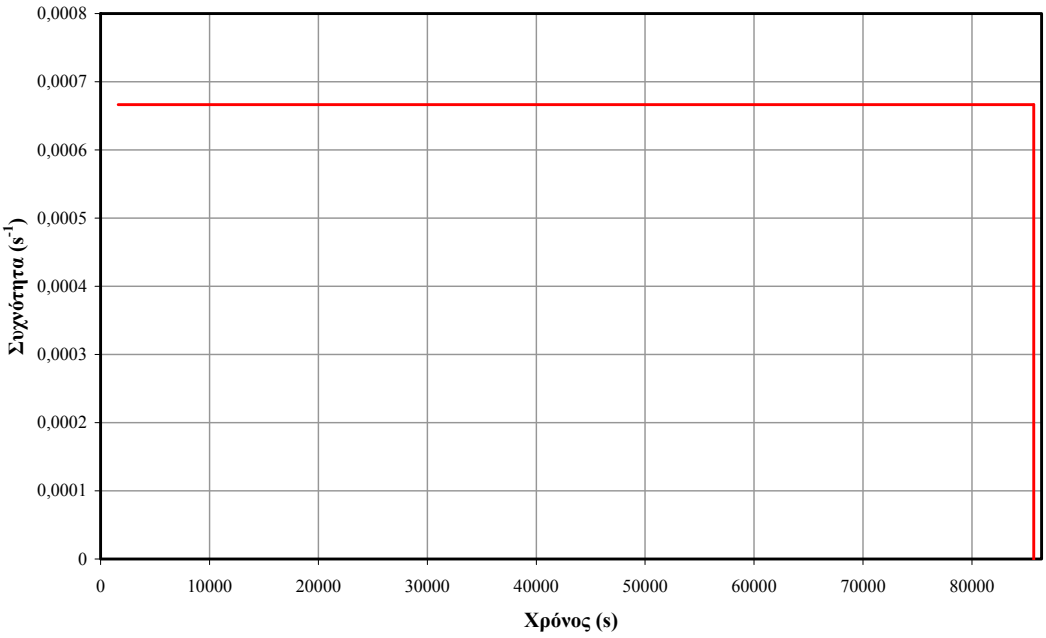


(β)

Σχήμα 4.10. Σύγκριση συχνότητας λειτουργίας υπομηχανής 5, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

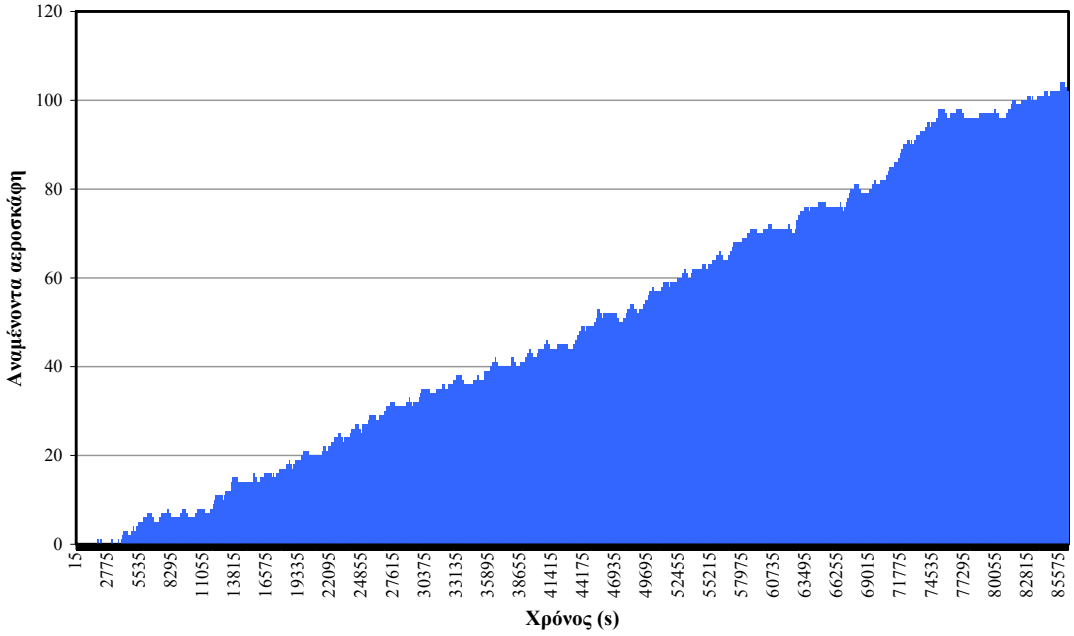


(α)

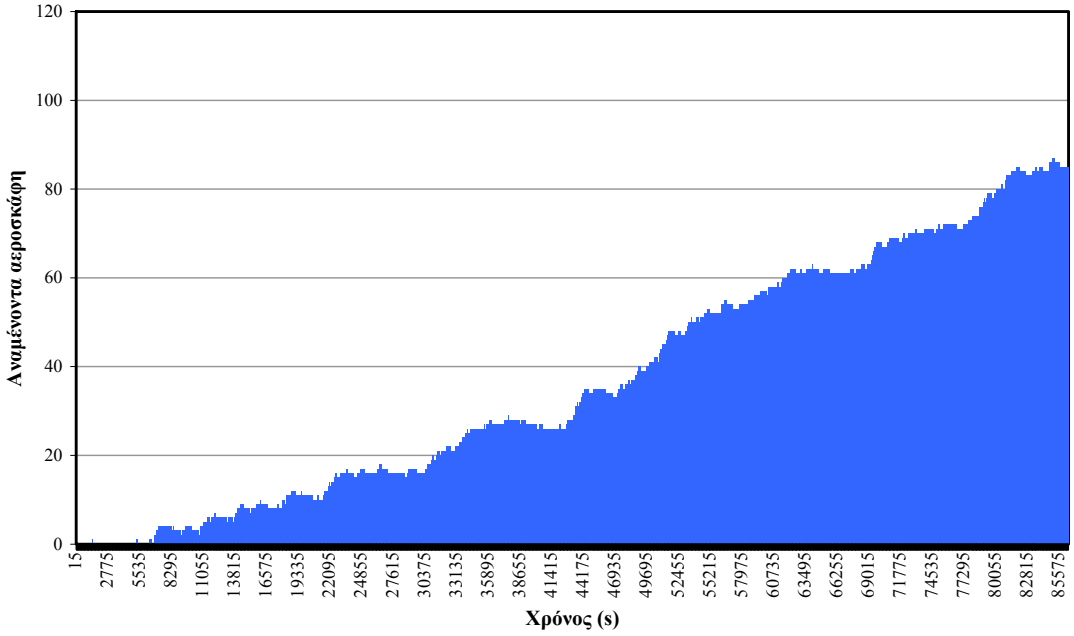


(β)

Σχήμα 4.11. Σύγκριση συχνότητας λειτουργίας υπομηχανής 6, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

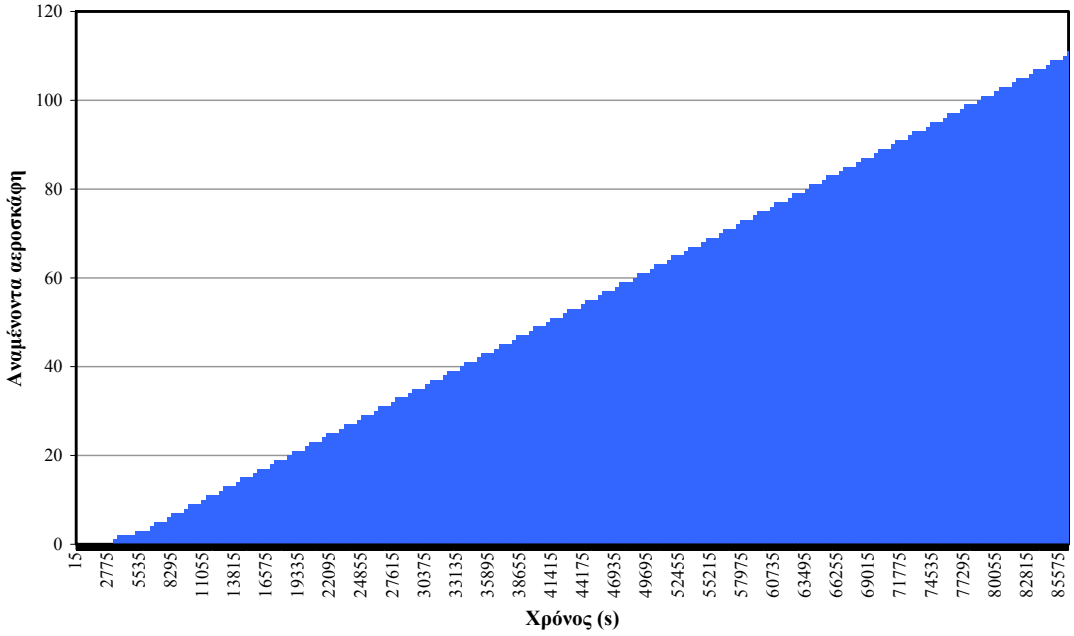


(α)

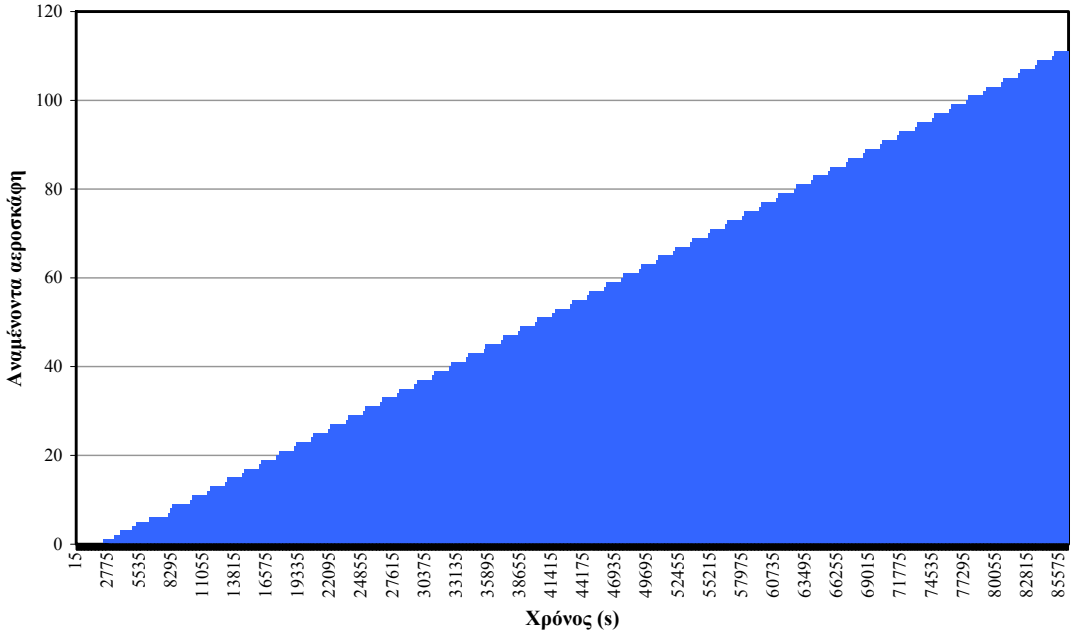


(β)

Σχήμα 4.12. Σύγκριση πλήθους αεροσκαφών στον buffer 1, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

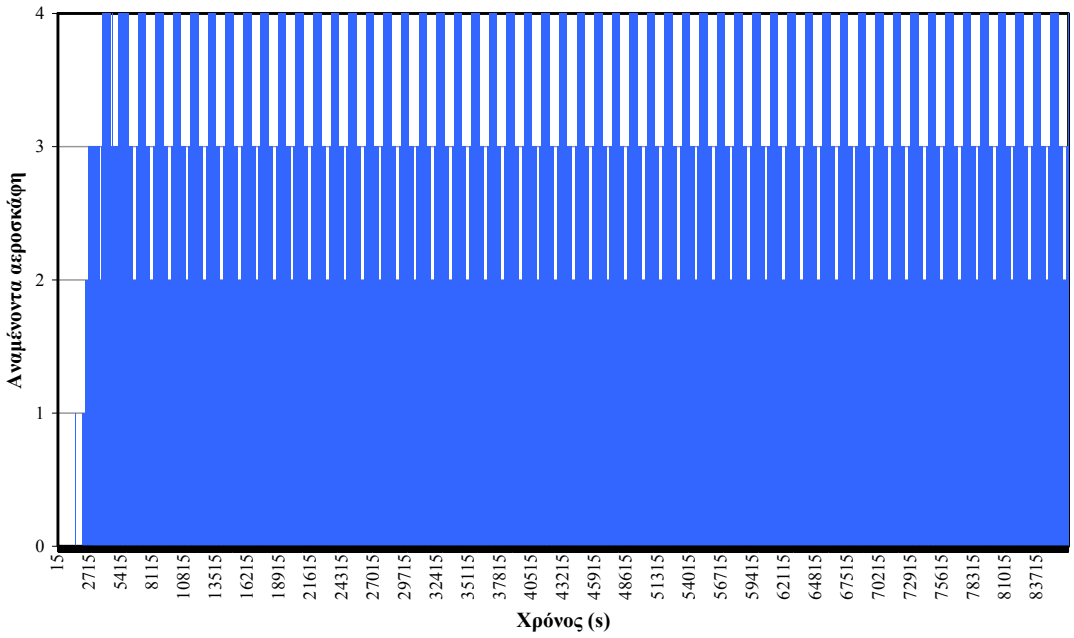


(α)

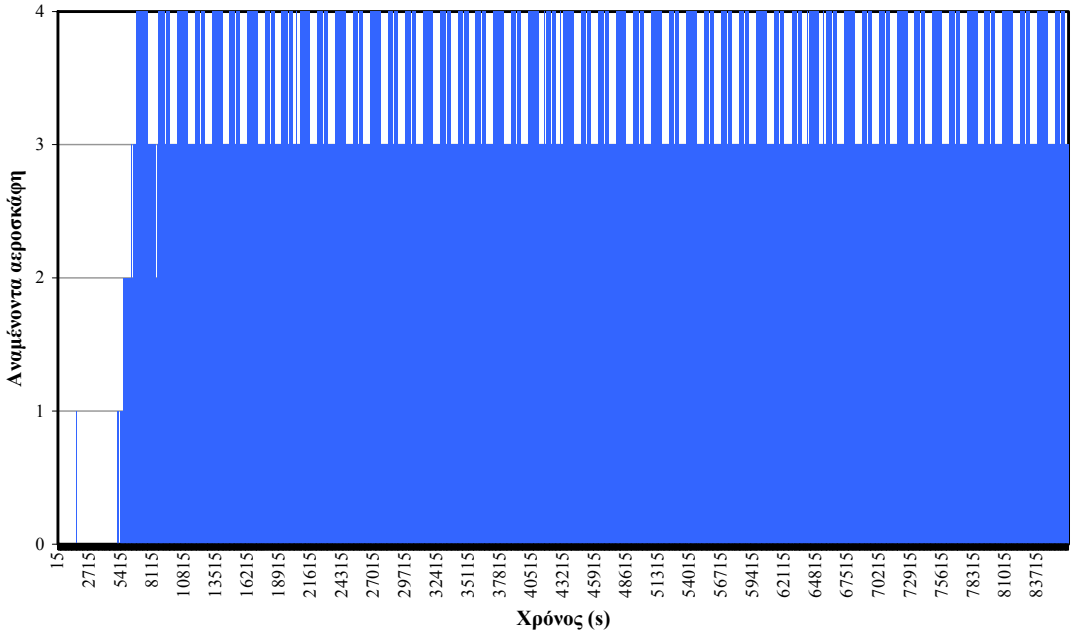


(β)

Σχήμα 4.13. Σύγκριση πλήθους αεροσκαφών στον buffer 2, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

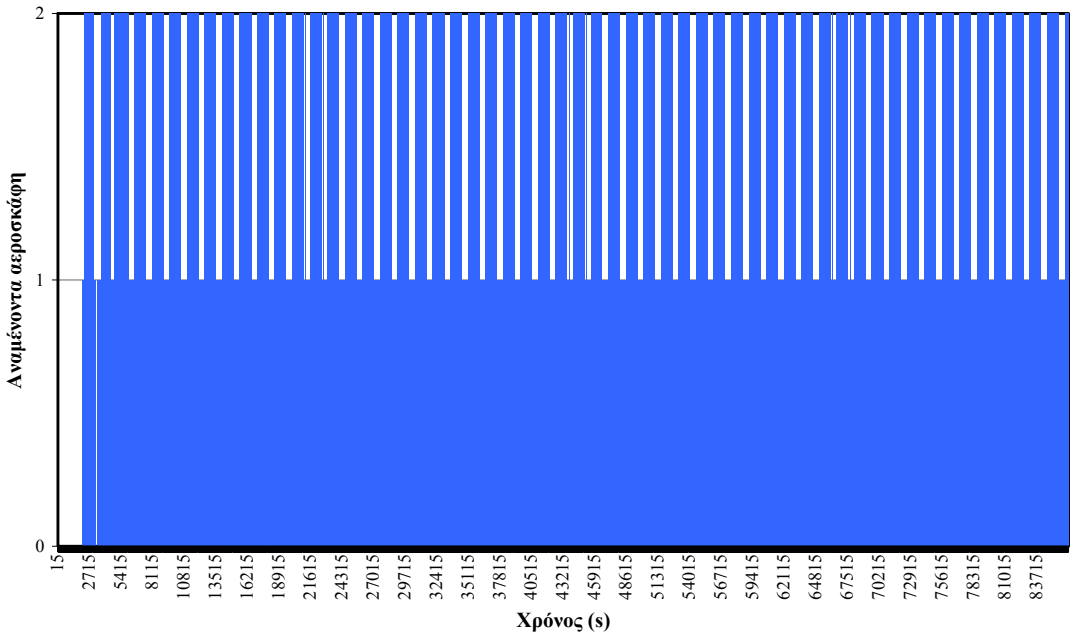


(α)

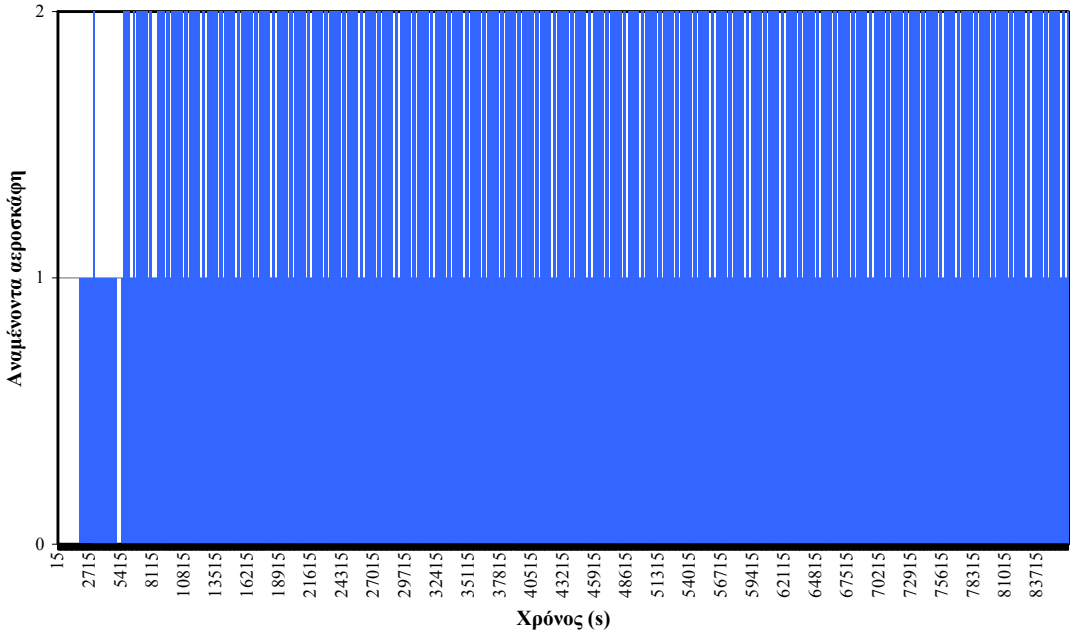


(β)

Σχήμα 4.14. Σύγκριση πλήθους αεροσκαφών στον buffer 3, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

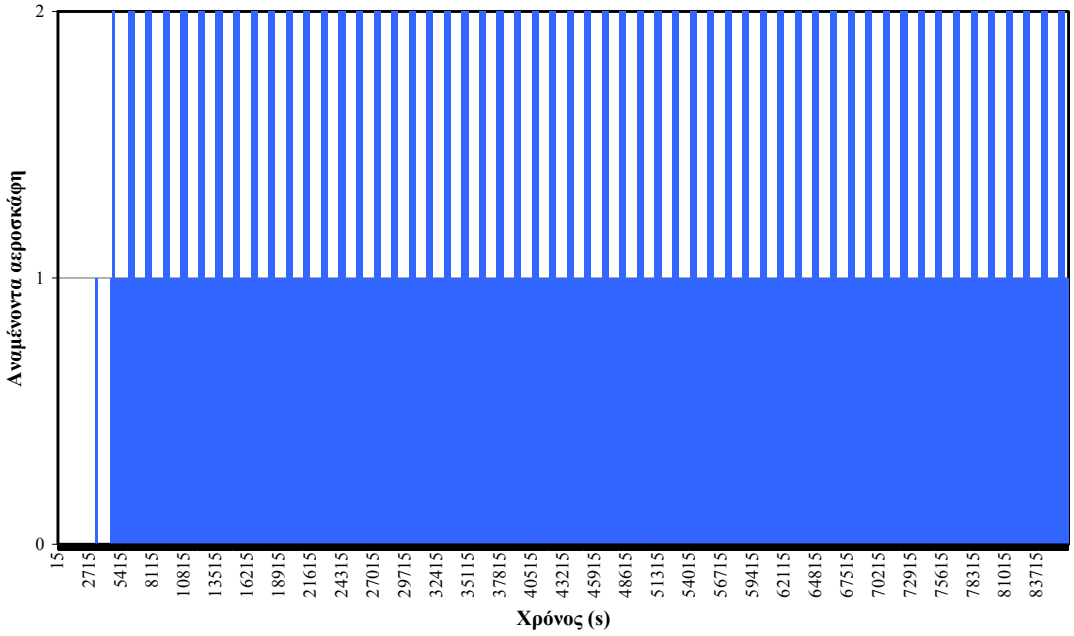


(α)

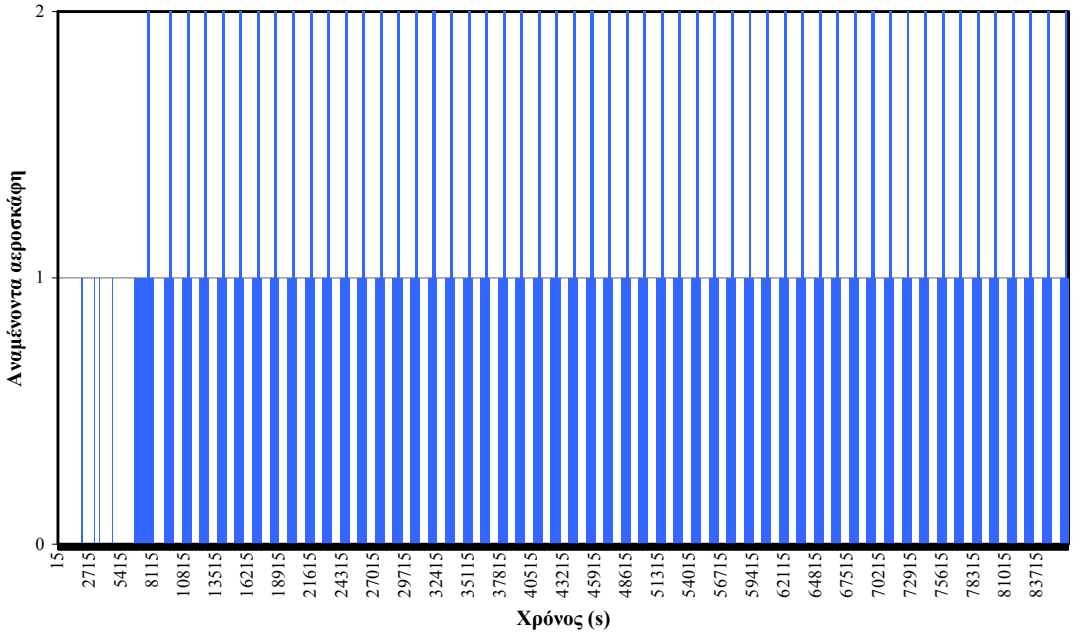


(β)

Σχήμα 4.15. Σύγκριση πλήθους αεροσκαφών στον buffer 4, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα A, (β) με χρήση του κώδικα B.



(α)



(β)

Σχήμα 4.16. Σύγκριση πλήθους αεροσκαφών στον buffer 5, σε ένα 24-ωρο για ρυθμό αφίξεων 10 αφη/hr: (α) με χρήση του κώδικα Α, (β) με χρήση του κώδικα Β.

ΚΕΦΑΛΑΙΟ 5. Συμπεράσματα

Στην παρούσα διπλωματική εργασία προσπαθήσαμε να μοντελοποιήσουμε και να προσομοιώσουμε την λειτουργία ενός αερολιμένα, χρησιμοποιώντας δύο διαφορετικές προσεγγίσεις ελέγχου.

Διαπιστώθηκε, πρώτον, ότι το πακέτο MATLAB ως υπολογιστικό εργαλείο καλύπτει τις ανάγκες διακριτής προσομοίωσης ελέγχου λειτουργίας αερολιμένα.

Όσον αφορά την συμπεριφορά και απόδοση των διαφορετικών τρόπων ελέγχου μπορούμε να πούμε ότι ο έλεγχος με βάσει το κριτήριο κόστους υπερτερεί έναντι του σειριακού ελέγχου, κυρίως στα θέματα διαχείρισης του αεροσκάφους στις μηχανές και στους buffers. Ωστόσο, η μοντελοποίηση που παρουσιάσθηκε σε αυτή την διπλωματική υστερεί σε κάποια σημεία σε σχέση με τις πραγματικές συνθήκες λειτουργίας ενός αερολιμένα καθώς δεν λαμβάνεται υπόψη το γεγονός ότι ο αριθμός αφίξεων ανά ώρα δεν είναι σταθερός κατά την διάρκεια ενός 24ώρου. Επίσης, στη παρούσα δεν μοντελοποιούνται αεροσκάφη που σταθμεύουν στον αερολιμένα και δεν μοντελοποιείται και η αξιοπιστία των μηχανών (χρόνος μη λειτουργίας της μηχανής λόγω βλάβης ή συντήρησης).

Θα αποτελούσε ενδιαφέρον αντικείμενο μελέτης η συμπεριφορά των δύο αυτών τρόπων ελέγχου για την διάρκεια ενός έτους, έχοντας εισαγάγει στην μοντελοποίηση ένα μη σταθερό ωριαίο αριθμό αφίξεων αεροσκαφών σε ένα 24ωρο καθώς και τις αξιοπιστίες των μηχανών.

ΠΑΡΑΡΤΗΜΑ Α

ΚΩΔΙΚΕΣ ΕΠΙΛΥΣΗΣ ΠΡΟΣΟΜΟΙΩΣΗΣ

A1.Κώδικας μοντέλου με σειριακό έλεγχο (ΚΩΔΙΚΑΣ Α)

A1.1. Αρχείο airport2a.m

```
% Airport ground operations simulation model
% Manufacturing cell modelling
% Case study 2: airport with two runways, one taxiway and two hubs
% (six submachines) - submachines connected with five buffers.
% Input and output buffers have infinite capacity.
% Submachine processing time, internal buffer capacity and arrival
rate
% are user-defined. Arrivals follow Poisson distribution, producing
output
% every 60 sec (limited to maximum one arrival/minute).
% Simulation time base: 1 sec, duration: user defined (in sec)
% System state data are stored every 15 sec

% definition of constants - user inputs - initialization of variables
% clear workspace
clear

% nr of submachines
nsub = 6;

% nr of buffers
nbuf = 5;

% user input for submachine processing times
proc_time(1)=input('\n Processing time for submachine 1 (sec): ');
proc_time(2)=input('\n Processing time for submachine 2 (sec): ');
proc_time(3)=input('\n Processing time for submachine 3 (sec): ');
proc_time(4)=input('\n Processing time for submachine 4 (sec): ');
proc_time(5)=input('\n Processing time for submachine 5 (sec): ');
proc_time(6)=input('\n Processing time for submachine 6 (sec): ');
% convert to column vector
proc_time = proc_time';

% user input for buffer capacities (buffers 3-5)
% negative means infinite capacity
buf_cap(1) = -1;
buf_cap(2) = -1;
buf_cap(3)=input('\n Capacity of buffer 3: ');
buf_cap(4)=input('\n Capacity of buffer 4: ');
buf_cap(5)=input('\n Capacity of buffer 5: ');
% convert to column vector
buf_cap = buf_cap';
% buffer 3 serves both incoming and outgoing traffic; dynamic
allocation
% of capacity to incoming and outgoing traffic. Initially, buffer 3
% capacity is totally allocated to incoming traffic
buf3_iq_cap = buf_cap(3);
buf3_oq_cap = 0;

% user input for arrival rate - the mean number of arrivals
% defines the lambda parameter (also mean value) of the Poisson
distribution
% that generates the arrivals every 1 minute
m = input('\n Mean number of arrivals per hour (0 < mean < 60):');
lambda = m./60;

% user input for simulation time
```

```
tsim = input('\n Simulation time (sec): ');

% initialization of buffers (all buffers empty)
buf1 = [];
buf2 = [];
buf3_iq = zeros(1,buf_cap(3));  % buffer 3 incoming queue
buf3_oq = zeros(1,buf_cap(3));  % buffer 3 outgoing queue
buf4 = zeros(1,buf_cap(4));
buf5 = zeros(1,buf_cap(5));

% initialization of buffer log vector (zero log for all buffers)
buf_log = zeros(nbuf,1);
buf3_iq_log = 0;
buf3_oq_log = 0;

% initialization of buffer full flags vector (all buffers not full,
except for buffer 3 outgoing queue
% (no capacity available))
buf_flag = zeros(nbuf,1);
buf3_iq_flag = 0;
buf3_oq_flag = 1;

% initialization of submachine flags vector (all submachines idle)
subm_flag = zeros(nsub,1);

% initialization of submachine outputs (zero output for all
submachines)
subm_out = zeros(nsub,1);

% initialization of submachines (all submachines empty)
subm1 = 0;
subm2 = 0;
subm3 = 0;
subm4 = 0;
subm5 = 0;
subm6 = 0;

% Initialization of submachines start/end processing times storing
matrices
% column 1: time instant, column2: submachine flag - k1,...,k6:
counters
subm1_proc_hist(1,:) = [1 0];
subm2_proc_hist(1,:) = [1 0];
subm3_proc_hist(1,:) = [1 0];
subm4_proc_hist(1,:) = [1 0];
subm5_proc_hist(1,:) = [1 0];
subm6_proc_hist(1,:) = [1 0];
k1 = 1;
k2 = 1;
k3 = 1;
k4 = 1;
k5 = 1;
k6 = 1;

% initialization of arrival counter and arrival vector, containing
the arrival
% times of all airplanes (in ascending order) - arrival time is
unique for each
% airplane (generation of one arrival/minute at maximum)

ar_count = 0;
```

```
ar_times = [];

% initialization of departure counter and departure vector,
containing the departure
% times of all airplanes served (in ascending order) - departure time
is unique for
% each airplane (one airplane is using the departure runway at a
time)

dep_count = 0;
dep_times = [];

% initialization of total time vector, containing total service times
for all
% airplanes served

total_times = [];


% main simulation loop
for t = 1:tsim

  % check for new arrival every 60 sec - arrivals follow Poisson
distribution
  % limited to maximum one arrival/minute
  if ~mod(t,60)
    new_ar = poissrnd(lambda);
    if new_ar >= 1
      new_ar = 1;
    else
      new_ar = 0;
    end

    % input of new arrival to the system (airplane arrival times are
propagated)
    if new_ar
      ar_count = ar_count + 1;
      ar_times(ar_count) = t;
      % push new arrival to buffer 1
      [buf_log(1), buf_flag(1), buf1] = push_buf(t, buf_log(1), buf1,
buf_cap(1));
    end
  end


  % input to submachine 1: buffer 1 not empty AND submachine 1 idle
AND
  % incoming queue of buffer 3 not full AND buffer 5 not full (the
last
  % condition is to prevent system output from blocking)
  if buf_log(1) & ~subm_flag(1) & ~buf3_iq_flag & ~buf_flag(5)
    % pop buffer 1 to submachine 1, which changes state to processing
    [subm1,   subm_flag(1),   start_proc_time1,   buf1,   buf_log(1),
buf_flag(1)] = start_proc(buf1, buf_log(1), t);
    % submachine 1 start processing time storing
    k1 = k1+1;
    subm1_proc_hist(k1,:) = [start_proc_time1 subm_flag(1)];

  else % check processing progress if submachine 1 is in processing
state
    if subm_flag(1)
```

```
      subm_out(1)    =    subm_proc(subm1,    start_proc_time1,    t,
proc_time(1));
      if subm_out(1)  % processing complete, push submachine 1 output
to buffer 3 incoming queue and change state to idle
        [subm1,  subm_flag(1),  subm_out(1),  end_proc_time1,  buf3_iq,
buf3_iq_log,    buf3_iq_flag]    =    end_proc(subm_out(1),    buf3_iq,
buf3_iq_log, buf3_iq_cap, t);
        % submachine 1 end processing time storing
        k1 = k1+1;
        subm1_proc_hist(k1,:) = [end_proc_time1 subm_flag(1)];
      end
    end
  end


  % input to submachine 3: buffer 3 incoming queue not empty AND
submachine 3 idle AND
  % submachine 4 idle AND buffer 4 not full
  if buf3_iq_log & ~subm_flag(3) & ~subm_flag(4) & ~buf_flag(4)
    % pop buffer 3 incoming queue to submachine 3, which changes
state to processing
    [subm3,  subm_flag(3),  start_proc_time3,  buf3_iq,  buf3_iq_log,
buf3_iq_flag] = start_proc(buf3_iq, buf3_iq_log, t);
    % submachine 3 start processing time storing
    k3 = k3+1;
    subm3_proc_hist(k3,:) = [start_proc_time3 subm_flag(3)];

  else % check processing progress if submachine 3 is in processing
state
    if subm_flag(3)
      subm_out(3)    =    subm_proc(subm3,    start_proc_time3,    t,
proc_time(3));
      if subm_out(3)  % processing complete, push submachine 3 output
to buffer 4 and change state to idle
        [subm3,  subm_flag(3),  subm_out(3),  end_proc_time3,  buf4,
buf_log(4),  buf_flag(4)]  =  end_proc(subm_out(3),  buf4,  buf_log(4),
buf_cap(4), t);
        % submachine 3 end processing time storing
        k3 = k3+1;
        subm3_proc_hist(k3,:) = [end_proc_time3 subm_flag(3)];
      end
    end
  end

  % allocation of capacity to outgoing queue of buffer 3 (positions
not used from incoming traffic)
  [buf3_oq_cap,  buf3_oq_flag]  =  cap_alloc(buf_cap(3),  buf3_iq_log,
buf3_oq_log, subm_flag(1));


  % input to hubs (submachines 5, 6) - starting from submachine 5,
pop buffer 4
  % to the first idle submachine
  if buf_log(4)  % buffer 4 not empty

    if ~subm_flag(5)  % submachine 5 idle
      % pop buffer 4 to submachine 5, which changes state to
processing
      [subm5,  subm_flag(5),  start_proc_time5,  buf4,  buf_log(4),
buf_flag(4)] = start_proc(buf4, buf_log(4), t);
      % submachine 5 start processing time storing
```

```
      k5 = k5+1;
      subm5_proc_hist(k5,:) = [start_proc_time5 subm_flag(5)];

    else
      if ~subm_flag(6)  % submachine 6 idle
        % pop buffer 4 to submachine 6, which changes state to
processing
        [subm6, subm_flag(6), start_proc_time6, buf4, buf_log(4),
buf_flag(4)] = start_proc(buf4, buf_log(4), t);
        % submachine 6 start processing time storing
        k6 = k6+1;
        subm6_proc_hist(k6,:) = [start_proc_time6 subm_flag(6)];
      end
    end
  end

  % check processing progress if submachines 5-6 are in processing
state
  if subm_flag(5)
    subm_out(5)    =    subm_proc(subm5,    start_proc_time5,    t,
proc_time(5));
  end
  if subm_flag(6)
    subm_out(6)    =    subm_proc(subm6,    start_proc_time6,    t,
proc_time(6));
  end

  if ~buf_flag(5)  % buffer 5 not full
    % check ouputs of submachines 5-6 to find the oldest (in case
that more than one
    % hubs are ready, the airplane with the smallest (oldest) system
input time is served first
    index = find_hub2(subm_out(5), subm_out(6));

    if index  % there is input available for buffer 5
      if index == 5    % push submachine 5 output to buffer 5 and
change state to idle
        [subm5, subm_flag(5), subm_out(5), end_proc_time5, buf5,
buf_log(5), buf_flag(5)] = end_proc(subm_out(5), buf5, buf_log(5),
buf_cap(5), t);
        % submachine 5 end processing time storing
        k5 = k5+1;
        subm5_proc_hist(k5,:) = [end_proc_time5 subm_flag(5)];

      else
        % push submachine 6 output to buffer 5 and change state to
idle
        [subm6, subm_flag(6), subm_out(6), end_proc_time6, buf5,
buf_log(5), buf_flag(5)] = end_proc(subm_out(6), buf5, buf_log(5),
buf_cap(5), t);
        % submachine 6 end processing time storing
        k6 = k6+1;
        subm6_proc_hist(k6,:) = [end_proc_time6 subm_flag(6)];
      end
    end
  end


  % input to submachine 4: buffer 5 not empty AND submachine 4 idle
AND
  % submachine 3 idle AND buffer 3 outgoing queue not full
```

```
  if buf_log(5) & ~subm_flag(4) & ~subm_flag(3) & ~buf3_oq_flag
    % pop buffer 5 to submachine 4, which changes state to processing
    [subm4,   subm_flag(4),   start_proc_time4,   buf5,   buf_log(5),
buf_flag(5)] = start_proc(buf5, buf_log(5), t);
    % submachine 4 start processing time storing
    k4 = k4+1;
    subm4_proc_hist(k4,:) = [start_proc_time4 subm_flag(4)];

  else % check processing progress if submachine 4 is in processing
state
    if subm_flag(4)
      subm_out(4)    =    subm_proc(subm4,    start_proc_time4,    t,
proc_time(4));
      if subm_out(4)  % processing complete, push submachine 4 output
to buffer 3 outgoing queue and change state to idle
        [subm4,  subm_flag(4),  subm_out(4),  end_proc_time4,  buf3_oq,
buf3_oq_log,    buf3_oq_flag]    =    end_proc(subm_out(4),    buf3_oq,
buf3_oq_log, buf3_oq_cap, t);
        % submachine 4 end processing time storing
        k4 = k4+1;
        subm4_proc_hist(k4,:) = [end_proc_time4 subm_flag(4)];
      end
    end
  end


  % input to submachine 2: buffer 3 outgoing queue not empty AND
submachine 2 idle
  if buf3_oq_log & ~subm_flag(2)
    % pop buffer 3 outgoing queue to submachine 2, which changes
state to processing
    [subm2,  subm_flag(2),  start_proc_time2,  buf3_oq,  buf3_oq_log,
buf3_oq_flag] = start_proc(buf3_oq, buf3_oq_log, t);
    % submachine 2 start processing time storing
    k2 = k2+1;
    subm2_proc_hist(k2,:) = [start_proc_time2 subm_flag(2)];

  else % check processing progress if submachine 2 is in processing
state
    if subm_flag(2)
      subm_out(2)    =    subm_proc(subm2,    start_proc_time2,    t,
proc_time(2));
      if subm_out(2)
        % processing complete, airplane departure
        dep_count = dep_count + 1;
        dep_times(dep_count) = t;
        total_times(dep_count) = t - subm_out(2);    % total service
time for departuring airplane

        % push submachine 2 output to buffer 2 and change state to
idle
        [subm2,  subm_flag(2),  subm_out(2),  end_proc_time2,  buf2,
buf_log(2),  buf_flag(2)]  =  end_proc(subm_out(2),  buf2,  buf_log(2),
buf_cap(2), t);
        % submachine 2 end processing time storing
        k2 = k2+1;
        subm2_proc_hist(k2,:) = [end_proc_time2 subm_flag(2)];
      end
    end
  end
```

```
  % allocation of capacity to incoming queue of buffer 3 (positions
not used from outgoing traffic)
  [buf3_iq_cap,  buf3_iq_flag]  =  cap_alloc(buf_cap(3),  buf3_oq_log,
buf3_iq_log, subm_flag(4));


  % buffer 3 update
  buf_log(3) = buf3_iq_log + buf3_oq_log;
  if buf_log(3) == buf_cap(3)
    buf_flag(3)=1;
  else
    buf_flag(3)=0;
  end


  % simulation history storing (every 15 sec)
  if ~mod(t,15)
    k = t./15;
    t_hist(k) = t;   % storing time points
    buf3_iq_hist(k,:) = buf3_iq;   % buffer3 incoming queue storing
    buf3_oq_hist(k,:) = buf3_oq;   % buffer3 outgoing queue storing
    buf4_hist(k,:) = buf4;   % buffer4 storing
    buf5_hist(k,:) = buf5;   % buffer5 storing
    buf_log_hist(k,:) = buf_log';    % buffer log storing
    buf_flag_hist(k,:) = buf_flag';    % buffer flag storing
    subm_hist(k,:)  =  [subm1  subm2  subm3  subm4  subm5  subm6];     %
submachine storing
    subm_flag_hist(k,:) = subm_flag';   % submachine flag storing
    ar_hist(k) = ar_count;   % number of arrivals history
    dep_hist(k) = dep_count;   % number of departures history
    popul_hist(k)  =  ar_count  -  dep_count;     %  system  population
history
  end

end % main simulation loop

% results storing to text files (simulation history, arrivals and
departures)
% simulation history
res1 = [t_hist' buf_log_hist buf_flag_hist subm_hist subm_flag_hist
ar_hist' dep_hist' popul_hist'...
        buf3_iq_hist buf3_oq_hist buf4_hist buf5_hist];
save .\air2a_results1.txt res1 -ascii -tabs

% arrival, departure and total times
res2 = zeros(ar_count, 3);
res2(:,1) = ar_times';
res2(1:dep_count,2) = dep_times';
res2(1:dep_count,3) = total_times';
save .\air2a_results2.txt res2 -ascii -tabs

% submachine processing time matrices
save .\air2a_sub1_hist.txt subm1_proc_hist -ascii -tabs
save .\air2a_sub2_hist.txt subm2_proc_hist -ascii -tabs
save .\air2a_sub3_hist.txt subm3_proc_hist -ascii -tabs
save .\air2a_sub4_hist.txt subm4_proc_hist -ascii -tabs
save .\air2a_sub5_hist.txt subm5_proc_hist -ascii -tabs
save .\air2a_sub6_hist.txt subm6_proc_hist -ascii –tabs
```

A1.2. Αρχείο subm_proc.m

```
function subm_out = subm_proc(subm, proc_start_time,t,subm_proc_time)

% Function implementing output status check of a submachine which is
in processing state
%
% The function returns the output of the submachine, which is nonzero
% in case the submachine has completed processing
%
% This function is designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m

timer = t - proc_start_time;
if timer < subm_proc_time
    subm_out = 0;
else
    subm_out = subm;
End
```

A1.3. Αρχείο pop_buf.m

```
function    [new_buf_log,    new_buf_flag,    new_buf,    output]    =
pop_buf(buf_log, buf)

% Function implementing POP operation on a FIFO buffer
% designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m

if ~buf_log
  fprintf('\nwarning - empty buffer, pop not possible\n')
  new_buf_log = 0;
  new_buf_flag = 0;
  new_buf = buf;
  output = [];
else
  output = buf(1);
  if buf_log > 1
    for j = 1:buf_log-1
      buf(j) = buf(j+1);
    end
    buf(buf_log)=0;
    buf_log = buf_log-1;
  else
    buf(1) = 0;
    buf_log = 0;
  end
  new_buf_log = buf_log;
  new_buf_flag = 0;
  new_buf = buf;
End
```

A1.4. Αρχείο push_buf.m

```
function [new_buf_log, new_buf_flag, new_buf] = push_buf(input,
buf_log, buf, buf_size)

% Function implementing PUSH operation on a FIFO buffer
% designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m

if buf_log == buf_size
  fprintf('\nwarning - full buffer, push not possible\n')
  new_buf_log = buf_log;
  new_buf_flag = 1;
  new_buf = buf;
else
  buf(buf_log + 1) = input;
  new_buf_log = buf_log + 1;
  new_buf = buf;
  if new_buf_log == buf_size
   new_buf_flag = 1;
  else
   new_buf_flag = 0;
  end
end
```

A1.5. Αρχείο cap_alloc.m

```
function [b2_cap, b2_flag] = cap_alloc(buf_cap, b1_log, b2_log,
subm_flag)

% Function implementing dynamic capacity allocation between incoming
% and outgoing queues of a bidirectional buffer
%
% The function returns the new capacity allocated to queue b2
together
% with its new availability status, considering present usage of
buffer
% capacity from queue b1
%
% This function is designed to work together with the airport models
% airport2a.m and airport2b.m

% subm: the submachine controlling push operation of b1 queue
% buf_cap: total buffer capacity

b2_cap = buf_cap-b1_log;
if subm_flag   % when submachine is in processing state, reserve one
more position for b1 queue
  b2_cap = b2_cap-1;
end
if b2_log == b2_cap
  b2_flag = 1;
else
  b2_flag = 0;
End
```

A1.6. Αρχείο end_proc.m

```
function [subm, subm_flag, subm_out, end_time, obuf, obuf_log,
obuf_flag] = end_proc(subm_out, obuf, obuf_log, obuf_cap, t)

% Function implementing the change of submachine state from
processing to idle
% (push submachine output to output buffer)
%
% The function returns new submachine state and new input buffer
state
%
% This function is designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m and
% assigns a call to function push_buf (push buffer)

[obuf_log, obuf_flag, obuf] = push_buf(subm_out, obuf_log, obuf,
obuf_cap);
subm_out = 0;
subm = 0;
subm_flag = 0;
end_time = t;
```

A1.7. Αρχείο find_hub2.m

```
function index = find_hub2(out5, out6)

% Function implementing the search for the airplane with the oldest
% (smallest) system entry time among the ones that have completed
% service in the hubs (submachines 5-6)
%
% The function returns the index of the submachine to provide input
% to buffer 5. If no submachine is ready, the function returns zero
%
% This function is designed to work together with the airport models
% airport2a.m, airport2b.m

buf5_in = 0;
index = 0;
if out5  % submachine 5 ready
  buf5_in = out5;
  index = 5;
end

if out6  % submachine 6 ready
  if ~index  % submachine 5 not ready
    index = 6;
    buf5_in = out6;
  else  % submachine 5 also ready
    if out6 < buf5_in   % output of submachine 6 older than output of
submachine 5
      index = 6;
      buf5_in = out6;
    end
  end
end
```

A1.8. Αρχείο start_proc.m

```
function [subm, subm_flag, start_time, ibuf, ibuf_log, ibuf_flag] =
start_proc(ibuf, ibuf_log, t)

% Function implementing the change of submachine state to processing
% (pop input buffer to submachine, which changes state to processing)
%
% The function returns new submachine state and new input buffer
state
%
% This function is designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m and
% assigns a call to function pop_buf (pop buffer)

[ibuf_log, ibuf_flag, ibuf, ibuf_out] = pop_buf(ibuf_log, ibuf);
subm = ibuf_out;
start_time = t;
subm_flag = 1;
```

Α2.Κώδικας μοντέλου με έλεγχο βάσει του κριτηρίου κόστους (ΚΩΔΙΚΑΣ Β)

Α2.1. Αρχείο airport2b.m

```
% Airport ground operations simulation model
% Manufacturing cell modelling
% Cost function for dispatcing decision between conflicting submachines
% of multiproduct machines
% Case study 2: airport with two runways, one taxiway and two hubs
% (six submachines) - submachines connected with five buffers.
% Input and output buffers have infinite capacity.
% Submachine processing time, internal buffer capacity and arrival rate
% are user-defined. Arrivals follow Poisson distribution, producing output
% every 60 sec (limited to maximum one arrival/minute).
% Simulation time base: 1 sec, duration: user defined (in sec)
% System state data are stored every 15 sec

% definition of constants - user inputs - initialization of variables
% clear workspace
clear

% nr of submachines
nsub = 6;

% nr of buffers
nbuf = 5;

% coefficients of cost functionals
% J3
l31m = 1;
l31p = 1;
l32p = 1;
l33p = 1;
l34p = 1;
% J4
l41m = 1;
l42m = 1;
l43m = 1;
l41p = 1;
l42p = 1;


% user input for submachine processing times
proc_time(1)=input('\n Processing time for submachine 1 (sec): ');
proc_time(2)=input('\n Processing time for submachine 2 (sec): ');
proc_time(3)=input('\n Processing time for submachine 3 (sec): ');
proc_time(4)=input('\n Processing time for submachine 4 (sec): ');
proc_time(5)=input('\n Processing time for submachine 5 (sec): ');
proc_time(6)=input('\n Processing time for submachine 6 (sec): ');
% convert to column vector
proc_time = proc_time';

% user input for buffer capacities (buffers 3-6)
% negative means infinite capacity
buf_cap(1) = -1;
buf_cap(2) = -1;
```

```
buf_cap(3)=input('\n Capacity of buffer 3: ');
buf_cap(4)=input('\n Capacity of buffer 4: ');
buf_cap(5)=input('\n Capacity of buffer 5: ');
% convert to column vector
buf_cap = buf_cap';
% buffer 3 serves both incoming and outgoing traffic; dynamic
allocation
% of capacity to incoming and outgoing traffic. Initially, buffer 3
% capacity is totally allocated to incoming traffic
buf3_iq_cap = buf_cap(3);
buf3_oq_cap = 0;

% user input for arrival rate - the mean number of arrivals
% defines the lambda parameter (also mean value) of the Poisson
distribution
% that generates the arrivals every 1 minute
m = input('\n Mean number of arrivals per hour (0 < mean < 60):');
lambda = m./60;

% user input for simulation time
tsim = input('\n Simulation time (sec): ');

% initialization of buffers (all buffers empty)
buf1 = [];
buf2 = [];
buf3_iq = zeros(1,buf_cap(3));  % buffer 3 incoming queue
buf3_oq = zeros(1,buf_cap(3));  % buffer 3 outgoing queue
buf4 = zeros(1,buf_cap(4));
buf5 = zeros(1,buf_cap(5));

% initialization of buffer log vector (zero log for all buffers)
buf_log = zeros(nbuf,1);
buf3_iq_log = 0;
buf3_oq_log = 0;

% initialization of buffer full flags vector (all buffers not full,
except for buffer 3 outgoing queue
% (no capacity available))
buf_flag = zeros(nbuf,1);
buf3_iq_flag = 0;
buf3_oq_flag = 1;

% initialization of submachine flags vector (all submachines idle)
subm_flag = zeros(nsub,1);

% initialization of submachine outputs (zero output for all
submachines)
subm_out = zeros(nsub,1);

% initialization of submachines (all submachines empty)
subm1 = 0;
subm2 = 0;
subm3 = 0;
subm4 = 0;
subm5 = 0;
subm6 = 0;

% Initialization of submachines start/end processing times storing
matrices
% column 1: time instant, column2: submachine flag - k1,...,k8:
counters
```

```
subm1_proc_hist(1,:) = [1 0];
subm2_proc_hist(1,:) = [1 0];
subm3_proc_hist(1,:) = [1 0];
subm4_proc_hist(1,:) = [1 0];
subm5_proc_hist(1,:) = [1 0];
subm6_proc_hist(1,:) = [1 0];
k1 = 1;
k2 = 1;
k3 = 1;
k4 = 1;
k5 = 1;
k6 = 1;

% initialization of arrival counter and arrival vector, containing
the arrival
% times of all airplanes (in ascending order) - arrival time is
unique for each
% airplane (generation of one arrival/minute at maximum)

ar_count = 0;
ar_times = [];

% initialization of departure counter and departure vector,
containing the departure
% times of all airplanes served (in ascending order) - departure time
is unique for
% each airplane (one airplane is using the departure runway at a
time)

dep_count = 0;
dep_times = [];

% initialization of total time vector, containing total service times
for all
% airplanes served

total_times = [];


% main simulation loop
for t = 1:tsim

  % check for new arrival every 60 sec - arrivals follow Poisson
distribution
  % limited to maximum one arrival/minute
  if ~mod(t,60)
    new_ar = poissrnd(lambda);
    if new_ar >= 1
      new_ar = 1;
    else
      new_ar = 0;
    end

    % input of new arrival to the system (airplane arrival times are
propagated)
    if new_ar
      ar_count = ar_count + 1;
      ar_times(ar_count) = t;
      % push new arrival to buffer 1
      [buf_log(1), buf_flag(1), buf1] = push_buf(t, buf_log(1), buf1,
buf_cap(1));
```

```
    end
  end


  % input to submachine 1: buffer 1 not empty AND submachine 1 idle
AND
  % incoming queue of buffer 3 not full AND buffer 5 not full (the
last
  % condition is to prevent system output from blocking)
  if buf_log(1) & ~subm_flag(1) & ~buf3_iq_flag & ~buf_flag(5)
    % pop buffer 1 to submachine 1, which changes state to processing
    [subm1,   subm_flag(1),   start_proc_time1,   buf1,   buf_log(1),
buf_flag(1)] = start_proc(buf1, buf_log(1), t);
    % submachine 1 start processing time storing
    k1 = k1+1;
    subm1_proc_hist(k1,:) = [start_proc_time1 subm_flag(1)];

  else % check processing progress if submachine 1 is in processing
state
    if subm_flag(1)
      subm_out(1)    =    subm_proc(subm1,    start_proc_time1,    t,
proc_time(1));
      if subm_out(1)  % processing complete, push submachine 1 output
to buffer 3 incoming queue and change state to idle
        [subm1,  subm_flag(1),  subm_out(1),  end_proc_time1,  buf3_iq,
buf3_iq_log,    buf3_iq_flag]    =    end_proc(subm_out(1),    buf3_iq,
buf3_iq_log, buf3_iq_cap, t);
        % submachine 1 end processing time storing
        k1 = k1+1;
        subm1_proc_hist(k1,:) = [end_proc_time1 subm_flag(1)];
      end
    end
  end

  % allocation of capacity to outgoing queue of buffer 3 (positions
not used from incoming traffic)
  [buf3_oq_cap,  buf3_oq_flag]  =  cap_alloc(buf_cap(3),  buf3_iq_log,
buf3_oq_log, subm_flag(1));


  % input to submachine 2: buffer 3 outgoing queue not empty AND
submachine 2 idle
  if buf3_oq_log & ~subm_flag(2)
    % pop buffer 3 outgoing queue to submachine 2, which changes
state to processing
    [subm2,  subm_flag(2),  start_proc_time2,  buf3_oq,  buf3_oq_log,
buf3_oq_flag] = start_proc(buf3_oq, buf3_oq_log, t);
    % submachine 2 start processing time storing
    k2 = k2+1;
    subm2_proc_hist(k2,:) = [start_proc_time2 subm_flag(2)];

    % allocation of capacity to incoming queue of buffer 3 (positions
not used from outgoing traffic)
    [buf3_iq_cap, buf3_iq_flag] = cap_alloc(buf_cap(3), buf3_oq_log,
buf3_iq_log, subm_flag(4));

  else % check processing progress if submachine 2 is in processing
state
    if subm_flag(2)
      subm_out(2)    =    subm_proc(subm2,    start_proc_time2,    t,
proc_time(2));
```

```
      if subm_out(2)
        % processing complete, airplane departure
        dep_count = dep_count + 1;
        dep_times(dep_count) = t;
        total_times(dep_count) = t - subm_out(2);    % total service
time for departuring airplane

        % push submachine 2 output to buffer 2 and change state to
idle
        [subm2,  subm_flag(2),  subm_out(2),  end_proc_time2,  buf2,
buf_log(2),  buf_flag(2)]  =  end_proc(subm_out(2),  buf2,  buf_log(2),
buf_cap(2), t);
        % submachine 2 end processing time storing
        k2 = k2+1;
        subm2_proc_hist(k2,:) = [end_proc_time2 subm_flag(2)];
      end
    end
  end


  % Operation of machine C (submachines 3 and 4)
  % Idling machine
  if ~subm_flag(3) & ~subm_flag(4)
    if  ~buf_log(5)  |  buf3_oq_flag    % buffer 5 empty or buffer 3
outgoing full -> submachine 4 cannot start processing
      if  buf3_iq_log & ~buf_flag(4)  % buffer 3 incoming queue not
empty AND buffer 4 not full
        % pop buffer 3 incoming queue to submachine 3, which changes
state to processing
        [subm3, subm_flag(3), start_proc_time3, buf3_iq, buf3_iq_log,
buf3_iq_flag] = start_proc(buf3_iq, buf3_iq_log, t);
      end

    else   % buffer 5 not empty and buffer 3 outgoing not full
      if ~buf3_iq_log | buf_flag(4)  % buffer 3 incoming queue empty
or buffer 4 full-> submachine 3 cannot start processing
        %  pop  buffer  5  to  submachine  4,  which  changes  state  to
processing
        [subm4,  subm_flag(4),  start_proc_time4,  buf5,  buf_log(5),
buf_flag(5)] = start_proc(buf5, buf_log(5), t);

      else  % buffer 3 incoming queue not empty and buffer 4 not full
        % cost functions for submachines 3 and 4 - submachine with
the higher
        % cost is given priority
        x3 = buf3_iq_log-buf3_iq_cap./2;
        x4 = buf_log(4)-buf_cap(4)./2;
        x5 = buf_log(5)-buf_cap(5)./2;
        x6 = buf3_oq_log-buf3_oq_cap./2;
        e = buf_log(2);
        J3      =      l31m.*logsig(x3)+l31p.*(1-logsig(x4))+l32p.*(1-
logsig(x5))+l33p.*(1-logsig(x6))+l34p.*((e.^2)./(1+e.^2));
        J4                                                          =
l41m.*logsig(x3)+l42m.*logsig(x4)+l43m.*logsig(x5)+l41p.*(1-
logsig(x6))+l42p.*((e.^2)./(1+e.^2));
        if J3 > J4
          %  pop  buffer  3  incoming  queue  to  submachine  3,  which
changes state to processing
          [subm3,     subm_flag(3),     start_proc_time3,     buf3_iq,
buf3_iq_log, buf3_iq_flag] = start_proc(buf3_iq, buf3_iq_log, t);
        else   % J3 <= J4
```

```
          % pop buffer 5 to submachine 4, which changes state to
processing
          [subm4, subm_flag(4), start_proc_time4, buf5, buf_log(5),
buf_flag(5)] = start_proc(buf5, buf_log(5), t);
        end
      end
    end

    % submachine 3 start processing time storing
    if subm_flag(3)
      k3 = k3+1;
      subm3_proc_hist(k3,:) = [start_proc_time3 subm_flag(3)];
    end

    % submachine 4 start processing time storing
    if subm_flag(4)
      k4 = k4+1;
      subm4_proc_hist(k4,:) = [start_proc_time4 subm_flag(4)];
    end

  % Machine in processing state
  elseif subm_flag(3)  % submachine 3 processing -> submachine 4 idle
    % check processing progress of submachine 3
    subm_out(3)     =     subm_proc(subm3,     start_proc_time3,     t,
proc_time(3));
    if subm_out(3)  % processing complete, push submachine 3 output
to buffer 4 and change state to idle
      [subm3,   subm_flag(3),   subm_out(3),   end_proc_time3,   buf4,
buf_log(4),  buf_flag(4)]  =  end_proc(subm_out(3),  buf4,  buf_log(4),
buf_cap(4), t);
    end
    % submachine 3 end processing time storing
    if ~subm_flag(3)
      k3 = k3+1;
      subm3_proc_hist(k3,:) = [end_proc_time3 subm_flag(3)];
    end

  else   % submachine 4 processing -> submachine 3 idle
    % check processing progress of submachine 4
    subm_out(4)     =     subm_proc(subm4,     start_proc_time4,     t,
proc_time(4));
    if subm_out(4)  % processing complete, push submachine 4 output
to buffer 3 outgoing queue and change state to idle
      [subm4,  subm_flag(4),  subm_out(4),  end_proc_time4,  buf3_oq,
buf3_oq_log,    buf3_oq_flag]    =    end_proc(subm_out(4),    buf3_oq,
buf3_oq_log, buf3_oq_cap, t);
    end
    % submachine 4 end processing time storing
    if ~subm_flag(4)
      k4 = k4+1;
      subm4_proc_hist(k4,:) = [end_proc_time4 subm_flag(4)];
    end
  end

  % allocation of capacity to outgoing queue of buffer 3 (positions
not used from incoming traffic)
  [buf3_oq_cap,  buf3_oq_flag]  =  cap_alloc(buf_cap(3),  buf3_iq_log,
buf3_oq_log, subm_flag(1));

  % allocation of capacity to incoming queue of buffer 3 (positions
not used from outgoing traffic)
```

```
  [buf3_iq_cap,  buf3_iq_flag]  =  cap_alloc(buf_cap(3),  buf3_oq_log,
buf3_iq_log, subm_flag(4));

  % buffer 3 update
  buf_log(3) = buf3_iq_log + buf3_oq_log;
  if buf_log(3) == buf_cap(3)
    buf_flag(3)=1;
  else
    buf_flag(3)=0;
  end


  % Operation of machines D, E (submachines 5, 6)
  % input to hubs - starting from submachine 5, pop buffer 4 to the
first idle submachine
  if buf_log(4)  % buffer 4 not empty

    if ~subm_flag(5)  % submachine 5 idle
      %  pop  buffer  4  to  submachine  5,  which  changes  state  to
processing
      [subm5,  subm_flag(5),  start_proc_time5,  buf4,  buf_log(4),
buf_flag(4)] = start_proc(buf4, buf_log(4), t);
      % submachine 5 start processing time storing
      k5 = k5+1;
      subm5_proc_hist(k5,:) = [start_proc_time5 subm_flag(5)];

    else
      if ~subm_flag(6)  % submachine 6 idle
        %  pop  buffer  4  to  submachine  6,  which  changes  state  to
processing
        [subm6,  subm_flag(6),  start_proc_time6,  buf4,  buf_log(4),
buf_flag(4)] = start_proc(buf4, buf_log(4), t);
        % submachine 6 start processing time storing
        k6 = k6+1;
        subm6_proc_hist(k6,:) = [start_proc_time6 subm_flag(6)];
      end
    end
  end

  % check processing progress if submachines 5-6 are in processing
state
  if subm_flag(5)
    subm_out(5)    =    subm_proc(subm5,    start_proc_time5,    t,
proc_time(5));
  end
  if subm_flag(6)
    subm_out(6)    =    subm_proc(subm6,    start_proc_time6,    t,
proc_time(6));
  end

  if ~buf_flag(5)  % buffer 5 not full
    % check ouputs of submachines 5-6 to find the oldest (in case
that more than one
    % hubs are ready, the airplane with the smallest (oldest) system
input time is served first
    index = find_hub2(subm_out(5), subm_out(6));

    if index  % there is input available for buffer 5
      if index == 5    % push submachine 5 output to buffer 5 and
change state to idle
```

```
        [subm5,  subm_flag(5),  subm_out(5),  end_proc_time5,  buf5,
buf_log(5),  buf_flag(5)]  =  end_proc(subm_out(5),  buf5,  buf_log(5),
buf_cap(5), t);
        % submachine 5 end processing time storing
        k5 = k5+1;
        subm5_proc_hist(k5,:) = [end_proc_time5 subm_flag(5)];

      else
        % push submachine 6 output to buffer 5 and change state to
idle
        [subm6,  subm_flag(6),  subm_out(6),  end_proc_time6,  buf5,
buf_log(5),  buf_flag(5)]  =  end_proc(subm_out(6),  buf5,  buf_log(5),
buf_cap(5), t);
        % submachine 6 end processing time storing
        k6 = k6+1;
        subm6_proc_hist(k6,:) = [end_proc_time6 subm_flag(6)];
      end
    end
  end


  % simulation history storing (every 15 sec)
  if ~mod(t,15)
    k = t./15;
    t_hist(k) = t;   % storing time points
    buf3_iq_hist(k,:) = buf3_iq;   % buffer3 incoming queue storing
    buf3_oq_hist(k,:) = buf3_oq;   % buffer3 outgoing queue storing
    buf4_hist(k,:) = buf4;   % buffer4 storing
    buf5_hist(k,:) = buf5;   % buffer5 storing
    buf_log_hist(k,:) = buf_log';    % buffer log storing
    buf_flag_hist(k,:) = buf_flag';    % buffer flag storing
    subm_hist(k,:)  =  [subm1  subm2  subm3  subm4  subm5  subm6];     %
submachine storing
    subm_flag_hist(k,:) = subm_flag';   % submachine flag storing
    ar_hist(k) = ar_count;   % number of arrivals history
    dep_hist(k) = dep_count;   % number of departures history
    popul_hist(k)  =  ar_count  -  dep_count;     %  system  population
history
  end

end % main simulation loop

% results storing to text files (simulation history, arrivals and
departures)
% simulation history
res1 = [t_hist' buf_log_hist buf_flag_hist subm_hist subm_flag_hist
ar_hist' dep_hist' popul_hist'...
        buf3_iq_hist buf3_oq_hist buf4_hist buf5_hist];
save .\air2b_results1.txt res1 -ascii -tabs

% arrival, departure and total times
res2 = zeros(ar_count, 3);
res2(:,1) = ar_times';
res2(1:dep_count,2) = dep_times';
res2(1:dep_count,3) = total_times';
save .\air2b_results2.txt res2 -ascii -tabs

% submachine processing time matrices
save .\air2b_sub1_hist.txt subm1_proc_hist -ascii -tabs
save .\air2b_sub2_hist.txt subm2_proc_hist -ascii -tabs
save .\air2b_sub3_hist.txt subm3_proc_hist -ascii -tabs
```

```
save .\air2b_sub4_hist.txt subm4_proc_hist -ascii -tabs
save .\air2b_sub5_hist.txt subm5_proc_hist -ascii -tabs
save .\air2b_sub6_hist.txt subm6_proc_hist -ascii -tabs
```

A2.2. Αρχείο subm_proc.m

```
function subm_out = subm_proc(subm, proc_start_time,t,subm_proc_time)

% Function implementing output status check of a submachine which is
in processing state
%
% The function returns the output of the submachine, which is nonzero
% in case the submachine has completed processing
%
% This function is designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m

timer = t - proc_start_time;
if timer < subm_proc_time
    subm_out = 0;
else
    subm_out = subm;
End
```

A2.3. Αρχείο pop_buf.m

```
function    [new_buf_log,    new_buf_flag,    new_buf,    output]    =
pop_buf(buf_log, buf)

% Function implementing POP operation on a FIFO buffer
% designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m

if ~buf_log
  fprintf('\nwarning - empty buffer, pop not possible\n')
  new_buf_log = 0;
  new_buf_flag = 0;
  new_buf = buf;
  output = [];
else
  output = buf(1);
  if buf_log > 1
    for j = 1:buf_log-1
      buf(j) = buf(j+1);
    end
    buf(buf_log)=0;
    buf_log = buf_log-1;
  else
    buf(1) = 0;
    buf_log = 0;
  end
  new_buf_log = buf_log;
  new_buf_flag = 0;
  new_buf = buf;
End
```

A2.4. Αρχείο push_buf.m

```
function [new_buf_log, new_buf_flag, new_buf] = push_buf(input,
buf_log, buf, buf_size)

% Function implementing PUSH operation on a FIFO buffer
% designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m

if buf_log == buf_size
  fprintf('\nwarning - full buffer, push not possible\n')
  new_buf_log = buf_log;
  new_buf_flag = 1;
  new_buf = buf;
else
  buf(buf_log + 1) = input;
  new_buf_log = buf_log + 1;
  new_buf = buf;
  if new_buf_log == buf_size
   new_buf_flag = 1;
  else
   new_buf_flag = 0;
  end
end
```

A2.5. Αρχείο cap_alloc.m

```
function [b2_cap, b2_flag] = cap_alloc(buf_cap, b1_log, b2_log,
subm_flag)

% Function implementing dynamic capacity allocation between incoming
% and outgoing queues of a bidirectional buffer
%
% The function returns the new capacity allocated to queue b2
together
% with its new availability status, considering present usage of
buffer
% capacity from queue b1
%
% This function is designed to work together with the airport models
% airport2a.m and airport2b.m

% subm: the submachine controlling push operation of b1 queue
% buf_cap: total buffer capacity

b2_cap = buf_cap-b1_log;
if subm_flag   % when submachine is in processing state, reserve one
more position for b1 queue
  b2_cap = b2_cap-1;
end
if b2_log == b2_cap
  b2_flag = 1;
else
  b2_flag = 0;
End
```

A2.6. Αρχείο end_proc.m

```
function [subm, subm_flag, subm_out, end_time, obuf, obuf_log,
obuf_flag] = end_proc(subm_out, obuf, obuf_log, obuf_cap, t)

% Function implementing the change of submachine state from
processing to idle
% (push submachine output to output buffer)
%
% The function returns new submachine state and new input buffer
state
%
% This function is designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m and
% assigns a call to function push_buf (push buffer)

[obuf_log, obuf_flag, obuf] = push_buf(subm_out, obuf_log, obuf,
obuf_cap);
subm_out = 0;
subm = 0;
subm_flag = 0;
end_time = t;
```

A2.7. Αρχείο find_hub2.m

```
function index = find_hub2(out5, out6)

% Function implementing the search for the airplane with the oldest
% (smallest) system entry time among the ones that have completed
% service in the hubs (submachines 5-6)
%
% The function returns the index of the submachine to provide input
% to buffer 5. If no submachine is ready, the function returns zero
%
% This function is designed to work together with the airport models
% airport2a.m, airport2b.m

buf5_in = 0;
index = 0;
if out5  % submachine 5 ready
  buf5_in = out5;
  index = 5;
end

if out6  % submachine 6 ready
  if ~index  % submachine 5 not ready
    index = 6;
    buf5_in = out6;
  else  % submachine 5 also ready
    if out6 < buf5_in   % output of submachine 6 older than output of
submachine 5
      index = 6;
      buf5_in = out6;
    end
  end
end
```

A2.8. Αρχείο start_proc.m

```
function [subm, subm_flag, start_time, ibuf, ibuf_log, ibuf_flag] =
start_proc(ibuf, ibuf_log, t)

% Function implementing the change of submachine state to processing
% (pop input buffer to submachine, which changes state to processing)
%
% The function returns new submachine state and new input buffer
state
%
% This function is designed to work together with the airport models
% airport1a.m, airport1b.m, airport2a.m, airport2b.m and
% assigns a call to function pop_buf (pop buffer)

[ibuf_log, ibuf_flag, ibuf, ibuf_out] = pop_buf(ibuf_log, ibuf);
subm = ibuf_out;
start_time = t;
subm_flag = 1;
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Αμπακούμκιν, Κ. Γ., Καθηγητής ΕΜΠ, *Αεροδρόμια*, Εκδ. Συμμετρία, Αθήνα, 1990.
2. Διεθνής Αερολιμένας Αθηνών, *http://www.aia.gr/*, επίσημη ιστοσελίδα.
3. Varvarigos, M., Voulgaris, P., Christodoulou, M. and Feron, E., *Advance information technology tools for the air traffic management systems of the future*.
4. Whitelegg, J. and Williams, N., *The Plane Truth: Aviation and the Environment*, Transport 2000 and The Ashden Trust.
5. Paulson, G., *Integrating environmental issues into Air Traffic Management (ATM) operations*, Colloquium on Environmental Aspects of Aviation, Montreal, 9-11 May 2001.
6. Pujet, N., Delcairey, B. and Feronz, E., *Input-output modeling and control of the departure process of congested airports*, American Institute of Aeronautics and Astronautics, 99-4299.
7. Andersson, K., Carr, F., Feron, E. and Hall, D.W., *Analysis and modeling of ground operations at hub airports*, 3rd USA/Europe Air Traffic Management R&D Seminar, Napoli, 13-16 June 2000.
8. Miedema H. M. and Vos, H., *Exposure-response relationships for transportation noise*, J Acoust Soc Am 1998; 104(6): 3432-45.
9. Ευρωπαϊκή Επιτροπή, *Αεροπορικές μεταφορές και περιβάλλον - Η πορεία που οδηγεί στην επίλυση των ζεόντων προβλημάτων της αειφόρου ανάπτυξης*, COM/99/0640.
10. Penner, E.J., Lister, H. D., Griggs, J. D., Dokken, J. D. and McFarland, M., *Aviation and the Global Atmosphere*, IPCC Special Report, Geneva, 1999.
11. Stylianos E. Perrakis, Neural Network Schedulers in Manfacturing Systems, Grecce 2001
12. Pooch, W.U. and Wall, A.J., *Discrete event simulation*, CPV Press, Inc., 1993.
13. Fishman, G.S., *Concepts and methods in discrete event simulation*, Wiley, Inc., 1994.
14. Cassandras, C.G., *Discrete event systems (modeling and performance analysis)*, Richard D. Irwin & Aksen Associates, Inc., 1993.
15. Khoshnevis, B., *Discrete systems simulation*, McGraw-Hill, Inc., 1994.

16. Ρουμελιώτης, Μ., *Τεχνικές προσομοίωσης*, Παρατηρητής, Θεσσαλονίκη, 1998.