ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΝΑΥΠΗΓΩΝ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΝΑΥΤΙΚΗΣ ΜΗΧΑΝΟΛΟΓΙΑΣ

"ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΕΝΕΡΓΕΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΧΡΗΣΗ MATLAB, SIMULINK"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΑΝΑΓΙΩΤΗ ΑΓΡΑΠΙΔΗ
ΝΑΥΠΗΓΟΥ ΜΗΧΑΝΟΛΟΓΟΥ ΜΗΧΑΝΙΚΟΥ Ε.Μ.Π

ΕΠΙΒΛΕΨΗ
ΚΑΘ.ΕΜΠ: ΧΡΙΣΤΟΣ ΦΡΑΓΚΟΠΟΥΛΟΣ

ΑΘΗΝΑ
ΜΑΪΟΣ 2006

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Αφιερώνεται

στις οικογένειές μου

Ο γράφων παραμένει στη διάθεση του αναγνώστη

για τυχόν διευκρινίσεις, διορθώσεις, προτάσεις και σχόλια :

agrapas@postmaster.co.uk

1. ΕΙΣΑΓΩΓΗ

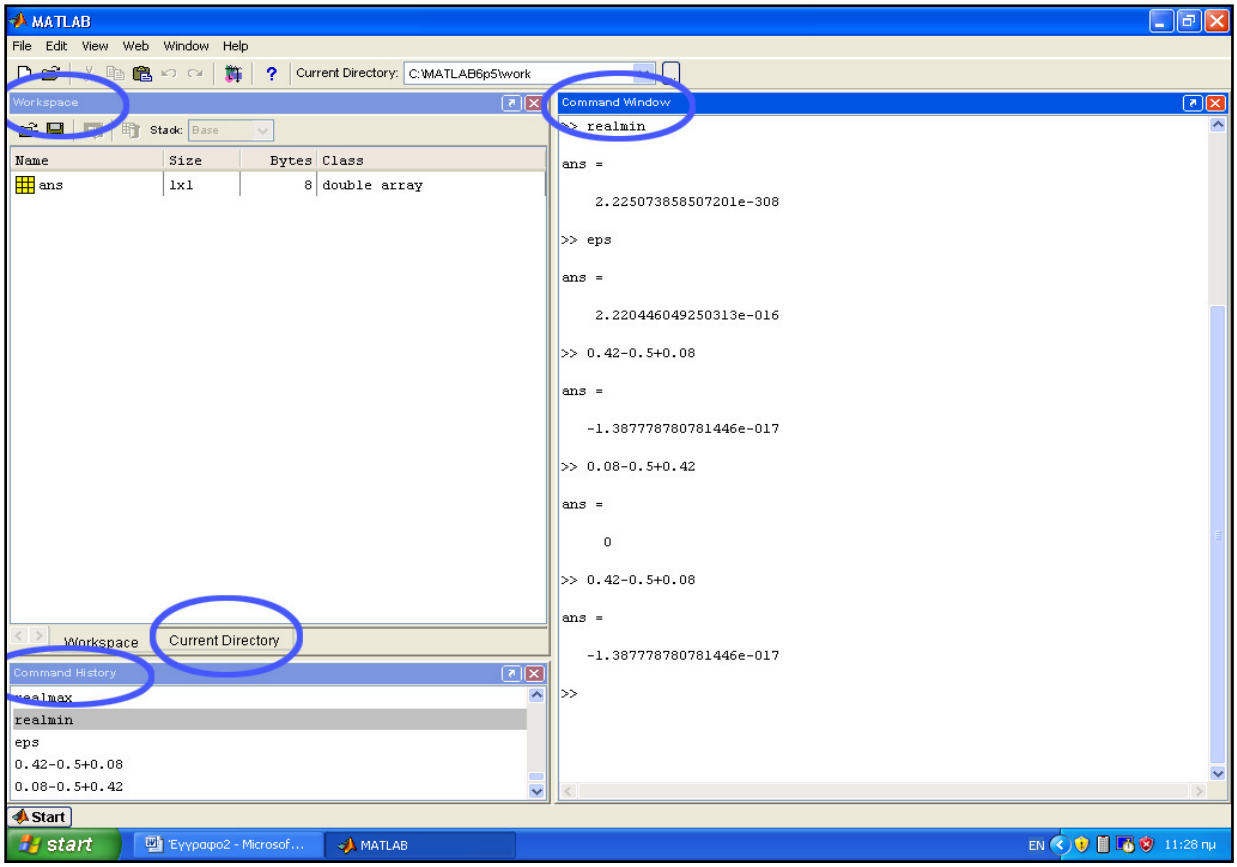
Σκοπός της παρούσας διπλωματικής εργασίας, είναι η διερεύνηση της δυνατότητας βελτιστοποίησης ενεργειακών συστημάτων με χρήση του Matlab.

Στο πρώτο μέρος της εργασίας, (Κεφάλαια 2-14), διερευνώνται τα γενικά στοιχεία γύρω από το Matlab, καθώς και κάποια πιο εξειδικευμένα που αφορούν τη διαδικασία βελτιστοποίησης ενός ενεργειακού συστήματος.
Στο Κεφάλαιο 16, προχωρούμε στην εφαρμογή όλων αυτών, βελτιστοποιώντας ένα ενεργειακό σύστημα συμπαραγωγής που περιγράφεται στο Παράρτημα Α'.
Στη συνέχεια του ίδιου κεφαλαίου πραγματοποιείται η ανάλυση ευαισθησίας του συστήματος, ως προς την επίδραση της τιμής του καυσίμου τόσο για απλό όσο και για διπλό κόστος κεφαλαίου.

Στο δεύτερο μέρος της εργασίας, διερευνάται η δυνατότητα επικοινωνίας μεταξύ της γλώσσας fortran και του Matlab. Συγκεκριμένα, επιδιώκεται, η κλήση κάποιας υπορουτίνας fortran (που περιέχει κάποια αντικειμενική συνάρτηση) από το Matlab, και η προσπάθεια ελαχιστοποίησής της μέσω των μεθόδων του προγράμματος.
Αφού αναλυθούν κάποια βασικά στοιχεία στο Κεφάλαιο 15, στο Κεφάλαιο 17, εφαρμόζονται τα παραπάνω στο σύστημα του απλού αεριοστρόβιλου, που περιγράφεται στο Παράρτημα Β'. Η αντικειμενική συνάρτηση και όλες οι απαραίτητες σχέσεις, προκύπτουν από το αρχείο GTG92.for, το οποίο επιλύει το προαναφερθέν σύστημα με τη μέθοδο της χρυσής τομής.

2. ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΕΡΓΑΣΙΑΣ ΣΤΟ MATLAB

Το Matlab έχει ένα πολύ φιλικό προς τον χρήστη περιβάλλον. Χρησιμοποιεί παράθυρα (windows) για την εύκολη πρόσβαση σε όλες τις λειτουργίες. Όταν εκκινήσει το πρόγραμμα, θα εμφανιστεί η παρακάτω οθόνη (Εικόνα 1.1). Όλα τα παράθυρα περιγράφονται συνοπτικά, μαζί με τις λειτουργίες τους στον Πίνακα 2.1

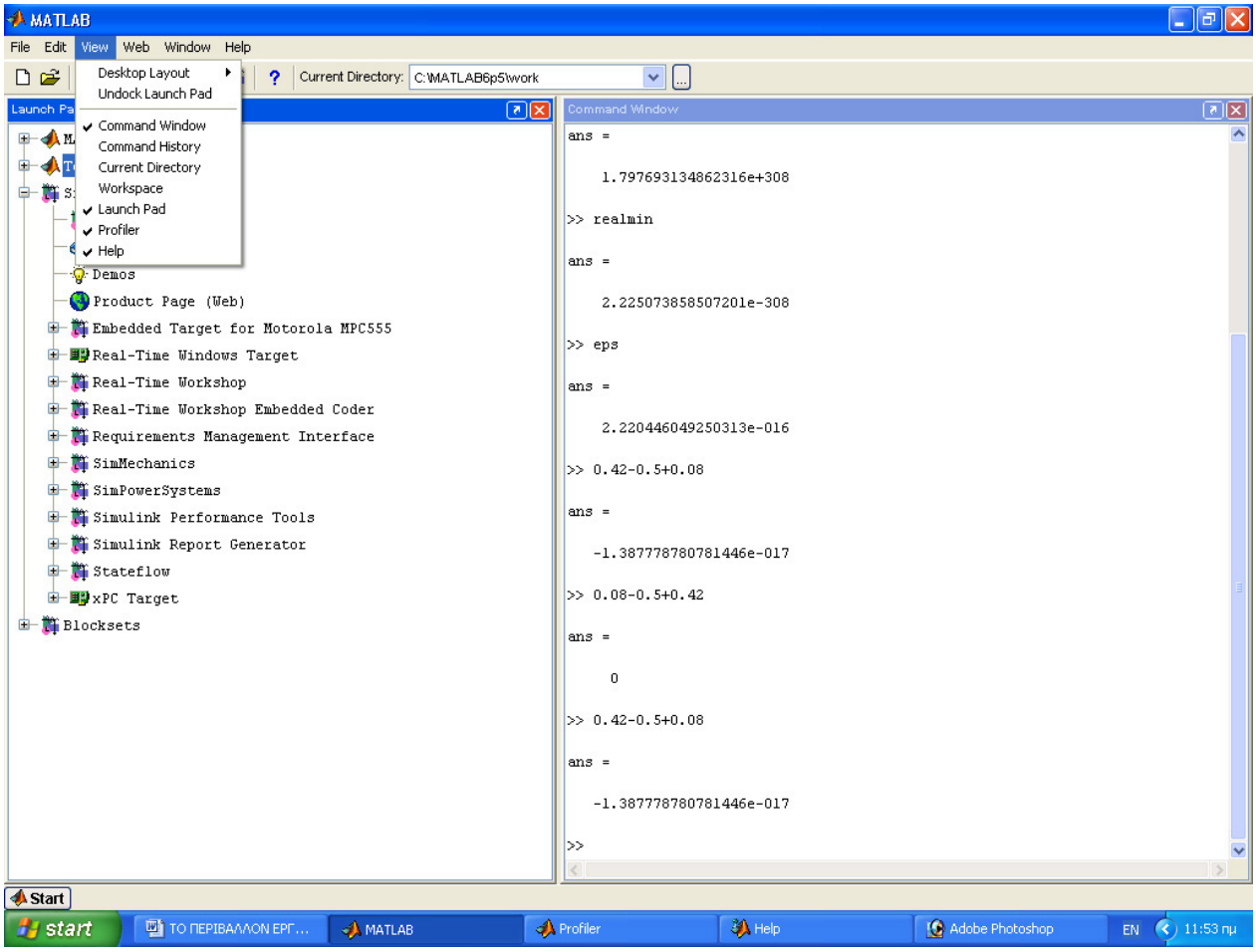


Εικόνα 2.1 Η οθόνη εργασίας του matlab 6.5

Πίνακας 2.1 Συνοπτική περιγραφή των παραθύρων στην οθόνη εκκίνησης

Παράθυρο	**Περιγραφή**
«command window»	Είναι το βασικό παράθυρο στο οποίο αναγράφονται οι δηλώσεις μεταβλητών και οι εντολές (Στα αγγλικά ονομάζεται «command prompt»). Είναι το κύριο περιβάλλον εργασίας. Το σημείο στο οποίο το πρόγραμμα αναμένει εντολές έχει το σύμβολο (>>).
Workspace	Ο χώρος στον οποίο αποθηκεύονται οι μεταβλητές που δηλώνονται στο «command prompt». Όταν για παράδειγμα πληκτρολογηθεί στο «command prompt»: **>>x=5** τότε το πρόγραμμα, αποθηκεύει την τιμή του x στο workspace. (Όπως μπορούμε να δούμε όλες οι μεταβλητές στο matlab αποθηκεύονται σε πινακοποιημένη μορφή. Για παράδειγμα, μία σταθερά αποθηκεύεται σαν ένας πίνακας 1X1).
Current directory	Παράθυρο, από το οποίο παρέχεται εύκολη πρόσβαση σε φακέλους με αρχεία εργασίας. Επίσης, χρησιμεύει κατά την εκτέλεση αρχείων. Όταν για παράδειγμα θέλουμε να εκτελέσουμε το αρχείο newt.m τότε πρέπει το current directory να είναι ο φάκελος του δίσκου στον οποίο είναι αποθηκευμένο, αλλιώς το matlab δεν θα μπορεί να το εντοπίσει για να το εκτελέσει.
Command history	Το matlab κρατά ιστορικό για τις εντολές που έχουν πληκτρολογηθεί προηγουμένως. Έτσι κάνοντας διπλό κλικ σε κάποια από αυτές μπορεί κάποιος να την ανακτήσει στο «command prompt» χωρίς να χρειαστεί να την πληκτρολογήσει πάλι.

Τα υπόλοιπα παράθυρα (Μπορούν να εμφανιστούν πατώντας το ιπτάμενο μενού **view**), περιγράφονται στον Πίνακα 2.2



Εικόνα 2.2 Υπόλοιπα παράθυρα εργασίας

Πίνακας 2.2 Περιγραφή υπολοίπων παραθύρων εργασίας

Παράθυρο	**Περιγραφή**
Launch Pad	Δενδροειδές διάγραμμα για πρόσβαση στις διάφορες επιπρόσθετες βιβλιοθήκες με ειδικούς αλγόριθμους, τα λεγόμενα toolbox, (Όπως τα Optimization toolbox, Fuzzy Logic toolbox, Genetic Algorithms toolbox κ.α), σε έτοιμα προγράμματα αυτών (demos), καθώς και σε αρχεία βοήθειας.
Profiler	Ο χρήστης πληκτρολογεί μια εντολή και το πρόγραμμα μετρά τον χρόνο που χρειάζεται να την υπολογίσει. Αυτό είναι χρήσιμο, διότι μια εντολή μπορεί να γραφεί με διαφορετικούς τρόπους ώστε να εκτελείται σε λιγότερο χρόνο. Με άλλα λόγια μπορεί κάποιος να βελτιστοποιήσει τον κώδικά του, στα σημεία εκείνα που δαπανάται ο μεγαλύτερος χρόνος κατά την εκτέλεση.
Help	Παρέχει πρόσβαση σε αρχεία βοήθειας.

3. ΑΡΙΘΜΗΤΙΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ, ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΣΦΑΛΜΑΤΑ ΣΤΟ MATLAB 6.5

3.1 Ελάχιστος και μέγιστος αριθμός που μπορεί να αναπαρασταθεί.

Όλοι οι αριθμοί στο matlab αναπαρίστανται με διπλή ακρίβεια (double precision). Υπάρχουν όμως κάποια όρια στην αναπαράσταση των αριθμών.
Ο μεγαλύτερος θετικός πραγματικός αριθμός που μπορεί να αναπαρασταθεί δίνεται αν πληκτρολογηθεί η εντολή **realmax** στο ««command prompt»».

Σημείωση: Τα αποτελέσματα των εντολών που πληκτρολογούνται αποθηκεύονται στην εσωτερική μεταβλητή ANS του matlab. Όταν πληκτρολογηθεί και δεύτερη συνεχόμενη εντολή, τότε το περιεχόμενο της ANS αντικαθίσταται με το νέο αποτέλεσμα.

>> realmax

ans =

1.797693134862316e+308

Ο μικρότερος θετικός πραγματικός αριθμός που μπορεί να αναπαρασταθεί δίνεται αν πληκτρολογήσουμε την εντολή realmin στο «command prompt».

>> realmin

ans =

2.225073858507201e-308

Ο μικρότερος αριθμός που μπορεί να προστεθεί στο 1 και να παράγει αριθμό μεγαλύτερο του 1 δίνεται αν πληκτρολογήσουμε την εντολή eps στο «command prompt».

>> eps

ans =

2.220446049250313e-016

3.2 Αριθμητικά σφάλματα

Δυστυχώς, κάποιοι αριθμοί στο matlab, όπως και σε άλλες γλώσσες προγραμματισμού, δεν έχουν ακριβή αναπαράσταση και γι' αυτό το λόγο απλώς προσεγγίζονται με όσο γίνεται μεγαλύτερη ακρίβεια. Συνεπώς, υπάρχει η περίπτωση να γίνονται κάποια λάθη στους υπολογισμούς. Το παρακάτω παράδειγμα είναι ενδεικτικό:

```
>> 0.42-0.5+0.08

ans =

  -1.387778780781446e-017
```

Παρατηρούμε ότι παίρνουμε κάποιο αποτέλεσμα πολύ κοντά στο μηδέν αλλά όχι μηδέν όπως θα έπρεπε. Ας γράψουμε την ίδια πράξη αλλιώς:

```
>> 0.08-0.5+0.42

ans =
   0
```

Όντως τώρα το αποτέλεσμα είναι ακριβώς μηδέν.

Άλλα επίσης λάθη στους αριθμητικούς υπολογισμούς, που είναι συνήθη στην αριθμητική ανάλυση και δεν αφήνουν το matlab αδιάφορο, είναι τα εξής:

- **Round-off error**, δηλαδή σφάλμα λόγω στρογγυλοποίησης. Τα λάθη λόγω στρογγυλοποίησης είναι λιγότερο πιθανό να συγκεντρωθούν σε επαναληπτικούς υπολογισμούς, αφού η πραγματική τιμή είναι μεγαλύτερη από την στρογγυλοποιημένη για το μισό του χρόνου εκτέλεσης και μικρότερη για τον άλλο μισό χρόνο της εκτέλεσης. Επίσης το λάθος που μπορεί να προκύψει από κόψιμο των δεκαδικών ψηφίων (chopping) είναι περίπου διπλάσιο από αυτό που προκύπτει λόγω στρογγυλοποίησης [1, σελ.16].

- **Cancellation error**, το σφάλμα δηλαδή που προκύπτει όταν έχουμε να αφαιρέσουμε δύο σχεδόν ίσους αριθμούς.

- **Truncation error**, δηλαδή σφάλμα που προκύπτει όταν χρησιμοποιούμε προσεγγιστικούς αλγορίθμους, όπως για παράδειγμα ο τραπεζοειδής κανόνας στην αριθμητική ολοκλήρωση.

3.3 Εμφάνιση αριθμών και αποτελεσμάτων

Με τις παρακάτω εντολές του πίνακα 3.3.1 μπορεί να καθοριστεί η εμφάνιση των αριθμών στην οθόνη.

Σημείωση: Οι εσωτερικές διαδικασίες αναπαράστασης (διπλή ακρίβεια) δεν αλλάζουν. Μόνο η αναπαράσταση στην οθόνη αλλάζει.

Πίνακας 3.3.1 Αναπαράσταση αριθμών και αποτελεσμάτων

Εντολή matlab	**Αναπαράσταση του αριθμού π**	**Σχόλια**
Format short	3.1416	5 δεκαδικά ψηφία (μαζί με την υποδιαστολή)
Format long	3.14159265358979	15 δεκαδικά ψηφία (μαζί με την υποδιαστολή)
Format short e	3.1416e+000	5 δεκαδικά ψηφία (μαζί με την υποδιαστολή) συν το εκθετικό
Format long e	3.14159265358979e+000	15 δεκαδικά ψηφία (μαζί με την υποδιαστολή) συν το εκθετικό
Format hex	400921fb54442d18	Δεκαεξαδικό σύστημα
Format bank	3.14	2 δεκαδικά ψηφία

Αυτές είναι οι πιο βασικές. Για άλλες αναπαραστάσεις, ανατρέξτε στο εγχειρίδιο του matlab, έκδοση 6.5.

3.4 Βασικές πράξεις

Στις απλές πράξεις γενικά, δεν υπάρχει πρόβλημα με την ύπαρξη κενών διαστημάτων. Δηλαδή οι υπολογισμοί δεν επηρεάζονται αν υπάρχει κενό διάστημα μεταξύ του αριθμού και του συμβόλου. Στο matlab η σειρά υπολογισμών πραγματοποιείται με βάση την **προτεραιότητα των πράξεων** που ξέρουμε από τα μαθηματικά. Προφανώς οι εκφράσεις που βρίσκονται μέσα σε παρενθέσεις υπολογίζονται πρώτες. Το matlab προσφέρει τις εξής αριθμητικές πράξεις:

Πίνακας 3.4.1 Απλές πράξεις με το matlab

Λειτουργία	**Σύμβολο**	**Παράδειγμα**
Πρόσθεση	+	3+22
Αφαίρεση	-	54.4-16.5
Πολλ/σμός	*	3.14*6
Διαίρεση	/ ή \	19.54/7 ή 7\19.54
Ύψωση σε δύναμη	^	2^5

Σημείωση: Το matlab δεν παρουσιάζει ιδιαιτερότητες όπως άλλες γλώσσες προγραμματισμού, σαν τη fortran. Για παράδειγμα στη fortran η διαίρεση δύο ακεραίων μπορεί να είναι προβληματική. Η πράξη 5/2, θα μας δώσει μόνο το ακέραιο μέρος της διαίρεσης. Στο matlab όμως η πράξη 5/2 δίνει αποτέλεσμα 2.5.

3.5 Μεταβλητές

Για τις μεταβλητές ισχύουν οι εξής κανόνες για την ονομασία τους:

- Μία μεταβλητή με κεφαλαία γράμματα αποτελεί διαφορετική μεταβλητή από μία με μικρά γράμματα. Για παράδειγμα οι μεταβλητές Cost, cost, CoSt, COST είναι όλες διαφορετικές μεταξύ τους.
- Τα ονόματα των μεταβλητών μπορούν να έχουν μέχρι 31 χαρακτήρες. Οποιοσδήποτε χαρακτήρας μετά τον 31°, αγνοείται.
- Τα ονόματα των μεταβλητών, πρέπει να ξεκινούν με κάποιο γράμμα και να ακολουθούνται από άλλα γράμματα, αριθμούς, και χαρακτήρες υπογράμμισης (πχ _). Οι χαρακτήρες στίξης δεν επιτρέπονται, διότι πολλοί από αυτούς έχουν ιδιαίτερη σημασία στο matlab.

Κάποια ονόματα δεν μπορούν να χρησιμοποιηθούν ως μεταβλητές. Αυτές είναι οι παρακάτω:

Πίνακας 3.5.1 Λέξεις των οποίων το όνομα δεν μπορούν να πάρουν οι μεταβλητές

for end if while function return elseif case otherwise switch continue else try catch global persistent break

Σημείωση: Γενικά, η χρήση των παραπάνω ονομάτων, προκαλεί την εμφάνιση κάποιου μηνύματος λάθους. Παρόλα αυτά μπορούν να χρησιμοποιηθούν, αν για παράδειγμα κάνουμε κεφαλαία, ένα ή περισσότερα γράμματα.

Επιπρόσθετα υπάρχουν και κάποιες ειδικές μεταβλητές όπως οι παρακάτω:

Πίνακας 3.5.2 Κατάλογος ειδικών μεταβλητών

Ειδικές μεταβλητές	**Περιγραφή**
ans	Η μεταβλητή στην οποία κάθε φορά αποθηκεύονται τα αποτελέσματα των πράξεων.
beep	Κάνει τον υπολογιστή να ηχήσει σήμα μπίπ.
pi	Είναι το γνωστό μας π.
eps	Ο ελάχιστος αριθμός, που αν προστεθεί στο 1 θα μας δώσει αριθμό μεγαλύτερο του 1.
inf	Αναπαριστά το άπειρο. Για παράδειγμα, η πράξη 1/0 προκαλεί αυτό το αποτέλεσμα.
Nan ή nan	Σημαίνει not-a-number. Εμφανίζεται στις πράξεις 0/0, ∞/∞ κλπ.
i ή j	Μιγαδικά σύμβολα. Για παράδειγμα η πράξη sqrt(-2) θα δώσει το αποτέλεσμα: 0 + 1.4142i
nargin	Ο αριθμός των ορισμάτων εισόδου, μιας συνάρτησης
nargout	Ο αριθμός των ορισμάτων εξόδου, μιας συνάρτησης
realmin	Ο μικρότερος θετικός πραγματικός αριθμός.
realmax	Ο μεγαλύτερος θετικός πραγματικός αριθμός.
bitmax	Ο μεγαλύτερος θετικός ακέραιος που μπορεί να χρησιμοποιηθεί.
varargin	Αριθμός των μεταβλητών των ορισμάτων εισόδου μιας συνάρτησης
varargout	Αριθμός των μεταβλητών των ορισμάτων εξόδου μιας συνάρτησης

4. ΕΙΣΑΓΩΓΗ ΣΤΗ ΔΗΜΙΟΥΡΓΙΑ ΑΡΙΘΜΗΤΙΚΩΝ ΠΙΝΑΚΩΝ

4.1 Δήλωση διανυσμάτων

- Η δήλωση ενός διανύσματος γίνεται με το κλείσιμο των στοιχείων του διανύσματος σε αγκύλες τα οποία χωρίζονται μεταξύ τους είτε με κόμμα (,) είτε με κενό διάστημα. Δηλαδή ως εξής:
  A=[1,2,3,4,5,6] ή A=[1 2 3 4 5 6].

- Μπορούμε να έχουμε πρόσβαση στα στοιχεία του διανύσματός μας, όταν θέλουμε για παράδειγμα να κάνουμε πράξεις μόνο με κάποια από τα στοιχεία του διανύσματος ή να αντικαταστήσουμε κάποιο στοιχείο που άλλαξε η τιμή του, χωρίς να χρειαστεί να δηλώσουμε το διάνυσμα πάλι. Αν θέλουμε για παράδειγμα το matlab να μας εμφανίσει το 4ο στοιχείο του διανύσματος Α πληκτρολογούμε:

  >>A(4)
  ans=
  4

- Αν τώρα, θέλουμε να αντικαταστήσουμε το τέταρτο στοιχείο του διανύσματος (το 4) και να του δώσουμε την τιμή 9, μπορούμε να γράψουμε:

  >> A(4)=9

  A =

  1 2 3 9 5 6

Η διαδικασία αυτή παρουσιάζεται στη βιβλιογραφία με την ονομασία **array indexing**.

4.2 Δήλωση πινάκων

- Η δήλωση ενός πίνακα πραγματοποιείται με το κλείσιμο των στοιχείων του πίνακα σε αγκύλες, τα οποία χωρίζονται μεταξύ τους είτε με κόμμα (,) είτε με κενό διάστημα. Το matlab αντιλαμβάνεται την αλλαγή των γραμμών όταν συναντήσει την αγγλική άνω τελεία (;). Για παράδειγμα, η κατασκευή ενός 3Χ3 πίνακα μπορεί να γίνει ως εξής:
  A=[1,2,3 ; 4,5,6 ; 7,8,9] ή **A=[1 2 3 ; 4 5 6 ; 7 8 9]**.

- Μπορούμε να έχουμε πρόσβαση στα στοιχεία του πίνακά, όταν θέλουμε για παράδειγμα να κάνουμε πράξεις μόνο με κάποια από τα στοιχεία του πίνακα ή να

αντικαταστήσουμε κάποιο στοιχείο που άλλαξε χωρίς να χρειαστεί να ξαναδηλώσουμε τον πίνακα. Αν θέλουμε το matlab να μας εμφανίσει το στοιχείο της 2ης γραμμής και της 3ης στήλης πληκτρολογούμε:

>>A(2,3)
ans=
6

- Αν τώρα, θέλουμε να αντικαταστήσουμε το στοιχείο της 2ης γραμμής και της 3ης στήλης (δηλαδή το 6) και να του δώσουμε την τιμή 20, μπορούμε να γράψουμε:

>> A(2,3)=20

A =

1 2 3
4 5 20
7 8 9

οπότε ο πίνακας γίνεται ο A=[1,2,3 ; 4,5,20 ; 7,8,9]

- Υπάρχει η δυνατότητα να αντικατασταθεί ολόκληρη γραμμή ή στήλη του πίνακα A. Αν για παράδειγμα γράψουμε:

>> A(2, :)=[12 13 14]

A =

1 2 3
12 13 14
7 8 9

Τότε αντικαθίσταται όλη η 2η γραμμή του πίνακα A με την [12 13 14]

4.3 Τρόποι κατασκευής πινάκων

Υπάρχουν πάρα πολλοί τρόποι κατασκευής πινάκων, μεταξύ των οποίων διακρίνουμε τους εξής:

- Αν πληκτρολογήσουμε **a=1:5**, θα πάρουμε έναν πίνακα που θα ξεκινά από το 1 και θα καταλήγει στο 5 με βήμα 1, δηλαδή τον

a=
1 2 3 4 5

- Αν πληκτρολογήσουμε **a=1:2:9**, θα πάρουμε έναν πίνακα που θα ξεκινά από το 1 και θα καταλήγει στο 9 με βήμα 2, δηλαδή τον:

```
a=
  1 3 5 7 9
```

- Αν πληκτρολογήσουμε **linspace(0,2,12)** θα πάρουμε ένα πίνακα με 12 σημεία μεταξύ του 0 και του 2 τα οποία απέχουν «γραμμικά» μεταξύ τους. Στη συγκεκριμένη περίπτωση, ο πίνακας που δίνει το matlab είναι ο:

```
ans =

 Columns 1 through 6
      0   0.1818   0.3636   0.5455   0.7273   0.9091

 Columns 7 through 12
  1.0909   1.2727   1.4545   1.6364   1.8182   2.0000
```

- Αν πληκτρολογήσουμε **logspace(0,2,12)** θα πάρουμε ένα πίνακα με 12 σημεία μεταξύ του 10^0 και του 10^2 τα οποία απέχουν «λογαριθμικά» μεταξύ τους. Στη συγκεκριμένη περίπτωση, ο πίνακας που δίνει το matlab είναι ο:

```
ans =

 Columns 1 through 6
   1.0000   1.5199   2.3101   3.5112   5.3367   8.1113

 Columns 7 through 12
  12.3285  18.7382  28.4804  43.2876  65.7933 100.0000
```

- Έστω οι πίνακες **a=[1 2;3 4]** και **b=[5 6;7 8]**. Μπορεί να κατασκευαστεί πίνακας από υποπίνακες, αν για παράδειγμα πληκτρολογήσουμε: **c=[a b]**. Τότε, θα πάρουμε τον εξής πίνακα:

```
c =

   1   2   5   6
   3   4   7   8
```

Δηλαδή ο πίνακας c κατασκευάζεται από τον a ο οποίος αποτελεί την $1^η$ στήλη του c και τον b ο οποίος αποτελεί την $2^η$ στήλη.

Αν όμως πληκτρολογήσουμε: c=[a;b], θα πάρουμε τον εξής πίνακα:

```
c =
    1    2
    3    4
    5    6
    7    8
```

Δηλαδή ο πίνακας c κατασκευάζεται από τον a ο οποίος αποτελεί την 1η γραμμή του c και τον b ο οποίος αποτελεί την 2η γραμμή.

4.4 Πράξεις πινάκων στοιχείο-στοιχείο

Πίνακας 4.4.1 Πράξεις πινάκων στοιχείο-στοιχείο

Πράξη στοιχείο-στοιχείο	**Δεδομένα** **A=[a1 a2…an]** **B=[b1 b2…bn], c=σταθερά**	**Παρατηρήσεις**
Πρόσθεση με σταθερά	A+c=[a1+c, a2+c…an+c]	
Αφαίρεση με σταθερά	A-c=[a1-c, a2-c…an-c]	
Πολλ/σμός με σταθερά	A*c=[a1*c, a2*c…an*c]	
Διαίρεση με σταθερά	A/c ή c/A=[a1/c, a2/c…an/c]	
Πρόσθεση πινάκων	A+B=[a1+b1, a2+b2…an+bn]	Οι πίνακες A και B πρέπει να έχουν τις **ίδιες διαστάσεις**
Πολλ/σμος πινάκων (στοιχείο-στοιχείο)	A .*B=[a1*b1, a2*b2…an*bn]	Οι πίνακες A και B πρέπει να έχουν τις **ίδιες διαστάσεις**. Αν παραλείπαμε την τελεία, τότε θα είχαμε κανονικό πολλ/σμό πινάκων (θα έπρεπε δηλαδή ο αριθμός των στηλών του 1ου πίνακα να είναι ίσος με τον αριθμό γραμμών του 2ου πίνακα)
Διαίρεση στα δεξιά πινάκων (στοιχείο-στοιχείο)	A . /B=[a1/b1, a2/b2…an/bn]	Οι πίνακες A και B πρέπει να έχουν τις **ίδιες διαστάσεις**. Αν παραλείπαμε την τελεία, τότε το matlab θα πραγματοποιούσε την επίλυση του συστήματος Ax=B, με τη μέθοδο Gauss (θα είχαμε δηλαδή υπολογισμό του αντιστρόφου του πίνακα A).
Διαίρεση στα αριστερά πινάκων (στοιχείο-στοιχείο)	B . \A=[a1/b1, a2/b2…an/bn]	Ισχύουν τα ίδια, όπως στη διαίρεση στα δεξιά.
Ύψωση σε δύναμη	A .^c=[a1^c, a2^c…an^c] c .^A= [c^a1, c^a2…c^an] A .^B=[a1^b1, a2^b2…an^bn]	

4.5 Βασικοί πίνακες

- **ones(3)** → 3Χ3 πίνακας με όλα τα στοιχεία του μονάδα.
- **ones(3,4)** → 3Χ4 πίνακας με όλα τα στοιχεία μονάδα.
- **zeros(2,5)** → 2Χ5 πίνακας με όλα τα στοιχεία μηδέν.
- **eye(4)** → 4Χ4 μοναδιαίος πίνακας.
- **eye(2,4)** → 2Χ4 μοναδιαίος πίνακας, δηλαδή ο 1 0 0 0
  0 1 0 0

- **rand(3)** → 3Χ3 πίνακας με τυχαία στοιχεία τα οποία έχουν τιμές από 0 έως 1 για παράδειγμα ο πίνακας:

  0,9501 0,4860 0,4565
  0,2311 0,8913 0,0185
  0,6068 0,7621 0,8214

5. ΚΑΤΑΣΚΕΥΗ ΠΙΝΑΚΩΝ ΚΕΛΙΩΝ (Cell-arrays)

Οι πίνακες κελιών είναι πίνακες του matlab των οποίων τα στοιχεία είναι κελιά. Κάθε κελί μπορεί να περιέχει οποιοδήποτε τύπο δεδομένων του matlab όπως: αριθμητικούς πίνακες, μιγαδικούς αριθμούς, ακολουθίες χαρακτήρων (strings), συμβολικούς χαρακτήρες, άλλους πίνακες κελιών, structures (βλέπε παρακάτω) κα.
Η χρησιμότητα των πινάκων κελιών, έγκειται στο ότι επιτρέπουν σε κάποιον να ομαδοποιήσει διαφορετικούς τύπους στοιχείων σε **μία και μοναδική μεταβλητή**. Αυτή η δυνατότητα, του να μπορεί κάποιος να χειριστεί ανομοιογενή δεδομένα σε μία και μόνο μεταβλητή, είναι πολύ χρήσιμη, τόσο από οργανωτική άποψη όσο και από απλότητα υπολογισμών.

Σημείωση 1: Στο matlab, όταν ο χρήστης θέλει να προσθέσει κάποιο σχόλιο στο πρόγραμμά του, τότε θα πρέπει να εισάγει τον χαρακτήρα (%). Το κείμενο που ακολουθεί αυτόν τον χαρακτήρα, εκλαμβάνεται ως σχόλιο.

Σημείωση 2: Πολλές φορές μας ενδιαφέρει σε ένα πρόγραμμα, να μην εμφανίζονται οι ενδιάμεσες τιμές μεταβλητών κατά τη διάρκεια των υπολογισμών. Όταν μετά την πληκτρολόγηση της εντολής εμφανίζεται ο χαρακτήρας αγγλική άνω τελεία (;), τότε το matlab δεν εμφανίζει το αποτέλεσμα της εντολής. Στην πρώτη περίπτωση του παρακάτω παραδείγματος, η απουσία του (;) προκαλεί την εμφάνιση των ενδιάμεσων αποτελεσμάτων. Αντίθετα, στη δεύτερη περίπτωση, φαίνεται μόνο το τελικό αποτέλεσμα.

1η περίπτωση	2η περίπτωση
>> a=2 a = 2 >> b=4 b = 4 >> c=a+b c = 6	>> a=2; >> b=4; >> c=a+b c = 6

5.1 Τρόποι κατασκευής πινάκων κελιών.

- Ο **πρώτος τρόπος** είναι -στη δήλωση των κελιών- τα δεδομένα να περιέχονται μέσα σε άγκιστρα { } στο δεξί μέρος. Για παράδειγμα:
  >>A(1,1)={ [1 2 3;4 5 6;7 8 9] }; %πίνακας
  >>A(1,2)={ 2+3i }; %μιγαδικός
  >>A(2,1)={ 'Ακολουθία χαρακτήρων' }; %ακολουθία χαρακτήρων
  >>A(2,2)={ 0:2:10 }; %διάνυσμα με στοιχεία από 0 έως 10 με βήμα 2

Αν τώρα πληκτρολογήσουμε:

>>A

Θα πάρουμε την ακόλουθη απάντηση:

```
A=
    [3X3 double]          [2.0000+3.0000i ]
    [1X20 char]               [1X6 double]
```

Παρατήρηση 1: Με τις παραπάνω εντολές, κατασκευάζουμε τον πίνακα κελιών, με την δήλωση καθενός κελιού ξεχωριστά. Έτσι, για παράδειγμα, η εντολή:
A(1,2)={ 2+3i }; σημαίνει, ότι το κελί που βρίσκεται στην 1η γραμμή και στην 2η στήλη του πίνακα κελιών, θα αποτελείται από ένα μιγαδικό αριθμό.

Παρατήρηση 2: Στην εμφάνιση του αποτελέσματος, το matlab μας δείχνει μόνο ότι ο A είναι ένας 2X2 πίνακας κελιών, αλλά δεν μας δείχνει τα περιεχόμενα όλων των κελιών. Εμφανίζονται μόνο τα κελιά εκείνα που δεν καταλαμβάνουν σημαντικό χώρο στην οθόνη. Για όλα τα υπόλοιπα απλώς παρουσιάζει τον τύπο τους. Αν ο χρήστης θέλει να δει τα περιεχόμενα ενός κελιού, πρέπει να χρησιμοποιηθεί η τεχνική που αναφέραμε προηγούμενα σαν array indexing, μόνο που τώρα αντί για παρενθέσεις, χρησιμοποιούνται άγκιστρα. Για παράδειγμα, η εμφάνιση των περιεχομένων του πρώτου κελιού, (1η γραμμή, 1η στήλη) πραγματοποιείται με την εξής εντολή:

```
>>A{1,1}
ans =

   1   2   3
   4   5   6
   7   8   9
```

- Ο **δεύτερος τρόπος** είναι τα άγκιστρα { } να πάνε στο αριστερό μέρος, δηλαδή:
  ```
  >>A{1,1}= [1 2 3;4 5 6;7 8 9] ;  %πίνακας
  >>A{1,2}= 2+3i ;  %μιγαδικός
  >>A{2,1}= 'Ακολουθία χαρακτήρων' ;  %ακολουθία χαρακτήρων
  >>A{2,2}= 0:2:10 ;  %διάνυσμα με στοιχεία από 0 έως 10 με βήμα 2
  ```

6. ΚΑΤΑΣΚΕΥΗ ΠΙΝΑΚΩΝ ΔΟΜΩΝ (structure arrays)

Οι πίνακες δομών μοιάζουν με τους πίνακες κελιών, στο ότι επιτρέπουν σε κάποιον να ομαδοποιήσει διαφορετικούς τύπους στοιχείων σε μία και μοναδική μεταβλητή. Όμως οι δομές δεν έχουν κελιά, αλλά έχουν πεδία (fields) και τα οποία αντιπροσωπεύονται από ονόματα. Ενώ οι πίνακες κελιών χρησιμοποιούν άγκιστρα { } για την πρόσβαση στα δεδομένα κάθε κελιού, τα structures χρησιμοποιούν την τελεία (.) για την πρόσβαση στα δεδομένα κάθε πεδίου. Για παράδειγμα:

```
>>circle.radius= 2.5;
>>circle.center= [0 1];
>>circle.linestyle= '- -';
>>circle.color= 'red'

circle =
        radius: 2.5
        center: [0 1]
        linestyle: '- -'
        color: 'red'
```

Εδώ βλέπουμε την δήλωση των δεδομένων ενός κύκλου (ακτίνα, κέντρο, τύπος γραμμής, χρώμα) σε μία δομή. Η δομή, ονομάζεται circle και έχει τα παρακάτω πεδία: radius, center, linestyle, color. Αν θέλουμε να εισάγουμε και τα δεδομένα ενός δεύτερου κύκλου μπορούμε να τον αποθηκεύσουμε, **σαν το δεύτερο στοιχείο της μεταβλητής circle** ως εξής:

```
>>circle(2).radius= 3.4;
>>circle(2).center= [2.3 -1.2];
>>circle(2).linestyle=  ' : ';
>>circle(2).color= 'green'

circle=
1X2 struct array with fields:
    radius
    center
    linestyle
    color
```

Παρατήρηση: όπως και στους πίνακες κελιών, έτσι και στους πίνακες δομών, το matlab δεν παρουσιάζει τα δεδομένα των πεδίων αναλυτικά διότι όλα μαζί δεν χωρούν στην οθόνη. Τα περιεχόμενα του κάθε πεδίου μπορούν να ανακτηθούν, αν πληκτρολογηθεί το όνομά του στο «command prompt».

```
>>circle (2).radius
ans =        3.4
```

7. ΑΚΟΛΟΥΘΙΕΣ ΧΑΡΑΚΤΗΡΩΝ (character strings)

Οι ακολουθίες χαρακτήρων στο matlab, είναι ειδικοί αριθμητικοί πίνακες, που περιέχουν τιμές κώδικα ASCII, που αντιστοιχούν στην αναπαράσταση του κάθε χαρακτήρα (γράμματος ή αριθμού). Οποιοδήποτε κείμενο το οποίο περικλείεται από απλά εισαγωγικά (‘ ‘), αποτελεί ακολουθία χαρακτήρων. Για παράδειγμα:

```
>>t=’ How about this character string? ‘
t=
How about this character string?
```

Με την εντολή **size**, μπορούμε να βρούμε το μέγεθος κάθε πίνακα (αριθμός γραμμών και στηλών).

```
>>size(t)
ans
      1       32
```

Η ακολουθία χαρακτήρων αποτελεί κατ’ ουσία έναν πίνακα-γραμμή με 32 στήλες (συμπεριλαμβανομένων των κενών διαστημάτων). Το συμπέρασμα που προκύπτει, είναι ότι **τις ακολουθίες χαρακτήρων μπορούμε να τις διαχειριστούμε όπως ακριβώς τους πίνακες (πρόσβαση και αλλαγή δεδομένων, αναστροφή κα)**.

Με την παρακάτω εντολή, μπορούμε να δούμε με ποιόν αριθμό αναπαρίσταται ο κάθε χαρακτήρας της μεταβλητής t που δηλώσαμε παραπάνω.

```
>>u=double(t)
 columns 1 through 8

  72  111  119   32   97   98  111  117

 columns 9 through 16

 116   32  116  104  105  115   32   99

 columns 17 through 24

 104   97  114   97   99  116  101  114

 columns 25 through 32

  32  115  116  114  105  110  103   63
```

Σαν πίνακες, οι ακολουθίες χαρακτήρων μπορούν να έχουν πολλές γραμμές, αλλά κάθε γραμμή, πρέπει να έχει **ίσο αριθμό στηλών**. Αυτό στην πράξη επιτυγχάνεται με την τοποθέτηση **κενών διαστημάτων** όπου χρειάζεται. Για παράδειγμα:

```
>>v=[ 'my thesis is almost over   '
      'I must start giving my     '
      'texts to Mr.Frangopoulos  '
```

Υπάρχουν και οι πολύτιμες συναρτήσεις **char** και **strvcat** για τη δημιουργία πινάκων χαρακτήρων με πολλές γραμμές, από ξεχωριστά strings διαφορετικού μήκους. Για παράδειγμα:

```
>>lezanda=char('Michael', 'Helen', 'John', 'James' )
lezanda=
Michael
Helen
John
James

>>lezanda=strvcat('Michael', 'Helen', 'John', 'James' )
lezanda=
Michael
Helen
John
James
>>size(lezanda)
ans =
     4     7
```

Η μόνη διαφορά μεταξύ των char και strvcat είναι ότι η strvcat αγνοεί τις «κενές» ακολουθίες χαρακτήρων (' '), ενώ η char εισάγει κενές γραμμές στις θέσεις των κενών strings. Για παράδειγμα:

```
>>char ('one', ' ', 'two' )
ans =
one

two

>>strvcat ('one', ' ', 'two' )
ans =
one
two
```

7.1 Μετατροπή αριθμών σε χαρακτήρες και χαρακτήρων σε αριθμούς

Υπάρχουν πάρα πολλές περιπτώσεις που χρειάζεται η μετατροπή αριθμητικών αποτελεσμάτων σε ακολουθίες χαρακτήρων και αντίστροφα. Πληθώρα εσωτερικών συναρτήσεων πραγματοποιούν αυτή τη μετατροπή.

- Με την εντολή **int2str** μετατρέπεται ένας αριθμητικός πίνακας ακεραίων σε πίνακα χαρακτήρων. Για παράδειγμα:

```
>> int2str(eye(3))
ans=
1  0  0
0  1  0
0  0  1
```

Αν πληκτρολογήσουμε τώρα την εντολή size για να δούμε τη διάσταση του πίνακα παίρνουμε:

```
>>size(ans)
ans=
     3     7
```

Δηλαδή ο 3Χ3 αριθμητικός πίνακας, μετατράπηκε σε έναν 3Χ7 πίνακα χαρακτήρων (συμπεριλαμβανομένων των κενών διαστημάτων).

- Με την εντολή **num2str** μετατρέπεται ένας αριθμητικός πίνακας πραγματικών αριθμών σε πίνακα χαρακτήρων.

```
>> num2str(rand(2,4))
ans=
0.95013   0.60684   0.8913   0.45647
0.23114   0.48598   0.7621   0.01850

>>size(ans)
ans=
     2     40
```

Δηλαδή ο 2Χ4 πίνακας, μετατράπηκε σε πίνακα χαρακτήρων με 2 γραμμές και 40 στήλες (συμπεριλαμβανομένων των κενών διαστημάτων).

Σημείωση: Υπενθυμίζεται, ότι μια ακολουθία χαρακτήρων μπορεί να αποτελεί το περιεχόμενο ενός ή περισσοτέρων κελιών, σε ένα πίνακα κελιών (cell array). Επίσης, είναι

δυνατόν, με την εντολή **cellstr**, να μετατρέψουμε ένα **πίνακα χαρακτήρων σε πίνακα κελιών.**

7.2 Υπολογισμός τιμής ακολουθίας χαρακτήρων (string evaluation)

Η έννοια του υπολογισμού της τιμής μιας ακολουθίας χαρακτήρων αφορά τη συγγραφή κάποιας συνάρτησης σαν ακολουθία χαρακτήρων και υπολογισμός της τιμής της σαν να ήταν μαθηματική έκφραση. Υπάρχουν πολλές συναρτήσεις που κάνουν αυτούς τους υπολογισμούς, θα μελετήσουμε όμως κάποιες από αυτές αργότερα, όταν θα μιλήσουμε για τις συναρτήσεις του optimization toolbox.
Για παράδειγμα, έχουμε την παρακάτω ακολουθία χαρακτήρων:

```
>>func=' x^2-5*x+6 '
```

Έστω ότι θέλουμε να υπολογίσουμε την έκφραση αυτή για την τιμή x=5. Για να γίνει αυτό πληκτρολογούμε τα παρακάτω:

```
>>x=5;
>>eval(func)

ans =
     6
```

8. ΕΙΣΑΓΩΓΗ ΣΤΑ ΑΡΧΕΙΑ ΚΕΙΜΕΝΟΥ (script m-files)

Για απλά προβλήματα, το να γράψει ο χρήστης τις εντολές που θέλει στο «command prompt» είναι γρήγορο και αποτελεσματικό. Καθώς, όμως, ο αριθμός των εντολών αυξάνει, ή όταν πρέπει να αλλάξουμε την τιμή μιας μεταβλητής, το να πληκτρολογήσουμε ξανά όλες τις εντολές από την αρχή στο «command prompt», μπορεί να είναι ιδιαίτερα επίπονο. Το πρόγραμμα παρέχει τη δυνατότητα στο χρήστη, να πληκτρολογήσει τις εντολές του σε ένα αρχείο κειμένου το οποίο μπορεί να «ανοίξει» και να εκτελέσει τις εντολές, ακριβώς όπως θα έκανε αν εισήγαγε τις εντολές του απ' ευθείας στο «command prompt». Αυτά τα αρχεία λέγονται **script m-files**. Λέγονται δε m-files διότι η επέκταση τους έχει την επέκταση ' .m'.

Για παράδειγμα ας δούμε το παρακάτω αρχείο:

```
%script m-file example.m

erasers=4; %number of each item
pads=6;
tape=2;
items=erasers+pads+tape
cost=erasers*25+pads*52+tape*99
average_cost=cost/items
```

Το αρχείο αυτό μπορεί να σωθεί στο σκληρό δίσκο και να εκτελεστεί άμεσα με την επιλογή **Run** από το μενού **Debug**, ή πιέζοντας το πλήκτρο **F5**.

Εναλλακτικά, μπορεί να σωθεί στο δίσκο σαν example.m και μετά να πληκτρολογηθεί το όνομά του στο «command prompt» χωρίς να γραφεί η επέκτασή του. Το matlab κάνει τους υπολογισμούς σαν να πληκτρολογούσαμε τις εντολές απ' ευθείας στο «command prompt». **Σαν αποτέλεσμα, οι εντολές που είναι γραμμένες στο m-file έχουν πρόσβαση σε όλες τις μεταβλητές που έχουν αποθηκευτεί στο matlab workspace, και όλες οι μεταβλητές που δημιουργήθηκαν από το m-file αποτελούν μέρος του workspace**.

Υπάρχουν αρκετές συναρτήσεις οι οποίες μπορούν να αποδειχθούν ιδιαίτερα χρήσιμες στην συγγραφή των m-files, όπως οι παρακάτω:

Πίνακας 8.1Χρήσιμες συναρτήσεις για τα m-files

Συνάρτηση	Περιγραφή
beep	Προκαλεί τον ήχο «μπίπ» στον υπολογιστή
disp(variablename)	Εμφάνιση του αποτελέσματος μιας μεταβλητής χωρίς το όνομά της.
echo	Με την εντολή 'echo on' το πρόγραμμα δείχνει τις εντολές που χρησιμοποιούνται βήμα προς βήμα, καθώς αυτές διαβάζονται και εκτελούνται. Σε άλλη περίπτωση μπορούμε να γράψουμε 'echo off'
input	Αναμένει τον χρήστη να εισάγει δεδομένα. (αυτή η συνάρτηση συναντάται και σε άλλες γλώσσες προγραμματισμού.
keyboard	Δίνει έλεγχο στο πληκτρολόγιο προσωρινά διακόπτοντας την εκτέλεση του m-file (ώστε να γίνει κάποια διόρθωση, κάποια αλλαγή της τιμής μιας μεταβλητής κα). Με την εντολή return συνεχίζεται η εκτέλεση του m-file.
pause ή pause(n)	Παύση της εκτέλεσης μέχρι να πατηθεί κάποιο πλήκτρο, και παύση για n δευτερόλεπτα και συνέχεια της εκτέλεσης
waitforbuttonpress	Παύση της εκτέλεσης μέχρι να πατηθεί κάποιο πλήκτρο, ή το κουμπί του ποντικιού.

Πολλές φορές, για λόγους εμφάνισης, είναι πιο βολικό να προβάλλουμε δεδομένα με την εντολή **disp**. Για παράδειγμα:

```
>>items
items=
   12
>>disp(items)
   12
```

Άλλες φορές, όταν χρειάζεται να μελετήσουμε το ίδιο πρόβλημα με διαφορετικές τιμές, είναι αρκετά βολικό να χρησιμοποιούμε την εντολή input. Με την εντολή **input** μπορεί το πρόγραμμα να σταματά προσωρινά, έως ότου δεχθεί δεδομένα από τον χρήστη. Ένα τέτοιο παράδειγμα φαίνεται παρακάτω:

```
%script m-file example.m

erasers=4; %number of each item
pads=6;
tape=input(' Enter the number of tapes>  ' );
items=erasers+pads+tape
cost=erasers*25+pads*52+tape*99
average_cost=cost/items
```

εκτελώντας το αρχείο παίρνουμε τα εξής:

```
>>example
Enter the number of tapes>  3
items=
   13
cost=
   709
average_cost=
        54.538
```

Όταν εμφανίστηκε το μήνυμα: "Enter the number of tapes>", δώσαμε τον αριθμό 3 και πατήσαμε το πλήκτρο «Enter». Όλοι οι υπολογισμοί, έπειτα έγιναν κανονικά όπως προηγούμενα.

9. ΕΙΣΑΓΩΓΗ ΣΤΑ ΑΡΧΕΙΑ ΣΥΝΑΡΤΗΣΕΩΝ (function m-files)

Τα function m-files είναι αρχεία του matlab τα οποία περιέχουν δηλώσεις συναρτήσεων με τα ορίσματα εισόδου και εξόδου τους. Είναι αρχεία που μπορούν να εκτελεστούν αν στο «command window» πληκτρολογήσουμε το όνομά τους και τις μεταβλητές εισόδου. Η διαφορά τους από τα script m-files είναι ότι πάντα ξεκινούν με τη λέξη function. Ας δούμε το παρακάτω παράδειγμα που υπολογίζει την τιμή μιας συνάρτησης όταν δίνονται οι παράμετροί της α, β. Επιλέγουμε τη διαδρομή **File>New>m file** και γράφουμε τα παρακάτω:

```
% ας υπολογίσουμε  την τιμή της παρακάτω συνάρτησης
function func=evaluate(α,β);
func=sqrt(α^2+β^2*a^b)
```

Σώζουμε το αρχείο σαν evaluate.m. Func ονομάζεται η συνάρτηση υπολογισμού της συνάρτησης, ενώ α, β είναι οι μεταβλητές.
Αν τώρα πάμε στο «command window» και πληκτρολογήσουμε:

```
>>evaluate(5,6)
```

Το matlab δίνει κατευθείαν την απάντηση:

```
ans =

 750.0167
```

Σημείωση: Για να μπορέσει να εκτελεστεί ένα function m-file, θα πρέπει πρώτα να βρισκόμαστε στο φάκελο (**current directory**) που αυτό είναι αποθηκευμένο. Αν δεν τηρείται αυτή η προϋπόθεση τότε το matlab θα εμφανίσει μήνυμα λάθους.

9.1 Κανόνες συγγραφής αρχείων συναρτήσεων (function m-files)

1. Η πρώτη γραμμή ενός αρχείου συναρτήσεων (function m-file) λέγεται γραμμή δήλωσης συνάρτησης και πρέπει να περιέχει τη λέξη **function** ακολουθούμενη από τη σύνταξη της συνάρτησης στη γενική της μορφή με τις μεταβλητές εισόδου και εξόδου. Οι μεταβλητές εισόδου τροφοδοτούν τη συνάρτηση με δεδομένα, ενώ οι μεταβλητές εξόδου περιέχουν δεδομένα μετά την εκτέλεση της συνάρτησης.

   Είναι χρήσιμο, στην αρχή της σύνταξης να υπάρχουν κάποια σχόλια σχετικά με τον σκοπό της συνάρτησης και το πρόβλημα που επιλύει, καθώς και τις μεταβλητές εισόδου και εξόδου αναλυτικά. Αυτό βοηθά στο να θυμάται ο συντάκτης το περιεχόμενο του αρχείου ακόμα και αν περάσει αρκετός

καιρός. Η εισαγωγή σχολίου στο matlab γίνεται πολύ απλά. Αρκεί να θέσουμε στην αρχή της γραμμής σύνταξης **το σύμβολο (%)**. Ότι γράφεται μετά από αυτό το σύμβολο, το matlab το αντιλαμβάνεται σαν σχόλιο.

2. ένα function m-file τερματίζεται όταν εκτελεστεί και η τελευταία γραμμή του αρχείου, ή όταν παρουσιαστεί κάποιο σφάλμα και διακοπεί η εκτέλεση. Η λειτουργία μιας συνάρτησης μπορεί να διακοπεί αν κληθεί η συνάρτηση **error**. Η συνάρτηση αυτή είναι πολύ χρήσιμη, διότι μπορούμε να κάνουμε έλεγχο πάνω στις μεταβλητές μας. Έστω για παράδειγμα ότι έχουμε τη μεταβλητή εισόδου var η οποία πρέπει να είναι σταθερά. Αν ο χρήστης τη θέσει σαν διάνυσμα (δηλαδή 2 διαστάσεις αντί για 1), πχ var=[1 2]; τότε μπορεί να γραφεί ένας κώδικας ελέγχου για την var ώστε το πρόγραμμα να τερματίζεται και να παίρνουμε μήνυμα λάθους, αν δεν δίνεται ως σταθερά από τον χρήστη:

```
if length(val)>1
  error(' val must be a scalar.' )
end
```

3. τα function m-files μπορούν να περιέχουν κλήσεις σε

- **Script m-files** τα οποία εξηγήθηκαν αναλυτικά στο 8ο κεφάλαιο.

- **Subfunctions** δηλαδή υποσυναρτήσεις, μέσα στο ίδιο function m-file. Ας δούμε το παρακάτω παράδειγμα υπολογισμού της μέσης τιμής ενός συνόλου. Το αρχείο ονομάζεται newstats.m

```
function avg= newstats(u) % Αρχική συνάρτηση
% NEWSTATS Βρές μέση τιμή με εσωτερικές συναρτήσεις
n = length(u);
avg = mean(u,n);

function a = mean(v,n)          % Subfunction
% Calculate average.
a = sum(v)/n;
```

Στο παραπάνω παράδειγμα η πρωταρχική συνάρτηση newstats υπολογίζει τη διάσταση του ορίσματος u και έπειτα καλεί την υποσυνάρτηση mean δίνοντας της την τιμή του n.
Οι υποσυναρτήσεις είναι ορατές μόνο από την πρωταρχική συνάρτηση (primary function) ή από άλλες υποσυναρτήσεις στο ίδιο αρχείο. Οι διάφορες υποσυναρτήσεις μπορούν να δηλωθούν σε οποιοδήποτε σειρά αρκεί η πρωταρχική συνάρτηση να έχει δηλωθεί πρώτη.

Πολλές συναρτήσεις μέσα στο ίδιο m-file **δεν μπορούν να έχουν πρόσβαση στις ίδιες μεταβλητές** εκτός αν αυτές δηλωθούν σαν ολικές μεταβλητές (global variables) μέσα σε κάθε υποσυνάρτηση, ή αν χρησιμοποιηθούν σαν ορίσματα εισόδου στις υποσυναρτήσεις. (Ενότητα 9.3)

- **Private m-files.** Είναι στην πραγματικότητα function m-files που περιέχονται σε υποφακέλους με την ονομασία private. Ας υποθέσουμε ότι εκτελούμε κάποιο αρχείο από τον φάκελο newmath (current directory). Ένας υποφάκελος του newmath που ονομάζεται private μπορεί να περιέχει αρχεία συναρτήσεων που μόνο τα αρχεία συναρτήσεων στον γονικό φάκελο (parentdirectory) newmath μπορούν να καλέσουν. Επειδή ένα private αρχείο συναρτήσεων δεν είναι ορατό όταν βρισκόμαστε έξω από το γονικό φάκελο μπορεί να έχει το ίδιο όνομα με συναρτήσεις σε άλλους φακέλους. Αυτό είναι ιδιαίτερα χρήσιμο όταν ο χρήστης θέλει να τροποποιήσει ένα αρχείο συναρτήσεων ενώ έχει το πρωτότυπο σε άλλο φάκελο.

9.2 Ορίσματα εισόδου-εξόδου στα αρχεία συναρτήσεων (input, output arguments)

1. Τα function m-files μπορούν να κληθούν με λιγότερα ορίσματα εισόδου και εξόδου απ' ότι όταν δημιουργήθηκαν. Δεν μπορούν όμως να εκτελεστούν με περισσότερα ορίσματα από όσα καθορίζονται από τη συνάρτηση.

Έστω, για παράδειγμα η συνάρτηση y: **function y=digit(x,n,b,t)**. Εδώ τα ορίσματα εισόδου είναι προφανώς τα x, n, b, t. Κατά την εκτέλεση αυτής της συνάρτησης μπορούμε να εισάγουμε λιγότερα ορίσματα, δηλαδή στο «command window» να πληκτρολογήσουμε **>>digit(1 1.4 2.3)**, και να μην δώσουμε τιμή για το t. Με την συνάρτηση **nargin** μπορούμε να ελέγξουμε τα ορίσματα εισόδου και να οδηγήσουμε το πρόγραμμα στο να ακολουθήσει άλλες εντολές ανάλογα με τον αριθμό των ορισμάτων. Για παράδειγμα:

```
function y=digit(x,n,b,t)
if nargin<2  %Αν τα ορίσματα εισόδου λιγότερα από 2 εμφάνιση μηνύματος λάθους
  error('Not enough input arguments')
elseif nargin==2  %ορίσματα εισόδου=2 εκτελούνται οι παρακάτω εντολές
……..
elseif nargin==3  %ορίσματα εισόδου=3 εκτελούνται οι παρακάτω εντολές
……..
elseif nargin==4  %ορίσματα εισόδου=4 εκτελούνται οι παρακάτω εντολές
……..
end
```

2. Στις δηλώσεις των συναρτήσεων μπορούμε να χρησιμοποιήσουμε απεριόριστο αριθμό ορισμάτων εισόδου. Όταν έχουμε πολλά ορίσματα, καλό είναι, για πρακτικούς λόγους, να υπάρχει ένας πίνακας που θα περιέχει όλες τις μεταβλητές μας. Στο προηγούμενο παράδειγμα, αντί να γράψουμε:

>> function y=digit(x,n,b,t),

μπορούμε να γράψουμε:

>>function y=digit(varargin)

όπου το x είναι το πρώτο στοιχείο του πίνακα varargin ενώ n είναι το δεύτερο στοιχείο του πίνακα varargin, κτλ. Το ίδιο συμβαίνει και με τα ορίσματα εξόδου. Μπορούμε να δηλώσουμε:

>>function varargout=digit(varargin)

Αν η συνάρτηση κληθεί σαν:

>>[a,b]=digit(varargin)

τότε το πρώτο στοιχείο του πίνακα varargout θα περιέχει το a (πρώτο όρισμα εξόδου) ενώ το δεύτερο στοιχείο του πίνακα varargout θα περιέχει το b (δεύτερο όρισμα εξόδου).

9.3 Χώροι εργασίας συναρτήσεων (function workspaces)

Οι συναρτήσεις θα μπορούσαν να περιγραφούν σαν μαύρα κουτιά. Δέχονται δεδομένα εισόδου, εργάζονται με αυτά και μετά δημιουργούν δεδομένα εξόδου. Όλες οι μεταβλητές που δημιουργούνται μέσα στη συνάρτηση είναι κρυφές από τον χώρο εργασίας του matlab, δηλαδή από τον κεντρικό χώρο στον οποίο αποθηκεύονται οι μεταβλητές (που ονομάζεται και βασικός χώρος εργασίας).

Κάθε συνάρτηση έχει το δικό της προσωρινό χώρο εργασίας που δημιουργείται με κάθε εκτέλεση της συνάρτησης και διαγράφεται όταν η συνάρτηση τερματιστεί. Παρόλα αυτά:

1. Οι συναρτήσεις μπορούν να μοιραστούν μεταβλητές με άλλες συναρτήσεις καθώς και με το χώρο εργασίας του matlab, αν αυτές δηλωθούν σαν **global**. Για να υπάρχει πρόσβαση σε μια μεταβλητή από κάποιο άλλο workspace, τότε αυτή **πρέπει να δηλωθεί σαν global τόσο στο παρόν workspace όσο και σε εκείνο που θέλει να την χρησιμοποιήσει**.

2. Για επαναληπτικές εκτελέσεις μια συνάρτησης, πολλές φορές οι μεταβλητές δηλώνονται σαν **persistent** αντί για global. Όταν μια μεταβλητή είναι persistent, η τιμή της διατηρείται στη μνήμη μεταξύ των επαναληπτικών εκτελέσεων και χρησιμοποιείται πάλι. Οι persistent μοιάζουν με τις global στο ότι το matlab δίνει μόνιμο χώρο αποθήκευσης γι' αυτές. Διαφέρουν όμως στο ότι είναι γνωστές μόνο στην συνάρτηση στην οποία δηλώνονται. Αυτό τις προφυλάσσει από το να αλλαχτούν οι τιμές τους από επίδραση άλλων συναρτήσεων ή από το ««command prompt»» του matlab.

3. Το matlab παρέχει τη συνάρτηση **evalin** που επιτρέπει σε μια συνάρτηση να μπει σε ένα άλλο χώρο εργασίας, να υπολογίσει μια έκφραση σε αυτό και να επιστρέψει το αποτέλεσμα στο παρόντα χώρο εργασίας. Κληθέντα χώρο εργασίας (caller workspace) ονομάζουμε το χώρο εργασίας από όπου κλήθηκε η συνάρτηση. Βασικός χώρος εργασίας (base workspace) ονομάζεται ο χώρος εργασίας του matlab. Για παράδειγμα η εντολή **A=evalin('caller','expression')** υπολογίζει το 'expression' στο κληθέν workspace και επιστρέφει το αποτέλεσμα σε μια μεταβλητή A στο παρών workspace.

4. Κάτι που μπορεί επίσης να γίνει, είναι να υπολογίσουμε κάποια έκφραση στο παρόν workspace και να εισάγουμε το αποτέλεσμα σαν μεταβλητή σε ένα άλλο workspace. Αυτό γίνεται με τη συνάρτηση **assignin**. Για παράδειγμα, αν γράψουμε **assignin('workspace','vname',X)** όπου 'workspace' είναι είτε το κληθέν είτε το βασικό, θέτει το περιεχόμενο της μεταβλητής X του παρόντος workspace, σε μια μεταβλητή στο κληθέν ή το βασικό workspace, που ονομάζεται 'vname'.

9.4 Υπολογισμός τιμής συνάρτησης (function evaluation)

Υπάρχουν δύο τρόποι να υπολογιστεί η τιμή μιας συνάρτησης σε ένα function m-file.

1. Με εκτέλεση του αρχείου από το «command window» του matlab. Μπορούμε να γράψουμε δηλαδή:

   >>a=myfunction(x)

   Με αυτό τον τρόπο εκτελείται το αρχείο myfunction.m για το x που επελέγη και το αποτέλεσμα αποθηκεύεται στη μεταβλητή a.

2. Με εκτέλεση του αρχείου κάνοντας χρήση της συνάρτησης **feval** (function evaluation). Η συνάρτηση feval δουλεύει με ακολουθίες χαρακτήρων (character strings). Είναι χρήσιμο πολλές φορές να μπορούμε να υπολογίσουμε την τιμή μιας ακολουθίας χαρακτήρων σαν να ήταν μία συνάρτηση του matlab. Υπενθυμίζεται ότι σαν ακολουθία χαρακτήρων θεωρείται οποιοδήποτε κείμενο είναι κλεισμένο σε απλά εισαγωγικά (‘ ’) (βλέπε, κεφάλαιο 7). Στο προηγούμενο παράδειγμα μπορεί να γραφεί ισοδύναμα:

   >>a=feval(‘myfunction’,x)

   Επίσης, ισοδύναμα, μπορεί να χρησιμοποιηθεί η έκφραση:

   >>a=feval(@myfunction,x)

   Παράδειγμα: Έστω ότι θέλουμε να υπολογίσουμε το συνημίτονο για την τιμή του x=π/2. Το συνημίτονο είναι εσωτερική συνάρτηση του matlab και περιέχεται στο αρχείο cos.m Οι παρακάτω εκφράσεις είναι ισοδύναμες:

   - >> x=pi/2;
     >> cos(x)

     ans =

      6.1232e-017

   - >> feval('cos',pi/2)
     ans =

      6.1232e-017

- >> feval(@cos,pi/2)

  ans =

  6.1232e-017

  Το σύμβολο @ είναι πολύ σημαντικό (σημαίνει function evaluation). Θα χρησιμοποιηθεί και αργότερα και στη σύνταξη της εντολής fmincon για να υπολογιστεί η αντικειμενική συνάρτηση για το δοθέν αρχικό σημείο.

Σημείωση 1: Ο υπολογισμός της τιμής μιας συνάρτησης δεν γίνεται απαραίτητα από κάποιο αρχείο m-file που θα την περιέχει, αλλά μπορεί να γίνει δηλώνοντας την συνάρτηση σαν **ακολουθία χαρακτήρων**. Με αυτό τον τρόπο, στον υπολογισμό της τιμής της συνάρτησης δεν χρησιμοποιείται η feval αλλά η **eval**. Έστω η συνάρτηση $f = 100 \cdot (1 - x^2)^2 + (1 - x)^2$ και θέλουμε να την υπολογίσουμε την τιμή της για για x=5. Τότε μπορεί να γραφεί:

```
>> myfun='100*(1-x^2)^2+(1-x)^2';
>>x=5;
>> a=eval(myfun)

a =

 215.6524
```

Σημείωση 2: Η συνάρτηση feval δεν χρησιμοποιείται μόνο για συναρτήσεις μίας μεταβλητής, αλλά και για συναρτήσεις πολλών μεταβλητών, με την παράθεση τους στη γραμμή σύνταξης. Για παράδειγμα: feval(@fun, x_1,x_2,x_3,…).

10. ΔΙΟΡΘΩΣΗ ΣΦΑΛΜΑΤΩΝ ΣΕ m-files (DEBUGGING)

Τα λάθη που εμφανίζονται σε μία έκφραση του matlab, είναι δύο ειδών:

- Συντακτικά λάθη (syntax errors)
- Λάθη χρόνου εκτέλεσης (run-time errors)

Στην πρώτη περίπτωση είναι πολύ εύκολος ο εντοπισμός και η διόρθωση του σφάλματος. Όταν υπάρξει σφάλμα κατά την εκτέλεση, τότε το matlab παρέχει πληροφορίες σχετικά με την αιτία του λάθους και σε ποιο σημείο ακριβώς (γραμμή και στήλη) στο αρχείο βρίσκεται αυτό. Έτσι, με κατάλληλο χειρισμό, το λάθος μπορεί να διορθωθεί.

Στη δεύτερη περίπτωση, τα περισσότερα run-time errors συμβαίνουν όταν τα αποτελέσματα πράξεων οδηγούν σε κενούς πίνακες (empty arrays) ή σε NaN's (Nan=not a number). Οι παρακάτω πράξεις παράγουν NaN:

- Πρόσθεση ή αφαίρεση: (+Inf)+(-Inf)
- Πολλαπλασιασμός: 0*Inf
- Διαίρεση: 0/0 και Inf/Inf
- Υπόλοιπο: όπως rem(x,y) όπου το y είναι μηδέν ή το x είναι άπειρο.
- Κάθε πράξη σε NaN, όπως sqrt(NaN)

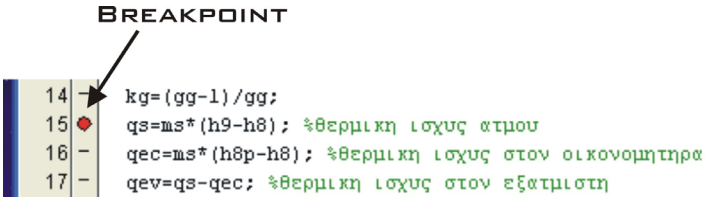
Για απλά προβλήματα προτείνονται συνδυασμοί των ακολούθων μεθόδων για τον εντοπισμό των λαθών:

- ✓ **Απομάκρυνση, όπου υπάρχει της αγγλικής άνω τελείας (;)** από το τέλος των εντολών ώστε τα ενδιάμεσα αποτελέσματα να εμφανίζονται στο «command window». Έτσι είναι πιο εύκολος ο εντοπισμός του λάθους.

- ✓ Εισαγωγή εντολών στο πρόγραμμα, που **εμφανίζουν τις τιμές των μεταβλητών** που μας ενδιαφέρουν μέσα στη συνάρτηση (όπως η **disp** που συζητήθηκε στο κεφάλαιο 8).

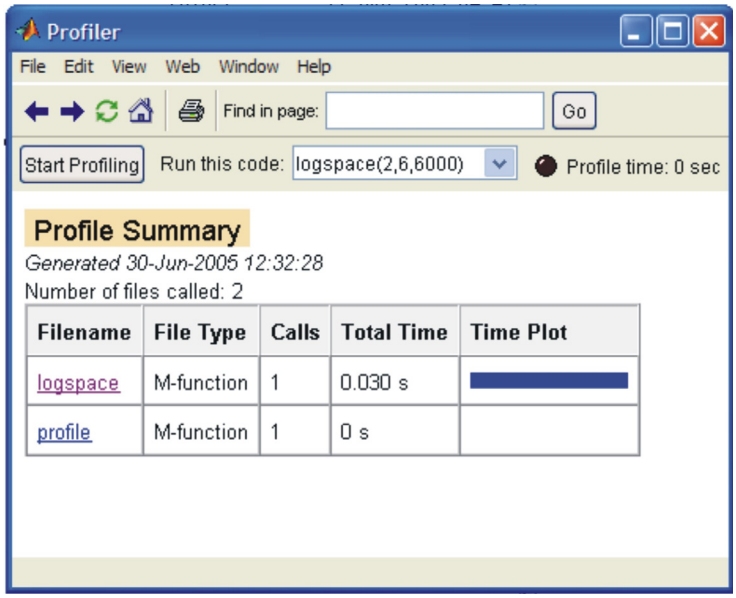
- ✓ **Τοποθέτηση της εντολής keyboard** σε συγκεκριμένα μέρη στο m-file, για να περάσει ο έλεγχος στον χρήστη μέσω του πληκτρολογίου. Με αυτό τον τρόπο, ο χώρος εργασίας (workspace) της συνάρτησης μπορεί να τροποποιηθεί και να αλλάξουν οι τιμές ορισμένων μεταβλητών εφόσον είναι απαραίτητο. Το πρόγραμμα συνεχίζει την εκτέλεσή του αν θέσουμε την εντολή **return** στο keyboard prompt δηλαδή **K>>return** .

- Υπάρχουν βέβαια και άλλες μέθοδοι που χρησιμοποιούνται στον έλεγχο των m-files και είναι τα λεγόμενα **breakpoints**. Τα breakpoints δημιουργούνται πολύ εύκολα, αν ο χρήστης κάνει «κλικ» στον αριθμό της γραμμής που θέλει να τοποθετήσει το breakpoint κατά τη σύνταξη του αρχείου. Αμέσως έχουμε την εμφάνιση μιας **κόκκινης** (συνήθως) κουκίδας, που υποδεικνύει την ύπαρξη breakpoint. Όταν κατά την εκτέλεση του αρχείου το πρόγραμμα φτάσει σε ένα breakpoint, τότε σταματά την εκτέλεση και εμφανίζει τα αποτελέσματα, επιτρέποντας την αλλαγή τιμών μεταβλητών στο workspace μέχρις ότου εντοπίσουμε το λάθος. (Βλέπε εικόνα 8.1)



Εικόνα 10.1 Ύπαρξη breakpoint σε m-file

- Ένα άλλο εργαλείο που μας δίνει το matlab, κυρίως για τη βελτιστοποίηση της εκτέλεσης function m-files είναι ο **profiler**. Αυτό, παρακολουθεί την εκτέλεση του κώδικα και βλέπει ποιες γραμμές του καταναλώνουν το μεγαλύτερο χρόνο για την εκτέλεσή τους σε σχέση με τον υπόλοιπο κώδικα. Ας δούμε το παράδειγμα: Ανοίγουμε το παράθυρο του **profiler** και στο πλαίσιο **run this code** πληκτρολογούμε τις εντολές που θέλουμε να ελεγχθεί και πατάμε **start profiling**. Όταν τελειώσει το profiling εμφανίζεται το **profile summary** που περιέχει όλες τις γραμμές τον εντολών που εκτελέστηκαν, με τις ενδείξεις του χρόνου εκτέλεσης σε δευτερόλεπτα, αλλά και σε ποσοστό του συνολικού χρόνου εκτέλεσης. (βλέπε εικόνα 8.2)



Εικόνα 10.2 Παράδειγμα χρήσης του profiler window

Σημείωση: Για να είναι αξιόπιστο το profiling, προτείνεται, η εκτέλεση του m-file να γίνει πολλές φορές. Γι' αυτό το σκοπό, χρήσιμη είναι η εντολή **profile**. Για παράδειγμα, μπορούμε να γράψουμε το παρακάτω πρόγραμμα και να το εκτελέσουμε:

```
>>profile on
>>for i=1:100
      [m,e]=mmlog10(x);
   End
>>profile report
```

Εδώ το αρχείο mmlog10 εκτελείται 100 φορές για να είμαστε σίγουροι ότι οι χρόνοι που θα υπολογιστούν είναι ακριβείς.

11. ΕΙΣΑΓΩΓΗ ΣΤΟ OPTIMIZATION TOOLBOX

Πριν αναφερθούμε στο πρόβλημα βελτιστοποίησης που έχουμε να επιλύσουμε, είναι χρήσιμο να αναφερθούμε γενικά στα προβλήματα που επιλύονται χρησιμοποιώντας συναρτήσεις του optimization toolbox. Το optimization toolbox είναι μία πρόσθετη βιβλιοθήκη του matlab που παρέχει συναρτήσεις για προβλήματα βελτιστοποίησης. Το πρόβλημα που έχουμε να επιλύσουμε στο 1ο μέρος της διπλωματικής εργασίας αφορά την ελαχιστοποίηση με περιορισμούς (constrained minimization) και γι' αυτό το σκοπό θα χρησιμοποιηθεί η συνάρτηση **fmincon** της οποίας η χρήση θα εξηγηθεί αργότερα (Κεφάλαιο 12).

11.1 Ορίσματα εισόδου-εξόδου, όλων των συναρτήσεων του «OPTIMIZATION TOOLBOX»

Ο πίνακας 11.1.1, περιγράφει όλα τα ορίσματα εισόδου στις συναρτήσεις που χρησιμοποιούνται στο Optimization Toolbox. Επίσης αναφέρονται και τα ονόματα των συναρτήσεων που χρησιμοποιούν το κάθε όρισμα.

Πίνακας 11.1.1 Ορίσματα εισόδου συναρτήσεων του Optimization toolbox

Όρισμα εισόδου (input argument)	**Περιγραφή**	**Συναρτήσεις που τα χρησιμοποιούν**
A, b	Ο πίνακας A και το διάνυσμα b είναι αντίστοιχα ο συντελεστής και το διάνυσμα στο δεξί μέρος των γραμμικών ανισοτικών περιορισμών της μορφής: A*x<=b	fgoalattain, fmincon, fminimax, fseminf, linprog, lsqlin, quadprog
Aeq, beq	Ο πίνακας Aeq και το διάνυσμα beq είναι αντίστοιχα ο συντελεστής και το διάνυσμα στο δεξί μέρος των γραμμικών ισοτικών περιορισμών της μορφής: Aeq*x=beq	fgoalattain, fmincon, fminimax, fseminf, linprog, lsqlin, quadprog
C, d	Ο πίνακας C και το διάνυσμα d είναι αντίστοιχα οι παράμετροι που ορίζουν το υπέρ ή υπο-ορισμένο γραμμικό σύστημα, σε προβλήματα ελαχίστων τετραγώνων. Θυμίζουμε, ότι τα προβλήματα ελαχιστοποίησης με τη μέθοδο των ελάχιστων τετραγώνων δηλώνονται ως εξής: $\min_x \|C \cdot x - d\|_2^2$, m equations, n variables	lsqlin, lsqnonneg
f	Το διάνυσμα των συντελεστών του γραμμικού όρου στη γραμμική εξίσωση f' *x ή στην τετραγωνική εξίσωση x' *H*x+f ' *x	linprog, quadprog

fun	Η αντικειμενική συνάρτηση	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin
goal	Διάνυσμα τιμών που οι αντικειμενικές συναρτήσεις προσπαθούν να πετύχουν. Το διάνυσμα πρέπει να είναι του ίδιου μεγέθους με τον αριθμό των αντικειμενικών συναρτήσεων (για προβλήματα με πολλές αντικειμενικές συναρτήσεις-multi objective minimization)	fgoalattain
H	Ο πίνακας των συντελεστών για τους τετραγωνικούς όρους στην τετραγωνική εξίσωση x' *H*x+f' *x. Ο H πρέπει να είναι συμμετρικός.	quadprog
lb,ub	Κατώτατα η ανώτατα όρια, αντίστοιχα, για τις μεταβλητές. Μπορούν να' ναι διανύσματα ή πίνακες. Τα lb, ub πρέπει να είναι της ίδιας διάστασης με το x (διάνυσμα που περιέχει τις ανεξάρτητες μεταβλητές). Παρόλα αυτά αν τα lb, ub έχουν λιγότερα στοιχεία από το x, έστω m, τότε μόνο τα m στοιχεία του x περιορίζονται. Οι μεταβλητές που δεν επιδέχονται κανένα περιορισμό στις τιμές που μπορούν να πάρουν, προσδιορίζονται χρησιμοποιώντας τα σύμβολα: **–inf** (για το κατώτερο όριο) ή **inf** (για το ανώτερο όριο). Για παράδειγμα, αν lb(i)=-inf τότε η μεταβλητή x(i) δεν έχει κάτω όριο, (προσεγγίζει το -∞).	fgoalattain, fmincon, fminimax, fseminf, linprog, lsqcurveefit, lsqlin, lsqnonlin, quadprog
nonlcon	Είναι συνήθως κάποιο αρχείο function m-file που περιέχει τους μη γραμμικούς ανισοτικούς ή ισοτικούς περιορισμούς.	fgoalattain, fmincon, fminimax
ntheta	Ο αριθμός των ημιάπειρων περιορισμών.	Fseminf
options	Δομή, που ορίζει διάφορες παραμέτρους για τον αλγόριθμο βελτιστοποίησης.	Όλες οι συναρτήσεις
P1,P2	Επιπρόσθετες παράμετροι που μπορούν να εισαχθούν στην αντικειμενική συνάρτηση.	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin
seminfcon	Η συνάρτηση που υπολογίζει τους μη-γραμμικούς και γραμμικούς περιορισμούς, και τους ημιάπειρους περιορισμούς.	Fseminf
weight	Διάνυσμα βάρους που ελέγχει την υποεκτίμηση ή την υπερεκτίμηση των αντικειμενικών συναρτήσεων (για προβλήματα με πολλές αντικειμενικές	fgoalattain

	συναρτήσεις, multi-objective minimization)	
xdata, ydata	xdata είναι τα δεδομένα εισόδου(σημεία). ydata είναι τα μετρούμενα σημεία, για τα οποία πρέπει να γίνει παρεμβολή (να βρεθεί δηλαδή η εξίσωση της καμπύλης που να διέρχεται από αυτά τα σημεία).	lsqcurvefit
x0	Το αρχικό σημείο εκκίνησης, (μπορεί να είναι σταθερά, διάνυσμα ή πίνακας)	Όλες οι συναρτήσεις εκτός της fminbnd
x1, x2	Τα όρια μεταξύ των οποίων θα κινείται το x σε αντικειμενική συνάρτηση μιας μεταβλητής (x1, x2 σταθερές).	Fminbnd

Ο πίνακας 11.1.2 περιγράφει όλα τα ορίσματα εξόδου, στις συναρτήσεις που χρησιμοποιούνται στο Optimization Toolbox. Επίσης αναφέρονται και τα ονόματα των συναρτήσεων από τις οποίες εξάγεται το κάθε όρισμα.

Πίνακας 11.1.2 Ορίσματα εισόδου συναρτήσεων του Optimization toolbox

Όρισμα εξόδου (output argument)	**Περιγραφή**	**Συναρτήσεις που τα χρησιμοποιούν**
attainfactor	Ο συντελεστής προσέγγισης της βέλτιστης λύσης x.	Fgoalattain
exitflag	Μεταβλητή που περιγράφει την κατάσταση εξόδου. Αν **exitflag>0** σημαίνει ότι βρέθηκε η βέλτιστη λύση. Αν **exitflag<0** σημαίνει ότι ο αλγόριθμος δεν βρήκε τη βέλτιστη λύση. Αν **exitflag=0** σημαίνει ότι ο μέγιστος αριθμός των επαναλήψεων που δηλώθηκαν στην δομή options του αλγορίθμου, ξεπεράστηκαν χωρίς να βρεθεί βέλτιστη λύση.	Όλες οι συναρτήσεις
fval	Η τιμή της αντικειμενικής συνάρτησης στην βέλτιστη λύση x.	fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, linprog, quadprog
grad	Η τιμή του διανύσματος κλίσης στη βέλτιστη λύση x.	fmincon, fminunc
hessian	Η τιμή του Εσσιανού πίνακα στη βέλτιστη λύση x.	fmincon, fminunc
jacobian	Η τιμή του Ιακωβιανού πίνακα στην βέλτιστη λύση x.	lsqcurvefit, lsqnonlin, fsolve
lambda	Οι τιμές των πολλαπλασιαστών Langrange στη βέλτιστη λύση x.	fgoalattain, fmincon, fminimax, fseminf,

		linprog, lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg, quadprog
maxfval	Δίνει τη μέγιστη τιμή της αντικειμενικής συνάρτησης στην βέλτιστη τιμή x.	fminimax
output	Δίνει πληροφορίες σχετικά με τα αποτελέσματα της βελτιστοποίησης. Παρέχει πληροφορίες σχετικά με τον αλγόριθμο (τον αριθμό των επαναλήψεων, τον αριθμό των υπολογισμών της αντικειμενικής συνάρτησης (function evaluations), το όνομα του χρησιμοποιούμενου αλγόριθμου κα).	Όλες οι συναρτήσεις
residual	Δίνει την τιμή του C*x-d στη βέλτιστη λύση x. (Για τα προβλήματα ελάχιστων τετραγώνων)	lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg
resnorm	Δίνει την τιμή της νόρμας $\|\|C*x-d\|\|^2$ στη βέλτιστη λύση x.	lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg
x	Η βέλτιστη λύση που βρέθηκε από τη συνάρτηση βελτιστοποίησης. Αν exitflag>0 τότε το x είναι λύση, αλλιώς, είναι η τιμή την οποία υπολόγιζε εκείνη τη στιγμή η συνάρτηση βελτιστοποίησης, όταν τερματίστηκε πρόωρα για άγνωστο λόγο.	Όλες οι συναρτήσεις

11.2 Προβλήματα τα οποία επιλύονται με το «OPTIMIZATION TOOLBOX»

Στους πίνακες 11.2.1, 11.2.2, 11.2.3 παρουσιάζονται τα προβλήματα που μπορούν να επιλυθούν με το optimization toolbox.

Πίνακας 11.2.1 Προβλήματα ελαχιστοποίησης

Ελαχιστοποίηση (Minimization)

Τύπος	Σημειώσεις	Συναρτήσεις
Ελαχιστοποίηση με όριο (Scalar minimization)	$\min_a f(a) \quad \text{such that} \quad a_1 < a < a_2$	fminbnd
Ελαχιστοποίηση χωρίς περιορισμούς (Unconstrained minimization)	$\min_x f(x)$	fminunc, fminsearch
Γραμμικός προγραμματισμός (Linear programming)	$\min_x f^T x \quad \text{such that}$ $A \cdot x \le b, \quad Aeq \cdot x = beq, \quad l \le x \le u$	linprog
Τετραγωνικός προγραμματισμός (Quadratic programming)	$\min_x \frac{1}{2} x^T H x + f^T x \quad \text{such that}$ $A \cdot x \le b, \quad Aeq \cdot x = beq, \quad l \le x \le u$	quadprog
Ελαχιστοποίηση με περιορισμούς (Constrained minimization)	$\min_x f(x) \quad \text{such that}$ $c(x) \le 0, \quad ceq(x) = 0$ $A \cdot x \le b, \quad Aeq \cdot x = beq, \quad l \le x \le u$	fmincon
Μέθοδος επίτευξης στόχου (Goal attainment)- Χρησιμοποιείται συνήθως για πολυδιάστατη βελτιστοποίηση (πολλές αντικειμενικές συναρτήσεις)	$\min_{x,\gamma} \gamma \quad \text{such that}$ $F(x) - w\gamma \le \text{goal}$ $c(x) \le 0, \quad ceq(x) = 0$ $A \cdot x \le b, \quad Aeq \cdot x = beq, \quad l \le x \le u$	fgoalattain
Μέθοδος minimax	$\min_x \max_{\{F_i\}} \{F_i(x)\} \quad \text{such that}$ $c(x) \le 0, \quad ceq(x) = 0$ $A \cdot x \le b, \quad Aeq \cdot x = beq, \quad l \le x \le u$	fminimax
Ελαχιστοποίηση ημι-άπειρων προβλημάτων (Semi-infinite minimization)	$\min_x f(x) \quad \text{such that}$ $K(x, w) \le 0 \text{ for all } w$ $c(x) \le 0, \quad ceq(x) = 0$ $A \cdot x \le b, \quad Aeq \cdot x = beq, \quad l \le x \le u$	fseminf

Πίνακας 11.2.2 Επίλυση εξισώσεων

Επίλυση εξισώσεων
(equation solving)

Τύπος	Σημειώσεις	Συναρτήσεις
Γραμμικές εξισώσεις (Linear equations)	$C \cdot x = d$, n equations, n variables	\ (slash)
Μη γραμμική εξίσωση μιας μεταβλητής (Nonlinear equation of one variable)	$f(a) = 0$	fzero
Μη γραμμικές εξισώσεις (Nonlinear equations)	$F(x) = 0$, n equations, n variables	fsolve

Πίνακας 11.2.3 Προβλήματα ελάχιστων τετραγώνων

Ελάχιστα τετράγωνα
(Least-Squares)

Τύπος	Σημειώσεις	Χρησιμοποιούμενες συναρτήσεις
Γραμμικά ελάχιστα τετράγωνα (Linear-Least-squares)	$\min_x \|C \cdot x - d\|_2^2$, m equations, n variables	\ (slash)
Μη αρνητικά γραμμικά ελάχιστα τετράγωνα (Nonnegative Linear-Least-Squares)	$\min_x \|C \cdot x - d\|_2^2$ such that $x \geq 0$	lsqnonneg
Γραμμικά ελάχιστα τετράγωνα με περιορισμούς (Constrained Linear-Least-Squares)	$\min_x \|C \cdot x - d\|_2^2$ such that $A \cdot x \leq b, \quad Aeq \cdot x = beq, \quad l \leq x \leq u$	lsqlin
Μη γραμμικά ελάχιστα τετράγωνα (Nonlinear Least-Squares)	$\min_x \frac{1}{2}\|F(x)\|_2^2 = \frac{1}{2}\sum_i F_i(x)^2$ such that $l \leq x \leq u$	lsqnonlin
Μη γραμμική ανάλυση προσαρμογής δεδομένων (Nonlinear Curve fitting)	$\min_x \frac{1}{2}\|F(x, xdata) - ydata\|_2^2$ such that $l \leq x \leq u$	lsqcurvefit

11.3 Χρήση των συναρτήσεων του «OPTIMIZATION TOOLBOX»

1. Οι περισσότεροι από τους αλγορίθμους βελτιστοποίησης απαιτούν τη δημιουργία αρχείου (function m-file) στο οποίο θα βρίσκεται η αντικειμενική συνάρτηση. Σε περίπτωση που έχουμε περιορισμούς, αυτοί αναγράφονται σε ξεχωριστό αρχείο.
2. Στην περίπτωση που μας ενδιαφέρει η μεγιστοποίηση τότε απλώς εφαρμόζουμε τους αλγόριθμους ελαχιστοποίησης για την αντικειμενική συνάρτηση –f.

11.4 Χρησιμοποιούμενες μέθοδοι βελτιστοποίησης για κάθε τύπο προβλήματος

Οι αλγόριθμοι χωρίζονται σε μέσης κλίμακας (MEDIUM-SCALE) και μεγάλης κλίμακας (LARGE-SCALE). Η διάκριση αυτή στηρίζεται συνήθως στο πλήθος των ανεξάρτητων μεταβλητών του προβλήματος βελτιστοποίησης. Όμως, στο εγχειρίδιο του matlab δεν διευκρινίζεται το όριο μεταξύ των δύο κατηγοριών.
Στους Πίνακες 11.4.1 και 11.4.2 παρουσιάζονται οι αλγόριθμοι βελτιστοποίησης που χρησιμοποιούνται ανάλογα με το πρόβλημα.

Πίνακας 11.4.1 Προβλήματα Μέσης Κλίμακας

Ελαχιστοποίηση χωρίς περιορισμούς (unconstrained minimization)	Nelder-Mead simplex search method
	BGFS (Broyden Fletcher Goldfarb Shanno) quasi-Newton method
Ελαχιστοποίηση με περιορισμούς (constrained minimization)	Sequential quadratic programming (SQP) method
Προβλήματα minimax	SQP method
Προβλήματα επίτευξης στόχου (goal attainment)	SQP method
Μη γραμμικά ελάχιστα τετράγωνα (Nonlinear least-squares)	Gauss-Newton method
	Levenberger-Marquardt method
Προβλήματα επίλυσης μη γραμμικών εξισώσεων	Trust-region dogleg algorithm

Πίνακας 11.4.2 Προβλήματα Μεγάλης Κλίμακας

Προβλήματα με περιορισμούς και όρια στις μεταβλητές (Bound constrained problems)	Newton methods
Προβλήματα ισοτικών περιορισμών (equality constrained problems)	Projective preconditioned conjugate gradient iteration
Γραμμικός προγραμματισμός (Linear programming)	Mehrotra predictor-corrector algorithm

Σημείωση: Βλέπουμε ότι ο χρήστης δεν έχει τη δυνατότητα να επιλέξει τη μέθοδο βελτιστοποίησης (για το πρόβλημα που μελετούμε εμείς). Σε άλλες περιπτώσεις όπως στα μη γραμμικά ελάχιστα τετράγωνα, στη συνάρτηση lsqnonlin μπορούμε να επιλέξουμε τη μέθοδο επίλυσης του προβλήματος. Έχουμε να διαλέξουμε μεταξύ των μεθόδων Gauss-Newton, Levenberger-Marquardt και Trust-region dogleg.

12. Η ΣΥΝΑΡΤΗΣΗ FMINCON

12.1 Σύνταξη της συνάρτησης fmincon

Η σύνταξη της συνάρτησης fmincon είναι γενικά η εξής:

[x,fval,exitflag,output,lambda,grad,hessian]=

fmincon(fun,x0,A,b,Aeq,Beq,lb,ub,nonlcon,options,P1,P2…)

Η σύνταξη της fmincon γίνεται πάντα με τον συγκεκριμένο τρόπο. Αυτό σημαίνει ότι η δήλωση των ορισμάτων εισόδου και εξόδου πρέπει να γίνεται με την συγκεκριμένη σειρά που δίνονται στην παραπάνω έκφραση.

Στο ***δεξί μέρος*** της έκφρασης που είναι κλεισμένο σε παρενθέσεις ( ) βρίσκονται πάντα τα ***ορίσματα εισόδου*** και τα οποία έχουν παρουσιαστεί αναλυτικά στον πίνακα 11.1.1

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Αν δεν θέλουμε να εισάγουμε ορίσματα εισόδου που βρίσκονται δεξιότερα του τελευταίου δεδομένου ορίσματος στη σύνταξη της εντολής, απλώς τα παραλείπουμε.
- Κατά περίπτωση, ανάλογα με το πρόβλημα, μπορεί για παράδειγμα να μην έχουμε γραμμικούς ανισοτικούς περιορισμούς. Σε αυτή την περίπτωση στη σύνταξη της εντολής θέτουμε στις θέσεις των A, b έναν κενό πίνακα (που παριστάνεται από τις αγκύλες [ ]). Γενικά θα πρέπει να αντικαταστήσουμε με τον κενό πίνακα τα ορίσματα που δεν δίνονται.
- Παραδείγματα:

…=fmincon(fun,x0,A,b) Με αυτή τη σύνταξη ζητείται να λυθεί ένα πρόβλημα βελτιστοποίησης της αντικ. συνάρτησης fun που υπόκειται στους γραμμικούς ανισοτικούς περιορισμούς της μορφής A*x<=b με αρχικό σημείο εκκίνησης το x0. Παρατηρούμε ότι δεν δίνονται πληροφορίες για τα επόμενα ορίσματα δηλαδή τα Aeq, beq, lb, ub, nonlcon, options, P1,P2

…=fmincon(fun,x0,[],[],[],[],[],[],nonlcon) Με αυτή τη σύνταξη ζητείται να λυθεί ένα πρόβλημα βελτιστοποίησης της αντικ. συνάρτησης fun που υπόκειται στους μη γραμμικούς περιορισμούς που δίνονται στο nonlcon, με αρχικό σημείο εκκίνησης το x0.

Βλέπουμε ότι δεν δίνονται πληροφορίες ούτε για γραμμικούς ανισοτικούς ή ισοτικούς περιορισμούς (A,b,Aeq,beq) ούτε για ανώτερα ή κατώτερα όρια στη μεταβλητή σχεδιασμού x (lb,ub). Όλα αυτά τα σχετικά ορίσματα αντικαθίστανται από αγκύλες.

Στο ***αριστερό μέρος*** της έκφρασης που είναι κλεισμένο σε αγκύλες [ ] βρίσκονται πάντα τα ***ορίσματα εξόδου*** και τα οποία έχουν παρουσιαστεί αναλυτικά στον Πίνακα 11.1.2. Μετά το πέρας του αλγορίθμου παρουσιάζονται τα αποτελέσματα σε πινακοποιημένη μορφή.

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Αν θέλουμε να πάρουμε ως έξοδο μόνο το βέλτιστο σημείο x τότε δεν χρειάζονται αγκύλες στη σύνταξη, (δηλαδή x=fmincon(…)).
- Αν δεν θέλουμε να έχουμε στην έξοδο κάποια ορίσματα που βρίσκονται δεξιότερα του τελευταίου επιθυμητού ορίσματος, απλώς παραλείπονται μέσα στις αγκύλες.
- Παραδείγματα:

x=fmincon(…)	Εμφάνιση μόνο του βέλτιστου σημείου x.
[x,fval,exitflag,output,lambda]=fmincon(…)	Εμφάνιση των x, fval, exitflag, output, lambda.

Στον πίνακα 12.1 παρατίθενται κάποια πιο ειδικά στοιχεία σχετικά με ορίσματα εισόδου της fmincon.

Πίνακας 12.1 Ορίσματα εισόδου της fmincon

fun	Είναι ένα function m-file το οποίο περιέχει την αντικειμενική συνάρτηση που θα βελτιστοποιηθεί.
nonlcon	Είναι ένα function m-file στο οποίο δηλώνονται οι μη γραμμικοί ανισοτικοί περιορισμοί c(x)<=0 και οι μη γραμμικοί ισοτικοί περιορισμοί ceq(x)=0. δηλαδή η συνάρτηση των περιορισμών έχει 2 ορίσματα εξόδου τα c(x) και ceq(x). **Παράδειγμα:** Έστω ότι έχουμε μια συνάρτηση $f(x_1, x_2)$ η οποία υπόκειται στους μη γραμμικούς περιορισμούς $x_1^2 + x_2 = 0$, $-x_1 \cdot x_2 - 10 \le 0$, $x_2^2 - x_1 - 3 \ge 0$. Τότε η δήλωση θα γίνει ως εξής: *function [c,ceq]=constraints(x1,x2);* *c=[-x1*x2-10; -x2^2+x1+3];* *ceq= x1^2+x2-1;*

	• Στη συνάρτηση c οι περιορισμοί γράφτηκαν σε πινακοποιημένη μορφή σε ένα ***πίνακα στήλη.*** • Επίσης παρατηρούμε ότι στη δεύτερη ανίσωση έχουν αλλάξει τα πρόσημα. Αυτό γίνεται διότι πάντα οι ανισοτικοί περιορισμοί **πρέπει να βρίσκονται στη μορφή c(x)<=0**. Αν η ανίσωση βρίσκεται στη μορφή c(x)>=0 πρέπει να τη φέρουμε στη μορφή –c(x)<=0

Στον πίνακα 12.2 παρατίθενται κάποια πιο ειδικά στοιχεία σχετικά με ορίσματα εξόδου της fmincon.

Πίνακας 12.2 Ορίσματα εξόδου της fmincon

exitflag	Περιγράφει την κατάσταση εξόδου από τον αλγόριθμο	
	>0	Επιτεύχθηκε η σύγκλιση στο βέλτιστο x
	0	Ο μέγιστος αριθμός των επαναλήψεων ξεπεράστηκε
	<0	Η σύγκλιση δεν επετεύχθη στο βέλτιστο x
lambda	Δομή που περιέχει τους πολλαπλασιαστές Langrange στη βέλτιστη λύση x. Τα πεδία της δομής αυτής είναι:	
	lower	Κατώτερα όρια, lb
	upper	Ανώτερα όρια, ub
	ineqlin	Γραμμικές ανισοϊσότητες
	eqlin	Γραμμικές ισότητες
	ineqnonlin	Μη γραμμικές ανισοϊσότητες
	eqnonlin	Μη γραμμικές ισότητες
output	Δομή που περιέχει πληροφορίες για τον αλγόριθμο βελτιστοποίησης	
	iterations	Ο αριθμός των επαναλήψεων που έγιναν
	funcCount	Ο αριθμός των υπολογισμών της αντικειμενικής συνάρτησης
	algorithm	Ο αλγόριθμος που χρησιμοποιήθηκε
	cgiterations	Ο αριθμός των επαναλήψεων PCG (Preconditioned conjugate gradient), μόνο για αλγορίθμους προβλημάτων μεγάλης κλίμακας
	stepsize	Το τελικό βήμα, (μόνο για αλγορίθμους προβλημάτων μέσης κλίμακας)
	firstorderopt	Μέτρο ικανοποίησης των συνθηκών βελτίστου πρώτης τάξης (first order optimality).

12.2 Περιορισμοί στη χρήση της fmincon

Η συνάρτηση η οποία πρόκειται να βελτιστοποιηθεί και οι περιορισμοί, πρέπει να είναι συνεχείς. Η fmincon ίσως να μπορεί να δώσει μόνο τοπικές λύσεις.

Όταν το πρόβλημα είναι αδύνατο, η fmincon προσπαθεί να ελαχιστοποιήσει τη μέγιστη τιμή των περιορισμών.

Η αντικειμενική συνάρτηση και οι περιορισμοί, πρέπει να δέχονται πραγματικές τιμές, που σημαίνει ότι δεν επιτρέπεται να δίνουν μιγαδικές τιμές.

12.3 Χρησιμοποιούμενες μέθοδοι επίλυσης από την fmincon

- **Προβλήματα Μέσης κλίμακας:**

  Χρησιμοποιείται η μέθοδος **SQP (sequential quadratic programming)**. Σε αυτή τη μέθοδο, σε κάθε επανάληψη, λύνεται ένα υποπρόβλημα τετραγωνικού προγραμματισμού (QP subproblem). Ο Εσσιανός πίνακας και ο Langrange εκτιμώνται σε κάθε επανάληψη χρησιμοποιώντας τη μέθοδο **BFGS.**

- **Προβλήματα Μεγάλης κλίμακας**

  Βασίζεται στην μέθοδο **interior-reflective Newton**. Σε κάθε επανάληψη, επιλύεται ένα μεγάλο γραμμικό σύστημα με χρήση της μεθόδου **PCG.**

13. ΟΙ ΕΠΙΛΟΓΕΣ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ FMINCON

Πριν γίνει η σύνταξη της fmincon, προκειμένου να ξεκινήσει ο αλγόριθμος βελτιστοποίησης είναι αναγκαίο να ορίσουμε κάποιες παραμέτρους του αλγορίθμου. Το πρώτο βήμα, αφορά στον καθορισμό του τύπου του προβλήματος, έτσι ώστε να χρησιμοποιηθεί η πιο κατάλληλη μέθοδος βελτιστοποίησης. Αν το πρόβλημα είναι μεγάλης κλίμακας (Large scale), τότε αυτό πρέπει να διευκρινιστεί, διότι αν όχι, τότε ο αλγόριθμος θα χρησιμοποιήσει μεθόδους βελτιστοποίησης μέσης κλίμακας (Medium scale) που είναι και οι προεπιλεγμένες (default), με συνέπεια πολλές φορές, τον πρόωρο τερματισμό του αλγορίθμου, ή τα λάθος αποτελέσματα.
Ο καθορισμός των επιλογών γίνεται χρησιμοποιώντας τη συνάρτηση optimset ως εξής:

>>options=optimset(' Largescale ', 'on ')

Ο πίνακας 13.1 παρουσιάζει τις παραμέτρους της συνάρτησης fmincon οι οποίες χρησιμοποιούνται τόσο για προβλήματα μέσης, όσο και για προβλήματα μεγάλης κλίμακας.

Πίνακας 13.1 Λίστα επιλογών της fmincon

Επιλογή	Πληροφορίες	Παράδειγμα
Diagnostics	Εκτύπωση διαγνωστικών πληροφοριών για την αντικειμενική συνάρτηση	
Display	Όταν είναι στο 'off' δεν δείχνει καθόλου τα αποτελέσματα του αλγορίθμου. Όταν είναι στο 'iter' δείχνει τα ενδιάμεσα αποτελέσματα σε κάθε βήμα του αλγορίθμου. Στο 'final' δείχνει μόνο το τελικό αποτέλεσμα.	Options=optimset('Display','iter')
GradObj	Όταν δίδεται το διάνυσμα κλίσης (gradient) της αντικειμενικής συνάρτησης από τον χρήστη τότε πρέπει να δηλωθεί στο options. Το διάνυσμα κλίσης πρέπει να το παρέχει **υποχρεωτικά** ο χρήστης όταν επιλύει προβλήματα μεγάλης κλίμακας. Είναι **προαιρετικό** για προβλήματα μέσης κλίμακας. (βλέπε τα παραδείγματα στη συνέχεια). **Σημείωση:** Όταν το διάνυσμα κλίσης δίνεται από τον χρήστη -όπου είναι δυνατό-, προκύπτουν ευεργετικά αποτελέσματα για τον αλγόριθμό, όπως, ακρίβεια στους υπολογισμούς και μικρότερο σφάλμα, που προκύπτει λόγω προσέγγισης με τη μέθοδο των πεπερασμένων διαφορών.	Options=optimset('GradObj','on')
MaxFunEvals	Ο μέγιστος αριθμός των υπολογισμών της αντικειμενικής συνάρτησης	Options=optimset('MaxFunEvals',100)
MaxIter	Ο μέγιστος αριθμός των επιτρεπόμενων	Options=optimset('MaxIter',50)

	επαναλήψεων	
TolFun	Ανοχή (tolerance) για τον τερματισμό του αλγορίθμου όσον αφορά την τιμή της αντικειμενικής συνάρτησης	Options=optimset('TolFun',1e-3)
TolCon	Ανοχή για τον τερματισμό του αλγορίθμου όσον αφορά την παραβίαση των περιορισμών (constraints violation).	Options=optimset('TolCon',1e-3)
TolX	Ανοχή για τον τερματισμό του αλγορίθμου όσον αφορά στην τιμή του x.	Options=optimset('TolX',1e-5)

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Όταν είναι επιθυμητή η εισαγωγή πολλαπλών παραμέτρων, τότε αυτές πρέπει να τις διαχωρίσουμε με κόμμα (,). Για παράδειγμα:

>>Options=optimset('Largescale','on','display','iter','tolX',1e-5)

- Όταν είναι επιθυμητή η εισαγωγή πολλαπλών παραμέτρων και δεν είναι πρακτικό να γραφούν σε μία συνεχόμενη γραμμή, τότε είναι δυνατή η συνέχεια σε άλλη γραμμή αν στην προηγούμενη τοποθετηθούν αποσιωπητικά (…). Για παράδειγμα:

>>Options=optimset('Largescale','on','display','iter','tolX',1e-5,…,
'TolFun',2e-3,'GradObj','on')

14. ΠΑΡΑΔΕΙΓΜΑΤΑ ΣΤΗ ΧΡΗΣΗ ΤΗΣ FMINCON

14.1 Παράδειγμα με μη γραμμικούς περιορισμούς

Θα ξεκινήσουμε το παράδειγμα, αναφέροντας για μία ακόμη φορά, ότι όταν η αντικειμενική συνάρτηση έχει μη γραμμικούς περιορισμούς, τότε δηλώνουμε σε ξεχωριστό αρχείο (m-file) την αντικειμενική συνάρτηση και σε άλλο αρχείο τους περιορισμούς. **Αν είχε γραμμικούς περιορισμούς, τότε θα μπορούσαμε να τους γράψουμε στο «command prompt» ακριβώς πριν την σύνταξη της fmincon.**
Το πρόβλημα εδώ, είναι να ελαχιστοποιήσουμε την συνάρτηση δύο μεταβλητών:

$$\underset{x}{\text{minimize}}\ f(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

η οποία υπόκειται στους παρακάτω μη γραμμικούς περιορισμούς:

$$x_1x_2 - x_1 - x_2 \le -1.5$$
$$x_1x_2 \ge -10$$

Όπως αναφέρθηκε στην ενότητα 12.1, είναι αναγκαίο οι ανισοτικοί περιορισμοί να γράφονται πάντα στην μορφή **c(x)<=0**. Γι' αυτό, μετατρέπουμε τις ανισώσεις, φέρνοντας τις στην κατάλληλη μορφή.

$$x_1x_2 - x_1 - x_2 + 1.5 \le 0$$
$$-x_1x_2 - 10 \le 0$$

Τώρα συντάσσουμε ένα καινούριο αρχείο (m-file) για την αντικειμενική συνάρτηση. Αυτό το αρχείο θα περιέχει μόνο την αντικειμενική συνάρτηση. Το ονομάζουμε objfun.m

```
function f = objfun(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
```

Το επόμενο βήμα είναι να δηλώσουμε σε ένα ξεχωριστό αρχείο τους μη γραμμικούς περιορισμούς. Το ονομάζουμε confun.m. Αυτό, περιέχει τις παρακάτω εντολές:

```
function [c, ceq] = confun(x)
% Nonlinear inequality constraints
c = [1.5 + x(1)*x(2) - x(1) - x(2);
     -x(1)*x(2) - 10];
% Nonlinear equality constraints
ceq = [];
```

ΠΑΡΑΤΗΡΗΣΗ

- Επειδή δεν υπάρχουν ισοτικοί περιορισμοί, πρέπει το όρισμα ceq να δηλωθεί ίσο με τον κενό πίνακα, (δηλαδή ceq=[ ]).

Έπειτα από αυτά, γράφονται οι παρακάτω εντολές στο «command window».

```
x0 = [-1,1];     % Make a starting guess at the solution
options = optimset('LargeScale','off');
[x, fval] = ...
fmincon(@objfun,x0,[],[],[],[],[],[],@confun,options)
```

Στην **1η γραμμή** θέτουμε το αρχικό σημείο, δηλαδή τις αρχικές τιμές της μεταβλητής x. Η τιμή -1 αντιστοιχεί στο x(1)=x1 και η τιμή 1 αντιστοιχεί στο x(2)=x2.

Στη **2η γραμμή**, γράφουμε τις επιλογές (options) του αλγορίθμου. Εδώ δηλώνουμε ότι θέλουμε το πρόβλημά μας να επιλυθεί με μεθόδους μέσης κλίμακας, και γι' αυτό θέτουμε 'Largescale','off'.

Στην **3η γραμμή** συντάσσεται η εντολη fmincon. Σαν ορίσματα εξόδου θέλουμε μόνο την βέλτιστη τιμή του x και την τιμή της αντικειμενικής συνάρτησης για το βέλτιστο x. Επειδή στο πρόβλημά μας δεν έχουμε ούτε γραμμικές σχέσεις (ανισοϊσότητες, ισότητες) ούτε όρια για τις μεταβλητές (lb,ub) τα ορίσματα αυτά τίθενται ίσα με τον κενό πίνακα [ ], δηλαδή, A=b=Aeq=beq=lb=ub=[ ].

Πατώντας enter και έπειτα από 38 επαναλήψεις παίρνουμε τα αποτελέσματα:

```
x =
   -9.5474   1.0474
fval =
    0.0236
```

Πληκτρολογώντας στο «command prompt» την παρακάτω εντολή, μπορούμε να πάρουμε τις τιμές που έχουν οι όροι των περιορισμών για την βέλτιστη τιμή του x που υπολογίστηκε.

```
[c,ceq] = confun(x)
c=
    1.0e-14 *
    0.1110
    -0.1776

 ceq =
      []
```

14.2 Παράδειγμα με όρια στις τιμές των μεταβλητών:

Η εντολή x=fmincon(@objfun,x0,[ ],[ ],[ ],[ ],lb,ub,@confun,options) περιορίζει την τιμή του x ώστε να βρίσκεται μεταξύ των τιμών που δίνονται στα ub (upper boundary) και lb (lower boundary). Έστω η εξής συνάρτηση:

$$\min f(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

η οποία υπόκειται στους περιορισμούς:

$$x_1x_2 - x_1 - x_2 \le -1.5$$
$$x_1x_2 \ge -10$$

Έστω ότι πρέπει να περιοριστεί το x ώστε να είναι μεγαλύτερο από το μηδέν, δηλαδή ($x_1 \ge 0, x_2 \ge 0$), τότε πληκτρολογούμε στο «command window»:

```
x0=[-1,1];  %σημείο εκκίνησης
lb=[0,0];   %κατώτερο όριο για το διάνυσμα χ
ub=[ ];     %ανώτερο όριο για το διάνυσμα χ
options=optimset('Largescale','off');
[x,fval]= fmincon(@objfun,x0,[ ],[ ],[ ],[ ],lb,ub,@confun,options);
```

ΠΑΡΑΤΗΡΗΣΗ:

- ✓ Προτείνεται, όταν έχουμε μεταβλητές οι οποίες δεν περιορίζονται προς τα πάνω ή προς τα κάτω, τότε είναι καλό να βάζουμε **–inf** στο lb για μεταβλητές που δεν περιορίζονται προς τα κάτω και **inf** στο ub για μεταβλητές που δεν περιορίζονται προς τα πάνω.
  Έστω ότι έχουμε τα όρια $x_1 \le 10, x_2 \ge 0$ (το x_1 δεν έχει όριο προς τα κάτω και το x_2 δεν έχει όριο προς τα πάνω). Έτσι στο matlab μπορούμε να δηλώσουμε τα όρια ως εξής:

  ```
  lb=[-inf,0];
  ub=[10,inf];
  ```

  Δηλαδή, –inf<x1<10 και 0<x2<inf

  Σημείωση: Στο εγχειρίδιο του «optimization toolbox», αναφέρεται ότι η χρήση των inf και –inf στα όρια των μεταβλητών δίνει **καλύτερα αριθμητικά αποτελέσματα** από αυτά που θα παίρναμε αν χρησιμοποιούσαμε πολύ μεγάλους θετικούς ή αρνητικούς αριθμούς προκειμένου να δείξουμε την απουσία του ορίου.

14.3 Παράδειγμα με μη γραμμικούς περιορισμούς και εισαγωγή των διανυσμάτων κλίσης.

Κανονικά σε αλγορίθμους μέσης κλίμακας, τα διανύσματα κλίσεως υπολογίζονται με μεθόδους πεπερασμένων διαφορών (finite-difference approximation). Με αυτή τη διαδικασία, πολλές φορές υπεισέρχονται λάθη στους υπολογισμούς, διότι συστηματικά αλλάζουν οι τιμές των μεταβλητών, προκειμένου να υπολογιστεί η αντικειμενική συνάρτηση και οι μερικές παράγωγοι τόσο αυτής, όσο και των περιορισμών. Εναλλακτικά, ο χρήστης μπορεί να παρέχει στην fmincon τα διανύσματα κλίσης τόσο της αντικειμενικής συνάρτησης όσο και των περιορισμών (εφόσον είναι εύκολος ο υπολογισμός) ώστε να υπολογιστούν αναλυτικά και όχι αριθμητικά. Έτσι το πρόβλημα λύνεται με περισσότερη ακρίβεια και αποτελεσματικότητα. Έστω ότι το πρόβλημα βελτιστοποίησης είναι:

$$\min f(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$$

με τους εξής περιορισμούς:

$$x_1x_2 - x_1 - x_2 \le -1.5$$
$$x_1x_2 \ge -10$$

ΒΗΜΑ 1

Κατασκευάζεται ένα function m-file, όπου δηλώνεται η αντικειμενική συνάρτηση και το διάνυσμα κλίσης. Δηλώνονται 2 συναρτήσεις. Η f αντιστοιχεί στην αντικειμενική συνάρτηση και η G στο διάνυσμα κλίσης της f. Η μεταβλητή t απλώς περιέχει ένα κομμάτι της έκφρασης του $\partial f / \partial x_1$,διότι είναι αρκετά μεγάλη για να χωρέσει σε μία γραμμή. Έτσι έχουμε:

```
function [f,G] = objfungrad(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
% Gradient of the objective function
t = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
G = [ t + exp(x(1)) * (8*x(1) + 4*x(2)),
    exp(x(1))*(4*x(1)+4*x(2)+2)];
```

Η G περιέχει σε πινακοποιημένη μορφή τις μερικές παραγώγους της αντικειμενικής συνάρτησης f ως προς κάθε στοιχείο του x:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1) + e^{x_1}(8x_1 + 4x_2) \\ \\ e^{x_1}(4x_1 + 4x_2 + 2) \end{bmatrix}$$

ΒΗΜΑ 2

Κατασκευάζεται ένα function m-file για τους περιορισμούς και τα διανύσματα κλίσης των περιορισμών. Πλέον, στη δήλωση των περιορισμών έχουμε τέσσερα ορίσματα. Το **c** για της μη γραμμικές ανισότητες, το **ceq** για τις μη γραμμικές ισότητες, το **DC** για το διάνυσμα κλίσης των μη γραμμικών ανισοτήτων και το **DCeq** για το διάνυσμα κλίσης των μη γραμμικών ισοτήτων. Στο συγκεκριμένο πρόβλημα, επειδή δεν υπάρχουν μη γραμμικές ισότητες, άρα ούτε και τα διανύσματα κλίσης τους, θέτουμε ceq=[ ] , DCeq=[ ]. Έτσι έχουμε:

```
function [c,ceq,DC,DCeq] = confungrad(x)
c(1) = 1.5 + x(1) * x(2) - x(1) - x(2);
c(2) = -x(1) * x(2)-10;
% Gradient of the constraints
DC= [x(2)-1, -x(2);
      x(1)-1, -x(1)];
% No nonlinear equality constraints
ceq=[];
DCeq = [ ];
```

Η DC περιέχει σε πινακοποιημένη μορφή τις μερικές παραγώγους για την κάθε ανισότητα ως προς κάθε στοιχείο του x:

$$\begin{bmatrix} \frac{\partial c_1}{\partial x_1} & \frac{\partial c_2}{\partial x_1} \\ \frac{\partial c_1}{\partial x_2} & \frac{\partial c_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} x_2 - 1 & -x_2 \\ x_1 - 1 & -x_1 \end{bmatrix}$$

Εφόσον παρέχονται τα διανύσματα κλίσης, τόσο της αντικειμενικής συνάρτησης, όσο και των περιορισμών, **η fmincon πρέπει να ενημερωθεί για την παρουσία τους**. Αυτό γίνεται μέσω της εντολής optimset. Πληκτρολογούμε:

>>Options=optimset('GradObj' , 'on' , 'GradConstr' , 'on');

Όταν δίνονται τα αναλυτικά διανύσματα κλίσης, μπορούν να συγκριθούν οι τιμές που δίνουν αυτά, με τις τιμές που υπολογίζονται από τις μεθόδους πεπερασμένων διαφορών (finite-difference). Αυτή η δυνατότητα είναι χρήσιμη στην ανακάλυψη λαθών τόσο στη σύνταξη της αντικειμενικής συνάρτησης, όσο και στη σύνταξη των δοθέντων διανυσμάτων κλίσης. Για να γίνει αυτή η σύγκριση, φτάνει να δηλωθεί στις επιλογές:

>>Options=optimset('DerivativeCheck' , 'on');

Στον πρώτο κύκλο της βελτιστοποίησης υπολογίζονται τα αναλυτικά διανύσματα κλίσης και εκείνα που προκύπτουν με τη μέθοδο των πεπερασμένων διαφορών. Αν τα αποτελέσματά τους δεν είναι συγκρίσιμα, με κάποια δοθείσα ανοχή, τότε παρουσιάζεται ένα προειδοποιητικό μήνυμα, που δείχνει τη διαφορά και δίνει τη δυνατότητα στο χρήστη να επιλέξει αν θέλει να τερματιστεί ο αλγόριθμος, ή να συνεχιστεί κανονικά.

15. ΚΛΗΣΗ ΥΠΟΡΟΥΤΙΝΑΣ FORTRAN ΑΠΟ ΤΟ MATLAB

Παρόλο που το matlab είναι ένα αυτόνομο περιβάλλον για προγραμματισμό και διαχείριση δεδομένων, εν τούτοις, είναι πολλές φορές χρήσιμη η επικοινωνία του με εξωτερικά δεδομένα και προγράμματα. Το matlab παρέχει κάποιο «interface» (που ονομάζεται MATLAB API) τόσο στη γλώσσα C όσο και στη γλώσσα FORTRAN.

15.1 Εισαγωγή στα αρχεία MEX

Η επικοινωνία μεταξύ του Matlab και άλλων γλωσσών προγραμματισμού γίνεται με τη δημιουργία αρχείων MEX. Με αυτό τον τρόπο, παρέχεται στο χρήστη η δυνατότητα, να μπορεί να καλεί υπορουτίνες, γραμμένες σε γλώσσα C ή FORTRAN μέσα από το matlab σαν αυτές να ήταν εσωτερικές συναρτήσεις. Τα αρχεία MEX είναι δυναμικά συνδεδεμένες υπορουτίνες (dynamically linked subroutines) που ο μεταφραστής του matlab μπορεί αυτόματα να εκτελέσει.

Ένας πίνακας του matlab στη γλώσσα C ή FORTRAN μπορεί να θεωρηθεί σαν ένα νέο τύπο πίνακα που ονομάζεται mxArray. Η δομή του mxArray αποθηκεύει πληροφορίες για τον τύπο του πίνακα, τη διάστασή του και τα δεδομένα του. Επίσης έχουμε και πιο συγκεκριμένες πληροφορίες όπως για παράδειγμα, τον αριθμό και τα ονόματα των πεδίων από τα οποία αποτελείται ένας πίνακας δομών (structure array).

Τα mex files είναι προγράμματα γραμμένα σε γλώσσα C ή Fortran που χρησιμοποιούν τύπους συναρτήσεων:

1. **mx functions** για την δημιουργία, την πρόσβαση στα δεδομένα και τον χειρισμό των mxArrays. Λέγονται mx functions διότι είναι ρουτίνες που ξεκινούν με το πρόθεμα mx.

2. **mex functions** για διάφορους υπολογισμούς πίσω στο περιβάλλον του MATLAB. Για παράδειγμα η συνάρτηση mexEvalString υπολογίζει την τιμή μιας ακολουθίας χαρακτήρων στο χώρο εργασίας του MATLAB. Λέγονται mex functions διότι είναι ρουτίνες που ξεκινούν με το πρόθεμα mex.

15.2 Εφαρμογές των αρχείων MEX

- Μεγάλα προγράμματα σε C ή FORTRAN μπορούν να κληθούν από το matlab χωρίς να πρέπει να ξαναγραφτούν σαν m-files.
- Υπολογισμοί που δεν εκτελούνται αρκετά γρήγορα (συνήθως βρόχοι for) μπορούν να μετατραπούν σε γλώσσα C ή FORTRAN για μεγαλύτερη αποδοτικότητα.

Τα αρχεία MEX μπορούν να κληθούν ακριβώς όπως τα m-files μέσα από το «command prompt».

Έστω ότι σε ένα φάκελο βρίσκεται το αρχείο conv2.mex, το οποίο πραγματοποιεί διδιάστατη συνέλιξη (2D convolution) μεταξύ πινάκων. Το αρχείο conv2.m είναι ένα m-file που περιέχει βοηθητικό κείμενο και περιέχεται στον ίδιο φάκελο με το conv2.mex. Όταν πληκτρολογήσουμε conv2 στο «command prompt» τότε το conv2.mex εκτελείται πρώτο. Δηλαδή, **τα αρχεία mex έχουν προτεραιότητα** στην εκτέλεση από τα αρχεία .m όταν αρχεία με το ίδιο όνομα υπάρχουν στον ίδιο φάκελο.

15.3 Τύποι δεδομένων στο Matlab

Όλες οι μεταβλητές στο matlab είτε είναι σταθερές, διανύσματα, πίνακες, ακολουθίες χαρακτήρων, πίνακες κελιών, πίνακες δομών είτε αντικείμενα αποθηκεύονται σαν MATLAB ARRAYS. Κάθε mxArray περιέχει μεταξύ άλλων:

- Τον τύπο του πίνακα
- Τις διαστάσεις του
- Τα δεδομένα του
- Αν είναι αριθμητικός, τον τύπο της μεταβλητής (πραγματική ή μιγαδική)
- Αν είναι «sparce», τους δείκτες για τα μη μηδενικά στοιχεία
- Αν είναι πίνακας δομών ή αντικείμενο, τον αριθμό των πεδίων και τα ονόματά τους.

Σημείωση: Ο χρήστης μπορεί να γράψει mex-files στη γλώσσα C, η οποία υποστηρίζει όλους τους τύπους δεδομένων που χρησιμοποιεί το matlab. Αντιθέτως στη γλώσσα FORTRAN μόνο η δημιουργία πινάκων δεδομένων και χαρακτήρων διπλής ακρίβειας (double precision) υποστηρίζεται. Όταν τα αρχεία mex σε γλώσσα C ή FORTRAN μεταφραστούν με κάποιον compiler, το matlab μπορεί να τα χειριστεί σαν m-functions.

15.4 Κατασκευή mex-files

Για την κατασκευή mex files σε C ή FORTRAN είναι απαραίτητη η εγκατάσταση κάποιου compiler.
Το matlab περιλαμβάνει έναν compiler για την γλώσσα C, που ονομάζεται LCC. Όμως, δεν περιλαμβάνει compiler για την FORTRAN, που σημαίνει ότι ο χρήστης πρέπει να εγκαταστήσει έναν compiler συμβατό με το matlab. «Συμβατός», είναι ο compiler ο οποίος (σε πλατφόρμα Microsoft windows) μπορεί να δημιουργήσει 32-bit δυναμικά συνδεδεμένες βιβλιοθήκες δεδομένων (dynamically linked libraries), δηλαδή αρχεία με την επέκταση .dll.

Η mathworks στην ιστοσελίδα της, παρέχει μια λίστα όλων των υποστηριζόμενων compilers από το matlab. (www.mathworks.com) στην ενότητα: support — product support — compatible compilers for matlab 6.5.

Πίνακας 15.4.1 Λίστα υποστηριζόμενων compilers για το Matlab 6.5

Microsoft Windows		**matlab**	**Matlab compiler**	**Matlab COM Builder**	**Matlab Excel Builder**
Compiler	**Version**	**6.5**	**3.0**	**1.0**	**1.1**
Microsoft Visual C/C++	.NET	x	x	x	x
	6.0	x	x	x	x
	5.0	x	x	x	x
Lcc - Win32	2.4.1	x	x		
Borland C++ Builder	6	x	x	x	x
	5	x	x	x	x
	4	x	x	x	x
	3	x	x	x	x
Borland C/C++ Compiler	5.5	x	x		
	5.2	x	x		
	5	x	x		
Compaq Visual Fortran	6.6	x			
	6.1	x			
Digital Visual Fortran	6.0	x			
	5.0	x			
Watcom C/C++	11	x			
	10.6	x			

15.4.1 Ρυθμίσεις του compiler

Πριν ξεκινήσουμε την δημιουργία κάποιου αρχείου mex, πρέπει πρώτα να ρυθμίσουμε τις παραμέτρους του αρχείου mexopts.bat, που αφορά τον compiler, που θα χρησιμοποιηθεί. Για να μπορέσουμε να κάνουμε τις ρυθμίσεις πληκτρολογούμε:

>>mex –setup

στο matlab «command prompt».

Το matlab χρησιμοποιεί τις επεκτάσεις των αρχείων, προκειμένου να καθορίσει ποιόν compiler θα χρησιμοποιήσει. Για παράδειγμα

>>mex test1.f
Κάνει compiling του αρχείου test1.f χρησιμοποιώντας compiler για FORTRAN.

>>mex test1.c
Κάνει compiling του αρχείου test1.c χρησιμοποιώντας compiler για C.

Όταν επιλεγεί κάποιος compiler την πρώτη φορά, δεν χρειάζεται την επόμενη να ξανακάνουμε τη ρύθμιση αυτή.
Αν πληκτρολογήσουμε mex –setup, παρουσιάζεται η λίστα των συμβατών compilers στο σύστημα, και καλούμαστε να επιλέξουμε κάποιον:

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n? n

Select a compiler:
[1] Borland C++Builder version 6.0
[2] Borland C++Builder version 5.0
[3] Borland C++Builder version 4.0
[4] Borland C++Builder version 3.0
[5] Borland C/C++ version 5.02
[6] Borland C/C++ version 5.0
[7] Borland C/C++ (free command line tools) version 5.5
[8] Compaq Visual Fortran version 6.1
[9] Compaq Visual Fortran version 6.6
[10] Digital Visual Fortran version 6.0
[11] Digital Visual Fortran version 5.0
[12] Lcc C version 2.4
[13] Microsoft Visual C/C++ version 7.0
[14] Microsoft Visual C/C++ version 6.0
[15] Microsoft Visual C/C++ version 5.0
[16] WATCOM C/C++ version 11
[17] WATCOM C/C++ version 10.6

[0] None

Compiler:
```

Στην τελευταία γραμμή, ο χρήστης καλείται να πληκτρολογήσει τον αριθμό του επιθυμητού compiler. Αν αυτός δεν μπορεί να εντοπιστεί τότε παρουσιάζεται το παρακάτω μήνυμα:

```
Compiler: 8

The default location for Compaq Visual Fortran compilers is C:\Program Files\Microsoft
Visual Studio,
but that directory does not exist on this machine.

Use C:\Program Files\Microsoft Visual Studio anyway [y]/n? n
```

Please enter the location of your compiler: [C:\Program Files\Microsoft Visual Studio]

Στην τελευταία γραμμή ο χρήστης καλείται να υποδείξει μια νέα τοποθεσία εύρεσης, για τον συγκεκριμένο compiler.

Σημείωση: Στον υπολογιστή του συντάκτη της διπλωματικής εργασίας βρίσκεται εγκατεστημένος ο compiler: Compaq Visual Fortran version 6.6

15.5 Κατασκευάζοντας mex-files στα windows

Το matlab περιέχει πολλά παραδείγματα πάνω στην κατασκευή mex-files. Αυτά βρίσκονται στην εξής τοποθεσία c:\matlab6p5\extern\examples\mex. Εκεί, για παράδειγμα υπάρχει το αρχείο yprimef.f. Αφού ο χρήστης πάει σε αυτό το φάκελο, μπορεί να πληκτρολογήσει:

```
>>mex yprimef.f
```

Αυτή η εντολή καλεί τον compiler και δημιουργείται το αρχείο yprimef.dll. Τώρα, ο χρήστης μπορεί να καλέσει το yprimef, σαν να ήταν function m-file. Στο «command prompt» πληκτρολογούμε:

```
>>yprimef(1, 1:4)
```

και το matlab δίνει την απάντηση

```
ans=
    2.0000   8.96885    4.0000   -1.0947
```

Το matlab δίνει και τη δυνατότητα στο χρηστή, το compiling να γίνει με διάφορες επιλογές που θέλει ο ίδιος. Για περισσότερες πληροφορίες ανατρέξτε στο [10] στην ενότητα custom building mex-files.

15.6 Επίλυση προβλημάτων

Πολλές φορές μπορούν να δημιουργηθούν προβλήματα στη διαδικασία μετάφρασης (compiling) που μπορούν να οφείλονται σε πολλούς παράγοντες:

Προβλήματα λόγω της εγκατάστασης του matlab.

- Πολλές φορές, αν μεταφερθεί ο φάκελος που είναι εγκατεστημένο το matlab σε κάποια άλλη τοποθεσία, τότε, ίσως ο χρήστης πρέπει να ρυθμίσει το αρχείο mex.bat ώστε να του υποδείξει την νέα τοποθεσία του matlab

- Συχνά δημιουργούνται προβλήματα, όταν υπάρχουν κενά στο όνομα του φακέλου που είναι εγκατεστημένο το matlab. Για παράδειγμα αν εγκατασταθεί στην τοποθεσία c:\program files\matlab6p5 έχει παρατηρηθεί δυσκολία στην δημιουργία mex-files για κάποιους compilers της γλώσσας C. Σε αυτή την περίπτωση πρέπει να εγκατασταθεί και πάλι το πρόγραμμα σε νέα τοποθεσία χωρίς κενά.

- Προβλήματα θα υπάρξουν επίσης όταν δεν μπορούν να εντοπιστούν τα αρχεία .dll. Θα πρέπει πάντα τα αρχεία mex να βρίσκονται στον ίδιο φάκελο με τα αρχεία .dll.

- Μερικά πακέτα πρόληψης ιών (antivirus software) μπορούν να προκαλέσουν προβλήματα. Αν ο χρήστης λάβει μήνυμα του τύπου:

  mex.batQ: internal error in sub get_compiler_info( ): don't recognize <string>

  Τότε ίσως πρέπει να απενεργοποιηθεί προσωρινά το "antivirus" και να πληκτρολογηθεί και πάλι η εντολή για το compiling. Όταν όλα γίνουν σωστά μπορούμε και πάλι να ενεργοποιήσουμε το antivirus.

Άλλα προβλήματα.

- Η πιο συχνή περίπτωση λάθους σε γλώσσα C είναι η κατασκευή mex-files με έναν compiler για γλώσσα C που δεν είναι κατά ANSI C συμβατός.

- Πολλές φορές ο χρήστης παίρνει λάθος αποτελέσματα, σε σχέση με τα αναμενόμενα. Αρχικά θα πρέπει να ελεγχθεί αν υπάρχει κάποιο λάθος στη λογική του προγράμματος. Αν όλα είναι σωστά, τότε το πρόγραμμα ίσως διαβάζει από μη-καθορισμένες περιοχές μνήμης. Για παράδειγμα, το να διαβάσει το 11^{ο} στοιχείο ενός διανύσματος 10 στοιχείων, μπορεί να προκαλέσει απρόβλεπτα αποτελέσματα. Άλλη μία πιθανή πηγή λάθους, είναι όταν το πρόγραμμα γράφει δεδομένα στην μνήμη, πάνω σε άλλα, λόγω κακής διαχείρισης της μνήμης. Έτσι κάποια μεταβλητή που χρησιμοποιείται μπορεί να διαγραφεί και την θέση της στην μνήμη να την πάρει κάποια άλλη.

- Ανάλογα με την πλατφόρμα εργασίας (WINDOWS, UNIX, WATCOM) μπορούν να υπάρξουν και διαφορετικά προβλήματα. Για τις επιμέρους περιπτώσεις ο χρήστης είναι καλό να επικοινωνεί με την mathworks (www.mathworks.com).

- Τα προβλήματα που έχουν να κάνουν με τη διαχείριση μνήμης (Από τη βιβλιογραφία, [10] στην ενότητα memory management compatibility issues).

15.7 Τα επιμέρους τμήματα ενός mex-file σε γλώσσα FORTRAN

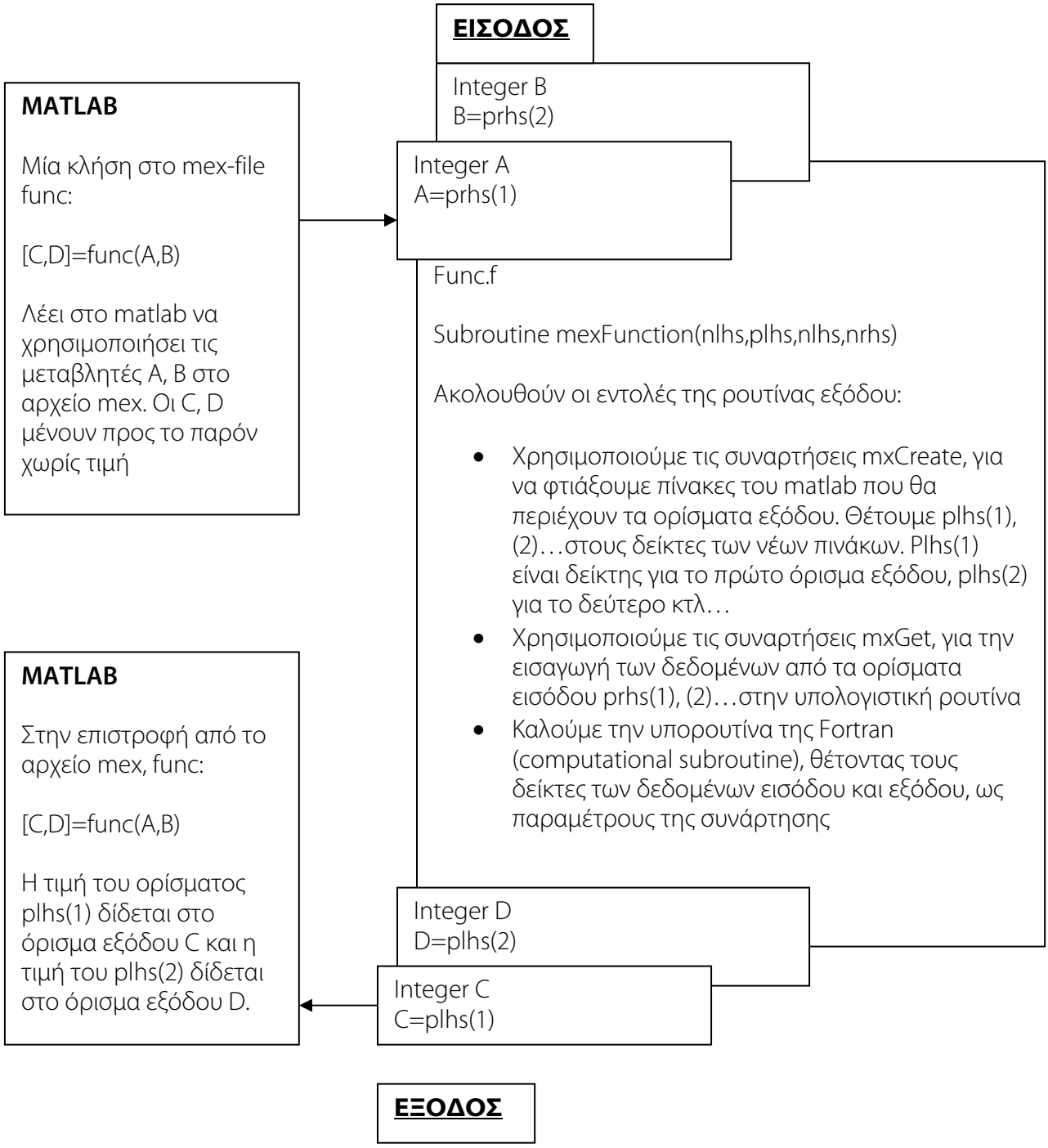
Ο κώδικας ενός mex-file σε γλώσσα FORTRAN αποτελείται από δύο ξεχωριστά τμήματα

- Την **υπολογιστική ρουτίνα** (computational routine) , που περιέχει τον πηγαίο κώδικα για τους υπολογισμούς. Οι υπολογισμοί, μπορεί να είναι αριθμητικοί υπολογισμοί αλλά και είσοδοι και έξοδοι δεδομένων.

- Την **ρουτίνα εξόδου** (gateway routine), που συνδέει την υπολογιστική ρουτίνα με το matlab, μέσω της υπορουτίνας mexFunction και των παραμέτρων της prhs, nrhs, plhs, nlhs.
  prhs είναι ένας πίνακας με τα ορίσματα εισόδου, **nrhs** είναι ο συνολικός αριθμός των ορισμάτων εισόδου, **plhs** είναι ένας πίνακας με τα ορίσματα εξόδου, και **nlhs** είναι ο συνολικός αριθμός των ορισμάτων εξόδου.

Το παρακάτω διάγραμμα είναι πολύ χρήσιμο και δείχνει με πολύ ωραίο τρόπο:

- Πώς εισέρχονται τα δεδομένα εισόδου
- Τι συναρτήσεις χρησιμοποιεί η ρουτίνα εξόδου, και
- Πώς η έξοδος επιστρέφει και πάλι στο matlab.



15.7.1 Η σημασία των δεικτών (pointers)

Το matlab δουλεύει με ένα μοναδικό τύπο δεδομένων, το mxArray. Επειδή στη fortran δεν υπάρχει τρόπος να δημιουργήσουμε ένα νέο τύπο δεδομένων, το matlab χρησιμοποιεί τους δείκτες. Θα λέγαμε πιο απλά, ότι με τον δείκτη η fortan μπορεί να αντιληφθεί και να χρησιμοποιήσει τα δεδομένα ενός mxArray.

Παρατήρηση:

- Για να χρησιμοποιηθούν οι δείκτες σωστά, πρέπει να δηλωθούν ώστε να είναι του κατάλληλου μεγέθους. Για παράδειγμα στην πλατφόρμα DEC Alpha όλοι οι δείκτες πρέπει να δηλωθούν σαν integer*8. Σε κάποιες άλλες πλατφόρμες (Unix, Windows) οι δείκτες πρέπει να δηλώνονται σαν integer*4. Η λανθασμένη δήλωση μεγέθους κάποιου δείκτη, μπορεί να προκαλέσει ανεπιθύμητο τερματισμό του προγράμματος.

15.7.2 Η ρουτίνα εξόδου (Gateway routine)

Η ρουτίνα εξόδου (σε γλώσσα Fortran) πρέπει πάντα να ξεκινά, με τις παρακάτω δηλώσεις:

```
Subroutine mexFunction(nlhs, plhs, nrhs, prhs)
Integer plhs(*), prhs(*)
Integer nlhs, nrhs
```

Στο matlab οι συναρτήσεις έχουν τη γενική μορφή:

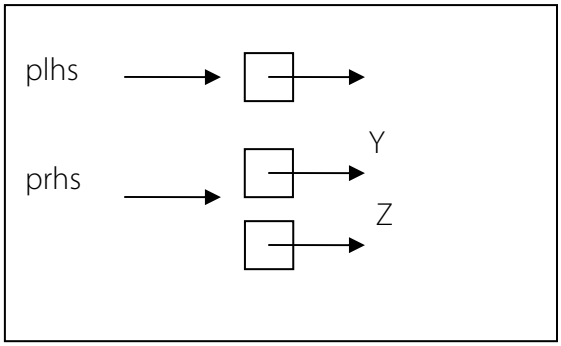
[a, b, c]=fun(d, e, f,…),

όπου τα a, b, c είναι αριστερά ορίσματα, ενώ τα d, e, f είναι δεξιά ορίσματα. Έστω ότι εκτελούμε το αρχείο fun.f (το οποίο όπως είπαμε μπορεί να κληθεί σαν ένα απλό αρχείο function m-file). Γράφουμε:

```
>>x=fun(y, z)
```

Ο μεταφραστής του matlab καλεί την συνάρτηση mexFunction (δηλαδή εκείνη που παρέχει την επικοινωνία μεταξύ matlab-fortran), με τα εξής ορίσματα:

nlhs=1 (ο αριθμός των αριστερών ορισμάτων είναι 1, δηλαδή το X)
nrhs=2 (ο αριθμός των δεξιών ορισμάτων είναι 2, δηλαδή τα Y, Z)



Plhs (δηλαδή ο πίνακας που περιέχει τους δείκτες για τα αριστερά ορίσματα –έξοδος), είναι ένας πίνακας-στοιχείο, όπου το μοναδικό του αυτό στοιχείο είναι ένας κενός ενδείκτης (null pointer), που σημαίνει ότι δεν αντιστοιχεί πουθενά. Αυτό συμβαίνει διότι η έξοδος x δεν δημιουργείται παρά μόνο όταν εκτελεστεί η υπορουτίνα της fortran που περιέχει τους υπολογισμούς.

Prhs (δηλαδή ο πίνακας που περιέχει τους δείκτες για τα δεξιά ορίσματα –είσοδος), είναι ένας πίνακας 1X2, με το πρώτο του στοιχείο να είναι ένας δείκτης σε ένα mxArray που ονομάζεται Y και το δεύτερο στοιχείο του είναι ένας δείκτης σε ένα άλλο mxArray που ονομάζεται Z.

- Η ρουτίνα εξόδου, πρέπει να ελέγχει τα ορίσματα εισόδου και να δίνει μήνυμα λάθους, αν κάποιο παραλείπεται. Η υπορουτίνα **mexErrMsgTxt** κάνει αυτή τη δουλειά. Ο έλεγχος αυτός περιλαμβάνει, έλεγχο του αριθμού, του τύπου και του μεγέθους τόσο των πινάκων εισόδου, όσο και των πινάκων εξόδου.

- Υπάρχουν πολλές συναρτήσεις για τη διαχείριση πινάκων του matlab. Όλες αυτές περιέχουν το πρόθεμα mx, και δίνουν τη δυνατότητα στο χρήστη να τροποποιήσει mxArrays. Για παράδειγμα η συνάρτηση **mxGetPr** παίρνει τις πραγματικές τιμές (real) ενός πίνακα του matlab. Επιπρόσθετες ρουτίνες υπάρχουν για τη μεταφορά δεδομένων μεταξύ matlab και fortran.

- Η ρουτίνα εξόδου, πρέπει να καλεί τις **mxCreateDoubleMatrix, mxCreateSparse**, ή **mxCreateString**, για τη δημιουργία πινάκων matlab του επιθυμητού μεγέθους, στους οποίους θα επιστρέφονται τα αποτελέσματα. Οι τιμές που αποθηκεύονται στους πίνακες αυτούς, πρέπει να αντιστοιχίζονται στα κατάλληλα στοιχεία των ορισμάτων plhs.

- Η ρουτίνα εξόδου, πρέπει να καλεί την υπολογιστική ρουτίνα (computational routine) για να πραγματοποιήσει τους επιθυμητούς υπολογισμούς.

- Όταν ένα αρχείο mex τελειώσει την δουλειά του, επιστρέφει τον έλεγχο πίσω στο matlab. Οι πίνακες που δημιουργήθηκαν από το mex-file και στους οποίους, στην έξοδο (αριστερό όρισμα) δεν αποδόθηκε κάποια τιμή, αυτοί αυτόματα καταστρέφονται.

15.8 Παράδειγμα

Στο παρακάτω παράδειγμα η υπορουτίνα σε fortran παίρνει ένα σταθερό αριθμό και τον διπλασιάζει.

```
C     computational subroutine
      subroutine timestwo(y, x)
      real*4 x, y

      y=2.0 * x
      return
      end

C     the gateway routine
      subroutine mexFunction(nlhs, plhs, nrhs, prhs)

      integer plhs(*), prhs(*)
      integer mxGetPr, mxCreateDoubleMatrix
      integer x_pr, y_pr

      integer nlhs, nrhs
      integer mxGetM,, mxGetN, mxIsNumeric
      integer m, n, size
      real*4 x, y

C     Έλεγχος για τον αριθμό των ορισμάτων
      if (nrhs .ne. 1) then
         call mexErrMsgTxt('One input required')
      elseif(nlhs .ne. 1) then
         call mexErrMsgTxt('One output required')
      endif

C     Παίρνουμε το μέγεθος του πίνακα των δεδομένων εισόδου.
      m=mxGetM(prhs(1))
      n=mxGetN(prhs(1))
      size=m*n

C     Έλεγχος για το αν τα δεδομένα εισόδου είναι αριθμοί
      if (mxIsNumeric(prhs(1)). eq. 0) then
         call mexErrMsgTxt('Input must be a number')
      endif

C     Δημιουργία πίνακα για την έξοδο
      plhs(1)=mxCreateDoubleMatrix(m, n, 0)
      x_pr=mxGetPr(prhs(1))
      y_pr=mxGetPr(plhs(1))
      call mxCopyPtrToReal4(x_pr, x, size)

C     call the computational subroutine
      call timestwo(y, x)

C     «Φόρτωσε» τα δεδομένα στην y_pr, που είναι η έξοδος
C     στο matlab
      call mxCopyReal4ToPtr(y, y_pr, size)

      return
      end
```

Η υπολογιστική ρουτίνα της fortran

Αυτό το γράφουμε πάντα

Δήλωση όλων των mx-functions και των υπολοίπων μεταβλητών που θα χρησιμοποιήσουμε

Με ένα βρόχο if, ελέγχουμε τον απαραίτητο αριθμό των ορισμάτων εισόδου και εξόδου καλώντας την υπορουτίνα mexErrMsgTxt

Παίρνουμε τον αριθμό των γραμμών και των στηλών του ορίσματος εισόδου και τον αποθηκεύουμε στις μεταβλητές

Δημιουργούμε πραγματικό πίνακα (αυτό δείχνει το (0)), μεγέθους m x n για το όρισμα εξόδου

Παίρνουμε τους πραγματικούς δείκτες (real pointers) της εισόδου και της εξόδου

Αντιγράφει τον δείκτη x_pr, σε ένα προσωρινό πίνακα real*4 της fortran με το όνομα x, ώστε να μπορεί να χρησιμοποιηθεί από την υπολογιστική ρουτίνα. Το size υποδεικνύει τον αριθμό των στοιχείων προς αντιγραφή

Κλήση της υπολογιστικής ρουτίνας για τους υπολογισμούς.

Αντιγράφει την τιμή real*4 του πίνακα της fortran y, σε ένα προσωρινό πίνακα του matlab, με το δείκτη y_pr. Το size υποδεικνύει τον αριθμό των στοιχείων προς αντιγραφή

Για να γίνει τώρα η μετάφραση του αρχείου, πληκτρολογούμε στο «command prompt»:

```
>>mex timestwo.f
```

Αυτή η εντολή κατασκευάζει το αρχείο mex που λέγεται timestwo.dll. Μπορούμε τώρα να εκτελέσουμε το αρχείο timestwo σαν να ήταν function m-file.

```
>>x=2;
>>y=timestwo(x)

ans

y=
     4
```

Περισσότερες πληροφορίες σχετικά με τα θέματα αυτά, μπορεί ο χρήστης να βρει στη βιβλιογραφία [10].

16. ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΣΥΜΠΑΡΑΓΩΓΗΣ CGAM ΜΕ ΧΡΗΣΗ ΤΟΥ MATLAB

16.1 Διατύπωση του προβλήματος βελτιστοποίησης ενεργειακού συστήματος

Τα παρακάτω αποτελούν εφαρμογή των όσων αναλύθηκαν έως τώρα. Επιχειρείται η βελτιστοποίηση του ενεργειακού συστήματος CGAM (Παράρτημα Α'). Για τις ανάγκες του προβλήματος κατασκευάζονται 3 αρχεία. Το cgam_objective.m που περιέχει την αντικειμενική συνάρτηση, το cgam_constraints.m που περιέχει τους περιορισμούς, και το cgam_optim alg.m που περιέχει τον αλγόριθμο βελτιστοποίησης.

16.2 Εισαγωγή σταθερών παραμέτρων (θερμοδυναμικές και οικονομικές παράμετροι)

Οι παράμετροι που δίνονται στον Πίνακα Α1 του Παραρτήματος Α' συμπεριλαμβανομένων και των παραμέτρων που προέκυψαν από τη μετατροπή των μεγεθών, αποθηκεύτηκαν όλες σε ένα αρχείο με το όνομα data2.mat.
Όλα τα δοθέντα στοιχεία πληκτρολογήθηκαν στο ««command window»» του matlab, χρησιμοποιώντας στο τέλος κάθε γραμμής την αγγλική άνω τελεία (;). Έτσι, κάθε παράμετρος αποθηκεύτηκε στη μνήμη (δηλαδή στο χώρο εργασίας του matlab). Αφού τελείωσε η πληκτρολόγηση των στοιχείων, επελέγη η διαδρομή **file/save workspace as**… και δόθηκε το όνομα data2.mat.

Για παράδειγμα:
>>t7min=373.15;
>>t8=298.15;
>>t9=485.52;

Εναλλακτικά, η εντολή αποθήκευσης θα μπορούσε να είχε δοθεί και από το ««command prompt»» αν πληκτρολογούσαμε την εντολή:
>>save data2
Με αυτό τον τρόπο, όλες οι μεταβλητές στο παρόν workspace (current workspace) αποθηκεύονται στο αρχείο data2.mat στον παρόντα φάκελο εργασίας (current directory).

Τα περιεχόμενα του αρχείου data2.mat εμφανίζονται αναλυτικά παρακάτω:

%THERMODYNAMIC PARAMETERS
w=30000; %kw
ms=14; %kg/sec
hu=50000; %kJ/kg
cpa=1.004; %kJ/Kg*K
cpg=1.170; %kJ/kg*K
dt8p=15; %K
t8p=470.52; %K διότι t8p=t9-dt8p

h8=106.619; %kJ/kg απο δ.Μολλιερ για p8=20bar και τ8=298.15K η 25 C
h8p=840.793; %kJ/kg απο δ.Μολλιερ για p8p=20bar και t8p=470.52K η 197.37 C
h9=2797.2; %kJ/kg απο δ.Μολλιερ για p9=20bar και t9=485.52K η 212.37 C
dtpmin=0; %K
p1=1.013; %bar
p7=1.013; %bar
p8=20; %bar
p9=20; %bar
ga=1.40; %constant
gg=1.33; %constant
nb=0.98; %constant
u=0.018; %kw/m^2*K
t0=298.15; %K
t1=298.15; %K
t7min=373.15; %K
t8=298.15; %K
t9=485.52; %K
Ra=0.287; %kJ/kg*K
Rg=0.290; %KJ/kg*K
raa=0.95; %constant
rag=0.97; %constant
rb=0.95; %constant
rr=0.95; %constant

%ECONOMIC PARAMETERS
t=28800000; %sec/year Για να είναι η αντικ. συνάρτηση στις ίδιες μονάδες (στο paper δίνεται σε h/year)
cf=0.000004; %$/kJ
fcr=0.182; %(year)^-1
feliniko=1.06; %constant
c11=39.5; %$/(kg/sec)
c12=0.9; %constant
c21=25.6; %$/(kg/sec)
c22=0.995; %constant
c23=0.018; %1/K
c24=26.4; %constant
c31=266.3; %$/(kg/sec)
c32=0.92; %constant
c33=0.036; %1/K
c34=54.4; %constant
c41=2290; %$/m^0.6
c51=3650; %$/(kW/K)^0.8
c52=11820; %$/(kg/sec)
c53=658; %$/(kg/sec)1.2

Σημείωση 1: Με την εντολή **save filename var1 var2…** μπορεί κάποιος να σώσει στο αρχείο filename.mat μόνο τις υποδεικνυόμενες μεταβλητές var1, var2 κτλ.

Όπως θα δούμε στη συνέχεια, με την εντολή **load data2.mat** θα μπορούμε να επαναφέρουμε στο χώρο εργασίας του matlab τις αποθηκευμένες μεταβλητές. Είναι χρήσιμο όμως, γενικά να γνωρίζουμε ότι για να μπορεί ένα αρχείο mat να διαβαστεί με την εντολή load, θα πρέπει όλες οι μεταβλητές που είναι αποθηκευμένες σε αυτό να έχουν τον **ίδιο αριθμό στηλών**.

Σημείωση 2: Τα σταθερά μεγέθη στο πρόβλημά μας δηλώνονται σαν μεταβλητές «**global**». Και αυτό, ώστε να εξασφαλιστεί η χρήση τους και στους χώρους εργασίας των αρχείων cgam_objective.m και cgam_constraints.m.

Σημείωση 3: Υπολογίστηκαν και κάποιες επιπλέον μεταβλητές (δεν δίδονται στον πίνακα Α1 του παραρτήματος Α) που χρειάζονται στους υπολογισμούς όπως οι **t8p**, (προέκυψε από το dt8p) **h8**, (προέκυψε για το p8 και t8 από το δ.Mollier), **h8p**, (προέκυψε για το p8 και t8p από το δ.Mollier), **h9**, (προέκυψε για το p9 και t9 από το δ.Mollier).

16.3 Το cgam_objective.m αναλυτικά.

Το αρχείο cgam_objective.m περιέχει τις παρακάτω γραμμές κώδικα:

```
1.  function F=cgam_objective(var)
2.  rc=var(1);
3.  nc=var(2);
4.  nt=var(3);
5.  t3=var(4);
6.  t4=var(5);

7.  %Ελεγχος αριθμου ορισματων εισοδου
8.  if length(var)<5
9.     error('5 arguments required')
10. end

11. % Δηλωση global μεταβλητων
12. global w ms hu cpa cpg dt8p t8p h8 h8p h9 dtpmin p1 p7 p8 p9 ga gg;
13. global nb u t0 t1 t7min t8 t9 Ra Rg raa rag rb rr;
14. global t cf fcr feliniko c11 c12 c21 c22 c23 c24 c31 c32 c33 c34 c41 c51 c52 c53;

15. %Σχέσεις μετατροπής
16. ka=(ga-1)/ga;
17. kg=(gg-1)/gg;
18. qs=ms*(h9-h8); %θερμική ισχύς ατμού
19. qec=ms*(h8p-h8); %θερμική ισχύς στον οικονομητηρα
20. qev=qs-qec; %θερμική ισχύς στον εξατμιστη

21. %Σχέσεις A1-A21
22. t2=t1*(1+((rc^ka)-1)/nc);
23. p2=rc*p1;
24. p3=raa*p2;
25. p4=rb*p3;
26. p6=p7/rr;
27. p5=p6/rag;
28. rt=p4/p5;
29. t5=t4*(1-nt*(1-(rt^(-kg))));
30. f=(cpg*(t4-t0)-cpa*(t3-t0))/(hu*nb-cpg*(t4-t0));
31. t6=t5-(cpa*(t3-t2))/((1+f)*cpg);
32. ma=w/((1+f)*cpg*(t4-t5)-cpa*(t2-t1));
33. mf=f*ma;
34. mg=ma+mf;
35. t7=t6-(qs/(mg*cpg));
36. t7p=t6-(qev/(mg*cpg));
```

```
37. wc=ma*cpa*(t2-t1);
38. wt=mg*cpg*(t4-t5);
39. dta=((t6-t2)-(t5-t3))/log((t6-t2)/(t5-t3));
40. aa=(mg*cpg*(t5-t6))/(u*dta);
41. dtec=((t7p-t8p)-(t7-t8))/log((t7p-t8p)/(t7-t8));
42. dtev=((t6-t9)-(t7p-t9))/log((t6-t9)/(t7p-t9));

43. tnt1=c12-nc;
44. if tnt1==0
45.    tnt1=tnt1+eps;
46. end
47. tnt2=c22-rb;
48. if tnt2==0
49.    tnt2=tnt2+eps;
50. end
51. tnt3=c32-nt;
52. if tnt3==0
53.    tnt3=tnt3+eps;
54. end
55. warning off MATLAB:divideByZero;

56. %Συντελεστές Ci Σχέσεις Α22-Α26
57. c1=((c11*ma)/tnt1)*rc*log(rc);
58. c2=((c21*ma)/tnt2)*(1+exp(c23*t4-c24));
59. c3=((c31*mg)/tnt3)*log(rc)*(1+exp(c33*t4-c34));
60. c4=c41*(aa^0.6);
61. c5=c51*(((qec/dtec)^0.8)+((qev/dtev)^0.8))+c52*ms+c53*(mg^1.2);

62. %Αντικειμενική συνάρτηση
63. F=fcr*feliniko*(c1+c2+c3+c4+c5)+cf*mf*hu*t;
```

16.3.1 Δήλωση συνάρτησης και αριθμού ορισμάτων. Γραμμές 1-6

Στις 6 πρώτες γραμμές δηλώνεται η συνάρτηση F η οποία εξαρτάται από το διάνυσμα var, που περιέχει τις 5 ανεξάρτητες μεταβλητές (rc, nc ,nt ,t3 ,t4). Το πρώτο στοιχείο του διανύσματος var τίθεται ίσο με το rc, το δεύτερο στοιχείο του nc, το τρίτο στοιχείο του nt κτλ.

Σημείωση: Αντί του:
>>function F=cgam_objective(var)
μπορούσαμε να γράψουμε εναλλακτικά:
>> function F=cgam_objective(rc,nc,nt,t3,t4)

16.3.2 Έλεγχος του αριθμού των ορισμάτων εισόδου. Γραμμές 7-10

Εδώ προστίθεται ένας βρόχος για τον έλεγχο του αριθμού των ορισμάτων εισόδου με χρήση της εντολής **length**. Όταν ο χρήστης κάνει λάθος και εισάγει λιγότερα από 5 ορίσματα, τότε καλείται η συνάρτηση **error**, εμφανίζεται το μήνυμα '5 arguments required' και τερματίζεται ο αλγόριθμος πρόωρα.

Σε ένα πίνακα x, η εντολή **length(x)** δίνει τον αριθμό των στοιχείων στην μεγαλύτερη διάσταση (δηλαδή, αριθμό γραμμών ή στηλών ανάλογα με το ποια έχει περισσότερα στοιχεία). Χρήσιμη εντολή είναι και η **size(x)**, η οποία δίνει δύο αριθμούς, τον αριθμό των γραμμών και των αριθμό των στηλών. Για παράδειγμα:

```
>> var=[1 2 3 4 5];
>> length(var)
ans =

   5

>> size(var)

ans =

   1     5
```

16.3.3 Δήλωση ολικών μεταβλητών. Γραμμές 11-14

Δηλώνονται σαν «global», όλες οι σταθερές παράμετροι του αρχείου data2.mat

16.3.4 Απαραίτητες σχέσεις μετατροπής. Γραμμές 15-20

Στις γραμμές 16-17 υπολογίζονται οι λόγοι των ειδικών θερμοχωρητικοτήτων για τον αέρα και το καυσαέριο ka,kg. Η μετατροπή είναι απαραίτητη αφού μας δίνονται τα γ_α και γ_g. Στις γραμμές 18-20 υπολογίζονται οι ροές θερμότητας qs, qec, qev, αναλυτικά από τις παροχές μάζας και τις ενθαλπίες.

16.3.5 Σχέσεις A1-A26. Γραμμές 21-42

Περιέχονται αναλυτικά οι σχέσεις A1-A26 στο APPENDIX A του παραρτήματος Α, με την σωστή σειρά για τον υπολογισμό των ενδιάμεσων μεγεθών που θα χρησιμοποιηθούν στους υπολογισμούς της αντικειμενικής συνάρτησης F.

Σημείωση: Στο matlab το μαθηματικό σύμβολο ln παριστάνεται από το σύμβολο **log**, ενώ το σύμβολο log (λογάριθμος με βάση το 10) παριστάνεται από το σύμβολο **log10**.

16.3.6 Βρόχος ελέγχου για την πρόληψη απειρισμού (infinity) στις επιμέρους σχέσεις. Γραμμές 43-54

Στη σχέση Α.22 θα υπάρξει πρόβλημα απειρισμού του συντελεστή C_1 αν $c_{12}=\eta_C=0.9$. Το ίδιο θα συμβεί στις σχέσεις Α.23 και Α.24 αν $c_{22}=r_B$ και $c_{32}=n_T=0.92$ αντίστοιχα.

Παρατηρήθηκε κατά την εκτέλεση του προγράμματος, ότι όταν στον καθορισμό των ανωτάτων ορίων των ανεξαρτήτων μεταβλητών χρησιμοποιούνταν αυτές οι τιμές, δηλαδή $n_C=0.9$ και $n_T=0.92$ το πρόγραμμα έβγαζε προειδοποιητικά μηνύματα, ότι το αποτέλεσμα είναι Nan (Not A Number) και έπειτα από λίγο σταματούσε τη λειτουργία του χωρίς αποτέλεσμα. Γι' αυτό θεωρήθηκε σκόπιμο να αποτρέψουμε αυτό το πρόβλημα με κάποιο βρόχο if.

Δημιουργήθηκαν οι νέες μεταβλητές tnt1=c12-nc, tnt2=c22-rb, tnt3=c32-nt. Ο βρόχος περιλαμβάνει τη **σύγκριση των μεταβλητών** με το μηδέν (γι' αυτό χρησιμοποιείται το σύμβολο (==) και όχι η απλή ισότητα (=)). Αν η δήλωση είναι αληθής τότε η τιμή της μεταβλητής αντικαθίσταται με μια νέα τιμή, που προκύπτει αν προστεθεί σε αυτή η συνάρτηση **eps**. Είχε ειπωθεί στην εισαγωγή ότι το eps είναι ο μικρότερος αριθμός που μπορεί να προστεθεί στη μονάδα και να δώσει αποτέλεσμα μεγαλύτερο της μονάδας.

Στο πρόβλημα που μελετούμε η μέθοδος αυτή είχε σαν αποτέλεσμα οι μεταβλητές tnt1, tnt2, tnt3 να αποκτήσουν τιμή «κάτι παραπάνω από μηδέν» με αποτέλεσμα ο αλγόριθμος να συνεχίζεται κανονικά.

Το μόνο πρόβλημα που προέκυψε ήταν η εμφάνιση του προειδοποιητικού μηνύματος **Divide by zero cgam_objective.m line 57** και **Divide by zero cgam_objective.m line 59**, χωρίς όμως αυτό να εμποδίζει την εκτέλεση του αλγορίθμου. Αυτό το ενοχλητικό μήνυμα μπορεί να εξαφανιστεί χρησιμοποιώντας την συνάρτηση **warning**. Έτσι με την εντολή **warning off MATLAB:divideByZero** το μήνυμα αυτό δεν εμφανίστηκε πάλι.

16.3.7 Δήλωση παραγόντων του κόστους κεφαλαίου. Γραμμές 56-61

Περιέχει τη δήλωση των παραγόντων που συνθέτουν το κόστος κεφαλαίου (σχέσεις Α.22-Α.26 του APPENDIX A).

16.3.8 Δήλωση της αντικειμενικής συνάρτησης. Γραμμές 62-63

Περιέχει τη δήλωση της αντικειμενικής συνάρτησης (σχέση Α.27 του APPENDIX A).

16.4 Το cgam_constraints.m αναλυτικά.

Το αρχείο constraints.m περιέχει τις παρακάτω γραμμές κώδικα:

```
1  function [c,ceq]=cgam_constraints(var)
2  rc=var(1);
3  nc=var(2);
4  nt=var(3);
5  t3=var(4);
6  t4=var(5);

7  %Ελεγχος αριθμου ορισματων εισοδου
8  if length(var)<5
9     error('5 arguments required')
10 end

11 % Δηλωση global μεταβλητων
12 global w ms hu cpa cpg dt8p t8p h8 h8p h9 dtpmin p1 p7 p8 p9 ga gg;
13 global nb u t0 t1 t7min t8 t9 Ra Rg raa rag rb rr;
14 global t cf fcr feliniko c11 c12 c21 c22 c23 c24 c31 c32 c33 c34 c41 c51 c52 c53;

15 %ΣΧΕΣΕΙΣ ΜΕΤΑΤΡΟΠΗΣ
16 ka=(ga-1)/ga;
17 kg=(gg-1)/gg;
18 qs=ms*(h9-h8); %θερμικη ισχυς ατμου
19 qec=ms*(h8p-h8); %θερμικη ισχυς στον οικονομητηρα
20 qev=qs-qec; %θερμικη ισχυς στον εξατμιστη
21 %ΣΧΕΣΕΙΣ Α1-Α21

22 t2=t1*(1+((rc^ka)-1)/nc);
23 p2=rc*p1;
24 p3=raa*p2;
25 p4=rb*p3;
26 p6=p7/rr;
27 p5=p6/rag;
28 rt=p4/p5;
29 t5=t4*(1-nt*(1-(rt^(-kg))));
30 f=(cpg*(t4-t0)-cpa*(t3-t0))/(hu*nb-cpg*(t4-t0));
31 t6=t5-(cpa*(t3-t2))/((1+f)*cpg);
32 ma=w/((1+f)*cpg*(t4-t5)-cpa*(t2-t1));
33 mf=f*ma;
34 mg=ma+mf;
35 t7=t6-(qs/(mg*cpg));
36 t7p=t6-(qev/(mg*cpg));
```

```
37 | wc=ma*cpa*(t2-t1);
38 | wt=mg*cpg*(t4-t5);
39 | dta=((t6-t2)-(t5-t3))/log((t6-t2)/(t5-t3));
40 | aa=(mg*cpg*(t5-t6))/(u*dta);
41 | dtec=((t7p-t8p)-(t7-t8))/log((t7p-t8p)/(t7-t8));
42 | dtev=((t6-t9)-(t7p-t9))/log((t6-t9)/(t7p-t9));

43 | %ΠΕΡΙΟΡΙΣΜΟΙ
44 | c=[t2-t3;t3-t5;
45 |    t3-t4;t2-t6;
46 |    t9-t6;t7min-t7;t9+dtpmin-t7p];
47 | ceq=[ ];
```

Γραμμές 1-6

Στη δήλωση των περιορισμών, που όπως είπαμε πρέπει να είναι σε πινακοποιημένη μορφή, έχουμε 2 ορίσματα εξόδου. Στο όρισμα c πρέπει να δηλωθούν οι ανισοτικοί περιορισμοί και στο ceq οι ισοτικοί περιορισμοί. Στην περίπτωση που είτε δεν έχουμε ισοτικούς είτε ανισοτικούς περιορισμούς, τότε τα c, ceq πρέπει να αντικατασταθούν ίσα με τον κενό πίνακα ([ ]). Τα c, ceq εξαρτώνται από την μεταβλητή var.

Γραμμές 7-42

Έχουμε τις ίδιες γραμμές κώδικα όπως στο cgam_objective.m. Αρχικά χρησιμοποιούμε το βρόχο ελέγχου των ορισμάτων εισόδου, ακολουθεί η δήλωση των σταθερών μεγεθών σαν global μεταβλητές, ώστε αυτές να μπορούν να χρησιμοποιηθούν στο χώρο εργασίας του cgam_constraints.m. Έπειτα παρατίθενται οι σχέσεις μετατροπής και ακολουθούν οι σχέσεις A1-A26.

Γραμμές 43-47

Παρατίθενται σε πινακοποιημένη μορφή οι μη γραμμικοί περιορισμοί. Κάθε γραμμή του πίνακα c αποτελεί και έναν ανισοτικό περιορισμό. Οι περιορισμοί του προβλήματος είναι εκείνοι που δίδονται στο Appendix A του Παραρτήματος Α.

Υπενθυμίζεται, ότι οι ανισοτικοί περιορισμοί πρέπει πάντα να γράφονται στη μορφή c(x)<=0, οπότε πολλές φορές είναι απαραίτητη η αλλαγή προσήμου.
Εφόσον δεν υπάρχουν ισοτικοί περιορισμοί, οφείλουμε, το όρισμα ceq να το θέσουμε ίσο με τον κενό πίνακα, δηλαδή ceq=[ ]

Σημείωση: Στη δημιουργία πινάκων η αλλαγή γραμμής μπορεί να γίνει με απλό πάτημα του πλήκτρου enter αντί για το διαχωρισμό με αγγλική άνω τελεία (;). Εδώ κρίθηκε απαραίτητο να το κάνουμε αυτό, προκειμένου να μην έχουμε μια τόσο εκτενή γραμμή στη σύνταξη. Δηλαδή θα μπορούσαμε να φτιάξουμε και την έκφραση:
c=[t2-t3;t3-t5;t3-t4;t2-t6;t9-t6;t7min-t7;t9+dtpmin-t7p];

16.5 Το cgam_optimization algorithm.m αναλυτικά

Το αρχείο αυτό δεν αποτελεί ούτε script m-file, ούτε function m-file. Αποτελεί απλώς ένα αποθηκευμένο κείμενο, με τις εντολές εκκίνησης του αλγορίθμου, προκειμένου να μην χρειάζεται κάθε φορά που εκτελείται ο αλγόριθμος να πληκτρολογούνται όλες αυτές τις εντολές μία προς μία στο «command window», αλλά με μια απλή αντιγραφή (copy) των εντολών και επικόλληση (paste) στο «command window» να εκτελείται σύντομα. Αποτελείται από τις παρακάτω γραμμές κώδικα:

```
1  clear

2  global w ms hu cpa cpg dt8p t8p h8 h8p h9 dtpmin p1 p7 p8 p9 ga gg;
3  global nb u t0 t1 t7min t8 t9 Ra Rg raa rag rb rr;
4  global t cf fcr feliniko c11 c12 c21 c22 c23 c24 c31 c32 c33 c34 c41 c51 c52 c53;

5  load data2.mat

6  format short e

7  var0=[7.0,0.75,0.82,900,1300];

8  %boundaries and options
9  lboundary=[5.0,0.6,0.6,700.0,1200.0];
10 uboundary=[10.0,0.9,0.92,1000.0,1800.0];

11 options=optimset('LargeScale','off','Display','iter','Diagnostics','on',...
12    'TolCon',0.0001,'TolX',0.0001,'DiffMinChange',1e-6);

13 [var,fval,exitflag,output,lambda,grad,hessian]=fmincon(@cgam_objective,var0,[],[],[],[],...
14    lboundary,uboundary,@cgam_constraints,options);

15 fprintf(1,'Final result from optimization\n');
16 fprintf(1,'rc = %10.5f \n',var(1));
17 fprintf(1,'nc = %10.5f \n',var(2));
18 fprintf(1,'nt = %10.5f \n',var(3));
19 fprintf(1,'t3 = %10.5f K\n',var(4));
20 fprintf(1,'t4 = %10.5f K\n',var(5));
21 fprintf(1,'F = %10.5e $/year\n',fval);
22 [c,ceq]=cgam_constraints(var);
23 fprintf(1,'Constraints values at the optimum point\n');
24 fprintf(1,'t3-t2 = %10.5f K\n',-c(1));
25 fprintf(1,'t5-t3 = %10.5f K\n',-c(2));
26 fprintf(1,'t4-t3 = %10.5f K\n',-c(3));
27 fprintf(1,'t6-t2 = %10.5f K\n',-c(4));
```

```
28 fprintf(1,'t6-t9 = %10.5f K\n',-c(5));
29 fprintf(1,'t7-t7min= %10.5f K\n',-c(6));
30 fprintf(1,'t7p-t9-dtpmin = %10.5f K\n',-c(7));
31 fprintf(1,'Langrange multipliers at the optimum point\n');
32 disp(lambda)
33 fprintf(1,'Gradient at the optimum point\n');
34 disp(grad)
35 fprintf(1,'Hessian at the optimum point\n');
36 disp(hessian)
```

Γραμμή 1

Με την εντολή **clear**, το matlab διαγράφει από το χώρο εργασίας του όλες τις αποθηκευμένες μεταβλητές. Αυτό γίνεται προκειμένου να μην προκληθεί σύγχυση από τις μεταβλητές που δημιουργήθηκαν κατά την εκτέλεση προηγούμενων προβλημάτων, αλλά και για να ελευθερωθεί η μνήμη.

Γραμμές 2-4

Όπως πριν, δηλώνονται οι σταθερές παράμετροι σαν global.

Γραμμή 5

Με την εντολή **load**, «επαναφέρουμε» τις σταθερές παραμέτρους στο χώρο εργασίας του matlab (matlab workspace).
Δηλώνοντας τις παραμέτρους αυτές σαν «global» τόσο εδώ, όσο και στα αρχεία cgam_objective.m και cgam_constraints.m, είναι δυνατή η κοινή κλήση και χρήση των παραμέτρων αυτών από το χώρο εργασίας του matlab μεταξύ των τοπικών χώρων εργασίας που δημιουργούν τα αρχεία αυτά.

Γραμμή 6

Με την εντολή **format short e**
το matlab εμφανίζει στην έξοδο όλους τους αριθμούς με τέσσερα ψηφία μετά την υποδιαστολή (5 μαζί με την υποδιαστολή) συν το εκθετικό.

Αυτή η αναπαράσταση επιλέγεται προκειμένου να εμφανίζονται με πιο κατανοητή μορφή, ο Ιακωβιανός πίνακας και ο Εσσιανός, στο βέλτιστο σημείο, καθότι προκύπτουν αριθμοί μεγάλης τάξης μεγέθους. Αναλυτικά στοιχεία για την αναπαράσταση των αριθμών μπορεί να βρει ο αναγνώστης στην ενότητα 3.3.

Γραμμή 7

Το var0 είναι το αρχικό σημείο εκκίνησης του αλγορίθμου βελτιστοποίησης. Το «0» στο τέλος του ονόματος της ανεξάρτητης μεταβλητής είναι απαραίτητο προκειμένου το matlab να καταλάβει ότι πρόκειται για το σημείο εκκίνησης. Έτσι έχουμε τα εξής σημεία εκκίνησης για τις ανεξάρτητες μεταβλητές:
rc=7.0
nc=0.75
nt=0.82
t3=900 K
t4=1300 K

Γραμμές 7-9

Εδώ δημιουργούμε δύο νέα διανύσματα, απαραίτητα για τον αλγόριθμο βελτιστοποίησης, που περιέχουν τα κατώτερα και τα ανώτερα όρια για τις μεταβλητές μας. Συγκεκριμένα δίνουμε τις εξής τιμές
5<= rc <=10
0.6<= nc <=0.9
0.6<= nt <=0.92
700<= t3 <=1000
1200<= t4 <=1800

Γραμμές 11-12

Καθορίζεται η δομή options, στην οποία δίδονται οι επιλογές του αλγορίθμου βελτιστοποίησης.

- Με την επιλογή '**Largescale', 'off'**, δηλώνουμε στο matlab ότι θέλουμε να χρησιμοποιήσουμε αλγόριθμο επίλυσης μέσης κλίμακας και όχι μεγάλης κλίμακας. Εξάλλου για τη χρήση μεθόδων μεγάλης κλίμακας, ο χρήστης πρέπει να παρέχει το διάνυσμα κλίσης της αντικειμενικής συνάρτησης αναλυτικά, κάτι που εδώ δεν είναι εφικτό.

- Με την επιλογή '**Display', 'iter'**, σε κάθε επανάληψη του αλγορίθμου αναγράφονται ο αριθμός των υπολογισμών της αντικειμενικής συνάρτησης και της τιμής της σε κάθε επανάληψη, το βήμα και άλλες πληροφορίες.

- Με την επιλογή **Diagnostics on**, δίδονται πληροφορίες σχετικά με το πρόβλημα της βελτιστοποίησης όπως ο αριθμός των ανεξάρτητων μεταβλητών, ο αριθμός των περιορισμών, η χρησιμοποιούμενη μέθοδος κ.α.

- Με την επιλογή **'TolCon',0.0001**καθορίζεται το κριτήριο τερματισμού όσον αφορά την παραβίαση των περιορισμών, με αυτή την ανοχή.

- Με την επιλογή **'TolX',0.0001**καθορίζεται το κριτήριο τερματισμού όσον αφορά την τιμή του x. Όταν $\|x_k - x_{k-1}\| \le 10^{-4}$ τότε ο αλγόριθμος τερματίζεται.

- Με την επιλογή **DiffMinChange** και **DiffMaxChange** μπορούμε να καθορίσουμε την ελάχιστη αλλαγή στις μεταβλητές όταν το matlab υπολογίζει τις παραγώγους με την μέθοδο των πεπερασμένων διαφορών (finite difference). Θέτουμε **'DiffMinChange' , '1e-6 '**.

Η ρύθμιση των **DiffMinChange** και **DiffMaxChange** αφορά μόνο τους αλγόριθμους μέσης κλίμακας. Στους αλγορίθμους μεγάλης κλίμακας δεν χρησιμοποιούνται, αφού εκεί ο χρήστης πρέπει να παρέχει το διάνυσμα κλίσης αναλυτικά.

Η ρύθμιση των DiffMinChange, DiffMaxChange, προτείνεται ως λύση, σε περιπτώσεις που αντιμετωπίζονται προβλήματα όπως:

α) Όταν έχουμε αργή σύγκλιση κοντά στη λύση που οφείλεται σε «truncation error» κατά τον υπολογισμό του διανύσματος κλίσης. Η ρύθμιση του DiffMinChange αλλά και του DiffMaxChange μπορεί να λύσει αυτό το πρόβλημα, αυξάνοντας την ακρίβεια στον υπολογισμό του.

b) Όταν η αντικειμενική συνάρτηση έχει ασυνέχειες. Η ρύθμιση του DiffMinChange και του DiffMaxChange, αλλάζει το εύρος των τιμών που παίρνει η μεταβλητή x έτσι ώστε πάντα DiffMinChange < Δx < DiffMaxChange. Με αυτόν τον τρόπο πολλές φορές μπορούν να ξεπεραστούν κάποιες μικρές ασυνέχειες.

Γραμμές 13-14

Στις γραμμές αυτές συντάσσεται η εντολή **fmincon**. Έχουμε σαν ορίσματα εξόδου τα:
α) var (Βέλτιστες τιμές των ανεξάρτητων μεταβλητών rc, nc, nt, t3, t4),
b) fval (Η τιμή της αντικειμενικής συνάρτησης στο βέλτιστο σημείο)
c) exitflag (δομή που μας παρέχει πληροφορίες για την κατάσταση τερματισμού)
d) Output (δομή που περιέχει πληροφορίες σχετικά με την έξοδο του αλγορίθμου).

Σαν ορίσματα εισόδου έχουμε:
a)την αντικειμενική συνάρτηση που περιέχεται στο αρχείο cgam_objective.m,
b) Το σημείο εκκίνησης var0,
c) Το ανώτερο και το κατώτερο όριο των μεταβλητών lboundary, uboundary,
d) Τους μη γραμμικούς περιορισμούς που περιέχονται στο αρχείο cgam_constraints.m,
e) Τις επιλογές (options) που καθορίστηκαν παραπάνω.

Σημείωση: Υπενθυμίζεται ότι η εντολή @objectiveh.m ισοδυναμεί με την εντολή feval(objectiveh,var0), με την οποία ξεκινά ο υπολογισμός της αντικειμενικής συνάρτησης

(function evaluation). Επίσης, εφόσον δεν έχουμε γραμμικές ισότητες και ανισότητες θέτουμε τα ορίσματα αυτά ίσα με τον κενό πίνακα.

Γραμμές 15-21

Γίνεται χρήση της εντολής fprintf με την οποία μπορούμε να έχουμε μια πιο βολική εμφάνιση των αποτελεσμάτων του αλγορίθμου. Με την εντολή fprintf είναι δυνατή η μορφοποίηση των αποτελεσμάτων του αλγορίθμου βελτιστοποίησης και η αποθήκευσή τους σε κάποιο εξωτερικό αρχείο (περισσότερες πληροφορίες στο ΠΑΡΑΡΤΗΜΑ Ε).

Για παράδειγμα στη γραμμή 16 έχουμε:

fprintf(1,'rc = %10.5f \n',var(1));

Με αυτή την εντολή μορφοποιείται το στοιχείο της πρώτης στήλης του βέλτιστου πίνακα var. Το στοιχείο αυτό θα έχει μέγεθος πεδίου 10 ψηφίων και ακρίβεια 5 δεκαδικών ψηφίων μετά την υποδιαστολή. Με τον χαρακτήρα διαφυγής \n δηλώνεται ότι τα επόμενα μορφοποιημένα αποτελέσματα θα τυπωθούν σε νέα γραμμή. Το αποτέλεσμα θα εμφανιστεί στην οθόνη και γι' αυτό ο αριθμός 1 υπάρχει στη θέση του ορίσματος fid.
Με την ίδια λογική μορφοποιούνται και τα υπόλοιπα στοιχεία του πίνακα var.

Γραμμή 22

Με αυτή την εντολή κατασκευάζονται οι μεταβλητές c, ceq οι οποίες περιέχουν τις τιμές των μη γραμμικών ανισοτικών και ισοτικών περιορισμών, αντίστοιχα, στο βέλτιστο σημείο var.

Γραμμές 23-30

Εδώ, απλώς εμφανίζουμε τις τιμές των περιορισμών στο βέλτιστο σημείο. Προφανώς πριν από κάθε όρισμα πρέπει να χρησιμοποιήσουμε το σύμβολο της αφαίρεσης (-) και αυτό διότι οι περιορισμοί έχουν γραφτεί στη μορφή –c(x)<=0. Αν δεν γίνει αυτό, τα αποτελέσματα που θα πάρουμε, απλώς θα είναι αρνητικά. Οι ρυθμίσεις που έγιναν στην fprintf είναι οι ίδιες όπως προηγούμενα.

Γραμμές 31-36

Για λόγους πληρότητας εμφανίζουμε και τις τιμές των πολλαπλασιαστών Langrange ,του Ιακωβιανού και του Εσσιανού πίνακα στο βέλτιστο σημείο.

Αντί της εντολής fprintf χρησιμοποιούμε της εντολή **disp** με την οποία μπορούμε να εμφανίζουμε μη μορφοποιημένα δεδομένα. Συντάσσεται, περικλείοντας τη μεταβλητή που θέλουμε να εμφανιστεί με απλές παρενθέσεις.

16.6 Τα αποτελέσματα του αλγορίθμου βελτιστοποίησης

Ανοίγουμε το cgam_optimization algorithm.m και επιλέγουμε **edit>select all** και έπειτα **edit>copy**. Μετά πηγαίνουμε στο «command prompt» και επιλέγουμε **edit>paste** και πατάμε το enter. Το matlab δίνει τα εξής αποτελέσματα:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 Diagnostic Information

Number of variables: 5

Functions
Objective:                    cgam_objective
Gradient:                     finite-differencing
Hessian:                      finite-differencing (or Quasi-Newton)
Nonlinear constraints:              cgam_constraints
Gradient of nonlinear constraints:    finite-differencing

Constraints
Number of nonlinear inequality constraints: 7
Number of nonlinear equality constraints:     0

Number of linear inequality constraints:     0
Number of linear equality constraints:       0
Number of lower bound constraints:          5
Number of upper bound constraints:          5

Algorithm selected
medium-scale


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
End diagnostic information


                         max                        Directional        First-order
Iter F-count      f(x)   constraint       Step-size        derivative      optimality Procedure
   1     14     1.17718e+007       -0.05          0.5              1.29e+007    4.27e+007
   2     22     1.14103e+007      -0.025         0.5              1.76e+006    2.06e+007
   3     33     1.13789e+007    -0.03516      0.0625              1.08e+006    1.26e+007
   4     41     1.11365e+007    -0.03991        0.5                1.17e+006    7.16e+006
   5     51     1.11206e+007    -0.03679       0.125              2.46e+006    1.24e+007
   6     59     1.08708e+007    -0.01839        0.5                8.12e+005    3.77e+007
```

7	68	1.08707e+007	-0.04292	0.25	1.67e+006	1.04e+007
8	76	1.08058e+007	-0.02146	0.5	8.29e+005	3.16e+007
9	86	1.06551e+007	-0.02826	0.125	5e+005	9.74e+006
10	95	1.0593e+007	-0.03507	0.25	9.19e+005	1.77e+007
11	104	1.05709e+007	-0.03555	0.25	6.75e+004	6.52e+005
12	113	1.05526e+007	-0.04118	0.25	1.27e+006	2.59e+006
13	120	1.05341e+007	0	1	1.85e+005	1.71e+006
14	129	1.05139e+007	-0.04099	0.25	9.54e+004	1.7e+006
15	136	1.04712e+007	-0.04195	1	1.24e+005	3.48e+007
16	148	1.04676e+007	-0.04229	0.0313	2.34e+005	4.91e+006
17	158	1.0463e+007	-0.04085	0.125	2.38e+004	4.79e+005
18	169	1.04629e+007	-0.04102	0.0625	2.81e+003	3.52e+006
19	179	1.04617e+007	-0.04393	0.125	2.9e+004	3.91e+006
20	190	1.04615e+007	-0.04406	0.0625	3.16e+003	1.8e+006
21	197	1.04607e+007	-0.04254	1	-199	2.22e+006
22	204	1.04605e+007	-0.04273	1	385	3.67e+005
23	211	1.04602e+007	-0.04237	1	-121	2.7e+005
24	218	1.04601e+007	-0.04258	1	-3.39	2.95e+005
25	225	1.04601e+007	-0.04263	1	11.8	1.14e+005
26	232	1.04601e+007	-0.04266	1	0.0721	1.39e+004
27	239	1.04601e+007	-0.04264	1	0.262	2.65e+003
28	246	1.04601e+007	-0.04265	1	0.0579	4.21e+003
29	257	1.04601e+007	-0.04265	0.0625	0.019	4.31e+003
30	264	1.04601e+007	-0.04265	1	0.000567	4.2e+003
31	271	1.04601e+007	-0.04265	1	0.0423	4.23e+003

Optimization terminated successfully:
Search direction less than 2*options.TolX and
maximum constraint violation is less than options.TolCon
No Active Constraints

Final result from optimization
rc = 8.52116
nc = 0.84698
nt = 0.87735
t3 = 915.14447 K
t4 = 1491.15354 K
F = 1.04601e+007 $/year

Constraints values at the optimum point
t3-t2 = 319.75034 K
t4-t3 = 576.00906 K
t5-t3 = 72.53788 K
t6-t2 = 122.30715 K

t6-t9 = 232.18128 K
t7-t7min= 27.51967 K
t7p-t9-dtpmin = 1.65752 K

Langrange multipliers at the optimum point
lower: [5x1 double]
upper: [5x1 double]
eqlin: [0x1 double]
eqnonlin: [0x1 double]
ineqlin: [0x1 double]
ineqnonlin: [7x1 double]

Gradient at the optimum point
-2.6077e-002
1.1643e+003
4.2310e+003
-2.2952e+002
3.7253e-003

Hessian at the optimum point
6.5721e+005 3.7054e+007 7.0698e+007 5.4400e+004 -2.2884e+004
3.7054e+007 7.0028e+009 2.0900e+010 3.1658e+006 -1.3448e+006
7.0698e+007 2.0900e+010 6.8892e+010 6.0779e+006 -2.4927e+006
5.4400e+004 3.1658e+006 6.0779e+006 4.7135e+003 -1.9753e+003
-2.2884e+004 -1.3448e+006 -2.4927e+006 -1.9753e+003 1.0126e+003

16.7 Σχόλια επί των αποτελεσμάτων

Προκύπτει ο παρακάτω συγκριτικός πίνακας των μεθόδων για τις ίδιες τιμές των παραμέτρων

Πίνακας 16.7.1 Αποτελέσματα και των τεσσάρων μεθόδων επίλυσης

ΜΕΤΑΒΛΗΤΗ	ΜΕΘΟΔΟΣ			
	GRG2	TFA	Modular	Matlab SQP
r_c	8.59730	8.59770	8.59050	8.52116
η_c	0.84641	0.84650	0.84653	0.84698
η_t	0.87886	0.87871	0.87878	0.87735
T_3(K)	912.77	913.14	912.93	915.14
T_4(K)	1491.40	1491.97	1491.50	1491.15
F($/year)	$1.0426\ 10^7$	$1.0426\ 10^7$	$1.0426\ 10^7$	$1.0460\ 10^7$

Προκύπτει **αύξηση** στην τιμή της αντικειμενικής συνάρτησης (συνάρτησης κόστους) ίση με **34.000$/χρόνο**.

Πίνακας 16.7.2 Ποσοστιαία μεταβολή της βέλτιστης λύσης μεταξύ της μεθόδου Matlab SQP και της GRG2

ΜΕΤΑΒΛΗΤΗ	ΠΟΣΟΣΤΙΑΙΑ ΜΕΤΑΒΟΛΗ %
$\Delta r_c/r_{c\ (GRG2)}$	-0,8856
$\Delta n_c/n_{c\ (GRG2)}$	0,0673
$\Delta n_t/n_{t\ (GRG2)}$	-0,1718
$\Delta T_3/T_{3\ (GRG2)}$	0,2596
$\Delta T_4/T_{4\ (GRG2)}$	-0,0168
$\Delta F/F_{(GRG2)}$	0,3261

Παρατηρήσεις:

- Στο συγκεκριμένο πρόβλημα δεν παίζουν ιδιαίτερο ρόλο οι ρυθμίσεις των κριτηρίων τερματισμού TolX και TolCon αφού όσο αυστηρά ή όχι καθοριστούν, η διαφορά στα αποτελέσματα είναι ελάχιστη. Πάντως, γενικά, τα κριτήρια τερματισμού παίζουν ρόλο στα σημεία που δεν έχουμε σύγκλιση. Μπορεί με λιγότερο αυστηρά κριτήρια τερματισμού, να επιτευχθεί σύγκλιση ή έστω να πάρουμε καλύτερα αποτελέσματα.

- Πολύ μεγάλο ρόλο παίζει η ρύθμιση της παραμέτρου DiffMinChange. Τα αποτελέσματα που παίρνουμε για διάφορες τιμές του DiffMinChange είναι αρκετά

μακριά μεταξύ τους. Ο παρακάτω πίνακας δίνει τα αποτελέσματα του αλγορίθμου για τιμές του DiffMinChange 10^{-3}, 10^{-4}, 10^{-9}

Πίνακας 16.7.3 Επίδραση της επιλογής DiffMinChange στα αποτελέσματα

ΜΕΤΑΒΛΗΤΗ	DiffMinChange		
	10^{-3}	10^{-4}	10^{-9}
r_c	8,38667	8,51857	8,52134
n_c	0,85160	0,84715	0,84698
n_t	0,88092	0,87697	0,87735
T_3(K)	908,22	915,64	915,14
T_4(K)	1488,46	1491,37	1491,15
F($/year)	1.04654e+007	1.04601e+007	1.04601e+007

Παρατηρούμε ότι ιδιαίτερα αυστηρή (μικρή) τιμή του DiffMinChange έχει πολύ μικρή επίδραση στα αποτελέσματα.

- Παρατηρήθηκε ότι για κάποια δοθέντα αρχικά σημεία ο αλγόριθμος δεν εμφάνισε αποτελέσματα και το πρόγραμμα σταμάτησε τη λειτουργία του. Αυτό μάλλον συνέβη διότι το πρόβλημά μας είναι ιδιαίτερα ευαίσθητο σε μικρές αλλαγές των ανεξάρτητων μεταβλητών.

- Εφόσον δοκιμάστηκαν διαφορετικά αρχικά σημεία τα οποία συνέκλιναν σε μια συγκεκριμένη λύση, μπορούμε να υποθέσουμε ότι η διαφορά στα αποτελέσματα δεν οφείλεται σε κάποιο λάθος χειρισμό στη σύνταξη του προγράμματος αλλά στην μέθοδο SQP που χρησιμοποιεί το matlab.

16.8 Προσπάθεια επίλυσης του ίδιου προβλήματος με μετατροπή των μεταβλητών στην ίδια τάξη μεγέθους (scaling).

Έγινε προσπάθεια επίλυσης του προβλήματος βελτιστοποίησης με μετατροπή των ανεξάρτητων παραμέτρων στην ίδια τάξη μεγέθους. Με τον τρόπο αυτό, σε αρκετές περιπτώσεις που οι μεταβλητές απόφασης είναι διαφορετικής τάξης μεγέθους, μπορούμε να πάρουμε καλύτερα αποτελέσματα, και πολλές φορές πιο γρήγορα απ' ότι χωρίς κλιμάκωση.

Στο πρόβλημα που μελετούμε, οι ανεξάρτητες μεταβλητές τροποποιήθηκαν ώστε να είναι όλες τάξεως μεγέθους 100. Έτσι δημιουργήθηκαν οι νέες μεταβλητές:

$$r_C* = r_C \cdot 10$$
$$n_C* = n_C \cdot 100$$
$$n_T* = n_T \cdot 100$$
$$T_3* = T_3 / 10$$
$$T_4* = T_4 / 10$$

Οι νέες μεταβλητές πριν τον υπολογισμό των τιμών της αντικειμενικής συνάρτησης και των περιορισμών πρέπει να μετατραπούν στις αρχικές. Να γίνει δηλαδή η αντίστροφη διαδικασία:

$$r_C = r_C* / 10$$
$$n_C = n_C* / 100$$
$$n_T = n_T* / 100$$
$$T_3 = T_3* \cdot 10$$
$$T_4 = T_4* \cdot 10$$

Η διαδικασία αυτή δεν ήταν καθόλου επίπονη. Χρειάστηκαν απλώς λίγες μεταβολές στον κώδικα των αρχείων cgam_objective, cgam_constraints, cgam_optimization algorithm.

Τα νέα αρχεία μετονομάστηκαν σε scale_cgam_objective, scale_cgam_constraints, scale_cgam_optimization algorithm αντίστοιχα.

16.8.1 Το τροποποιημένο αρχείο scale_cgam_optimization algorithm.m

Οι μόνες αλλαγές που έγιναν είναι στη δήλωση του αρχικού σημείου και στα όρια των μεταβλητών. Έτσι δηλώνουμε ως νέο αρχικό σημείο το:

$$(r_C*, n_C*, n_T*, T_3*, T_4*) = (r_C \cdot 10, n_C \cdot 100, n_T \cdot 100, T_3 / 10, T_4 / 10)$$

Και για
$(r_C, n_C, n_T, T_3, T_4) = (7, 0.75, 0.82, 900, 1300)$ προκύπτει:

$$(r_C^*, n_C^*, n_T^*, T_3^*, T_4^*) = (70, 75, 82, 90, 130)$$

Παρόμοια, τα νέα όρια στις μεταβλητές θα είναι:

$$50 \le r_C^* \le 100$$
$$60 \le n_C^* \le 90$$
$$60 \le n_T^* \le 92$$
$$70 \le T_3^* \le 100$$
$$120 \le T_4^* \le 180$$

Στο scale_cgam_optimization algorithm.m αλλάζουν μόνο οι γραμμές 7-10 (στις οποίες δηλώνονται το νέο αρχικό σημείο και τα νέα όρια στις μεταβλητές) σε σχέση με το cgam_objective.m:

```
7.  var0=[70,75,82,90,130];
8.  %oria kai options
9.  lboundary=[50,60,60,70,120];
10. uboundary=[100,90,92,100,180];
```

16.8.2 Τα τροποποιημένα αρχεία scale_cgam_objective.m και scale_cgam_constraints.m

Εδώ, όπως ειπώθηκε προηγούμενα γίνεται η μετατροπή των μεταβλητών $(r_C^*, n_C^*, n_T^*, T_3^*, T_4^*)$ στις $(r_C, n_C, n_T, T_3, T_4) = (r_C^*/10, n_C^*/100, n_T^*/100, T_3^* \cdot 10, T_4^* \cdot 10)$ και κατόπιν γίνεται ο υπολογισμός των τιμών της αντικειμενικής συνάρτησης και των περιορισμών.

Στο **scale_cgam_objective.m** αλλάζουν μόνο οι γραμμές 2-6 (στις οποίες γίνεται η μετατροπή) σε σχέση με το cgam_objective.m:

```
1 function F=scale_cgam_objective(var)
2 rc=var(1)/10;
3 nc=var(2)/100;
4 nt=var(3)/100;
5 t3=var(4)*10;
6 t4=var(5)*10;
```

Στο **scale_cgam_constraints.m** αλλάζουν μόνο οι γραμμές 2-6 (στις οποίες γίνεται η μετατροπή) σε σχέση με το cgam_constraints.m:

```
1  function [c,ceq]=scale_cgam_constraints(var)
2  rc=var(1)/10;
3  nc=var(2)/100;
4  nt=var(3)/100;
5  t3=var(4)*10;
6  t4=var(5)*10;
```

16.8.3 Αποτελέσματα του αλγορίθμου με χρήση της κλιμάκωσης

Ανοίγουμε το scale_cgam_optimization algorithm.m και επιλέγουμε **edit>select all** και έπειτα **edit>copy**. Μετά πηγαίνουμε στο «command prompt» και επιλέγουμε **edit>paste** και πατάμε το enter. Το matlab έπειτα από **30 επαναλήψεις** δίνει τα εξής αποτελέσματα:

Final result from optimization
rc = 8.52118
nc = 0.84698
nt = 0.87735
t3 = 915.14439 K
t4 = 1491.15354 K
F = 1.04601e+007 $/year

Constraints values at the optimum
t3-t2 = 319.74987 K
t4-t3 = 576.00915 K
t5-t3 = 72.53760 K
t6-t2 = 122.30680 K
t6-t9 = 232.18132 K
t7-t7min= 27.51967 K
t7p-t9-dtpmin = 1.65752 K

Πίνακας 16.8.3.1 Σύγκριση αποτελεσμάτων της μεθόδου SQP για τις ίδιες παραμέτρους με scaling και χωρίς scaling.

ΜΕΤΑΒΛΗΤΗ	ΜΕΘΟΔΟΣ	
	Matlab SQP	Matlab SQP (scaling of the variables)
r_c	8.52116	8.52118
n_c	0.84698	0.84698
n_t	0.87735	0.87735
T_3(K)	915.14447	915.14439
T_4(K)	1491.15354	1491.15354
F($/year)	$1.0460\ 10^7$	$1.0460\ 10^7$
t3-t2 (K)	319.75034	319.74987
t4-t3 (K)	576.00906	576.00915
t5-t3 (K)	72.53788	72.53760
t6-t2 (K)	122.30715	122.30680
t6-t9 (K)	232.18128	232.18132
t7-t7min (K)	27.51967	27.51967
t7p-t9-dtpmin(K)	1.65752	1.65752

Παρατηρήσεις

- Δυστυχώς, σε αντίθεση με ότι αναμενόταν, δεν πήραμε καλύτερα αποτελέσματα από τα ήδη υπάρχοντα με την μέθοδο της κλιμάκωσης.

- Παρόλα αυτά, με το συγκεκριμένο πρόβλημα δεν υπήρχαν προβλήματα πρόωρου τερματισμού του αλγορίθμου λόγω κακής επιλογής του αρχικού σημείου όπως προηγούμενα. Σχεδόν σε κάθε δοκιμή αρχικού σημείου, ακόμα και αν αυτό ήταν έξω από τα καθορισμένα όρια (lboundary,uboundary), είχαμε σύγκλιση. Με άλλα λόγια το scaling επέδρασε θετικά στο υπό μελέτη πρόβλημα και καλό θα είναι να εφαρμόζεται σε περιπτώσεις που η αντικειμενική συνάρτηση είναι πολύ ευαίσθητη σε μικρές μεταβολές των μεταβλητών της.

16.9 Γενικά συμπεράσματα

- Το matlab, μπορεί με αποδοτικό τρόπο να επιλύσει προβλήματα βελτιστοποίησης. Αυτό που το κάνει πολύτιμο είναι η ευκολία στη χρήση και στην διατύπωση των προβλημάτων. Όμως, το πρόβλημα πρέπει να έχει τεθεί σωστά, ώστε να αποφεύγονται οι δυσάρεστες εκπλήξεις.

- Η **καλή εκλογή του αρχικού σημείου** παίζει τον πιο σημαντικό ρόλο. Ένα καλό αρχικό σημείο κρατά το πρόβλημα στη γειτονιά της λύσης, όπου οι αριθμητικοί υπολογισμοί είναι πιο σταθεροί.

- Σημαντικό ρόλο στα προβλήματα βελτιστοποίησης παίζει η **κλιμάκωση**. Με χρήση της κλιμάκωσης γενικά παίρνουμε καλύτερα αποτελέσματα και είναι δυνατή η αποφυγή ορισμένων ασυνεχειών. Επίσης, σχεδόν πάντα η σύγκλιση είναι ταχύτερη.

- Από τις επιλογές του αλγορίθμου βελτιστοποίησης, το **DiffMinChange** επηρεάζει περισσότερο τη σύγκλιση. Σωστή χρήση του είναι σε θέση να αναιρέσει τυχόν ασυνέχειες.

- Αποφυγή συναρτήσεων που δεν είναι συνεχείς. Οι ασυνέχειες οδηγούν σε απόκλιση από τη λύση.

- Πρέπει να είμαστε βέβαιοι ότι η συνάρτηση δεν επιστρέφει τιμές **Inf** και **NaN** που συνήθως οδηγούν σε αποτυχία σύγκλισης. Οι συναρτήσεις **isreal**, **isnan** και **isfinite** μπορούν να χρησιμεύσουν για τον έλεγχο των αποτελεσμάτων.

- Όταν τα κριτήρια τερματισμού είναι πολύ αυστηρά, δημιουργούνται αρκετά προβλήματα στη σύγκλιση. Μια καλή λύση είναι η **χαλάρωση των κριτηρίων**.

- Όταν είναι εφικτό, ωφελεί πολύ τον αλγόριθμο βελτιστοποίησης από άποψη αριθμητικού σφάλματος, η παροχή από τον χρήστη των **διανυσμάτων κλίσης**, τόσο της αντικειμενικής συνάρτησης, όσο και των περιορισμών.

16.10 Εισαγωγικά στην ανάλυση ευαισθησίας του CGAM

Για να μπορέσουμε να μελετήσουμε το πόσο επηρεάζεται η βέλτιστη λύση από συγκεκριμένες παραμέτρους, γίνεται η ανάλυση ευαισθησίας. Γενικά επειδή, οι οικονομικοί παράγοντες είναι οι πιο αβέβαιοι, είναι σημαντικό να μελετήσουμε την επίδρασή τους. Στο παρόν κεφάλαιο, μελετήθηκε η επίδραση της τιμής του καυσίμου (φυσικού αερίου) στο βέλτιστο σχεδιασμό και στην τιμή της αντικειμενικής συνάρτησης.

Η ανάλυση ευαισθησίας πραγματοποιήθηκε για τιμές του φυσικού αερίου στο διάστημα από 2-8*10^{-6} $/KJ για απλό κόστος κεφαλαίου αλλά και για διπλό κόστος κεφαλαίου. Όλοι αυτοί οι υπολογισμοί μπορούν να γίνουν με το matlab πολύ εύκολα. Εκμεταλλευόμενοι μάλιστα και τα πανίσχυρα γραφικά του εργαλεία, μπορούμε να πάρουμε πολύ καλές γραφικές παραστάσεις.

Για την ανάλυση ευαισθησίας δημιουργήθηκαν 2 επιπλέον αρχεία, το cgam_sensitivity analysis.m το οποίο περιέχει τον αλγόριθμο για την ανάλυση ευαισθησίας και το cgam_objective_double.m το οποίο περιέχει την αντικειμενική συνάρτηση που αντιστοιχεί σε διπλάσιο κόστος κεφαλαίου.

Η λογική του προγράμματος συνίσταται στην δημιουργία ενός βρόχου FOR ο οποίος θα εκτελεί τον αλγόριθμο βελτιστοποίησης επαναληπτικά για κάθε μια τιμή του καυσίμου cf.

Τα αποτελέσματα του αλγορίθμου σε κάθε βήμα (βέλτιστο σημείο, αντικειμενική συνάρτηση) καταγράφονται σε ένα m-file υπό μορφή στηλών. Έπειτα τα δεδομένα του αρχείου εισάγονται στο χώρο εργασίας του matlab (matlab workspace) και κατασκευάζονται οι απαιτούμενες γραφικές παραστάσεις: rc(cf), nc(cf), nt(cf), t3(cf), t4(cf), h(cf).

Το πρόβλημα λύθηκε και με κλιμάκωση και χωρίς κλιμάκωση. Εδώ θα μελετηθεί η περίπτωση χωρίς κλιμάκωση.

16.11 Το αρχείο cgam_sensitivity analysis.m αναλυτικά

Το αρχείο cgam_sensitivity analysis.m περιέχει τον εξής κώδικα:

```
1  delete sens.m;
2  delete sens_double.m;

3  clear all
4  global w ms hu cpa cpg dt8p t8p h8 h8p h9 dtpmin p1 p7 p8 p9 ga gg;
5  global nb u t0 t1 t7min t8 t9 Ra Rg raa rag rb rr;
6  global t cf fcr feliniko c11 c12 c21 c22 c23 c24 c31 c32 c33 c34 c41 c51 c52 c53;
7  load data2.mat

8  format short e
9  var0=[6.0,0.8,0.85,900,1400];
10 %oria kai options
11 lboundary=[5.0,0.6,0.6,700.0,1200.0];
12 uboundary=[10.0,0.9,0.92,1000.0,1800.0];
13 options=optimset('LargeScale','off',...
14    'TolCon',0.0001,'TolX',0.0001,'DiffMinChange',1e-6);

15 for cf=[2:1:8]*1e-6;
16 [var,fval]=fmincon(@cgam_objective,var0,[],[],[],[],...
17    lboundary,uboundary,@cgam_constraints,options);
18 y=[cf,var(1),var(2),var(3),var(4),var(5),fval];
19 fid=fopen('sens.m','a+')
20 fprintf(fid,'%12.8f %8.4f %8.4f %8.4f %12.4f %12.4f %12.5e\n',y);
21 fclose(fid)
22 end

23 for cf=[2:1:8]*1e-6;
24 [var,fval]=fmincon(@cgam_objective_double,var0,[],[],[],[],...
25    lboundary,uboundary,@cgam_constraints,options);
26 y_double=[cf,var(1),var(2),var(3),var(4),var(5),fval];
27 fid=fopen('sens_double.m','a+')
28 fprintf(fid,'%12.8f %8.4f %8.4f %8.4f %12.4f %12.4f %12.5e\n',y_double);
29 fclose(fid)
30 end

31 load sens.m
32 load sens_double.m
33 subplot(2,3,1)
34 plot(sens(:,1),sens(:,2),'b-o'),title('FIG1 rc(cf)'),grid on,hold on
35 plot(sens_double(:,1),sens_double(:,2),'r-*')
```

```
36 xlabel('price of natural gas cf')
37 ylabel('rc')
38 hold off

39 subplot(2,3,2)
40 plot(sens(:,1),sens(:,3),'b-o',sens_double(:,1),sens_double(:,3),'*'),hold on
41 cfi=linspace(2e-6,8e-6,10);
42 nci=interp1(sens_double(:,1),sens_double(:,3),cfi,'cubic');
43 plot(cfi,nci,'m-'),hold off
44 title('FIG2 nc(cf)'),grid on
45 xlabel('price of natural gas cf'),ylabel('nc')

46 subplot(2,3,3)
47 plot(sens(:,1),sens(:,4),'b-o',sens_double(:,1),sens_double(:,4),'r-*')
48 title('FIG3 nt(cf)'),grid on
49 xlabel('price of natural gas cf'),ylabel('nt')

50 subplot(2,3,4)
51 plot(sens(:,1),sens(:,5),'b-o',sens_double(:,1),sens_double(:,5),'r-*')
52 title('FIG4 t3(cf)'),grid on
53 xlabel('price of natural gas cf'),ylabel('t3')

54 subplot(2,3,5)
55 plot(sens(:,1),sens(:,6),'b-o',sens_double(:,1),sens_double(:,6),'r-*')
56 title('FIG5 t4(cf)'),grid on
57 xlabel('price of natural gas cf'),ylabel('t4')

58 subplot(2,3,6)
59 plot(sens(:,1),sens(:,7),'b-o',sens_double(:,1),sens_double(:,7),'r-*')
60 title('FIG6 F(cf)'),grid on
61 xlabel('price of natural gas cf'),ylabel('F')
```

Γραμμές 1-2

```
1 delete sens.m;
2 delete sens_double.m;
```

Θα εξηγηθεί στις παρατηρήσεις (Ενότητα 16.13) το γιατί χρησιμοποιούνται αυτές οι 2 εντολές.

Γραμμές 3-14

```
3  clear all
4  global w ms hu cpa cpg dt8p t8p h8 h8p h9 dtpmin p1 p7 p8 p9 ga gg;
5  global nb u t0 t1 t7min t8 t9 Ra Rg raa rag rb rr;
6  global t cf fcr feliniko c11 c12 c21 c22 c23 c24 c31 c32 c33 c34 c41 c51 c52 c53;
7  load data2.mat

8  format short e
9  var0=[6.0,0.8,0.85,900,1400];
10 %oria kai options
11 lboundary=[5.0,0.6,0.6,700.0,1200.0];
12 uboundary=[10.0,0.9,0.92,1000.0,1800.0];
13 options=optimset('LargeScale','off',...
14    'TolCon',0.0001,'TolX',0.0001,'DiffMinChange',1e-6);
```

Οι γραμμές αυτές περιέχουν τα αναγκαία μεγέθη για τον αλγόριθμο βελτιστοποίησης όπως καταγράφονται επακριβώς και στο αρχείο cgam_optimization algorithm.m. Το μόνο αξιοσημείωτο είναι η χρήση της εντολής **clear**. Με αυτή την εντολή διαγράφονται αυτόματα, όλες μεταβλητές που έχουν αποθηκευτεί στο workspace μέχρι εκείνη τη στιγμή. Η χρήση της εντολής clear ελευθερώνει τη μνήμη και γενικά την «καθαρίζει» από μεταβλητές που ίσως να προέρχονται από προηγούμενα προβλήματα.

Σημείωση: Όταν έχει αποθηκευτεί η τιμή μιας μεταβλητής στο «workspace», και την δηλώσουμε με νέα τιμή στο «command prompt» με το ίδιο όνομα, τότε η τιμή της αρχικής μεταβλητής αντικαθίσταται.

Γραμμή 15

```
15 for cf=[2:1:8]*1e-6;
```

Εδώ έχουμε την έναρξη του βρόχου for. Οι τιμές που θα πάρει το cf έχουν καθοριστεί από το -50% έως το +100% της δοθείσας τιμής του καυσίμου που είναι $4*10^{-6}$ $/KJ.

Κατασκευάζεται ένας πίνακας γραμμή, του οποίου το 1ο στοιχείο είναι το 2 και το τελευταίο του στοιχείο το 8 με βήμα 1, δηλαδή 2, 3, 4…Τα στοιχεία αυτού του πίνακα πολλαπλασιάζονται με τον αριθμό 10^{-6}.

Ίσως κάποιος αναρωτηθεί, το πώς γίνεται να δίνονται νέες τιμές στο cf αφού η τιμή του είναι σταθερή και έχει αποθηκευτεί στο χώρο εργασίας από το αρχείο data2.mat που περιέχει τα δεδομένα. Η απάντηση είναι ότι σε κάθε επανάληψη του βρόχου for, η τιμή του cf αντικαθίσταται με τη νέα στο χώρο εργασίας του matlab. Εξάλλου, αφού η cf έχει δηλωθεί σαν «global», σε κάθε επανάληψη, οι τιμές της αντικειμενικής συνάρτησης και των περιορισμών, θα υπολογίζονται για την κάθε νέα τιμή του cf.

Γραμμές 16-18

```
16 | [var,fval]=fmincon(@cgam_objective,var0,[],[],[],[],...
17 |    lboundary,uboundary,@cgam_constraints,options);
18 | y=[cf,var(1),var(2),var(3),var(4),var(5),fval];
```

Αφού δηλώθηκε ο βρόχος for, το matlab λύνει επαναληπτικά το πρόβλημα βελτιστοποίησης (γραμμές 16-17) και σε κάθε επανάληψη το αποτέλεσμα (βέλτιστο σημείο, αντικειμενική συνάρτηση) αποθηκεύεται σε ένα πίνακα γραμμή με το όνομα y. Δηλαδή y=[cf,var(1),var(2),var(3),var(4),var(5),fval];

Γραμμή 19

```
19 | fid=fopen('sens.m','a+')
```

Με την εντολή **fopen** δημιουργείται το αρχείο sens.m στο οποίο θα αποθηκεύονται τα αποτελέσματα του αλγορίθμου σε κάθε βήμα. (Περισσότερες πληροφορίες για την fopen στο Παράρτημα Ε)

Χρησιμοποιείται το όρισμα 'a+' , διότι κάθε νέο αποτέλεσμα που προκύπτει σε κάθε βήμα, πρέπει να προστίθεται στα ήδη υπάρχοντα .

Αν χρησιμοποιούταν το όρισμα 'w+' τότε θα αποθηκευόταν μόνο το τελευταίο βέλτιστο σημείο, διότι με το όρισμα αυτό, σε κάθε βήμα αντικαθίστανται τα περιεχόμενα του αρχείου.

Το αποτέλεσμα της fopen αποθηκεύτηκε στο όρισμα fid. Το όρισμα αυτό είναι απαραίτητο για την επόμενη εντολή την fprintf η οποία θα μορφοποιήσει κατάλληλα τα αποτελέσματα.

Γραμμή 20

```
20 | fprintf(fid,'%12.8f %8.4f %8.4f %8.4f %12.4f %12.4f %12.5e\n',y);
```

Με αυτή την εντολή, μορφοποιούνται τα στοιχεία του πίνακα y, όσον αφορά τον τρόπο με τον οποίο αυτά θα εμφανίζονται σε μορφή στηλών στο αρχείο sens.m. (Περισσότερες πληροφορίες για την fprintf στο Παράρτημα Ε)

Το πρώτο στοιχείο του πίνακα y (το cf) θα έχει μέγεθος πεδίου 12 (μαζί με την υποδιαστολή) και ακρίβεια 8 ψηφίων μετά την υποδιαστολή. Το δεύτερο στοιχείο του y (rc) θα έχει μέγεθος πεδίου 8 και 4 ψηφία μετά την υποδιαστολή, κτλ.
Ο χαρακτήρας διαφυγής \n σημαίνει ότι σε κάθε βήμα τα αποτελέσματα θα εμφανίζονται σε ξεχωριστή γραμμή.

Γραμμή 21-22

```
21 | fclose(fid)
22 | end
```

Κλείνουμε το αρχείο sens.m που καθορίζεται από το όρισμα fid με την εντολή fclose και τερματίζουμε τον βρόχο for. (Περισσότερες πληροφορίες για την fclose στο Παράρτημα Ε)

Γραμμές 23-30

```
23 | for cf=[2:1:8]*1e-6;
24 | [var,fval]=fmincon(@cgam_objective_double,var0,[],[],[],[],...
25 |    lboundary,uboundary,@cgam_constraints,options);
26 | y_double=[cf,var(1),var(2),var(3),var(4),var(5),fval];
27 | fid=fopen('sens_double.m','a+')
28 | fprintf(fid,'%12.8f %8.4f %8.4f %8.4f %12.4f %12.4f %12.5e\n',y_double);
29 | fclose(fid)
30 | end
```

Δημιουργείται ένας νέος βρόχος for για την αντικειμενική συνάρτηση του αρχείου cgam_objective_double.m η οποία υπολογίζεται για διπλάσιο κόστος κτήσεως κεφαλαίου.

Χρησιμοποιείται ο ίδιος αλγόριθμος επίλυσης και οι ίδιες επιλογές με προηγούμενα. Η μόνη διαφορά σε σχέση με πριν, είναι ότι σε κάθε επανάληψη το αποτέλεσμα (βέλτιστο σημείο, αντικειμενική συνάρτηση) μορφοποιείται σε ένα πίνακα γραμμή με το όνομα y_double και τα αποτελέσματα του αλγορίθμου αποθηκεύονται σε ένα δεύτερο αρχείο το sens_double.m.

Γραμμές 31-32

```
31 | load sens.m
32 | load sens_double.m
```

Τα αποτελέσματα των δύο αρχείων που δημιουργήθηκαν, (sens.m και sens_double.m) πρέπει να αποθηκευτούν στο χώρο εργασίας του matlab προκειμένου τα δεδομένα τους να είναι προσβάσιμα για την σχεδίαση των απαιτούμενων γραφικών παραστάσεων. Με τις εντολές **load sens.m** και **load sens_double.m** πραγματοποιείται η αποθήκευση τους στο «matlab workspace» ως αριθμητικοί πίνακες 7Χ7.

Γραμμές 33-38

```
33 | subplot(2,3,1)
34 | plot(sens(:,1),sens(:,2),'b-o'),title('FIG1 rc(cf)'),grid on,hold on
35 | plot(sens_double(:,1),sens_double(:,2),'r-*')
36 | xlabel('price of natural gas cf')
37 | ylabel('rc')
38 | hold off
```

Πριν την ανάγνωση των παρακάτω σχολίων, καλό θα ήταν ο αναγνώστης να διαβάσει το ΠΑΡΑΡΤΗΜΑ Δ που αφορά στην δημιουργία γραφικών παραστάσεων.

Γραμμή 33

subplot(2,3,1) ← Δημιουργείται ένα υπογράφημα (subplot) το οποίο θα αποτελείται από 2 σειρές με 3 γραφήματα η κάθε μία, και **ενεργό γράφημα** το 1ο.

Γραμμή 34

```
plot(sens(:,1),sens(:,2), 'b-o' ),title('FIG1 rc(cf)'),grid on, hold on
```

- Με την εντολή plot σχεδιάζουμε τα δεδομένα από την 1η και την 2η στήλη του αρχείου sens.m.
- Σχεδιάζεται απλή γραμμή(-), με μπλε χρώμα (b) και το σύμβολο (o) στα δεδομένα.
- Δίδεται ο τίτλος του γραφήματος και ενεργοποιείται το πλέγμα (grid).
- Με την εντολή hold on κρατάμε ενεργή την συγκεκριμένη περιοχή, προκειμένου μία νέα εντολή plot να σχεδιάσει τα δεδομένα **στην ίδια περιοχή** και όχι σε διαφορετικό παράθυρο.

Γραμμή 35

```
plot(sens_double(:,1),sens_double(:,2), 'r-*' )
```

- Με την εντολή plot σχεδιάζουμε τα δεδομένα από την 1η και την 2η στήλη του αρχείου sens_double.m.
- Σχεδιάζεται απλή γραμμή(-), με κόκκινο χρώμα (r) και το σύμβολο (*) στα δεδομένα.

Γραμμές 36-37-38

xlabel('price of natural gas cf')

Δίδεται το όνομα του άξονα των x.

ylabel('rc')

Δίδεται το όνομα του άξονα των y.

hold off

Με την εντολή hold off απενεργοποιείται η περιοχή 1

Γραμμές 39-45

```
39 | subplot(2,3,2)
40 | plot(sens(:,1),sens(:,3),'b-o',sens_double(:,1),sens_double(:,3),'*'),hold on
41 | cfi=linspace(2e-6,8e-6,10);
42 | nci=interp1(sens_double(:,1),sens_double(:,3),cfi,'cubic');
43 | plot(cfi,nci,'m-'),hold off
44 | title('FIG2 nc(cf)'),grid on
45 | xlabel('price of natural gas cf'),ylabel('nc')
```

Εδώ παρουσιάζεται ένας άλλος τρόπος για τη δημιουργία 2 γραφικών παραστάσεων στο ίδιο παράθυρο, χωρίς την χρήση της εντολής plot άλλη μία φορά. Η εντολή hold on στη γραμμή 40 αναφέρεται στην γραμμή 43, όπου θέλουμε να σχεδιάσουμε και τρίτη γραφική παράσταση στο ίδιο παράθυρο που περιέχει την κυβική παρεμβολή για τα διάφορα σημεία.

Γραμμή 39

subplot(2,3,2)

Δημιουργείται ένα υπογράφημα (subplot) το οποίο θα αποτελείται από 2 σειρές με 3 γραφήματα η κάθε μία, και **ενεργό γράφημα** το 2ο.

Γραμμή 40

plot(sens(:,1),sens(:,3),'b-o',sens_double(:,1),sens_double(:,3),'*'),hold on

Από την 1η και την 3η στήλη του sens σχεδιάζεται απλή γραμμή(-), με μπλε χρώμα (b) και το σύμβολο (o) στα δεδομένα.
Από την 1η και την 3η στήλη του sens_double σχεδιάζεται μόνο το σύμβολο (*) στα δεδομένα.
Και οι 2 γραφικές παραστάσεις σχεδιάζονται **στο ίδιο παράθυρο.**

Με την εντολή hold on κρατάμε ενεργή την περιοχή 2, προκειμένου μία νέα εντολή plot να σχεδιάσει τα δεδομένα της **στην ίδια περιοχή** και όχι σε διαφορετικό παράθυρο.

Γραμμή 41

cfi=linspace(2e-6,8e-6,10);

Δημιουργείται ένας πίνακας για την τιμή του καυσίμου cfi που αποτελείται από 10 σημεία από το 2*10-6 έως 8*10-6 τα οποία απέχουν γραμμικά μεταξύ τους. Με αυτό τον τρόπο γίνεται εξομάλυνση των δεδομένων, κάτι που στη βιβλιογραφία ονομάζεται **data smoothing**. Προφανώς όσο περισσότερα σημεία πάρουμε, τόσο πιο εξομαλυμένη γραφική παράσταση θα έχουμε.

Γραμμή 42

nci=interp1(sens_double(:,1),sens_double(:,3),cfi,'cubic');

Η εντολή **yi = interp1(x,Y,xi)** επιστρέφει το διάνυσμα yi που περιέχει στοιχεία που ανταποκρίνονται στο διάνυσμα xi και καθορίζονται από παρεμβολή μεταξύ των διανυσμάτων x και Y. Το διάνυσμα x δίνει τα σημεία στα οποία ανταποκρίνονται τα δεδομένα στο διάνυσμα Y. Αν το Y είναι πίνακας, τότε η παρεμβολή πραγματοποιείται για κάθε στήλη του Y.
Η εντολή **yi = interp1(x,Y,xi,method)** κάνει παρεμβολή χρησιμοποιώντας μεθόδους όπως τις: **'nearest'** Nearest neighbor interpolation
'linear' Γραμμική παρεμβολή (προκαθορισμένη μέθοδος)
'spline' Παρεμβολή με κυβικά πολυώνυμα
'pchip' Παρεμβολή με κυβικά πολυώνυμα Hermite
'cubic' (Η ίδια μέθοδος όπως η 'pchip')

Σημείωση: Η παρεμβολή γίνεται κυρίως για λόγους πληρότητας και όχι για λόγους εμφάνισης του γραφήματος, καθώς η διαφορά στο γράφημα της κυβικής παρεμβολής σε σχέση με τη γραμμική παρεμβολή γίνεται αισθητή σε πολύ μεγάλη μεγέθυνση.

Γραμμή 43

plot(cfi,nci,'m-'),hold off

Δημιουργείται μια γραφική παράσταση από τα δεδομένα cfi nci, με απλή γραμμή και μώβ χρώμα. Με την εντολή hold off απενεργοποιείται η περιοχή 2.

Γραμμή 44

title('FIG2 nc(cf)'),grid on

Δίδεται ο τίτλος του γραφήματος και ενεργοποιείται το πλέγμα (grid).

Γραμμή 45

xlabel('price of natural gas cf'),ylabel('nc')

Δίδεται το όνομα του άξονα των x και των y.

Γραμμές 46-61

Στις γραμμές 46-61 επαναλαμβάνονται οι ίδιες εντολές. Αυτά που αλλάζουν είναι η ενεργή περιοχή του παραθύρου γραφημάτων, οι στήλες των 2 αρχείων που χρησιμοποιούνται σαν ορίσματα στην εντολή plot, καθώς και οι τίτλοι των υπολοίπων γραφημάτων. Επίσης σε αυτά τα δεδομένα δεν γίνεται παρεμβολή.

16.12 Τα αποτελέσματα του αλγορίθμου

Ανοίγουμε το cgam_sensitivity analysis.m και επιλέγουμε **edit>select all** και έπειτα **edit>copy**. Μετά πηγαίνουμε στο ««command prompt»» και επιλέγουμε **edit>paste** και πατάμε το **enter**.

Κατασκευάζονται από το πρόγραμμα τα δύο αρχεία sens.m και sens_double.m με τα εξής περιεχόμενα:

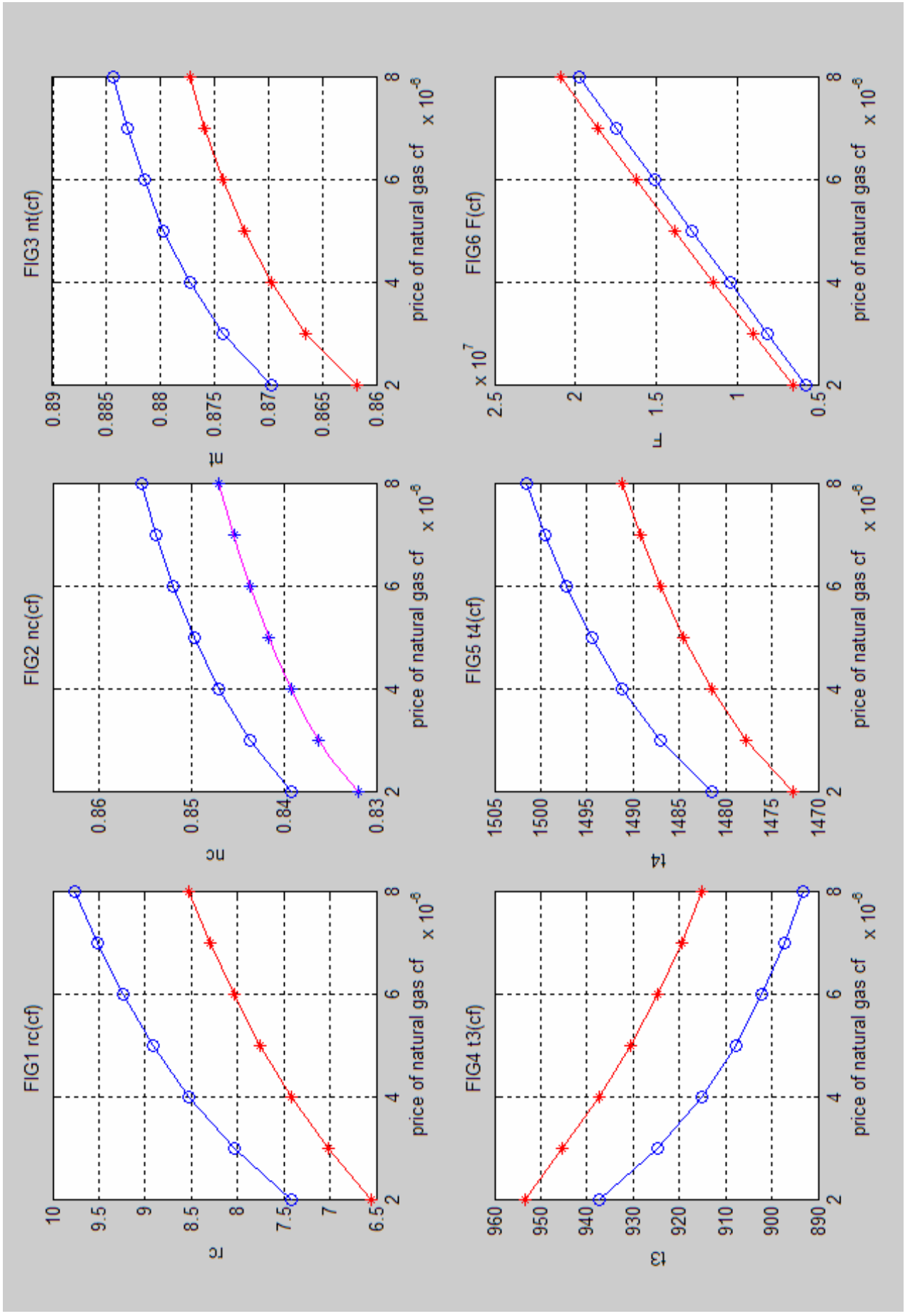
sens.m

cf	rc	nc	nt	t3	t4	F
0.00000200	7.4152	0.8392	0.8697	937.4881	1481.5098	5.72106e+006
0.00000300	8.0400	0.8436	0.8742	924.6267	1487.0521	8.10381e+006
0.00000400	8.5211	0.8470	0.8773	915.1645	1491.1616	1.04601e+007
0.00000500	8.9105	0.8497	0.8797	907.8802	1494.4415	1.27990e+007
0.00000600	9.2363	0.8519	0.8815	902.0319	1497.1679	1.51251e+007
0.00000700	9.5156	0.8538	0.8831	897.2387	1499.5266	1.74414e+007
0.00000800	9.7595	0.8554	0.8844	893.1565	1501.5840	1.97496e+007

sens_double.m

cf	rc	nc	nt	t3	t4	F
0.00000200	6.5479	0.8320	0.8618	953.3884	1472.7432	6.57404e+006
0.00000300	7.0203	0.8362	0.8665	945.5058	1477.7558	9.02731e+006
0.00000400	7.4152	0.8392	0.8697	937.4869	1481.5094	1.14421e+007
0.00000500	7.7503	0.8416	0.8722	930.5436	1484.5246	1.38333e+007
0.00000600	8.0399	0.8436	0.8742	924.6272	1487.0523	1.62076e+007
0.00000700	8.2945	0.8454	0.8759	919.5458	1489.2305	1.85691e+007
0.00000800	8.5212	0.8470	0.8773	915.1446	1491.1536	2.09202e+007

Ο αλγόριθμος συνεχίζεται αυτόματα και μας δίνει το γράφημα που παρουσιάζεται στην επόμενη σελίδα:



Εικόνα 16.12.1 Αποτελέσματα της ανάλυσης ευαισθησίας

Πίνακας 16.12.1 Ευαισθησία της βέλτιστης λύσης για την τιμή του καυσίμου και το κόστος κεφαλαίου

Μεταβολή	Τιμή καυσίμου	Κόστος κεφαλαίου
	+100%	+100%
$\Delta r_C^* / r_C^*$ (%)	+14.53	-12.79
$\Delta n_C^* / n_C^*$ (%)	+0.99	-0.92
$\Delta n_T^* / n_T^*$ (%)	+0.81	-0.87
$\Delta T_3^* / T_3^*$ (%)	-2.40	+2.44
$\Delta T_4^* / T_4^*$ (%)	+0.70	-0.65
$\Delta F^* / F^*$ (%)	+88.81	+9.38

16.13. Παρατηρήσεις

- Παρατηρήθηκε ότι για κάποια δοθέντα αρχικά σημεία ο αλγόριθμος εμφάνισε λανθασμένα αποτελέσματα για κάποιες τιμές του cf, μα το πρόγραμμα δεν σταμάτησε τη λειτουργία του πρόωρα. Για παράδειγμα χρησιμοποιήθηκε το αρχικό σημείο **var0=[6.0,0.8,0.85,900,1300]**. Το αρχείο sens περιείχε τα εξής:

0.00000200	5.0000	0.6844	0.9200	939.1267	1334.3180	-1.27573e+020
0.00000300	8.0400	0.8436	0.8742	924.6342	1487.0547	8.10381e+006
0.00000400	8.5212	0.8470	0.8773	915.1429	1491.1528	1.04601e+007
0.00000500	10.0000	0.9000	0.9200	800.8157	1476.4690	-7.51638e+019
0.00000600	9.2363	0.8519	0.8815	902.0296	1497.1677	1.51251e+007
0.00000700	9.5158	0.8538	0.8831	897.2154	1499.5145	1.74414e+007
0.00000800	9.7595	0.8554	0.8843	893.1928	1501.5980	1.97496e+007

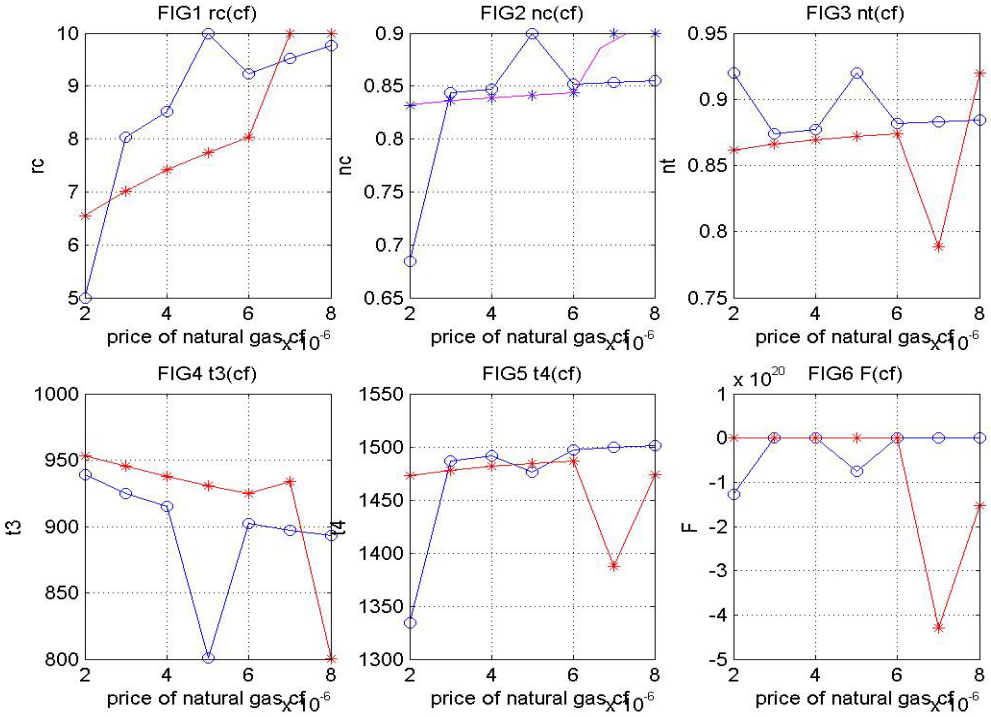
Συγκεκριμένα, για τις τιμές του καυσίμου $2*10^{-6}$ $/KJ και $5*10^{-6}$ $/KJ εμφανιζόταν το μήνυμα **"maximum number of iterations exceeded"** που σημαίνει ότι ο αλγόριθμος δεν βρήκε το βέλτιστο για τον προκαθορισμένο αριθμό επαναλήψεων.

Το αρχείο sens_double περιείχε τα εξής:

0.00000200	6.5479	0.8320	0.8618	953.3883	1472.7433	6.57404e+006
0.00000300	7.0203	0.8362	0.8665	945.5069	1477.7562	9.02731e+006
0.00000400	7.4152	0.8392	0.8697	937.4898	1481.5105	1.14421e+007
0.00000500	7.7503	0.8416	0.8722	930.5419	1484.5239	1.38333e+007
0.00000600	8.0400	0.8436	0.8742	924.6274	1487.0524	1.62076e+007
0.00000700	10.0000	0.9000	0.7884	933.7680	1387.8650	-4.29519e+020
0.00000800	10.0000	0.9000	0.9200	800.0286	1473.9757	-1.53112e+020

Συγκεκριμένα, για τις τιμές του καυσίμου $7*10^{-6}$ $/KJ και $8*10^{-6}$ $/KJ εμφανιζόταν το μήνυμα **" Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 3.190339e-019."**

Προέκυψε το Σχ. 16.13.1



Σχήμα 16.13.1 Παράδειγμα αποτυχίας του αλγορίθμου

- **Με χρήση της κλιμάκωσης** είχαμε πάλι ευεργετικά αποτελέσματα όσον αφορά στη σύγκλιση και στην ακρίβεια της λύσης. Για το σημείο **var0=[60,80,85,90,130]** δεν παρουσιάστηκε κάποιο πρόβλημα. Το ίδιο και για τα σημεία: **var0=[60,70,85,90,150], var0=[60,70,85,90,130], var0=[60,80,70,90,130];**

  Με αρχικό σημείο **var0=[60,80,70,80,130]** και τιμή της παραμέτρου **'DiffMinChange',** ίση με **'1e-6'** το πρόγραμμα δεν μπόρεσε να βρει λύση. Με ρύθμιση της παραμέτρου, **'DiffMinChange'** ίση με **'1e-4'** το πρόβλημα λύθηκε αν και με λίγο μικρότερη ακρίβεια απ' ότι προηγούμενα.

  Μόνο στο σημείο **var0=[60,80,85,80,150] το πρόγραμμα δεν ανταποκρίθηκε.**

  Γενικά, με την μέθοδο της κλιμάκωσης, μπορούμε να ξεπεράσουμε αρκετά προβλήματα που δημιουργούνται. Η μέθοδος αυτή προτείνεται και στα ηλεκτρονικά αρχεία βοήθειας του "optimization toolbox" [8] στο κεφάλαιο "typical problems and how to deal with them" στη σελίδα 2-76.

- Ένα ακόμα ευεργετικό αποτέλεσμα με χρήση της κλιμάκωσης είναι ο **χρόνος σύγκλισης** (για το ίδιο αρχικό σημείο, όρια στις μεταβλητές, επιλογές). Για να γίνει η σύγκριση πληκτρολογήθηκαν οι παρακάτω εντολές στο "«command window»" :

```
profile on
clear all
global w ms hu cpa cpg dt8p t8p h8 h8p h9 dtpmin p1 p7 p8 p9 ga gg;
global nb u t0 t1 t7min t8 t9 Ra Rg raa rag rb rr;
global t cf fcr feliniko c11 c12 c21 c22 c23 c24 c31 c32 c33 c34 c41 c51 c52 c53;
load data2.mat
var0=[6.0,0.8,0.85,900,1400];
%oria kai options
lboundary=[5.0,0.6,0.6,700.0,1200.0];
uboundary=[10.0,0.9,0.92,1000.0,1800.0];
options=optimset('LargeScale','off',...
   'TolCon',0.0001,'TolX',0.0001,'DiffMinChange',1e-6);
for cf=[2:1:8]*1e-6
[var,fval]=fmincon(@cgam_objective,var0,[],[],[],[],...
   lboundary,uboundary,@cgam_constraints,options);
end
profile report
```

Κατασκευάστηκε ένα αρχείο «htm» με διάφορες πληροφορίες μεταξύ των οποίων ήταν και ο χρόνος που έκανε να εκτελεστεί η fmincon:

MATLAB Profile Report: Summary

Report generated 02-Jul-2006 21:03:18

Total recorded time:	11.08 s
Number of M-functions:	22
Number of M-subfunctions:	3
Number of MEX-functions:	1
Clock precision:	0.0000001 s
Clock Speed:	858 Mhz

Function List

Name	Time		Calls	Time/call	Self time		Location
fmincon	11.0060000	99.4%	7	1.57228571429	0.2400000	2.2 %	C:/MATLAB6p5/toolbox/optim/fmincon.m

Η ίδια διαδικασία **για το πρόβλημα με την κλιμάκωση**, δίδει τα παρακάτω αποτελέσματα για την fmincon:

MATLAB Profile Report: Summary

Report generated 02-Jul-2006 21:14:36

Total recorded time:	8.38 s
Number of M-functions:	22
Number of M-subfunctions:	3
Number of MEX-functions:	1
Clock precision:	0.0000001 s
Clock Speed:	900 Mhz

Function List

Name	Time		Calls	Time/call	Self time		Location
fmincon	8.3220000	99.3%	7	1.18885714286	0.2100000	2.5%	C:/MATLAB6p5/toolbox/optim/fmincon.m

Παρατηρείται μεγάλη διαφορά στο χρόνο εκτέλεσης. Για το πρόβλημα χωρίς κλιμάκωση, η fmincon εκτελείται σε **11sec** ενώ για το πρόβλημα με κλιμάκωση σε **8.322sec.**

- Κάθε φορά που ο χρήστης θέλει να κάνει ανάλυση ευαισθησίας στο πρόβλημα, θα πρέπει **να διαγράφει** τα αρχεία sens.m και sens_double.m διότι όταν εκτελεστεί ο αλγόριθμος, τα αποτελέσματα δεν θα αποθηκευτούν από την αρχή στα αρχεία αυτά, αλλά θα προστεθούν ως νέες γραμμές στα ήδη υπάρχοντα με αποτέλεσμα να γίνεται λάθος στον σχεδιασμό των γραφημάτων. Γι' αυτό και στην αρχή του προγράμματος τοποθετούμε τις εντολές:
  delete sens.m και
  delete sens_double.m

16.14. Γενικά συμπεράσματα

- Το matlab παρέχει **πανίσχυρα γραφικά εργαλεία** με εξαιρετικές δυνατότητες απεικόνισης, τόσο για διδιάστατα όσο και για τρισδιάστατα γραφικά.

- Εξαιρετικά μεγάλη **ευκολία κατασκευής γραφημάτων**.

- Δυνατότητα **πολυωνυμικής παρεμβολής** για καλύτερα αποτελέσματα όσον αφορά και τη διδιάστατη (y=f(x)) αλλά και την τρισδιάστατη (z=f(x,y)) περίπτωση. Επίσης, παρέχεται μεγάλη ποικιλία μεθόδων παρεμβολής (cubic splines, Hermite polynomials, nearest neighbor interpolation κ.α)

- Σε μερικές περιπτώσεις, είναι αναγκαία η προσαρμογή μιας καμπύλης σε κάποια μετρούμενα δεδομένα (προβλήματα **curve-fitting**). Τα προβλήματα curve-fitting λύνονται πολύ εύκολα με το matlab.

- Η mathworks παρέχει καταπληκτικό «user community», όπου μπορεί κάποιος να ανταλλάξει απόψεις και να κάνει ερωτήσεις για τα προβλήματα που αντιμετωπίζει με το matlab.

17. ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΕΡΙΟΣΤΡΟΒΙΛΟΥ ΜΕ ΧΡΗΣΗ ΤΟΥ MATLAB

Στο κεφάλαιο αυτό προσπαθούμε να εφαρμόσουμε όσα αναφέρθηκαν στα προηγούμενα κεφάλαια εξετάζοντας το πρόβλημα του απλού αεριοστροβίλου. Στόχος είναι, η κλήση από το matlab της υπορουτίνας της fortran που υπολογίζει την συνάρτηση κόστους του συστήματος του απλού αεριοστροβίλου και αυτή να προσπαθήσουμε να την ελαχιστοποιήσουμε με το matlab.

Περισσότερες πληροφορίες για το πρόβλημα και τα δεδομένα του υπάρχουν στο παράρτημα Β.

Ο «compiler» που χρησιμοποιήθηκε για το συγκεκριμένο πρόβλημα ήταν ο Compaq Visual Fortran 6.6. Το matlab ενημερώθηκε για την παρουσία του «compiler» μέσω της γνωστής εντολής:

>>mex -setup

Παρατηρήσεις:

- Για λόγους παρουσίασης, το δοθέν από τον επιβλέποντα αρχείο GTG92.for τροποποιήθηκε, με την αφαίρεση του υπόλοιπου κώδικα εκτός του προγράμματος REAL FUNCTION. Έπειτα προστέθηκε ένας νέος κώδικας (το λεγόμενο interface) ο οποίος πραγματοποίησε τη σύνδεση matlab και fortran.

- Όλες οι μεταβλητές δηλώθηκαν σαν REAL*8 (Double precision)

17.1. Το αρχείο cost_realfunction.f

```
1       real*8 function F(RPC,TR3,HC,RPB,HT)

2       real*8 W,CF,HU,P1,T1,CP,RG,RK,HR,C11,C12,C21,C22,C23,
3      * C24,C25,C31,C32,C33,C34,C35,RPM,TRM
4       real*8 RR,TR2,RPT,TR4,MA,MF,Z1,Z2,Z3,ZF
5       real*8 RPC,TR3,HC,RPB,HT,CRF,MIR,N,FCR,FM,FE

6       W=10000.0
7       HU=42500.0
8       T1=293.0
9       HR=7500.0
10      RPM=100.0
11      TRM=5.2
12      P1=1.000
13      CF=0.200
14      CP=1.000
15      RG=0.287
16      RK=1.400
17      C11=39.5
18      C12=0.900
19      C21=25.600
20      C22=0.995
21      C23=5.000
22      C24=0.018
23      C25=28.000
24      C31=266.300
25      C32=0.920
26      C33=5.000
27      C34=0.036
28      C35=56.000
29      MIR=0.100
30      N=20.0
31      FM=0.015
32      FE=0.025

33      CRF = MIR*(1.+MIR)**N/((1.+MIR)**N-1.)
34      FCR = CRF + FM + FE
35      C11 = .001*FCR*C11
36      C21 = .001*FCR*C21
37      C31 = .001*FCR*C31
38      RR  = 1.0-1.0/RK
```

```
39      TR2 = 1.0+(RPC**RR-1.0)/HC
40      RPT = RPB*RPC
41      TR4 = TR3*(1.0-HT*(1.0-RPT**(-RR)))
42      MA = W/(CP*T1*(TR3-TR4-TR2+1.0))
43      MF = MA*CP*T1*(TR3-TR2)/HU
44      Z1 = C11*MA*RPC*dlog(RPC)/(C12-HC)
45      Z2 = C21*MA*(1.+C23*dexp(C24*TR3*T1-C25))/(C22-RPB)
46      Z3 = C31*MA*dlog(RPT)*(1.0+C33*dexp(C34*TR3*T1-C35))/(C32-HT)
47      ZF = 3.6*CF*MF*HR
48      F  = Z1+Z2+Z3+ZF
49      return
50      end

51    C   -------------------------
52    C   THE GATEWAY ROUTINE
53    C   -------------------------
54      subroutine mexFunction(nlhs,plhs,nrhs,prhs)
55    C   ---------------------------
56    C  DECLARATION OF THE VARIABLES
57    C   ---------------------------
58      integer plhs(*), prhs(*)
59      integer nlhs, nrhs
60      integer mxGetPr, mxCreateDoubleMatrix, mxGetM, mxGetN
61      integer m, n, size
62      integer F_PR, RPC_PR, TR3_PR, HC_PR, RPB_PR, HT_PR
63      real*8 FD, RPC, TR3, HC, RPB, HT
64    C   ------------------------------------
65    C  CHECK FOR PROPER NUMBER OF ARGUMENTS
66    C   ------------------------------------
67      If(nrhs.ne.5) then
68      call mexErrMsgTxt('5 ARGUMENTS REQUIRED')
69      elseif(nlhs.ne.1) then
70      call mexErrMsgTxt('1 OUTPUT ARGUMENT REQUIRED')
71      endif
72    C   ---------------------------
73    C   CREATE POINTERS
74    C   ---------------------------
75      RPC_PR=mxGetPr(prhs(1))
76      TR3_PR=mxGetPr(prhs(2))
77      HC_PR=mxGetPr(prhs(3))
78      RPB_PR=mxGetPr(prhs(4))
79      HT_PR=mxGetPr(prhs(5))
80      m=mxGetM(prhs(1))
81      n=mxGetN(prhs(1))
```

```
82      size=m*n
83      if (size.gt.1) then
84      call mexErrMsgTxt('THIS IS NOT A SCALAR VARIABLE')
85      endif
86    C  ----------------------------
87    C  CREATE OUTPUT
88    C  ----------------------------
89      plhs(1)=mxCreateDoubleMatrix(m,n,0)
90      F_PR=mxGetPr(plhs(1))
91      call mxCopyPtrToReal8(RPC_PR, RPC, size)
92      call mxCopyPtrToReal8(TR3_PR, TR3, size)
93      call mxCopyPtrToReal8(HC_PR, HC, size)
94      call mxCopyPtrToReal8(RPB_PR, RPB, size)
95      call mxCopyPtrToReal8(HT_PR, HT, size)
96    C  ----------------------------------------------------
97    C   CALL THE COMPUTATIONAL SUBROUTINE
98    C       ----------------------------------------------------
99      FD=F(RPC,TR3,HC,RPB,HT)
100   C  ----------------------------------------------------
101   C  LOAD THE DATA INTO F_PR WHICH IS THE OUTPUT TO MATLAB
102   C       ----------------------------------------------------
103     call mxCopyReal8ToPtr(FD,F_PR,size)

104     return
105     end
```

Γραμμές 1-5

Εδώ δηλώνεται η πραγματική συνάρτηση F καθώς και όλες οι μεταβλητές με διπλή ακρίβεια **(real*8)**. Προφανώς δεν δημιουργείται πρόβλημα από τη χρήση των μικρών γραμμάτων αφού η fortran δεν κάνει διάκριση μεταξύ κεφαλαίων και μικρών γραμμάτων.

Γραμμές 6-50

Σε αυτές τις γραμμές δίδονται τιμές στα σταθερά μεγέθη του προβλήματος και παρατίθενται οι σχέσεις που οδηγούν στον υπολογισμό της συνάρτησης F. Προφανώς, όπως σε κάθε πρόγραμμα REAL FUNCTION ή SUBROUTINE περιέχονται οι εντολές **return**, **end**, για επιστροφή στο σημείο απ' όπου κλήθηκε η συνάρτηση ή η υπορουτίνα.

Γραμμές 51-54

Η εντολή **subroutine mexFunction(nlhs,plhs,nrhs,prhs)** είναι στην ουσία το σημείο έναρξης στη σύνδεση της υπολογιστικής ρουτίνας με το matlab. Η εντολή αυτή γράφεται αυτούσια σε κάθε αρχείο MEX της fortran. Τα ορίσματα nlhs, plhs, nrhs, prhs έχουν αναλυθεί στην ενότητα 15.7.

Γραμμές 55-63

Οι εντολές στις γραμμές 58-59:

Integer plhs(*), prhs(*)
Integer nlhs, nrhs

μπαίνουν πάντα αυτούσιες σε κάθε αρχείο MEX. Εξασφαλίζουν ότι ο αριθμός των ορισμάτων στην είσοδο και την έξοδο, καθώς και οι δείκτες στα δεδομένα εισόδου και εξόδου του mxArray είναι ακέραιοι.

Στις γραμμές 60-62 δηλώνονται ως ακέραιοι (integers):

- όλες οι συναρτήσεις mx που θα χρησιμοποιήσουμε για τη διαχείριση των δεδομένων του mxArray.
- Οι μεταβλητές m (αριθμός γραμμών mxArray), n (αριθμός στηλών mxArray) και size=m*n
- Οι δείκτες για τα δεδομένα του mxArray που θα προκύψουν από τη χρήση της εντολής mxGetPr για κάθε όρισμα εισόδου (F_PR, RPC_PR, TR3_PR, HC_PR, RPB_PR, HT_PR)

Στη γραμμή 63 δηλώνονται ως real*8 οι ανεξάρτητες μεταβλητές του προβλήματος (RPC, TR3, HC, RPB, HT) και το αποτέλεσμα από την κλήση της συνάρτησης (FD) για τις τιμές αυτών των μεταβλητών.

Γραμμές 64-71

Οι εντολές στις γραμμές 64-71 κάνουν έναν έλεγχο για τον αριθμό των ορισμάτων που δίνει ο χρήστης μέσα από το «command prompt».

Αν ο αριθμός των ορισμάτων εισόδου (nrhs) είναι μικρότερος από 5, τότε καλείται η υπορουτίνα **mexErrMsgTxt** η οποία εμφανίζει το μήνυμα "5 INPUT ARGUMENTS REQUIRED" και προκαλεί τερματισμό του προγράμματος.

Επίσης αν ο αριθμός των ορισμάτων εξόδου (nlhs) είναι μικρότερος από 1, τότε καλείται η υπορουτίνα **mexErrMsgTxt** η οποία εμφανίζει το μήνυμα "1 OUTPUT ARGUMENT REQUIRED" και προκαλεί τερματισμό του προγράμματος.

Για παράδειγμα, αν στο «command prompt» πληκτρολογήσουμε:

```
>> fd=cost_realfunction(20,4.8)
```

Θα πάρουμε το μήνυμα λάθους:
??? 5 ARGUMENTS REQUIRED

Error in ==>
C:\MATLAB6p5\work\ΠΤΥΧΙΑΚΗ\callingFORTRANfromMATLAB\cost_realfunction.dll

Γραμμές 72-79

Οι εντολές στις γραμμές 72-79 για κάθε δείκτη στα ορίσματα εισόδου του mxArray δίνουν τα πραγματικά δεδομένα του (real data elements). Αυτό γίνεται με την εντολή mxGetPr (mx get pointer to real data).

Συντάσσεται ως εξής: **mxGetPr(pointer).**

Η εντολή αυτή, επιστρέφει την «διεύθυνση» του πραγματικού δεδομένου (real data) του mxArray. Αν δεν υπάρχουν πραγματικά δεδομένα τότε επιστρέφει την τιμή NULL.

Για παράδειγμα, η εντολή **HC_PR=mxGetPr(prhs(3))** επιστρέφει τη «διεύθυνση» του τρίτου ορίσματος εισόδου του mxArray στην ακέραια μεταβλητή HC_PR.

Γραμμές 80-85

Σε αυτό το σημείο γίνεται επίδειξη των εντολών mxGetM και mxGetN. Αυτές οι εντολές δίνουν αντίστοιχα, τον αριθμό των γραμμών και των αριθμό των στηλών κάποιου ορίσματος του mxArray. Προφανώς στο συγκεκριμένο πρόβλημα, επειδή τα ορίσματα εισόδου είναι νούμερα και όχι πίνακες δεν είναι απαραίτητη η χρήση αυτών των εντολών.

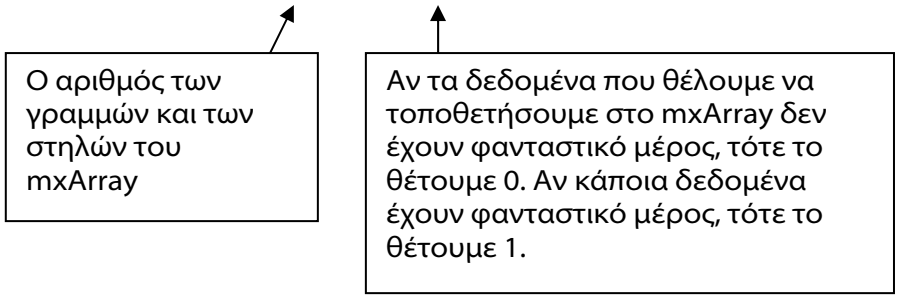
```
m=mxGetM(prhs(1))
n=mxGetN(prhs(1))
size=m*n
if (size.gt.1) then
call mexErrMsgTxt('THIS IS NOT A SCALAR VARIABLE')
endif
```

Εδώ, παίρνουμε τον αριθμό των γραμμών και των στηλών του πρώτου ορίσματος του mxArray. Δημιουργείται η νέα μεταβλητή size και γίνεται έλεγχος για το μέγεθος του πίνακα. Αν είναι μεγαλύτερος του 1 (που σημαίνει ότι δεν είναι σταθερά) τότε καλείται η υπορουτίνα mexErrMsgTxt προβάλλοντας το μήνυμα "THIS IS NOT A SCALAR VARIABLE" και το πρόγραμμα τερματίζεται.

Γραμμές 86-90

Πριν κληθεί η υπορουτίνα για τον υπολογισμό της συνάρτησης, πρέπει να δημιουργηθεί ο χώρος για το όρισμα εξόδου. Έτσι γίνεται χρήση της εντολής mxCreateDoubleMatrix. Όπως υποδεικνύει και το όνομα, με την εντολή αυτή δημιουργείται ένα νέο mxArray, διδιάστατο με στοιχεία διπλής ακρίβειας (Double precision). Συντάσσεται ως εξής:

mxCreateDoubleMatrix(m, n, ComplexFlag)



Αν η διαδικασία είναι επιτυχής, η mxCreateDoubleMatrix δίνει ένα δείκτη στο δημιουργούμενο mxArray. Αν κάτι πάει στραβά, η mxCreateDoubleMatrix δίνει 0.

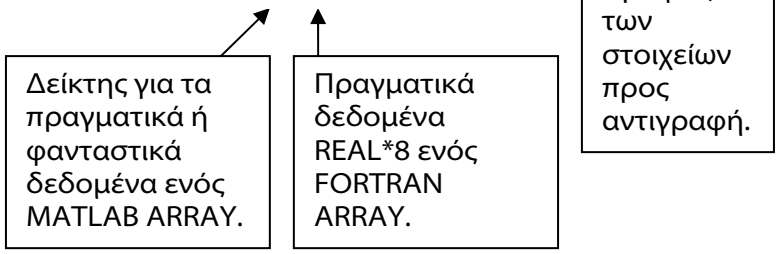
Μετά την δημιουργία του δείκτη για το νέο mxArray, πρέπει να πάρουμε τη «διεύθυνση» για τα πραγματικά δεδομένα. Αυτό γίνεται με την εντολή **F_PR=mxGetPr(plhs(1))** η οποία εξηγήθηκε προηγούμενα.

Γραμμές 91-95

Μέχρι στιγμής, αυτό που έχουμε είναι διάφορα mxArrays. Ας θυμηθούμε ότι τα mxArrays είναι ουσιαστικά πίνακες του matlab. Για να μπορέσει η υπορουτίνα να πραγματοποιήσει τους υπολογισμούς, τα mxArrays πρέπει να μετατραπούν σε Fortran Arrays. Αυτή η διαδικασία είναι απαραίτητη κάθε φορά πριν κληθεί η ρουτίνα υπολογισμών. Η μετατροπή γίνεται με τη χρήση της υπορουτίνας **mxCopyPtrToReal8**.

Συντάσσεται ως εξής:

mxCopyPtrToReal8(px, y, n)



Η υπορουτίνα mxCopyPtrToReal8 αντιγράφει **n** δεδομένα REAL*8 από τον πίνακα του MATLAB τα οποία δίδονται από το δείκτη **px**, στον REAL*8 πίνακα **y** της Fortran.

Για παράδειγμα, η εντολή **call mxCopyPtrToReal8(RPC_PR, RPC, size)**, αντιγράφει size δεδομένα real*8 από τον πίνακα του matlab με το δείκτη RPC_PR σε ένα real*8 πίνακα της fortran RPC.
Προφανώς επειδή το PRC είναι σταθερά και όχι πίνακας, μπορούσαμε να γράψουμε:
call mxCopyPtrToReal8(RPC_PR, RPC, 1).

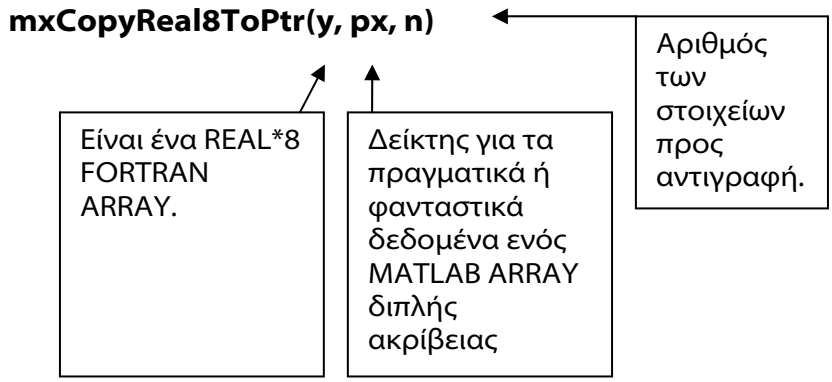
Γραμμές 96-99

Αφού γίνουν οι απαραίτητες αντιγραφές για όλα τα ορίσματα εισόδου, μπορούμε να καλέσουμε το πρόγραμμα REAL FUNCTION και να υπολογίσουμε την τιμή της F. Η τιμή αυτή αποθηκεύεται στη real*8 μεταβλητή με το όνομα FD.

Γραμμές 100-105

Η μεταβλητή FD που δημιουργήθηκε, αποτελεί ένα fortran Array. Αυτό σημαίνει ότι η τιμή της δεν μπορεί να εμφανιστεί στο «command prompt» του matlab. Γι' αυτό πρέπει τώρα να γίνει η αντίστροφη διαδικασία, μετατροπής του fortran array σε matlab array. Αυτό γίνεται με κλήση της υπορουτίνας mxCopyReal8ToPtr.

Συντάσσεται ως εξής:



Η υπορουτίνα mxCopyReal8ToPtr αντιγράφει **n** δεδομένα REAL*8 από τον πίνακα **y** της Fortran σε ένα πίνακα του matlab που υποδεικνύεται από τον δείκτη **px**.
Έτσι, με την εντολή **call mxCopyReal8ToPtr(FD,F_PR,size),** αντιγράφτηκαν τα πραγματικά δεδομένα του πίνακα FD στο δείκτη F_PR που αποτελεί και την έξοδο του matlab.

17.2. Μετάφραση του cost_realfunction.f (compiling)

Μετά από αυτά, το αρχείο cost_realfunction.f πρέπει να μεταφραστεί, ώστε να δημιουργηθεί το απαραίτητο αρχείο cost_realfunction.dll. Αφού ρυθμίσουμε τις επιλογές του μεταφραστή (Βλέπε Ενότητα 15.4.1), πληκτρολογούμε στο «command prompt»:

>>mex cost_realfunction.f

Το αρχείο cost_realfunction.dll που δημιουργείται, αποτελεί το αρχείο σύνδεσης με το matlab.

Μπορούμε πλέον, να χειριστούμε το μεταφρασμένο αρχείο της fortran σαν απλό m-file του matlab. Μπορούμε να πληκτρολογήσουμε:

```
>>fd=cost_realfunction(15,4.8,0.85,0.98,0.88)
```

Παίρνουμε το αποτέλεσμα:

```
fd =

  3.7804e+003
```

17.3. Το αρχείο υπολογισμού της αντικειμενικής συνάρτησης openc.m

Στην σύνταξη της εντολής fmincon χρειαζόμαστε το αρχείο στο οποίο είναι αποθηκευμένη η αντικειμενική συνάρτηση (όρισμα fun).

[x,fval,exitflag,output,lambda,grad,hessian]=
fmincon(fun,x0,A,b,Aeq,Beq,lb,ub,nonlcon,options,P1,P2…)

Αν συντάξουμε την fmincon ως εξής:

…=fmincon(@cost_realfunction,var0,…)

Θα εμφανιστεί το μήνυμα λάθους:

??? Error using ==> fmincon
FMINCON cannot continue because user supplied objective function
failed with the following error:

Error using ==> feval
5 ARGUMENTS REQUIRED

Για να είναι σωστή η σύνταξη της fmincon, η αντικειμενική συνάρτηση πρέπει να είναι δηλωμένη στο αρχείο cost_realfunction.m. Εμείς όμως αυτό που έχουμε είναι το αρχείο σύνδεσης cost_realfunction.dll.
Γι' αυτό το λόγο, πρέπει να κατασκευάσουμε ένα m-file, το οποίο σε κάθε επανάληψη θα δίνει τα 5 ορίσματα εισόδου στο cost_realfunction.dll και θα επιστρέφει την τιμή της αντικειμενικής συνάρτησης.

Κατασκευάζουμε λοιπόν, το αρχείο openc.m που περιέχει τον εξής απλό κώδικα:

```
1  function F=openc(var)
2  RPC=var(1);
3  TR3=var(2);
4  HC=var(3);
5  RPB=var(4);
6  HT=var(5);
7  F=feval('cost_realfunction', RPC, TR3, HC, RPB, HT);
```

Στο αρχείο αυτό, δηλώνουμε κανονικά την αντικειμενική συνάρτηση F όπως θα κάναμε σε οποιοδήποτε πρόβλημα. Η μόνη διαφορά είναι ότι αντί να γράψουμε «τον τύπο» της F χρησιμοποιούμε την συνάρτηση **feval** η οποία πραγματοποιεί "function evaluation" του αρχείου cost_realfunction.dll για τα 5 ορίσματα εισόδου. Το αποτέλεσμα επιστρέφει στο όρισμα εξόδου F. (περισσότερες πληροφορίες για την feval στην ενότητα 9.4)

Πλέον, μπορούμε να συντάξουμε την εντολή fmincon ως εξής:

…=fmincon(@openc,var0,…)

χωρίς να παρουσιαστεί κάποιο πρόβλημα στους υπολογισμούς.

17.4. Το αρχείο των περιορισμών openc_constraints.m

Είναι απαραίτητο, να δημιουργηθούν και κάποιοι περιορισμοί όσον αφορά το πρόβλημα, καθότι μπορούμε εύκολα να οδηγηθούμε σε αρνητικές τιμές της συνάρτησης κόστους που θέλουμε να ελαχιστοποιήσουμε. Οι περιορισμοί του προβλήματος (Παράρτημα Β.2) είναι οι εξής:

TR2>1
TR3>TR2
TR3>TR4
TR3-TR4-TR2+1>0 (εξασφαλίζει θετική παροχή μάζας αέρα)

Σημείωση: στην πραγματικότητα μόνο ο τελευταίος περιορισμός να χρησιμοποιηθεί εξασφαλίζει το πρόβλημα, αφού εμπεριέχει κατά κάποιον τρόπο τους προηγούμενους. Το πρόβλημα επιλύεται σωστά, και μόνο με την παράθεση του 4ου περιορισμού.

Κατασκευάζουμε λοιπόν, το αρχείο openc_constraints.m που περιέχει τον εξής κώδικα:

```
1  function [c,ceq,DC,DCeq]=openc_constraints(var)
2  RPC=var(1);
3  TR3=var(2);
4  HC=var(3);
5  RPB=var(4);
6  HT=var(5);

7  load parameters.mat

8  %SXESEIS METATROPHS
9  CRF=MIR*(1+MIR)^N/((1+MIR)^N-1);
10 FCR=CRF+FM+FE;
11 C11=0.001*FCR*C11;
12 C21=0.001*FCR*C21;
13 C31=0.001*FCR*C31;
14 RR=1-1/RK;
15 TR2=1+(RPC^RR-1)/HC;
16 RPT = RPB*RPC;
17 TR4 = TR3*(1-HT*(1-RPT^(-RR)));
18 MA = W/(CP*T1*(TR3-TR4-TR2+1));
19 MF = MA*CP*T1*(TR3-TR2)/HU;

20 %PERIORISMOI
21 c=[1-TR2
22    TR2-TR3
23    TR4-TR3
24    TR4-TR3-1+TR2];
```

```
25  ceq=[ ];

26  DCeq=[ ];

27  DC=[(-RR*RPC^(RR-1))/HC,(RR*RPC^(RR-1))/HC,-TR3*HT*RR*RPB^(-RR)*RPC^(-RR-
    1),(RR*RPC^(RR-1))/HC
28    0,-1,-HT*(1-RPT^(-RR)),-HT*(1-RPT^(-RR))
29    (RPC^RR-1)/HC^2,(1-RPC^RR)/HC^2,0,(1-RPC^RR)/HC^2
30    0,0,-TR3*HT*RR*RPC^(-RR)*RPB^(-RR-1),0
31    0,0,-TR3*(1-RPT^(-RR)),-TR3*(1-RPT^(-RR))];
```

Γραμμές 1-6

Δηλώνεται η συνάρτηση των περιορισμών. Η μόνη διαφορά σε σχέση με προηγούμενα είναι ότι έχουμε **4 ορίσματα εξόδου** αντί για 2. Αυτή η σύνταξη γίνεται όταν θέλουμε εκτός από τους περιορισμούς να παρέχουμε στον αλγόριθμό και τα **διανύσματα κλίσης** αυτών (δηλαδή τις μερικές τους παραγώγους). Αυτό γίνεται κυρίως, για μεγαλύτερη ακρίβεια στους υπολογισμούς. Επίσης είναι δυνατό να ξεπεραστούν κάποια μικρά προβλήματα ασυνέχειας. (προφανώς αν ήταν δύσκολος ο υπολογισμός των παραγώγων δεν θα μπαίναμε σε αυτή τη διαδικασία).

Το όρισμα DC θα περιέχει τα διανύσματα κλίσης των ανισοτικών περιορισμών και το DCeq τα διανύσματα κλίσης των ισοτικών περιορισμών. Προφανώς στο πρόβλημα αυτό ceq=DCeq=[ ] αφού δεν υφίστανται ισοτικοί περιορισμοί.

Γραμμή 7

Με αυτή την εντολή εισάγουμε στο χώρο εργασίας του cgam_constraints.m τις σταθερές παραμέτρους του προβλήματος που έχουν αποθηκευτεί στο αρχείο parameters.mat, και θα χρησιμεύσουν για τον υπολογισμό των υπόλοιπων σχέσεων.

Γραμμές 8-19

Εδώ καταγράφονται οι ενδιάμεσες σχέσεις για τον υπολογισμό των τιμών των περιορισμών.

Γραμμές 20-31

Δηλώνεται ο πίνακας-στήλη c των ανισοτικών περιορισμών καθώς και οι ceq και DCeq. Ο πίνακας DC που περιέχει τα διανύσματα κλίσης των ανισοτικών περιορισμών είναι ουσιαστικά ο παρακάτω:

$$\begin{bmatrix} \frac{\partial c_1}{\partial RPC} & \frac{\partial c_2}{\partial RPC} & \frac{\partial c_3}{\partial RPC} & \frac{\partial c_4}{\partial RPC} \\ \frac{\partial c_1}{\partial TR3} & \frac{\partial c_2}{\partial TR3} & \frac{\partial c_3}{\partial TR3} & \frac{\partial c_4}{\partial TR3} \\ \frac{\partial c_1}{\partial HC} & \frac{\partial c_2}{\partial HC} & \frac{\partial c_3}{\partial HC} & \frac{\partial c_4}{\partial HC} \\ \frac{\partial c_1}{\partial RPB} & \frac{\partial c_2}{\partial RPB} & \frac{\partial c_3}{\partial RPB} & \frac{\partial c_4}{\partial RPB} \\ \frac{\partial c_1}{\partial HT} & \frac{\partial c_2}{\partial HT} & \frac{\partial c_3}{\partial HT} & \frac{\partial c_4}{\partial HT} \end{bmatrix} =$$

$$\begin{bmatrix} -\frac{RR}{HC}\cdot RPC^{RR-1} & \frac{RR}{HC}\cdot RPC^{RR-1} & -TR3\cdot HT\cdot RR\cdot RPB^{-RR}\cdot RPC^{-RR-1} & \frac{RR}{HC}\cdot RPC^{RR-1} \\ 0 & -1 & -HT\cdot(1-RPT^{-RR}) & -HT\cdot(1-RPT^{-RR}) \\ \frac{RPC^{RR}-1}{HC^2} & \frac{1-RPC^{RR}}{HC^2} & 0 & \frac{1-RPC^{RR}}{HC^2} \\ 0 & 0 & -TR3\cdot HT\cdot RR\cdot RPC^{-RR}\cdot RPB^{-RR-1} & 0 \\ 0 & 0 & -TR3\cdot(1-RPT^{-RR}) & -TR3\cdot(1-RPT^{-RR}) \end{bmatrix}$$

17.5. Το αρχείο του αλγορίθμου βελτιστοποίησης optim_alg.m

```
1  %ΑΡΧΙΚΟ ΣΗΜΕΙΟ
2  var0=[30,4.8,0.85,0.98,0.88];
3  %boundaries and options
4  lboundary=[2,3,0.6,0.8,0.6];
5  uboundary=[100,10,0.895,0.99,0.915];
6  options=optimset('LargeScale','off','Display','iter','Diagnostics','on',...
7     'GradConstr','on','TolCon',0.0001,'TolX',0.0001,'DiffMinChange',1e-6);

8  [var,fval]=fmincon(@openc,var0,[],[],[],[],lboundary,uboundary,@openc_constraints,options)

9  fprintf(1,'Final result from optimization\n');
10 fprintf(1,'RPC = %9.5f \n',var(1));
11 fprintf(1,'TR3 = %9.5f \n',var(2));
12 fprintf(1,'HC = %9.5f \n',var(3));
13 fprintf(1,'RPB = %9.5f \n',var(4));
14 fprintf(1,'HT = %9.5f \n',var(5));
15 fprintf(1,'F = %9.5f \n',fval);
16 c=openc_constraints(var);
17 fprintf(1,'Constraints values at optimum point\n');
18 fprintf(1,'TR2-1 = %9.5f \n',-c(1));
19 fprintf(1,'TR3-TR2 = %9.5f \n',-c(2));
20 fprintf(1,'TR3-TR4 = %9.5f \n',-c(3));
21 fprintf(1,'TR3-TR4-TR2+1 = %9.5f \n',-c(4));
```

Γραμμές 1-7

Δίνεται το αρχικό σημείο εκκίνησης του αλγορίθμου var0 καθώς και τα όρια στις ανεξάρτητες μεταβλητές ώστε:
2<RPC<100
3<TR3<10
0.6<HC<0.895
0.8<RPB<0.99
0.6<HT<0.915
Δίνονται επίσης οι επιλογές του αλγορίθμου. Υπενθυμίζουμε ότι ο αλγόριθμος πρέπει να ενημερωθεί για την ύπαρξη των διανυσμάτων κλίσης των περιορισμών, οπότε προστίθεται η επιλογή **“GradConstr”,”on”**.

Γραμμές 8-21

Συντάσσεται η εντολή fmincon με το γνωστό τρόπο και παρουσιάζονται τα αποτελέσματα του αλγορίθμου με χρήση της εντολής fprintf. (Περισσότερες πληροφορίες για την fprintf στο Παράρτημα Ε)

17.6. Τα αποτελέσματα του αλγορίθμου βελτιστοποίησης

Κάνοντας "copy" τα δεδομένα του optim_alg.m και "paste" στο «command prompt» παίρνουμε τα αποτελέσματα του αλγορίθμου:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
   Diagnostic Information

Number of variables: 5

Functions
 Objective:                    openc
 Gradient:                     finite-differencing
 Hessian:                      finite-differencing (or Quasi-Newton)
 Nonlinear constraints and gradient:   openc_constraints

Constraints
 Number of nonlinear inequality constraints: 4
 Number of nonlinear equality constraints:   0

 Number of linear inequality constraints:    0
 Number of linear equality constraints:      0
 Number of lower bound constraints:          5
 Number of upper bound constraints:          5

Algorithm selected
   medium-scale


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
 End diagnostic information



                                 max                    Directional     First-order
 Iter F-count        f(x)   constraint    Step-size   derivative     optimality Procedure
    1     17       3921.15    -0.009375       0.0625          353    7.31e+003
    2     25       3653.84    -0.004687         0.5           334     7.9e+003
    3     37       3651.89     -0.01048       0.0313          398    2.63e+003
    4     50       3647.85     -0.01328       0.0156          225    5.53e+003
    5     61       3617.81     -0.01245       0.0625        -49.8    6.91e+003
    6     73       3615.25     -0.01206       0.0313         6.83    1.64e+003
```

```
   7    84      3615.16    -0.01131      0.0625       24.9        486
   8    92      3609.06    -0.005655        0.5        5.89    1.54e+003
   9   104      3608.88    -0.006709      0.0313       9.12    1.61e+003
  10   117      3608.87    -0.007718      0.0156       49.4         827
  11   124      3608.55    -0.006995           1    -0.0452        58.9
  12   131      3608.55    -0.006813           1   0.000118        21.8
  13   138      3608.54    -0.006808           1   3.59e-005        1.55
  14   145      3608.54    -0.006811           1   3.03e-006       0.492
  15   152      3608.54     -0.00681           1   5.53e-007       0.843 Hessian modified
Optimization terminated successfully:
 Search direction less than 2*options.TolX and
  maximum constraint violation is less than options.TolCon
 No Active Constraints

var =

  16.4755    5.0184    0.8503    0.9832    0.8978


fval =

  3.6085e+003

Final result from optimization
RPC =  16.47548
TR3 =   5.01844
HC =   0.85030
RPB =   0.98319
HT =   0.89776
F = 3608.54490
Constraints values at optimum point
TR2-1 =   1.59241
TR3-TR2 =   2.20759
TR3-TR4 =   2.41887
TR3-TR4-TR2+1 =   0.82646
```

17.7. Σύγκριση αποτελεσμάτων

Πίνακας 17.7.1 Σύγκριση αποτελεσμάτων
μεταξύ του προγράμματος GTG92.for και του Matlab

	Program GTG92.for	**Matlab**
RPC	16.49335	16.47548
TR3	5.01839	5.01844
HC	0.85024	0.85030
RPB	0.98298	0.98319
HT	0.89775	0.89776
F	3.60855E+03	3.608545E+03

Πίνακας 17.7.2 Ποσοστιαία μεταβολή της βέλτιστης λύσης που έδωσε το Matlab
σε σύγκριση με το πρόγραμμα GTG92.for

	Δ %
Δ(RPC)%	-0,10835
Δ(TR3)%	+0,00100
Δ(HC)%	+0,00706
Δ(RPB)%	+0,02136
Δ(HT)%	+0,00111
Δ(F)%	$-1{,}386\ 10^{4}$

17.8. Παρατηρήσεις

1) Το πρόβλημα αρχικά λύθηκε παίρνοντας υπόψη όλους τους περιορισμούς. Το πρόβλημα λύθηκε ξανά, με χρήση μόνο του 4ου περιορισμού (Παράρτημα Β, ενότητα 2 σχέση 20) . Τα αποτελέσματα που προέκυψαν ήταν αυτούσια.

2) Το πρόβλημα λύθηκε και μέσω Matlab, δηλαδή όλες οι σχέσεις και η αντικειμενική συνάρτηση γράφτηκαν από την αρχή σε ένα m-file. Χρησιμοποιήθηκαν τα ίδια αρχικά σημεία, τα ίδια όρια στις μεταβλητές, οι ίδιες επιλογές. Τα αποτελέσματα ήταν αυτούσια με προηγούμενα.

3) Το πρόβλημα λύθηκε και μέσω Matlab με την χρησιμοποίηση της μεθόδου της κλιμάκωσης (scaling). Τα αποτελέσματα ήταν αυτούσια με προηγούμενα. Παρατηρήθηκε σύγκλιση σε αρκετά αρχικά σημεία. Για uboundary=[100,10,0.9,0.99,0.92] δεν παρουσιάστηκε σύγκλιση στο πρόβλημα χωρίς κλιμάκωση. Αντίθετα για uboundary=[100,100,90,99,92] είχαμε σύγκλιση στο πρόβλημα με κλιμάκωση.

4) Στο πρόγραμμα GTG92.for, όλες οι μεταβλητές ήταν απλής ακρίβειας (REAL) καθώς επίσης και οι συναρτήσεις του εκθετικού (EXP) και του φυσικού λογάριθμου (ALOG). Στο cost_realfunction.f έπρεπε να γίνουν κάποιες μετατροπές. Όλες οι μεταβλητές, έπρεπε να δηλωθούν σαν REAL*8 (διπλής ακριβείας) και οι συναρτήσεις του εκθετικού και του φυσικού λογάριθμου να γίνουν διπλής ακρίβειας, δηλαδή DEXP και DLOG αντίστοιχα.

Για του λόγου το αληθές, στο αρχείο cost_realfunction.f δηλώθηκαν όλες οι μεταβλητές REAL*8 σαν REAL*4 και αντίστοιχα τα σύμβολα DLOG και DEXP έγιναν ALOG και EXP. Το πρόγραμμα μεταφράστηκε κανονικά και έγινε η προσπάθεια να υπολογιστεί η F για κάποιες τιμές των ανεξαρτήτων μεταβλητών:

>> F=cost_realfunction(30, 4.8, 0.85, 0.98, 0.88)

Το πρόγραμμα επέστρεψε την παρακάτω εξωφρενική τιμή για την F:

F =

2.1199e-314

Αυτή η συμπεριφορά του προγράμματος οφείλεται στο ότι το matlab αναπαριστά τους αριθμούς με διπλή ακρίβεια. Άρα η έξοδος του προγράμματος πρέπει να είναι διπλής ακρίβειας για να μπορεί να την χειριστεί το matlab. Γι' αυτό το λόγο, στη γραμμή 89 του cost_realfunction.f η έξοδος κατασκευάστηκε με την εντολή mxCreateDoubleMatrix (δημιουργία mxArray διπλής ακριβείας).

5) Το πρόγραμμα FUNCTION θα μπορούσε να γίνει και SUBROUTINE, δηλαδή αντί να γράψουμε:

real*8 function F(RPC,TR3,HC,RPB,HT)

μπορούμε να γράψουμε:

subroutine opencycle(F,RPC,TR3,HC,RPB,HT)

και για την πραγματοποίηση των υπολογισμών, αντί της κλήσης της συνάρτησης:

FD=F(RPC,TR3,HC,RPB,HT)

μπορούμε να καλέσουμε την υπορουτίνα:

call opencycle (F,RPC,TR3,HC,RPB,HT)

Ο λόγος που παρατίθενται τα προηγούμενα σχόλια είναι για να αναδείξουμε μία δυνατότητα που έχουν οι υπορουτίνες: την **δομή %val**.

Η δομή %val υποστηρίζεται από τους περισσότερους compilers, όπως και από τον Compaq visual fortran 6.6.

Σε κάποιον compiler που δεν την υποστηρίζει, πρέπει οι τιμές του mxArray να αντιγραφούν σε προσωρινούς πίνακες της fortran, με ειδικές ρουτίνες όπως την mxCopyPtrToReal8. Επίσης, όταν κληθεί η υπορουτίνα και τελειώσει τους υπολογισμούς, τα αποτελέσματα πρέπει πάλι να αντιγραφούν από τον πίνακα της fortran σε ένα mxArray που θα είναι η έξοδος για το matlab, με την ρουτίνα mxCopyReal8ToPtr.

Όμως, χρησιμοποιώντας την δομή %val, όλα αυτά μπορούν να αποφευχθούν. Στους παρακάτω πίνακες δίνουμε δύο παραδείγματα για το πρόβλημα του απλού αεριοστροβίλου που μελετάται, το ένα χωρίς χρήση και το άλλο με χρήση της δομής.

Πίνακας 17.8.1 Υπορουτίνα για το πρόβλημα του απλού αεριοστροβίλου χωρίς τη χρήση της δομής %val

ΧΩΡΙΣ ΧΡΗΣΗ ΤΗΣ ΔΟΜΗΣ %VAL

```
.
.
.
C    ----------------------------
C    CREATE OUTPUT
C    ----------------------------
     plhs(1)=mxCreateDoubleMatrix(m,n,0)
     F_PR=mxGetPr(plhs(1))
     call mxCopyPtrToReal8(RPC_PR, RPC, size)
     call mxCopyPtrToReal8(TR3_PR, TR3, size)
     call mxCopyPtrToReal8(HC_PR, HC, size)
     call mxCopyPtrToReal8(RPB_PR, RPB, size)
     call mxCopyPtrToReal8(HT_PR, HT, size)
C    ------------------------------------------------
C    CALL THE COMPUTATIONAL SUBROUTINE
C    ------------------------------------------------
     call opencycle(F,RPC,TR3,HC,RPB,HT)
C    ------------------------------------------------
C    LOAD THE DATA INTO F_PR WHICH IS THE OUTPUT TO MATLAB
C    ------------------------------------------------
     call mxCopyReal8ToPtr(F,F_PR,size)
     return
     end
```

Πίνακας 17.8.2 Υπορουτίνα για το πρόβλημα του απλού αεριοστροβίλου με χρήση της δομής %val

ΜΕ ΧΡΗΣΗ ΤΗΣ ΔΟΜΗΣ %VAL

```
.
.
.
C    ----------------------------
C    CREATE OUTPUT
C    ----------------------------
     plhs(1)=mxCreateDoubleMatrix(m,n,0)
     F_PR=mxGetPr(plhs(1))
C    ------------------------------------------------
C     CALL THE COMPUTATIONAL SUBROUTINE
C    ------------------------------------------------
      call opencycle(%val(F_PR),
    *     %val(RPC_PR),%val(TR3_PR),
    *      %val(HC_PR),%val(RPB_PR),%val(HT_PR))
     return
     end
```

17.9. Γενικά συμπεράσματα

- Παρέχονται μεγάλες δυνατότητες **επικοινωνίας του Matlab με εξωτερικά προγράμματα**:
    - Συναρτήσεις γλώσσας C και υπορουτίνες Fortran μπορούν να κληθούν από το matlab χρησιμοποιώντας τα mex-files.
    - Εντολές του matlab μπορούν να εκτελεστούν από προγράμματα σε C ή Fortran και τα αποτελέσματα των υπολογισμών να χρησιμοποιηθούν πάλι από τα προγράμματα αυτά, χρησιμοποιώντας το Matlab Engine.
    - Προγράμματα σε C ή Fortran έχουν τη δυνατότητα ανάγνωσης και γραφής δεδομένων από mat-files.
    - Δυνατότητας συνεργασίας με Java.
    - Δυνατότητας συνεργασίας με προγράμματα που βασίζονται στη γλώσσα Visual Basic, όπως το Word, το Excel και το Powerpoint.
- Προκύπτει μεγάλο όφελος σε περιπτώσεις που έχουμε **μεγάλους κώδικες** γραμμένους σε γλώσσα C ή Fortran. Δεν χρειάζεται να ξαναγραφτούν από την αρχή σαν m-files, αλλά μπορεί πολύ εύκολα (με κάποια σχετική εξοικείωση) να πραγματοποιηθεί η επικοινωνία του matlab με τη fortran ή τη C.
- Συνήθως οι **βρόχοι For** μπορεί να είναι πιο αποδοτικοί αν είναι γραμμένοι σε γλώσσα C ή Fortran. Έτσι, στις περιπτώσεις που θέλουμε **ταχύτητα και αποδοτικότητα** στους υπολογισμούς, η επικοινωνία με το matlab είναι μία λειτουργική λύση.

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ Α.
ΑΠΟΣΠΑΣΜΑΤΑ ΑΠΟ ΤΟ ΑΡΘΡΟ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ CGAM

Pergamon

Energy Vol. 19, No. 3, pp. 323–342, 1994

0360-5442/94 $6.00+0.00

APPLICATION OF THE THERMOECONOMIC FUNCTIONAL APPROACH TO THE CGAM PROBLEM

CHRISTOS A. FRANGOPOULOS

National Technical University of Athens
Department of Naval Architecture and Marine Engineering
P.O. Box 640 70, 157 10 Zografou, Greece

(Received 18 May 1993)

Abstract - A gas-turbine cogeneration system with a regenerative air preheater and a single-pressure exhaust gas boiler serves as an example for application of three different analysis and optimization procedures: (i) direct use of a nonlinear programming algorithm, (ii) thermoeconomic functional approach, and (iii) modular simulation and optimization of the system. The results obtained with the three methods are compared with each other. The sensitivity of the optimal solution to certain parameters and of the objective function to the independent variables is studied. Conclusions are drawn regarding the applicability of each procedure to more complicated optimization problems.

1. INTRODUCTION

In the introductory paper of this volume, "CGAM Problem : Definition and Conventional Solution," a gas-turbine cogeneration system is described (Fig. 1), the optimization objective is stated and the thermodynamic and economic equations pertinent to the system are given, which are valid under the stated assumptions.

In the following, several methods of analysis and optimization will be applied to the CGAM problem. Part of the material presented in the introductory paper will be repeated here in a form, which is appropriate for the complete presentation of each method. The emphasis is placed on the optimization of the system, while the analysis is performed at a level, which serves each particular optimization procedure.

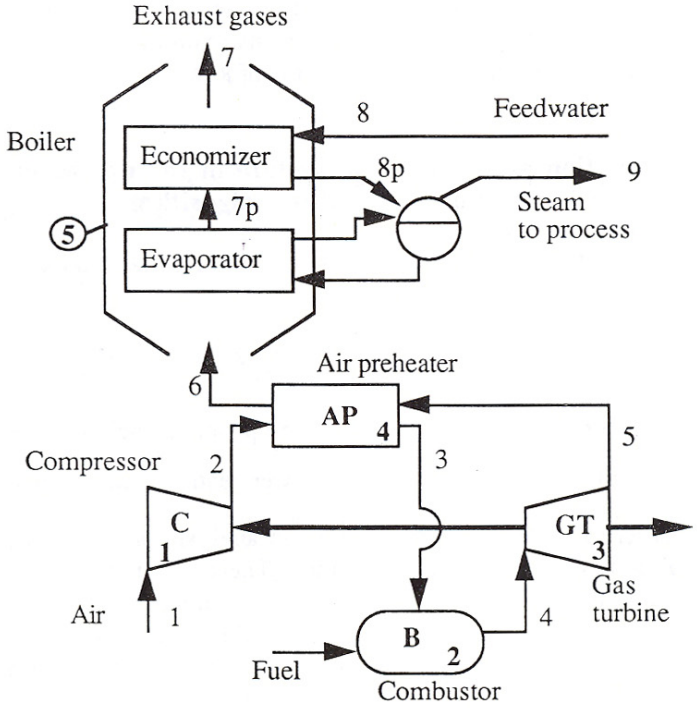


Fig. 1. Flow diagram of the gas-turbine cogeneration system.

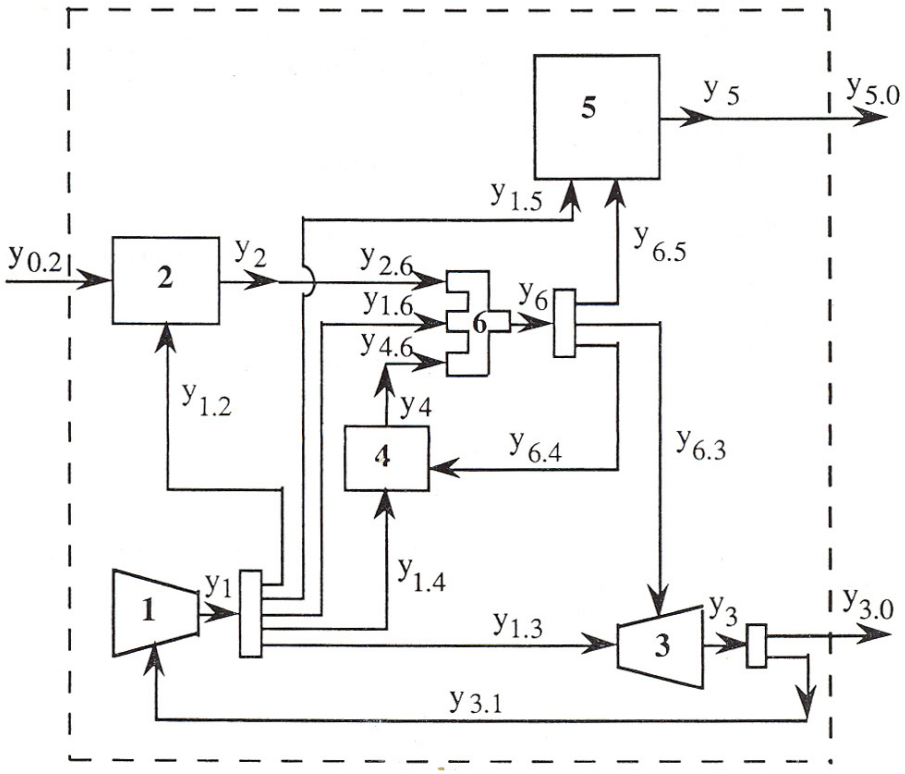


Fig. 2. Functional diagram of the system.

Table 1. Optimization results for the nominal values of parameters.

Variable	Method		
	Direct use of GRG2	TFA	Modular
r_C	8.59730	8.59770	8.59050
η_C	0.84641	0.84650	0.84653
η_T	0.87886	0.87871	0.87878
T_3 (K)	912.77	913.14	912.93
T_4 (K)	1491.40	1491.97	1491.50
F ($/year)	$1.0426 \cdot 10^7$	$1.0426 \cdot 10^7$	$1.0426 \cdot 10^7$

Table 2. TFA values of functions at the optimum point (in kW).

$y_1 = 27\,476$	$y_{1.2} = 437$	$y_{3.1} = 29\,846$
$y_2 = 56\,292$	$y_{1.3} = 16\,731$	$y_{4.6} = 18\,894$
$y_3 = 59\,846$	$y_{1.4} = 695$	$y_{6.3} = 45\,237$
$y_4 = 18\,894$	$y_{1.5} = 437$	$y_{6.4} = 20\,407$
$y_5 = 12\,745$	$y_{1.6} = 9\,176$	$y_{6.5} = 17\,028$
$y_6 = 82\,672$	$y_{2.6} = 56\,292$	

Table 3. TFA values of Lagrange multipliers and unit product costs at the optimum point (in $/$10^6$ kJ).

$\lambda_1 = 8.7621$	$c_1 = 8.8211$
$\lambda_2 = 5.8668$	$c_2 = 5.8672$
$\lambda_3 = 7.7614$	$c_3 = 7.8158$
$\lambda_4 = 7.7861$	$c_4 = 7.9552$
$\lambda_5 = 3.7305$	$c_5 = 10.0070$
$\lambda_6 = 6.7467$	$c_6 = 6.7922$

Table 4. Sensitivity of the optimal solution to the fuel price and capital cost.

Variation of variable	Fuel price + 100 %	Capital cost + 100 %
$\Delta r^*_C/r^*_C$ (%)	+ 13.76	- 13.75
$\Delta \eta^*_C/\eta^*_C$ (%)	+ 1.03	- 0.88
$\Delta \eta^*_T/\eta^*_T$ (%)	+ 0.80	- 0.84
$\Delta T^*_3/T^*_3$ (%)	- 2.39	+ 2.53
$\Delta T^*_4/T^*_4$ (%)	+ 0.66	- 0.60
$\Delta F^*/F^*$ (%)	+ 89.00	+ 9.21

Table 5. Sensitivity of the objective function to the independent variables.

$(x_i-x_i^*)/x_i^*$,(%) :	- 10	- 5	+ 5
x_i	(F-F*)/F* , (%)		
r_C	0.834	0.269	**
η_C	9.448	3.520	**
η_T	19.854	7.711	**
T_3	8.508	3.885	**
T_4	**	**	14.05
** Infeasible points			

APPENDIX A

Formulation of the Problem for Solution by Direct Application of the GRG2 Algorithm

Inequality constraints:

$$T_3 - T_2 \geq 0, \qquad T_5 - T_3 \geq 0, \qquad T_6 - T_2 \geq 0, \qquad T_7 - T_{7min} \geq 0,$$

$$T_4 - T_3 \geq 0, \qquad T_6 - T_9 \geq 0, \qquad T_{7p} - T_9 \geq \Delta T_{pmin}.$$

Equality constraints and objective function:

$$T_2 = T_1\left(1 + \frac{r_C^{k_a} - 1}{\eta_C}\right), \quad (A.1) \qquad P_2 = r_C P_1, \quad (A.2)$$

$$P_3 = r_{Aa} P_2, \quad (A.3) \qquad P_4 = r_B P_3, \quad (A.4)$$

$$P_6 = P_7/r_R, \quad (A.5) \qquad P_5 = P_6/r_{Ag}, \quad (A.6)$$

$$r_T = P_4/P_5, \quad (A.7) \qquad T_5 = T_4\left[1 - \eta_T(1 - r_T^{-k_g})\right], \quad (A.8)$$

$$f = \frac{c_{pg}(T_4 - T_o) - c_{pa}(T_3 - T_o)}{H_u\eta_B - c_{pg}(T_4 - T_o)}, \quad (A.9) \qquad T_6 = T_5 - \frac{c_{pa}(T_3 - T_2)}{(1+f)c_{pg}}, \quad (A.10)$$

$$\dot{m}_a = \frac{\dot{W}}{(1+f)c_{pg}(T_4 - T_5) - c_{pa}(T_2 - T_1)}, \quad (A.11) \qquad \dot{m}_f = f\,\dot{m}_a, \quad (A.12)$$

$$\dot{m}_g = \dot{m}_a + \dot{m}_f, \quad (A.13) \qquad T_7 = T_6 - \frac{\dot{Q}_R}{\dot{m}_g c_{pg}}, \quad (A.14)$$

$$T_{7p} = T_6 - \frac{\dot{Q}_{ev}}{\dot{m}_g c_{pg}}, \quad (A.15) \qquad \dot{W}_C = \dot{m}_a c_{pa}(T_2 - T_1), \quad (A.16)$$

$$\dot{W}_T = \dot{m}_g c_{pg}(T_4 - T_5), \quad (A.17) \qquad \Delta T_A = \frac{(T_6 - T_2) - (T_5 - T_3)}{\ln\frac{T_6 - T_2}{T_5 - T_3}}, \quad (A.18)$$

$$A_A = \frac{\dot{m}_g c_{pg}(T_5 - T_6)}{U\,\Delta T_A}, \quad (A.19) \qquad \Delta T_{ec} = \frac{(T_{7p} - T_{8p}) - (T_7 - T_8)}{\ln\frac{T_{7p} - T_{8p}}{T_7 - T_8}}, \quad (A.20)$$

$$\Delta T_{ev} = \frac{(T_6 - T_9) - (T_{7p} - T_9)}{\ln\frac{T_6 - T_9}{T_{7p} - T_9}}, \quad (A.21) \qquad C_1 = \frac{c_{11}\dot{m}_a}{c_{12} - \eta_C} r_C \ln r_C, \quad (A.22)$$

$$C_2 = \frac{c_{21}\dot{m}_a}{c_{22} - r_B}\left[1 + \exp(c_{23}T_4 - c_{24})\right], \quad (A.23)$$

$$C_3 = \frac{c_{31}\dot{m}_g}{c_{32} - \eta_T} \ln r_C \left[1 + \exp\left(c_{33} T_4 - c_{34}\right)\right], \tag{A.24}$$

$$C_4 = c_{41} A_A^{0.6}, \tag{A.25}$$

$$C_5 = c_{51}\left[\left(\frac{\dot{Q}_{ec}}{\Delta T_{ec}}\right)^{0.8} + \left(\frac{\dot{Q}_{ev}}{\Delta T_{ev}}\right)^{0.8}\right] + c_{52}\dot{m}_s + c_{53}\dot{m}_g^{1.2}, \tag{A.26}$$

$$F = FCR\ \phi \sum_{r=1}^{5} C_r + c_f \dot{m}_f H_u t. \tag{A.27}$$

Table A1. Nominal set of parameter values.

Thermodynamic parameters		
$\dot{W}$ = 30 000 kW $\dot{m}_s$ = 14 kg/s H_u = 50 000 kJ/kg c_{pa} = 1.004 kJ/kg K c_{pg} = 1.170 kJ/kg K ΔT_{8p} = 15 K ΔT_{pmin} = 0 K	P_1 = 1.013 bar P_7 = 1.013 bar P_8 = 20.0 bar P_9 = 20.0 bar γ_a = 1.40 γ_g = 1.33 η_B = 0.98 U = 0.018 kW/m^2. K	$T_0 = T_1$ = 298.15 K T_{7min} = 373.15 K T_8 = 298.15 K T_9 = 485.52 K R_a = 0.287 kJ/kg K R_g = 0.290 kJ/kg K r_{Aa} = 0.95 r_{Ag} = 0.97 $r_B = r_R$ = 0.95
Economic parameters		
t = 8000 h/year c_f = 4·10^{-6} \$/kJ FCR = 0.182 (year)$^{-1}$ ϕ = 1.06 c_{11} = 39.5 \$/(kg/s) c_{12} = 0.9	c_{21} = 25.6 \$/(kg/s) c_{22} = 0.995 c_{23} = 0.018 K^{-1} c_{24} = 26.4 c_{31} = 266.3 \$/(kg/s) c_{32} = 0.92	c_{33} = 0.036 K^{-1} c_{34} = 54.4 c_{41} = 2290 \$/m$^{1.2}$ c_{51} = 3650 \$/(kW/K)$^{0.8}$ c_{52} = 11820 \$/(kg/s) c_{53} = 658 \$/(kg/s)$^{1.2}$
Cost functions and parameters are adapted from Refs. 10-13.		

NOMENCLATURE

A_A heat-transfer area of the air preheater
C_r capital cost of component r (installed)
c_f price of fuel
c_p specific heat capacity at constant pressure
c_r unit cost of the function of component r
c_{ri} capital cost coefficients
$\Delta T_{pmin} = (T_{7p}-T_9)_{min}$ minimum pinch point temperature difference
$\dot{E}_s^Q$ increase of exergy flow rate from water to steam
F objective function
f fuel to air ratio,
f_ν objective function of subsystem ν (in decomposition)
FCR fixed charge rate
g set of inequality constraints
H_u lower heating value of fuel
h specific enthalpy
h set of equality constraints
k specific heat capacity ratio
L Lagrangian
$\dot{m}$ mass flow rate
P pressure
p set of parameters
$\dot{Q}$ heat rate
R gas constant,
number of units and junctions in a system
T temperature
t operation time per year
U overall heat transfer coefficient of air preheater
$\dot{W}$ power,
net electric power output of the system
$\mathbf{w}_r$ set of dependent variables appearing in the simulation model of module r only.
x set of independent decision variables for optimization
y set of dependent variables
$\hat{\mathbf{y}}_{r,0}$ fixed product of the system
$\mathbf{y}_r$ the function of unit r
$\mathbf{y}_{r,r'}$ function going from unit r to unit r'
$\mathbf{y}_{ri}$ set of input dependent variables to module r
$\mathbf{y}_r$ set of output dependent variables from module r
$\dot{Z}$ capital cost rate including operation and maintenance (except of fuel)

Greek letters

$\Gamma_{0.2}$ fuel cost rate, Eq. (3.22)
Γ_r as defined by Eq. (3.28)
ε specific flow exergy
η efficiency
λ Lagrange multiplier
ϕ maintenance factor

Subscripts

a air
A air preheater
Aa air-side of the air preheater
Ag gas-side of the air preheater
B combustor
C compressor
ec economizer
ev evaporator
f fuel
g exhaust gas
o reference conditions
R recuperator-boiler
r the *r*th unit or module of the system (r=0: environment)
s steam
T turbine

Superscripts

T thermal exergy
° initial value
* optimum value

Overmark

· per unit time

ΠΑΡΑΡΤΗΜΑ Β.
ΔΙΑΤΥΠΩΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΑΠΛΟΥ ΑΕΡΙΟΣΤΡΟΒΙΛΟΥ

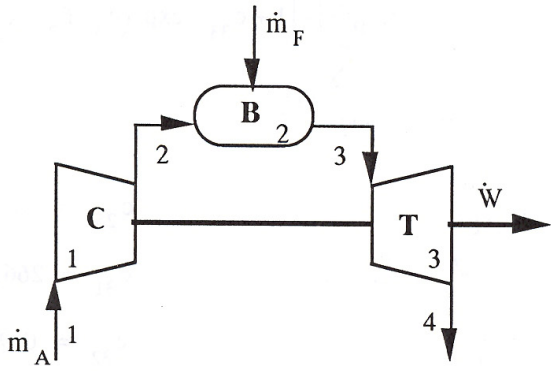
Β1. Απόσπασμα από το πρόβλημα του απλού αεριοστροβίλου

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΝΑΥΠΗΓΩΝ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Μάθημα : **Ανάλυση και Βελτιστοποίηση Ενεργειακών Συστημάτων**

2ο Θέμα

Η εγκατάσταση αεριοστροβίλου του σχήματος μπορεί να θεωρηθεί ότι λειτουργεί με εργαζόμενο μέσο αέρα σταθερής ποσότητας και σύνθεσης.



Ζητείται να λυθεί το πρόβλημα βελτιστοποίησης του συστήματος με αντικειμενική συνάρτηση

α. τον βαθμό απόδοσης,

β. τη συγκέντρωση ισχύος,

γ. το ετήσιο κόστος κτήσεως και λειτουργίας,

περιορισμούς εκείνους που προκύπτουν από τη θερμοδυναμική ανάλυση και ανεξάρτητες μεταβλητές

$$\mathbf{x} = (r_{pC}, r_{pB}, \tau_3, \eta_C, \eta_T)$$

όπου $r_{pC} = P_2/P_1$, $r_{pB} = P_3/P_2$, $\tau_3 = T_3/T_1$.

Επίσης, να μελετηθεί γραφικά η ευαισθησία της λύσης ως προς δύο παραμέτρους, οι οποίες θα ορισθούν για κάθε σπουδαστή σε συνεννόηση με τον διδάσκοντα, και να σχολιασθούν τα αποτελέσματα.

Δεδομένα:

Καθαρή ισχύς	$\dot{W} = 20.000$ kW
Καύσιμο : LNG με θερμογόνο ικανότητα	$H_u = 50.000$ kJ/kg
και βασική τιμή	$c_f = 0{,}20$ \$/kg
Κατάσταση αέρα στην είσοδο του συμπιεστή	$P_1 = 1$ bar
	$T_1 = 20$ °C
Διάρκεια λειτουργίας	$t = 7500$ h/έτος

Κόστος κτήσεως των στοιχείων του συστήματος :

$$C_1 = \frac{c_{11}\dot{m}_A}{c_{12} - \eta_c} \cdot \frac{P_2}{P_1} \cdot \ln \frac{P_2}{P_1}$$

$$C_2 = \frac{c_{21}\dot{m}_A}{c_{22} - r_B} \left[1 + c_{23} \cdot \exp\left(c_{24}\, T_3 - c_{25}\right)\right]$$

$$C_3 = \frac{c_{31}\dot{m}_A}{c_{32} - \eta_T} \cdot \left(\ln \frac{p_3}{P_4}\right) \cdot \left[1 + c_{33} \cdot \exp\left(c_{34} T_3 - c_{35}\right)\right]$$

Βασικές τιμές παραμέτρων κόστους :

$c_{11} = 39{,}5 \ \frac{\$}{kg/s}$ $\qquad$ $c_{25} = 28$

$c_{12} = 0{,}90$ $\qquad$ $c_{31} = 266{,}3 \ \frac{\$}{kg/s}$

$c_{21} = 25{,}6 \ \frac{\$}{kg/s}$ $\qquad$ $c_{32} = 0{,}92$

$c_{22} = 0{,}995$ $\qquad$ $c_{33} = 5$

$c_{23} = 5$ $\qquad$ $c_{34} = 0{,}036 \ K^{-1}$

$c_{24} = 0{,}018 \ K^{-1}$ $\qquad$ $c_{35} = 56$

Βασικές τιμές παραμέτρων οικονομικής ανάλυσης :

Διάρκεια ζωής του συστήματος $N = 20$ έτη

Επιτόκιο αγοράς $i = 10\ \%$

Το ετήσιο κόστος θεωρείται ότι αποτελείται από την απόσβεση κεφαλαίου, δαπάνες συντήρησης ίσες με το 1,5% του κόστους κτήσεως, άλλα πάγια έξοδα ίσα με το 2,5% του κόστους κτήσεως και δαπάνες καυσίμου.

Παρατηρήσεις

Η αριθμητική επίλυση και η παραμετρική μελέτη θα γίνουν σε προσωπικό υπολογιστή (PC). Υπάρχει σχετικό πρόγραμμα σε γλώσσα FORTRAN 77, που δίνεται στους σπουδαστές.

Διανομή : 17.12.2002
Οδηγίες για την εκπόνηση : 19.12.2002
Παράδοση : 27.1.2003

Πίνακας Β.1 Δεδομένα του προβλήματος και αποτελέσματα της βελτιστοποίησης με χρήση του προγράμματος GTG92.

```
PROGRAM GTG92
-------------

DATA :

P1 =    1.000          W  =  10000.0
CF =    0.200          HU =  42500.0
CP =    1.000          T1 =    293.0
RG =    0.287          HR =   7500.0
RPM=  100.000          TRM=       5.2
RK =    1.400

C11 =  39.500          C31 = 266.300
C12 =   0.900          C32 =   0.920
C21 =  25.600          C33 =   5.000
C22 =   0.995          C34 =   0.036
C23 =   5.000          C35 =  56.000
C24 =   0.018
C25 =  28.000

N   =  20              FM  =   0.015
MIR =   0.100          FE  =   0.025

RPCI =   20.000         HCI =  0.850
TR3I =    4.800         HTI =  0.880          RPBI =  0.980
KPR  =   0


NUMBER OF ITERATIONS :     6
FUNCTION EVALUATIONS :   696
TOLP =  0.0100        TOLT =  0.0010        TOLH =   0.0010


FINAL RESULTS :

P2 =     16.49335          P3 =     16.21267
T3 =   1470.38800          TR3=      5.01839
HC =      0.85024          T2 =    715.98110
RPB=      0.98298          RB =      0.99511
HT =      0.89775          RC =      2.22742
WC =  14028.39000          MA =     33.16552
Z1 =  1.91639E+02          MF =      0.58871
Z2 =  2.31323E+01          ZF =  3.17905E+03
Z3 =  2.14734E+02          F  =  3.60855E+03
```

Β.2. Μαθηματική διατύπωση του προβλήματος

$$(1)\quad CRF = \frac{MIR \cdot (1+MIR)^N}{(1+MIR)^N - 1}$$

$$(2)\quad FCR = CRF + FM + FE$$

$$(3)\quad C11 = 0.001 \cdot FCR \cdot C11$$

$$(4)\quad C21 = 0.001 \cdot FCR \cdot C21$$

$$(5)\quad C31 = 0.001 \cdot FCR \cdot C31$$

$$(6)\quad RR = 1 - \frac{1}{RK}$$

$$(7)\quad TR2 = 1 + \frac{RPC^{RR} - 1}{HC}$$

$$(8)\quad RPT = RPB \cdot RPC$$

$$(9)\quad TR4 = TR3 \cdot (1 - HT \cdot (1 - RPT^{-RR}))$$

$$(10)\quad MA = \frac{W}{CP \cdot T1 \cdot (TR3 - TR4 - TR2 + 1)}$$

$$(11)\quad MF = \frac{MA \cdot CP \cdot T1 \cdot (TR3 - TR2)}{HU}$$

$$(12)\quad Z1 = C11 \cdot MA \cdot RPC \cdot \frac{\ln(RPC)}{C12 - HC}$$

$$(13)\quad Z2 = \frac{C21 \cdot MA}{C22 - RPB} \cdot (1 + C23 \cdot \exp(C24 \cdot TR3 \cdot T1 - C25))$$

$$(14)\quad Z3 = \frac{C31 \cdot MA \cdot \ln(RPT)}{C32 - HT} \cdot (1 + C33 \cdot \exp(C34 \cdot TR3 \cdot T1 - C35))$$

$$(15)\quad ZF = 3.6 \cdot CF \cdot MF \cdot HR$$

$$(16)\quad F = Z1 + Z2 + Z3 + ZF$$

Με τους παρακάτω περιορισμούς:

$$(17)\quad TR2 \geq 1$$

$$(18)\quad TR3 \geq TR2$$

$$(19)\quad TR3 \geq TR4$$

$$(20)\quad TR3 - TR4 - TR2 + 1 \geq 0$$

ΠΑΡΑΡΤΗΜΑ Γ.

ΜΕΘΟΔΟΙ ΕΠΙΛΥΣΗΣ ΜΗ ΓΡΑΜΜΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΤΟ MATLAB

Υπάρχουν δύο τρόποι, που μπορεί ο χρήστης να χρησιμοποιήσει για την επίλυση ενός μη γραμμικού συστήματος εξισώσεων.

Ο πρώτος τρόπος είναι να προγραμματιστεί το matlab, με αναλυτικές εντολές ώστε να μπορούν να εφαρμοστούν μέθοδοι γνωστές από την αριθμητική ανάλυση για μη γραμμικά συστήματα.

Ο δεύτερος τρόπος είναι να χρησιμοποιήσουμε τις ενσωματωμένες (built-in) μεθόδους που χρησιμοποιούνται στην επίλυση μη γραμμικών συστημάτων και παρέχονται από το «optimization toolbox». Εδώ θα χρησιμοποιηθεί η εντολή **fsolve** για την επίλυση μη γραμμικών συστημάτων. Το «optimization toolbox» παρέχει τρεις μεθόδους που μπορούν να χρησιμοποιηθούν. Η προεπιλεγμένη (default μέθοδος) είναι η trust-region dogleg που βασίζεται στην μέθοδο dogleg του Powell. Εναλλακτικά ο χρήστης μπορεί να επιλέξει τη μέθοδο Gauss-Newton της κατηγορίας μεθόδων line-search ή την Levenberg - Marquardt της ίδιας κατηγορίας.

Γ.1 Επίλυση του συστήματος αναλυτικά

- Μπορούμε να χρησιμοποιήσουμε τη μέθοδο Newton για την επίλυση μη γραμμικών συστημάτων. Με βάση αυτή τη μέθοδο το μη γραμμικό σύστημα:

$$
\begin{aligned}
&f_1(x_1, x_2, \ldots, x_n) = 0\\
&f_2(x_1, x_2, \ldots, x_n) = 0\\
&f_3(x_1, x_2, \ldots, x_n) = 0\\
&\cdot\\
&\cdot\\
&f_n(x_1, x_2, \ldots, x_n) = 0
\end{aligned}
$$

Μπορεί να γραφτεί σε διανυσματική μορφή σαν $F(x) = 0$ όπου

$$x = [x_1, x_2, \ldots, x_n]$$

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ . \\ . \end{bmatrix}$$

Η μέθοδος Newton χρησιμοποιεί τον Ιακωβιανό πίνακα, δηλαδή τον πίνακα που περιέχει τις μερικές παραγώγους του συστήματος δηλαδή τον πίνακα:

$$J(x_1, x_2..., x_n) = \begin{bmatrix} \partial f_1 / \partial x_1 & . & . & \partial f_1 / \partial x_n \\ . & & & . \\ . & & & . \\ \partial f_n / \partial x_1 & . & . & \partial f_n / \partial x_n \end{bmatrix}$$

Σε κάθε στάδιο της επαναληπτικής διαδικασίας ένα νέο διάνυσμα x_new από την προσωρινή λύση υπολογίζεται σύμφωνα με την εξίσωση:

$$x\_new = x\_old - J^{-1}(x\_old) \cdot F(x\_old)$$

Στην περίπτωση εξίσωσης μίας μεταβλητής, αυτή η μέθοδος, θυμίζει την γνωστή μέθοδο Newton για εξισώσεις μιας μεταβλητής η οποία δίνεται από τον αναδρομικό τύπο:

$$x\_new = x\_old - \frac{f(x\_old)}{f'(x\_old)}$$

Επειδή στην πράξη είναι δύσκολος ο υπολογισμός του αντιστρόφου του Ιακωβιανού πίνακα, γι' αυτό, πρακτικά, λύνεται το ισοδύναμο γραμμικό σύστημα:

$$J(x\_old)y = -F(x\_old) \text{ όπου } y = x\_new - x\_old$$

Και λύνοντας ως προς το x_new έχουμε:

$$x\_new = x\_old + y$$

Έτσι, για αυτή την απλή επαναληπτική διαδικασία εφαρμόζουμε τα εξής βήματα:

ΒΗΜΑ 1

- Γράφουμε ένα function m-file, το sys.m που περιέχει το σύστημα που θέλουμε να λύσουμε σε πινακοποιημένη μορφή (διάνυσμα-στήλη). Έστω ότι θέλουμε να επιλύσουμε το παρακάτω σύστημα:

$$f_1(x_1, x_2..., x_n) = x_1^2 + x_2^2 + x_3^2 - 1 = 0$$
$$f_2(x_1, x_2..., x_n) = x_1^2 + x_3^2 - 1/4 = 0$$
$$f_3(x_1, x_2..., x_n) = x_1^2 + x_2^2 - 4x_3 = 0$$

Το αρχείο sys.m είναι το παρακάτω:

```
function f=sys(x)
f=[x(1)^2+x(2)^2+x(3)^2-1
   x(1)^2+x(3)^2-0.25
   x(1)^2+x(2)^2-4*x(3)]
```

ΒΗΜΑ 2

- Έπειτα γράφουμε το function m-file Jac_matrix.m που περιέχει τον Ιακωβιανό πίνακα. Είναι αυτό που φαίνεται παρακάτω:

```
function g=Jac_matrix(x)
g=[2*x(1) , 2*x(2) , 2*x(3) ; 2*x(1) , 0 , 2*x(3);2*x(1) , 2*x(2) , -4]
```

ΒΗΜΑ 3

- Στη συνέχεια γράφουμε το function m-file Newton_sys το οποίο περιέχει τον αλγόριθμο. Σαν δεδομένα εισόδου έχει:

α) το αρχείο που περιέχει το σύστημα, **β)** το αρχείο που περιέχει τον Ιακωβιανό πίνακα, **γ)** το αρχικό σημείο εκκίνησης σαν πίνακα-γραμμή, **δ)** την ανοχή για τον τερματισμό του αλγορίθμου, δηλαδή τη νόρμα $\|x\_new - x\_old\|$ και **ε)** τον μέγιστο αριθμό των επαναλήψεων.

```
function x=Newton_sys(sys,Jac_matrix,x0,tol,max_it)
%επίλυση συστήματος μη γραμμικών εξισώσεων
%με τη μέθοδο Newton
%sys, Jac_matrix m-files που περιέχουν το σύστημα και τον Ιακωβιανό σε
%πινακοποιημένη μορφή
%σταματά όταν ικανοποιηθεί το tol η επιτευχθούν οι μέγιστες επαναλήψεις
x_old=x0;
iter=1;
while iter<=max_it
   y=-feval(Jac_matrix,x_old)\feval(sys,x_old); %επίλυση με Gaussian elimination
   x_new=x_old+y';
   dif=norm(x_new-x_old);
    if dif<=tol
      x=x_new;
      disp('Η μέθοδος Newton συνέκλινε')
      return;
   else
      x_old=x_new;
   end
   iter=iter+1;
end
disp('Η μέθοδος Newton δεν συνέκλινε')
x=x_new
```

Έτσι, ξεκινώντας από το αρχικό σημείο [1 1 1] πληκτρολογώντας στο «command prompt» την εντολή:

```
>>Newton_sys(@sys,@Jac_matrix,[1 1 1],1e-05,20)
```

Έπειτα από 7 επαναλήψεις παίρνουμε το αποτέλεσμα:

```
Η μέθοδος Newton συνέκλινε

ans =

   0.4408    0.8660    0.2361
```

Γ.2 Επίλυση του συστήματος με την εντολή FSOLVE του "optimization toolbox"

Η εντολή fsolve συντάσσεται γενικά:

[x,fval,exitflag,output,jacobian]=fsolve(fun,x0,options,P1,P2,…)

Πίνακας Γ.2.1 Ορίσματα εισόδου-εξόδου της fsolve

Ορίσματα εισόδου	
fun	Είναι το function m-file που περιέχει τις εξισώσεις σε πινακοποιημένη μορφή.
x0	Το αρχικό σημείο (σημείο εκκίνησης του αλγορίθμου)
options	Κάποιες επιλογές που αφορούν τον αλγόριθμο.
P1,P2	Επιπλέον μεταβλητές που εισάγονται στο πρόβλημα.
Ορίσματα εξόδου	
x	Το σημείο επίλυσης
Fval	Οι τιμές της συνάρτησης fun στη λύση x.
Exitflag	Περιγράφει την κατάσταση εξόδου
>0	Η fun συνέκλινε στη λύση x
0	Ο μέγιστος αριθμός των υπολογισμών της fun ξεπεράστηκε
<0	Η fun δεν συνέκλινε σε κάποια λύση
Output	Πεδίο που περιέχει πληροφορίες για την κατάσταση εξόδου
iterations	Ο αριθμός των επαναλήψεων
funcCount	Ο αριθμός των υπολογισμών της αντικειμενικής συνάρτησης
algorithm	Ο αλγόριθμος που χρησιμοποιήθηκε
cgiterations	Ο αριθμός των PCG (Preconditioned conjugate gradient) επαναλήψεων (Μόνο για αλγορίθμους προβλημάτων μεγάλης κλίμακας)
stepsize	Το τελικό βήμα. (Μόνο για αλγορίθμους προβλημάτων μέσης κλίμακας)
firstorderopt	Μέτρο ικανοποίησης των συνθηκών βελτίστου πρώτης τάξης (first order optimality).
Jacobian	Δίνει την τιμή του Ιακωβιανού πίνακα της συνάρτησης fun στη λύση x.

Πίνακας Γ.2.2 Οι επιλογές της εντολής fsolve

Επιλογές που χρησιμοποιούνται σε αλγόριθμους μέσης και μεγάλης κλίμακας		
ΕΠΙΛΟΓΗ	**ΠΕΡΙΓΡΑΦΗ**	**ΠΑΡΑΔΕΙΓΜΑ**
Diagnostics	Εκτύπωση διαγνωστικών πληροφοριών για την αντικειμενική συνάρτηση	Options=optimset('Diagnostics','on')
Display	Όταν είναι στο 'off' δεν δείχνει καθόλου τα αποτελέσματα του αλγορίθμου. Στο 'iter' δείχνει το αποτέλεσμα σε κάθε επανάληψη. Στο 'final' δείχνει μόνο το τελικό αποτέλεσμα	Options=optimset('Display','iter')
Jacobian	Όταν είναι στο 'on' τότε η fsolve χρησιμοποιεί τον Ιακωβιανό πίνακα που έχει δηλώσει ο χρήστης στην συνάρτηση fun. Αν είναι στο 'off' (default) η fsolve υπολογίζει τον Ιακωβιανό πίνακα χρησιμοποιώντας τη μέθοδο των πεπερασμένων διαφορών	Options=optimset('Jacobian','on')
MaxFunEvals	Ο μέγιστος επιτρεπόμενος αριθμός των υπολογισμών της αντικειμενικής συνάρτησης	Options=optimset('MaxFunEvals',100)
MaxIter	Ο μέγιστος αριθμός των επιτρεπόμενων επαναλήψεων	Options=optimset('MaxIter',100)
TolFun	Ανοχή (tolerance) για τον τερματισμό του αλγορίθμου όσον αφορά την τιμή της fun	Options=optimset('TolFun',1e-2)
TolX	Ανοχή (tolerance) για τον τερματισμό του αλγορίθμου όσον αφορά στην τιμή του x.	Options=optimset('TolX',1e-5)
Επιλογές που χρησιμοποιούνται μόνο σε αλγόριθμους μέσης κλίμακας		
DerivativeCheck	Δίδεται στην fsolve η δυνατότητα να συγκριθούν τα αποτελέσματα του	Options=optimset('DerivativeCheck','on')

	Ιακωβιανού πίνακα που έδωσε ο χρήστης με αυτά που υπολόγισε το matlab με τη μέθοδο των πεπερασμένων διαφορών	
DiffMaxChange	Μέγιστη αλλαγή στις τιμές των μεταβλητών όταν χρησιμοποιείται η μέθοδος των πεπερασμένων διαφορών	Options=optimset('DiffMaxChange',1e-02)
DiffMinChange	Ελάχιστη αλλαγή στις τιμές των μεταβλητών όταν χρησιμοποιείται η μέθοδος των πεπερασμένων διαφορών	Options=optimset('DiffMinChange', 1e-02)
NonlEqnAlgorithm	Επιλογή του αλγορίθμου Levenberg-Mardquardt ή του Gauss-Newton και του trust-region dogleg	Options=optimset('NonlEqnAlgorithm','dogleg') Options=optimset('NonlEqnAlgorithm','gn') Options=optimset('NonlEqnAlgorithm','lm')
LineSearchType	Επιλογή του αλγορίθμου για το Line search	Options=optimset('LineSearchType','quadcubic') Αυτή η μέθοδος είναι ένας συνδιασμός μιας τετραγωνικής και μιας κυβικής παρεμβολής. Options=optimset('LineSearchType','qubicpoly') Η qubicpoly βασίζεται σε κυβικά πολυώνυμα. Γενικά χρειάζεται λιγότερους υπολογισμούς της συνάρτησης, αλλά περισσότερους υπολογισμούς για τα διανύσματα κλίσης. Έτσι αυτή η μέθοδος είναι προτιμότερη εφόσον ο χρήστης παρέχει στην fsolve τα διανύσματα κλίσης

Σημείωση: Υπάρχουν και επιλογές που χρησιμοποιούνται μόνο από αλγόριθμους μεγάλης κλίμακας, αλλά είναι αρκετά εξειδικευμένες και ξεφεύγουν από τα όρια της εργασίας αυτής. Ο αναγνώστης μπορεί να κοιτάξει στη βιβλιογραφία [8] σελ. 5-85 για περισσότερες πληροφορίες.

Γ.2.1 Λίγα λόγια για τις μεθόδους επίλυσης

- Στη μέθοδο Gauss-Newton, σε κάθε επανάληψη υπολογίζεται μία νέα κατεύθυνση εύρεσης d_k , που είναι η λύση του προβλήματος ελάχιστων τετραγώνων

$$\min \left\| J(x_k)d_k - F(x_k) \right\|_2^2$$

Η κατεύθυνση έρευνας d_k υπολογίζεται σαν ένα κομμάτι μιας στρατηγικής line search ώστε να είναι βέβαιο ότι σε κάθε επανάληψη η συνάρτηση f(x) μειώνεται.

- Η μέθοδος Levenberg-Marquardt χρησιμοποιεί μια κατεύθυνση έρευνας που προκύπτει ως η λύση του παρακάτω συστήματος γραμμικών εξισώσεων:

$$(J(x_k)^T J(x_k) + \lambda_k I) d_k = -J(x_k) F(x_k)$$

Όπου η σταθερά $\lambda_κ$ ελέγχει και το μέγεθος αλλά και την κατεύθυνση του d_k. Όταν το $\lambda_κ$ είναι μηδέν, η κατεύθυνση d_k συμπίπτει με αυτή της μεθόδου Gauss-Newton. Όσο το $\lambda_κ$ τείνει στο άπειρο η κατεύθυνση d_k τείνει σε ένα μηδενικό διάνυσμα και σε κατεύθυνση μέγιστης καθόδου (steepest descent). Αυτό σημαίνει ότι για αρκετά μεγάλο λ_k η ανισότητα $F(x_k + d_k) < F(x_k)$ είναι αληθής.

Αν και η Levenberg-Marquardt είναι σε ελάχιστες περιπτώσεις λιγότερο αξιόπιστη, θεωρείται ότι και οι δύο μέθοδοι είναι πολύ αποδοτικές.

Παρατήρηση: Η προκαθορισμένη μέθοδος dogleg μπορεί να χρησιμοποιηθεί μόνο σε συστήματα που ο αριθμός των εξισώσεων είναι ίσος με τον αριθμό των αγνώστων. Αντίθετα στις μεθόδους Gauss-Newton και Levenberg-Marquardt αυτό δεν είναι απαραίτητο.

Γ.2.2 Μέθοδος επίλυσης

Έστω το σύστημα μη-γραμμικών εξισώσεων:

$$f_1(x_1, x_2 ..., x_n) = x_1^{\,2} + x_2^{\,2} + x_3^{\,2} - 1 = 0$$
$$f_2(x_1, x_2 ..., x_n) = x_1^{\,2} + x_3^{\,2} - 1/4 = 0$$
$$f_3(x_1, x_2 ..., x_n) = x_1^{\,2} + x_2^{\,2} - 4x_3 = 0$$

ΒΗΜΑ 1

Κατασκευάζουμε το αρχείο fun_jac.m το οποίο περιέχει το σύστημα προς επίλυση και τον Ιακωβιανό πίνακα σε πινακοποιημένη μορφή:

```
function [F,J]=fun_jac(x)

F=[x(1)^2+x(2)^2+x(3)^2-1
   x(1)^2+x(3)^2-0.25
   x(1)^2+x(2)^2-4*x(3)];

J=[2*x(1) 2*x(2) 2*x(3)
   2*x(1) 0 2*x(3)
   2*x(1) 2*x(2) -4];
```

ΒΗΜΑ 2

Στο «command window», καθορίζουμε το αρχικό σημείο, της επιλογές της fsolve και τέλος δίδουμε την εντολή επίλυσης του συστήματος. Να σημειωθεί ότι στις επιλογές έχουμε επιλέξει σαν μέθοδο επίλυσης την Gauss-Newton.

```
>> x0=[1 1 1];
options=optimset('Jacobian','on','display','iter',...
   'DerivativeCheck','on','NonlEqnAlgorithm','gn');
[x,fval,exitflag,output,jacobian]=fsolve(@fun_jac,x0,options)
```

Και το matlab μετά από πέντε επαναλήψεις βγάζει τα εξής αποτελέσματα:

```
                                                 Directional
 Iteration  Func-count    Residual    Step-size     derivative
Maximum discrepancy between derivatives = 1.98787e-007
    1           3          11.0625          1          -22.1
    2           7          0.414148         1.1        -0.183
    3          11          0.00140473       1.27        -0.00578
    4          15          1.20553e-008     1          -7.83e-009
    5          19          3.27547e-017     1          -2.45e-017
Optimization terminated successfully:
 Search direction less than tolX

x =

   0.4408    0.8660    0.2361

fval =

  1.0e-012 *

   0.0215
   0.0164
  -0.4943

exitflag =

    1

output =

     iterations: 5
      funcCount: 20
```

```
      stepsize: 1.0001
   cgiterations: []
  firstorderopt: []
     algorithm: 'medium-scale: Gauss-Newton, line-search'

jacobian =

   0.8815    1.7321    0.4721
   0.8815         0    0.4721
   0.8815    1.7321   -4.0000
```

Αν τώρα χρησιμοποιήσουμε τη μέθοδο Levenberg-Marquardt δηλαδή στις επιλογές δηλώσουμε 'NonlEqnAlgorithm','lm', θα πάρουμε το παρακάτω αποτέλεσμα για το x:

```
                                                  Directional
 Iteration  Func-count    Residual      Step-size   derivative      Lambda
Maximum discrepancy between derivatives = 1.98787e-007
    1             3        11.0625            1          -22.1
    2             7       0.414148          1.1         -0.183    0.371279
    3            12     0.00212714         1.26        -0.00963    0.161382
    4            17    2.45684e-005         1.23        -2.66e-007    0.0723901
    5            21    3.62948e-008         1.04        -6.23e-011    0.0355461
    6            25    1.41959e-011         1.05        -1.14e-015    0.0173015
Optimization terminated successfully:
 Gradient in the search direction less than tolFun
 Gradient less than 10*(tolFun+tolX)

x =

   0.4408    0.8660    0.2361
```

Σημείωση: Το αποτέλεσμα είναι το ίδιο, αλλά η μέθοδος Levenberg-Marquardt έκανε 6 επαναλήψεις αντί για 5 της Gauss-Newton.

Παραλείποντας τελείως τον παράγοντα 'NonlEqnAlgorithm' στη συνάρτηση optimset, χρησιμοποιείται η μέθοδος dogleg, που είναι η προκαθορισμένη (default) μέθοδος. Παίρνουμε το εξής αποτέλεσμα:

Iteration	Func-count	Norm of f(x)	step	First-order optimality	Trust-region radius
Maximum discrepancy between derivatives = 0					
1	4	11.0625	15.5	1	
2	5	0.494963	0.709558	1.66	1
3	6	0.0182743	0.284572	0.265	1.77
4	7	0.000101911	0.076353	0.0202	1.77
5	8	5.4073e-009	0.00651576	0.000147	1.77
6	9	1.61328e-017	4.81557e-005	8.03e-009	1.77

Optimization terminated successfully:
First-order optimality is less than options.TolFun.

x =

0.4408 0.8660 0.2361

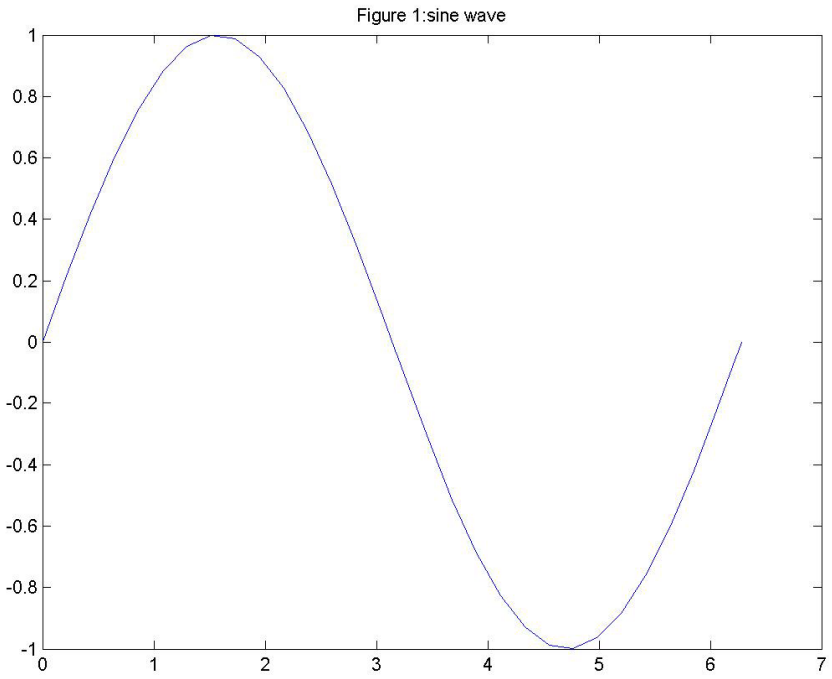
ΠΑΡΑΡΤΗΜΑ Δ. ΓΡΑΦΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΜΕ ΤΟ MATLAB

Δ.1 Η εντολή PLOT για την γραφική απεικόνιση συναρτήσεων

Η δημιουργία γραφικών παραστάσεων γίνεται με την εντολή plot. Στην πιο απλή μορφή της η εντολή συντάσσεται με την εξής ακολουθία:

Πίνακας Δ.1.1 Παράδειγμα εφαρμογής της εντολής plot

Εντολή	**Περιγραφή**
>>x=linspace(0, 2*pi, 30);	Κατασκευάζεται το διάνυσμα στήλη x που αποτελείται από 30 σημεία τα οποία απέχουν γραμμικά και παίρνουν τιμές από 0 έως 2π (συμπεριλαμβανομένων και των 0 και 2π).
>>y=sin(x);	Για κάθε τιμή του x υπολογίζονται οι τιμές του ημιτόνου που αποθηκεύονται στην μεταβλητή y.
>>plot(x,y), title('Figure 1:sine wave')	Κατασκευή του γραφήματος από τα ζεύγη τιμών (x,y). Με την εντολή title δίδεται ο τίτλος του γραφήματος. **Σημείωση 1**: Παρατηρούμε ότι είναι εφικτό να έχουμε 2 εντολές στην ίδια γραμμή, αρκεί να διαχωρίζονται με κόμμα. **Σημείωση 2**: Τα σημεία ενώνονται με ευθείες γραμμές. Οι άξονες κατασκευάζονται αυτόματα με τέτοια κλίμακα, ώστε να περιέχουν τα δεδομένα. Το χρώμα καθορίζεται αυτόματα.

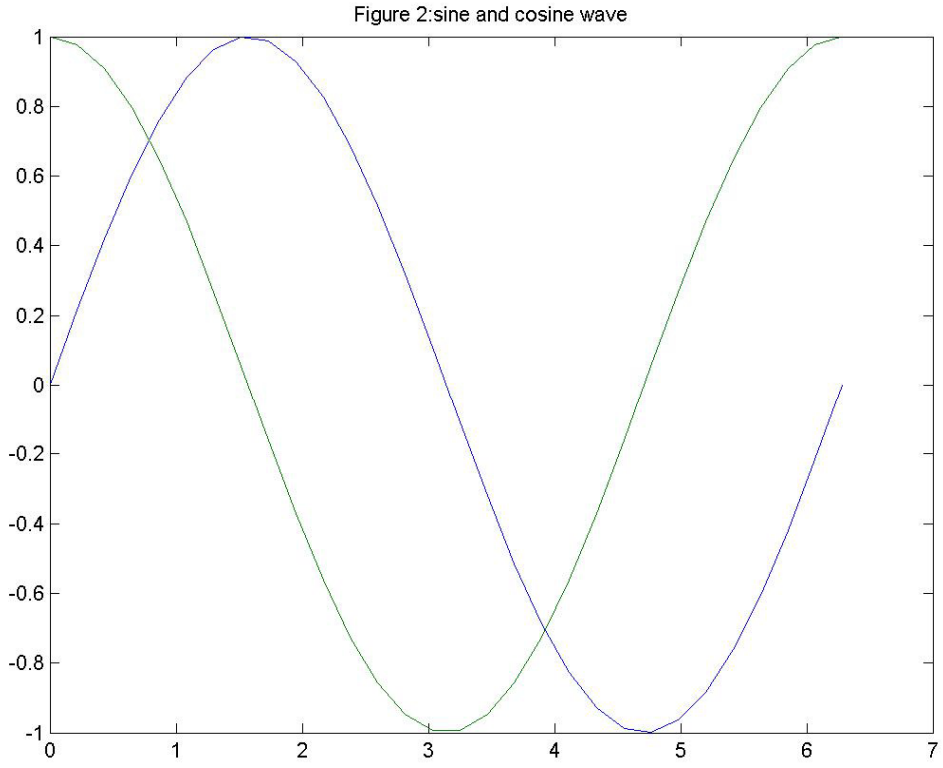


Σχήμα Δ.1.1 Εφαρμογή της εντολής plot

Μπορούν επίσης να σχεδιαστούν πολλαπλά διαγράμματα στο ίδιο γράφημα. Για παράδειγμα, έστω ότι στις ήδη υπάρχουσες γραμμές προσθέσω και αυτές του πίνακα Δ.1.2:

Πίνακας Δ.1.2 Πολλαπλές γραφικές παραστάσεις στο ίδιο γράφημα

Εντολή	**Περιγραφή**
>>z=cos(x);	Για κάθε τιμή του x υπολογίζονται οι τιμές του συνημιτόνου, οι οποίες αποθηκεύονται στην μεταβλητή z.
>>plot(x,y,x,z);	Κατασκευή του γραφήματος από τα ζεύγη (x,y) και (x,z). **Σημείωση:** Το matlab δίνει αυτόματα άλλο χρώμα στο δεύτερο διάγραμμα.
>>title('Figure 2:sine and cosine wave')	Στη γραφική παράσταση με την εντολή title δίνεται ο τίτλος του γραφήματος.



Δ.1.2 Εφαρμογή της εντολής plot για πολλαπλά διαγράμματα

Δ.2 Καθορισμός, τύπου γραμμής, ενδείκτη, χρώματος

Ο χρήστης έχει την ελευθερία να καθορίσει τον τύπο της γραμμής (linestyle), τον ενδείκτη στα δεδομένα (marker) καθώς και το χρώμα της γραμμής σύμφωνα με τον πίνακα Δ.2.1:

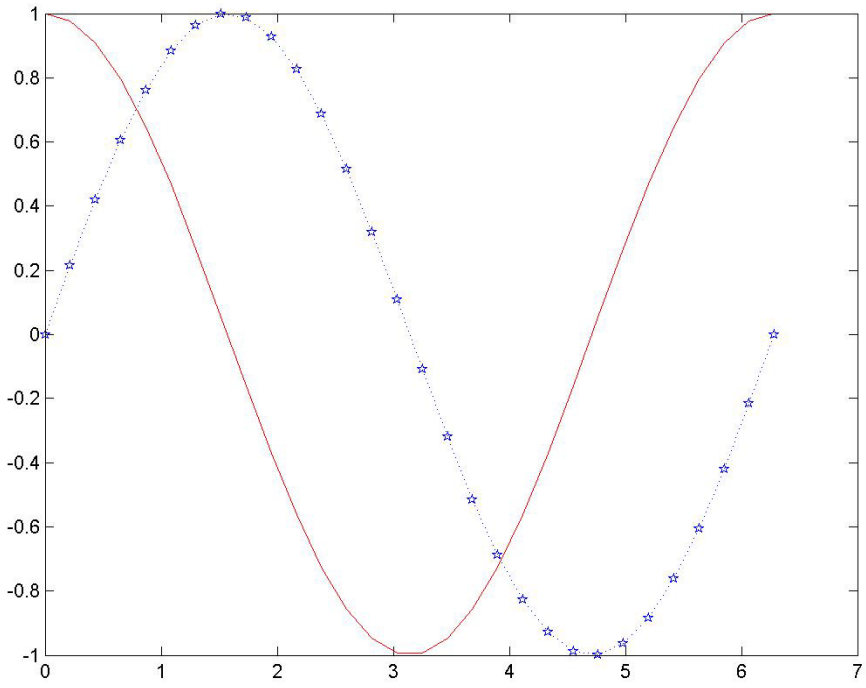
Πίνακας Δ.2.1 Καθορισμός, τύπου γραμμής, ενδείκτη, χρώματος

Σύμβολο	**Χρώμα**	**Σύμβολο**	**Ενδείκτης**	**Σύμβολο**	**Τύπος γραμμής**
b	Μπλέ	.	Τελεία	-	Γραμμή
g	Πράσινο	o	Κύκλος	:	Διάστικτη γραμμή
r	Κόκκινο	x	Σταυρός	-.	Ακολουθία παύλα-τελεία
c	Κυανό	+	Πρόσθεση	--	Γραμμή με παύλες
m	Ματζέντα	*	Αστερίσκος		
y	Κίτρινο	s	Τετράγωνο		
k	Μαύρο	d	Διαμάντι		
w	άσπρο	v	Τρίγωνο (κάτω)		
		^	Τρίγωνο (πάνω)		
		<	Τρίγωνο (αριστερά)		
		>	Τρίγωνο (δεξιά)		
		p	Αστέρι με πέντε κορυφές		
		h	Αστέρι με έξι κορυφές		

Αυτές οι ιδιότητες μπορούν να ρυθμιστούν όταν προστεθεί και ένα τρίτο όρισμα μετά από τα ζεύγη των τιμών μέσα σε απλά εισαγωγικά (‘ ‘). Για παράδειγμα η εντολή:

>>plot(x, y, ‘b:p’, x, z, ‘-r’)

σχεδιάζει το διάγραμμα από τα δεδομένα (x, y) και (x, z) χρησιμοποιώντας για τα (x, y) διάστικτη γραμμή (:), χρώμα μπλέ (b), και τα δεδομένα απεικονίζονται με αστέρι πέντε κορυφών (p), ενώ για τα (x, z) χρησιμοποιεί απλή γραμμή (-) με κόκκινο χρώμα (r). Το αποτέλεσμα φαίνεται στο σχήμα Δ.2.1



Σχήμα Δ.2.1 Καθορισμός, τύπου γραμμής, ενδείκτη, χρώματος

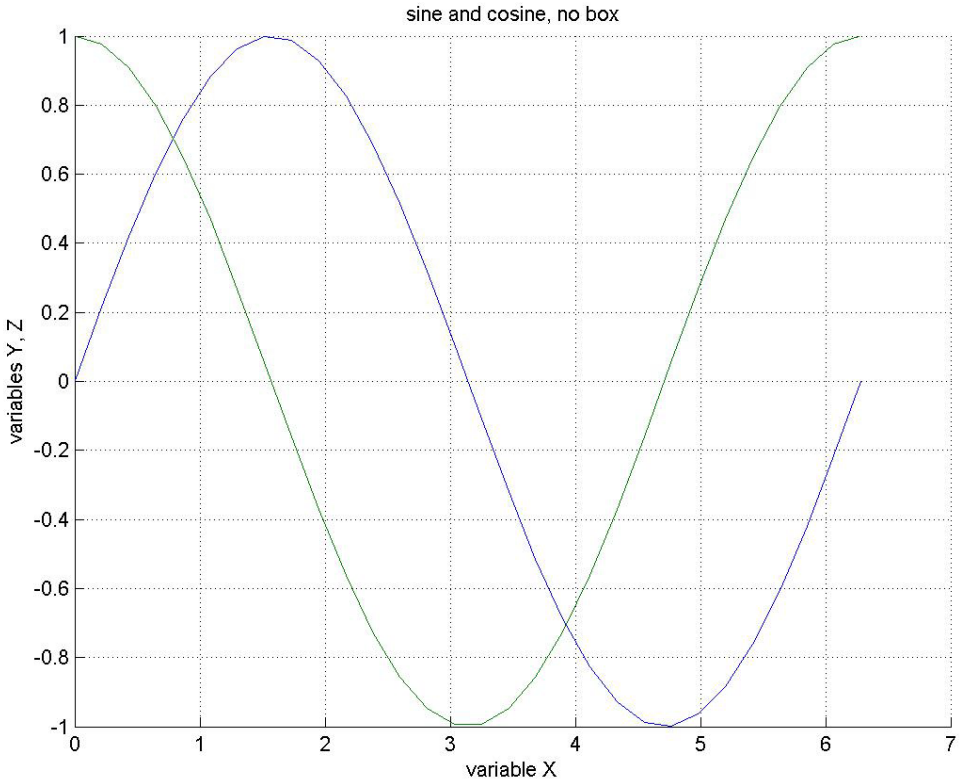
Σημείωση: Παρατηρούμε ότι το matlab έδωσε το μπλε χρώμα, και στην γραμμή αλλά και στους ενδείκτες. Αυτό το πρόβλημα μπορεί εύκολα να διορθωθεί, χρησιμοποιώντας τα γραφικά εργαλεία του παραθύρου γραφημάτων (figure window), (βλέπε ενότητα Δ.7).

Δ.3 Πλέγμα (GRID), Πλαίσιο αξόνων (AXES BOX), Ετικέτες (LABELS)

Με την εντολή **grid on** ενεργοποιείται το πλέγμα στην αντίστοιχη γραφική παράσταση. Η εντολή **grid off** απενεργοποιεί το πλέγμα (είναι η προκαθορισμένη επιλογή). Επίσης, στα διδιάστατα γραφικά, οι άξονες των γραφημάτων περικλείονται από ένα πλαίσιο (box). Αυτό το πλαίσιο μπορεί να απενεργοποιηθεί με την εντολή **box off**.
Τέλος με την εντολή:
xlabel('κείμενο'), δίνουμε τον τίτλο για τον άξονα x, και με την εντολή:
ylabel('κείμενο'), δίνουμε τον τίτλο για τον άξονα y. Ας δούμε το παρακάτω παράδειγμα:

```
>>x=linspace(0, 2*pi, 30);
>>y=sin(x);
>>z=cos(x);
>>plot(x,y,x,z)
>>box off
>>xlabel ('variable X')
>>ylabel ('variables Y, Z')
>>title ('sine and cosine, no box')
>>grid on
```

Θα πάρουμε το παρακάτω γράφημα



Σχήμα Δ.3.1 Παράδειγμα χρήσης ετικετών, πλέγματος και κουτιού αξόνων

Δ.4 Διαμόρφωση αξόνων

Το matlab δίνει απόλυτο έλεγχο, τόσο στην κλίμακα, όσο και στην εμφάνιση των αξόνων με την εντολή axis. Επειδή αυτή η εντολή έχει πάρα πολλά χαρακτηριστικά, θα περιγράψουμε τα πιο απαραίτητα που δίδονται στον πίνακα Δ.4.1

Πίνακας Δ.4.1 Η εντολή axis για τη διαμόρφωση των αξόνων γραφήματος

Εντολή	**Περιγραφή**
axis([xmin xmax ymin ymax])	Καθορίζονται τα μέγιστα και ελάχιστα όρια σε κάθε άξονα
axis ij	Ο άξονας τίθεται σε «matrix mode». Ο οριζόντιος άξονας αυξάνει από τα αριστερά προς τα δεξιά, ενώ ο κατακόρυφος άξονας αυξάνει από πάνω προς τα κάτω.
axis xy	Ο άξονας τίθεται σε Καρτεσιανή μορφή. Ο οριζόντιος άξονας αυξάνει από τα αριστερά προς τα δεξιά, ενώ ο κατακόρυφος άξονας αυξάνει από κάτω προς τα πάνω.
axis equal	Καθορίζει το λόγο 2 διαστάσεων (aspect ratio), ώστε ίσες αποστάσεις στο χωρισμό του άξονα, είναι ίσα σε μέγεθος
axis square	Κάνει το κουτί των αξόνων (axis box) τετράγωνο
axis vis3d	«Κλειδώνει» το λόγο 2 διαστάσεων (aspect ratio), έτσι ώστε

	να μπορεί να γίνει τρισδιάστατη περιστροφή (σε τρισδιάστατα αντικείμενα) χωρίς να γίνουν αλλαγές στο μέγεθος των αξόνων.

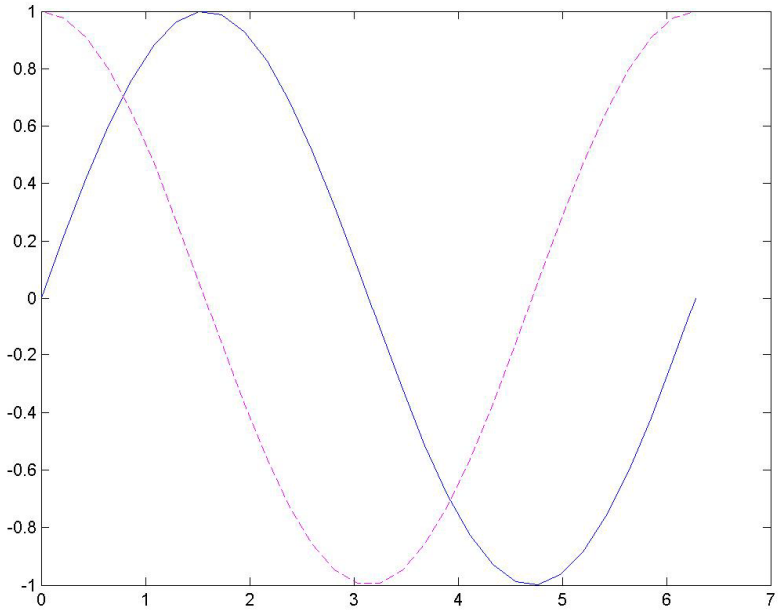
Σημείωση: Όταν κάποιος απλώς θέλει να αλλάξει τα όρια μόνο σε ένα άξονα, δεν μπορεί να χρησιμοποιήσει την εντολή axis διότι πρέπει να δώσει τα όρια και στον άλλο άξονα. Γι' αυτό το matlab δίνει τις συναρτήσεις xlim, ylim, zlim με την εξής σύνταξη:
xlim([xmin xmax]). Για περισσότερες πληροφορίες ανατρέξτε στο εγχειρίδιο του matlab.

Δ.5 Πολλαπλά γραφήματα

Μπορούμε να προσθέσουμε νέα γραφήματα στο ήδη υπάρχον χρησιμοποιώντας την εντολή hold. Όταν θέσουμε **hold on**, το matlab δεν καταργεί τους υπάρχοντες άξονες όταν χρησιμοποιείται πάλι η εντολή plot, αλλά στους ήδη υπάρχοντες προσθέτει καινούρια γραφική παράσταση. Όταν θέσουμε **hold off** τότε το παρόν παράθυρο (figure window) «ελευθερώνεται» και με κάθε νέα χρήση της εντολής plot δημιουργείται νέο παράθυρο. Για παράδειγμα:

```
>>x=linspace(0, 2*pi, 30);
>>y=sin(x);
>>z=cos(x);
>>plot(x,y)
>>hold on
>> plot(x,z,'- -m')
>>hold off
```

Το αποτέλεσμα φαίνεται στο σχήμα Δ.5.1



Σχήμα Δ.5.1 Εφαρμογή της εντολής hold on

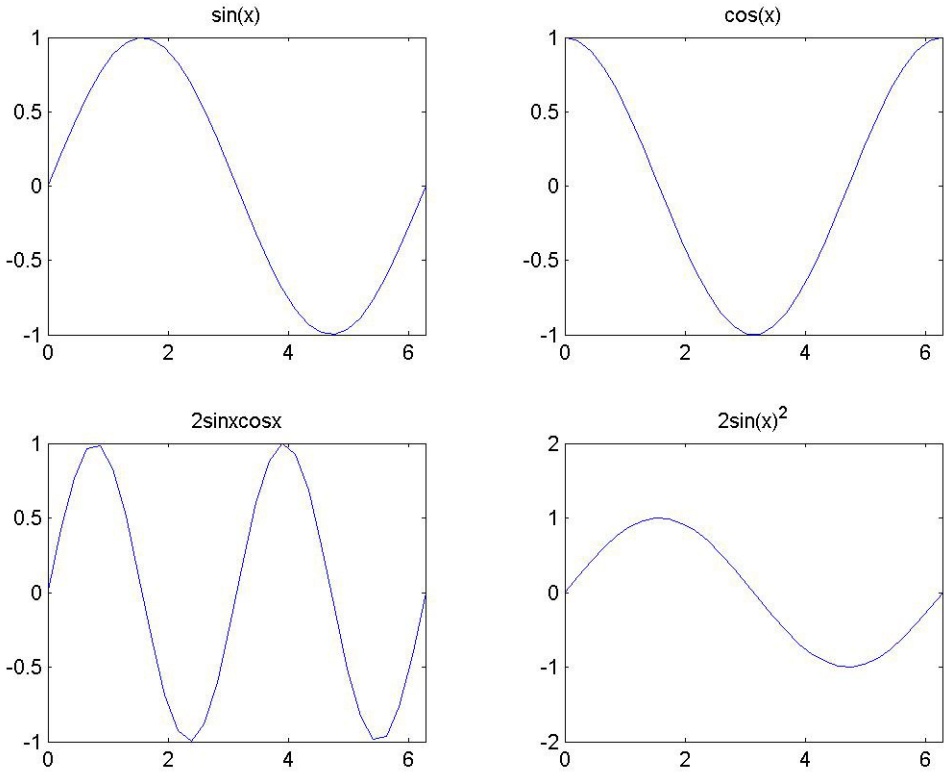
Σημείωση: Εφόσον υπάρχει η εντολή «hold on», με την δεύτερη πληκτρολόγηση της εντολής plot, παίρνουμε και δεύτερο γράφημα στο ίδιο παράθυρο γραφικών (figure window). Αν δεν υπήρχε η «hold on», η δεύτερη γραφική παράσταση θα δημιουργούταν σε νέο παράθυρο γραφικών.

Δ.6 Υπογραφήματα (SUBPLOTS)

Ένα παράθυρο γραφικών (figure window) μπορεί να περιέχει πολλαπλά γραφήματα, με διαφορετικούς άξονες το κάθε ένα. Η εντολή **subplot(m, n, p)** υποδιαιρεί το παράθυρο γραφικών σε m x n περιοχές, με ενεργή την p περιοχή. Η αρίθμηση των περιοχών ξεκινά στην πρώτη γραμμή από αριστερά προς τα δεξιά, συνεχίζει στη δεύτερη γραμμή πάλι από αριστερά προς τα δεξιά κτλ. Ας δούμε το παρακάτω παράδειγμα:

```
>>x=linspace(0, 2*pi, 30);
>>y=sin(x);
>>z=cos(x);
>>a=2*sin(x).*cos(x);  %πολλαπλασιασμός πινάκων στοιχείο-στοιχείο. Γι' αυτό το ( .* )
>>b=2*sin(x).^2;       %ύψωση σε δύναμη, δηλ πολλ/σμος πινάκων στοιχείο-στοιχείο ( .^ )
>>subplot(2,2,1)       %χωρίζω το figure window σε 2X2 περιοχές, με ενεργή την πρώτη
>>plot(x,y), axis([0 2*pi -1 1]), title('sin(x)')
>>subplot(2,2,2)       %χωρίζω το figure window σε 2X2 περιοχές, με ενεργή την δεύτερη
>>plot(x,z), axis([0 2*pi -1 1]), title('cos(x)')
>>subplot(2,2,3)       %χωρίζω το figure window σε 2X2 περιοχές, με ενεργή την τρίτη
>>plot(x,a), axis([0 2*pi -1 1]), title('2sinxcosx')
>>subplot(2,2,4)       %χωρίζω το figure window σε 2X2 περιοχές, με ενεργή την τέταρτη
>>plot(x,y), axis([0 2*pi -2 2]), title('2sin(x)^2')
```

Προκύπτει το σχήμα Δ.6.1:



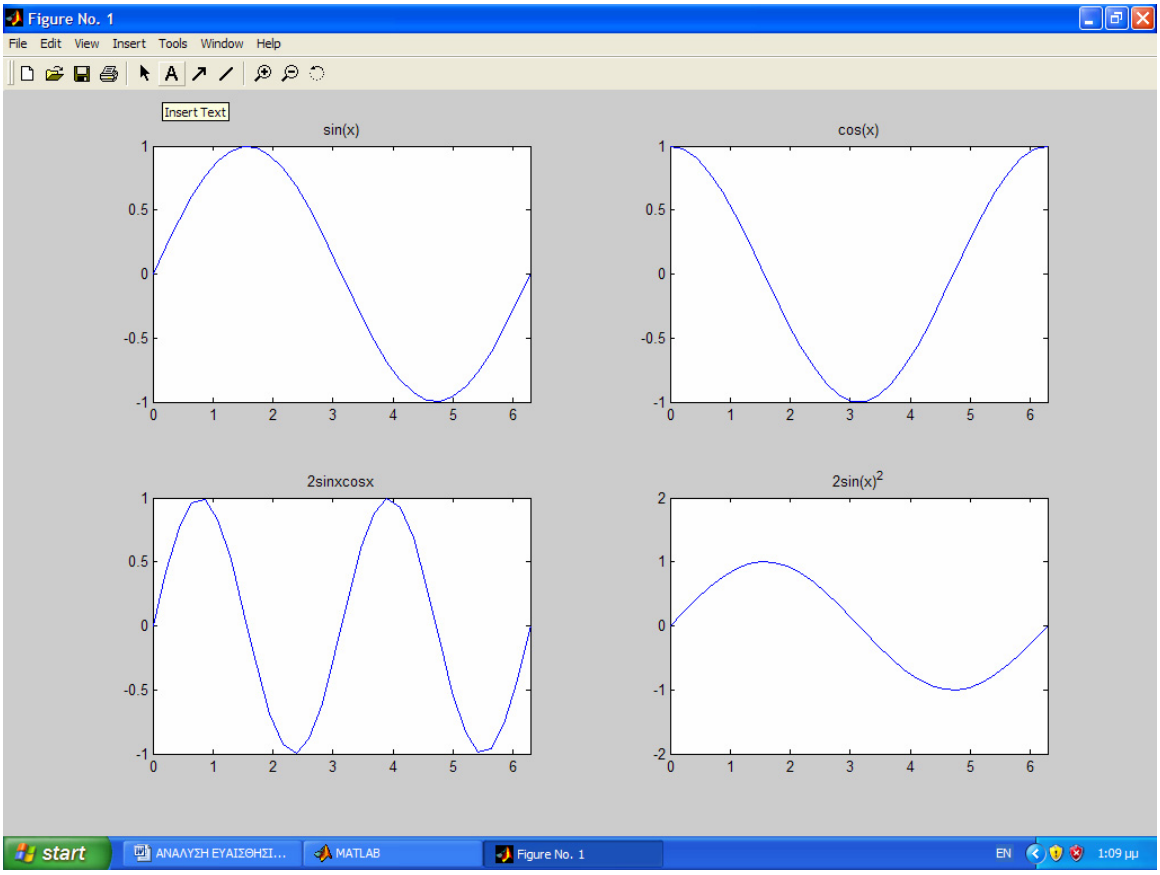
Σχήμα Δ.6.1 Παράδειγμα χρήσης υπογραφημάτων

Σημείωση 1: Όταν ένα συγκεκριμένο «subplot» είναι ενεργό, μόνο αυτό επηρεάζεται από τη χρήση των εντολών axis, hold, xlabel, ylabel, title, grid, box. Όλα τα άλλα subplots παραμένουν ανεπηρέαστα.

Σημείωση 2: Αν με νέα εντολή subplot καθορίζεται καινούρια υποδιαίρεση περιοχών, τα παρόντα subplots διαγράφονται.

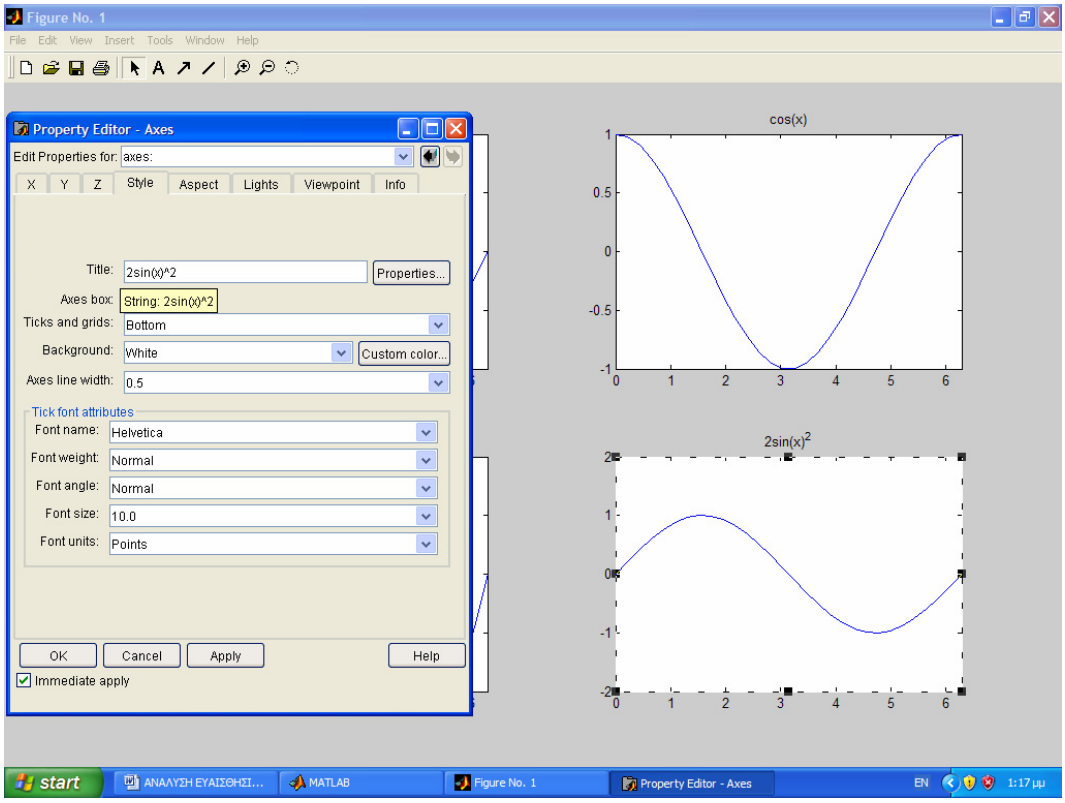
Δ.7 Χρήση του «figure window» για τον καθορισμό των ιδιοτήτων σε ένα γράφημα

Ένας άλλος τρόπος να αλλάξουμε τις ρυθμίσεις των subplots, αλλά και γενικά του κάθε γραφήματος, είναι μέσω του «figure window». Ένα παράδειγμα παρουσιάζεται στο σχήμα Δ.7.1



Σχήμα Δ.7.1 Παράδειγμα χρήσης του "figure window"

Από το μενού επιλέγουμε **Tools>Edit plot** και μπορούμε με το ποντίκι να επιλέξουμε κάποιο subplot και να αλλάξουμε τις ιδιότητές του (κλίμακα αξόνων, τίτλοι, χρώματα, ενδείκτες, λεζάντες κτλ.).
Για παράδειγμα, αφού επιλέξουμε **Tools>Edit plot**, με διπλό κλικ στο τέταρτο γράφημα εμφανίζεται η καρτέλα «Property editor», από την οποία μπορούμε με γραφικό τρόπο να αλλάξουμε τις ρυθμίσεις του γραφήματος αυτού (Βλέπε σχήμα Δ.7.2)



Σχήμα Δ.7.2 Παράδειγμα του παραθύρου «property editor»

Δ.8 Επιπλέον γραφικά εργαλεία

Αφού επιλέξουμε tools>edit plot μπορούμε να χρησιμοποιήσουμε και κάποια άλλα εργαλεία όπως:

- Πατώντας **insert>legend** μπορούμε να εισάγουμε αυτόματα μία λεζάντα στην γραφική παράσταση.
- Πατώντας **insert>arrow, text, line** διαδοχικά μπορούμε να εισάγουμε αντίστοιχα, ένα βέλος, ένα κείμενο, ή μία ευθεία γραμμή, τα οποία μπορούν να μετακινηθούν σε οποιοδήποτε σημείο του γραφήματος.
- Πατώντας **tools>zoom in**, κάνοντας αριστερό κλικ πάνω σε μια περιοχή του γραφήματος, γίνεται μεγέθυνση της περιοχής κατά ένα συντελεστή 2. Κάνοντας τώρα δεξί κλικ στο ίδιο γράφημα έχουμε μια σμίκρυνση της περιοχής κατά ένα συντελεστή 2.
- Πατώντας **tools>rotate 3D** μπορούμε με το ποντίκι να δώσουμε μια τρισδιάστατη απεικόνιση στο γράφημά.

Περισσότερες πληροφορίες και λειτουργίες στη βιβλιογραφία [1] και [6].

ΠΑΡΑΡΤΗΜΑ Ε
ΕΓΓΡΑΦΗ ΜΟΡΦΟΠΟΙΗΜΕΝΩΝ ΔΕΔΟΜΕΝΩΝ ΣΕ ΑΡΧΕΙΟ

Η εγγραφή μορφοποιημένων δεδομένων σε αρχείο, γίνεται με χρήση της εντολής fprintf. Η εντολή συντάσσεται ως εξής:

fprintf (fid,format,A,...)

Με αυτή την εντολή μετατρέπονται τα δεδομένα του πίνακα A (και όποιων άλλων επιπρόσθετων πινάκων) στην επιθυμητή μορφή και αποθηκεύονται στο αρχείο που συνδέεται με το όρισμα **fid**.

Το όρισμα **fid** προκύπτει με την χρήση της εντολής **fopen**.
Η εντολή **fopen** δημιουργεί ένα νέο αρχείο στο οποίο μπορούν να αποθηκευτούν τα μορφοποιημένα αποτελέσματα κάποιου προγράμματος.

Η εντολή fopen συντάσσεται ως εξής:

fid = fopen(filename,permission).

Η fopen ανοίγει το αρχείο filename, με τους όρους που καθορίζει το όρισμα permission.

Πίνακας Ε.1 Το όρισμα «permission»

'r'	Άνοιγμα αρχείου για ανάγνωση (default)
'w'	Άνοιγμα αρχείου, δημιουργία νέου αρχείου για εγγραφή, παραλείποντας υπάρχοντα περιεχόμενα, αν υπάρχουν
'a'	Άνοιγμα αρχείου, δημιουργία νέου αρχείου για εγγραφή, προσθέτοντας τα νέα περιεχόμενα στο τέλος του αρχείου
'r+'	Άνοιγμα αρχείου για ανάγνωση και για εγγραφή
'w+'	Άνοιγμα αρχείου, δημιουργία νέου αρχείου για ανάγνωση και για εγγραφή, παραλείποντας υπάρχοντα περιεχόμενα αν υπάρχουν
'a+'	Άνοιγμα αρχείου, δημιουργία νέου αρχείου για ανάγνωση και για εγγραφή, προσθέτοντας τα νέα περιεχόμενα στο τέλος του αρχείου

Σημείωση:Αν δεν είναι επιθυμητή η αποθήκευση των αποτελεσμάτων σε αρχείο, αλλά απλώς η εμφάνισή τους στην οθόνη του υπολογιστή θέτουμε **fid=1**.

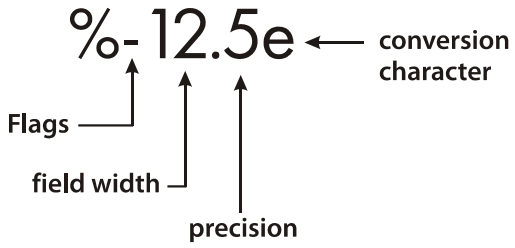
Το όρισμα **format** είναι μια ακολουθία χαρακτήρων που περιέχει κώδικα της γλώσσας C για την εφαρμογή της μετατροπής δεδομένων. Ο χρήστης έχει τη δυνατότητα να καθορίσει την στοίχιση των δεδομένων (alignment), τον αριθμό των σημαντικών ψηφίων (significant digits), το πλάτος του πεδίου που θα καλύπτουν τα δεδομένα (field width) κα. Η ακολουθία χαρακτήρων μπορεί να περιέχει και χαρακτήρες διαφυγής (escape characters) δηλαδή

χαρακτήρες για την αναπαράσταση μη-εκτυπώσιμων στοιχείων (όπως η δημιουργία νέας γραμμής, το κενό που δημιουργείται με το πάτημα του πλήκτρου tab κ.α).

Για να γίνει η μετατροπή, πρέπει να πληκτρολογήσουμε αρχικά τον χαρακτήρα (%) και έπειτα πρέπει να προστεθούν τα παρακάτω στοιχεία από τα οποία κάποια είναι προαιρετικά και κάποια υποχρεωτικά:

1. Σύμβολα στοίχισης αριθμών (Flags) *(προαιρετικό)*
2. Μέγεθος πεδίου και ακρίβεια (Width and precision fields) *(προαιρετικό)*
3. Δείκτης τύπου (subtype specifier) (όπως βλέπουμε την τελεία (.) πιο κάτω) *(προαιρετικό)*
4. Χαρακτήρας μετατροπής (Conversion character) *(υποχρεωτικό)*

Η σειρά γραφής των στοιχείων αυτών καθορίζεται στο σχήμα Ε.1.



Σχήμα Ε.1 Διαμόρφωση χαρακτήρων με την fprintf

Πίνακας Ε.2 Σύμβολα στοίχισης αριθμών (Flags)

Χαρακτήρας	Περιγραφή	παράδειγμα
Το σύμβολο της αφαίρεσης (-)	Αριστερή στοίχιση του αριθμού στο πεδίο του.	%-5.2d
Το σύμβολο της πρόσθεσης (+)	Τυπώνει το πρόσημο του αριθμού (+ ή -)	%+5.2d
Το μηδέν (0)	Τυπώνει μηδενικά αντί για κενά (όταν υπάρχουν κενές θέσεις στο πεδίο του αριθμού)	%05.2d

Πίνακας Ε.3 Μέγεθος πεδίου, ακρίβεια (Field width, precision)

Χαρακτήρας	Περιγραφή	παράδειγμα
Μέγεθος πεδίου	Αριθμός που καθορίζει τον ελάχιστο αριθμό των ψηφίων που θα τυπωθούν	%6f
Ακρίβεια	Αριθμός που συμπεριλαμβάνει το σύμβολο (.) και καθορίζει τον αριθμό των ψηφίων στα δεξιά της υποδιαστολής	%6.2f

Πίνακας Ε.4 Χαρακτήρες μετατροπής (Conversion characters)

Χαρακτήρας	Περιγραφή
%d	Αναπαράσταση δεκαδικού αριθμού
%e	Αναπαράσταση εκθετικού
%f	Αναπαράσταση fixed-point
%g	Πιο πρακτικό από το %e, %f. Τα μηδενικά που δεν χρειάζονται παραλείπονται

Στον προηγούμενο πίνακα παρατίθενται οι πιο κοινοί χαρακτήρες μετατροπής. Περισσότερες πληροφορίες στα αρχεία βοήθειας του matlab.

Πίνακας Ε.5 Χαρακτήρες διαφυγής (Escape characters).

Χαρακτήρας	Περιγραφή
\\	Εμφάνιση του συμβόλου backslash (\ ) μεταξύ των στοιχείων.
\f	Το κάθε στοιχείο του πίνακα A εμφανίζεται σε σχέση με τα επόμενα με συνεχή ροή, δηλαδή στη σειρά (form feed).
\n	Το κάθε στοιχείο του πίνακα A εμφανίζεται σε σχέση με τα επόμενα σε διαφορετική γραμμή.
\t	Το κάθε στοιχείο του πίνακα A εμφανίζεται σε σχέση με τα επόμενα σε απόσταση ενός διαστήματος του πλήκτρου «tab».

Στον προηγούμενο πίνακα παρατίθενται οι πιο κοινοί χαρακτήρες διαφυγής. Περισσότερες πληροφορίες στα αρχεία βοήθειας του matlab.

Όταν τελειώσει η εγγραφή των αποτελεσμάτων σε ένα αρχείο, τότε μπορούμε να το κλείσουμε με την εντολή **fclose**. Συντάσσεται ως εξής:

status = fclose(fid)
status = fclose('all')

Η εντολή **status = fclose(fid)** κλείνει το συγκεκριμένο αρχείο, αν είναι ανοικτό, που καθορίζεται από το όρισμα fid, επιστρέφοντας την τιμή 0 αν είναι επιτυχές το κλείσιμο και την τιμή -1 αν είναι ανεπιτυχές.

Η εντολή **status = fclose('all')** κλείνει όλα τα ανοικτά αρχεία, επιστρέφοντας την τιμή 0 αν είναι επιτυχές το κλείσιμο και την τιμή -1 αν είναι ανεπιτυχές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Duane Hanselman, Bruce Littlefield, "Mastering Matlab 6", Prentice Hall, 2001

[2] Laurene V. Fausette, "Applied Numerical Analysis Using Matlab", Prentice Hall, 1999

[3] John H. Mathews, Kurtis D. Fink "Numerical methods using Matlab", Prentice Hall, 1999

[4] William J.Palm III, "Matlab for engineering applications", Mc Graw Hill, 1999

[5] Ευάγγελος Χατζίκος, «Matlab 6 για Μηχανικούς», Εκδόσεις Τζιόλα, 2003

[6] Στερ. Κ. Κλημόπουλος, Αθαν.Γ.Τσουροπλής, «Από τη fortran77 στη fortran90», Εκδόσεις Πελεκάνος, 1994

[7] Papalambros and Wilde, "Principles of optimal design", Cambridge Univercity Press, 1988

ΗΛΕΚΤΡΟΝΙΚΑ ΑΡΧΕΙΑ ΒΟΗΘΕΙΑΣ

Optimization

[8] Matlab 6.5, 3rd CD, \help\pdf_doc\optim\optim_tb.pdf

[9] Matlab 6.5, 3rd CD, \help\pdf_doc\optim\rn.pdf

External interfaces

[10] Matlab 6.5, 3rd CD, \help\pdf_doc\matlab\apiext.pdf

[11] Matlab 6.5, 3rd CD, \help\pdf_doc\matlab\apiref.pdf