Βοηθητικό Κείμενο 1: Μια μικρή εισαγωγή στο Matlab

Περιεχόμενα

1. Εισαγωγή

Το MATLAB (το οποίο σημαίνει «MATrix LABoratory») είναι μια από τις πιο ευρέως χρησιμοποιούμενες πλατφόρμες λογισμικού μηχανικής στον κόσμο. Ο κύριος λόγος για την ευρεία υιοθέτησή του είναι ότι κάνει πολλές από τις πιο κοινές εργασίες στη μηχανική (επίλυση συστημάτων εξισώσεων, γραφική παράσταση) πολύ εύκολες. Τα διαγράμματα που δημιουργούνται από το MATLAB είναι επαρκούς ποιότητας για να είναι «έτοιμα για δημοσίευση» και είναι εύκολο να αναγνωριστούν διαγράμματα από το MATLAB σε μεγάλο μέρος της σύγχρονης τεχνολογίας.

Αυτό το μάθημα (Σχεδιασμός και ανάλυση μηχανισμών) θα δώσει έμφαση στη χρήση του MATLAB στη Μηχανολογική Σχεδίαση, αλλά θα βρείτε ότι το MATLAB είναι χρήσιμο και σε όλα τα άλλα μαθήματα μηχανικής σας. Παρά τη δύναμη και την απλότητα του MATLAB, πολλοί μαθητές το βρίσκουν πολύ απογοητευτικό στη χρήση – ειδικά στην αρχή. Δεδομένου ότι μεγάλο μέρος του MATLAB γράφτηκε τη δεκαετία του 1980, πολλά από τα μηνύματα λάθους είναι κρυπτικά, στην καλύτερη περίπτωση. Η ηλεκτρονική βοήθεια για το MATLAB είναι διεξοδική και καλά οργανωμένη, αλλά μερικές φορές είναι δύσκολο για τους αρχάριους να καταλάβουν το "lingo" (π.χ. είναι δύσκολο να κατανοήσουν πώς να χρησιμοποιήσουν έναν χειρισμό συνάρτησης εάν δεν γνωρίζουν τι είναι ο χειρισμός συνάρτησης!).

Αυτό το βοηθητικό υλικό θα σας παρουσιάσει μερικές από τις βασικές έννοιες του MATLAB. Θα επικεντρωθούμε στο να μάθουμε να κάνουμε πράγματα που θα είναι χρήσιμα αργότερα στο μάθημα, όπως η επίλυση συστημάτων εξισώσεων και η γραφική παράσταση.

2. Μια απλοποιημένη παρουσίαση: Το Command Window

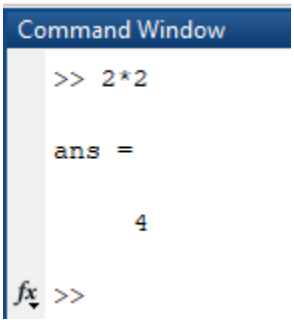
Όταν ξεκινάτε για πρώτη φορά το MATLAB, θα δείτε ένα παράθυρο με ένα μήνυμα κειμένου σε αυτό. Αυτό το παράθυρο ονομάζεται Παράθυρο εντολών (Command Window) και μπορείτε να πληκτρολογήσετε εντολές στη γραμμή εντολών και να λάβετε μια άμεση απάντηση. Άλλοι συνήθεις πίνακες περιλαμβάνουν τον Τρέχοντα φάκελο (Current Folder), ο οποίος περιέχει μια λίστα με τα αρχεία στον τρέχοντα φάκελο στον οποίο εργάζεστε και τον πίνακα Χώρος εργασίας (Workspace), ο οποίος παραθέτει όλες τις μεταβλητές του MATLAB που βρίσκονται αυτή τη

στιγμή στη μνήμη. Επιπλέον στο tab που βρίσκεται το Workspace μπορείτε να επιλέξετε να δείτε το πίνακα του Ιστορικού Εντολών (Command History). Εφόσον μόλις ξεκινήσατε, αυτό το παράθυρο θα πρέπει να είναι κενό (σε αντίθεση με την εικόνα που ακολουθεί).

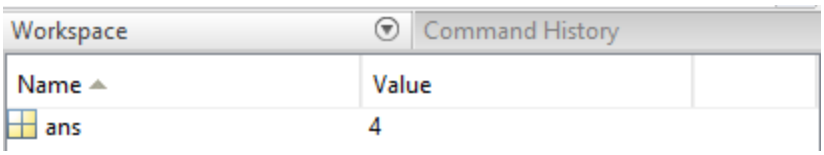
Στο πιο βασικό του επίπεδο, μπορείτε να χρησιμοποιήσετε το MATLAB ως μια (πολύ) ακριβή αριθμομηχανή. Στο παράθυρο εντολών πληκτρολογήστε:

>>2*2

και πατήστε Enter. Θα λάβετε την απάντηση



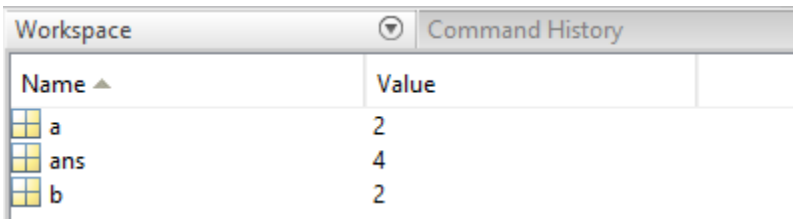
που είναι καθησυχαστικό. Το παράθυρο Workspace περιέχει τώρα τη μεταβλητή ans, η οποία αποθηκεύει την τιμή ό,τι υπολογίσατε τελευταία.



Αντί να πληκτρολογούμε αριθμούς απευθείας, μπορούμε να ορίσουμε μεταβλητές για την εκτέλεση της ίδιας λειτουργίας:

```
>> a=2

a =

     2

>> b=2

b =

     2

>> a*b

ans =

     4

>>
```

Αφού πληκτρολογήσετε αυτό, θα δείτε τις μεταβλητές a και b να εμφανίζονται στο παράθυρο Workspace, μαζί με τις τιμές τους.



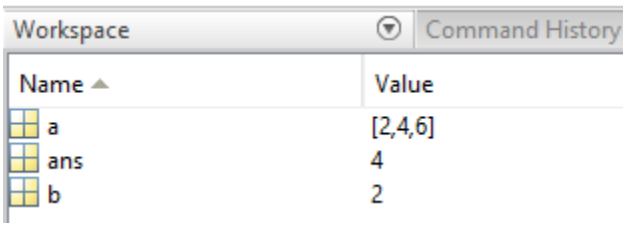
Το παράθυρο Workspace είναι μια πολύ εύχρηστη δυνατότητα που σας δίνει την ευκαιρία να βεβαιωθείτε ότι οι μεταβλητές σας αποθηκεύουν τις τιμές που πιστεύετε ότι θα έπρεπε να αποθηκεύουν.

Ο κύριος λόγος που τόσοι πολλοί μηχανικοί έχουν υιοθετήσει το MATLAB είναι η ικανότητά του να εργάζεται εύκολα με διανύσματα και πίνακες. Ένα διάνυσμα είναι απλώς μια σειρά (ή στήλη)

αριθμών και ένας πίνακας είναι ένας δισδιάστατος πίνακας αριθμών. Για να εισάγουμε ένα διάνυσμα στο MATLAB χρησιμοποιούμε αγκύλες:

```
>>  a=[2 4 6]

a =

     2     4     6
```

Σημείωση: Αν κοιτάξετε στο Workspace πλέον η μεταβλητή a άλλαξε από 2 στο διάνυσμα που μόλις θέσαμε. Η σημασία του Workspace όπως είπαμε παραπάνω είναι η ευκαιρία που σας δίνει να ελέγχετε ότι οι μεταβλητές που έχετε θέσει είναι οι σωστές και δεν άλλαξαν κατά λάθος.



Επειδή χρησιμοποιήσαμε κενά μεταξύ των αριθμών, το MATLAB έχει αποθηκεύσει το διάνυσμα ένα ως διάνυσμα γραμμή. Θα είχαμε πετύχει το ίδιο αποτέλεσμα αν χρησιμοποιούσαμε κόμματα μεταξύ των αριθμών.

```
>> a=[2,4,6]

a =

     2     4     6

>>
```

Αν κοιτάξετε στο παράθυρο Workspace, θα δείτε ότι το a now έχει την τιμή [2,4,6] και η προηγούμενη τιμή του 2 έχει αντικατασταθεί. Μερικές φορές προκαλεί σύγχυση για τους αρχάριους ότι ένα μόνο γράμμα (a) μπορεί να αποθηκεύσει πολλές τιμές όπως αυτή, αλλά είναι ένα από τα πράγματα που κάνουν το MATLAB τόσο ισχυρό. Μια άλλη ελαφρώς δύσκολη έννοια είναι αυτή του δείκτη. Ο δείκτης μπορεί να θεωρηθεί ως η διεύθυνση ενός συγκεκριμένου αριθμού σε ένα διάνυσμα. Δεδομένου ότι ο αριθμός 6 είναι αποθηκευμένος στην τρίτη θέση του a, ο δείκτης του είναι 3. Εάν θέλετε να αποκτήσετε πρόσβαση σε μια συγκεκριμένη καταχώρηση στο a, θα πρέπει να χρησιμοποιήσετε το δείκτη της σε κανονικές παρενθέσεις. Στο παράθυρο εντολών πληκτρολογήστε

```
>> a(3)

ans =

     6

>>
```

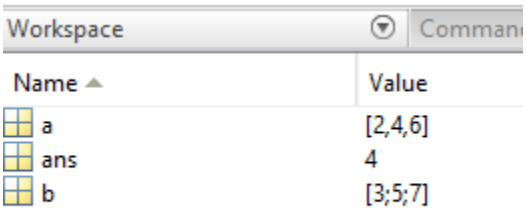
Δοκιμάστε επίσης:

```
>> a(2)

ans =

     4

>>
```

Στην πρώτη περίπτωση, ο δείκτης είναι 3 και έχουμε πρόσβαση στην τρίτη καταχώρηση στο a, και στη δεύτερη περίπτωση ο δείκτης είναι 2 - η δεύτερη καταχώρηση στο a έχει την τιμή 4. Τώρα ας προσπαθήσουμε να εισάγουμε ένα διάνυσμα στήλη

```
>> b = [3;5;7]

b =

     3
     5
     7

>>
```

Σημειώστε ότι χρησιμοποιήσαμε ερωτηματικά για να διαχωρίσουμε κάθε γραμμή και ότι τα ερωτηματικά έχουν εμφανιστεί επίσης στο παράθυρο Workspace.



Για να τονίσετε τη διαφορά μεταξύ a και b, πληκτρολογήστε τις ακόλουθες εντολές στο παράθυρο εντολών

```
>> size(a)

ans =

     1     3

>> size(b)

ans =

     3     1

>>
```

Η εντολή **size()** είναι μια ενσωματωμένη συνάρτηση του MATLAB που δίνει τις διαστάσεις ενός διανύσματος ή πίνακα. Η πρώτη διάσταση είναι ο αριθμός των σειρών και η δεύτερη διάσταση είναι ο αριθμός των στηλών. Έτσι, το διάνυσμα α έχει μία γραμμή και τρεις στήλες: είναι διάνυσμα γραμμή. Το διάνυσμα b έχει τρεις γραμμές και μία στήλη: είναι διάνυσμα στήλη. Τώρα δοκιμάστε να πληκτρολογήσετε την ακόλουθη (ελαφρώς πονηρή) εντολή:

```
>> b(3)

ans =

     7

>>
```

Μπορείτε να καταλάβετε γιατί αυτό είναι πονηρό; Χρησιμοποιήσαμε την εντολή a(3) για πρόσβαση στην τρίτη στήλη στο a και b(3) για πρόσβαση στην τρίτη σειρά στο b, αλλά η εντολή λειτούργησε μια χαρά και στις δύο περιπτώσεις. Αυτή η συμπεριφορά είναι, δυστυχώς, η πηγή σύγχυσης για πολλούς αρχάριους προγραμματιστές MATLAB. Για να καταλάβουμε γιατί συμβαίνει αυτό, θα πρέπει να δούμε λίγο πιο προσεκτικά τις διανυσματικές πράξεις. Στο παράθυρο εντολών πληκτρολογήστε

```
>> a*b

ans =

    68

>>
```

Στη συνέχεια, πληκτρολογήστε

```
>>  b*a

ans =

     6    12    18
    10    20    30
    14    28    42

>>
```

Δεδομένου ότι έχουμε συνηθίσει την αντιμεταθετική ιδιότητα στον πολλαπλασιασμό, αυτό το αποτέλεσμα είναι πολύ περίεργο. Να θυμάστε ότι πολλαπλασιάζοντας το a και το b, πολλαπλασιάζουμε δύο διανύσματα, όχι δύο αριθμούς.

Στην πρώτη περίπτωση, υπολογίσαμε το εσωτερικό γινόμενο, το οποίο θα συζητήσουμε διεξοδικά στο μάθημα. Το εσωτερικό γινόμενο ορίζεται ως:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

και αν πραγματοποιήσετε αυτές τις πράξεις στο μυαλός σας, θα διαπιστώσετε ότι η απάντηση είναι, πράγματι, 68. Η άλλη πράξε είναι γνωστή ως εξωτερικό προϊόν (προσοχή όχι γινόμενο!!!)

$$\mathbf{b} \otimes \mathbf{a} = \begin{bmatrix} a_1 b_1 & a_2 b_1 & a_3 b_1 \\ a_1 b_2 & a_2 b_2 & a_3 b_2 \\ a_1 b_3 & a_2 b_3 & a_3 b_3 \end{bmatrix}$$

Παρακαλώ μην απομνημονεύσετε τη φόρμουλα για το εξωτερικό προϊόν - αυτή είναι η τελευταία φορά που θα το δούμε! Το σημαντικό πράγμα που πρέπει να παρατηρήσετε είναι το μέγεθος της εξόδου σε κάθε περίπτωση. Στην πρώτη περίπτωση, το μέγεθος του a*b είναι 1 × 1, με άλλα λόγια ένας μοναδικός αριθμός (ένα βαθμωτό μέγεθος). Το μέγεθος του b*a, από την άλλη πλευρά, είναι 3 × 3. Στην πρώτη περίπτωση, πραγματοποιήσαμε την πράξη

$$(1\times 3)^*(3\times 1)\rightarrow(1\times 1)$$

Και στην δεύτερη περίπτωση:

$$(3\times 1)^*(1\times 3)\rightarrow(3\times 3)$$

Έτσι, το αποτέλεσμα ενός διανυσματικού πολλαπλασιασμού έχει το μέγεθος των εξωτερικών διαστάσεων των δύο διανυσμάτων. Όπως θα δούμε σε λίγο, οι εσωτερικές διαστάσεις των δύο διανυσμάτων πρέπει να είναι ίδιες για να λειτουργήσει ο πολλαπλασιασμός.

Για να μετατρέψουμε ένα διάνυσμα γραμμής σε διάνυσμα στήλης μπορούμε να χρησιμοποιήσουμε τη λειτουργία αναστροφής. Με μαθηματικούς συναισθηματικούς όρους αυτό σημαίνει ότι

$$\mathbf{a}^T = \begin{Bmatrix} 2 \\ 4 \\ 6 \end{Bmatrix}$$

$$\mathbf{b}^T = \begin{Bmatrix} 3 & 5 & 7 \end{Bmatrix}$$

Στο MATLAB εκτελούμε την αναστροφή χρησιμοποιώντας την απόστροφο. Έτσι, μπορούμε να πληκτρολογήσουμε

```
>>  a'

ans =

     2
     4
     6

>>
```

Τώρα δοκιμάστε να χρησιμοποιήσετε τον τελεστή αναστροφής για να πολλαπλασιάσετε δύο διανύσματα στήλες.

```
>> a'*b
Error using  *
Inner matrix dimensions must agree.

fx >>
```

Συγχαρητήρια – λάβατε το πρώτο σας κρυπτικό μήνυμα σφάλματος MATLAB! Λάβαμε το μήνυμα επειδή προσπαθήσαμε να εκτελέσουμε έναν πολλαπλασιασμό που δεν ορίζεται:

$$(3\times1)*(3\times1) \rightarrow \text{undefined}$$

Τώρα που ξέρουμε τι σημαίνει το μήνυμα σφάλματος, θα πρέπει να είναι λίγο λιγότερο κρυπτικό. Εάν λαμβάνετε αυτό το μήνυμα όταν γράφετε ένα περίπλοκο πρόγραμμα, σημαίνει σχεδόν πάντα ότι μια από τις μεταβλητές που νομίζατε ότι ήταν διάνυσμα γραμμή είναι στην πραγματικότητα ένα διάνυσμα στήλη ή το αντίστροφο. Για να αποφύγουμε τέτοιου είδους προβλήματα στο μέλλον θα υιοθετήσουμε την ακόλουθη σύμβαση.

1. Εάν μια μεταβλητή έχει δύο διαστάσεις – ας πούμε τις συντεταγμένες x και y ενός σημείου, θα αποθηκεύσουμε τις δύο διαστάσεις ως διάνυσμα στήλης. Έτσι, οι συντεταγμένες του σημείου C θα αποθηκευτούν ως

$$\mathbf{x}_C = \begin{Bmatrix} x_C \\ y_C \end{Bmatrix}$$

Και όχι

$$x_C = \begin{Bmatrix} x_C & y_C \end{Bmatrix}$$

2. Κατά τον υπολογισμό μιας μεταβλητής σε πολλές χρονικές στιγμές – ας πούμε τη γωνία σύζευξης για πολλαπλές τιμές της γωνίας στροφάλου, θα αποθηκεύσουμε τα αποτελέσματα σε διάνυσμα σειρά. Έτσι, η γωνία ζεύξης που βρέθηκε σε τρεις χρονικές στιγμές θα είναι

$$\theta_3 = \begin{Bmatrix} \theta_3(1) & \theta_3(2) & \theta_3(3) \end{Bmatrix}$$

3. Εδώ είναι το δύσκολο: όταν υπολογίζουμε μια δισδιάστατη μεταβλητή σε πολλές χρονικές στιγμές – ας πούμε τη θέση του σημείου P του εμβόλου για πολλαπλές γωνίες στροφάλου, θα αποθηκεύουμε κάθε στοιχείο στη δική του σειρά, με κάθε στήλη να αντιπροσωπεύει μια ξεχωριστή στιγμή στο χρόνο:

$$\mathbf{x}_P = \begin{bmatrix} x_P(1) & x_P(2) & x_P(3) \\ y_P(1) & y_P(2) & y_P(3) \end{bmatrix}$$

Εάν παραμείνουμε συνεπείς με αυτήν τη σύμβαση, τότε θα αποφύγουμε το μήνυμα σφάλματος που εμφανίζεται παραπάνω. Το γεγονός ότι μπορείτε να αποκτήσετε πρόσβαση στην τρίτη καταχώριση ενός διανύσματος γραμμή ή ενός διανύσματος στήλη πληκτρολογώντας a(3) ή b(3) μπορεί να οδηγήσει σε σύγχυση επειδή το MATLAB θα επιλέξει την τρίτη καταχώρηση εκ των δύο. Θα ήταν καλύτερο μακροπρόθεσμα εάν το MATLAB απαιτούσε από τον χρήστη να πληκτρολογήσει a(1,3) για την τρίτη καταχώρηση ενός διανύσματος γραμμή και b(3,1) για την τρίτη καταχώρηση ενός διανύσματος στήλη, αλλά εν γένει αυτό δεν αλλάζει εύκολα. Η ουσία είναι η εξής: εάν λάβετε ένα μήνυμα σφάλματος που λέει ότι οι διαστάσεις του εσωτερικού πίνακα πρέπει να είναι ίσες, το πρώτο πράγμα που πρέπει να ελέγξετε είναι εάν ένα διάνυσμα γραμμς έχει εισαχθεί ως στήλη ή το αντίστροφο. Το παράθυρο Workspace είναι ένα καλό μέρος για να το ελέγξετε αυτό.

Παρεμπιπτόντως, εάν θέλετε να εισαγάγετε έναν πίνακα, όπως ο πίνακας x_P που φαίνεται παραπάνω, θα πληκτρολογήσετε

```
>> A =[2 4 6; 3 5 7]

A =

     2     4     6
     3     5     7

fx >>
```

Παρατηρήστε ότι οι σειρές έχουν διαχωριστεί με ερωτηματικό και οι στήλες χωρίζονται με κενά. Ο πίνακας Α έχει διαστάσεις 2 × 3. Για πρόσβαση στην τιμή στην πρώτη γραμμή, τρίτη στήλη του Α, πληκτρολογήστε

```
>> A(1,3)

ans =

     6

fx >>
```

Και για πρόσβαση στην τιμή στη δεύτερη σειρά, δεύτερη στήλη τύπου Α

```
>>  A(2,2)

ans =

     5

fx >>
```

Τώρα δοκιμάστε να εισαγάγετε τα παρακάτω

```
>> A(3,2)
Index exceeds matrix dimensions.

fx >>
```

Αυτό είναι ένα άλλο μήνυμα σφάλματος που συναντάται συχνά. Σημαίνει ότι ζητήσαμε να δούμε ένα μέρος του πίνακα Α που δεν υπάρχει αφού έχει μόνο δύο σειρές και ζητήσαμε μια καταχώρηση στην τρίτη σειρά. Όταν λάβετε αυτό το μήνυμα, θα πρέπει να εξετάσετε τον ορισμό της μεταβλητής στο Workspace για να βεβαιωθείτε ότι έχει τις σωστές διαστάσεις και επίσης να βεβαιωθείτε ότι δεν έχετε αλλάξει κατά λάθος τα ορίσματα γραμμής και στήλης μέσα στις παρενθέσεις. Αν πληκτρολογήσετε

```
>> A(2,3)

ans =

     7

>>
```

έχετε το αναμενόμενο αποτέλεσμα. Για να ελέγξετε την κατανόησή σας για τις παραπάνω έννοιες, ποια από τις ακόλουθες λειτουργίες ορίζεται (δηλαδή, ταιριάζουν οι εσωτερικές διαστάσεις);

1. **A*b**	2. **A*a**
3. **b*A**	4. **a*A**
5. **b*A**^{T}	6. **a*A**^{T}

Απάντηση: τα στοιχεία 1 και 6 ορίζονται, τα υπόλοιπα όχι.

Παρεμπιπτόντως, θα υιοθετήσουμε τη σύμβαση αποθήκευσης διανυσμάτων ως πεζά γράμματα και πίνακες ως κεφαλαία γράμματα. Τόσο οι πίνακες όσο και τα διανύσματα είναι γραμμένα με έντονη γραμματοσειρά.

4. Διανυσματική σημειογραφία στο Matlab

Η εισαγωγή διανυσμάτων έναν αριθμό κάθε φορά είναι καλή για μικρά διανύσματα, αλλά θα υπάρξουν στιγμές που θέλουμε να ορίσουμε διανύσματα με εκατοντάδες, ή ακόμα και χιλιάδες, καταχωρήσεις. Ευτυχώς, το MATLAB έχει έναν εύκολο τρόπο να το κάνει αυτό χρησιμοποιώντας τον απλό τελεστή άνω και κάτω τελείας. Δοκιμάστε να πληκτρολογήσετε στο παράθυρο εντολών

```
>> a=0:10

a =

     0     1     2     3     4     5     6     7     8     9    10

>>
```

Η άνω και κάτω τελεία λέει στο MATLAB ότι καθορίζουμε ένα διάνυσμα που ξεκινά από το μηδέν και τελειώνει στο δέκα, με διάστημα ένα μεταξύ κάθε καταχώρησης. Αν θέλουμε διαφορετική απόσταση, τοποθετούμε την προσαύξηση μεταξύ του πρώτου και του τελευταίου αριθμού

```
>> a=0:2:10

a =

     0     2     4     6     8    10

fx >>
```

δίνει τους ζυγούς αριθμούς που ξεκινούν από το μηδέν και τελειώνουν στο δέκα. Τι θα γινόταν αν θέλαμε τη λίστα των περιττών αριθμών μεταξύ 1 και 11; Θα μπορούσαμε να χρησιμοποιήσουμε

```
>> b=1:2:11

b =

     1     3     5     7     9    11

fx >>
```

Ή, ακόμα καλύτερα, θα μπορούσαμε να χρησιμοποιήσουμε τον ορισμό του a για να υπολογίσουμε τους περιττούς αριθμούς

```
>> b = a + 1

b =

     1     3     5     7     9    11

fx >>
```

Η δεύτερη μέθοδος μπορεί να είναι λίγο περίεργη. Εξάλλου, το a είναι το διάνυσμα των ζυγών αριθμών από το μηδέν έως το δέκα, αλλά το "1" είναι βαθμωτό μέγεθος. Το MATLAB το ερμηνεύει ως εντολή για προσθήκη 1 σε κάθε καταχώρηση στο διάνυσμα a. Ομοίως, θα μπορούσαμε να χρησιμοποιήσουμε το διάνυσμα a για να υπολογίσουμε τις πρώτες έξι ζυγές δυνάμεις του 2.

```
>> c=2^a
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.

fx >>
```

Ωχ! Εδώ είναι ένα άλλο κρυπτικό μήνυμα σφάλματος MATLAB. Το MATLAB χρησιμοποιεί τον τελεστή caret (^) ως συντομογραφία για τον πολλαπλασιασμό ενός τετραγωνικού πίνακα με τον εαυτό του, όπου ένας τετραγωνικός πίνακας έχει τον ίδιο αριθμό γραμμών και στηλών. Αν το A είναι τετράγωνος πίνακας, τότε A^2 = A*A. Αυτό που θέλαμε να κάνουμε με το c = 2^a είναι να

αυξήσουμε τον αριθμό 2 στη δύναμη καθενός από τους αριθμούς του a. Ευτυχώς, το MATLAB μας έδωσε μια υπόδειξη για το πώς να το κάνουμε αυτό - τη σημειογραφία dot-carat (.^)

```
>> c=2.^a

c =

     1     4    16    64   256  1024

fx >>
```

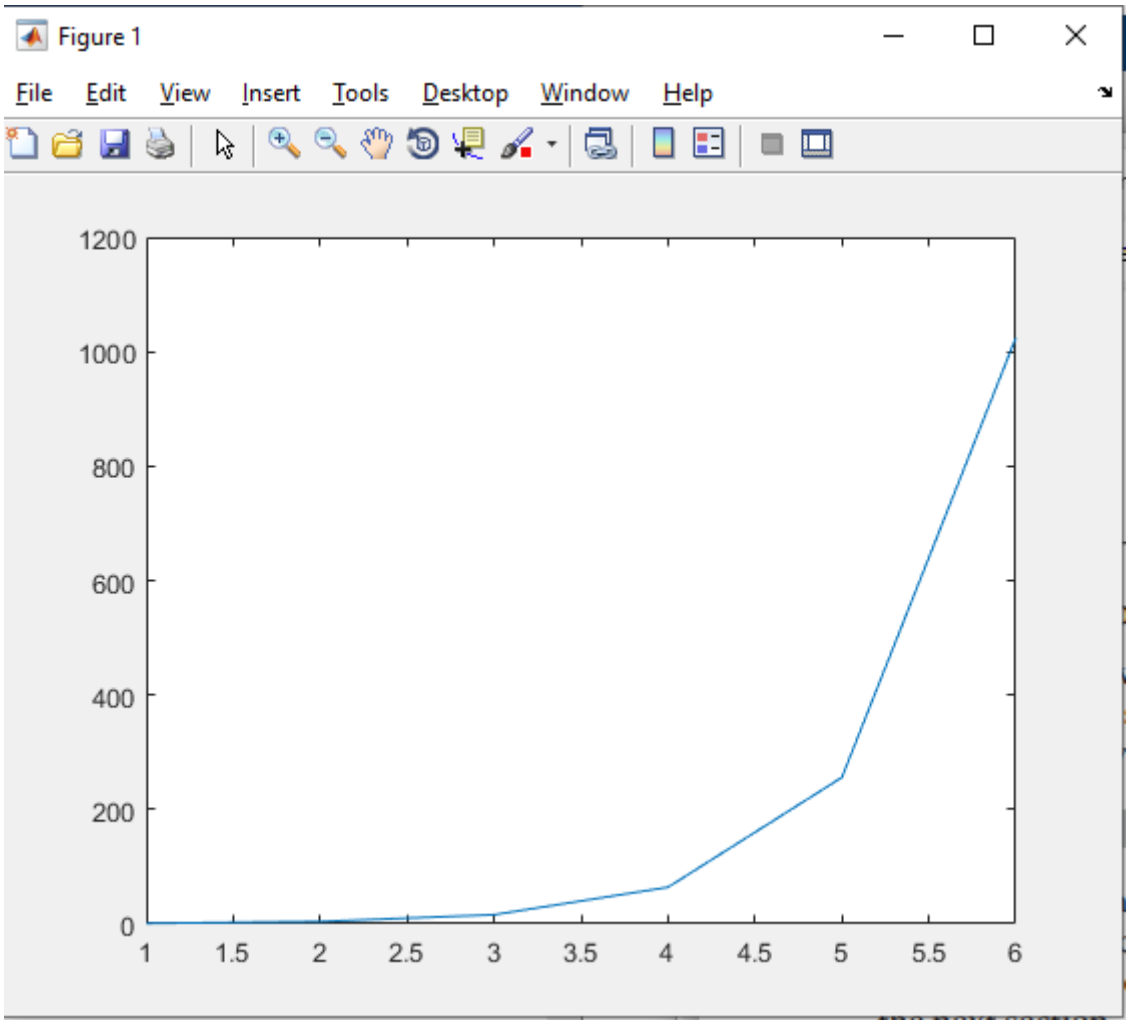
Πολύ καλύτερα! Η τοποθέτηση της τελείας πριν από το carat (ή ένα σύμβολο πολλαπλασιασμού) λέει στο MATLAB να εκτελέσει τη λειτουργία σε κάθε στοιχείο του διανύσματος ή πίνακα και όχι στο διάνυσμα (ή τον πίνακα) ως σύνολο. Δεν είναι καν σαφές τι θα σήμαινε η αύξηση του αριθμού 2 στη δύναμη ενός ολόκληρου διανύσματος!

5. Το πρώτο διάγραμμα

Τώρα έχουμε ένα διάνυσμα c που περιέχει τις πρώτες έξι ζυγές δυνάμεις του 2. Μπορεί να είναι ενδιαφέρον να το σχεδιάσουμε έτσι ώστε να παρατηρήσουμε την εκθετική του αύξηση. Η εντολή για σχεδίαση στο MATLAB είναι πολύ απλή

plot(c)

Αφού πατήσετε Enter, θα πρέπει να ανοίξει ένα νέο παράθυρο σχεδίασης που μοιάζει με το σχήμα που ακολουθεί. Είναι μια εκθετικά αυξανόμενη καμπύλη, όπως περιμέναμε. Το διάγραμμα είναι ελλιπές ως έχει γιατί δεν έχουμε τοποθετήσει ετικέτες στους άξονες ή τίτλο. Θα μάθουμε να το κάνουμε αυτό ως μέρος ενός σεναρίου στην επόμενη ενότητα.

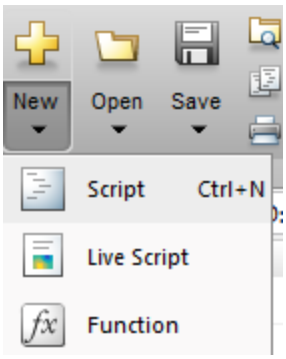


6. Γράφοντας ένα απλό script

Θεωρητικά, θα μπορούσαμε να κάνουμε σχεδόν όλη τη μοντελοποίηση μας στο παράθυρο εντολών πληκτρολογώντας μια μεγάλη σειρά εντολών, τη μία μετά την άλλη. Η δύναμη του MATLAB, ωστόσο, έγκειται στην ικανότητά του να εκτελεί script. Ένα script είναι μια σειρά από εντολές του MATLAB που αποθηκεύουμε σε ένα αρχείο που ονομάζεται m-file. Για παράδειγμα, θα μπορούσαμε να γράψουμε ένα script, που ονομάζεται SinePlot.m, το οποίο θα σχεδίαζε ένα ημιτονοειδές κύμα για εμάς και θα περιέχει όλες τις εντολές που είναι απαραίτητες για να είναι έτοιμο το τελικό διάγραμμα προς αναφορά (ετικέτες άξονα, τίτλος, κ.λπ.)

Η αποθήκευση των εντολών σε ένα script μας δίνει τη δυνατότητα να κάνουμε μικρές «αλλαγές» στο διάγραμμα χωρίς να χρειάζεται να πληκτρολογήσουμε ξανά τα πάντα. Τώρα θα αφήσουμε πίσω το παράθυρο εντολών, γιατί θα περάσουμε το υπόλοιπο σεμινάριο στον Επεξεργαστή (Editor).

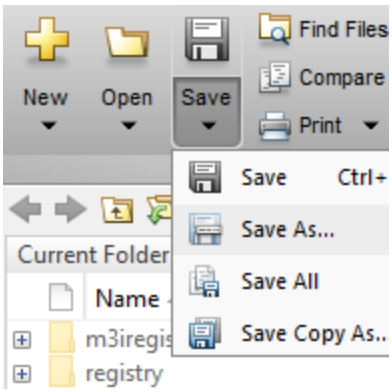
Στην καρτέλα Αρχική σελίδα στο παράθυρο εντολών κάντε κλικ στο κουμπί Νέο και επιλέξτε script από το αναπτυσσόμενο μενού.

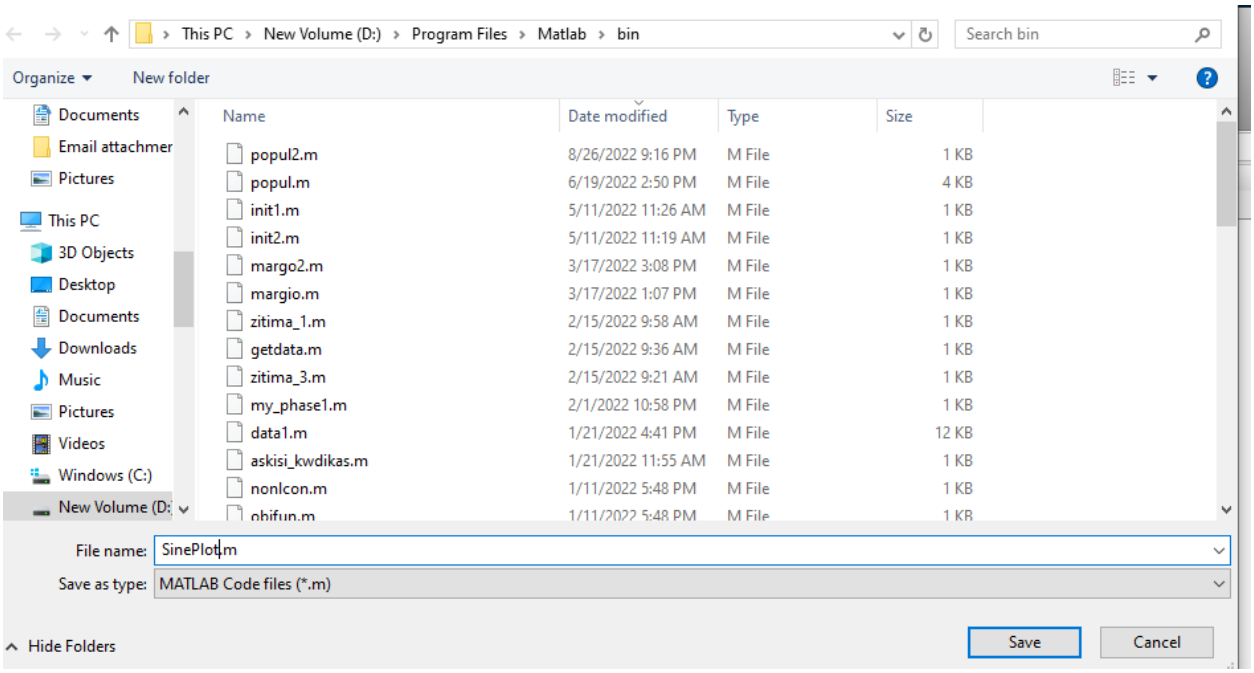


Θα ανοίξει ένα νέο παράθυρο: το παράθυρο Editor.

Θα χρησιμοποιήσουμε το παράθυρο Editor για να εισάγουμε τα script μας ως αρχεία κειμένου. Όταν τελειώσουμε με την πληκτρολόγηση του script, μπορούμε να πατήσουμε το κουμπί Εκτέλεση (Run) (ή το πλήκτρο F5) για να το εκτελέσουμε. Το αποτέλεσμα θα είναι το ίδιο, ως επί το πλείστον, σαν να είχαμε πληκτρολογήσει κάθε εντολή, τη μία μετά την άλλη, στο παράθυρο εντολών.

Το πρώτο πράγμα που πρέπει να κάνετε όταν εισάγετε ένα νέο script είναι να χρησιμοποιήσετε το Save As για να το ονομάσετε. Όταν εμφανιστεί το παράθυρο διαλόγου Αποθήκευση ως, περιηγηθείτε στον επιθυμητό φάκελο (π.χ. στην επιφάνεια εργασίας) και αποθηκεύστε το αρχείο ως SinePlot.m.





Επειδή το MATLAB γράφτηκε αρχικά στα παλιά χρόνια του DOS, δεν επιτρέπει κενά ή ειδικούς χαρακτήρες στα ονόματα των αρχείων. Με άλλα λόγια, μην προσπαθήσετε να το αποθηκεύσετε ως Sine Plot.m - θα λάβετε ένα πολύ μυστηριώδες μήνυμα σφάλματος όταν προσπαθήσετε να εκτελέσετε το σενάριο αργότερα. Τα ονόματα αρχείων MATLAB πρέπει να ξεκινούν με γράμμα και δεν μπορούν να περιέχουν κενά.

Το πρώτο μέρος κάθε script που γράφετε πρέπει να περιέχει ένα σύνολο σχολίων που περιγράφουν τον σκοπό του προγράμματος και τη λειτουργία του. Ένα σχόλιο είναι ένα μέρος του script που δεν εκτελείται, αλλά χρησιμεύει για να δώσει στον αναγνώστη σημαντικές πληροφορίες σχετικά με το script. Ο σχολιασμός του κώδικά σας είναι ένα από τα πιο σημαντικά πράγματα που θα κάνετε ως προγραμματιστής, γιατί αν γράψετε κάτι χρήσιμο, κάποιος άλλος θα θέλει να το χρησιμοποιήσει. Επιπλέον, ο μελλοντικός σας εαυτός θα είναι ευγνώμων στον παρόντα εαυτό σας για τον σχολιασμό του κώδικα, αφού ο μελλοντικός σας εαυτός δεν θα θυμάται τη διαδικασία σκέψης που περάσατε για να δημιουργήσετε το πρόγραμμα. Φυσικά, είναι πιθανό να το παρακάνετε σχολιάζοντας:

```
a = 2; % set a equal to 2
```

Ο μελλοντικός σας εαυτός θα γνωρίζει πολύ καλά ότι το a = 2 θέτει τη μεταβλητή ίσο με δύο, επομένως αυτό το συγκεκριμένο σχόλιο μπορεί να παραληφθεί. Ωστόσο, οτιδήποτε δεν είναι τετριμμένο στα προγράμματά σας αξίζει ένα σχόλιο. Στο επάνω μέρος του παραθύρου Editor πληκτρολογήστε

```
% SinePlot.m
% makes a nice plot of a sine wave from zero to 360 degrees
```

Χρησιμοποιήστε το σύμβολο του ποσοστού για να υποδείξετε ένα σχόλιο. Τα σχόλια εμφανίζονται με πράσινη γραμματοσειρά από προεπιλογή. Όπως περιγράφουν τα σχόλια, το πρόγραμμα SinePlot θα σχεδιάσει ένα ημιτονοειδές κύμα μεταξύ 0° και 360°. Η επόμενη γραμμή που πρέπει να τοποθετήσετε σε όλα τα σενάρια MATLAB είναι η εξής:

```
clear variables; close all; clc
```

Αυτό διαγράφει όλες τις μεταβλητές που υπάρχουν στη μνήμη, ώστε να μπορείτε να ξεκινήσετε με μια "καθαρή μνήμη". Εάν ξεχάσετε να το κάνετε αυτό μπορεί να οδηγήσει σε πολύ μυστηριώδη σφάλματα, εάν, για παράδειγμα, ξεχάσετε να ορίσετε μια μεταβλητή στη σωστή τιμή της και εξακολουθεί να περιέχει τιμές από την εκτέλεση ενός προηγούμενου προγράμματος. Η εντολή `close all` κλείνει όλα τα ανοιχτά παράθυρα γραφικής παράστασης και το clc διαγράφει το παράθυρο εντολών. Το ερωτηματικό λέει στο MATLAB ότι έχετε ολοκληρώσει την εισαγωγή μιας εντολής (π.χ. διαγραφή μεταβλητών) και ότι θα πρέπει να είναι έτοιμο για την επόμενη εντολή (π.χ. `close all`). Θα υπολογίσουμε τη συνάρτηση του ημιτονοειδούς σε προσαυξήσεις 1°, απλώς για να κρατήσουμε τα πράγματα απλά

```
x = 0:360;
```

Έχουμε ορίσει το x ως διάνυσμα που ξεκινά από το 0 και μετράει προς τα πάνω (κατά ένα) έως το 360. Πόσα στοιχεία περιέχει το x; Δεδομένου ότι ξεκινά από το 0 και τελειώνει στο 360, περιέχει 361 στοιχεία! Αυτό είναι κάπως αντιφατικό, οπότε θεωρήστε ότι το πρώτο στοιχείο, x(1) = 0. Το δεύτερο στοιχείο είναι x(2) = 1. Και το τελικό στοιχείο x(361) = 360. Ένας γενικός τύπος μπορεί να γραφεί x(i) = i − 1. Σημειώστε ότι το πρώτο στοιχείο σε ένα διάνυσμα (ή πίνακα) MATLAB έχει τον δείκτη 1, όχι μηδέν όπως σε πολλές άλλες γλώσσες.

Η ημιτονοειδής συνάρτηση στο MATLAB απαιτεί να δίνουμε τη γωνία σε ακτίνια και όχι σε μοίρες. Για να μετατρέψετε από μοίρες σε ακτίνια, απλώς πολλαπλασιάστε με π/180.

```
theta = x*pi/180;
```

Αυτό πολλαπλασιάζει κάθε τιμή στο διάνυσμα x επί π/180. Σημειώστε ότι το pi (το γνωστό π) είναι μια ενσωματωμένη σταθερά στο MATLAB. Αντίστοιχα, μπορείτε επίσης να χρησιμοποιήσετε την έτοιμη εντολή του matlab:

```
theta=deg2rad(x);
```

και τα δύο κάνουν ακριβώς το ίδιο.

Τώρα που έχουμε την μεταβλητή theta σε ακτίνια, μπορούμε να υπολογίσουμε το ημίτονο της χρησιμοποιώντας την εντολή

```
y = sin(theta);
```

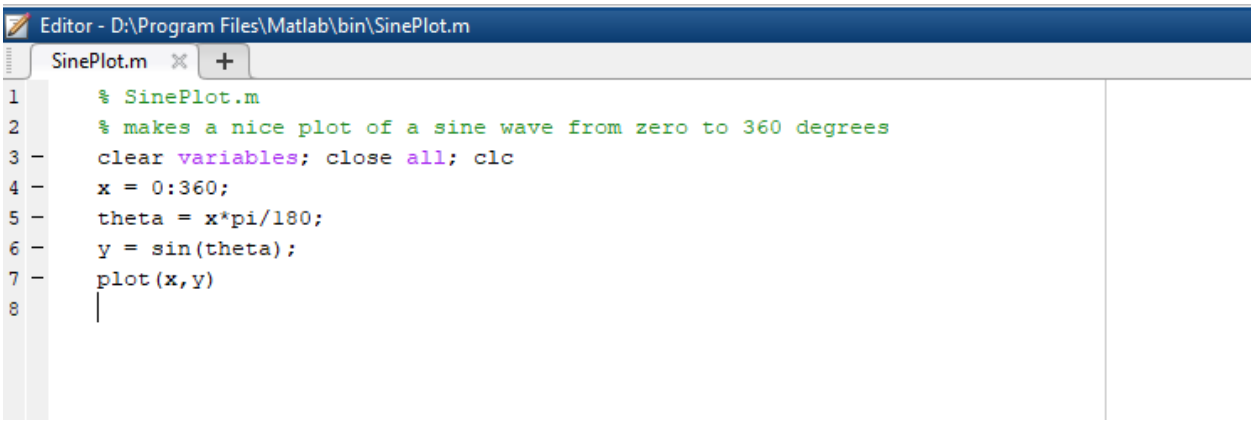
Θυμηθείτε ότι το theta είναι ένα διάνυσμα με 361 στοιχεία. Υπολογίζοντας το ημίτονο του theta προκύπτει επίσης ένα διάνυσμα 361 στοιχείων. Έτσι, y(1) = sin(0) και y(361) = sin(2π). Η γραφική παράσταση της συνάρτησης ημιτόνου προκύπτει με την απλή εντολή:

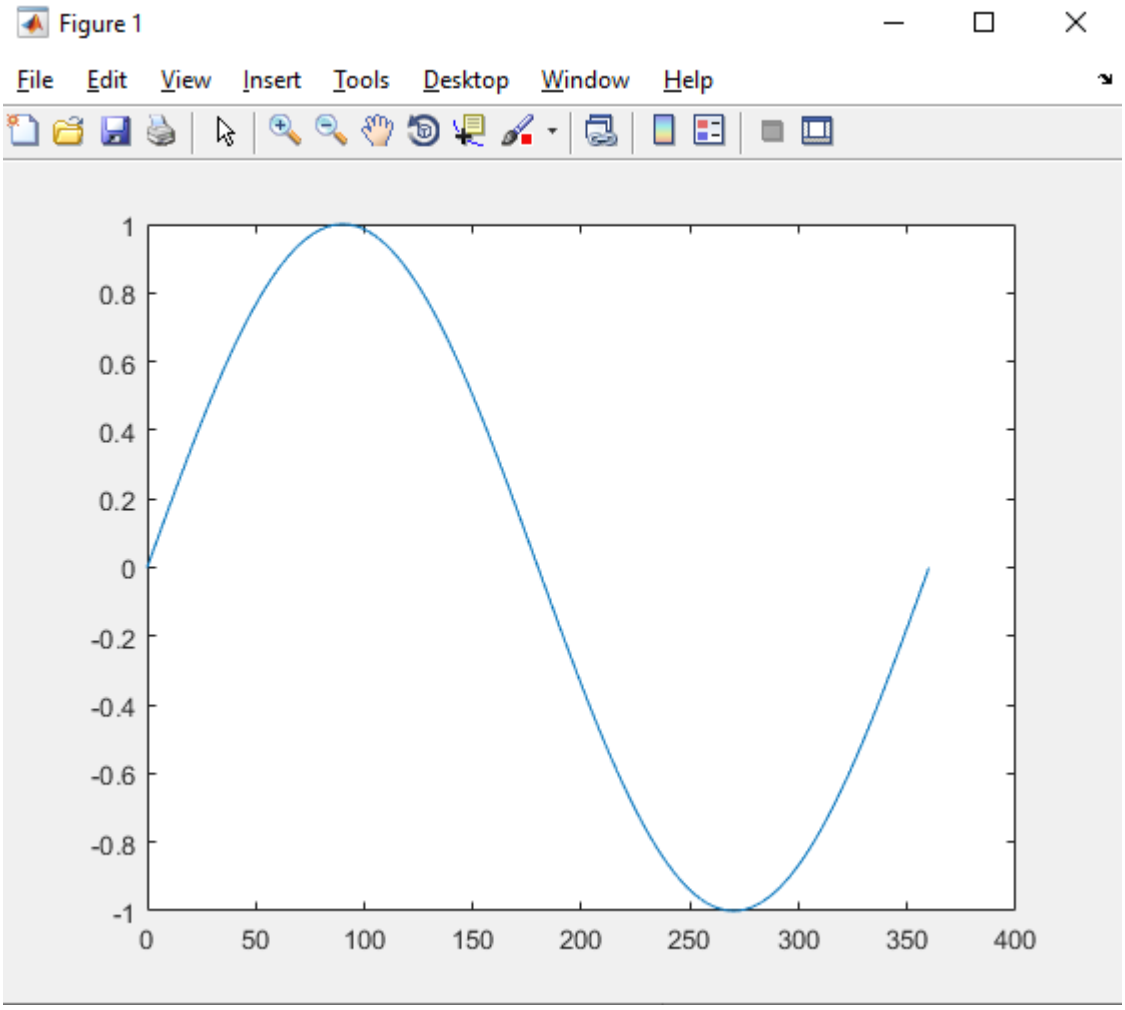
```
plot(x,y)
```

Αυτή είναι η γενική σύνταξη για την εντολή plot. Το πρώτο όρισμα είναι ένα διάνυσμα τιμών x στο διάγραμμα και το δεύτερο όρισμα είναι ένα διάνυσμα των αντίστοιχων τιμών y. Στην περίπτωση αυτή, οι τιμές x δίνουν τις γωνίες (σε μοίρες) και οι τιμές y δίνουν το ημίτονο κάθε γωνίας. Αν θέλαμε να έχουμε τη γωνία σε ακτίνια στον άξονα x θα είχαμε πληκτρολογήσει

```
plot(theta,y)
```

Ας δοκιμάσουμε το πρόγραμμα για να δούμε αν δημιουργεί το διάγραμμα που περιμένουμε. Ο απλούστερος τρόπος αποθήκευσης και εκτέλεσης του προγράμματος είναι να πατήσετε το πλήκτρο λειτουργίας F5. Εάν πατήσετε F5, το MATLAB θα αποθηκεύσει όλα τα ανοιχτά αρχεία στο παράθυρο του Editor και θα εκτελέσει το αρχείο που βρίσκεται στην κορυφή (σε αυτήν την περίπτωση, το σενάριο SinePlot.m). Εάν έχετε πληκτρολογήσει τα πάντα σωστά, θα πρέπει να λάβετε το διάγραμμα που φαίνεται στις εικόνες που ακολουθούν Το διάγραμμα είναι πράγματι μια ημιτονοειδής καμπύλη, αλλά δεν είναι πολύ ελκυστικό.





Υπάρχουν μερικά πράγματα που πρέπει να προσθέσουμε για να κάνουμε το διάγραμμα πιο επαγγελματικό. Το πρώτο πράγμα μπορεί να είναι να αλλάξετε το πλάτος της γραμμής που χρησιμοποιήθηκε για τη σχεδίαση του ημιτονοειδούς κύματος. Το προεπιλεγμένο πλάτος γραμμής είναι αρκετά στενό στο MATLAB και μια πιο παχιά γραμμή θα ήταν ευκολότερο να φανεί. Αλλάξτε την εντολή plot στην ακόλουθη

```
plot(x,y,'LineWidth',2)
```

Έχουμε ορίσει την παράμετρο LineWidth σε 2. Αυτός είναι ένας πολύ συνηθισμένος τρόπος για να τροποποιήσουμε μια εντολή σχεδίασης στο MATLAB: λέμε πρώτα στο MATLAB τι να σχεδιάσει και μετά ορίζουμε τις κατάλληλες παραμέτρους για την γραφική παράσταση. Δεδομένου ότι υπάρχουν τόσες πολλές πιθανές παράμετροι για ρύθμιση, είναι δύσκολο να τις θυμάστε όλες. Ευτυχώς, η βοήθεια είναι κοντά: πληκτρολογήστε help plot στο παράθυρο εντολών και θα λάβετε μια πλήρη λίστα παραμέτρων που μπορείτε να ορίσετε, μαζί με μερικά χρήσιμα παραδείγματα.

Τα επόμενα πράγματα που ίσως θέλουμε να προσθέσουμε στην γραφική παράσταση είναι ετικέτες για τους άξονες x και y. Κάτω από την εντολή plot προσθέστε τις ακόλουθες γραμμές:

```
xlabel('theta (degrees)'); xticks(0:60:360); xlim([0 360])
ylabel('sin(theta)')
```

Υπάρχουν αρκετά ενδιαφέροντα πράγματα που συμβαίνουν εδώ, οπότε θα τα δούμε ένα προς ένα.

Οι εντολές xlabel και ylabel προσθέτουν κείμενο στους άξονες x και y, αντίστοιχα. Το κείμενο πρέπει να περικλείεται σε ένα μόνο εισαγωγικό, διαφορετικά το MATLAB θα σκεφτεί ότι προσπαθείτε να χρησιμοποιήσετε μια μεταβλητή για την ετικέτα.

Ο άξονας x του διαγράμματος πρέπει να είναι από 0° έως 360°, αλλά η αρχική μας γραφική παράσταση πήγε από 0° σε 400°. Για να ορίσουμε τα όρια στον άξονα x, χρησιμοποιούμε την εντολή xlim, η οποία απαιτεί ένα διάνυσμα δύο στοιχείων που καθορίζει το κατώτερο και το ανώτερο όριο. Θυμηθείτε ότι τα διανύσματα περικλείονται σε αγκύλες.

Τέλος, τοποθετούμε έναν αριθμό στον άξονα x κάθε 60° χρησιμοποιώντας την εντολή xticks. Το όρισμα σε παρένθεση είναι απλώς ένα άλλο διάνυσμα που ξεκινά από το μηδέν και προχωρά σε βήματα των 60° μέχρι να φτάσει τις 360°. Τέλος, ας προσθέσουμε έναν τίτλο και ένα πλέγμα στο διάγραμμα

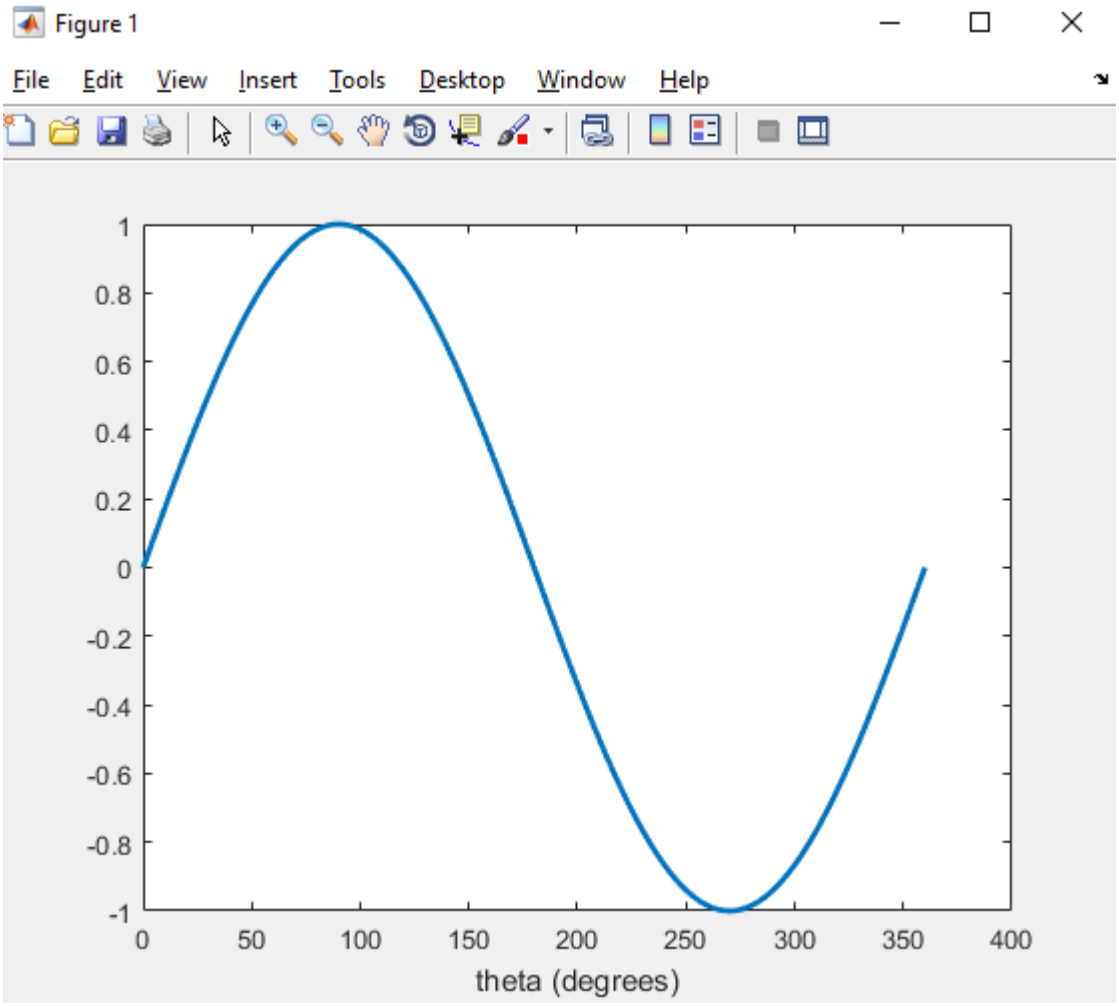
```
title('The sine function')
grid on
```

Εάν έχετε πληκτρολογήσει τα πάντα σωστά, θα πρέπει να λάβετε το διάγραμμα που φαίνεται στην παρακάτω. Συγχαρητήρια – μόλις δημιουργήσατε το πρώτο σας script στο MATLAB! Το πλήρες script φαίνεται παρακάτω.

Editor - D:\Program Files\Matlab\bin\SinePlot.m

SinePlot.m

```
1     % SinePlot.m
2     % makes a nice plot of a sine wave from zero to 360 degrees
3 -   clear variables; close all; clc
4 -   x = 0:360;
5 -   theta = x*pi/180;
6 -   y = sin(theta);
7 -   plot(x,y,'LineWidth',2)
8 -   xlabel('theta (degrees)'); xticks(0:60:360); xlim([0 360])
9 -   ylabel('sin(theta)')
10 -  title('The sine function')
11 -  grid on
```



7. Σχεδιάζοντας ένα κλειστό και γεμισμένο τετράγωνο

Το ημιτονοειδές κύμα ήταν μια μονοδιάστατη καμπύλη, οπότε ας προσπαθήσουμε να σχεδιάσουμε ένα γεμάτο 2D αντικείμενο: ένα τετράγωνο. Ανοίξτε ένα νέο σενάριο MATLAB και εισαγάγετε την ακόλουθη κεφαλίδα:

```
% SquarePlot.m
% plots a filled square
clear variables; close all; clc
```

Για να δημιουργήσουμε το τετράγωνο θα χρειαστεί να εισάγουμε τις συντεταγμένες των κορυφών. Για να γίνει αυτό, θα δημιουργήσουμε έναν πίνακα που ονομάζεται UnitSquare που έχει δύο σειρές και τέσσερις στήλες. Η πρώτη σειρά θα περιέχει όλες τις συντεταγμένες x των κορυφών και η δεύτερη σειρά θα περιέχει τις συντεταγμένες y. Θα είναι μοναδιαίο τετράγωνο γιατί οι πλευρές θα έχουν μήκος 1 και θα είναι κεντραρισμένο στην αρχή.

Αν δημιουργήσουμε έναν πίνακα συντεταγμένων για τις κορυφές, μπορεί να μοιάζει με τον Πίνακα που ακολουθεί.

Coordinates of Unit Square

	x	y
1	–0.5	–0.5
2	0.5	–0.5
3	0.5	0.5
4	–0.5	0.5

Για να το εισαγάγετε στο MATLAB, πληκτρολογήστε τις ακόλουθες γραμμές

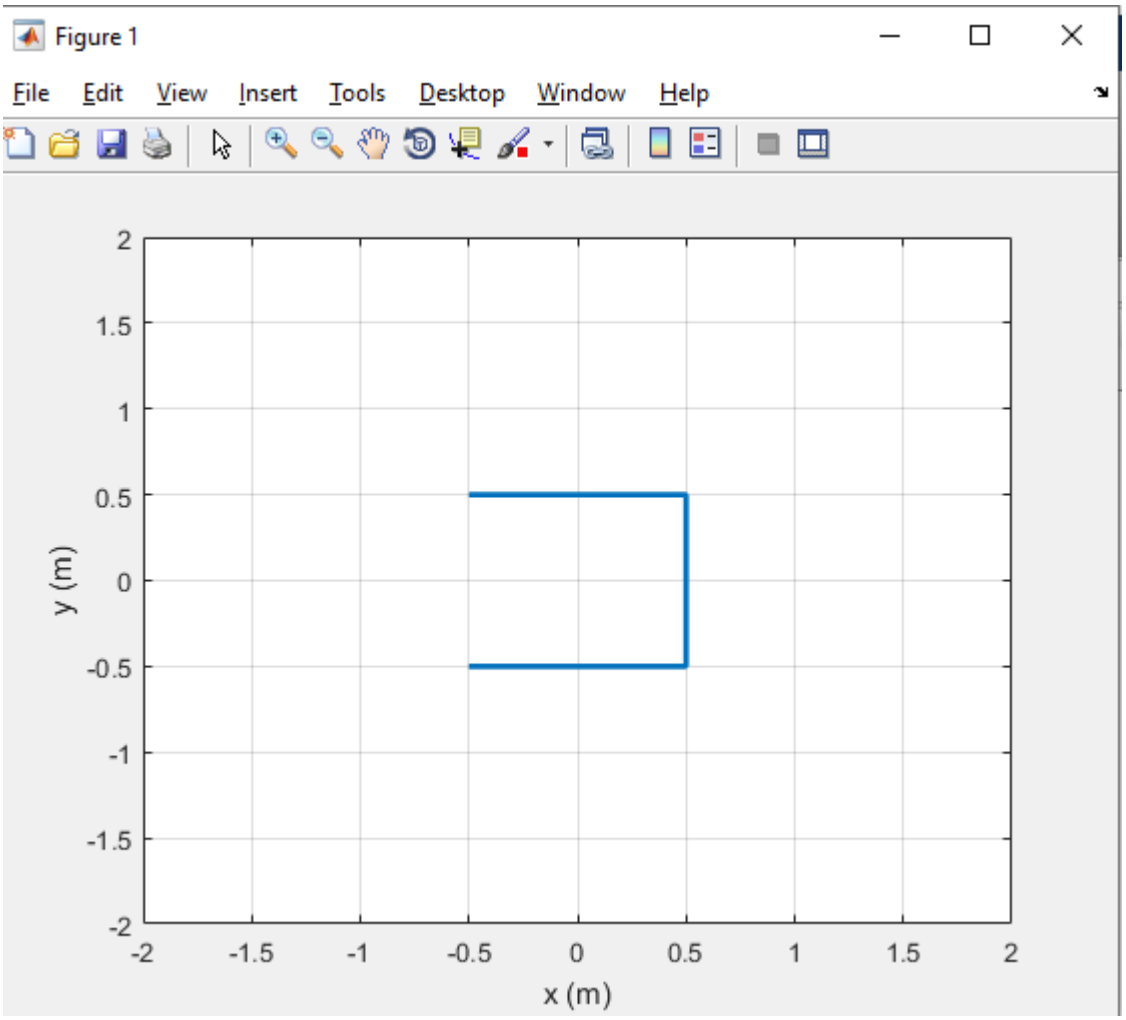
```
UnitSquare = 0.5*[-1 1 1 -1; -1 -1 1 1];
```

Το 0,5 μπροστά κλιμακώνει όλα μέσα στο πίνακα στο μισό. Η πρώτη γραμμή δίνει τις συντεταγμένες x και η δεύτερη τις συντεταγμένες y. Βεβαιωθείτε ότι έχετε εισάγει το ερωτηματικό μετά την πρώτη γραμμή, έτσι ώστε το MATLAB να γνωρίζει ότι υπάρχουν δύο σειρές σε αυτόν τον πίνακα. Για να σχεδιάσετε το τετράγωνο, εισαγάγετε τις ακόλουθες εντολές σχεδίασης

```
xlabel('x (m)'); xlim([-2 2])
ylabel('y (m)'); ylim([-2 2])
grid on
```

Η εντολή plot μπορεί να φαίνεται λίγο περίεργη στην αρχή. Θυμηθείτε ότι εισάγουμε πρώτα τις συντεταγμένες x, ακολουθούμενες από τις συντεταγμένες y. Η πρόταση UnitSquare(1,:) λέει στο MATLAB να χρησιμοποιήσει την πρώτη σειρά του πίνακα UnitSquare για τις συντεταγμένες x και το UnitSquare(2,:) δίνει τη δεύτερη σειρά για τις συντεταγμένες y. Η άνω και κάτω τελεία στο σύμβολο κράτησης θέσης στήλης σημαίνει ότι το MATLAB θα πρέπει να περιηγηθεί σε όλες τις διαθέσιμες στήλες στο UnitSquare (σε αυτήν την περίπτωση, στήλες 1–4). Τέλος, συμπεριλάβαμε την παράμετρο LineWidth για να κάνουμε τις γραμμές να ξεχωρίζουν λίγο περισσότερο από τις γραμμές πλέγματος.

Όταν αποθηκεύετε και εκτελείτε το σενάριο, θα πρέπει να λάβετε το διάγραμμα που φαίνεται στην Εικόνα που ακολουθεί.



Αυτό το διάγραμμα δεν είναι ικανοποιητικό για δύο λόγους: πρώτον, το τετράγωνο είναι ανοιχτό στην αριστερή πλευρά και δεύτερον, φαίνεται να είναι στριμωγμένο στην κατεύθυνση y. Η εντολή plot σχεδιάζει γραμμές μεταξύ των σημείων που καθορίζουμε και δεν προσδιορίσαμε ότι το τελικό σημείο στο τετράγωνο πρέπει να είναι το ίδιο με το σημείο εκκίνησης. Προσθέστε μια ακόμη στήλη στον πίνακα UnitSquare ως εξής:

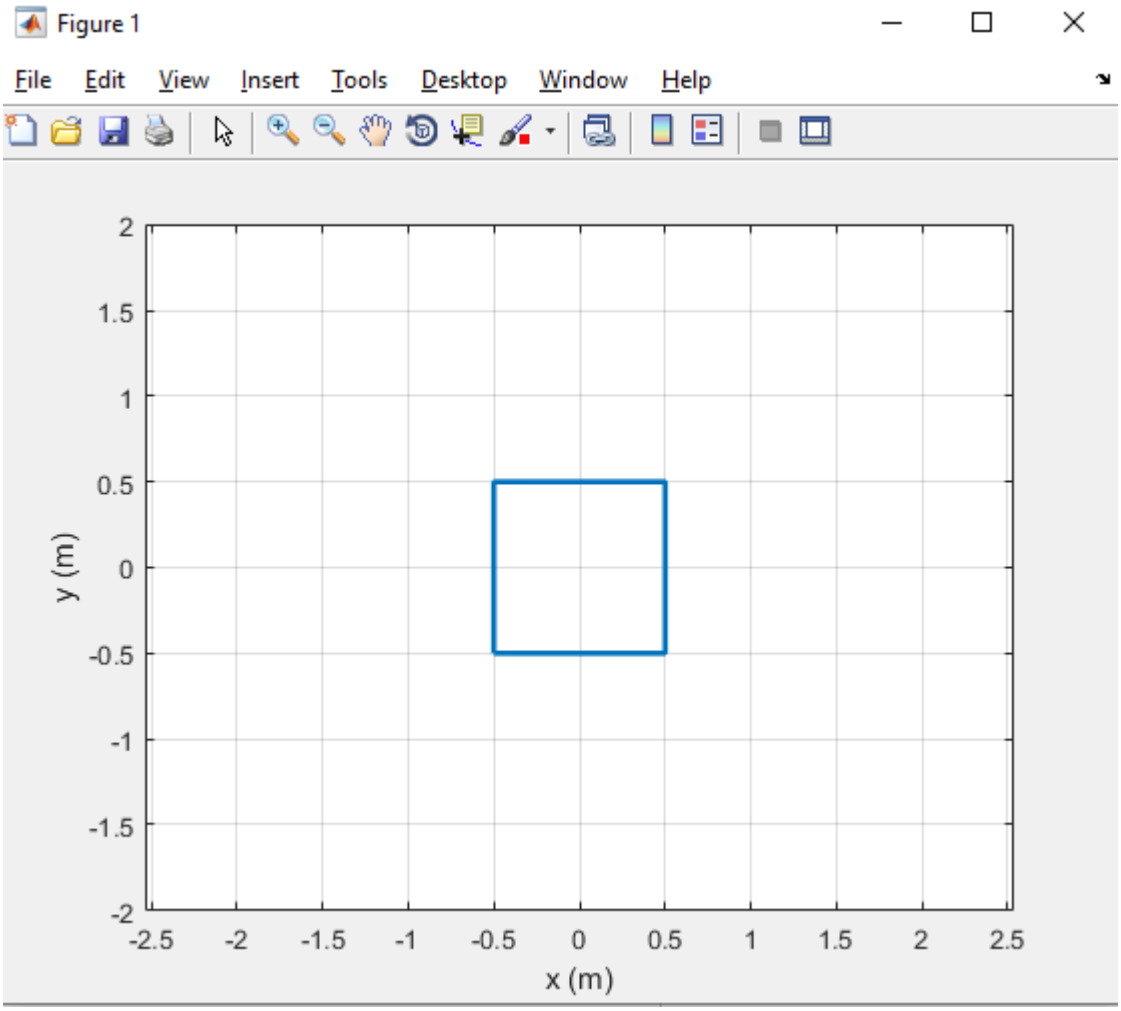
```
UnitSquare = 0.5*[-1 1 1 -1 -1; -1 -1 1 1 -1];
```

Σημειώστε ότι η τελευταία στήλη είναι ίδια με την πρώτη στήλη. Όταν εκτελείτε ξανά το πρόγραμμα, θα πρέπει να λάβετε ένα κλειστό ορθογώνιο (αλλά όχι ένα τετράγωνο!) Η σύνθλιψη συμβαίνει επειδή το MATLAB προσπαθεί να γεμίσει όσο το δυνατόν μεγαλύτερο μέρος του παραθύρου της γραφικής παράστασης με το ζητούμενο σχέδιο. Δεδομένου ότι ο άξονας x είναι πολύ ευρύτερος από τον άξονα y, το διάγραμμα καταλήγει να τεντώνεται προς την κατεύθυνση x. Για να το διορθώσετε, προσθέστε την εντολή

```
axis equal
```

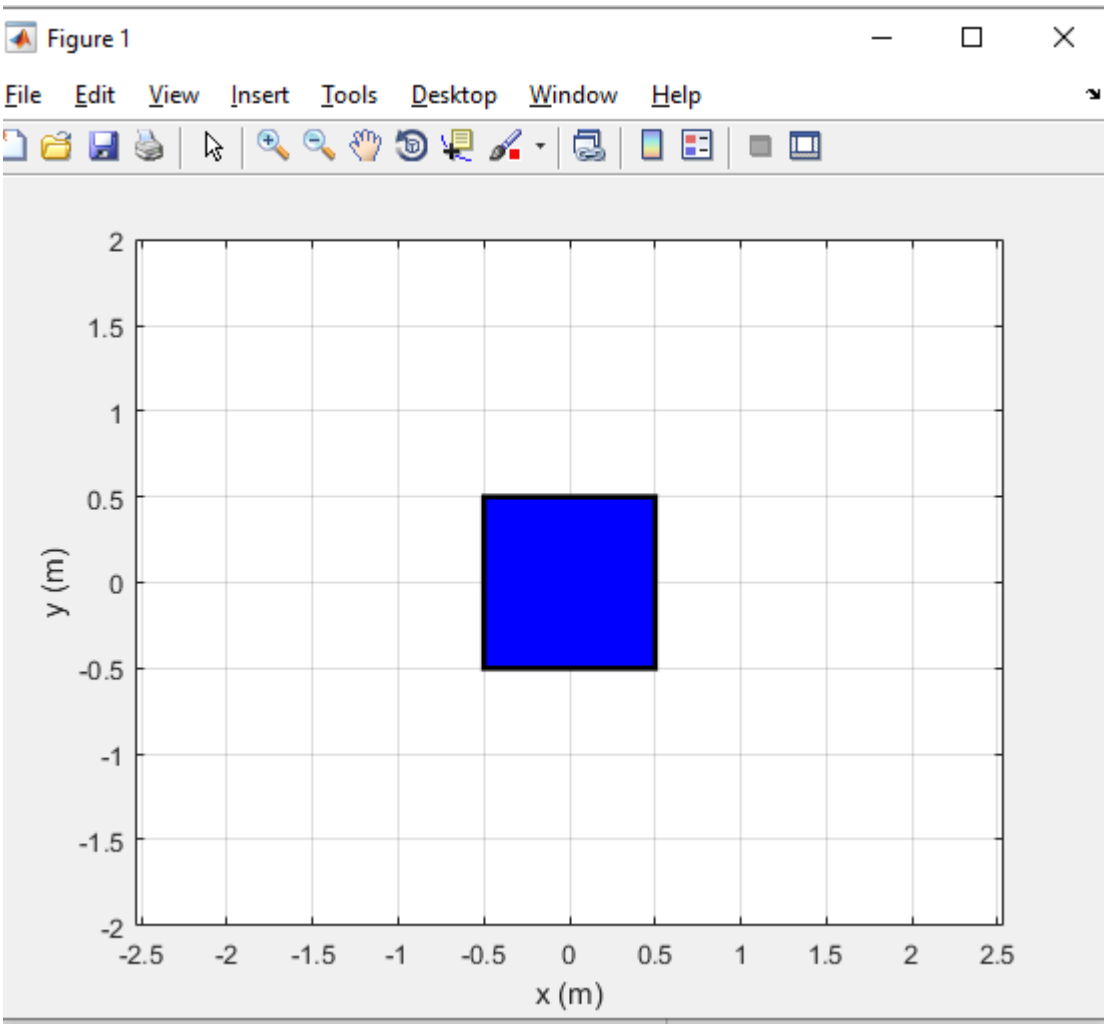
στο τέλος του script.

Θα πρέπει τώρα να αποκτήσετε το διάγραμμα που φαίνεται στην Εικόνα που ακολουθεί.



Μοιάζει με τετράγωνο, αλλά δεν συμπληρώνεται όπως απαιτείται από τη δήλωση προβλήματος. Για να φτιάξουμε ένα γεμάτο πολύγωνο μπορούμε να χρησιμοποιήσουμε την εντολή fill, η οποία λειτουργεί σχεδόν με τον ίδιο τρόπο όπως η εντολή plot. Σχολιάστε την εντολή plot (βάζοντας μπροστά από την εντολή το σύμβολο %) και πληκτρολογήστε την ακόλουθη εντολή γέμισης. Θα πρέπει να λάβετε το διάγραμμα που φαίνεται στην Εικόνα που ακολουθεί.

```
%plot(UnitSquare(1,:),UnitSquare(2,:),'LineWidth',2)
fill(UnitSquare(1,:),UnitSquare(2,:),'b','LineWidth',2)
```



Η εντολή fill απαιτεί να καθορίσετε ένα χρώμα (σε αυτήν την περίπτωση το «b» σημαίνει μπλε) αμέσως αφού δώσετε τις συντεταγμένες x και y. Έχουμε επιτύχει τώρα τον στόχο να σχεδιάσουμε ένα γεμάτο τετράγωνο. Τι θα γινόταν αν θέλαμε να σχεδιάσουμε ένα τετράγωνο με μήκος πλευράς 2 με κέντρο τις συντεταγμένες (1, 1); Μπορούμε απλά να ορίσουμε έναν νέο πίνακα, που ονομάζεται NewSquare, ως εξής:

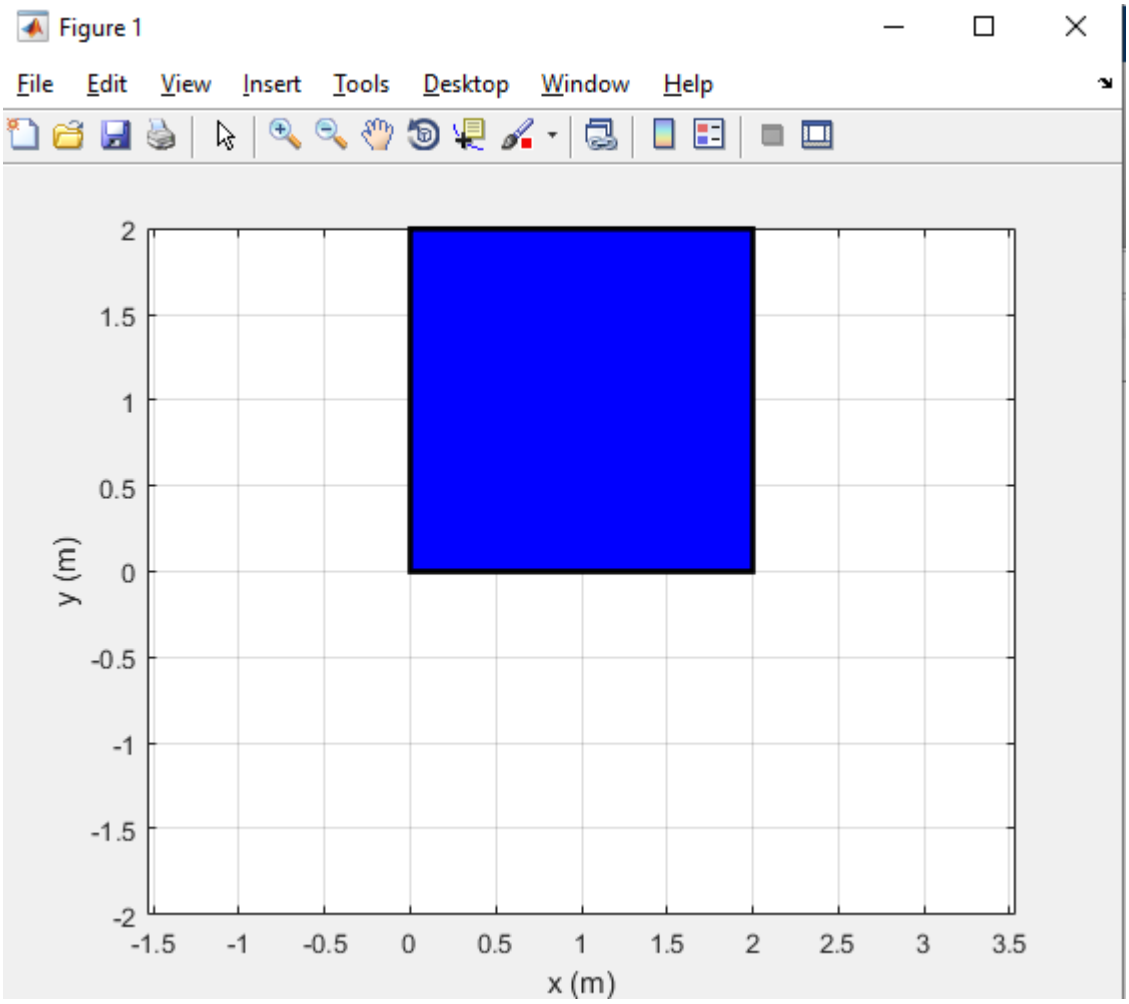
```
NewSquare = 2*UnitSquare + [1 1 1 1 1; 1 1 1 1 1];
```

Αυτή η εντολή πολλαπλασιάζει κάθε καταχώρηση στον πίνακα UnitSquare επί 2 (δηλαδή, κλιμακώνει τον πίνακα κατά συντελεστή 2) και προσθέτει (1, 1) σε κάθε κορυφή. Εάν αλλάξετε την εντολή γέμισης σε σχεδίαση NewSquare αντί για UnitSquare, θα πρέπει να λάβετε την γραφική παράσταση που φαίνεται στην Εικόνα που ακολουθεί, ενώ το script στην τελική του μορφή είναι το ακόλουθο:

Editor - D:\Program Files\Matlab\bin\SquarePlot.m*

SinePlot.m | SquarePlot.m*

```
1     % SquarePlot.m
2     % plots a filled square
3 -   clear variables; close all; clc
4 -   UnitSquare = 0.5*[-1 1 1 -1 -1; -1 -1 1 1 -1];
5 -   NewSquare = 2*UnitSquare + [1; 1];
6
7     %plot(UnitSquare(1,:),UnitSquare(2,:),'LineWidth',2)
8 -   fill(NewSquare(1,:),NewSquare(2,:),'b','LineWidth',2)
9 -   xlabel('x (m)'); xlim([-2 2])
10 -  ylabel('y (m)'); ylim([-2 2])
11 -  grid on
12 -  axis equal
```



8. Προσθέτοντας μια δομή: Το for Loop

Μέχρι στιγμής, χρησιμοποιήσαμε διανυσματικές πράξεις για να εκτελέσουμε επαναλαμβανόμενους υπολογισμούς. Αυτό λειτουργεί καλά για απλούς τύπους, όπως το ημίτονο μιας γωνίας, αλλά θα είναι εξαιρετικά δύσκολο για πιο περίπλοκους υπολογισμούς, όπως η ανάλυση θέσης και δύναμης. Ευτυχώς, υπάρχει μια απλή δομή που μπορούμε να χρησιμοποιήσουμε για να εκτελέσουμε επαναλαμβανόμενους υπολογισμούς: ο βρόχος for. Η σύνταξη αυτού του βρόχου είναι

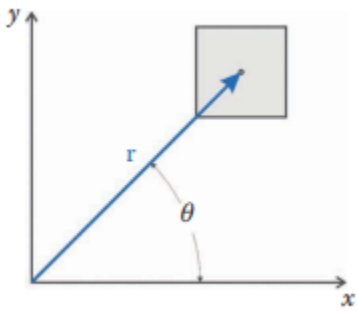
for i = 1:N

    do some stuff N times

end

Η μεταβλητή i ονομάζεται μετρητής βρόχου και αυτό το σύνολο εντολών λέει στο MATLAB να εκτελέσει οτιδήποτε βρίσκεται μέσα στη δομή for…end συνολικά N φορές. Την πρώτη φορά που εκτελεί τους υπολογισμούς η μεταβλητή i παίρνει την τιμή του 1, τη δεύτερη φορά την τιμή του 2 και ούτω καθεξής μέχρι να φτάσει σε μια τιμή N. Μετά την εκτέλεση των υπολογισμών N φορές ο βρόχος ολοκληρώθηκε και το MATLAB συνεχίζει στην επόμενη δήλωση. Σημειώστε ότι κάθε δήλωση for πρέπει να έχει μια αντίστοιχη δήλωση end.

Ας προσπαθήσουμε να χρησιμοποιήσουμε έναν βρόχο for για να δημιουργήσουμε έναν «κύκλο τετραγώνων» που περιστρέφεται γύρω από την αρχή. Θα χρησιμοποιήσουμε τον πίνακα UnitSquare που δημιουργήσαμε στο τελευταίο σενάριο για να σχεδιάσουμε ένα τετράγωνο, επαναλαμβάνοντας τη διαδικασία καθώς κινούμαστε γύρω από την αρχή. Το κέντρο ενός δεδομένου τετραγώνου βρίσκεται στο τέλος του διανύσματος r, που φαίνεται στο σχήμα, όπου



$$\mathbf{r} = 2\begin{Bmatrix} \cos\theta \\ \sin\theta \end{Bmatrix}$$

Δημιουργήστε ένα νέο script που ονομάζεται SquareCircle.m. Στην κορυφή του αρχείου πληκτρολογήστε την κεφαλίδα:

```
% SquareCircle.m
```

```
% makes a plot of eight squares arranged in a circle
clear variables; close all; clc
UnitSquare = 0.5*[-1 1 1 -1 -1; -1 -1 1 1 -1];
```

Θέλουμε να σχεδιάσουμε οκτώ τετράγωνα, οπότε πληκτρολογήστε τον παρακάτω βρόχο for

```
for i = 1:8

end
```

Το πρώτο πράγμα που πρέπει να υπολογίσουμε μέσα στον βρόχο είναι η γωνία του διανύσματος r, την οποία συμβολίζουμε με theta. Θέλουμε το theta να ξεκινά από το μηδέν και να αυξάνεται κατά 45° σε κάθε επανάληψη του βρόχου. Εφόσον το i ξεκινάει από το 1, πρέπει να αφαιρέσουμε το 1 εάν θέλουμε να ξεκινήσουμε από το 0. Η γωνία 45° είναι ίση με 2π/8 σε ακτίνια. Έτσι, η πρώτη γραμμή μέσα στο βρόχο for θα πρέπει να είναι:

```
for i = 1:8
    theta = (i-1)*2*pi/8; % angle at which to place square
end
```

Είναι σύνηθες να εισάγετε εσοχές σε όλες τις γραμμές μέσα στο βρόχο for έτσι ώστε να είναι εύκολο να δείτε ποιες εντολές εκτελούνται πολλές φορές. Ο αναγνώστης μπορεί εύκολα να επαληθεύσει ότι το θήτα αρχίζει στο 0 (όταν i = 1) και τελειώνει στο (7/8) 2π = 315° όταν i = 8. Στη συνέχεια, υπολογίζουμε το διάνυσμα r, το οποίο δίνει το κέντρο ενός δεδομένου τετραγώνου

```
for i = 1:8
    theta = (i-1)*2*pi/8; % angle at which to place square
    r = 2*[cos(theta) cos(theta) cos(theta) cos(theta) cos(theta); sin(theta)
sin(theta) sin(theta) sin(theta) sin(theta)]; % center of square
end
```

Το διάνυσμα r έχει μήκος 2. Στη συνέχεια, θα δημιουργήσουμε ένα νέο τετράγωνο με κέντρο στο r

```
for i = 1:8
    theta = (i-1)*2*pi/8; % angle at which to place square
    r = 2*[cos(theta) cos(theta) cos(theta) cos(theta) cos(theta); sin(theta)
sin(theta) sin(theta) sin(theta) sin(theta)]; % center of square
    NewSquare = UnitSquare + r; % create new square
end
```

Δεδομένου ότι θα δημιουργήσουμε γεμάτα τετράγωνα, θα ήταν ωραίο να μπορούσαμε να γεμίσουμε κάθε τετράγωνο με διαφορετικό χρώμα. Το MATLAB καθορίζει τα χρώματα ως διάνυσμα τριών στοιχείων δίνοντας τα κόκκινα, πράσινα και μπλε στοιχεία (R, G, and B), αντίστοιχα. Οι τιμές των R, G και B πρέπει να βρίσκονται μεταξύ μηδέν και ένα με το ένα να δίνει πλήρη κορεσμό ενός δεδομένου χρώματος. Έτσι, το τριπλό [1 1 1] καθορίζει το λευκό χρώμα, το [1 0 0] δίνει ένα κόκκινο χρώμα, το [0 0 0] δίνει μαύρο και το [0,5 0,5 0,5] δίνει το γκρι χρώμα

στα μισά του δρόμου μεταξύ λευκού και μαύρου. Θα σχεδιάσουμε τα τετράγωνά μας σε κλίμακα του γκρι ξεκινώντας με μαύρο για το πρώτο τετράγωνο και τελειώνοντας σε ανοιχτό γκρι για το τελευταίο τετράγωνο. Επειδή το χρώμα πρέπει να είναι διαφορετικό για κάθε τετράγωνο, πρέπει να χρησιμοποιήσουμε τον μετρητή βρόχου i για να ορίσουμε τα χρώματα. Στην επόμενη γραμμή του βρόχου, ορίστε τη μεταβλητή col (συντομογραφία για το χρώμα).

```
for i = 1:8
    theta = (i-1)*2*pi/8; % angle at which to place square
    r = 2*[cos(theta) cos(theta) cos(theta) cos(theta) cos(theta); sin(theta)
sin(theta) sin(theta) sin(theta) sin(theta)]; % center of square
    NewSquare = UnitSquare + r; % create new square
     col = (i-1)*[1 1 1]/8; % fill and edge color of square
end
```

Σημειώστε το διάνυσμα τριών στοιχείων [1 1 1] στο κέντρο του ορισμού. Αυτό πολλαπλασιάζεται με (i-1)/8, το οποίο ξεκινά στο 0 όταν i = 1 και τελειώνει στο 7/8 όταν i = 8. Έτσι, το πρώτο χρώμα ορίζεται ως [0 0 0] (μαύρο) και το Το τελικό χρώμα ορίζεται ως [0,875 0,875 0,875] (πραγματικά ανοιχτό γκρι). Τώρα είμαστε έτοιμοι να σχεδιάσουμε τα τετράγωνα

```
for i = 1:8
    theta = (i-1)*2*pi/8; % angle at which to place square
    r = 2*[cos(theta) cos(theta) cos(theta) cos(theta) cos(theta); sin(theta)
sin(theta) sin(theta) sin(theta) sin(theta)]; % center of square
    NewSquare = UnitSquare + r; % create new square
     col = (i-1)*[1 1 1]/8; % fill and edge color of square
     fill(NewSquare(1,:),NewSquare(2,:),col,'EdgeColor',col,'LineWidth',2)
end
```

Εάν αποθηκεύσετε και εκτελέσετε το πρόγραμμα τώρα, θα διαπιστώσετε ότι φαίνεται να έχει σχεδιάσει μόνο το τελικό τετράγωνο στην ακολουθία. Αυτό συμβαίνει επειδή η εντολή γέμισμα διαγράφει αυτόματα οτιδήποτε υπήρχε στο παράθυρο γραφικής παράστασης πριν σχεδιάσει το νέο τετράγωνο. Αμέσως μετά την εντολή γέμισμα, πληκτρολογήστε την εντολή

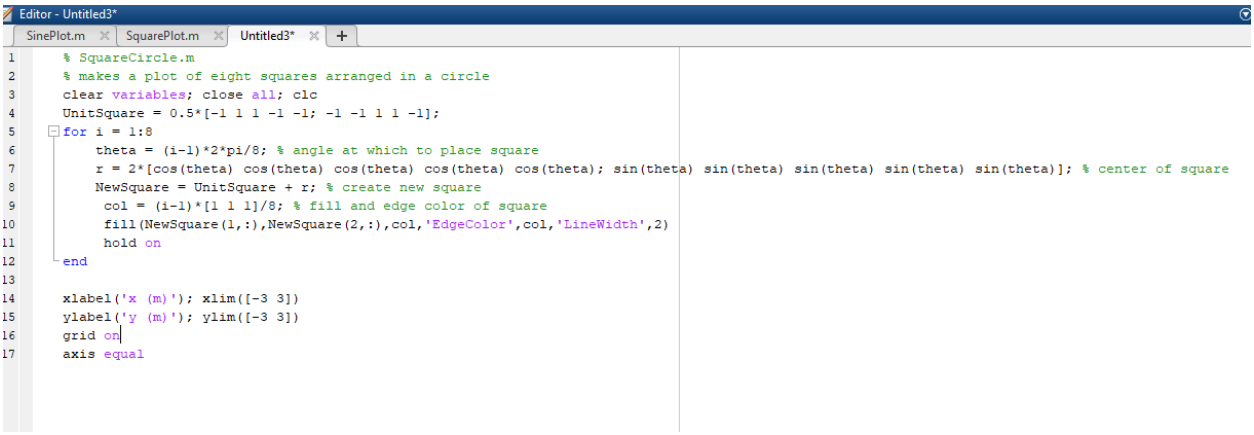
```
for i = 1:8
    theta = (i-1)*2*pi/8; % angle at which to place square
    r = 2*[cos(theta) cos(theta) cos(theta) cos(theta) cos(theta); sin(theta)
sin(theta) sin(theta) sin(theta) sin(theta)]; % center of square
    NewSquare = UnitSquare + r; % create new square
     col = (i-1)*[1 1 1]/8; % fill and edge color of square
     fill(NewSquare(1,:),NewSquare(2,:),col,'EdgeColor',col,'LineWidth',2)
     hold on
end
```

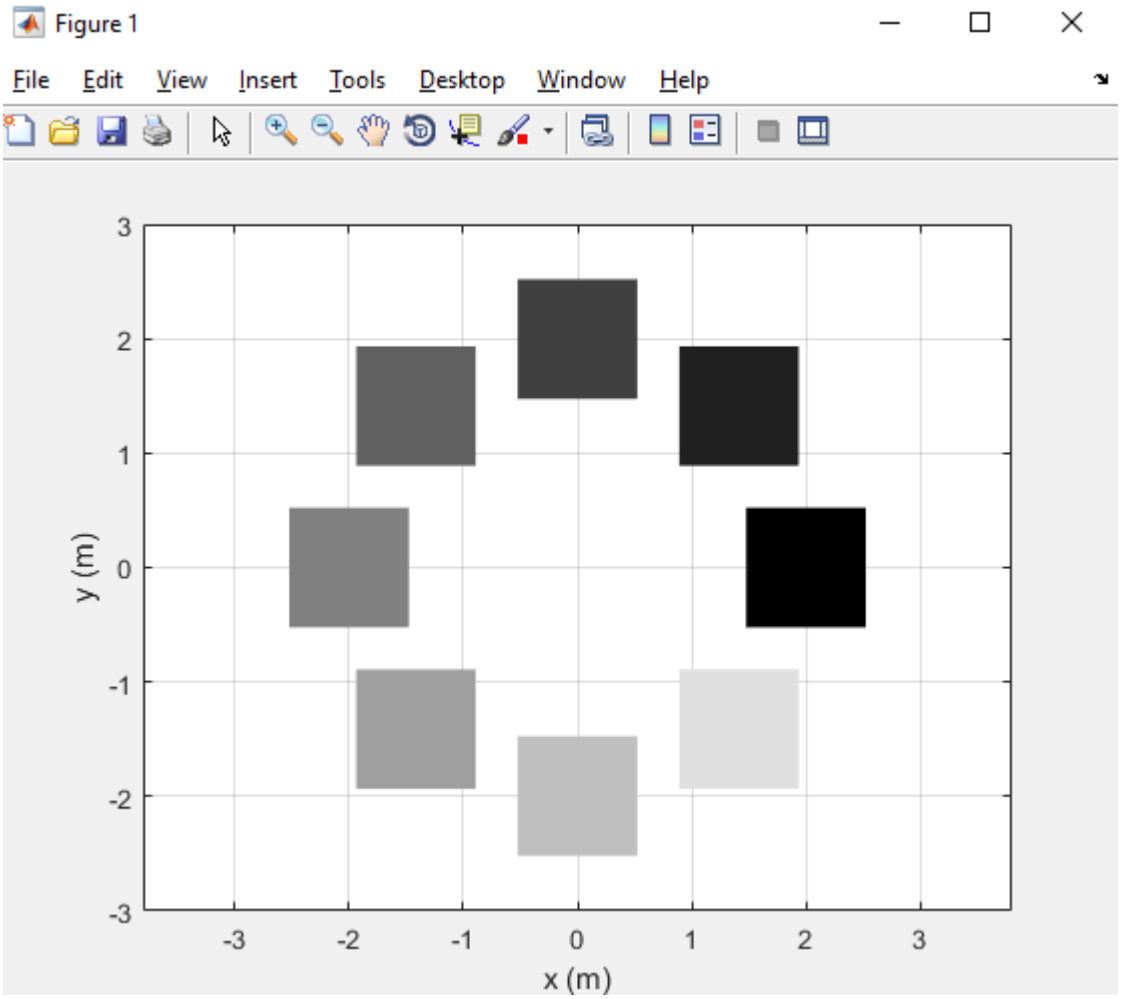
Αυτό λέει στο MATLAB να διατηρήσει τα πάντα στο παράθυρο γραφικής παράστασης, έτσι ώστε η επόμενη εντολή σχεδίασης απλώς να προσθέσει ένα νέο τετράγωνο στην υπάρχουσα γραφική παράσταση. Αφού τελειώσει ο βρόχος, θα πρέπει να προσθέσουμε τις εντολές που κάνουν την γραφική παράσταση να φαίνεται επαγγελματική.

```
xlabel('x (m)'); xlim([-3 3])
```

```
ylabel('y (m)'); ylim([-3 3])
grid on
axis equal
```

Εάν αποθηκεύσετε και εκτελέσετε τη δέσμη ενεργειών (χρησιμοποιώντας το F5) θα λάβετε το διάγραμμα που φαίνεται στην Εικόνα που ακολουθεί. Η πλήρης λίστα του script φαίνεται παρακάτω.





9. Ένα απλοϊκό παράδειγμα Animation

Μερικές φορές ο καλύτερος τρόπος για να οπτικοποιήσετε την κίνηση ενός μηχανισμού είναι να τον ζωντανέψετε. Το MATLAB διαθέτει μια πλήρη σειρά δυνατοτήτων για τη δημιουργία αρχείων ταινιών, αλλά εδώ θα επικεντρωθούμε στην κίνηση μιας γραφικής παράστασης μέσα σε ένα κανονικό παράθυρο διαγράμματος. Για αυτό το παράδειγμα, θα ζωντανέψουμε την κίνηση ενός βλήματος που εκτοξεύεται στον αέρα υπό γωνία 45°. Το βλήμα έχει αρχική ταχύτητα και στις δύο κατευθύνσεις x και y 50m/s. Ανοίξτε ένα νέο σενάριο που ονομάζεται Projectile.m και πληκτρολογήστε την ακόλουθη κεφαλίδα:

```
% Projectile.m
% animates the path of a projectile
clear variables; close all; clc
t = 0:0.1:15; % [s] vector of times
vx0 = 50; % [m/s] initial velocity in the x direction
vy0 = 50; % [m/s] initial velocity in the y direction
g = 9.81; % [m/s^2] acceleration of gravity
```

Στη συνέχεια θα πρέπει να πούμε στο MATLAB να ανοίξει ένα νέο παράθυρο χωρίς να έχει τοποθετήσει τίποτα ακόμα σε αυτό.

```
figure
```

Το προεπιλεγμένο μέγεθος παραθύρου γραφικής παράστασης MATLAB είναι λίγο μικρό για τους σκοπούς μας, επομένως θα το αλλάξουμε χρησιμοποιώντας την ακόλουθη εντολή

```
set(gcf,'Position',[50 50 1200 500])
```

Το όρισμα gcf αντιπροσωπεύει το "graphics current figure" και η εντολή ορίζει το τρέχον παράθυρο σχεδίασης σε πλάτος 1200 pixel και ύψος 500 pixel. Εντοπίζει την κάτω γωνία του παραθύρου της πλοκής σε απόσταση 50 pixel από το κάτω μέρος της οθόνης και 50 pixel από την αριστερή πλευρά της οθόνης. Τώρα ας ξεκινήσουμε τον βρόχο for. Ο βρόχος πρέπει να κάνει μία επανάληψη για κάθε στοιχείο στο διάνυσμα t (time), οπότε ορίζουμε τον βρόχο ως

```
for i = 1:length(t)

end
```

Η δήλωση μήκους δίνει τον αριθμό των στοιχείων στο διάνυσμα t. Στην αρχή του βρόχου, θα πρέπει να υπολογίσουμε την τρέχουσα θέση του βλήματος. Αυτό το κάνουμε χρησιμοποιώντας μερικές βασικές εξισώσεις από την εισαγωγική Φυσική:

$$x = v_{x0}t$$

$$y = v_{y0}t - \frac{1}{2}gt^2$$

Η μετάφραση αυτών των εξισώσεων σε MATLAB δίνει

```
for i = 1:length(t)
    x = vx0 * t(i); % x position
    y = vy0 * t(i) - 0.5*g*t(i)^2; % y position
end
```

Θυμηθείτε ότι το t είναι ένα διάνυσμα, όχι ένας αριθμός. Η πρόταση t(i) δίνει τον i^o αριθμό στο διάνυσμα t. Εφόσον έχουμε ορίσει τον βρόχο έτσι ώστε το i να κυμαίνεται από 1 έως length (t), θα χρησιμοποιήσουμε κάθε τιμή του t καθώς επαναλαμβάνουμε τον βρόχο. Η πρώτη επανάληψη του βρόχου χρησιμοποιεί την πρώτη τιμή μέσα στο διάνυσμα t, η δεύτερη επανάληψη χρησιμοποιεί τη δεύτερη τιμή στο t και ούτω καθεξής.

Στη συνέχεια σχεδιάζουμε την τρέχουσα θέση του βλήματος

```
for i = 1:length(t)
    x = vx0 * t(i); % x position
    y = vy0 * t(i) - 0.5*g*t(i)^2; % y position
    plot(x,y,'o','MarkerFaceColor','b')
```

```
end
```

Σε αντίθεση με τις προηγούμενες γραφικές παραστάσεις μας, οι ποσότητες x και y περιέχουν έναν μοναδικό αριθμό η καθεμία – τις συντεταγμένες της τρέχουσας θέσης του βλήματος. Σε ένα δεδομένο χρονικό βήμα, σχεδιάζουμε τη θέση ενός μόνο σημείου, την τρέχουσα θέση του βλήματος. Για το λόγο αυτό, δεν υπάρχει γραμμή καμπύλη για να σχεδιάσουμε, αφού δεν έχουμε περισσότερα από ένα σημεία. Η δήλωση 'o' λέει στο MATLAB να σχεδιάσει το τρέχον σημείο με έναν μικρό κυκλικό δείκτη και η εντολή «MarkerFaceColor» κάνει τον κυκλικό δείκτη σε σταθερό μπλε χρώμα. Στη συνέχεια, πληκτρολογήστε τις εντολές που κάνουν το διάγραμμα εμφανίσημο:

```
for i = 1:length(t)
    x = vx0 * t(i); % x position
    y = vy0 * t(i) - 0.5*g*t(i)^2; % y position
    plot(x,y,'o','MarkerFaceColor','b')
    axis equal
    xlabel('x (m)'); xlim([0 600])
    ylabel('y (m)'); ylim([0 150])
    grid on
end
```

Τα όρια του άξονα βρέθηκαν μέσω δοκιμής και λάθους και θα χρειαστεί να τα αλλάξετε εάν το βλήμα σας έχει διαφορετική αρχική ταχύτητα. Η τελική εντολή στον βρόχο είναι

```
for i = 1:length(t)
    x = vx0 * t(i); % x position
    y = vy0 * t(i) - 0.5*g*t(i)^2; % y position
    plot(x,y,'o','MarkerFaceColor','b')
    axis equal
    xlabel('x (m)'); xlim([0 600])
    ylabel('y (m)'); ylim([0 150])
    grid on
    drawnow
end
```

που λέει στο MATLAB να τοποθετήσει όλα όσα βρίσκονται στην προσωρινή μνήμη γραφικών στο παράθυρο γραφικής παράστασης. Αυτό ολοκληρώνει τον βρόχο for, και μπορείτε να αποθηκεύσετε και να εκτελέσετε το πρόγραμμα τώρα. Όταν εκτελείτε το πρόγραμμα, θα πρέπει να ανταμειφθείτε με ένα όμορφο κινούμενο σχέδιο του βλήματος να πετάει στον αέρα και μετά να προσγειώνεται (ή να πέφτει κάτω από τον άξονα x). Η πλήρης λίστα του προγράμματος φαίνεται παρακάτω

Editor - Untitled4*

SinePlot.m | SquarePlot.m | SquareCircle.m | Untitled4*

```
% Projectile.m
% animates the path of a projectile
clear variables; close all; clc
t = 0:0.1:15; % [s] vector of times
vx0 = 50; % [m/s] initial velocity in the x direction
vy0 = 50; % [m/s] initial velocity in the y direction
g = 9.81; % [m/s^2] acceleration of gravity
figure
set(gcf,'Position',[50 50 1200 500])
for i = 1:length(t)
    x = vx0 * t(i); % x position
    y = vy0 * t(i) - 0.5*g*t(i)^2; % y position
    plot(x,y,'o','MarkerFaceColor','b')
    axis equal
    xlabel('x (m)'); xlim([0 600])
    ylabel('y (m)'); ylim([0 150])
    grid on
    drawnow
end
```