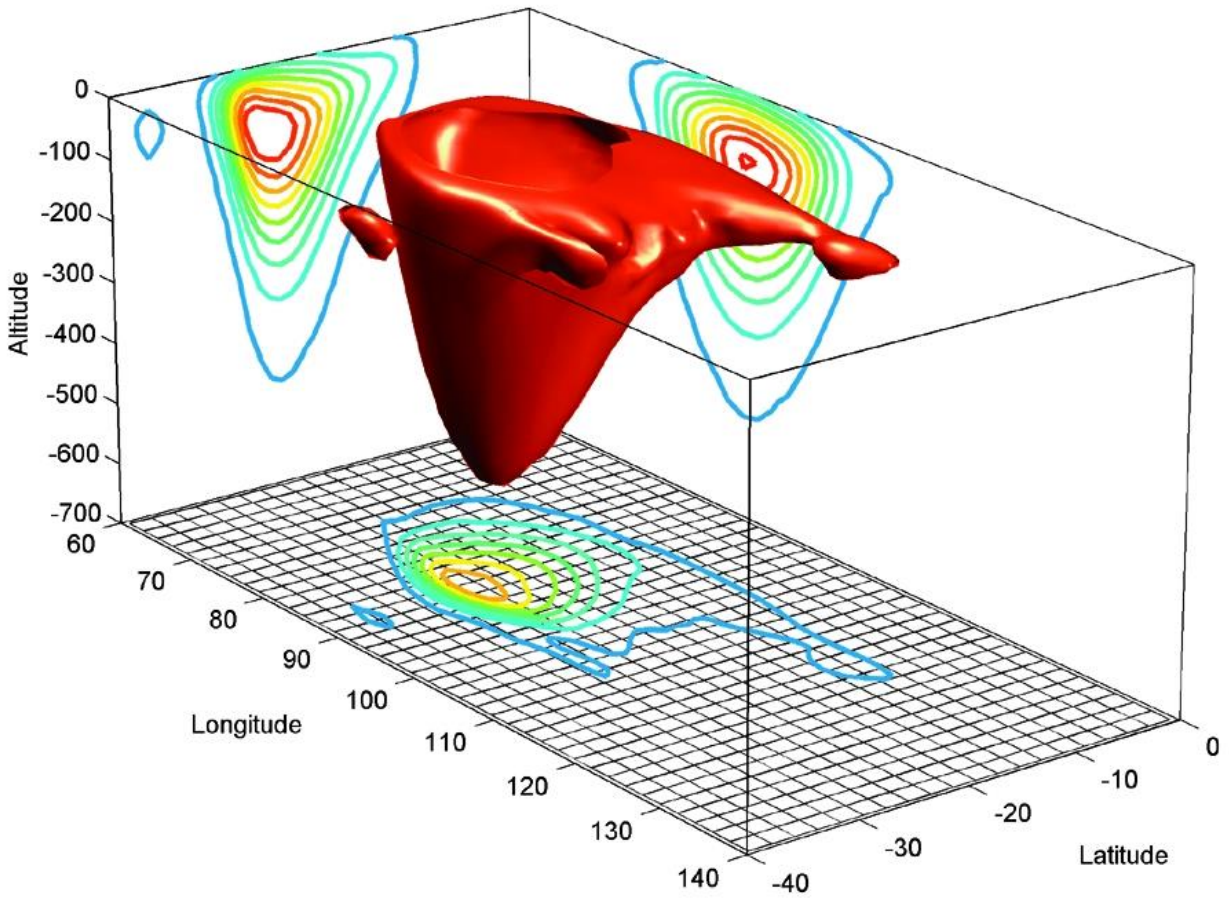
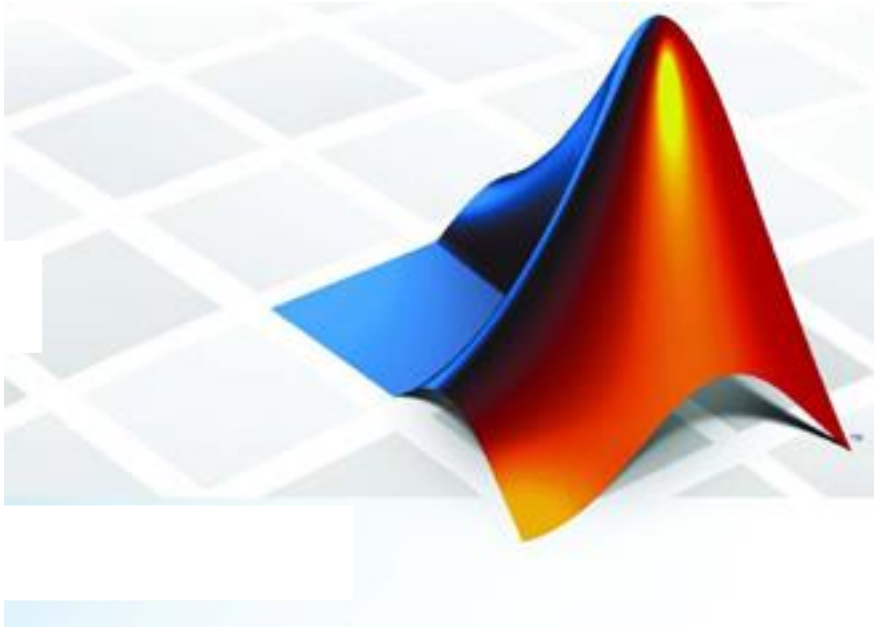
Εισαγωγή στη MATLAB

Μέρος 1

Πετσαγκουράκης Παναγιώτης





Εκδόσεις

- Βγαίνουν 2 νέες κάθε χρόνο…είναι στο R2015a
  - Πρόσβαση από το ΕΜΠ

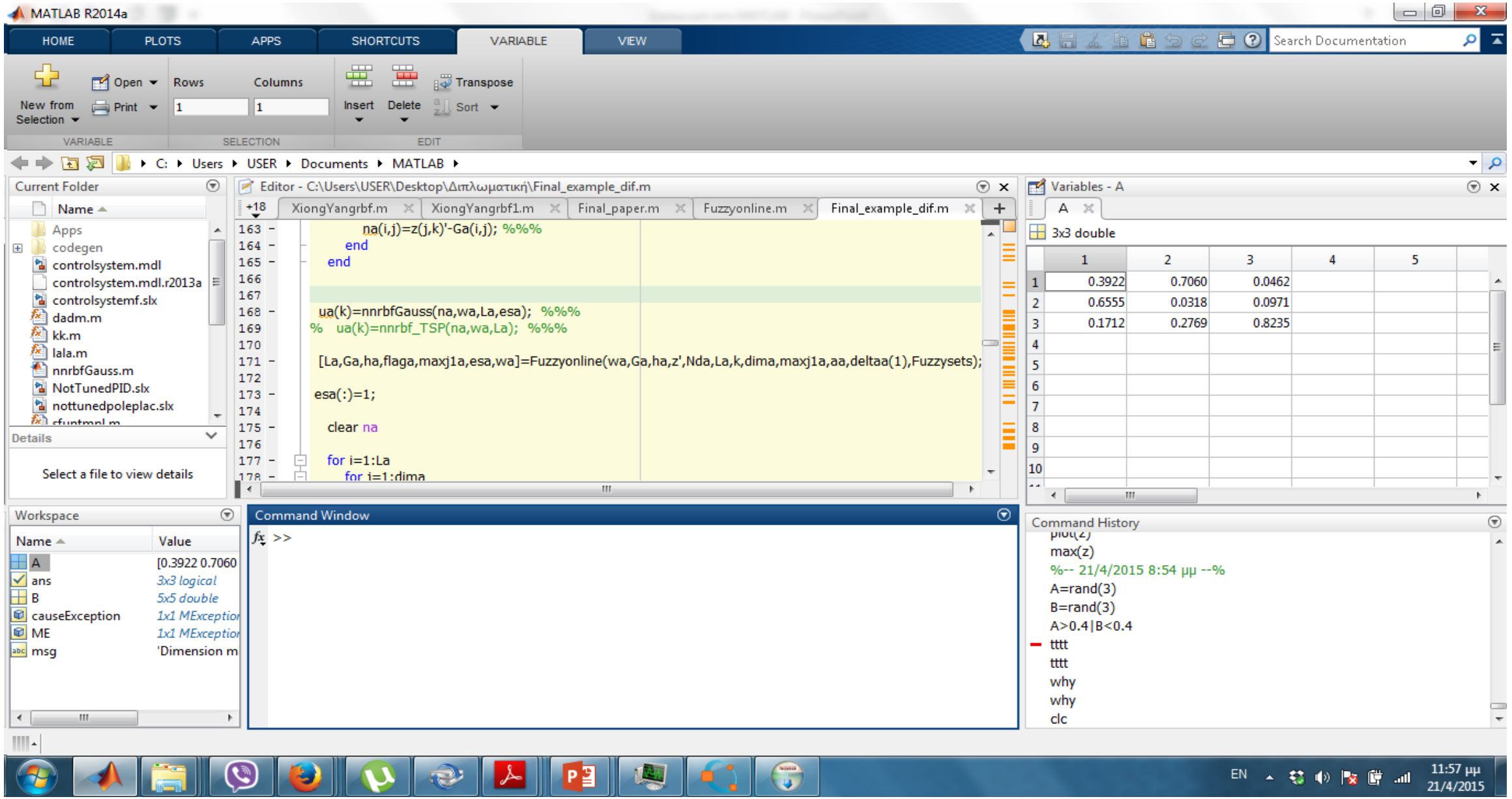
- Opensource-GNU OCTAVE

Πλεονεκτήματα

- Γρήγορος και εύκολος προγραμματισμός(high level language)
- Προσφέρει αντικειμενοστραφή Προγραμματισμό
- Ελάχιστη προσπάθεια για εισαγωγή-δήλωση μεταβλητών
- Εύκολη χρήση διανυσμάτων και πινάκων(MATrix LABoratory)
- Easy plotting functions
- Πολύ εύκολο debugging
- Πολλά και εύχρηστα toolboxes
- Μεγάλη κοινότητα που ανεβάζει αρχεία και βοηθούν (mathworks file exchange website)
- Το help της MATLAB είναι ικανό αν σου μάθει τέλεια τα πάντα..

΄Εχει και από αυτά (μειονεκτήματα)

- Δεν κάνει συμβολικούς υπολογισμούς με τόση ευκολία (Mathematica ή Mathcad καλύτερα)
- Όχι τόσο γρήγορη όσο η C++ ή Fortran, ειδικά σε computationally demanding problems (βλέπε monte carlo)
- Όχι ελεύθερο λογισμικό…για αυτό *Octave*


MATLAB R2014a
HOME
PLOTS
APPS
SHORTCUTS
VARIABLE
VIEW
Search Documentation
New from Selection
Open
Print
Rows
Columns
Insert
Delete
Transpose
Sort
VARIABLE
SELECTION
EDIT
C: ▸ Users ▸ USER ▸ Documents ▸ MATLAB ▸
Current Folder
Name
Apps
codegen
controlsystem.mdl
controlsystem.mdl.r2013a
controlsystemf.slx
dadm.m
kk.m
lala.m
nnrbfGauss.m
NotTunedPID.slx
nottunedpoleplac.slx
Details
Select a file to view details
Editor - C:\Users\USER\Desktop\Διπλωματική\Final_example_dif.m
XiongYangrbf.m
XiongYangrbf1.m
Final_paper.m
Fuzzyonline.m
Final_example_dif.m
163 na(i,j)=z(j,k)'-Ga(i,j); %%%
164 end
165 end
166
167
168 ua(k)=nnrbfGauss(na,wa,La,esa); %%%
169 % ua(k)=nnrbf_TSP(na,wa,La); %%%
170
171 [La,Ga,ha,flaga,maxj1a,esa,wa]=Fuzzyonline(wa,Ga,ha,z',Nda,La,k,dima,maxj1a,aa,deltaa(1),Fuzzysets);
172
173 esa(:)=1;
174
175 clear na
176
177 for i=1:La
178 for j=1:dima
Variables - A
A
3x3 double
1 2 3 4 5
1 0.3922 0.7060 0.0462
2 0.6555 0.0318 0.0971
3 0.1712 0.2769 0.8235
4
5
6
7
8
9
10
Workspace
Name Value
A [0.3922 0.7060
ans 3x3 logical
B 5x5 double
causeException 1x1 MExceptior
ME 1x1 MExceptior
msg 'Dimension m
Command Window
>>
Command History
max(z)
%-- 21/4/2015 8:54 μμ --%
A=rand(3)
B=rand(3)
A>0.4|B<0.4
tttt
tttt
why
why
clc
EN
11:57 μμ
21/4/2015

Βασικά

“;” αν δε μπει στο τέλος μιας γραμμής, εκτυπώνει τα αποτελέσματα στο command window

- `a = 2;`
- `b = 3;`
- `c = a + b;` Αποτέλεσμα στο «c»
- `a + b` Αποτέλεσμα στο «ans»
  - Αν ξεχαστούν τότε θα εκτυπωθούν όλα τότε…πιθανότατα να έχετε πάρετε πτυχίο πριν τρέξει ο κώδικας

Διαφορά Διανύσματος γραμμής και στήλης

- Γραμμή:
```
a = [1 2 3]
a = [1, 2, 3]
```
- Στήλη:
```
b = [1; 2; 3]
```
- Ορισμός διανύσματος:
```
c = 0:5:100 (από 0 έως 100 με βήμα 100)
```
- Δημιουργία με ορισμό μεγέθους διανύσματος:
```
d = linspace(0, 100, 21)
```
- Transpose:
```
f = e'
```

Πράξεις Διανυσμάτων

Πρόσθεση

Αφαίρεση

Εσωτερικό γινόμενο (dot(a,b))

- a=[1 3 2]
- b=[4 0 3]
- Inner_product=dot(a,b) ή a*b’

Εξωτερικό γινόμενο (cross(a,b))

cross_product=cross(a,b) (a x b)

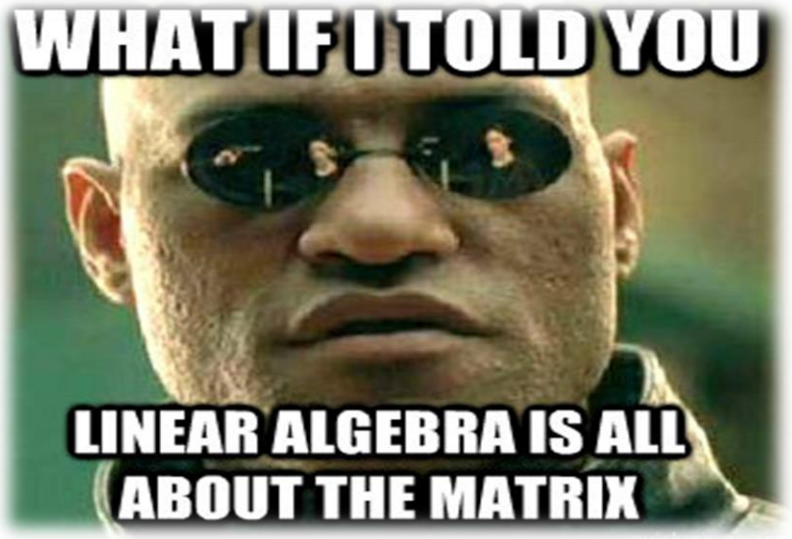
SOS με διαστάσεις…σύνηθες λάθος στη MATLAB… (P1=(A-B*K1)'*P1*(A-B*K1)+Q+K1'*R*K1)...εδώ θα γίνει πανικός

Βασικά (II)

Πράξεις στοιχείου επί στοιχείου

- `a.^2`
- `d = d./a`

- `Επίλυση γραμμικών εξισώσεων`
- `X=A\b (A x=b) … Πολλοί τρόποι για επίλυση.. Θα δούμε…`



Ορισμός πινάκων... κι άλλα

Creating matrices (try these out!)

- Matrix of zeros: `B = zeros(3); B = zeros(3,2);`
- Matrix of ones: `C = ones(3); C = ones(3,2);`
- Random matrix: `R = rand(3); R = rand(3,2);`
- Normally distributed: `RD = randn(3)`

Χαρακτηριστικά Πίνακα

- Size
```
[nRows, nColumns] = size(A)
nColumns = size(A,2)
```
- Largest dimension `maxDim = length(A)`
- Number of elements `nElements = numel(A)`

Πριν από κάθε επανάληψη ή στην αρχή του script πρέπει να ορίζονται οι πίνακες για εξοικονόμηση χρόνου

Πολύ χρήσιμο για κάποια loops

- Ένα στοιχείο:
- περισσότερα:
- Διάστημα στοιχείων:
- Τελευταίο:
- όλα:

```
>> a(3)
```

```
>> a([1,3])
```

```
>> a(2:4)
```

```
>> a(end)
```

```
>> a(:)
```

Χρήσιμα

- `sum(A,dim);`
- `sum(A(:));`

- `det(A);`
- `inv(A);`
- `eig(A);`
- `cond(A);`
- `norm(A,p);`

Αντιστροφή Πίνακα

- inv(A)…κακή ακρίβεια υπολογισμών και πολύ αργός υπολογισμός
- A^(-1)…πολύ κακή πρακτική για τους ίδιους λόγους..
- Μπορεί να χρησιμοποιηθεί αντίθετα το εξής

```
Inv_A=A\eye(size(A,2))
```

?

Αντιστροφή Πίνακα

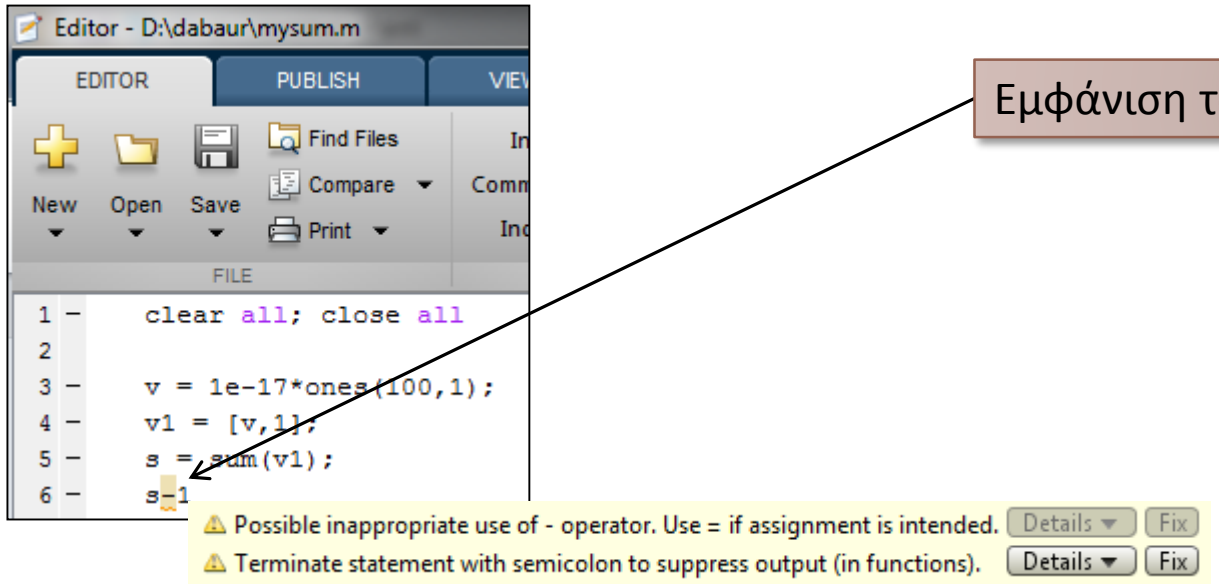
- inv(A)…κακή ακρίβεια υπολογισμών και πολύ αργός υπολογισμός
- A^(-1)…πολύ κακή πρακτική για τους ίδιους λόγους..
- Μπορεί να χρησιμοποιηθεί αντίθετα το εξής

Inv_A=A\eye(size(A,2))

$A*Inv\_A=I_N$, N=dimension(A)

Σαν γραμμικό σύστημα με
άγνωστο τον αντίστροφο…

Script..



Εμφάνιση του προβλήματος

Λογικές συνθήκες

- `<, >, <=, >=, ==, ~=`

Το όχι συμβολίζεται ως«~»

AND - OR

- `&&, ||` Για βαθμωτά
- `&, |` Σύγκριση στοιχείου προς στοιχείου

Λογικές συνθήκες

- <, >, <=,

Το όχι συμβολίζε

AND - OR

- &&, ||
- &, |

Παράδειγμα

```
A=rand(3);
B=rand(3);
A>0.5|B<0.5
```

```
Command Window
>> exampleseminar

ans =

   1   1   1
   1   1   0
   1   0   1
```

Κι άλλα

- `any(A,dim);` True αν τουλάχιστον ένα≠ 0
- `all(A,dim);` True αν όλα είναι ≠ 0
- `xor(A,B);` True αν ένα είναι = 0 και στο άλλο ≠ 0
- `isnumeric(A);` True αν είναι αριθμός
- `isfinite(A);` True για κάθε στοιχείο που δεν είναι NaN nor inf

Μπορείς πάρεις τα στοιχεία που σε ενδιαφέρουν!

- `A(A<0);`
- `A(isfinite(A));`
- `A(A == B);`

For-loops

General form of *for*-loops:

```
for variable = expression
      statements;
end
```

Example:

```
>> exp2 = 0;
for i = 0:20
exp2 = exp2 + 2^i / factorial(i);
end
>> exp2

exp2 =

    7.3891
```

ctrl+c *τερματίζει την επανάληψη*

if

- ```
  if expression
      statements;
  elseif expression
      statements;
  else
      statements;
  end
  ```

Example

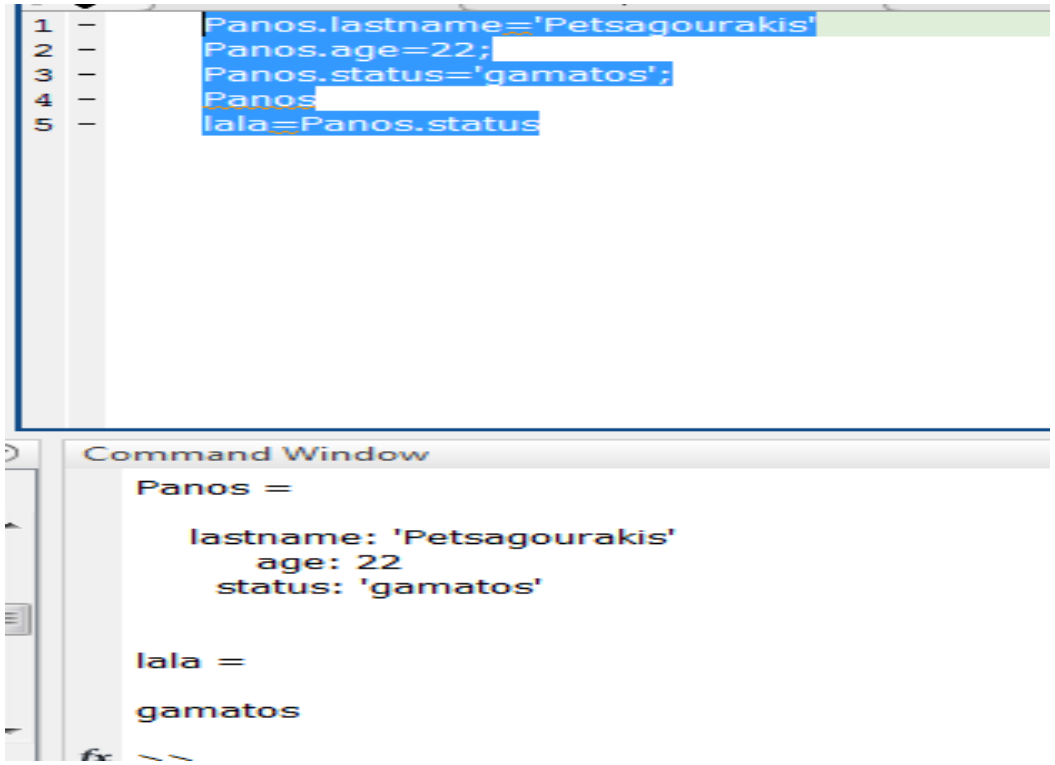
```
if YourNumber < 0
    disp('Negative')
elseif YourNumber > 0
    disp('Positive')
else
    disp('Zero')
end
```

`break;` Exits the loop

Μπορεί να χρησιμοποιηθεί για βγει από for…
```

Struct..what?

Structs είναι πίνακες με την ιδιότητα να έχουν «fields». Fields είναι διαφορετικά ήδη τύπων και λαμβάνονται ως εξής A.home, με A να είναι ο πίνακας. Χρησιμοποιείται για να αποθηκεύονται διαφορετικά ήδη πληροφορίας



Functions

function *outvar* = *funcname*(*arglist*)

% *helpcomments*

statements

outvar = *value*;

Αποθήκευση ως funcname.m στο path που θα τρέξει το main script

Function

function *outvar = fu*

% helpcomments

statements

outvar = value;

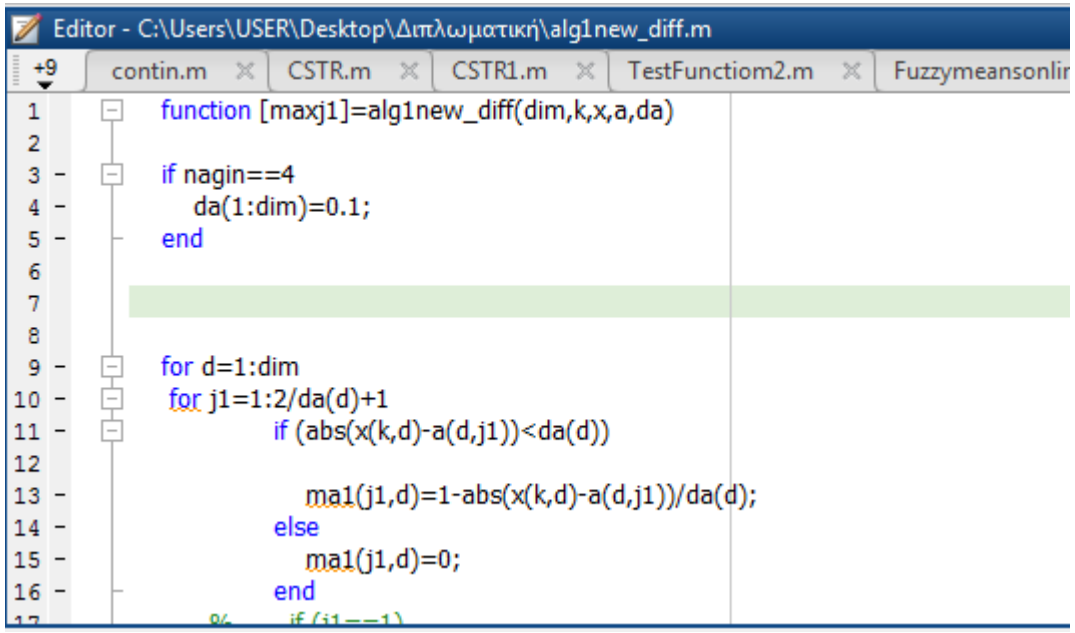
Αποθήκευση ως fun

```
1   function [maxj1]=alg1new_diff(dim,k,x,a,da)
2
3   for d=1:dim
4    for j1=1:2/da(d)+1
5           if (abs(x(k,d)-a(d,j1))<da(d))
6
7              ma1(j1,d)=1-abs(x(k,d)-a(d,j1))/da(d);
8           else
9              ma1(j1,d)=0;
10          end
11       %     if (j1==1)
12        %       max1=ma1(1);
13        %       maxj1=1
14        %    else
15        %       if(max1<ma1(j1));
16        %          max1=ma1(j1);
17        %          maxj1=j1
18        %    end
19        %end
20     end
21             [max1(d) maxj1(d)]=max(ma1(:,d));
22
23  end
24
25  end
```

Variable Argument List

Στις συναρτήσεις υπάρχει η πιθανότητα, όταν θα τρέξει να μην είναι γνωστές πάντα όλες οι μεταβλητές εισόδου, για αυτό υπάρχει η μεταβλητή nargin, τις οποία η τιμή είναι ίση με τον αριθμό που όντως μπήκαν μέσα στη συνάρτηση

Άλλο παράδειγμα είναι σε ένα φυσικό πρόβλημα που μπορεί να μην είναι γνωστός κάποιος συντελεστής τριβής, και τότε να λαμβάνεται μια Default τιμή από τον κώδικα.



Variable Argument List (II)

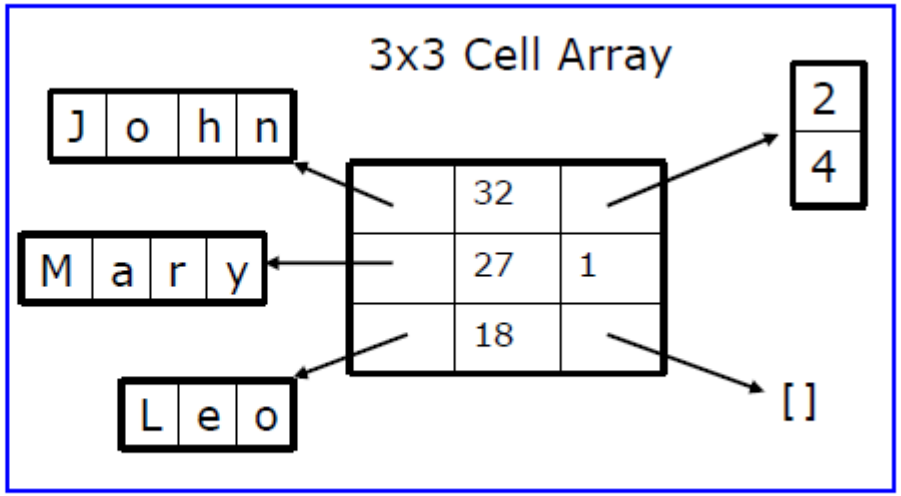
Ο χρήστης έχει τη δυνατότητα να ορίσει
να περνάνε όσες μεταβλητές θέλει
μέσα στη συνάρτηση

```
1   function c=exampleseminar(varargin)
2
3 -  if (nargin==2)
4 -      t=cell2mat(varargin(1));
5 -      r=cell2mat(varargin(2));
6 -  end
7
8 -  c=t+r;
9
```

SOS το varargin είναι τύπου cell..
Cell: είναι πίνακες όπου μπορούν να
αποθηκεύουν πολλά είδη μεταβλητών
και για αυτό πρέπει να μετατραπούν σε
αριθμούς (cell2mat) τέτοιες εσωτερικές
συναρτήσεις υπάρχουν πολλές και συναντώνται
πολύ έντονα στην ανάπτυξη εφαρμογών.

3x3 Matrix

1.2	-3	5.5
-2.4	15	-10
7.8	-1.1	4



Variable Argument List (III)

Το αντίστοιχο υπάρχει για την έξοδο

Όπου είναι σίγουρα πιο χρήσιμο γιατί συχνά μπορεί

Να θέλουμε να εξάγουμε μια τιμή υπό συνθήκη.

```
1   function [c,varargout]=exampleseminar(varargin)
2
3 -  if (nargin==2)
4 -      t=cell2mat(varargin(1));
5 -      r=cell2mat(varargin(2));
6 -  end
7
8 -  c=t+r;
9 -  if c>0.1
10 -  varargout{1}=[2];
11 -  end
12 -  end
13
```

Hand

fhandle =

Π.χ. fhand

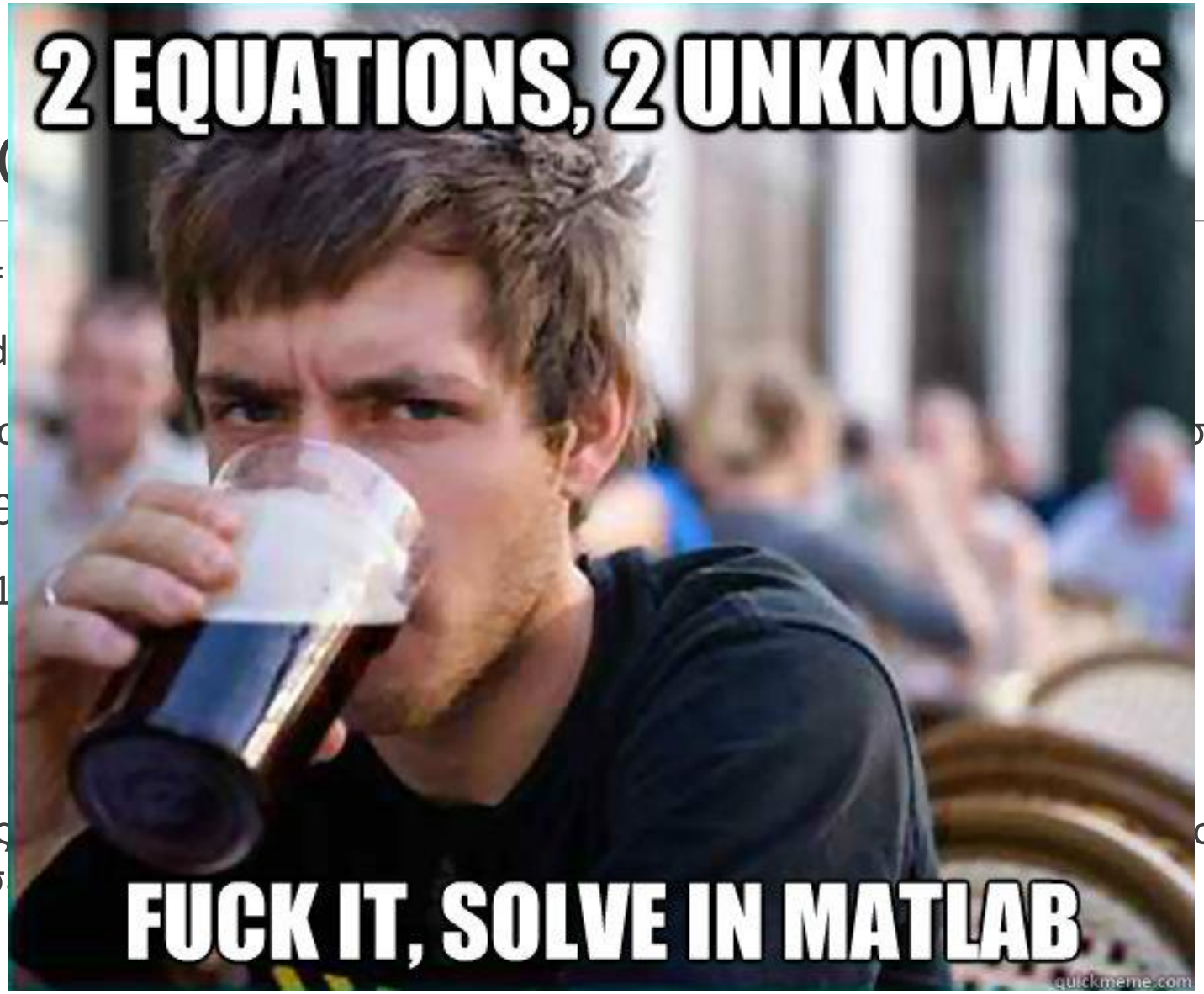
Πλέον μπο συναρτήσεις

Έστω ότι θ

$f_1=x^2+y^2 -1$

$f_2=y-x=0$

Χρήση της σωτερικές συναρτήσ



Handle Function

fhandle = @(arglist) expression

Π.χ. fhandle=@(x,y) x^2+y^2

Πλέον μπορούν γίνουν υπολογισμοί και να μπουν σε εσωτερικές ρουτίνες συναρτήσεις

Έστω ότι θέλουμε να λύσουμε το σύστημα

$f_1=x^2+y^2 -1=0$

$f_2=y-x=0$

```
1   function [f]=exampleseminar(x)
2
3 -   f1=x(1)^2+x(2)^2-1;
4 -   f2=x(2)-x(1);
5 -   f=[f1;f2];
```

Χρήση της fsolve-> fsolve(@exampleseminar,X0)…Έτσι δουλεύουν πολλές εσωτερικές συναρτήσεις.

Global variables…

Ο προγραμματιστής μπορεί να διαλέξει μεταβλητές που τις ορίζει global και απλά εισέρχονται στη function (εννοείται και στο script)

```
function lala=lk(x,y)

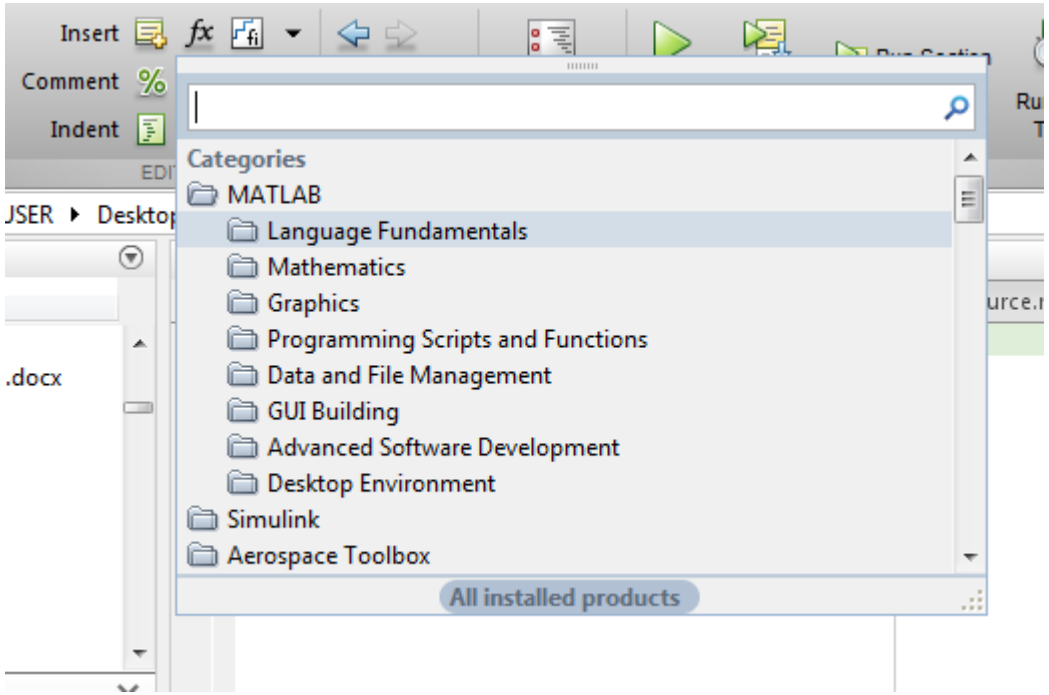
global ALPHA BETA

lala=ALPHA*BETA/(x+y);

end
```

Δεν θεωρείται καλή πρακτική αφού μπορεί να γίνει συγχυση

Integration-diff-ode




```
h=0.1;
X=[0:h:1];
Y=exp(-X.^2);
Q = trapz(X,Y)

r=diff(Y)/h
```

Example ode45

```
1   function [dx_dt]= TestFunctiom(t,x,u)
2   %    dx_dt(2)=-x(1)*x(2)/(1+x(1))-(u+10)*x(2)/1+50+0.1*cos(0.001*t);
3   %    dx_dt(1)=x(1)*x(2)/(1+x(1))-0.5*x(1)-(u+10)*x(1)/1+(u)*5+0.1*cos(0.001*t)
4   %
5       dx_dt(2)=-x(1)*x(2)/(1+x(1))-(u+10)*x(2)+50+0.1*cos(0.001*t);
6       dx_dt(1)=x(1)*x(2)/(1+x(1))-0.5*x(1)-(u+10)*x(1)+(u)*10+0.1*cos(0.001*t);
7   %    dx_dt(2)=0.5*x(1)*x(2)/(1+x(1))-0.1*x(1)^2-(u)*x(2)+0.1*cos(0.001*t);
8   %    dx_dt(1)=-0.5*x(1)*x(2)/(1+x(1))-(u)*x(1)+u*10+0.1*cos(0.001*t);
9
10  dx_dt=dx_dt';
11  return
```

```
199
200 %   um(k)= (u(k)+1)*0.5*(300-1000)+1000;
201 %   xm(k)= (x2(k)+1)*0.5*(400-100)+100;
202 % u(1)=1
203 %     [ts xs]=ode45(@TestFunctiom,[0 100],[x1(1),x2(1)],[],u(1));
204 % plot(xs)
205     [ts xs]=ode45(@TestFunctiom,[0 0.1],[x1(k),x2(k)],[],u(k));
206     x1(k+1)=xs(end,1);
207     x2(k+1)=xs(end,2);
208 %   x2(k+1)=2*(x2(k+1)-100)/(400-100)-1;
```

Runge-kutta

Διάστημα επίλυσης

Άλλη εσωτερική μεταβλητή

Αρχικές συνθήκες

Διάφορα option που μπορούν να μπούν

Input-Output Data

n = input('*promptstring*')

disp(*value*)

fprintf('ότι θες και μετά το format', x, ...)

Format Code	**Description**
%d	Integer format
%e	Scientific format with lowercase e
%E	Scientific format with uppercase E
%f	Decimal format
%g	The more compact of %e or %f

Control Code	**Description**
\n	Start new line
\t	Tab

Import-Output from txt

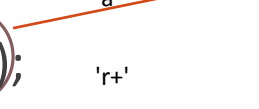
```
x = 0:1:10;

A = [x; exp(x)];

fileID = fopen('exp.txt','w');

fprintf(fileID,'%d\t',x);

fclose(fileID);
```



'r'	Open file for reading.
'w'	Open or create new file for writing. Discard existing contents, if any.
'a'	Open or create new file for writing. Append data to the end of the file.
'r+'	Open file for reading and writing.
'w+'	Open or create new file for reading and writing. Discard existing contents, if any.
'a+'	Open or create new file for reading and writing. Append data to the end of the file.
'A'	Append without automatic flushing of the current output buffer. (Used with tape drives.)
'W'	Write without automatic flushing of the current output buffer. (Used with tape drives.)

)

Input-Output Excel

Num=xlsread(filename,sheet,xlrange)

- wa=xlsread('on1.xlsx','A1:J1');
- wc=xlsread('on1.xlsx','A2:J2');

xlswrite(filename,variables,sheet,xlrange)

Matlab wants the D