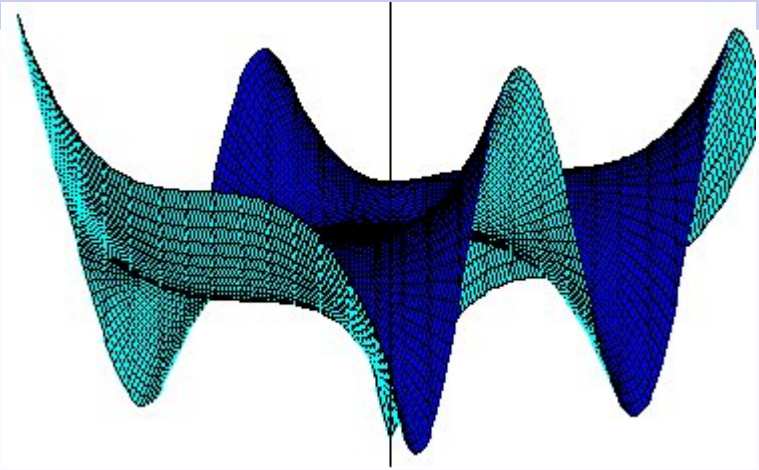


Scilab

Prof. S. A. Katre
Dept. of Mathematics, University of Pune
sakatre@math.unipune.ernet.in
sakatre@gmail.com
sakatre@bprim.org

December 25, 2009

1. factorial

Scilab does exact computations to some extent. But KASH is a better software for this purpose.

```
-->factorial(6)
 ans  =
     720.

-->factorial(20)
 ans  =
     2.433D+18

-->factorial(12)
 ans  =
     4.790D+08

-->factorial(11)
 ans  =
     39916800.
```

2. factor a number

Scilab can factor small numbers up to 10-12 digits, but not larger numbers. KASH software can factor numbers more efficiently.

```
-->factor(15)
 ans  =

    3.    5.

-->factor(1534561890)
 ans  =

    2.    3.    5.    271.    188753.

-->factor(1534561890345267 9)
 !--error 17
stack size exceeded!
Use stacksize function to increase it.
Memory used for variables: 34614
Intermediate memory needed: 123877436
Total memory available: 5000000
```

```
at line      20 of function primes called by
line    25 of function factor called by :
factor(1534561890345267 9)

-->stacksize()
 ans  =

    5000000.    34066.

-->stacksize(50000000)

-->stacksize()
 ans  =

    50000000.    34059.
```

```
-->factor(1534561890345267 9)
 !--error 17
stack size exceeded!
Use stacksize function to increase it.
Memory used for variables: 34611
Intermediate memory needed: 123877436
Total memory available: 50000000
at line      20 of function primes called by :
line    25 of function factor called by :
factor(1534561890345267 9)

-->stacksize(500000000)
                      !--error 1504
stacksize:
Out of bounds value.Not in[180000,268435454].
```

```
-->stacksize(260000000)
                        !--error 999
stacksize: Cannot allocate more memory.
Try stacksize('max').

 -->stacksize('max')

-->stacksize()
 ans  =

    50000000.    34059.

-->factor(15345618903452)
 ans  =

    2.    2.    457813.    8379851.
```

3. factoring real polynomials

The ‘polfact’ function factorises the given polynomial with real coefficients into irreducible polynomials of degree 1 and 2 over reals. Note that since complex (nonreal) roots, if any, of a polynomial with real coefficients occur in cojugate pairs, combining 2 such roots we get a quadratic polynomial with real coefficients and this polynomial is irreducible over reals. Hence any nonconstant polynomial with real coefficients factorises as a product of linear and quadratic irreducible polynomials. The ‘polfact’ command in Scilab gives the leading coefficient at the beginning, then monic linear and quadratic irreducible factors. ‘polfact’ function does not factorise a polynomial with complex (nonreal) coefficients.

```
-->x=poly(0,'x')

 x   =

    x
```

```
-->polfactor(x^4-1)
                  !--error 4
Undefined variable: polfactor
```

The command was wrong.

```
 -->polfact(x^4-1)
 ans  =

                                        2
    1   - 1 + x     1 + x     1 + x
```

```
-->polfact(x^7-1)
 ans  =
           column 1 to 4
                            2                  2
 1  -1+x    1-1.2469796x+x  1+0.4450419x+x

         column 5
                       2
    1 + 1.8019377x + x
```

```
-->roots(x^7-1)
 ans  =

    0.6234898 + 0.7818315i
    0.6234898 - 0.7818315i
  - 0.9009689 + 0.4338837i
  - 0.9009689 - 0.4338837i
  - 0.2225209 + 0.9749279i
  - 0.2225209 - 0.9749279i
    1.
```

See how the conjugate roots combine to give real quadratic factors.

```
-->polfact(3x^7+x-1)
            !--error 276
Missing operator, comma, or semicolon.
```

The error was that we had to write $3 * x^7$. $*$ was missing.

```
-->polfact(3*x^7+x-1)
 ans  =

         column 1 to 3

                              2
 3  0.8749381+1.6575743x+x   0.8088448+0.2295991x+x

         column 4 to 5

                            2
 0.6591506-1.1725921x+x    -0.7145812+x
```

Here 3 is the leading coefficient seen at the beginning.

```
-->polfact(x^2+%i)
 !--error 10000
polfact: Input argument #1 must be real.
at line 19 of function polfact called by :
polfact(x^2+%i)
```

4. n-th roots

```
->%i

 %i  =

    i

 -->sqrt(%i)
 ans  =

    0.7071068 + 0.7071068i

 -->(%i)^(1/3)
 ans  =
    0.8660254 + 0.5i

-->(1)^(1/5)
 ans  =

    1.
```

5. Newton-Raphson Method

$f(x)$ and initial guess x_0 is given, find out 'zero' for $f(x)$.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Let $f(x) = \cos(x)$, and $x_0 = 10$

```
-->deff('[y]=f(x)','y=cos(x)');
```

```
-->deff('[y]=f1(x)','y=-sin(x)');
```

```
-->deff('[y]=g(x)','y=x-f(x)/f1(x)');
```

```
-->g(10)
 ans  =

    11.542351

-->g(ans)
 ans  =

    10.933672

-->g(ans)
 ans  =

    10.995653
```

```
-->g(ans)
 ans  =
    10.995574

-->g(ans)
 ans  =
    10.995574
```

We get a fixed point of the function g. The value of g obtained is a root of the original function f.
(N-R confusion. Mai Kahan Hoon? Ami Kothay?)

6. Newton-Raphson: More zeros

```
-->x=[0:.2:10]
x  =

        column  1 to 10
0. 0.2  0.4  0.6  0.8  1. 1.2  1.4  1.6  1.8

       column 11 to 20
2. 2.2  2.4  2.6  2.8  3. 3.2  3.4  3.6  3.8

       column 21 to 30
4. 4.2  4.4  4.6  4.8  5. 5.2  5.4  5.6  5.8

       column 31 to 40
6. 6.2  6.4  6.6  6.8  7. 7.2  7.4  7.6  7.8

       column 41 to 50
8. 8.2  8.4  8.6  8.8  9. 9.2  9.4  9.6  9.8

       column 51
10.
```

```
-->cos(x)
 ans  =

         column 1 to 5

1. 0.9800666  0.9210610  0.8253356   0.6967067

        column  6 to 10

0.5403023  0.3623578  0.1699671  -0.0291995 -0.2272021

        column 11 to 15

-0.4161468  -0.5885011  -0.7373937 -0.8568888 -0.942222

        column 16 to 20

-0.9899925 -0.9982948 -0.9667982 -0.8967584 -0.7909677
```

```
        column 21 to 25

-0.6536436 -0.4902608 -0.3073329 -0.1121525 0.0874990

         column 26 to 30

0.2836622 0.4685167  0.6346929  0.7755659  0.8855195

         column 31 to 35

0.9601703 0.9965421  0.9931849  0.9502326  0.8693975

         column 36 to 40

0.7539023 0.6083513  0.4385473 0.2512598  0.0539534
```

```
         column 41 to 45

-0.1455000 -0.3391549 -0.5192887 -0.6787200 -0.8110930

         column 46 to 50

-0.9111303 -0.9748436 -0.9996930 -0.9846879 -0.9304263

         column 51

-0.8390715
```

We find from the above list of values of $\cos x$ from 0 to 10, that $\cos$ changes sign 3 times in [0, 10]. We thus expect to find 3 zeros of $\cos x$, one between 1.4 and 1.6, one between 4.6 and 4.8, and one between 7.8 and 8.

We use the Newton-Raphson formula
$f(x_{n+1}) = x_n - f(x_n)/f'(x_n)$.
Let $f(x) = \cos(x)$, and consider the values of x for which $f(x)$ is nearer to zero.

Thus let $x_0 = [1.6, 4.8, 7.8]$. The other way to get these values of x is to count the number of the terms starting from the first value i.e. 1. Here x is a vector. Since the 8th and the 9th values of $\cos x$ have opposite signs and the 9th value is closer to 0, we take x(9) as the initial approximation (which is 1.6). Similarly, $x(25) = 4.8$ and $x(40) = 7.8$ and we can as well take $x_0 = [x(9), x(25), x(40)]$

```
-->deff('[y]=f(x)','y=cos(x)');
-->deff('[y]=f1(x)','y=-sin(x)');
-->deff('[y]=g(x)','[y]= x-f(x)./f1(x)');

-->x0=[1.6,4.8,7.8]    (vector of approximations to root
 x0  =
    1.6    4.8    7.8
(Alternatively x0=[x(9),x(25),x(40)])
```

```
-->g(x0)
 ans  =
    1.570788    4.7121641    7.8540341

-->g(ans)
 ans  =
    1.5707963    4.712389    7.8539816

-->g(ans)
 ans  =
    1.5707963    4.712389    7.8539816

(We have got the 3 zeros of cos x
between 0 and 10.)

-->%pi/2

 ans  =
    1.5707963
```

This checks that the zeros are $\pi/2, 3\pi/2$ and $5\pi/2$ as expected.

```
-->x=poly(0,'x')
 x  =

    x

-->roots(x^3-x+1)
 ans  =

    0.6623590 + 0.5622795i
    0.6623590 - 0.5622795i
  - 1.324718
```

The polynomial has 1 real root and 2 nonreal roots which are complex conjugates. Recall that complex roots of a polynomial with real coefficients occur in conjugate pairs.

Let us find a complex root of $f(x) = x^3 - x + 1$ by Newton Raphson method. To get it we have to start with a complex (nonreal) initial approximation, e.g. $1 + i$.

```
-->f(1+%i)
 ans  =
   - 2. + i


-->deff('[y]=f(x)','y=x^3-x+1')

-->deff('[y]=f1(x)','y=3*x^2-1')

-->deff('[y]=g(x)','y=x-f(x)/f1(x)')

-->g(1+%i)
 ans  =
     0.7837838 + 0.7027027i
```

```
-->g(ans)
 ans  =
     0.6860940 + 0.5813564i

-->g(ans)
 ans  =
     0.6633325 + 0.5625179i

-->g(ans)
 ans  =
     0.6623599 + 0.5622788i

-->g(ans)
 ans  =
     0.6623590 + 0.5622795i

-->g(ans)
 ans  =
     0.6623590 + 0.5622795i
```

7. Bisection Method

The bisection method is very slow as compared to Newton Raphson method.
To find a root of $f(x)$ we first find some CLOSE values of a and b for which $f(a)$ and $f(b)$ have opposite signs.
Then find $c = (a + b)/2$.
If $f(a)$ and $f(c)$ have same signs (so that $f(b)$ and $f(c)$ have opposite signs, consider c as better than a and replace a by c (i.e. put $a = c$) and keep b as it is.
If $f(b)$ and $f(c)$ have same signs, replace b by c, i.e. put $b = c$.
Then find $c = (a + b)/2$ for the new a and b and proceed till you get sufficiently close values of a and b. Then declare $c = (a + b)/2$ as an (approximate) root of $f(x)$.
We shall now find a real root of $f(x) = x^3 - x + 1$ by bisection method.

```
-->x=poly(0,'x')
 x  =
     x

-->roots(x^3-x+1)
 ans  =
     0.6623590 + 0.5622795i
    0.6623590 - 0.5622795i
  - 1.324718

-->deff('[y]=f(x)','y=x^3-x+1')

-->f(3)
 ans  =
     25.

-->f(2)
 ans  =
     7.
```

```
-->f(0)
 ans  =
     1.
 -->f(-2)
 ans  =
   - 5.

-->a=0,b=-2
 a  =
     0.
 b  =
   - 2.

-->c=(a+b)/2
 ans  =
   - 1.
 -->d=[a,b,c]
 d  =
     0.  - 2.  - 1.
 -->f(d)
 ans  =
     1.  - 5.    1.
```

```
-->a=c
 a   =
   - 1.
 -->c
 c   =
   - 1.
 (This c is not relevant.)

-->c=(a+b)/2
 c   =
   - 1.5

-->d=[a,b,c],f(d)
 d   =
   - 1.   - 2.   - 1.5
 ans   =
     1.   - 5.   - 0.875
```

```
-->b=c
 b  =
   - 1.5
 -->c=(a+b)/2
 c  =
   - 1.25
 -->d=[a,b,c],f(d)
 d  =
   - 1.   - 1.5   - 1.25
 ans  =
     1.   - 0.875    0.296875

-->b=c,c=(a+b)/2,d=[a,b,c],f(d)
 b  =
   - 1.25
 c  =
   - 1.125
 d  =
   - 1.   - 1.25   - 1.125
 ans  =
     1.    0.296875    0.7011719
(We made a mistake in the choice of a,b.)
```

```
-->a=-1,b=-1.5,c=-1.25
 a   =
   - 1.
 b   =
   - 1.5
 c   =
   - 1.25
-->a=c
 a   =
   - 1.25

-->c=(a+b)/2,d=[a,b,c],f(d)
 c   =
   - 1.375
 d   =
   - 1.25   - 1.5   - 1.375
 ans   =
     0.296875   - 0.875   - 0.2246094
```

```
-->b=c
 b  =
   - 1.375

-->c=(a+b)/2,d=[a,b,c],f(d)
 c  =
   - 1.3125
 d  =
   - 1.25   - 1.375   - 1.3125
 ans  =
     0.296875   - 0.2246094    0.0515137

 --> a=c;

-->c=(a+b)/2,d=[a,b,c],f(d)
 c  =
   - 1.34375
 d  =
   - 1.3125   - 1.375   - 1.34375
 ans  =
     0.0515137   - 0.2246094   - 0.0826111
```

```
-->b=c
 b  =
   - 1.34375

-->c=(a+b)/2,d=[a,b,c],f(d)
 c  =
   - 1.328125
 d  =
   - 1.3125  - 1.34375  - 1.328125
 ans  =
     0.0515137  - 0.0826111  - 0.0145760
-->b=c
 b  =
   - 1.328125

-->c=(a+b)/2,d=[a,b,c],f(d)
 c  =
   - 1.3203125
 d  =
   - 1.3125  - 1.328125  - 1.3203125
 ans  =
     0.0515137  - 0.0145760    0.0187106
```

```
-->a=c
 a   =
   - 1.3203125
-->c=(a+b)/2,d=[a,b,c],f(d)
 c   =
   - 1.3242188
 d   =
   - 1.3203125  - 1.328125  - 1.3242188
 ans  =
     0.0187106  - 0.0145760    0.0021279

-->a=c
 a   =
   - 1.3242188

-->c=(a+b)/2,d=[a,b,c],f(d)
 c   =
   - 1.3261719
 d   =
   - 1.3242188  - 1.328125  - 1.3261719
 ans  =
     0.0021279  - 0.0145760  - 0.0062088
```

```
-->b=c
 b  =
   - 1.3261719
-->c=(a+b)/2,d=[a,b,c],f(d)
 c  =
   - 1.3251953
 d  =
   - 1.3242188  - 1.3261719  - 1.3251953
 ans  =
     0.0021279  - 0.0062088  - 0.0020367

-->b=c
 b  =
   - 1.3251953

-->c=(a+b)/2,d=[a,b,c],f(d)
 c  =
   - 1.324707
 d  =
   - 1.3242188  - 1.3251953  - 1.324707
 ans  =
     0.0021279  - 0.0020367    0.0000466
```

```
-->a=c
 a   =
    - 1.324707

-->c=(a+b)/2,d=[a,b,c],f(d)
 c   =
    - 1.3249512
 d   =
    - 1.324707   - 1.3251953   - 1.3249512
 ans   =
      0.0000466   - 0.0020367   - 0.0009948
```

The answer by Newton Raphson method comes out to be -1.324718 in a few steps only and $f(-1.324718)$ is $-.0000002$. This example illustrates that Newton Raphson method which has quadratic convergence is far superior to bisection method

8. Linear Algebra

Finding eigenvalues and eigenvectors

```
                    scilab-5.0.2

              Consortium Scilab (DIGITEO)
            Copyright (c) 1989-2008 (INRIA)
            Copyright (c) 1989-2007 (ENPC)
```

```
Startup execution:
  loading initial environment

-->A=10*rand(3,3)
 A  =

    2.1132487    3.3032709    8.4974524
    7.5604385    6.653811     6.8573102
    0.0022113    6.2839179    8.7821648
```

```
-->x=poly(0,'x')
 x  =

    x


-->det(x*eye(3,3)-A)
 ans  =

                                                2    3
  - 216.73055 + 22.971179x - 17.549225x + x

-->v=roots(ans)
 v  =

    0.3004267 + 3.5633462i
    0.3004267 - 3.5633462i
    16.948371
```

```
-->v(1)
 ans  =

    0.3004267 + 3.5633462i

-->spec(A)
 ans  =

    0.3004267 + 3.5633462i
    0.3004267 - 3.5633462i
    16.948371

-->rref(A-v(1)*eye(3,3))
 ans  =

    1.    0     0.0401986 + 1.113006i
    0     1.    1.349739 - 0.5674497i
    0     0     0
```

```
-->w=[-ans(1,3);-ans(2,3);1]
 w   =

  - 0.0401986 - 1.113006i
  - 1.349739 + 0.5674497i
    1.

-->(A-v(1)*eye(3,3))*w
 ans   =

   1.0D-14 *

  - 1.9290125i
    0.2664535 - 0.0888178i
  - 0.0444089i

-->clean(ans)
 ans   =

    0
    0
    0
```

```
-->roots(det(x*eye(3,3)-A))
 ans  =

    0.3004267 + 3.5633462i
    0.3004267 - 3.5633462i
    16.948371
```

9. Lagrange's Interpolation Formula

```
                    scilab-5.1.1

                Consortium Scilab (DIGITEO)
              Copyright (c) 1989-2009 (INRIA)
              Copyright (c) 1989-2007 (ENPC)
```

```
Startup execution:
  loading initial environment

-->v=[2,3]
 v  =

    2.    3.

-->sum(v)
 ans  =

    5.

-->deff('z=f(x,y,t)','z=x+y+t')
Warning : redefining function: f
Use funcprot(0) to avoid this message
```

```
-->f(1,2,3)
 ans  =

    6.

-->v=[1,2,9]
 v  =

    1.    2.    9.

 -->f(v(1),v(2),v(3))
 ans  =

    12.

-->a=[2,4,6,-10]
 a  =

    2.    4.    6.  - 10.

-->b=[5,10,18,-25]
 b  =

    5.    10.    18.  - 25.


-->x=poly(0,'x')
```

```
 x  =

    x


-->[a,b]
 ans  =

    2.    4.    6.  - 10.    5.    10.    18.  - 25.

-->[a;b]
 ans  =

    2.    4.     6.   - 10.
    5.    10.    18.  - 25.


-->deff('w=f(y,z,t,u,v)','w=v*(x-z)*(x-t)*(x-u)/((y-z)*(y-t)*(y-u))')
Warning : redefining function: f                          . Use funcprot(0) to avoid this m


-->f(2,4,6,-10,5)
 ans  =

                                    3
    12.5 - 3.9583333x + 0.0520833x

-->c=[f(a(1),a(2),a(3),a(4),b(1)),f(a(2),a(1),a(3),a(4),b(2)),f(a(3),a(2),a(1),a(4),b(3)
```

```
c  =


         column 1

                                    3
   12.5 - 3.9583333x + 0.0520833x

         column 2

                                           2             3
  - 21.428571 + 12.142857x - 0.3571429x - 0.1785714x

         column 3

                                 2           3
   11.25 - 7.3125x + 0.5625x + 0.140625x

         column 4

                                           2             3
  - 0.4464286 + 0.4092262x - 0.1116071x + 0.0093006x

-->sum(c)
 ans  =

                                  2            3
   1.875 + 1.28125x + 0.09375x + 0.0234375x
```

This gives the required interpolating polynomial.

This presentation is prepared using

PDFScreen

package with LaTeX.

10. About Scilab

- Scilab is a free software and is similar to the commercial software MatLab. It is prepared by a team of French mathematicians/computer scientists (INRIA).
- Version scilab-5.1.1 is available in 2009.
- It can work on windows as well as linux.
- Scilab understands many mathematical data types like vector, matrix, polynomial etc.
- Scilab has inbuilt functions.
- It also allow us to do programming, in which we can use inbuilt commands e.g. rank, inv etc.
- Scilab is case sensitive. So V and v are different in scilab.

1. Scilab has an editor called Scipad.

2. Scilab has graphics window.

3. Scilab provides good help.

4. You can get help on a specific topic by a simple command such as ‘help Matrix‘.

5. Scilab has demos.

In Scilab, vector and matrix are basic data types. A vector may also be considered as a matrix for computations.

```
-->V=[2,-4,5]

-->V
 V  =

    2.   - 4.    5.

-->V'
 ans  =

    2.
  - 4.
    5.
```

Define two vectors U and V and try $U + V$.
$U * V$
$U * V'$ What are your observations?
Try with other inbuilt functions of scilab for vectors such as
norm(V)
size(U) etc.

Products of vectors:
Dot product (a scalar) and component-wise product (a vector):

```
-->u
 u   =
    1.    2.    3.
 v   =
    2.    3.    4.

-->u*v
    !--error 10
inconsistent multiplication

-->u.*v
 ans   =
    2.    6.    12.
```

```
-->u*v'
 ans  =
    20.

-->u'*v
 ans  =
    2.    3.    4.
    4.    6.    8.
    6.    9.    12.
```

Other interesting data type in scilab is Matrix.
Here is an example of writing a matrix in scilab.

```
B=[1,2;3,4]
 B   =

    1.    2.
    3.    4.
```

Now try to write a matrix A such that $A = \begin{bmatrix} 1 & -3 & 2.4 \\ .5 & 0 & 3 \end{bmatrix}$

Try the following matrix operations for matrices of appropriate size :

- A+B
- 3*A
- A*B
- B*A
- A-B
- A^{20}

Using various commands for matrix operations and in built functions, it becomes easy to explain concepts in linear algebra at the college level.
For matrices there are in built commands to get elements, like

- A(2,1) (Gives the element in the second row and first column.)
- A(4) (Gives 4th element of the matrix, considering the matrix as a column-wise array)
- A(:,1) (All elements of the 1st column.)
- A(2,:) (All elements of the 2nd row.)
- size(A) (The number of rows and columns.)
- length(A) (The number of elements.)
- sum(A) (The sum of all elements.)
- trace(A)
- det(A)
- inv(A)
- spec(A)

11. Special Symbols in scilab

```
-->%pi
 %pi  =

    3.1415927
-->%i
 %i  =

    i

-->%e
 %e  =

    2.7182818
-->%inf
 %inf  =

   Inf
```

Trignometric functions

```
-->cos(0)
 ans  =

    1.

-->sin(%pi/2)
 ans  =

    1.
```

Other types of matrices:

- eye(3,3)

```
-->eye(3,3)
 ans  =

    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
```

- zeros(3,2)
- ones(3,2)
- clean(inv(A))
-

```
-->int(10*rand(3,3))
 ans  =

    8.    9.    3.
    6.    2.    2.
    3.    3.    5.
```

For symbolic computation

```
-->x=poly(0,'x')
 x   =

    x
```

```
-->A=[x,2*x;x^2,x+3]
 A   =

    x      2x

     2
    x      3 + x
```

```
-->det(A)
 ans   =

             2     3
    3x + x - 2x
```

It is also possible to find inverse of symbolic matrix.

```
-->inv(A)
 ans  =

        3 + x                - 2
     -----------          ---------
              2    3                2
     3x + x - 2x          3 + x - 2x

       - x                    1
     ---------            ---------
                2                   2
     3 + x - 2x           3 + x - 2x
```

```
-->roots(x^2-3*x+4)
 ans  =

    1.5 + 1.3228757i
    1.5 - 1.3228757i
```

Another way to define polynomial

```
-->V= [2 -4 5]

ans =
    2 -4 5

-->poly(V,'x','coeff')
 ans  =

                  2
    2 - 4x + 5x

-->poly([1,-2,3],'y','coeff')
 ans  =

                  2
    1 - 2y + 3y
```

11.1. Gauss-Jordan elimination

row-reduced echelon form
First form augmented matrix:

```
-->A
 A   =

!   3.     2.   - 5.  !
!   5.   - 1.     4.  !
! - 2.     6.   - 5.  !

-->b
 b   =

!   100.  !
!   250.  !
!   300.  !
```

```
-->A_aug = [A b]
 A_aug  =

!   3.    2.  - 5.    100. !
!   5.  - 1.    4.    250. !
! - 2.    6.  - 5.    300. !
```

Use function `rref` to get the row-reduced echelon form:

```
-->rref(A_aug)
 ans  =

!   1.    0.    0.    36.809816 !
!   0.    1.    0.    96.01227  !
!   0.    0.    1.    40.490798 !
```

This matrix gives the solution of X, Y, Z

11.2. Eigenvalues

To form the characteristic matrix, first define the polynomial variable 'lam' i.e. λ:

```
-->lam = poly(0,'lam')
 lam  =

    lam

-->[m n] = size(A)
 n  =

    3.
 m  =

    3.
```

Form the characteristic matrix as follows:

```
-->Ch_Mat = A-lam*eye(m,n)
 Ch_Mat  =

!   3 - lam      2          - 5          !
!                                        !
!   5          - 1 - lam      4          !
!                                        !
! - 2            6          - 5 - lam    !
```

The characteristic polynomial is the determinant of the characteristic matrix:

```
-->p = det(Ch_Mat)
 p  =

                          2      3
  - 163 + 57lam - 3lam - lam
```

The eigenvalues are the roots of the characteristic polynomial. These can be found using function ‘roots’:

```
-->lam = roots(p)
 lam   =

!   3.5878707 + 1.7736044i !
!   3.5878707 - 1.7736044i !
! - 10.175741               !
```

Function ‘spec’ calculates eigenvalues in the fourth column.

11.3. Quiz

1. To get help in scilab what should you enter?
2. To see the demo of plot command what should you enter?
3. How to get transpose of a vector?
4. What is the difference between `u*v` and `u.*v`

Problem Set

1. Find factorial(6). Get it also for 10, 11, 12, 50 and see the difference.
2. Find factors of numbers upto 10-12 digits. Try factoring numbers with more digits.
3. Express the numbers i, π, e using Scilab.
4. Find factors of a polynomial with real coefficients using 'polfact'.
   Note: First use "$x = poly(0,' x')$" to define the variable x. Using $polfact$ command you will get factors which are of degree 1 or 2 with real coefficients. Try factoring a polynomial with complex coefficients.
5. Find a real root of $f(x) = x^3 - x + 1$ using Newton-Raphson Method. Also find all roots using 'roots' command.
6. Find all roots of $f(x) = 2\cos x - 1$ between 1 and 10 using Newton-Raphson method.
7. Use bisection method to find a root of $f(x) = e^x - 2$.
8. Can you find a root of $f(x) = e^x$ using Scilab?

12. User defined functions

Inline function:

```
-->deff('[y]=f(x)','y=3*x+sin(x)')

-->f(3)
 ans  =

    9.14112

-->u
 u  =

    1.    2.    3.

-->f(u)
 ans  =

    3.841471    6.9092974    9.14112
```

13. Rows and Columns

Consider matrix A

```
A   =

    2.    3.    4.
    4.    6.    8.
    6.    9.    12.
```

Now to obtain 1st row:

```
-->A(1,:)
 ans   =

    2.    3.    4.
```

To change $R_1 \longrightarrow 2R_1$

```
-->A(1,:)=2*A(1,:)
 A   =

    4.    6.    8.
    4.    6.    8.
    6.    9.    12.
```

To change $C_1 \longrightarrow 5C_1$

```
-->A(:,1)=5*A(:,1)
 A   =
    20.    6.    8.
    20.    6.    8.
    30.    9.    12.
```

To perform operation as $R_1 \rightarrow R_1 - R_2$

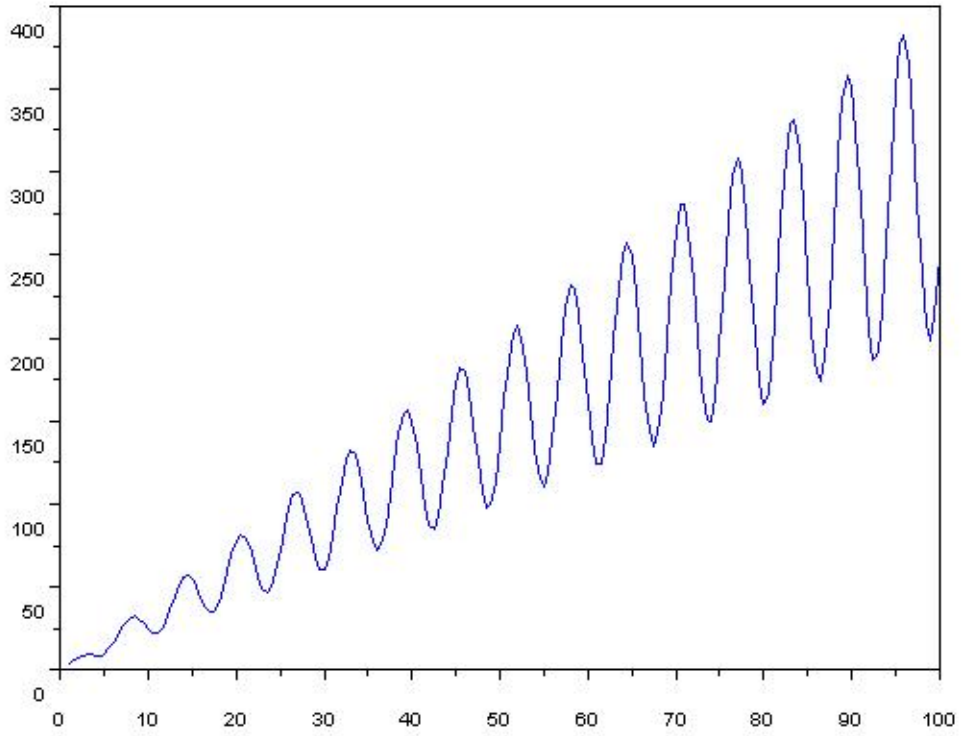
```
-->A(1,:)=A(1,:)-A(2,:)
 A   =

    0.      0.      0.
    20.     6.      8.
    30.     9.      12.
```

To perform operation as $R_2 \rightarrow R_2 - \frac{2}{3}R_3$

```
-->A(2,:)=A(2,:)-(2/3)*A(3,:)
 A   =

    0.      0.      0.
    0.      0.      0.
    30.     9.      12.

-->
```

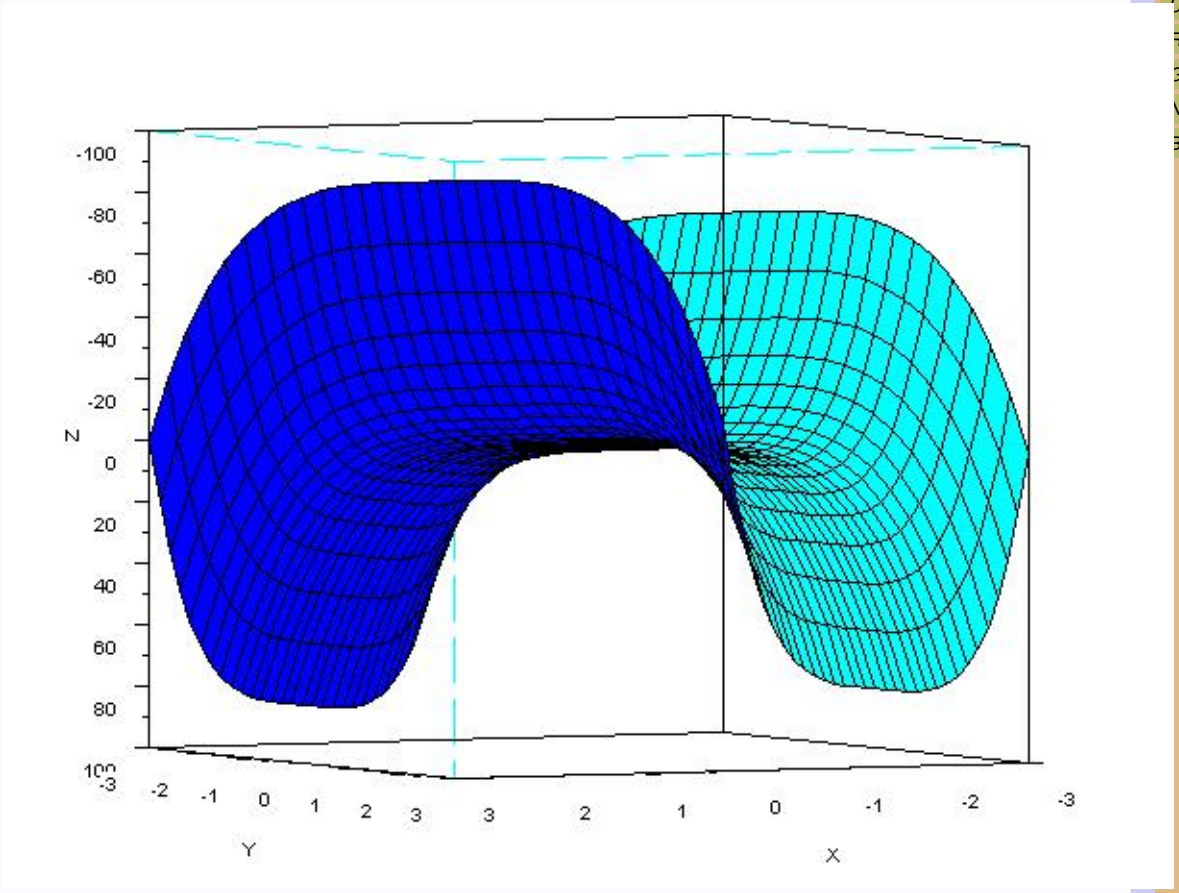
14. Graphics

```
-->deff('[y]=f2(x)','y=3*x+x.*sin(x)')
-->x=1:.5:100;
 -->y=f2(x);


 -->plot(x,y)
```

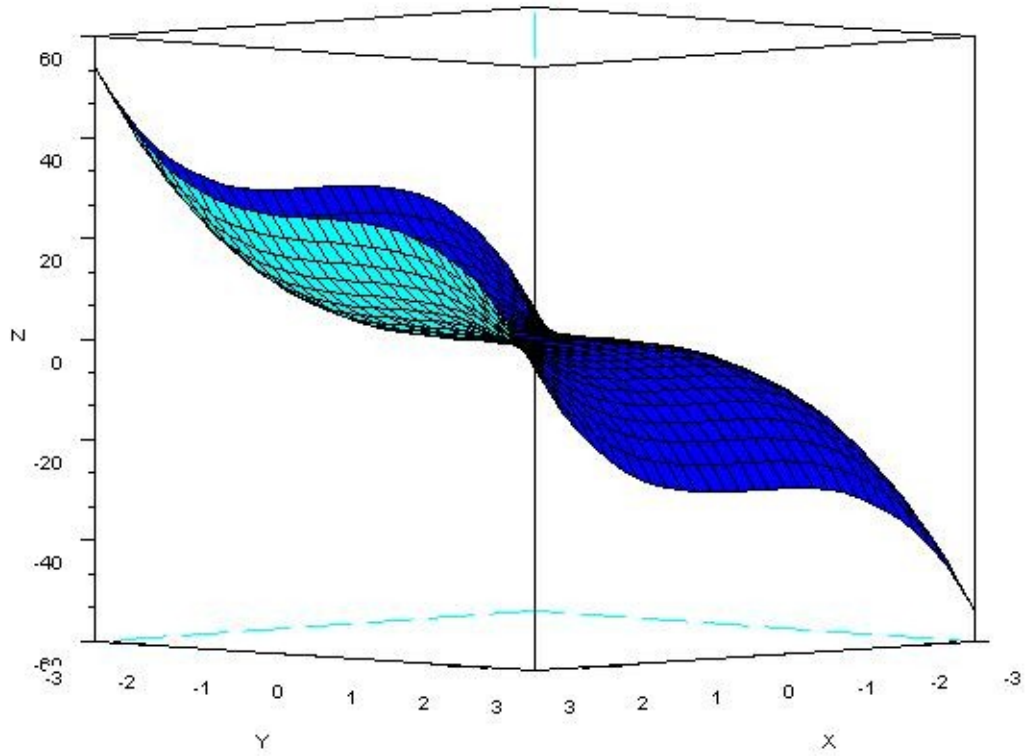


To obtain 3-D figure for the equation $z = x^4 - y^4$.
Note that command **fplot3d** has arguments as x, y and the function f.

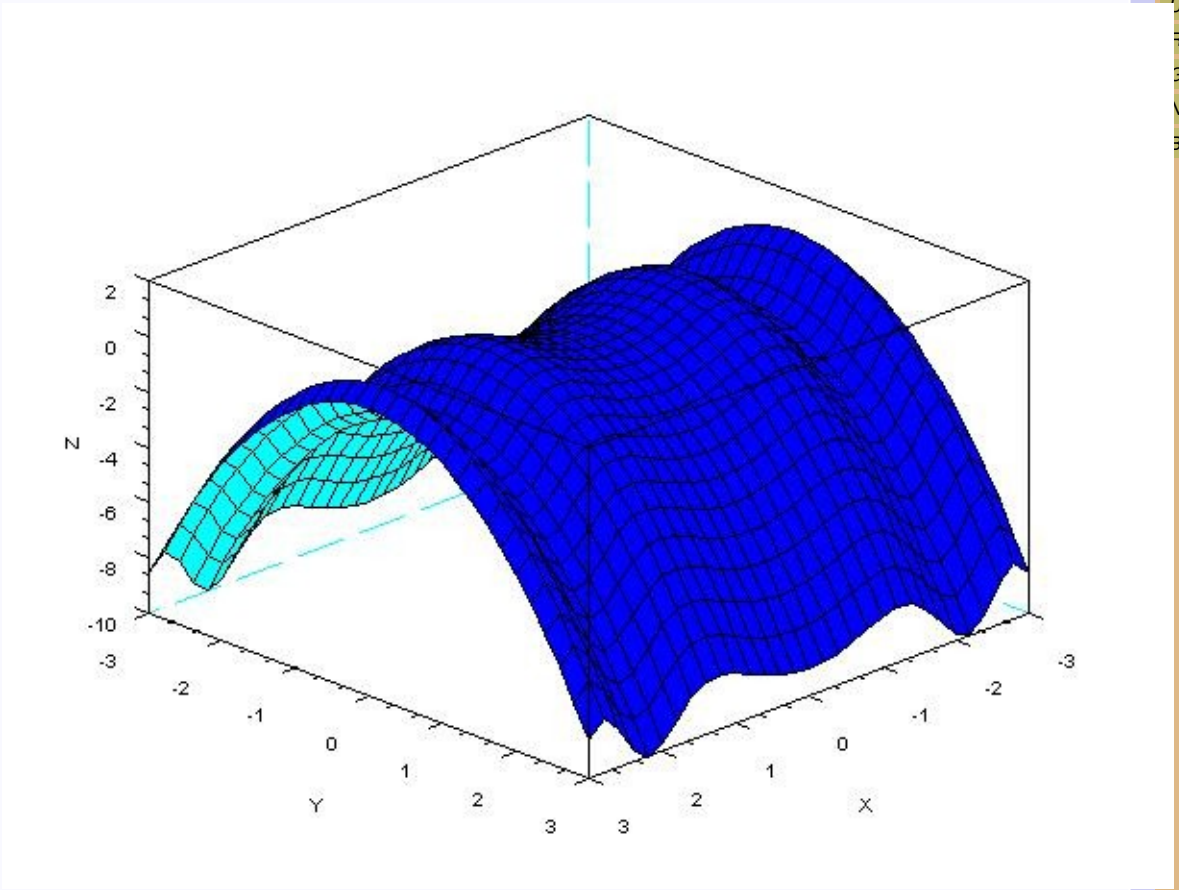
```
-->deff('z=f(x,y)','z=x^4-y^4')

-->x=-3:0.2:3 ;y=x ;

-->clf() ;fplot3d(x,y,f)
```



```
-->deff('z=f(x,y)','z=x^3-y^3')
Warning :redefining function: f


-->x=-3:0.2:3 ;y=x ;

-->clf() ;fplot3d(x,y,f)
```



```
-->deff('z=f(x,y)','z=sin(x^2)-y^2')
Warning :redefining function: f

-->x=-3:0.2:3 ;y=x ;

-->clf() ;fplot3d(x,y,f)
```



Use of scilab by various ways:

- Solve Linear System of Equations
- Find roots of higher degree polynomials
- Find eigen values
- Evaluate matrix and polynomials with complex numbers
- Draw 2D and 3D figures

15. Newton-Raphson Method

$f(x)$ and initial guess x_0 is given, find out 'zero' for $f(x)$.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Let $f(x) = \cos(x)$, and $x_0 = 10$

```
-->deff('[y]=f(x)','y=cos(x)');
```

```
-->deff('[y]=f1(x)','y=-sin(x)');
```

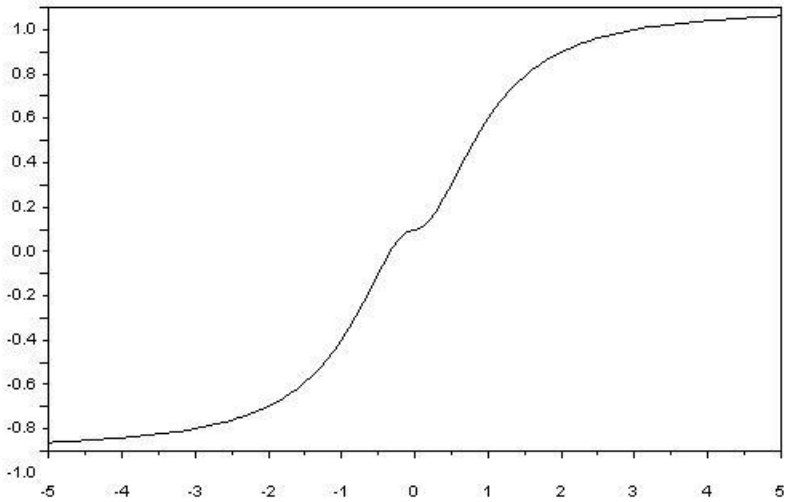
```
-->deff('[y]=g(x)','y=x-f(x)/f1(x)');
```

```
-->g(10)
 ans  =

    11.542351

-->g(ans)
 ans  =

    10.933672

-->g(ans)
 ans  =

    10.995653
```

```
-->g(ans)
 ans  =

    10.995574

-->g(ans)
 ans  =

    10.995574
```

```
    -->function [y]=f(x)
-->y=x*abs(x)/(1+x^2);
-->endfunction
```

```
-->x=(-5:0.1:5);

-->fplot2d(x,f)
```



Scilab Book Functions

- gausselimPP (also find inv(A)*B)
- CrossProd
- eigenvectors (clean using singular matrix)
- secant

Find the solution to the following linear system

$$\begin{aligned} 4x_1 + 8x_2 + 4x_3 &= 8 \\ x_1 + 5x_2 + 4x_3 + -3x_4 &= -4 \\ x_1 + 4x_2 + 7x_3 + 2x_4 &= 10 \\ x_1 + 3x_2 - 2x_4 &= -4 \end{aligned}$$

Use Gauss Elimination Method. (*Hint*: `gausselimPP`)

Program in Newton Raphson

To compare alogorithms which are time wise faster we can use function
timer()

Differential Equations:
solve and plot graph

```
-->function udot = f(t,u)
-->udot = cos(t)
-->endfunction

-->t=0:.1:10;

-->u=ode(0,0,t,f);
-->xbasc();plot2d(t,u)
```

16. Books on Scilab

Scilab for Computational Mathematics
Bhaskaracharya Pratishthana, 2010
Modeling and Simulation in Scilab/Scicos
Springer Publications
STEPHEN L. CAMPBELL
JEAN-PHILIPPE CHANCELIER
RAMINE NIKOUKHAH

==========

References:

1. `engr2200_lecture4scilab.txt`
2. `engre2200_ulecturescilab2.txt`
3. lectures from 1 to 5

==========

www.scilab.org

Thanks!