Scilab — Osnove in še kaj

Andrej Taranenko

Kazalo

1 Uvod

1.1 Kaj je Scilab?

Scilab je znanstveni programski paket za numerično računanje. Nudi močno programsko okolje za inženirske in znanstvene aplikacije. V razvoju je od leta 1990. Razvili so ga raziskovalci iz inštitutev INRIA in ENPC. Je prosto dostopen preko interneta (http://www-rocq.inria.fr/Scilab/). Uporablja se v izobraževalnih kot tudi industrijskih okoljih povsod po svetu.

1.2 Kako namestiti Scilab?

1.3 Zmožnosti Scilaba

Nekatere od zmožnosti Scilaba so:

1. 2D in 3D grafika, animacija,
2. linearna algebra, razpršene matrike,
3. polinomi, racionalne funkcije,
4. simulacija: reševanje navadnih diferencialnih enačb (ODEPACK),
5. statistika
6. in še mnoga druga področja.

Scilab deluje na mnogih operacijskih sistemih, kot so Unix, Linux, Windows 9X/NT/2000/XP. Vsebuje tudi izvorno kodo, on-line pomoč in angleški uporabniški vodič. Binarne verzije so na voljo.

2 Začetki dela s Scilabom

2.1 Zagon programa

Program zaženemo z dvoklikom na ikono programa. Odpreta se dve okni. V konzolnem oknu se izvaja pomožni program, ki ga Scilab uporablja, in ga moramo pustiti odprtega. V drugem oknu se pokaže vnosna vrstica, pred katero je simbol `-->`.

```
                    ==========
                    scilab-2.7.2
          Copyright (C) 1989-2003 INRIA/ENPC
                    ==========


Startup execution:
  loading initial environment

-->
```

Osnove dela s Scilabom si lahko ogledate tako, da kliknete na menu `Demos` in v oknu, ki se odpre, izberete `Introduction to Scilab`, nato sledite navodilom, ki se izpišejo na ekranu.

2.2 Menuji v SciLabu

File menu *Datoteka* nudi naslednje možnosti:

- **Getf** uporabite, da naložite funkcijo. Napišite *help getf* za več informacij.
- **Exec** uporabite, da zaženete skriptno datoteko. Napišite *help exec* za več informacij.
- **Save** uporabite, da shranite spremenljivko. Napišite *help save* za več informacij.
- **Load** uporabite, da naložite shranjeno spremenljivko. Napišite *help load* za več informacij.
- **Change Directory** uporabite, da spremenite delovno mapo.
- **Get Current Directory** uporabite, da izpišete delovno mapo.

Edit menu *Urejanje* nudi standardne možnosti podobnih menujev.

Controls menu nudi naslednje možnosti:

- **Restart** izbriše vse uporabniško definirane spremenljivke in inicializira spremenljivke okolja.
- **Pause** preklopi kontrolo v *pause* način. V glavnem uporabljamo za razhroščevanje funkcij v Scilabu.
- **Resume** nadaljuje z izvajanje, po pavzi.

Abort preneha z izvajanjem trenutnega programa.

Interupt prekine izvajanje trenutnega programa.

Demos nudi nekaj primerov uporabe Scilaba.

Graphic Window ... omogoča nadzor nad grafičnim oknom.

Help odpre okno s pomočjo.

Editor odpre urejevalnik, v katerem lažje urejamo daljše programe.

2.3 Spremenljivke in osnovne računske operacije

Scilab je *case sensitive*, kar pomeni, da loči velike in male črke. Tako sta spremenljivki a in A dve različni spremenljivki.

Primer:

```
-->a=1;

-->A=2;

-->a+A
 ans  =

    3.

-->//Več ukazov v isti vrstici

-->c=[1 2 3]; b=1.5
 b  =

    1.5

-->//Ukaz v večih vrsticah

-->u=10*(a*sin(A)^2)+...
-->     20*b*cos(a/A)+...
-->     30*a*b*cos(A)^2
 u  =

    42.388713
```

Na začetku spremenljivkama `a` in `A` določimo vrednosti. Znak `;` na koncu vnosa pomeni, da se rezultat ne izpiše. Več ukazov v isti vrstici ločimo s pomočjo `;`, kot je razvidno v primeru. Rezultat se je izpisal, ker za vrstico `b=1.5` ni `;`. S pomočjo simbola ... zapišemo ukaz v večih vrsticah. S simbolom `//` označimo komentarje.

S pomočjo ukaza `who` izpišemo vse definirane spremenljivke.

2.4 Konstante

V naprej definirane konstante v Scilabu se pričnejo z znakom `%`. Nekatere od teh konstant so:

- `%i` predstavlja $\sqrt{-1}$,
- `%pi` predstavlja $\pi = 3.1415926535\ldots$,
- `%e` predstavlja Eulerjevo število $e = 2.7182818284\ldots$,
- `%eps` predstavlja natančnost računalnika (`%eps` je največje število za katerega velja $1 + $ `%eps` $= 1$),
- `%inf` predstavlja neskončno (angl. *infinity*),
- `%nan` pomeni "ni število" (angl. *not a number*),
- `%t` logična vrednost *true* (`%t` je enako kot `1==1`),
- `%f` logična vrednost *false* (`%f` je enako kot `~%t`),
- `%s` je polinom $s =$ `poly(0,'s')`[1] s simbolom `s`.

2.5 Komentarji

Komentarje v Scilabu označimo z dvojno poševnico (//). Vse kar sledi // se smatra kot komentar.

[1] Splošno, z danim vektorjem `nicle`, ukaz `p=poly(nicle,'x')` definira polinom $p(x)$, katerega ničle so vrednosti v vekotrju `nicle`

2.6 Matrike

2.6.1 Vnašanje matrik

Začnimo z matriko, ki je tudi magični kvadrat:

$$D = \begin{bmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{bmatrix}$$

Matriko lahko vnesemo v Scilabu na naslednji način:

```
-->D = [16 3 2 13
-->5 10 11 8
-->9 6 7 12
-->4 15 14 1]
```

Scilab matriko izpiše kot:

```
 D  =

!   16.    3.    2.    13. !
!   5.     10.   11.   8.  !
!   9.     6.    7.    12. !
!   4.     15.   14.   1.  !
```

V splošnem matrike zapišemo med oglate oklepaje, kjer elemente ločimo s presledki, z novo vrstico pa ločimo vrstice v matriki. Vrstice lahko ločimo tudi s podpičjem:

```
-->D=[16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

2.6.2 Transponiranje

Enojni narekovaj, ', uporabljamo, da transponiramo matriko:

```
-->D'
 ans  =

!   16.    5.    9.    4.  !
!   3.     10.   6.    15. !
!   2.     11.   7.    14. !
!   13.    8.    12.   1.  !
```

Če je `D` kompleksna matrika, je `D'` konjugirana transponiranka `D`. Razlikovanje vrstičnih in stolpičnih vektorjev je v Scilabu zelo pomembno. Konjugiranje nam služi, da ene pretvorimo v druge:

```
-->v=[1 2 3 4]
 v  =

!   1.    2.    3.    4. !

-->w=v'
 w  =

!   1. !
!   2. !
!   3. !
!   4. !

-->x=w'
 x  =

!   1.    2.    3.    4. !
```

2.6.3 Vsote

Magični kvadrati imajo lastnost, da je vsota poljubne vrstice, stolpca ali diagonale vedno enaka:

```
-->sum(D, 'c')
 ans  =

!   34. !
!   34. !
!   34. !
!   34. !

-->sum(D,'r')
 ans  =

!   34.    34.    34.    34. !
```

V splošnem `sum(a,'c')` vrne stolpični vektor, keterega elementi so vsote posameznih *vrstic*, ukaz `sum(a,'r')` vrne vrstični vektor, katerega elementi so

vsote posameznih *stolpcev*. Ukaz `sum(a)` vrne vsoto vseh elementov. Diagonalo matrike dobimo s pomočjo ukaza `diag(a)`. Preverimo še vsoto elementov na diagonali.

```
-->diag(D)
 ans  =

!   16. !
!   10. !
!   7.  !
!   1.  !

-->sum(ans)
 ans  =

    34.
```

2.6.4 Indeksi

Element v i-ti vrstici in v j-tem stolpcu matrike `D` označimo z `D(i,j)`. Tako lahko vsoto druge diagonale matrike `D` izračunamo tudi tako:

```
-->D(1,4)+D(2,3)+D(3,2)+D(4,1)
 ans  =

    34.
```

Zapis z indeksi lahko uporabimo, da spremenimo posamezni element v matriki:

```
-->D(2,3)=1000
 D  =

!   16.    3.     2.       13. !
!   5.     10.    1000.    8.  !
!   9.     6.     7.       12. !
!   4.     15.    14.      1.  !

-->D(2,3)=11
 D  =

!   16.    3.     2.     13. !
!   5.     10.    11.    8.  !
```

```
!   9.      6.      7.      12. !
!   4.      15.     14.     1.  !
```

Pri vrstičnih ali stolpičnih vektorjih potrebujemo samo en indeks.

```
-->v=[1 2 3 4]
 v  =

!   1.    2.    3.    4. !

-->v(3)
 ans  =

    3.

-->v(3)=10
 v  =

!   1.    2.    10.    4. !
```

2.6.5 Operator dvopičje

operator dvopičje, `:`, ima v konstruiranju vektorjev in matrik veliko uporab. Najpreprostejša uporaba je, da dobimo vektor vseh željenih števil na nekem intervalu:

```
-->n=1:10
 n  =

!   1.    2.    3.    4.    5.    6.    7.    8.    9.    10. !

-->n=10:-2:0
 n  =

!   10.    8.    6.    4.    2.    0. !
```

Operator dvopičje lahko uporabljamo za ekstrahiranje delov matrik:

- `D(i,:)` je i-ta vrstica matrike `D`,
- `D(:,j)` je j-ti stolpec matrike `D`,
- `D(:,j:k)` je matrika vseh stolpcev matrike `D` od j-tega do k-tega,

- `D(i:j,k:l)` je podmatrika z vsemi elementi od (i,k) do (j,l), itd.

Sledi nekaj primerov:

```
-->D(2,:)
 ans  =

!   5.    10.    11.    8. !

-->D(:,3)
 ans  =

!   2.  !
!   11. !
!   7.  !
!   14. !

-->D(:,2:3)
 ans  =

!   3.     2.  !
!   10.    11. !
!   6.     7.  !
!   15.    14. !

-->D(1:2,3:4)
 ans  =

!   2.     13. !
!   11.    8.  !
```

V zadnjem primeru smo izpisali podmatriko sestavljeno iz prve in druge vrstice ter tretjega in četrtega stolpca teh vrstic matrike `D`.

3 Izrazi

3.1 Še enkrat spremenljivke

Imena spremenljivk v Scilabu so sestavljena iz črke, sledi ji poljubno število črk, števk ali podpičij. Imena so občutljiva na velikost velikih in malih znakov, zato sta `D` in `d` različni spremenljivki. Da izpišete vrednost, ki je neki spremenljivki prirejena, enostavno vnesete ime spremenljivke:

```
-->D
 D  =

!   16.    3.     2.     13. !
!   5.     10.    11.    8.  !
!   9.     6.     7.     12. !
!   4.     15.    14.    1.  !

-->d
  !--error     4
undefined variable : d


-->d=D
 d  =

!   16.    3.     2.     13. !
!   5.     10.    11.    8.  !
!   9.     6.     7.     12. !
!   4.     15.    14.    1.  !

-->d
 d  =

!   16.    3.     2.     13. !
!   5.     10.    11.    8.  !
!   9.     6.     7.     12. !
!   4.     15.    14.    1.  !
```

3.2 Števila

Scilab pozna realna in kompleksna števila. Znanstveni zapis uporablja `e`, da zapiše potenco 10, tako se število 1.234×10^{-45} zapiše kot `1.234e-45`. Kompleksna števila za zapis uporabljajo `%i`, tako $2.3 - 6.7i$ zapišemo kot `2.3 - 4.5 * %i`

3.3 Operatorji

Aritmetični operatorji uporabljeni v Scilabu so:

- `+` seštevanje

- `-` odštevanje
- `*` množenje
- `/` deljenje
- `^` potenciranje
- `'` konjugiranje

Vse te operatorje lahko uporabljamo tudi z vektorji in matrikamo. Seštevanje, odštevanje, množenje in potenciranje delujejo kot pričakovano:

```
-->D+D
 ans  =

!   32.    6.    4.    26. !
!   10.    20.   22.   16. !
!   18.    12.   14.   24. !
!   8.     30.   28.   2.  !

-->D-D
 ans  =

!   0.    0.    0.    0. !
!   0.    0.    0.    0. !
!   0.    0.    0.    0. !
!   0.    0.    0.    0. !

-->D*D
 ans  =

!   341.    285.    261.    269. !
!   261.    301.    309.    285. !
!   285.    309.    301.    261. !
!   269.    261.    285.    341. !

-->D^3
 ans  =

!   10306.    9474.    9410.    10114. !
!   9602.     9922.    9986.    9794.  !
!   9858.     9666.    9730.    10050. !
```

```
!   9538.     10242.    10178.    9346.   !
```

Operator deljenja / se uporablja pri reševanju linearnih enačb in ga bomo opisali kasneje. Kot že omenjeno, se operator ', da dobimo konjugirano transponiranko neke matrike:

```
-->C=[1+2*%i 3-2*%i
-->0 2-%i]
 C  =

!   1. + 2.i    3. - 2.i !
!   0           2. - i   !

-->C
 C  =

!   1. + 2.i    3. - 2.i !
!   0           2. - i   !

-->C'
 ans  =

!   1. - 2.i    0        !
!   3. + 2.i    2. + i   !
```

Pri množenju matrik in vektorjev je potrebno upoštevati pravila za množenje:

```
-->v=[1 2 3 4]
 v  =

!   1.    2.    3.    4. !

-->w=v'
 w  =

!   1. !
!   2. !
!   3. !
!   4. !

-->D*w
```

```
 ans  =

!   80. !
!   90. !
!   90. !
!   80. !

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


-->v*w
 ans  =

    30.

-->w*v
 ans  =

!   1.    2.    3.     4.  !
!   2.    4.    6.     8.  !
!   3.    6.    9.     12. !
!   4.    8.    12.    16. !

-->v*D
 ans  =

!   69.    101.    101.    69. !

-->w*D
    !--error    10
inconsistent multiplication
```

3.4 Operator pika

Operator pika . se uporablja skupaj z operatorji *, / in ^, da izvedemo operacije nad elementi vektorjev in matrik. Na primer:

```
-->v.*v
 ans  =
```

```
!   1.    4.    9.    16. !

-->v.^3
 ans  =

!   1.    8.    27.    64. !

-->D.*D
 ans  =

!   256.    9.      4.      169. !
!   25.     100.    121.    64.  !
!   81.     36.     49.     144. !
!   16.     225.    196.    1.   !

-->D.^3
 ans  =

!   4096.    27.      8.       2197. !
!   125.     1000.    1331.    512.  !
!   729.     216.     343.     1728. !
!   64.      3375.    2744.    1.    !

-->D./D
 ans  =

!   1.    1.    1.    1. !
!   1.    1.    1.    1. !
!   1.    1.    1.    1. !
!   1.    1.    1.    1. !
```

Pomembne razlike so torej:

1. `D*D` je matrični produkt matrike `D` same s sabo, `D.*D` je matrika, katere elementi so kvadrati elementov matrike `D`,

2. `D^3` je kub matrike `D`, `D.^3` je matrika katere elementi so kubi posameznih elementov matrike `D`.

3. `D./D` deli vsak element matrike `D` s samim seboj.

3.5 Funkcije

Scilab ima veliko število vgrajenih funkcij, npr. `srqt, sin, exp`, itd. Ko jih uporabimo nad vektorji ali matrikami, se (običajno) izvedejo nad posameznimi elementi:

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

!   1.    1.4142136    1.7320508    2. !

-->sqrt(D)
 ans  =

!   4.          1.7320508    1.4142136    3.6055513 !
!   2.236068    3.1622777    3.3166248    2.8284271 !
!   3.          2.4494897    2.6457513    3.4641016 !
!   2.          3.8729833    3.7416574    1.        !

-->sin(D)
 ans  =

! -  .2879033     .1411200     .9092974     .4201670 !
! -  .9589243  -  .5440211  -  .9999902     .9893582 !
!    .4121185  -  .2794155     .6569866  -  .5365729 !
! -  .7568025     .6502878     .9906074     .8414710 !

-->exp(D)
 ans  =

!   8886110.5    20.085537    7.3890561    442413.39 !
!   148.41316    22026.466    59874.142    2980.958  !
!   8103.0839    403.42879    1096.6332    162754.79 !
!   54.59815     3269017.4    1202604.3    2.7182818 !
```

Naštejmo še nekatere vgrajene funkcije:

- Osnovne funkcije: `sum, prod, sqrt, diag, cos, max, round, sign, fft`
- Urejanje: `sort, gsort, find`
- Posebne matrike: `zeros, eye, ones, matrix, empty`

- Linearna algebra: `det, inv, qr, svd, bdiag, spec, schur`
- Polinomi: `poly, roots, coeff, horner, clean, freq`
- Linearni sistemi: `syslin`
- Naključna števila: `rand`
- Programiranje: `function, deff, argn, for, if, end, while, select, warning, error, break, return`
- Simboli primerjanja: ==, >=, >, =, & (and),| (or)
- Izvajanje datoteke: `exec`
- Razhroščevanje: `pause, return, abort`
- Interpolacija, zlepki: `splin, interp, interpln`
- Nizi znakov: `string, part, evstr, execstr`
- Delo z grafika: `plot, xset, driver, plot2d, xgrid, locate, plot3d, Graphics`
- Navadne diferencialne enačbe: `ode, dassl, dassrt, odedc`
- Optimizacija: `optim, quapro, linpro, lmitool`
- Povezani dinamični sistemi: `scicos`
- Dodajanje C ali Fortran rutin: `link, fort, addinter, intersci`

4 Preprosti grafi

Najpreprostejši graf prejme kot parameter dva vektorja, pri katerem so v prvem vektorju vrednosti neodvisne spremenljivke, v drugem pa funkcijske vrednosti:

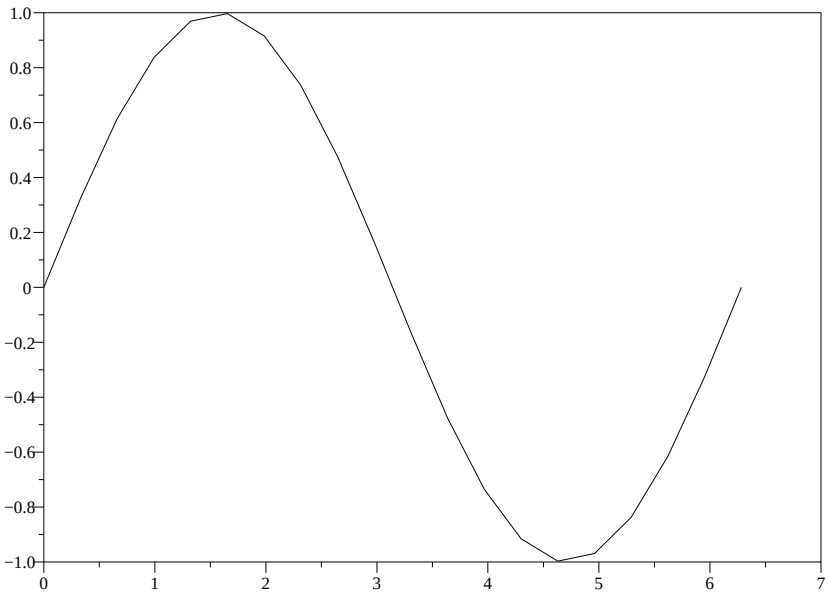
```
-->x=linspace(0, 2*%pi, 20)
 x  =

         column 1 to 5

!   0.     .3306940     .6613879     .9920819    1.3227759 !

         column  6 to 10
```

```
!   1.6534698    1.9841638    2.3148577    2.6455517    2.9762457 !

         column 11 to 15

!   3.3069396    3.6376336    3.9683276    4.2990215    4.6297155 !

         column 16 to 20

!   4.9604095    5.2911034    5.6217974    5.9524913    6.2831853 !

-->y=sin(x)
 y  =


         column 1 to 5

!   0.     .3246995     .6142127     .8371665     .9694003 !

         column  6 to 10

!    .9965845     .9157733     .7357239     .4759474     .1645946 !

         column 11 to 15

! -  .1645946  -  .4759474  -  .7357239  -  .9157733  -  .9965845 !

         column 16 to 20

! -  .9694003  -  .8371665  -  .6142127  -  .3246995  - 2.449E-16 !

-->plot2d(x,y)
```

Opombe:

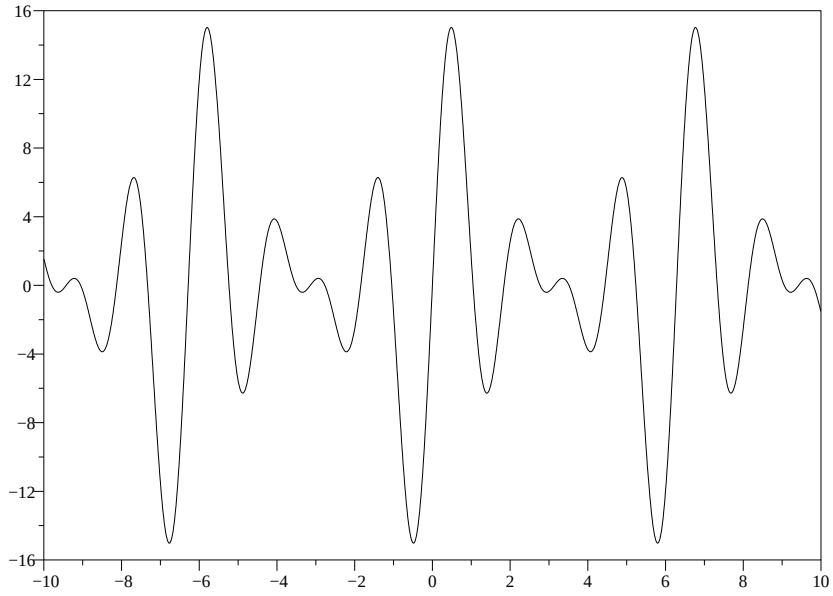
1. `linspace(0, 2*%pi, 20)` ustvari 20 ekvidistančnih točk z intervala $[0, 2\pi]$ in jih vrne kot stolpični vektor.
2. π v Scilabu zapišemo z `%pi`.
3. Grafi v Scilabu povežejo točke z ravnimi črtami, zaradi česar graf izgleda poligonsko.

Še en primer:



Slika 1: Slika, kot jo izriše Scilab

```
-->x=linspace(-10,10,1000);

-->y=2*sin(x) + 3*sin(2*x)+7*sin(3*x)+5*sin(4*x);

-->plot2d(x,y)
```



Slika 2: Graf, ki ga izrišejo ukazi v drugem primeru

Opombe:

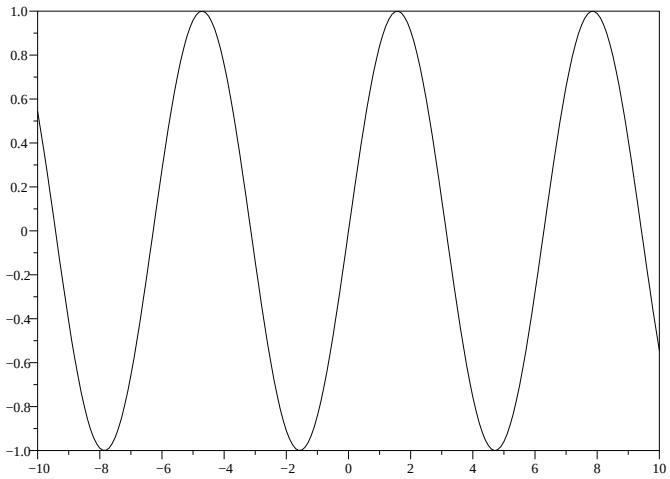
1. Podpičje na koncu ukaza prepreči izpis rezultata (nočemo izpisati vektorja s 1000 elementi).

2. Običajno želimo obstoječi graf, preden narišemo novega, izbrisati. To storimo preko menuja `File/Clear`, v oknu grafa.

Vektorji, ki jih podamo kot parametre, so lahko vrstični ali stolpični, zaželjena je uporaba stolpičnih, ker jih uporabljamo pri risanju večih krivulj.

```
-->x=(-10:.1:10)';
```

```
-->y=sin(x);

-->plot2d(x,y)
```



Slika 3: Rezultat zadnjega primera

Narišemo lahko tudi samo en vektor, ki se izriše glede na $1, 2, \ldots, n$, kjer je n dolžina vektorja:

```
-->plot2d(y)
```

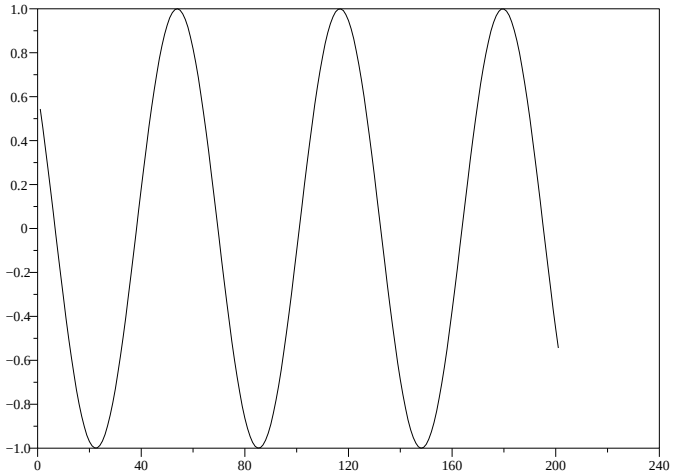
Bodite pozorni ne spremembo območja x-osi.

Grafi, ki smo si jih do sedaj ogledali, so bili zvezne krivulje. S pomočjo opcije `style` ukaza `plot2d` lahko podatke izrišemo kot točke. Negativne vrednosti stilov ustrezajo različnim tipom točk, pozitivne pa pomenijo različne barve. Z ukazom `xset()` odprete okno z izbiro različnih stilov in barv, ter drugih stvari, kot je debelina črt, itd.

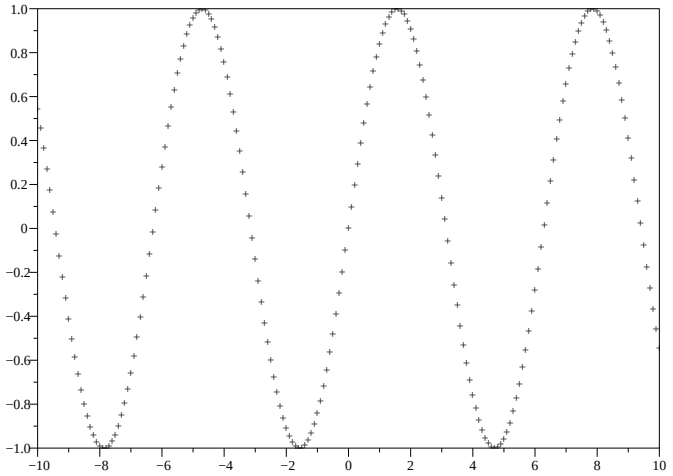
```
-->plot2d(x,y, style = -1)
```

4.1 Več krivulj hkrati

Več krivulj lahko hkrati izrišemo na en graf tako, da ne izbrišemo ekrana. Če je v ukazu `plot2d(x,y)` `y` matrika, potem se vsak stolpec v matriki izriše



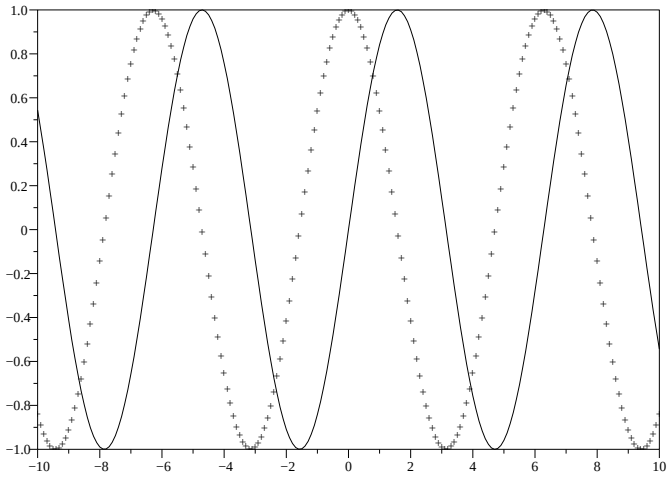
Slika 4: Rezultat zadnjega primera



Slika 5: Rezultat primera s spreminjanjem stilov

kot krivulja zase. V tem primeru *mora* `x` biti stolpični vektor. Krivulje lahko izrišemo v različnih stilih tako, da je `style` vektor stilskih števil.

```
-->plot2d(x, [sin(x) cos(x)], style = [1 -1])
```



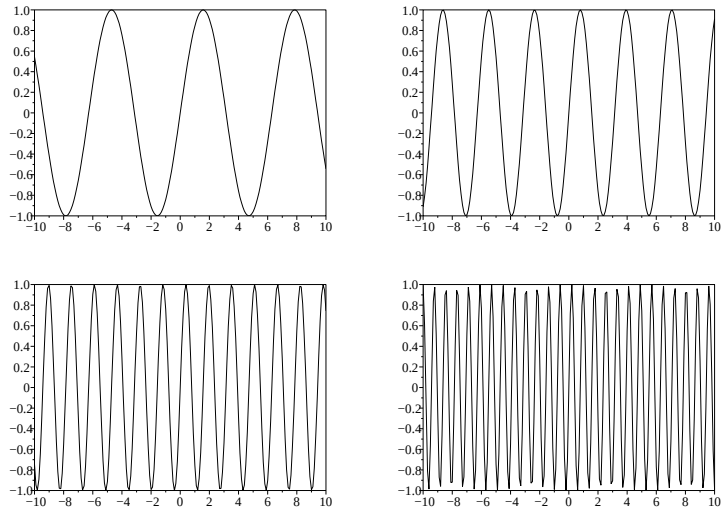
Slika 6: Rezultat primera z dvema krivuljama

4.2 Več grafov hkrati

Več ločenih grafov lahko hkrati izrišemo na eno sliko s pomočjo ukaza `subplot`.

```
-->subplot(2, 2, 1)

-->plot2d(x, sin(x))

-->subplot(2, 2, 2)

-->plot2d(x, sin(2*x))

-->subplot(2, 2, 3)

-->plot2d(x, sin(4*x))
```

```
-->subplot(2, 2, 4)

-->plot2d(x, sin(8*x))
```



Slika 7: Več grafov na eni sliki

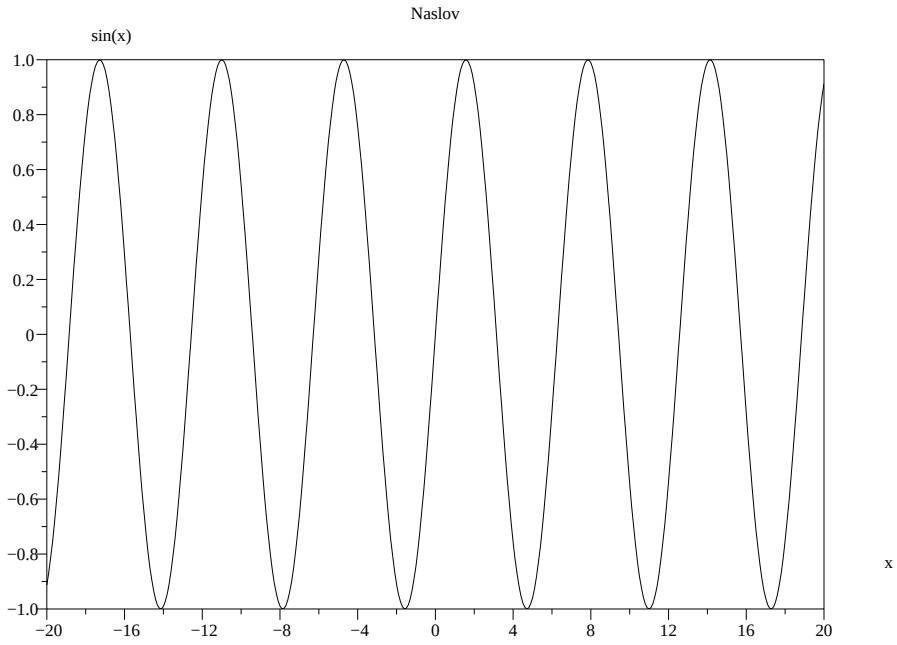
4.3 Naslovi in oznake

Naslove in oznake lahko dodamo po tem, ko je graf že narisan, s pomočjo ukaza `xtitle`. Ukaz prejme tri parametre: naslov grafa, oznaka x-osi in oznaka y-osi. Če naslova ne potrebujete, uporabite prazen niz, t.j. `''`. Glede pozicij oznak, ne moremo narediti ničesar.

```
-->x = (-20:.1:20)';

-->plot2d(x, sin(x))

-->xtitle('Naslov', 'x', 'sin(x)')
```

4.4 Druge možnosti

V Scilabu lahko z 2D grafi delate še veliko stvari. Glejte vodnike na domači strani Scilaba.

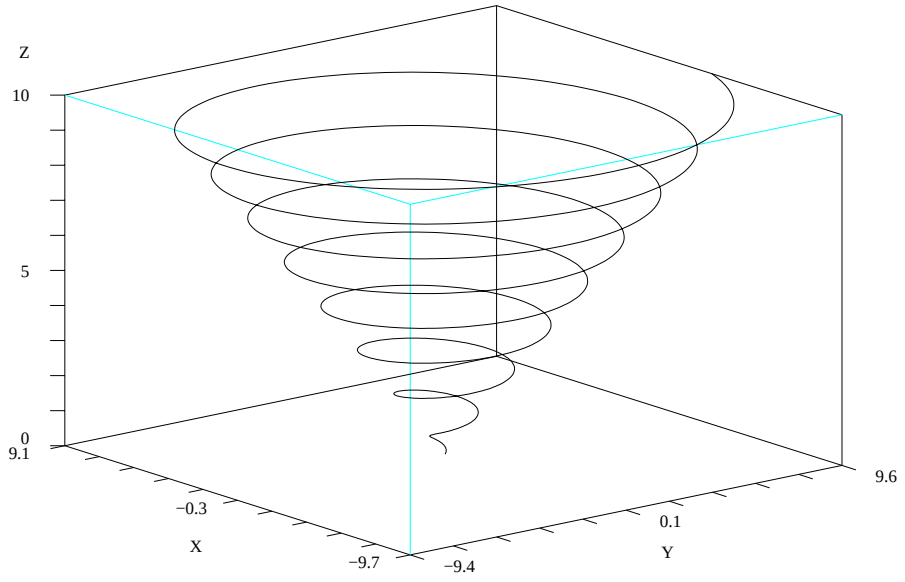


Slika 8: Dodajanje naslova in oznak

5 3D grafi

5.1 3D krivulje

Krivulje v 3 dimenzionalnem prostoru lahko rišemo s pomočjo ukaza `param3d`. Za to potrebujemo tri vektorje, ki vsebujejo koordinate x, y in z točk na krivulji. S klikom na `3D Rot.` menu v grafičnem oknu in premikanjem miške, lahko spremenite pogled na graf.



Slika 9: Primer 3D krivulje

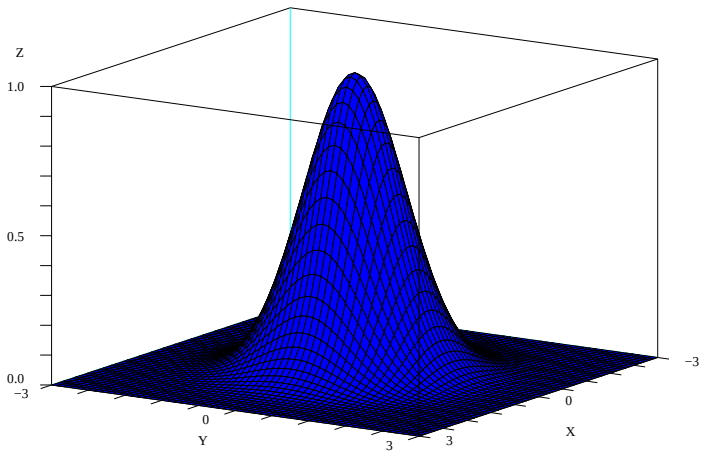
5.2 3D ploskve

Osnovni ukaz za risanje 3D ploskev je `plot3d(x, y, z)`. Tukaj sta `x` in `y` vektorja x in y koordinat dolžin n_1 in n_2, `z` je $n_1 \times n_2$ matrika z vrednosti. Narisali bomo graf funkcije

$$z = \mathrm{e}^{(x^2+y^2)}$$

```
-->x = (-3:.1:3)';
```

```
-->y = x;

-->z = zeros(61, 61);

-->for i = 1:61
-->  for j = 1:61
-->    z(i,j) = exp(-x(i)^2 - y(j)^2);
-->  end
-->end

-->plot3d(x, y, z)
```



Slika 10: Primer 3D ploskve

6 Programiranje v Scilabu

6.1 Delo z datotekami

Uporabljali bomo vgrajeni urejevalnik datotek, do katerega lahko dostopamo preko menuja `Editor` v glavnem oknu programa. Poglavitna težava, ki se pojavlja je, da ne povemo, kje se datoteke nahajajo. Scilab uporablja ukaz `exec` za zagon skriptne datoteke in `loadf`, da naloži funkcijsko datoteko. To lahko uporabimo na dva načina:

1. Preko menuja `Datoteka` in podmeuja `exec` in `load`. Odpre se pogovorno okno, kjer poiščemo mapo datoteke.

2. Ta možnost zahteva poznavanje strukture map v Windows okoljih. Z uporabo ukaza `chdir` spremenite trenutno delovno mapo v tisto, kamor shranjujete datoteke.

6.2 Skriptne datoteke

Skriptne datoteke so datoteke, ki vsebujejo Scilabove ukaze. Ti se izvedejo, kot da bi jih direktno tipkali v Scilab.

Odprite Scilab urejevalnik datotek, vnesite spodnje ukaze:

```
x = 0.995 : 0.0001 : 1.005;
y1 = (x - 1).^6;
y2 = x.^6 - 6*x.^5 + 15*x.^4 -20*x.^3 + 15*x.^2 - 6*x + 1;
plot2d(x, y1)
plot2d(x, y2)
```

Shranite datoteko z imenom `test.sce` v svojo delovno mapo in zaprite urejevalnik besedil. V galvnem oknu izberite menu `File/Exec`, poiščite in izberite prej shranjeno datoteko. Ukazi se izvedejo, kot bi jih posamezno vnašali v urejevalnik.

Skriptne datoteke se uporabljajo, kadar želimo ponavljati zaporedje ukazov, da izvedemo določene poskuse. Skripto lahko naredimo tudi bolj fleksibilno, recimo da spremenimo razpon med števili.

V datoteki `test.sce` spremenimo prvo vrstico:

```
x = 0.995 : n : 1.005;
y1 = (x - 1).^6;
y2 = x.^6 - 6*x.^5 + 15*x.^4 -20*x.^3 + 15*x.^2 - 6*x + 1;
plot2d(x, y1)
plot2d(x, y2)
```

Spremenljivka `n` še nima dodeljene vrednosti, zato moramo to storiti, preden znova zaženemo skripto, npr:

```
-->n=0.001;

-->exec('test.sce');

-->n=0.00001;

-->exec('test.sce');
```

6.3 For zanka

Začnimo s preprostim primerom `for` zanke:

```
-->v = zeros(1, 10)
 v  =

!   0.    0.    0.    0.    0.    0.    0.    0.    0.    0. !

-->for i = 1:10
-->  v(i) = i;
-->end

-->v
 v  =

!   1.    2.    3.    4.    5.    6.    7.    8.    9.    10. !
```

V zankah običajno vse stavke zaključimo s podpičjem. Poskusite:

```
-->v = zeros(1, 10);

-->for i = 1:10
-->  v(i) = i
-->end
```

Ukaz `eye(5,5)` ustvari identično matriko dimenzije 5×5. Enako lahko storimo s `for` zanko na naslednji način:

```
-->ident=zeros(5,5);

-->for i=1:5
-->  ident(i,i)=1;
-->end

-->ident
 ident  =

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

V Scilabu lahko zanke seveda tudi gnezdimo. Skonstruirajmo Hilbertovo matriko H_5. Hilbertova matrika, ja matrika naslednje oblike:

$$H_n = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix}$$

Element na koordinatah (i, j) ima vrednost $1/(i + j - 1)$. V Scilabu to naredimo tako:

```
-->h5=zeros(5,5);

-->for i = 1:5
-->  for j = 1:5
-->    h5(i,j) = 1/(i+j-1);
-->  end
-->end

-->h5
 h5  =

!   1.            .5            .3333333      .25           .2          !
!    .5           .3333333      .25           .2           .1666667    !
!    .3333333     .25           .2            .1666667     .1428571    !
!    .25          .2            .1666667      .1428571     .125        !
!    .2           .1666667      .1428571      .125         .1111111    !
```

Zamiki za poravnavo `for` zanke z ustreznim `end` ukazom, omogočajo lažje branje programov.

6.4 Funkcije

Da ustvarimo Hilbertovo matriko H_n za poljuben n, je najbolje definirati funkcijo:

```
-->function h = hilbert(n)
-->  h = zeros(n,n)
-->  for i = 1:n
-->    for j = 1:n
-->      h(i,j) = 1/(i + j -1)
```

```
-->    end
-->  end
-->endfunction

-->hilbert(5)
 ans  =

!   1.            .5           .3333333     .25          .2          !
!    .5           .3333333     .25          .2           .1666667   !
!    .3333333     .25          .2           .1666667     .1428571   !
!    .25          .2           .1666667     .1428571     .125       !
!    .2           .1666667     .1428571     .125         .1111111   !

-->hilbert(3)
 ans  =

!   1.            .5           .3333333  !
!    .5           .3333333     .25       !
!    .3333333     .25          .2        !
```

V tem primeru je:

1. `hilbert` ime funkcije,
2. `n` je parameter funkcije. Funkcije lahko imajo poljubno število parametrov,
3. `h` je vrednost, ki jo funkcija vrne. Ta vrednost je tista, ki jo ima spremenljivka `h`, tik preden se izvede ukaz `endfunction`.

6.5 Funkcijske datoteke

Funkcijske datoteke vsebujejo eno ali več funkcijskih definicij, kot je v prejšnjem primeru funkcija `hilbert`. Običajno funkcije pišemo v datoteke, ker (a) jih shranimo za nadaljno uporabo, in (b) jih enostavno spremenimo in popravimo.

Ustvarite datoteko `hilbert.sci` (funkcije se običajno shranjujejo v datoteke s pripono `.sci`), ki vsebuje gornjo funkcijo:

```
function h = hilbert(n)
  h = zeros(n,n)
  for i = 1:n
    for j = 1:n
```

```
      h(i,j) = 1/(i + j -1)
    end
  end
endfunction
```

Funkcijo naložimo preko menuja `File / getf` ali z ukazom `getf`.

6.6 While zanka

`for` zanka ponavlja zaporedje stavkov določeno število krat. `while` zanka ponavlja zaporedje stavkov, dokler je dani pogoj izpolnjen.

Naslednji primer pokaže omejeno natančnost računalniške aritmetike; začnemo z `eps = 1` in ga ponavljajoč razpolavljamo, dokler ne velja `1 + eps = 1`. `~=` se v Scilabu uporablja za "ni enako".

```
-->eps = 1;

-->while (1 + eps ~= 1)
-->  eps = eps/2;
-->end

-->eps
 eps  =

    1.110E-16

-->(1 + eps) - 1
 ans  =

    0.

-->(1 + 2*eps) - 1
 ans  =

    2.220E-16

-->%eps
 %eps  =

    2.220E-16
```

`%eps` je v Scilabu *strojni epsilon* ε_{stroj}, t.j. razlika med 1 in naslednjim večjim predstavljivim številom. Drugače: ε_{stroj} je najmanjše število, za katerega še velja $1 + \varepsilon_{stroj} \neq 1$.

6.7 If stavek

Pogojni if stavek omogoča, da izvedemo različne stavke, glede na pogoj. Primer funkcije, ki vrne predznak števila:

```
-->function s = signum(x)
-->  if (x > 0)
-->    s = 1
-->  elseif (x < 0)
-->    s = -1
-->  else
-->    s = 0
-->  end
-->endfunction

-->signum(12345)
 ans  =

    1.

-->signum(-12345)
 ans  =

  - 1.

-->signum(0)
 ans  =

    0.
```