Ján Pribiš

SCILAB

Táto publikácia vznikla s prispením grantovej agentúry SR KEGA v tematickej oblasti „Nové technológie vo výučbe“– projekt: 3/2158/04 – „Využitie OPENSOURCE softvéru vo výučbe na vysokých školách“.

Recenzovali: Ján Buša
Ladislav Ševčovič

ISBN 80-8073-655-3

Sadzba programom pdfTEX

Obsah

Úvod

Táto knižka je jednou zo série učebníc, ktoré vznikli v rámci projektu KEGA o využití Open Source softvéru vo výučbe. Jej cieľom je oboznámiť čitateľa s programom Scilab (Scientific Laboratory). Ostatné učebnice je možné nájsť na stránke http://people.tuke.sk/jan.busa/kega/.

Scilab bol vyvinutý v Scilab Group INRIA-Rocquencourt Metalau Project a je to integrované prostredie na 2D a 3D grafiku, matematické a technické výpočty, modelovanie a simuláciu, spracovanie signálov, analýzu a vizualizáciu dát a množstvo iného... V prvom priblížení je to nekomerčná verzia balíka MATLAB. Je voľne šíriteľný a spolu s dokumentáciou v anglickom jazyku ho môžeme stiahnuť z domovskej stránky http://www.scilab.org.

Pokračovaním balíka Scilab je Scilab//, ktorý umožňuje paralelné výpočty a obsahuje rozhranie pre paralelné knižnice (PBLAS) (Parallel Basic Linear Algebra Subprograms) a (ScaLAPACK) (Scalable Linear Algebra PACKage).

Scilab sa skladá z troch hlavných častí: interpretačného prekladača, knižnice funkcií (Scilab procedúry) a knižnice prekladača programov z jazykov Fortran a C. Tieto programy (ktoré, presnejšie povedané, nepatria k Scilabu, ale sú interaktívne volané Scilabom) sú nezávislé a väčšina z nich je prístupná cez Netlib (http://www.netlib.org). Niektoré z nich boli trocha modifikované pre lepšiu kompatibilitu so Scilabovským prekladačom.

Základné charakteristiky Scilabu:

- Je voľne šíriteľný.
- Existujú verzie pre operačný systém UNIX (vrátane Linux) aj Windows.
- Obsahuje systém pomoci (help).
- Obsahuje algoritmy na riešenie úloh lineárnej algebry a algebry polynómov.
- Obsahuje algoritmy numerickej matematiky.
- Existuje v ňom možnosť programovania.
- Existuje možnosť pracovať nielen v číselnej, ale aj v symbolickej forme.
- Existuje možnosť práce s grafikou.
- Existuje možnosť použitia skompilovaných funkcií v jazykoch C a Fortran.

Scilab je svojou štruktúrou a možnosťami veľmi podobný MATLABu. Obsahuje aj prekladač MATLABovských príkazov (`M2SCI_tools`) a `m-súbory` sa dajú spúšťať priamo v prostredí Scilab pomocou menu `Import Matlab files`. Aj keď moja vlastná skúsenosť s verziou Scilab 3.1.1 z roku 2005 ale aj s poslednou verziou 4.0 z roku 2006 je taká, že nie všetky príkazy sa prekladajú korektne a nezaobíde sa to bez zásahu užívateľa. To však v žiadnom prípade nemá byť kritikou Scilabu a skôr radou, že je asi lepšie

vykonať preklad samostatne s použitím výkladu, ktorý je možné pozrieť v dokumentácii v `Matlab/Scilab functions` (MATLAB/SCILAB FUNCTIONS, 1989–2006).

Taktiež boli zaznamenané niektoré zmeny v rôznych verziách Scilabu. Úplne viditeľné sú napríklad zmeny (k „lepšiemu“) v hlavnom menu pracovného okna, ako aj v menu iných častí, napr. editora, grafického okna a pod., čo len potvrdzuje jeho aktívne zdokonaľovanie.

Dokumentácia Scilabu je veľmi rozsiahla a vyžaduje si veľa času na preštudovanie. Táto učebnica by mohla pomôcť začínajúcim užívateľom rýchlejšie si osvojiť základy práce s týmto balíkom. Pri jej písaní bol za osnovu vzatý úvod do Scilabu z dokumentácie. Nekopíruje však tento úvod doslovne, niektoré časti sú rozšírené, niektoré skrátené. Použili sme tiež príručku (PAVLOVA, M. I., 2003).

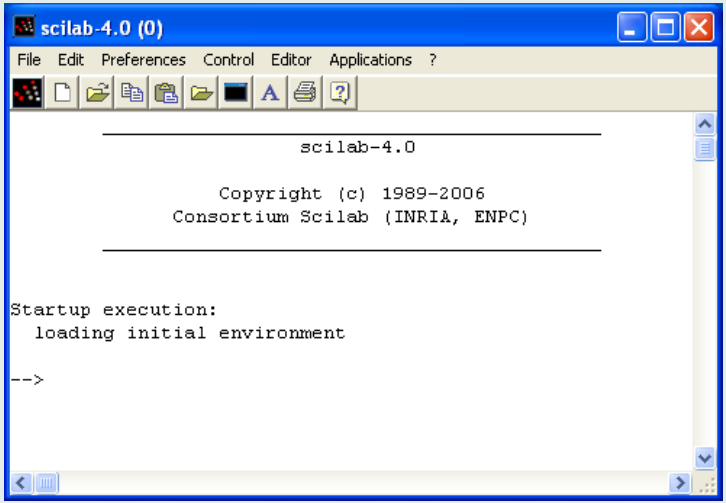
Scilab obsahuje celkovo 1021 príkazov. Ich úplný opis sa nachádza v dokumentácii v súbore `manual.pdf`. Tento dokument prakticky kopíruje on-line dokumentáciu, ktorá je k dispozícii na domovskej stránke (SCILAB, ON-LINE HELP, 1989–2006). Syntax použitých príkazov a funkcií tiež nie je podaný vyčerpávajúco, pretože niektoré z nich obsahujú veľké množstvo nepovinných parametrov. Snažili sme sa skôr, pokiaľ to bolo možné, podať ich stručný opis a ukázať ich na demonštračných príkladoch.

Na tomto mieste sa ešte chcem poďakovať recenzentom Jánovi Bušovi a Ladislavovi Ševčovičovi za pozorné prečítanie rukopisu, ako aj za cenné rady a námety pri písaní tejto učebnice.

Košice 17. 10. 2006 Ján Pribiš

1. Prvé kroky

Po úspešnej inštalácii a spustení súboru `\Scilab\bin\WScilex.exe` sa otvorí hlavné pracovné okno s príkazovým riadkom, ktoré vyzerá nasledovne



V rôznych operačných systémoch, ako aj v rôznych verziách Scilabu, sú rozdiely v hlavnom menu pracovného okna.

Výstup z prostredia Scilab je moňé vykonať pomocou príkazov `exit` alebo `quit` a stlačením klávesu `Enter` alebo pomocou hlavného menu `File/Exit`.

Scilab rozlišuje veľké a malé písmená, a to nielen v operačnom systéme Linux, ale aj v OS Windows.

1.1. Demos a help

Demonštračné príklady je možné pozrieť pomocou položky ?/Scilab demos horného menu a postupovať podľa pokynov. Príklady sa budú otvárať v novom okne, ktoré je potrebné po skončení zatvoriť.

Systém pomoci môžeme otvoriť v položke ?/Scilab help hlavného menu alebo pomocou klávesu F1. Otvorí sa tak nové okno Scilab Browse Help, kde si môžeme vybrať podľa tém, resp. podľa abecedného zoznamu. Ak chceme zistiť úplnú syntax konkrétneho príkazu (máme na mysli funkciu Scilabu), je možné otvoriť pomoc priamo z dialógového riadku pomocou help príkaz. Otvorí sa nové okno s popisom danej funkcie, jej syntaxou, popisom parametrov, výkladom a vo väčšine prípadov aj s demonštračným príkladom.

1.2. Aktuálny priečinok

Názov a umiestnenie aktuálneho pracovného priečinku je možné zistiť pomocou príkazu getcwd() alebo pomocou hlavného menu File/Get Current Directory. Napr.:

```
-->getcwd()
 ans =
D:\scilab\bin
```

Zmeniť aktuálny priečinok môžeme pomocou príkazu ierr=chdir('cesta/názov'), kde ierr je celé číslo rovné 1 ak príkaz nedokáže zmeniť prie-

činok (chyba) a 0 v opačnom prípade, `'cesta/názov'` je cesta a meno nového priečinku. Tak isto je to možné vykonať v hlavnom menu v ponuke `File/Change Directory` a nastaviť sa do požadovaného priečinku.

1.3. Zadávanie príkazov

Príkazy sa zadávajú v dialógovom riadku. Scilab sa tak môže veľmi jednoduchým spôsobom použiť ako kalkulačka. Príkaz, ktorý napíšeme, sa vykoná po stlačení klávesu `Enter`. Ak príkaz v dialógovom riadku nie je ukončený bodkočiarkou, výsledok sa zobrazí ihneď po jeho vykonaní, ak je ukončený bodkočiarkou, výsledok sa nezobrazí. V jednom riadku je možné zadávať aj viac príkazov tak, že ich oddelíme bodkočiarkou (ak nechceme bezprostredne vidieť výsledok) alebo čiarkou. Napr.:

```
-->2+3^2
 ans =
    11.
-->a=1;b=7;
-->x=a+b
 x =
   8.
```

Premenné môžeme zadávať aj pomocou príkazu `input`. Jeho syntax je `[x]=input(správa,["string"])`, kde `x` je reťazec, ak je použitý parameter `"string"` a číslo, ak tento parameter nepoužijeme. Napr.:

```
-->x=input("Ako sa voláš?","string")
Ako sa voláš?-->Ján
 x  =
 Ján
-->x=input("Koľko máš rokov?")
Koľko máš rokov?-->34
 x  =
    34.
```

Niekedy je výhodné zadávať premenné alebo funkcie interaktívne v dialógovom režime. Na tieto účely slúžia príkazy `x_dialog`, `x_choose`, `x_matrix`, `x_message`, `x_mesage_modeless`, ktoré je možné podrobne pozrieť pomocou `help príkaz`.

Ak je príkaz veľmi dlhý, je možné ho rozdeliť a preniesť do druhého riadku pomocou dvoch bodiek, napr.:

```
-->plotframe(hranice,značky,[%t,%t],..
-->["My plot with grids and automatic bounds","x","y"])
```

Komentáre sa v Scilabe vykonávajú pomocou dvoch lomiek //, napr.:

```
-->z=3456; // to je moje číslo
```

Vymazanie aktuálnych premenných z pamäte sa vykonáva pomocou príkazu `clear`. Tento príkaz vymaže všetky, aj globálne premenné (okrem tých, ktoré sú v Scilabe preddefinované). Ak nechceme vymazať všetky premenné, len niektoré, použijeme `clear premenná1␣premenná2␣...␣premennán`. Na vymazanie iba globálnych premenných slúži príkaz `clearglobal`, ktorý je podrobnejšie popísaný v oddiele 3.6.

Vykonané príkazy môžeme v dialógovom riadku opätovne zavolať postupným stláčaním klávesu ↑ alebo v hlavnom menu `Edit/History` a vybrať príslušnú položku. Históriu vykonaných príkazov je možné vymazať pomocou príkazu `resethistory()` alebo opäť v hlavnom menu v `Preferences/Clear History`.

1.4. Ukládanie a načítavanie údajov

Aktuálne premenné je možné ukladať do súborov s príponami `.dat`, `.bin` alebo `.sav` pomocou hlavného menu `File/Save...` alebo v príkazovom riadku príkazom `save('cesta/názov'[,x1,x2,...,xn])`. Ak chceme uložiť všetky aktuálne premenné, stačí `save('cesta/názov')`.

Načítavanie údajov z uloženého súboru je možné pomocou príkazu `load('cesta/názov',[,x1,...,xn])`. Ak chceme načítať všetky premenné zo súboru, stačí použiť `load('cesta/názov')` alebo položku hlavného menu `File/Load...` a vybrať príslušný súbor.

Príklad. Zadáme 2 matice typu 2/2, uložíme ich a načítame.

```
-->a=eye(2,2);b=ones(a); //jednotková matica a matica jednotiek
-->save('d:\mydir\vals.bin',a,b); //uloženie
-->clear a b //vymazanie premenných
-->load('d:\mydir\vals.bin','b'); //načítanie
-->b  =
! 1. 1. !
! 1. 1. !
```

Analógiou príkazov `save` a `load` sú príkazy `write` a `read`.

Výsledky výpočtov, ako aj históriu, môžeme taktiež uložiť do textového súboru pomocou príkazu `savehistory('cesta/názov')`. Ak však nechceme ukladať celú históriu, len jej časť, môžeme to vykonať pomocou príkazu `diary`. Zápis začína `diary('cesta/názov')` a ukončí sa príkazom `diary(0)`. Všetko, čo sa nachádza medzi týmito príkazmi, vrátane výsledkov aj komentárov, sa uloží, napr.:

```
-->diary('D:/slon/priklad.txt')
-->a=5;b=6; // moje čísla}
-->c=a*b+3
 c  =
   33.
-->diary(0)
```

Poznámka. Ak sme nastavení v aktuálnom adresári, cestu v príkaze zadávať nemusíme, stačí zadať názov súboru.

1.5. Súbory – scenáre

Nie vždy je vhodné pracovať priamo v hlavnom pracovnom okne Scilabu, hlavne ak potrebujeme vykonať nejakú postupnosť zložitých príkazov. Na uľahčenie takejto činnosti slúžia scenáre (skripty, anglicky skripts). Sú alternatívou postupnosti príkazov priamo v dialógovom riadku. Súbory – scenáre – majú prípony `.sce` a dajú sa vytvárať v prostredí Scilab v editore Scipad. Tento editor môžeme otvoriť pomocou hlavného menu v ponuke `Editor` alebo zadaním príkazu `editor` alebo scipad v dialógovom riadku. Po napísaní požadovanej postupnosti príkazov v editore uložíme tento text do súboru s príponou `.sce`. Na jeho spustenie potom existujú tri spôsoby:

1. Ak máme v editore Scipad otvorený daný súbor, tak v menu editora vyberieme ponuku `Execute/Load into Slilab`, čomu tiež odpovedá dvojica klávesov `Ctrl + l`.
2. V hlavnom menu Scilabu v ponuke `File/exec...` vyberieme uložený súbor a otvoríme ho, ten sa automaticky po otvorení spustí.
3. Napíšeme v príkazovom riadku `exec('cesta/názov.sce')`. Ak je súbor uložený priamo v aktuálnom priečinku, stačí napísať v dialógovom riadku `exec názov.sce`.

Scenár nemusí byť napísaný bezprostredne v editore `Scipad`, môžeme použiť ľubovoľný textový ASCII editor a uložiť text do súboru s príponou `.sce`. Na spustenie potom použijeme druhý alebo tretí spôsob.

Týmto spôsobom umožňuje Scilab vytvárať programy a funkcie, o ktorých bude bližšie písané v oddiele 3.1 kapitoly Programovanie.

2. Typy premenných

V Scilabe sú zabudované rôzne typy premenných. Hlavné z nich sú:

- matice (reálne alebo komplexné, reťazcové, boolovské),
- reálne a komplexné čísla ako špeciálne prípady matíc,
- funkcie (makrá).

Základným typom premennej je matica. Teda aj skalárne veličiny sa chápu ako jednorozmerné matice. Pri opise syntaxe príkazov a jeho parametrov budeme nepovinné veličiny uvádzať v hranatých zátvorkách `[]`.

Funkcie sú podrobnejšie opísané v nasledujúcej kapitole Programovanie. V tejto časti sa bližšie zoznámime s prvým a druhým typom premenných a tiež s niektorými základnými operáciami s týmito premennými.

2.1. Skalárne veličiny a špeciálne konštanty

Skalárne veličiny v Scilabe môžu byť reálne alebo komplexné, napríklad

```
-->a=5-2*%i
 a =
5.-2.i
```

Medzi skalárne veličiny patria aj špeciálne konštanty, ktoré sú predefinované a nemôžu byť užívateľom prepísané (môžu byť však predefinované). Tieto konštanty sa začínajú znakom % a niektoré z nich sú:

`%i` je imaginárna jednotka ($i^2 = -1$)
`%pi` je Ludolfovo číslo $\pi = 3.1415927\ldots$
`%e` je Eulerovo číslo $e = 2.7182818\ldots$
`%eps` je počítačová nula, t. j. najväčšie kladné číslo ε, pre ktoré je $1 + \varepsilon = 1 - \varepsilon = 1$ (`%eps=2.220E-16`)
`%inf` je nekonečno
`%nan` je nedefinované (not a number)
`%t` je boolovské true = T
`%f` je boolovské false = F
`~%t` je zápor true = F!
`~%f` je zápor false = T!

Používateľ si môže definovať svoje konštanty aj bez znaku `%`.

2.2. Presnosť výpočtu a formát výstupu

V Scilabe sa skalárne premenné uchovávajú a tiež všetky výpočty sa vykonávajú s dvojnásobnou presnosťou (double precision). Na určenie formátu výstupu čísla slúži príkaz `format`, ktorého syntax je `format([typ],[dĺžka])`. Parameter `typ` môže mať hodnotu `"v"` – s pevnou desatinnou bodkou alebo `"e"` – s plávajúcou desatinnou bodkou. Parameter `dĺžka` je prirodzené číslo a určuje počet zobrazených znakov. Preddefinovaný formát je `format("v",10)`[1].

[1] To znamená, že sa zobrazí spolu 9 znakov (cifra+bodka+7 cifier), v prípade záporného čísla sa zobrazí aj znamienko

Príklad. Vyskúšame si výpis čísla ε v rôznych formátoch.

```
-->format("e",19)
-->%eps
 %eps  =
    2.220446049250D-16
-->format("v",19)
-->%eps
 %eps  =
    0.0000000000000002
```

Na formátovanie výpisu môžeme tiež použiť príkaz `printf`, napr.:

```
-->printf("%1.10f",%pi);
 3.1415926536
```

Vo výpise sa teda objaví 10 cifier po desatinnej bodke (`"%1.10f"`).

2.3. Vektory

Pri zadávaní vektora alebo matice v prostredí Scilab sa okrúhle zátvorky nahrádzajú hranatými a pri výpise sa zase okrúhle zátvorky zobrazujú pomocou znakov `!`. Riadkový vektor môžeme zadať tak, že súradnice oddelíme čiarkami alebo medzerami, napr.:

```
-->u=[1,-3*%i,7], v=[-1 3*%i +7]
 u=
!  1.   -3.i   7.  !
 v=
!  -1.   3.i   7.  !
```

Operácia transponovania sa vykonáva pomocou príkazu ', napr.:

```
-->b=[1 4];b'
 ans =
! 1. !
! 4. !
```

Stlpcový vektor teda môžeme zadávať pomocou riadkového vektora a operácie transponovania alebo priamo tak, že súradnice oddelíme bodkočiarkou:

```
-->v=[4;3+%i;3.33]
 v =
! 4.     !
! 3. + i !
! 3.33    !
```

Analogicky, riadkový vektor môžeme zapísať pomocou stĺpcového a operácie transponovania. Pre vektor, ktorého zložky sú `x[i]=x[i-1]+delta`, `xmin=x[0]` a `xmax=x[n]` je prípustný zápis `x=[xmin:delta:xmax]`. Napr.:

```
-->x=[0:0.1:2*%pi];
```

2.4. Matice

Scilab pracuje s rôznymi typmi matíc. V tejto časti sa obmedzíme na vytváranie a prácu s dvojrozmernými maticami. V Scilabe však existuje možnosť zadávať aj matice vyšších rozmerov ako 2 pomocou príkazu `hypermat`. Podrobnejšie sa s týmto príkazom môžeme zoznámiť pomocou `help hypermat`.

Spôsob 1. Priame zadávanie matice:

```
-->b=[11 22 33;21 22 23;31 32 33]

 b =
! 11. 22. 33. !
! 21. 22. 23. !
! 31. 32. 33. !
```

Spôsob 2. Zostavenie matice z niekoľkých podmatíc:

Príklad.

```
-->a=[1 2;3 4]

 a  =
! 1.   2. !
! 3.   4. !

-->b=[5;6]
```

```
 b  =
! 5. !
! 6. !

-->c=[7 8 9]
 c  =
! 7.   8.   9. !

-->d=[a,b;c]
 d  =
! 1.   2.   5. !
! 3.   4.   6. !
! 7.   8.   9. !
```

Spôsob 3. V Scilabe, podobne ako v MATLABe ale aj v iných prostrediach, existuje typ matice, tzv. riedka matica (sparse matrix). Je to matica, ktorej väčšina prvkov je rovná nule. Použitie premennej tohto typu je výhodné pre matice veľkých rozmerov. Umožňuje tak šetriť pamäť počítača a zvyšuje rýchlosť výpočtov. Na zadanie takejto matice slúži príkaz `[sp]=sparse(ij,v)`, kde parametre sú:

`ij` je matica typu $n/2$, ktorej riadky predstavujú pozície nenulových prvkov,
`v` je vektor dĺžky n s hodnotami na daných pozíciách.

Na konvertovnie riedkej matice na klasický typ zase slúži príkaz `full`.

Príklad.

```
-->sp=sparse([1,2;4,5;3,10],[1.35,-2.25,3.45])
 sp =
(  4,  10) sparse matrix

(  1,   2)   1.35
(  3,   8)   3.45
(  4,   5)  -2.25

-->r=full(sp)
 r =
!  0.   1.35  0.   0.    0.     0.   0.   0.     !
!  0.   0.    0.   0.    0.     0.   0.   0.     !
!  0.   0.    0.   0.    0.     0.   0.   3.45   !
!  0.   0.    0.   0.   -2.25   0.   0.   0.     !
```

2.5. Špeciálne typy matíc

V Scilabe existuje možnosť vytvárania niektorých špeciálnych typov matíc pomocou jednoduchých príkazov. Slúžia na to nasledujúce štandardné funkcie:

- `eye` – jednotková matica,
- `ones` – matica, ktorej všetky prvky sú jednotky,
- `zeros` – matica, ktorej prvky sú nuly,
- `rand` – matica vytvorená pomocou generátora náhodných čísel s rovnomerným rozdelením na intervale $[0, 1]$.

Jednotková matica sa vytvára pomocou príkazu `eye`. Pod pojmom jednotková však teraz máme na mysli nie v pravom slova zmysle jednotkovú. V Scilabe nemusí byť jednotková matica štvorcová. Musí mať však prvky na hlavnej diagonále, teda $a_{11}, a_{22}, \ldots a_{nn}$ rovné jednej a ostatné sú nuly, teda môže to byť aj vektor. Syntax príkazu je:

```
X=eye(m,n)
X=eye(A)
X=eye()
```

kde parametre

`A`, `X` sú matice rovnakého typu a
`m`, `n` sú celé čísla, `m` je počet riadkov, `n` je počet stĺpcov

Príklad 1. Vytvorenie jednotkovej matice typu $2/3$:

```
-->eye(2,3)
 ans =
! 1. 0. 0. !
! 0. 1. 0. !
```

Príklad 2. Vytvorenie matice rovnakého typu ako matica v (vektor), ktorá má na hlavnej diagonále jednotky a ostatné prvky sú rovné nule:

```
-->v=[2 -3.5 7];
-->a=eye(v)
 a =
! 1. 0. 0. !
```

Príklad 3. Vytvorenie matice s jednotkami na hlavnej diagonále a ostatnými prvkami rovnými nule neurčitého rozmeru. Rozmer tejto matice sa určí, až keď k nej pripočítame maticu s konkrétnym rozmerom:

```
-->r=eye();
-->q=[1 2 3;10 20 30];
-->w=q+r
 w =
! 2.  2.  3.  !
! 10. 21. 30. !
```

Maticu, ktorá má všetky prvky jednotky, resp. nuly, môžeme zadať pomocou príkazu `ones`, resp. `zeros`. Ich syntax je podobná ako v predchádzajúcom prípade. Najjednoduchší prípad `ones(<počet riadkov>,<počet stĺpcov>)`, `zeros(<počet riadkov>,<počet stĺpcov>)`

Príklad. Vytvorenie matice jednotiek typu 2/3:

```
-->ones(2,3)
```

```
 ans =
! 1. 1. 1. !
! 1. 1. 1. !
```

Príkaz `rand` generuje postupnosť náhodných čísel. Syntax je:

```
rand(m1,m2,.. [,kľúč])
rand(x [,kľúč])
rand()
rand(kľúč)
rand("seed" [,n])
rand("info")
```

Parametre:

`mi` je celé číslo (integer),

`kľúč` je symbolická premenná (character string), ktorá môže mať hodnoty `"uniform"` alebo `"normal"`. Hodnota `"uniform"` odpovedá rovnomernému rozdeleniu a `"normal"` normálnemu (Gaussovmu) rozdeleniu na intervale $[0, 1]$,

`x` je matica.

Príklady použitia:

- `rand(m1,m2)` – vytvorenie náhodnej matice typu m_1/m_2.

- `rand(m1,m2,...,mn)` – vytvorenie n-rozmernej náhodnej matice typu $m_1/m_2/\ldots/m_n$.
- `rand(A)` – vytvorenie náhodnej matice rovnakého typu ako bola matica `A`. Matica `rand(A)` bude komplexná, ak matica `A` bola komplexná.
- `rand()` – náhodné číslo.
- `rand("uniform") alebo rand("normal")` – umožňuje zadať typ rozdelenia (rovnomerné alebo normálne).
- `rand("info")` – vracia hodnotu premennej `kľúč`.

2.6. Operácie s maticami

V prostredí Scilab sú definované nasledujúce základné operácie s maticami:

`A+B` – súčet dvoch matíc
`A-B` – rozdiel dvoch matíc
`c*B` – násobok matice číslom
`A*B` – násobenie matíc
`A'` – transponovanie matice
`A^c` – umocňovanie matice
`A\B` – ľavé delenie, t. j. $A^{-1} \cdot B$
`A/B` – pravé delenie, t. j. $A \cdot B^{-1}$

`A.*B` – násobenie prvkov matíc po zložkách
`A.\B` – ľavé delenie prvkov matíc po zložkách, t. j. b_{ij}/a_{ij}
`A./B` – pravé delenie prvkov matíc po zložkách, t. j. a_{ij}/b_{ij}
`A.^c` – umocňovanie jednotlivých prvkov matice

Predpokladáme, že `A, B` sú matice, ktorých rozmer vyhovuje vykonávaným operáciám a `c` je reálne alebo komplexné číslo. Syntax týchto operácií si čitateľ môže pozrieť v opise priamo v prostredí Scilab alebo v dokumentácii.

Súčet všetkých prvkov matice nájdeme pomocou príkazu `y=sum(x)`,-`y=sum(x,'r')` alebo `y=sum(x,'c')`, kde `y` je číslo alebo vektor, `x` reálny alebo komplexný vektor alebo matica. Príkaz `y=sum(x)` vracia súčet všetkých prvkov matice alebo vektora. Príkaz `y=sum(x,'r')` je ekvivalentný príkazu `y=sum(x,1)` a vracia riadkový vektor, ktorého prvky sú rovné súčtu prvkov tvoriacich stĺpce matice `x`, teda $y(j) = \sum_{i=1}^{m} x(i,j)$. Analogicky príkaz `y=sum(x,'r')`, ktorý je ekvivalentný príkazu `y=sum(x,2)`, vracia stĺpcový vektor, s hodnotami súčtov prvkov tvoriacich riadky matice `x`, teda $y(i) = \sum_{j=1}^{n} x(i,j)$.

Príklad.

```
-->a=[10,20;300,400]
 a =
```

```
! 10.  20.  !
! 300. 400. !

-->b=sum(a)
 b =
  730.

-->c=sum(a,'r') // to isté ako c=sum(a,1)
 c =
! 310. 420. !

-->c=sum(a,2)   // to isté ako c=sum(a,'c')
 c =
! 30.  !
! 700. !
```

Súčin prvkov matice môžeme nájsť pomocou príkazu `prod`. Syntax príkazu a parametre sú analogické ako v predchádzajúcom prípade, teda `y=sum(x)`, `y=sum(x,'r')` (to isté ako `y=sum(x,1)`), `y=sum(x,'c')` (to isté ako `y=sum(x,2)`).

Príklad.

```
-->a=[10,20;300,400]
 a =
! 10.  20.  !
```

```
! 300. 400. !

-->prod(a)
 ans =
    24000000.

-->prod(a,1)   // to isté ako c=prod(a,'r')
 ans =
! 3000. 8000. !

-->prod(a,'c') // to isté ako c=prod(a,2)
 ans =
! 200.    !
! 120000. !
```

2.7. Reťazcové a symbolické matice

Okrem číselných matíc predstavuje Scilab možnosť pracovať s reťazcovými a symbolickými maticami. Medzi symbolické matice patria napr. matice, ktorých prvky sú polynómy alebo racionálne funkcie. O nich bude podrobnejšie písané v nasledujúcom oddiele, resp. v oddiele 5.1 poslednej kapitoly.

Reťazcové matice predstavujú matice, ktorých prvky sú reťazce (anglicky string). Tieto prvky sa musia zadávať v apostrofoch alebo úvodzovkách. Napríklad:

```
-->a=["2+3" 'bb";'cc' "dd']
 a =
! 2+3  bb !
!         !
! cc   dd !
```

Reťazcové matice sa môžu navzájom spájať a výsledok takéhoto spájania je nasledovný:

```
-->a=["a","b";"c","d"];
-->b=["x1","y1";"z1","w1"];
-->a+b
 ans =
! ax1  by1 !
!          !
! cz1  dw1 !
```

So symbolickými a reťazcovými maticami v Scilabe sú dovolené operácie, ktoré je možné pozrieť v manuále, resp. pomocou príkazu `help`. Najdôležitejšie z nich sú:

- `addf` – symbolické sčítanie,
- `mulf` – symbolické násobenie,
- `trianfml` – symbolická úprava na trojuholníkový tvar,

- `solve` – symbolické riešenie sústavy lineárnych rovníc, ktorých matica je trojuholníková,
- `trisolve` – analogicky s príkazom `solve`.

Príklad 1. Reťazcové spájanie:

```
-->w1="a";w2="b";
-->c=w1+w2
 c =
   ab
```

Príklad 2. Symbolické sčítanie a násobenie (medzi prvky doplní znak +, resp. *):

```
-->s=addf(w1,w2)
 s =
   a+b
-->p=mulf(w1,w2)
 p =
   a*b
```

Príklad 3. Úprava matice na trojuholníkový tvar:

```
-->A=['1','a';'2','-3']
 A  =
! 1  a  !
!       !
! 2  -3 !
```

```
-->B=trianfml(A)
 B  =
! 2  -3    !
!          !
! 0  2*a+3 !
```

Príklad 4. Symbolické riešenie sústavy, resp. maticovej rovnice `B*r=b`, kde `B` je trojuholníková matica z predchádzajúceho príkladu:

```
-->b=['x';'y'];
-->r=solve(B,b)
 r  =
! 2\(x+3*((2*a+3)\y)) !
!                      !
! (2*a+3)\y            ! // Pozor! - ľave delenie[2]
```

Rovnaký výsledok dosiahneme aj použitím príkazu `r=trisolve(B,b)`.

2.8. Polynómy a polynomické matice

Algebrický polynóm môžeme zadávať nasledujúcimi spôsobmi:

Spôsob 1. Zadávanie polynómu pomocou príkazu `poly`, ktorého syntax je `[p]=poly(a,"x", ["typ"])`. Parametre sú:

[2] Ide o ľave delenie, `(2*a+3)\y` je vlastne `y/(2*a+3)`.

a je reálna matica alebo reálne číslo,
x je symbolická premenná,
"typ": premenná, ktorá môže mať hodnoty "roots" – korene polynómu alebo "coeff" – koeficienty polynómu. Preddefinovaná je hodnota "roots". Môžeme ich použiť aj skrátene, teda "roots"="r" a "coeff"="c".

Príklad.

```
-->x=poly([1 5],"y","r")
 x  =
                2
    5 - 6y + y

-->poly([1 2 3],"x","c")
 ans  =
                 2
    1 + 2x + 3x
```

Poznámka. Exponenty sa pri výpise zobrazujú o jeden riadok vyššie, nie v tvare ^, ako sa zadávajú.

Spôsob 2. Priame zadávanie polynómu:

```
-->x=poly(0,"x"); // deklarácia symbolickej premennej
-->1+2*x+3*x^2
```

```
 ans  =
                    2
    1 + 2x + 3x
```

Poznámka. Scilab môže pracovať len s jednou takto definovanou premennou. Napríklad:

```
-->a=poly(0,"a");
-->b=poly(0,"b");
-->c=a+b
```

dáva výsledok:

```
      !--error 4
undefined variable : %p_a_p
```

Racionálne funkcie môžeme vytvoriť pomococou príkazu `poly` a znaku delenia `/`, napr.:

```
-->poly([1 -1],"x","c")/poly([1 2 3],"x","c")
 ans  =
       1 - x
    -----------
                2
    1 + 2x + 3x
```

Polynomickou maticou (polynomial matrix) nazveme symbolickú maticu, ktorej prvky sú polynómy. Takéto matice sa bežne vyskytujú v rôznych

oblastiach aplikovanej algebry, pozri napr. (HENRION, LASSERRE, 2006). Polynomickú maticu môžeme zadávať napríklad príkazmi:

```
-->s=poly(0,"s");
-->p=1+2*s+s^2;
-->M=[p,p-1;p+1,2]

 M =
!            2          2  !
!  1 + 2s + s     2s + s   !
!                          !
!            2             !
!  2 + 2s + s     2         !
```

Ďalšie informácie o operáciách s maticami budú uvedené v oddiele 5.1 podkapitoly Lineárna algebra.

2.9. Boolovské matice

Boolovské matice sú matice, ktorých prvkami sú boolovské (logické) konštanty

```
%t = T  TRUE
%f = F  FALSE
```

Syntax je rovnaká ako pri číselných maticiach. Nad boolovskými maticami sú definované nasledujúce operácie:

== – rovnosť (equal)
~ – logický zápor
| – logické „alebo“ (OR)
& – logické „a“ (AND)

Príklad 1.

```
-->[1,2]==[1,3]
 ans =
! T F !
```

Príklad 2.

```
-->a=[1 2 4 5 1.5];
-->a>2
 ans =
! F F T T F !
```

Príklad 3.

```
-->A=[%t,%f,%t,%f,%f,%f];
-->B=[%t,%f,%t,%f,%t,%f];
-->A | B
 ans =
! T F T F T F !
-->A&B
 ans =
! T F T F F F !
```

2.10. Celočíselné matice

V Scilabe je deklarovaných 6 typov premenných typu integer:

32 bit signed integer (podtyp 4)
32 bit unsigned integer (podtyp 14)
16 bit signed integer (podtyp 2)
16 bit unsigned integer (podtyp 23)
8 bit signed integer (podtyp 2)
8 bit unsigned integer (podtyp 12)

Na konvertovanie premenných jedného typu na iný slúžia nasledujúce operátory:

`y=int8(X)` – vracia hodnoty z intervalu $[-128, 127]$
`y=uint8(X)` – vracia hodnoty z intervalu $[0, 255]$
`y=int16(X)` – vracia hodnoty z intervalu $[-32768, 32767]$
`y=uint16(X)` – vracia hodnoty z intervalu $[0, 65535]$
`y=int32(X)` – vracia hodnoty z intervalu $[-2147483648, 2147483647]$
`y=uint32(X)` – vracia hodnoty z intervalu $[0, 4294967295]$.

Parametre:

`X` je reálna alebo celočíselná matica
`y` je celočíselná matica.

Príklad.

```
-->x=[0.11 3.2 27.5 187];
-->int16(x)
 ans =
!0 3 27 187 !
-->int8(x)
 ans =
!0 3 27 -69!
```

Na konvertovanie typu integer na premenné typu real slúži príkaz `y=double(X)`, kde `X` je reálna alebo celočíselná matica a `y` je reálna matica.

2.11. Rozmer matice a indexovanie

Indexovanie prvkov matice sa vykonáva podobne ako v algebre. Prvá súradnica určuje riadok, druhá stĺpec. Na jednoduchých príkladoch ukážeme, ako môžeme vypísať resp. predefinovať prvky na daných pozíciách.

Vygenerovanie náhodnej matice typu `3/4`:

```
-->format('v',4);
-->A=rand(3,4)
 A  =
!  0.4    0.5    0.5    0.6  !
!  0.3    0.3    0.4    0.4  !
!  0.6    0.6    0.3    0.9  !
```

Výpis prvku na pozícii (2,3), teda v druhom riadku a v treťom stĺpci:

```
-->A(2,3)
 ans  =
    0.4
```

Výpis vektora – tretieho riadku matice:

```
-->A(3,:)
 ans  =
!  0.6    0.6    0.3    0.9  !
```

Prepísanie druhého stĺpca matice novými hodnotami:

```
-->A(:,2)=[1 2 3]'
 A  =
!  0.4    1.    0.5    0.6  !
!  0.3    2.    0.4    0.4  !
!  0.6    3.    0.3    0.9  !
```

Na zisťovanie rozmerov[3] a typov matíc, resp. dĺžky vektorov slúžia príkazy `size` resp. `length`. Napríklad:

[3] Počet prvkov vektora `size(A)` určuje rozmer matice `A`.

Typ matice:

```
-->size(A)
 ans  =
    3.    4.
```

Typ vektora tvoriaceho prvý riadok matice:

```
-->size(A(1,:))
 ans  =
    1.    4.
```

Dĺžka vektora tvoriaceho prvý riadok matice:

```
-->length(A(1,:))
 ans  =
    4.
```

3. Programovanie

3.1. Štruktúra funkcií

V prostredí Scilab existuje možnosť používať a vytvárať funkcie, ktoré majú veľkú úlohu pri programovaní. Pre programy (skripty, scenáre) sú odporúčané prípony súborov `.sce` a pre funkcie `.sci`.

Na písanie programov je vhodný editor Scipad, s ktorým sme sa zoznámili už v prvej kapitole v oddiele 1.5. Tento editor sa otvára z hlavného pracovného okna Scilabu v položke menu `Editor`, alebo napíšeme v príkazovovom riadku príkaz `editor` alebo `scipad`.

Pri konštrukcii podmienok a cyklov sa používajú tieto operátory:

`==` – rovná sa
`<` – menší ako
`>` – väčší ako
`<=` – menší alebo rovný ako
`>=` – väčší alebo rovný ako
`<>` alebo `~=` nerovná sa

Syntax pre tvorbu funkcií

```
Function[y1,...,yn]=názov(x1,...xm)
<telo funkcie>
endfunction
```

kde `názov` je meno funkcie, `xi` sú vstupné parametre (ich počet je `m`) a `yi` sú výstupné parametre (ich počet je `n`).

Príklad. Napíšme program na výpočet faktoriálu.

```
function [x]=fact(k)
k=int(k)
if k<1 then k=1, end
x=1;
for j=1:k,x=x*j;end
endfunction
```

Tento text môže byť napísaný v ľubovoľnom editore a uložíme ho napr. ako `fact.sci`. Prípona `.sci` však nie je nevyhnutná. Potom voláme tento súbor pomocou príkazu `getf(názov)` alebo `exec(názov,-1)`[4]. To isté môžeme vykonať myšou v položke `File/Exec...` hlavného menu pracovného okna Scilabu a vybrať príslušný súbor.

Funkcie je možné definovať aj pomocou príkazu `deff`, ktorého syntax je `deff('[s1,s2,...]=názov(e1,e2,....)',text)`, kde `e1,e2,...` sú vstupné parametre (premenné), `s1,s2,...` sú výstupné parametre, `názov` je meno funkcie a `text` je vektor alebo matica s telom funkcie. Tento príkaz je vhodný, ak definujeme funkciu priamo v príkazovom riadku pracovného okna, ale môže sa samozrejme použiť aj pri definovaní funkcie v súbore `názov.sci`.

Poznámka. V Scilabe nie je nevyhnutné zadávať názov súboru rovnaký ako meno funkcie.

[4] Parameter -1 potláča výpisy systému (pozri 3.5)

Príklad. Zadefinujme funkciu dvomi spôsobmi.

```
deff('[z1,z2]=f1(x,y)',['z1=x+y';'z2=x-y'])
```

je to isté ako

```
function [z1,z2]=f1(x,y)
z1=x+y
z2=x-y
endfunction
```

3.2. Cykly

V Scilabe sú definované 2 druhy cyklov, a to `for` a `while`.

Spôsob 1. Pomocou príkazu `for`. Syntax je nasledovná:

```
for premenná=n1:krok:n2,
    <telo cyklu>
end
```

Ak je `krok` rovný jednej, tak sa nemusí zadávať.

Príklad. Výpočet faktoriálu.

```
-->x=1;for k=1:4,x=x*k,end
 x =
   1.
```

```
 x =
   2.
 x =
   6.
 x =
   24.
```

Spôsob 2. Pomocou príkazu `while`, ktorého syntax je

```
while <podmienka>, <telo cyklu> end
```

Príklad.

```
-->x=1; while x<6, x=x*2, end
 x =
   2.
 x =
   4.
 x =
   8.
```

Cykly `for` a `while` môžu byť prerušené pomocou príkazu `break`.

Príklad.

```
-->a=0; for i=1:5:100, a=a+1; if i>10 then break, end;end
-->a
 a =
   3.
```

3.3. Podmienky

Konštrukcia podmienky môže byť uskutočnená pomocou `if-then-else` alebo `select-case`.

Spôsob 1. Konštrukcia `if-then-else`. Syntax je:

```
if podmienka1 then
     vykonaj1
elseif podmienka2 then
     vykonaj2
...
else vykonajn
end
```

Príklad.

```
i=2;
for j = 1:3,
  if i == j then
     a(i,j) = 2;
  elseif abs(i-j) == 1 then
     a(i,j) = -1;
  else a(i,j) = 0;
  end,
end
```

Výsledok:

```
-->a
 a =
!   0.   0.   0. !
! - 1.   2. - 1. !
```

Spôsob 2. Konštrukcia pomocou operátora `select-case`. Syntax je:

```
select výraz0,
case výraz1 then vykonaj1,
case výraz2 then vykonaj2,
...
case výrazn then vykonajn,
[else vykonaj],
end
```

Príklad.

```
-->x=-1;
-->select x, case 1,y=x+5,case -1,y=sqrt(36),end
```

Výsledok:

```
  y  =
     6.
```

3.4. Pauza

Na dosiahnutie pauzy v programe slúži príkaz `xpause(mikrosekundy)`. Tento príkaz zastaví výpočtový proces na počet mikrosekúnd, ktoré zadávame v jeho argumente `mikrosekundy`. Skutočný čas však môže byť oveľa väčší, jednak z dôvodu výpočtu a jednak z dôvodu času, ktorý je potrebný na zavolanie tohoto príkazu. Príkaz `xpause` je výhodné použiť napr. pri animácii.

Príklad.

```
xbasc();
xset("color",12);
xstring(1,1,"Blue");
xpause(5.e6); // pauza 5 sekund
xbasc();
xset("color",5);
xstring(0,1,"Red");
```

3.5. Spúšťanie funkcií

Na spúšťanie funkcií slúžia príkazy `exec` a `getf`. Ak sú funkcie definované pomocou `function[y1,...,yn]=názov(x1,...,xn)`, vstupné a výstupné parametre `xi` a `yi` môžu byť ľubovoľné objekty Scilabu, okrem týchto samotných funkcií. Spúšťaný súbor môže obsahovať aj niekoľko funkcií.

Funkcie môžu byť definované aj bezprostredne v samotnom pracovnom okne Scilabu pomocou `function/endfunction`. Tento spôsob je výhodný, ak chceme definovať funkciu ako výstupný parameter inej funkcie.

Funkcie sa dajú spúšťať rovnako ako scenáre aj priamo z editora Scipad. Ak máme v tomto editore otvorený súbor s funkciou, stačí ísť do menu Scipadu a v ponuke `Execute` vybrať položku `Load into scilab` alebo jednoducho pomocou dvojice kláves `Ctrl+l`.

Spôsob 1. Spúšťanie pomocou príkazu `exec`, ktorého syntax je:

```
chyba=exec(názov [,mód])
chyba=exec(cesta/názov [,mód])
```

Parametre sú:

`názov` – meno súboru, v ktorom sa nachádza funkcia (ak je funkcia definovaná priamo v Scilabe alebo ak je uložená v aktuálnom priečinku)

`cesta/názov` – úplná cesta a meno súboru, v ktorom sa nachádza funkcia

`chyba` – celé číslo – 0 alebo číslo chyby

`mód` – celé číslo, ktoré môže nadobúdať nasledujúce hodnoty:

`0` – preddefinované

`-1` – nič nezobrazí

`1` – (echo) zobrazí každý príkazový riadok

2 – (prompt)
3 – echo+prompt
4 – stop
7 – promt+stop+echo

Príklad.

```
-->exec('C:\program files\scilab\macros\algebre\aff2ab.sci')
```

Spôsob 2. Spúšťanie funkcií pomocou príkazu `getf`, ktorého syntax je `getf(názov [,mód])`. Parameter `názov` je názov súboru a `mód` je ten istý ako v predchádzajúcom spôsobe. Tento príkaz sa však už považuje za zastaralý, uprednostňuje sa príkaz `exec`.

3.6. Globálne a lokálne premenné

Lokálne premenné

Ak hodnoty premenných vo funkciách nie sú definované a nie sú ani vstupnými parametrami, tak sa berú ako premenné z aktuálneho pracovného prostredia.

Príklad. Napíšeme v ľubovoľnom editore funkciu:

```
function [y1,y2]=f(x1,x2)
y1=x1+x2;
```

```
y2=x1-x2;
endfunction
```

a uložíme ju napr. do súboru `D:\zzz\fact.sci`. Potom ju spustíme príkazom:

```
-->exec('D:\zzz\fact.sci')
-->[y1,y2]=f(1,1)
```

Výsledok:

```
 y2  =
    0.
 y1  =
    2.
```

Lepšie je ale pri výpočte nepoužívať označenie premenných, ktoré sú použité v definícii funkcie, ale zaviesť nové, npr. `[a,b]=f(1,1)`.

Globálne premenné

Globálne premenné možu byť deklarované pomocou príkazu `global`. Syntax je:

```
global('premenná1',...,'premennán')
global premenná1␣...␣premennán
```

Parametre `premenná1,...,premennán` sú názvy premenných.

Príklad. Deklarácia globálnych premenných `a`, `b` a `c`:

```
global a b c
a=1;b=2;c=3;
```

Na vymazanie globálnych premenných z pamäte slúži príkaz:

`clearglobal()` – vymaže všetky globálne premenné
`clearglobal premenná1␣...␣premennán`, resp.
`clearglobal('premenná1', ..,'premennán')` – vymaže globálne premenné s názvami `premenná1` až `premennán`

3.7. Niektoré špeciálne príkazy pre funkcie

`argn` – pri spustení funkcie vracia počet vstupných a výstupných parametrov

`error` – umožňuje tlačiť oznamy o chybách alebo vetveniach pre prípad hľadania chyby

`pause` – používa sa pri ladení programu

`abort` – ukončenie príkazu `pause`

`warning` – varovanie

`break` – ukončenie cyklu

`return` a `resume` – slúžia na prenos lokálnych premenných z funkcie do tela programu

Príklad.

```
function [z]=funkcia(x,y)
if x==0 then,
error('delenie nulou');
end,
z=x/y
endfunction
```

Výsledok použitia funkcie:

```
-->funkcia(6,2)
 ans =
    3.
-->funkcia(6,0)
 !--error 27
delenie nulou...
at line      5 of function foo called by :
funkcia(6,0)
```

4. Grafika

Keďže Scilab, bol vyvinutý hlavne na prácu s maticami a grafikou, jeho možnosti sú skutočne veľmi široké a dajú sa porovnať s MATLABom, aj keď, čo sa týka kvality výstupov, trochu za ním zaostávajú. V tejto učebnici nie je prakticky možné ich všetky popísať. Vyberieme len tie najdôležitejšie z pohľadu autora.

4.1. Grafické okno a grafické prostredie

Ako už bolo spomínané, grafika v Scilabe sa vyvíja a v poslednej verzii 4.0 je práca s grafickým oknom a jeho menu čiastočne iná ako v starších verziách. V dokumentácii sa však tieto zmeny ešte neodrazili.

Grafické aplikácie v prostredí Scilab sa uskutočňujú v oddelenom okne, ktoré sa nazýva grafickým. V starších verziách sa grafické okno dá otvoriť pomocou menu v hlavnom okne Scilab `Graphic Window"x"/Set(Create) Window`, kde `"x"` je aktuálne číslo grafického okna. Preddefinovaná hodnota je 0 a otvára sa okno s týmto číslom. Číslo grafického okna je uvedené v jeho záhlaví. Vo verziách `3.1.1` (2005) a `4.0` (apríl 2006) sa takáto možnosť v hlavnom menu nenachádza a grafické okno sa otvorí priamo po zadaní konkrétneho príkazu na kreslenie, napr. `plot()`, `plot(číslo)` alebo `scf(číslo)`.

Je prípustné otvárať súčasne viacero okien, no aktívne je len jedno, tzv. aktuálne. Otváranie ďalších okien sa vykonáva v menu už otvoreného

grafického okna v ponuke `File/New...` a aktuálne ostáva stále posledné okno. Ak chceme nastaviť niektoré iné okno ako aktuálne, je to možné vykonať v menu daného okna v `Edit/Select figure as current` alebo príkazom `scf(číslo)` (anglicky set current figure).

V krátkosti ešte opíšeme menu grafického okna. Ponuka `File` obsahuje možnosť otvárať nové grafické okná, otvárať už existujúce uložené obrázky, ukladať, exportovať a tlačiť obrázky a zatvoriť grafické okno. V ponuke `Tools` je možné zväčšovať, zmenšovať a rotovať obrázok. Všetky tieto možnosti sú k dispozícii aj priamo v menu grafického okna. Ponuka `Edit` zase obsahuje možnosť prekreslenia obrázku, očistenie grafického okna a editovania obrázku. Možnosť editovania je takisto vysvietená pod samotným menu okna.

Vyčistiť, ale nie zavrieť, aktuálne grafické okno je možné pomocou menu `Edit/Erase figure` ale aj priamo v príkazovom riadku Scilabu pomocou príkazu `xbasc()`. Ak chceme vyčistiť súčasne niekoľko grafických okien, použijem príkaz `xclear([vektor])`, kde `vektor` je celočíselný vektor alebo skalár s číslami okien, ktoré chceme vyčistiť. Napr. pomocou príkazu `-->xclear([0,3,5])` sa vyčistia okná s číslami 0, 3 a 5.

V jednom grafickom okne sa dá zobraziť aj viac grafov. Slúži na to príkaz `subplot` alebo tiež `subplot(mnp)`. Parametre `m`, `n` a `p` sú kladné celé čísla. Okno sa takto rozdelí na tabuľku `m`×`n` podokien a v každom z nich sa môže zobraziť graf. Parameter `p` určuje aktuálne podokno. Príklad sa nachádza nižšie pri opise príkazu `plot2d` v nasledujúcom oddiele.

Popisom osí, názvom grafu, štýlom a veľkosťou písma, škálami, farbami a pod. sa špeciálne zaoberať nebudeme. Tieto opisy je možné vykonávať aj v hotovom grafe priamo v grafickom okne v interaktívnom režime pomocou menu `Edit/Figure properties` alebo `Edit/Current axes properties`. Graf s názvom, popisom osí a rôznymi štýlmi čiar je uvedený v príklade 4 nasledujúceho oddielu.

Obrázok je možné pomocou menu grafického okna `File/Save...` uložiť do súboru s príponou `.scg` ako `scilab graphics`. Ten sa potom môže otvárať buď z grafického okna v menu `File/Load...`, alebo z príkazového riadku pomocou `load('cesta/názov')`. Hotový obrázok je možné exportovať do niekoľkých formátov, napr. `.eps, .gif, .bmp` opäť pomocou menu grafického okna `File/Export`.

4.2. 2D grafika

Kreslenie grafov funkcií jednej premennej je v Scilabe možné pomocou základných príkazov `plot` a `plot2d`. Ak sú funkcie zadané ako vonkajšie, používame príkazy `fplot` a `fplot2d`. Môžu obsahovať veľké množstvo rôznych parametrov, s ktorými je možné sa zoznámiť pomocou `help plot`, resp. `help plot2d`. My si ukážeme niektoré minimálne možnosti použitia.

Spôsob 1. Pomocou príkazu plot `plot(x,y)` alebo len `plot(y)`, kde `x` a `y` sú vektory rovnakej dĺžky, ktoré predstavujú hodnoty argumentov a funkčné hodnoty. Ako už vieme, vektory je možné zadávať rôznymi spôsobmi. Ukážeme si niekoľko príkladov použitia tohto príkazu.

Výsledok všetkých však uvádzať nebudeme, čitateľ si ich môže sám vyskúšať.

Príklad 1.

```
x=[1 2 3 4 5 6];
y=x^2;
plot(y)
```

Príklad 2.

```
x=[1 2 3 4 5 6 7 8];
y=[1 4 9 16 25 36 49 64];
plot(x,y)
```

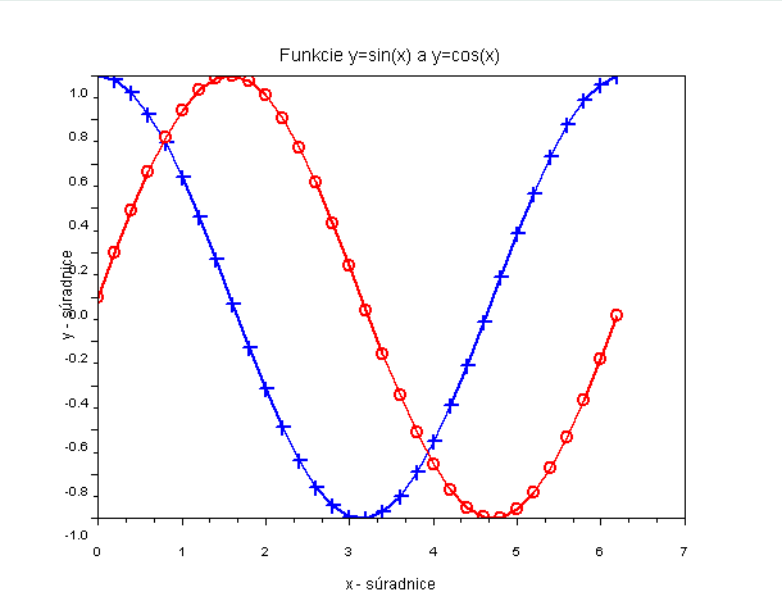
Príklad 3.

```
x=0:0.1:2*%pi;
plot(x,[sin(x);cos(x)])
```

Príklad 4.

```
x=0:0.1:2*%pi;
plot(x,sin(x),'ro-','linewidth',2,'markersize',8);
plot(x,cos(x),'b+-','linewidth',2,'markersize',8)
xtitle("Funkcie y=sin(x) a y=cos(x)","x - súradnice",..
"y - súradnice");
```

Výsledok:



Poznámka. Príkaz `plot(x,[sin(x);cos(x)])` dá ten istý výsledok ako príkazy `plot(x,sin(x))`, `plot(x,cos(x))`, len funkcie budú zobrazené inými farbami

Spôsob 2. Pomocou príkazu `plot2d`. Jeho syntax je nasledovná

```
plot2d([x],y)
plot2d([x],y,[voľby])
```

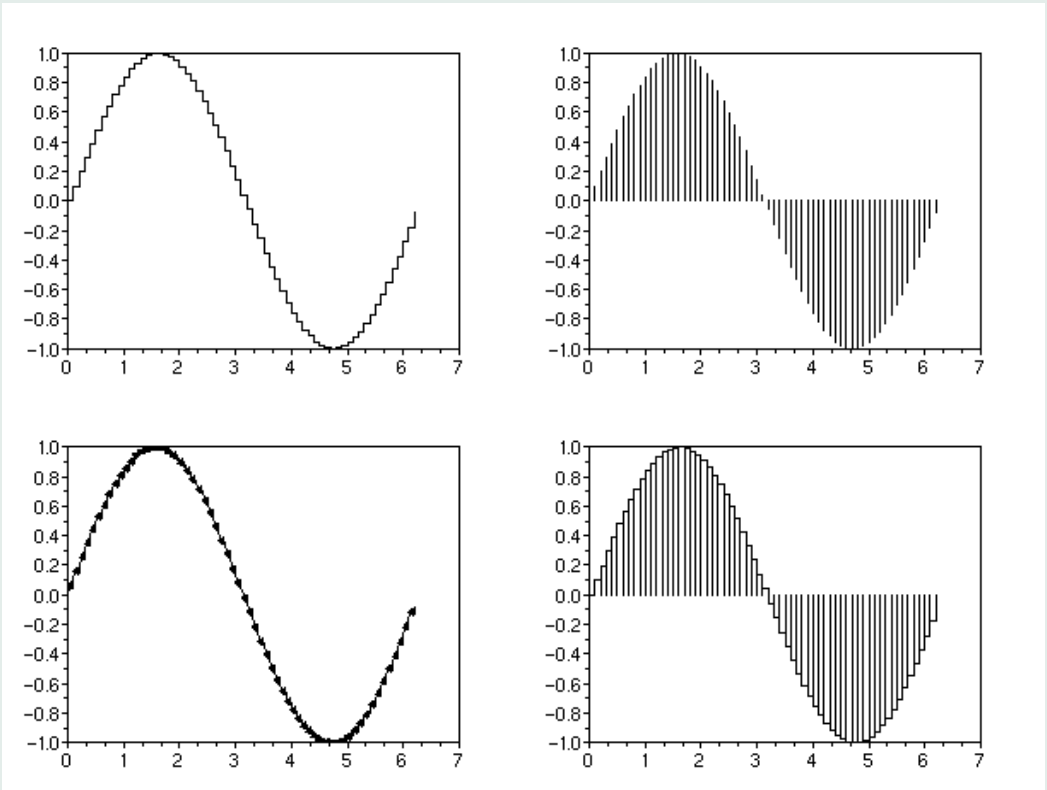
kde `x` a `y` sú vektory rovnakej dĺžky (argumenty a funkčné hodnoty). Vektor `x` nie je povinný, v takom prípade sa berie postupnosť prirodzených čísel `1:n`, kde `n` je dĺžka vektora `y`. Nepovinný parameter `[voľby]` môže obsahovať postupnosť parametrov, ktorú môžeme zadávať v tvare: `kľúč1=hodnota1,kľúč2=hodnota2,...` Parametre `kľúč1`, `kľúč2`,... môžu byť napr. štýl krivky, legenda krivky, hranice grafu, rozsah na osiach (lineárna alebo logaritmická), sieťka ako aj niektoré iné, ktoré je možné podrobne pozrieť pomocou príkazu `help plot2d`.

Existujú aj možnosti kreslenia grafov vyššej úrovne, a to pomocou príkazov, ktoré sú analogické príkazu `plot2d`. Tie sú:

`plot2d1` = `plot2d`
`plot2d2` je histogram bez vertikálnych delení
`plot2d3` je schodíková funkcia
`plot2d4` je krivka zobrazená šípkami

Príklad. Vyskúšajme si tieto príkazy na grafe funkcie sínus.

```
-->x=[0:0.1:2*%pi]';
-->subplot(221);plot2d2(x,sin(x));
-->subplot(222);plot2d3(x,sin(x));
-->subplot(223);plot2d4(x,sin(x));
-->subplot(224);plot2d2(x,sin(x));plot2d3(x,sin(x));
```


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0 1 2 3 4 5 6 7

4.3. Niektoré špeciálne 2D grafy a grafy geometrických útvarov

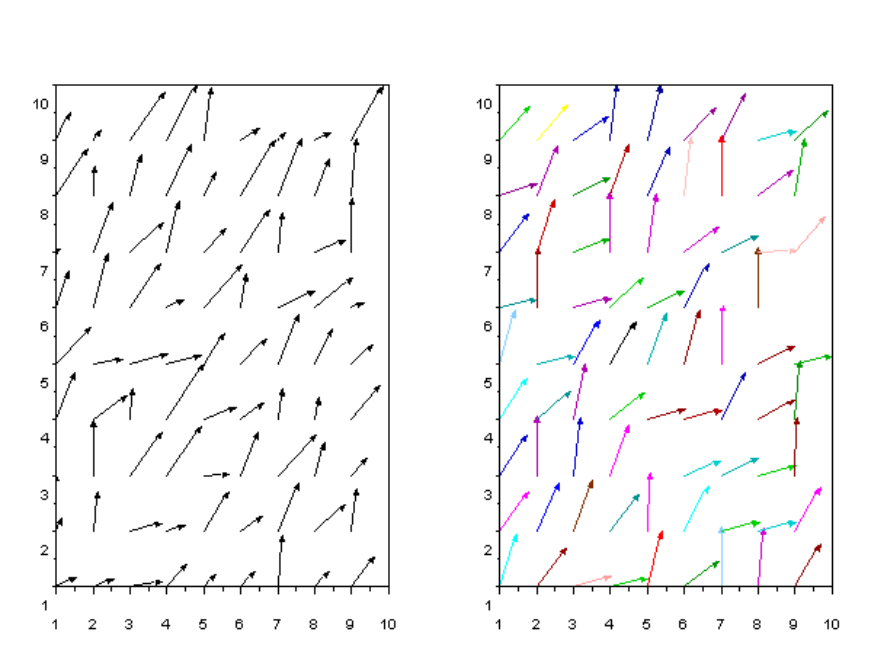
Vektorové pole

Pomocou príkazov `champ`, `champ1`, resp. `fchamp` (ak je funkcia zadaná ako vonkajšia) môžeme zobraziť vektorové pole v rovine. Ich povinná syntax je `champ(x,y,fx,fy)`, kde `x` a `y` sú vektory určujúce súradnicový systém, `fx` a `fy` sú matice, ktoré opisujú x-ové a y-ové zložky vektorového poľa.

Príkaz `champ` zobrazí čierne šípky, ktorých dĺžka bude závisieť od intenzity poľa. Na zobrazenie intenzity poľa pomocou farieb použijeme príkaz `champ1`.

Príklad.

```
-->subplot(121);champ(1:10,1:10,rand(10,10),rand(10,10),1.0);
-->subplot(121);champ1(1:10,1:10,rand(10,10),rand(10,10),1.0);
```



Histogramy

Histogram zobrazíme pomocou príkazu `histplot`, resp. `hist3d` v priestore (bude ukázaný nižšie).

Príklad.

```
-->histplot([-6:0.2:6],rand(1,2000,'n'));
```

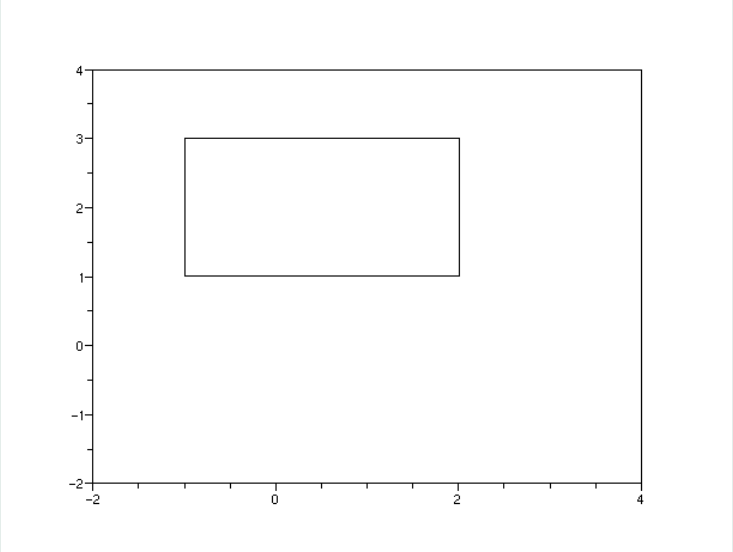
Obdĺžniky

Pomocou príkazu `rect(x,y,v,h)` môžeme nakresliť obdĺžnik v rovine, kde parametre sú x-ová a y-ová súradnica pravého horného vrcholu a následne jeho dĺžka a hĺbka.

Príklad. Nakreslime obdĺžnik v rovine.

```
-->plotframe([-2,-2,4,4],[3,4,1,7]);//určenie hraníc obrázku
-->xrect(0,2,1,1)
```

Výsledok:



Príkaz `plotframe(hranice,značky)`, ktorý určuje hranice a škálu osí má nasledujúce parametre:

`hranice` je vektor `[xmin,ymin,xmax,ymax]`, ktorý udáva hranice obrázku,

`značky` je vektor `[nx,mx,ny,my]`, ktorý určuje delenia osí. Doplní `mx` hodnôt s číslami medzi `xmin` a `xmax` a potom ešte do každeho podintervalu doplní `nx` hodnôt. Analogicky `ny` a `my`.

4.4. 3D grafika

Dvojrozmeré plochy v priestore môže predstavovať funkcia dvoch premenných, resp. matice. Graf plochy v priestore môžeme zostrojiť pomocou nasledujúcich príkazov:

`plot3d` – kreslí graf cez body zadané maticami x, y, z

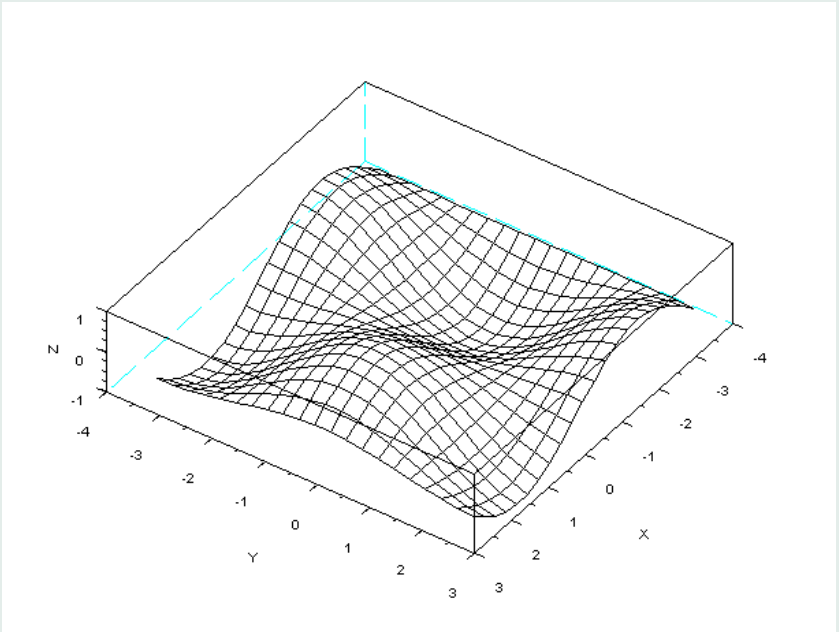
`plot3d1` – kreslí graf cez body zadané maticami x, y, z pomocou úrovní farieb

`fplot3d` – to isté ako `plot3d`, no plocha je zadaná pomocou vonkajšej funkcie

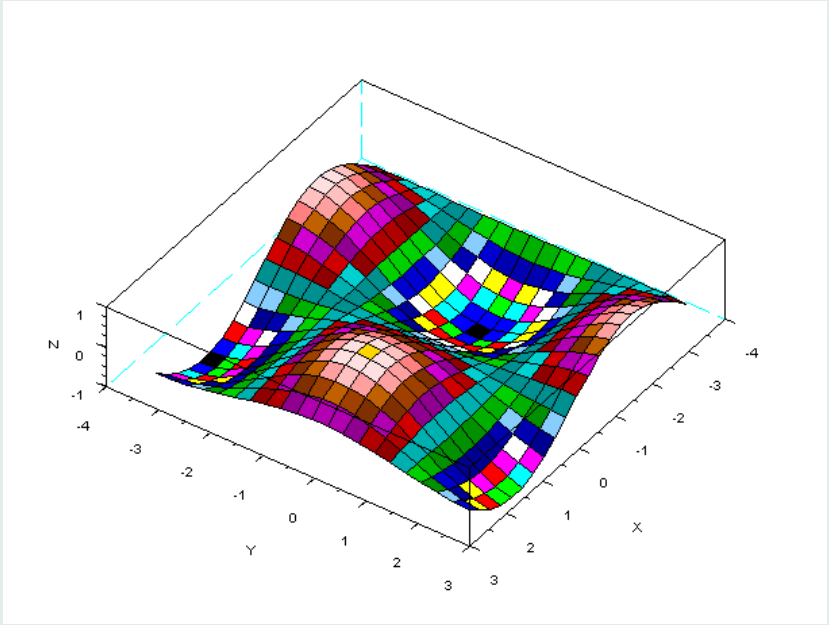
`fplot3d1` – to isté ako `plot3d1`, no plocha je zadaná pomocou vonkajšej funkcie

Syntax príkazov môže obsahovať množstvo rôznych parametrov, ktoré je možné pozrieť pomocou príkazu `help`, my odskúšame tieto príkazy na jednoduchom príklade

```
-->t=-%pi:0.3:%pi;
-->plot3d(t,t,sin(t)'*cos(t),35,45,"X@Y@Z",[0,2,3]);
```



```
-->plot3d1(t,t,sin(t)'*cos(t),35,45);
```

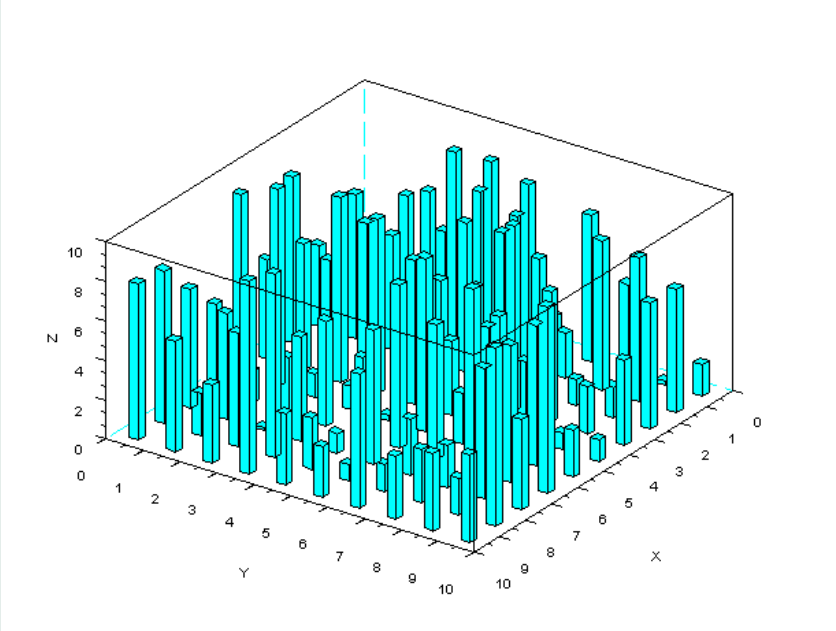


Rovnaké grafy by sme dosiahli aj nasledujúcim spôsobom:

```
-->deff('[z]=surf(x,y)','z=sin(x)*cos(y)');
-->t=-%pi:0.3:%pi;
-->fplot3d(t,t,surf,35,45);
-->fplot3d1(t,t,surf,35,45);
```

Na vykreslenie histogramu v priestore použijeme príkaz `hist3d`, napríklad:

```
-->hist3d(10*rand(10,10))
```

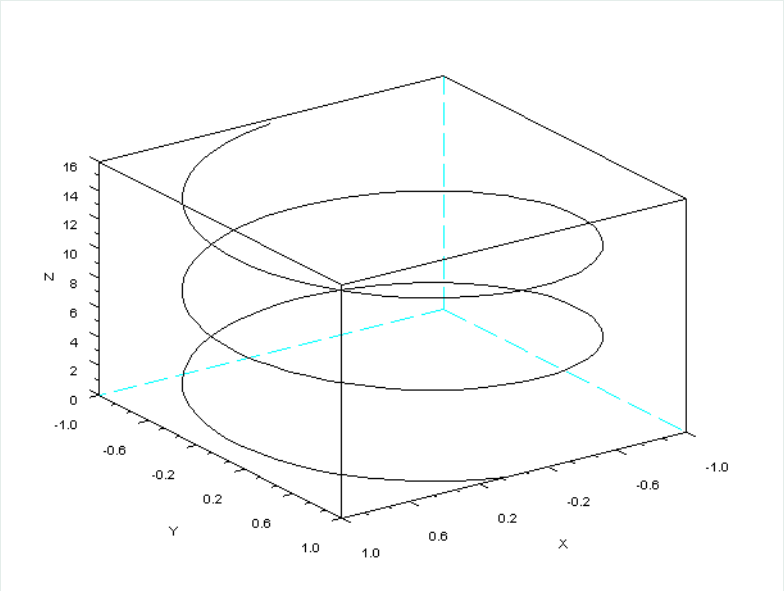


4.5. Krivka v priestore

Parametrickú krivku v priestore zostrojíme pomocou príkazu `param3d` alebo `param3d1`. Uvedieme iba povinnú syntax, ktorá je v oboch príkazoch rovnaká, a to `param3d(x,y,z)`, kde `x`, `y` a `z` sú vektory, ktoré predstavujú súradnice bodov parametrickej krivky. Príkaz `param3d1` použijeme, ak chceme zobraziť niekoľko funkcií tých istých argumentov v jednom grafe. Pre jednu funkciu je výsledok ten istý ako pomocou `param3d`.

Príklad.

```
t=0:0.1:5*%pi;
param3d(sin(t),cos(t),t,55,10);
```

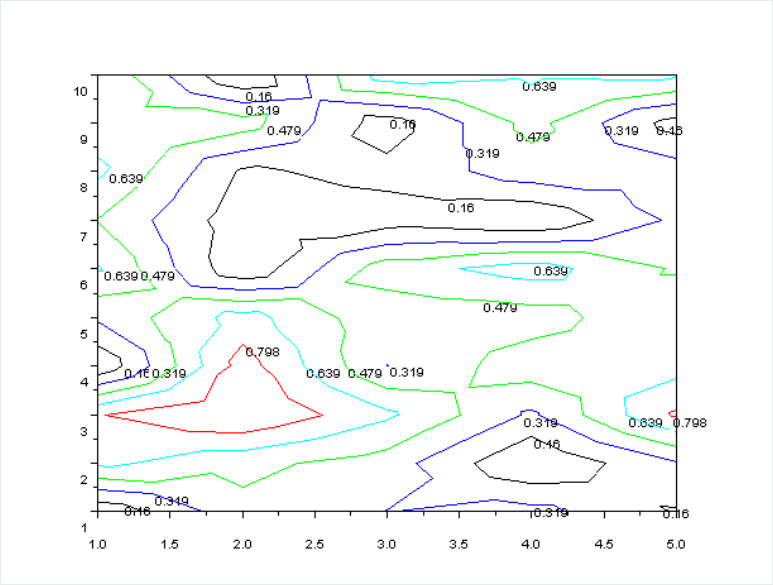


4.6. Zobrazenie vrstevníc

Na vykreslenie vrstevníc (izolínií) funkcií dvoch premenných slúžia príkazy `contour`, `contour2d`, `fcontour` a `fcontour2d`. Druhá dvojica je analogická s prvou, no krivka musí byť zadaná v tvare vonkajšej funkcie. Príkaz `fcontour2d` použijeme, ak chceme nakresliť v jednom grafe viac kriviek tých istých argumentov.

Príklad.

```
-->contour(1:5,1:10,rand(5,10),5);
```



4.7. Plocha v rovine

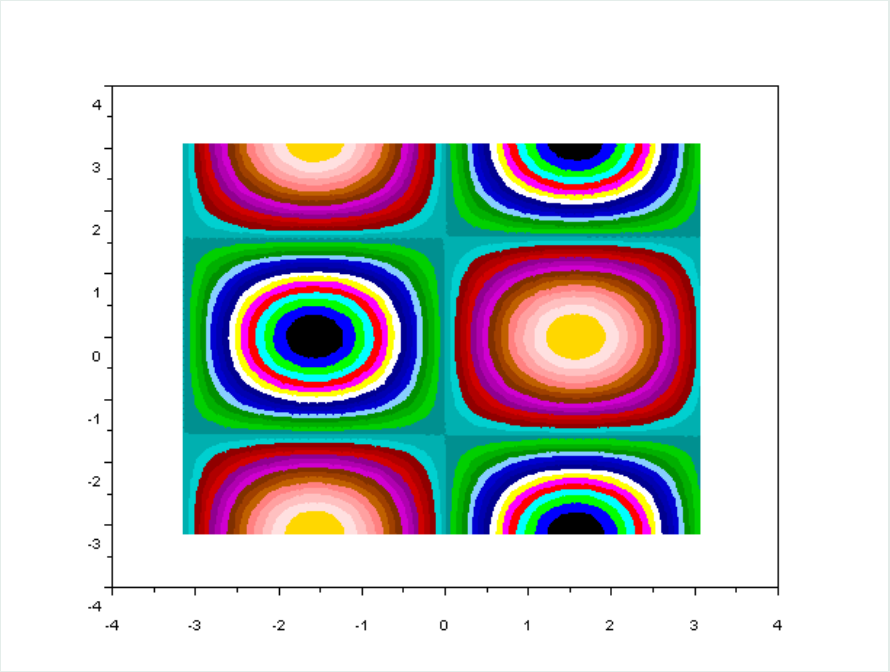
Pri zobrazení „izolínií“ (vrstevníc) môžeme jednotlivé úrovne farebne odlíšiť.

V Scilabe je možné zobrazovať 3D plochy v rovine pomocou farieb. Tzv. „izooblasti“ (oblasti medzi izolíniami) sa zobrazia rovnakými farbami. Takéto grafy je možné vytvoriť pomocou príkazov `grayplot` a `Sgrayplot`[5]. Druhý príkaz sa od prvého líši tým, že prechod farieb je plynulý, bez ostrých hraníc, čo odpovedá plynulej zmene farieb v závislosti od funkčnej hodnoty odpovedajúcej vrstevnice. Ak je funkcia dvoch premenných zadaná zvonku, použijeme príkazy `fgrayplot` a `Sfgrayplot`.

Príklad.

```
t=-%pi:0.1:%pi;
m=sin(t)'*cos(t);
Sgrayplot(t,t,m);
```

[5]Oblasť v skutočnosti nie je šedá (gray je anglicky šedý) ale farebná



4.8. Animácie

Na vytvorenie animácie je potrebná zväčša nejaká postupnosť príkazov, ktorá väčšinou obsahuje cykly a pauzy. Najjednoduchší spôsob je teda zrejme napísať program a ten potom spúšťať. Uvedieme dva príklady animácie „bežiacej“ sínusoidy so zmenšujúcou sa periódou.

Príklad 1.

```
i_beg=1;
i_end=7.5;
step=0.5;
t=0:0.1:2*%pi;
for i=i_beg:step:i_end,
xbasc();      // vyčistí sa okno
plot2d(t,sin(i*t),5)
// vykreslí sa graf funkcie so zmeneným argumentom
xpause(1.e6); // pauza na 1 sekundu
end
```

Príklad 2.

```
kp=xget("pixmap");
xset("pixmap",1);
i_beg=1;
i_end=7.5;
step=0.05;
t=0:0.1:2*%pi;
for i=i_beg:step:i_end,
xset("wwpc");
plot2d(t,sin(i*t),5)
xset("wshow");
end
xset("pixmap",kp);
```

5. Vybrané kapitoly

5.1. Lineárna algebra

Lineárna algebra je snáď najtesnejšie spojená s mnohými inými oblasťami matematiky a je dôležitým nástrojom pri riešení viacerých aplikovaných úloh. Aj keď je v balíku Scilab vydelená špeciálna algebrická knižnica, príkazy vzťahujúce sa k lineárnej algebre sú porozhadzované po iných knižniciach. V tejto časti ukážeme niekoľko najdôležitejších príkazov vzťahujúcich sa k maticovým operáciam a výpočtom a úloh spojených s nimi. Ukážeme tu aj niektoré základné príkazy, ktoré nie sú bezprostredne v knižniciach lineárnej algebry, ale sú nevyhnutné na prácu s maticami.

Prístup k algebrickým, ale aj ostatným základným knižniciam je štandardný, preto pri použití príkazov nie je potrebné uvádzať názov knižnice.

Uvedieme slovensko-anglický slovník niekoľkých kľúčových termínov lineárnej algebry. V zátvorke uvádzame zodpovedajúce príkazy Scilabu:

determinant	determinant (`det`)
identity matrix	jednotková matica (`eye`)
absolute value, magnitude	modul matice (`abs`)
matrix norm	norma matice (`norm`)
inverse of matrix	inverzná matica (`inv`)
pseudoinverse	pseudoinverzná (`pinv`)
dimension	rozmer (dimenzia), typ (`size`)

rank of a matrix	hodnosť matice (`rank`)
sparse matrix	riedka matica (`sparse`)
eigenvalues of matrices	vlastné hodnoty matice (`spec`)
singular value decomposition	singulárny rozklad (`svd`)

5.1.1. Inverzná matica

Spôsob 1. Na určenie inverznej matice použijeme príkaz `inv(X)`. Parameter `X` je reálna, komplexná, polynomická alebo racionálna štvorcová matica. Pre polynomickú maticu môžeme použiť aj príkaz `invr`, výsledok bude rovnaký. Inverzná matica sa hľadá metódou LU-rozkladu a využíva sa knižnica LAPACK.

Príklad 1.

```
-->a=[1 2 3;2 2 3; 5 -1 2];
-->b=inv(a)
```

Výsledok:

```
 b =
! - 1.          1.          -8.327E-17 !
! - 1.5714286   1.8571429   - .4285714 !
!   1.7142857 - 1.5714286     .2857143 !

-->c=a*b
```

Výsledok:

```
 c =
!   1.          0.          1.110E-16 !
! - 8.882E-16   1.          0.        !
!   4.441E-16 - 8.882E-16   1.        !
```

Dostali sme prakticky jednotkovú maticu.

Príklad 2. Určenie inverznej matice k matici zadanej v symbolickom tvare.

```
-->r=[poly([1 2 3],"x","c"),1;poly([1 -2],"x","c"),2]
 r =
! 1 + 2x + 3x2   1 !
!                  !
! 1 - 2x          2 !

-->inv(r)
```

Výsledok:

```
 ans =
!        2              - 1        !
!   -----------      -----------   !
!               2                2 !
!   1 + 6x + 6x      1 + 6x + 6x   !
!                                  !
```

```
!                                  2 !
!   - 1 + 2x         1 + 2x + 3x    !
!   ----------       -----------    !
!               2                2 !
!   1 + 6x + 6x      1 + 6x + 6x    !
```

Ak je matica singulárna alebo „takmer“ singulárna (determinant je veľmi blízky k nule, resp. najmenšie singulárne číslo je veľmi malé) pri hľadaní inverznej matice dostaneme varovnú správu.

Príklad 3. Určenie inverznej matice k singulárnej matici.

```
-->a=[1 2 3;11 22 33; 110 220 330];!
-->inv(a)
     !--error    19
Problem is singular
```

Spôsob 2. Určovanie inverznej matice k polynomickej matici sa vykonáva pomocou príkazu `coffg`. Tento príkaz počíta inverznú maticu metódou ko-faktorov (co-factors method) (GALLUP, NORBECK, 1973). Ako výsledok dostaneme dva parametre `Ns` a `d`, ktoré úplne charakterizujú inverznú maticu. `Ns` je čitateľ (polynomická matica) a `d` je spoločný menovateľ (polynóm). Inverzná matica je teda `Ns/d`.

Príklad.

```
-->r=[poly([1 2 3],"x","c"),1;poly([1 -2],"x","c"),2]
```

```
 r =
! 1 + 2x + 3x2   1 !
!                  !
! 1 - 2x          2 !

-->[a,d]=coffg(r)
```

Výsledok:

```
 d =
1 + 6x + 6x2

 a =
!  2       - 1            !
!                         !
!- 1 + 2x    1 + 2x + 3x2 !
```

Inverzná matica je teda:

```
-->a/d
 ans  =

!        2             - 1        !
!   -----------     -----------   !
!               2                2 !
!   1 + 6x + 6x     1 + 6x + 6x   !
!                                 !
```

```
!                                  2 !
!   - 1 + 2x          1 + 2x + 3x   !
!   ----------        -----------   !
!               2                 2 !
!   1 + 6x + 6x       1 + 6x + 6x   !
```

5.1.2. Pseudoinverzná matica

Pseudoinverznú maticu (označovanú aj A^+) získame použitím príkazu `pinv(A[,tol])`, kde `A` je reálna alebo komplexná matica a `tol` je reálne číslo. Hľadá sa taká matica `X` rovnakého typu ako trasponovaná matica k matici `A`, tak, aby bola splnená podmienka `A*X*A=A, X*A*X=X` a aby matice `A*X` aj `X*A` boli Hermitovské. Výpočet je založený na metóde singulárneho rozkladu (SVD – Singular Value Decomposition) a všetky singulárne hodnoty menšie ako prípustné `tol` sa kladú rovné nule. Tento parameter však v príkaze `pinv` nie je povinný.

Pseudoinverzná matica slúži napr. na riešenie sústav lineárnych rovníc, resp. maticovej rovnice `A*x+b=0` (pozri oddiel 5.1.10) v zmysle metódy najmenších štvorcov, t j. $\|A * x + b\|_2 \longrightarrow \min_x$. Pomocou príkazu `x=pinv(A)*(-b)` dostaneme jedno z riešení (s najmenšou normou 2).

5.1.3. Singulárny rozklad matice

Singulárny rozklad matice môžeme vykonať pomocou príkazu `svd` (singular value decomposition). Syntax je nasledovná:

```
s=svd(X)
[U,S,V]=svd(X)
[U,S,V,rk]=svd(X[,tol])
```

Premenné:

`X` je reálna alebo komplexná matica
`s` je reálny vektor (singulárna hodnota)
`S` je reálna diagonálna matica (singulárna hodnota)
`U,V` sú ortogonálne alebo unitárne štvorcové matice (singulárne vektory).
`tol` je reálne číslo, tolerancia.

Pomocou príkazu `[U,S,V]=svd(X)` sa počítajú tri matice (`X`, `S` a `V`), ktoré majú nasledujúce vlastnosti: matica `S` je diagonálna a ma rovnaký rozmer ako matica `X`. Jej diagonálne prvky sú nezáporné a sú obyčajne zoradené podľa veľkosti. Matice `U` a `V` sú unitárne matice, ktoré spĺňajú podmienku `X=U*S*V'`. Príkaz `s=svd(X)` vracia vektor `s` so singulárnymi hodnotami. Príkaz `[U,S,V,rk]=svd(X[,tol])` vracia doplňujúci parameter `rk`, čo je vlastne hodnosť matice `X`, teda počet singulárnych hodnôt väčších ako hodnota `tol`.

Príklad.

```
-->X=rand(3,2)*rand(2,3)
 X  =
!  0.2536239    0.2743007    0.4459047 !
!  0.1426247    0.1681725    0.2710768 !
!  0.4812589    0.5921015    0.9506647 !
```

```
-->[U,S,V]=svd(X);
-->S
 S  =
!  1.3949496    0.           0.         !
!  0.           0.0240125    0.         !
!  0.           0.           4.224D-17  !

-->U*S*V'
 ans  =
!  0.2536239    0.2743007    0.4459047  !
!  0.1426247    0.1681725    0.2710768  !
!  0.4812589    0.5921015    0.9506647  !
// dostali sme pôvodnu maticu X
```

5.1.4. Vlastné čísla matice

Vlastné čísla matice určujeme pomocou príkazu `spec` (anglicky spectrum). Syntax je

```
evals=spec(A)
[X,diagevals]=spec(A)
```

Premenné:

`A` je reálna alebo komplexná štvorcová matica
`evals` je reálny alebo komplexný vektor, obsahujúci vlastné čísla matice

`diagevals` je reálna alebo komplexná diagonálna matica (vlastné čísla sú na diagonále)

Príkaz `evals=spec(A)` vracia vektor `evals` s vlastnými hodnotami matice `A`. Príkaz `[X,diagevals]=spec(A)` vracia vlastné hodnoty matice `A` v tvare diagonálnej matice `diagevals` a im prislúchajúce vlastné vektory v stĺpcoch matice `X`.

Príklad.

```
-->A=diag([1,2,3]);X=rand(3,3);A=inv(X)*A*X;
-->spec(A)

 ans  =
  !  1.  !
  !  3.  !
  !  2.  !

-->x=poly(0,'x');
-->pol=det(x*eye()-A)

 pol  =
                2   3
 - 6 + 11x - 6x + x

-->roots(pol)
```

```
 ans  =
  !  1.  !
  !  3.  !
  !  2.  !

 -->[S,X]=bdiag(A)

 X  =
! - 0.7534614    1.9951835    2.7200963 !
!   0.5753820  - 0.4412417  - 2.1637336 !
!   0.3181690  - 1.1598936    0.1029292 !

 S  =
!  1.    0.    0. !
!  0.    3.    0. !
!  0.    0.    2. !

-->clean(inv(X)*A*X)

 ans  =
!  1.    0.    0. !
!  0.    3.    0. !
!  0.    0.    2. !
```

5.1.5. Norma matice

Normy matíc sú často nevyhnutné pri analýze maticových algoritmov. Na ich výpočet slúži príkaz `[y]=norm(x [,typ])`

Parametre:

`x` – reálny alebo komplexný vektor alebo matica (typ `full` alebo `sparse`)
`typ` – reťazec (typ normy) (preddefinovaná hodnota je 2).

Norma matíc:

Príkaz `norm(x)` je ekvivalentný príkazu `norm(x,2)` a vracia najväčšiu hodnotu singulárneho rozkladu, t. j. `max(svd(x))`. Príkaz `norm(x,1)` vracia stĺpcovú normu matice, teda `max(sum(abs(x),'r'))`. Príkaz `norm(x, 'inf')`, `norm(x,%inf)` slúži na výpočet riadkovej normy matice `x`, teda `max(sum((x),'c'))`. Pomocou príkazu `norm(x,'fro')` určujeme Frobeniovu normu, t. j. `sqrt(sum(diag(x'*x)))`.

Norma vektorov:

Pre vektory tento príkaz udáva tzv. p-normy. Príkazom `norm(x,p)` určujeme normu `(sum(v(i)^p))^(1/p)`. Príkaz `norm(v)` zodpovedá príkazu `norm(v,2)`, teda prípadu $p = 2$. Príkaz `norm(v,'inf')` vracia hodnotu maximálneho prvku v absolútnej hodnote, teda `max(abs(v))`, t. j. $\max_i |v(i)|$.

Príklad.

```
-->v=[1,2,3];
-->norm(v,1)
 ans  =
    6.          // 6=1+2+3

-->norm(v,'inf')
 ans  =
    3.          // maximálny prvok v absolútnej hodnote

-->A=[1,2;3,4]
-->norm(A,1)
 ans  =
    6.          // stĺpcová norma
```

5.1.6. Funkcia `abs`

Príkaz `abs(A)` slúži na výpočet absolútnych hodnôt prvkov matice `A`, ktorými môžu byť reálne alebo komplexné čísla.

Príklad 1. Výpočet absolútnych hodnôt prvkov reálnej matice.

```
-->a=[1.1   -2.2   3;4.4  -5.4  0]
```

```
 a =
! 1.1  - 2.2   3. !
! 4.4  - 5.4   0. !

-->t=abs(a)
```

Výsledok:

```
 t =
! 1.1   2.2   3. !
! 4.4   5.4   0. !
```

Príklad 2. Výpočet absolútnych hodnôt prvkov komplexnej matice.

```
-->x=[1,%i,2;-1,-%i,1+%i]
 x =
!   1.    i    2.     !
! - 1.  - i    1. + i !

-->t=abs(x)
```

Výsledok:

```
 ans =
! 1.   1.   2.         !
! 1.   1.   1.4142136 !
```

5.1.7. Determinant matice

Spôsob 1. Výpočet determinantu pomocou príkazu `det`, ktorého syntax je `det(X)` alebo `[e,m]=det(X)`. `X` je reálna, komplexná, polynomická alebo racionálna štvorcová matica, `m` je reálne alebo komplexné číslo, `e` je celé číslo, exponent. Výsledkom príkazu `[e,m]=det(X)` je teda hodnota $m \cdot 10^e$. Pre determinant polynomickej matice je príkaz `det(X)` ekvivalentný príkazu `determ(X)` a pre racionálne matice zase príkazu `detr(X)`.

Príklad 1. Vytvoríme polynomickú maticu a určíme jej determinant.

```
-->x=poly(0,'x');
-->a=[x,1+x;2-x,5]
 a=
!x        1 + x!
!              !
!2 - x    5    !

-->D=det(a)
```

Výsledok:

```
D =
                2
 - 2 + 4x + x
```

Príklad 2. Výpočet determinantu reálnej matice.

```
-->s=[1,2.3;-4,5.2];
-->[e,m]=det(s)
-->D=det(s)
```

Výsledok:

```
m = 1.44
e = 1.  // determinant je teda 1.44*10^1,
D=14.4  // čo je vlastne D
```

Spôsob 2. Pomocou príkazu `detr(h)`, kde `h` je polynomická alebo racionálna štvorcová matica. Výpočet je založený na algoritme Leverriera (pozri `help nlev`).

Spôsob 3. Pomocou príkazu `determ(W)`. Tento príkaz slúži len na výpočet determinantov polynomických matíc. Syntax je `r=determ(W,k)`, kde `W` je polynomická štvorcová matica a `k` je celé číslo, ktoré slúži ako horná hranica stupňa polynómu výsledného determinantu matice `W`. Tento parameter nie je povinný a výsledok je potom rovnaký ako pri použití príkazu `det(W)`.

Príklad.

```
-->s=poly(0,'s');
-->w=[s, 1+s^2;s-2, s+4]
```

```
 w  =
!                   2 !
!   s         1 + s   !
!                     !
! - 2 + s     4 + s   !

-->determ(w)
```

Výsledok:

```
 ans =
                 2
    1 + 3s + 3s

-->determ(w,2)
```

Výsledok:

```
 ans =
    5 + 2s
```

5.1.8. Výber diagonály matice

Pomocou príkazu `diag` je možné vybrať z matice vektory tvoriace diagonály matice. Syntax je `y=diag(vm,k)`, kde `vm` je vektor alebo matica, `k` je celé číslo, `y` je vektor alebo matica. Pre maticu `vm` príkaz `y=diag(vm,k)` vracia

vektor, ktorý tvoria prvky k-tej diagonály matice. Príkaz `diag(vm)` zodpovedá príkazu `diag(vm,0)`, vracia teda vektor s prvkami hlavnej diagonály matice `vm`. Hodnoty `k>0` odpovedajú diagonálam nad hlavnou diagonálou a `k<0` pod. Ak je `vm` vektor dĺžky n, tak príkaz `y=diag(vm,k)` vracia štvorcovú maticu rádu $n + |k|$, ktorej k-tu diagonálu tvoria jeho prvky.

```
-->A=[1,2;3,4]
 A =
! 1.  2. !
! 3.  4. !

-->d_1=diag(A) // hlavná diagonála
 d_1 =
! 1. !
! 4. !

-->diag(A,1)
 ans =
    2.

-->diag(A,-1)
 ans =
    3.
```

```
-->v=[1 2];
-->diag(v,-1)
 ans  =
! 0.    0.    0. !
! 1.    0.    0. !
! 0.    2.    0. !
```

5.1.9. Hodnosť matice

Hodnosť matice určujeme príkazom `rank`. Syntax je `[i]=rank(X)`, resp. `[i]=rank(X,tol)`, kde `X` je reálna alebo komplexná matica, `tol` je nezáporné celé číslo, ktoré určuje toleranciu a prejavuje sa v prípade, ak je matica „takmer“ singulárna.

Príklad.

```
-->a=[1 2 3; -6 3 -2;10 20 30]
 a  =
!   1.     2.     3.  !
! - 6.     3.   - 2.  !
!   10.    20.    30. !
-->rank(a)
```

Výsledok:

```
 ans  =
    2.
```

```
-->rank(a,1D-17)
```

Výsledok:

```
 ans  =
    3.
```

5.1.10. Systém lineárnych rovníc

Na riešenie sústav lineárnych algebrických rovníc slúži príkaz `linsolve`. Syntax je `[x]=linsolve(A,b)`, kde `A` je reálna matica typu m/n, `b` je stĺpcový vektor dĺžky m, `x` je reálny vektor. Príkaz `linsolve` určuje všetky riešenia rovnice `A*x+b=0`.

Príklad 1. Riešme sústavu rovníc, ktorá má práve jedno riešenie:

$$x_1 - x_2 + 1 = 0$$
$$-2x_1 - x_2 + 3 = 0$$

```
-->A=[1 -1;-2 -1];
-->b=[1;3];
-->[x]=linsolve(A,b)
```

Výsledok:

```
 x =
!  .6666667  !
!  1.6666667 !
```

Skúška správnosti:

```
-->r=A*x+b
```

Výsledok:

```
 r =
1.0E-15 *
! .3330669 !
! .4440892 !
```

Príklad 2. Riešme sústavu, ktorá má nekonečne veľa riešení:

$$-2x_1 + 6x_2 + 3 = 0$$
$$4x_1 - 12x_2 - 6 = 0$$

```
-->A=[-2 6;4 -12];
-->b=[3;-6];
-->[x]=linsolve(A,b)
```

Výsledok:

```
 x =
!   0.15 !
! - 0.45 !
```

Určuje sa pseudoriešenie, teda $A^+ \cdot (-b)$:

```
-->pinv(A)*(-b)
```

Výsledok:

```
 ans =
!   0.15 !
! - 0.45 !
```

Príklad 3. Riešme sústavu, ktorá nemá riešenie:

$$x_1 + 2x_2 + 1 = 0$$
$$10x_1 + 20x_2 + 20 = 0$$

Výsledok:

```
WARNING:Conflicting linear constraints!
 x =
    []
```

Po takomto výsledku však môžeme ešte určiť pseudoriešenie $A^+ \cdot (-b)$:

```
-->pinv(A)*(-b)
```

Výsledok:

```
 ans  =
! - 0.3980198 !
! - 0.7960396 !
```

V Scilabe existuje možnosť riešenia sústavy lineárných rovníc v symbolickom tvare a to pre sústavy rovníc, ktorých matica je horná trojuholníková. Slúžia na to príkazy `solve` a `trisolve`, ktoré už boli uvedené v oddiele 2.7.

5.2. Interpolácia

Jednou z najčastejších úloh je spracovanie experimentálnych údajov. Ide o fitovanie, aproximáciu, interpoláciu, extrapoláciu a pod. Experimentálne údaje často predstavujú usporiadané dvojice (x_i, y_i), kde y_i sú experimentálne hodnoty nejakej hľadanej funkcie a x_i hodnoty argumentov. Tieto usporiadané dvojice je potrebné pred ďalším spracovaním usporiadať, napríklad podľa veľkosti argumentov. Môžeme to urobiť manuálne, no pri veľkých masívoch je to často nereálne. Žiada sa teda napísať program na automatické usporiadanie týchto údajov. Uvedieme malý príklad programu na takéto sporiadanie:

```
x=[5 3 7 4 11];
y=[50 30 70 40 110];      // v=[x;y] je matica údajov
[xx,k]=gsort(x,"g","i"); // gsort je príkaz na usporiadanie
yy=y(k);
vv=[xx;yy]               // usporiadaná matica
```

Výsledok:

```
 vv  =
!   3.    4.    5.    7.    11.  !
!   30.   40.   50.   70.   110. !
```

Pri spracovaní údajov často potrebujeme preložiť cez dané body krivku, v najhoršom prípade pospájať body lomenými čiarami. V prostredí Scilab

existujú na tento účel prostriedky lineárnej interpolácie (`interpln`) a interpolácie splajnami (`interp` a `splin`).

5.2.1. Úloha interpolácie

Majme na intervale $[a, b]$ zadaných $n + 1$ uzlových bodov $a = x_0 < x_1 < < x_2 < \cdots < x_n = b$. Je zadaných $n + 1$ im odpovedajúcich hodnôt y_i, teda $y_i = f(x_i)$. Úlohou je nájsť polynóm $P_n(x)$ stupňa nanajvýš n taký, aby $P_n(x_i) = y_i$. Takýto polynom vždy existuje práve jeden. Interpolácia slúži na určovanie hodnôt medzi uzlovými bodmi, na výpočet hodnôt mimo intervalu $[a, b]$ slúži extrapolácia. Interpolácia na veľkých intervaloch (t. j. s relatívne malým počtom uzlových bodov) ma však svoje špecifické komplikácie:

- presnosť pri veľkých vzdialenostiach medzi uzlovými bodmi je dosť nízka,
- interpolačné polynómy vysokého stupňa sa na koncoch intervalu interpolácie veľmi „vlnia“ a kazia tak charakter funkcie, čo je potom veľmi dôležité pri následnom derivovaní.

5.2.2. Po častiach lineárna interpolácia

Po častiach lineárnu interpoláciu môžeme uskutočniť pomocou príkazu `interpln`. Syntax je `[y]=interpln(xyd,x)`, kde parameter `xyd` je dvojriadková matica so súradnicami bodov, `x` je vektor nových argumentov a `y` je

vektor hodnôt v týchto uzloch. Príkaz `interpln` interpoluje body matice `xyd` pomocou úsečiek a vracia hodnoty interpolácie v diskrétnych uzloch zadaných vektorom `x`.

Príklad. Urobme po častiach lineárnu interpoláciu nasledujúcich hodnôt

```
-->x=[1 10 20 30 40];  // hodnoty argumentov
-->y=[1 30 -10 20 40]; // namerané hodnoty
-->xyd=[x;y];
-->plot2d(x',y',[-3],"011"," ",[-10,-40,50,50]);
// vykreslí uzlové body
-->x_new=(-4:45);
-->yi=interpln(xyd,x_new);
-->plot2d((-4:45)',yi'); // pospája uzlové body úsečkami
```

Výsledok: Budú určené hodnoty interpolačnej funkcie v 50 bodoch na intervale $[-4, 45]$ s krokom 1.

Pomocou príkazu `z=interpln(xyd,10.5)` vieme určiť hodnotu interpolačnej funkcie v bode $x = 10,5$. Jej hodnota je `z = 28`.

5.2.3. Splajnová interpolácia

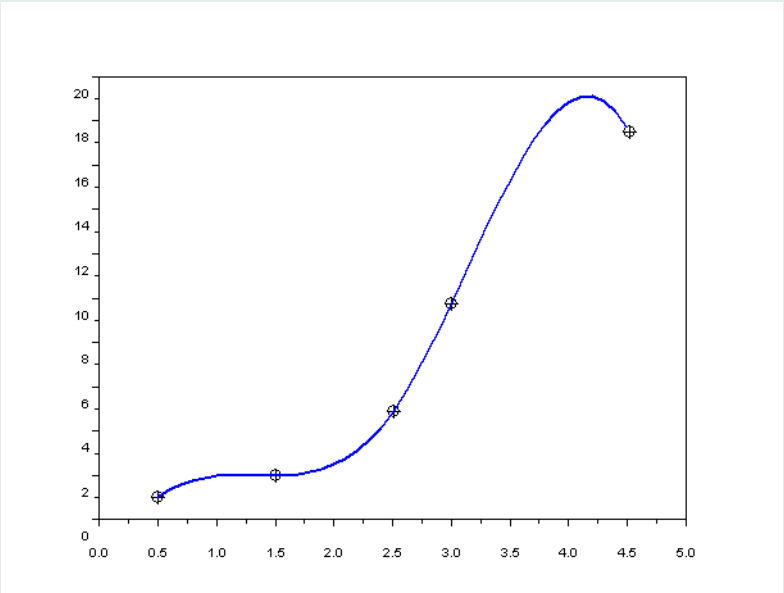
Interpoláciu hodnôt splajnom dosiahneme pomocou príkazu `splin`. Píšeme `d=splin(x,f)`, kde `x` je reálny vektor, ktorého zložky musia byť usporiadané od najmenšej po najväčšiu, `f` je reálny vektor rovnakej dĺžky ako `x`.

Aproximuje sa pomocou kubického splajnu (podrobne pozri `help splin`). Tento príkaz sa obyčajne používa spolu s príkazom `interp`, ktorého syntax je `[s0,s1,s2,s3]=interp(xd,x,f,d)`. Tu sú `x` a `f` tie isté vektory ako v predchádzajúcom príkaze, `xd` je reálny usporiadaný vektor s novými hodnotami argumentov, v ktorých nám príkaz `interp` vracia hodnoty splajnu `d` v podobe vektora `s0`. Vektory `s1`, `s2` a `s3` sú reálne a udávajú hodnoty prvej, druhej a tretej derivácie vektora `s0` v uzlových bodoch `xd`.

Príklad.

```
x=[0.5 1.51 2.51 3. 4.52];
f=[1. 2. 4.9 9.75 17.5];
plot2d(x',f',[-3],"011"," ",[0,0,5,20])
// vykreslí diskrétne body
d=splin(x,f);
xx=0.5:0.1:4.52;
s=interp(xx,x,f,d);
s0=interp(xx,x,f,d); // s0 sú interpolačné hodnoty funkcie
plot2d(xx,[s0'],2); // vykreslí interpolačnú funkciu
```

Výsledok:



Pri interpolácii kubickým splajnom existujú v príkaze aj ďalšie (nepovinné) parametre. Jedným z nich je ešte parameter `typ_splajnu`. Syntax príkazu je `splin(x,y,"typ_splajnu")`. Výber závisí od toho, čo o experimentálnej funkcii vieme. Ukážeme niekoľko základných typov:

- `"not_a_knot"` – je preddefinovaný typ splajnu, ak parameter typu nie je zadaný, používa sa práve tento typ. Je to kubický splajn, ktorý sa hľadá tak, aby sa tretia derivácia zľava rovnala tretej derivácii sprava v druhom a predposlednom uzlovom bode.

- `"clamped"` – používa sa v prípade, ak sú známe prvé derivácie v prvom a poslednom uzlovom bode. Tie sa potom zadávaju v príkaze `splin(x,y,"clamped",[der1,dern])`
- `"natural"` – kubický splajn, ktorý spĺňa podmienku, že druhé derivácie v prvom a poslednom uzlovom bode sú rovné nule.
- `"periodic"`[6] – periodický kubický splajn, ktorý využíva podmienku, že v prvom a v poslednom uzlovom bode sa odpovedajúce prvé a druhé derivácie rovnajú.

Existujú ešte niektoré ďalšie typy ako `"monotone"`, `"fast"` a pod., ktoré je možné bližšie spoznať pomocou `"help spline"`

V tejto časti by sme ešte mali spomenúť, že v prostredí Scilab existujú aj možnosti viacrozmerných interpolácií. Tie však v učebnici neopisujeme. Užívateľ si ich môže pozrieť priamo v Scilabe pomocou `help interp2d`, `help splin2d`, resp. `help interp3d`, `help splin3d`.

5.2.4. Vyhladzovanie pomocou splajnov

Na vyhladzovanie experimentálnych hodnôt vhodnou funkciou môžeme použiť príkaz `smooth`, ktorý zapisujeme `[pt]=smooth(body,krok)`, kde parameter `body` je reálna matica typu `2/n`, ktorej prvý riadok tvorí vektor

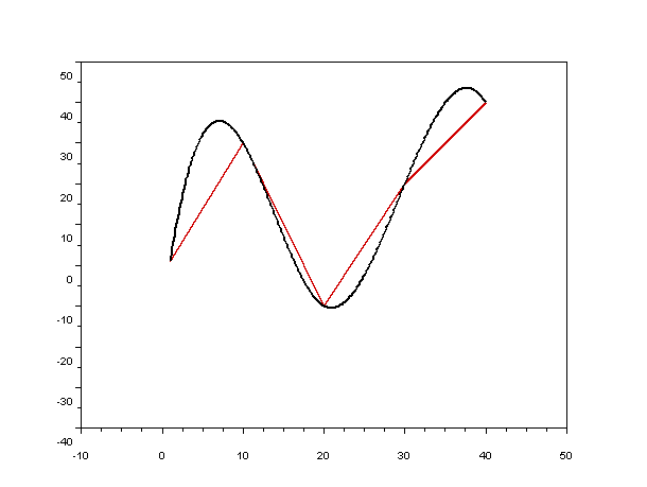
[6] Je vhodný pri interpolácií periodických funkcií

argumentov (musí byť usporiadaný vzostupne) a druhý vektor experimentálnych hodnôt. Parameter `krok` je reálne číslo, ktoré určuje krok diskretizácie osi argumentu. Na vyhladzovanie sa používa kubický splajn, rovnako ako v príkaze `interp`.

Príklad.

```
-->x=[1 10  20 30 40];
-->y=[1 30 -10 20 40];
-->plot2d(x',y',[3],"011"," ",[-10,-40,50,50]);
-->yi=smooth([x;y],0.1);  // yi(1,:)=x(1):0.1:x(5)
-->plot2d(yi(1,:)',yi(2,:)');
```

Výsledok:



5.2.5. Metóda najmenších štvorcov

Na fitovanie experimentálnych hodnôt sa môže použiť príkaz `datafit`. Daný algoritmus je založený na metóde najmenších štvorcov a optimálne opisuje namerané údaje polynómami stupňa aspoň 2. Príkaz obsahuje veľké množstvo rôznych parametrov a podrobnejšie sa dá pozrieť pomocou `help datafit`. My ukážeme na jednoduchých príkladoch jeho použitie v tvare `[aa,chyba]=datafit(G,f,a0)` s parametrami: `G(a,z)` funkcia, ktorá sa hľadá ako vektor najlepších parametrov a pre aproximáciu hodnôt `f` funkciou `sum(a(i)*z(i))`, `a0` sú začiatočné parametre vektora `a`, `aa` je výstupný vektor optimálnych parametrov a `chyba` je skalár, určujúci najmenšiu sumárnu kvadratickú odchýlku.

Príklad 1. Namodelujeme najprv nasledujúcim spôsobom experimentálnu krivku. Vezmeme kvadratickú funkciu a naložíme na ňu nejaký aditívny šum pomocou generátora náhodných čísel.

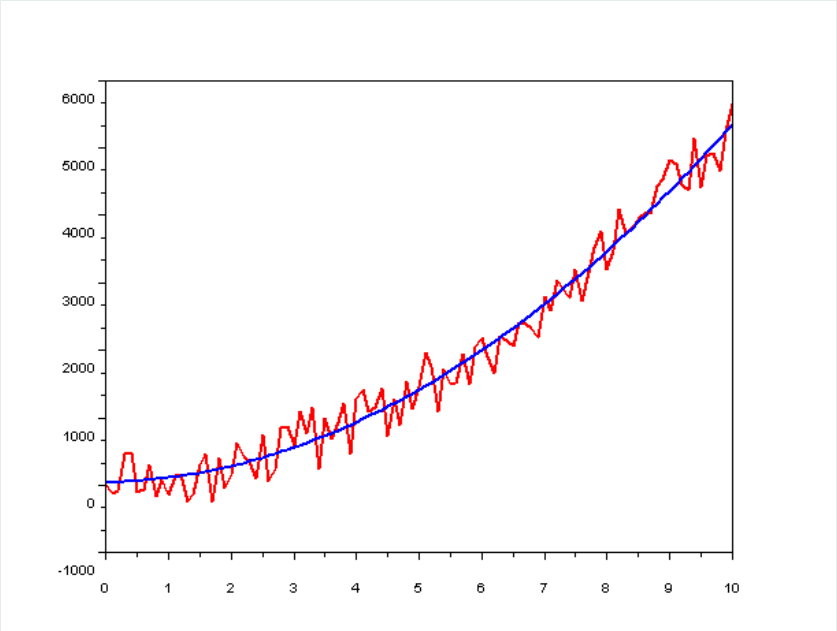
```
x=[0:0.1:10];
y=20+30*x+50*x^2;
e=1000*(rand(1,length(x))-0.5); //šum
y1=y+e;
plot2d(x,y1,[5]); //červená krivka - experimentálne hodnoty
```

Predpokladajme teda, že uzly `(x,y1)` predstavujú namerané hodnoty.

```
f=[x;y1]; // zostavime maticu
```

```
// zadáme funkciu odchýlok
deff("[e]=G(a,z)","e=z(2)-a(1)-a(2)*z(1)-a(3)*z(1)^2");
a0=[1;1;1]; // začiatočné parametre
// pre funkciu G nájdeme hodnoty aa(1),aa(2) č aa(3)
[aa,chyba]=datafit(G,f,a0);
yy=aa(1)+aa(2)*x+aa(3)*x^2;
plot2d(x,yy,[2])
```

Výsledok:



Príklad 2. Pozrieme sa, čo predstavuje hodnota `er`, ktorú vracia príkaz `datafit`.

```
x=[1 2 3 5 9];
y=[3 4 7 9 17];
plot2d(x, y,[5]);
f=[x;y];
deff("[e]=G(a,z)","e=z(2)-a(1)-a(2)*z(1)-a(3)*z(1)^2");
a0=[1;1;1];
[aa,chyba]=datafit(G,f,a0);
yy=aa(1)+aa(2)*x+aa(3)*x^2;
plot2d(x,yy,[2]);
e=(y-yy)^2;d=sum(e);
// vektor kvadratických odchýlok a jeho suma

d,chyba
 d  =
    1.2903226
 chyba  =
    1.2903226
```

Vidíme, že parameter `chyba` je skutočne súčet kvadratických odchýlok.

5.3. Integrovanie

Výpočet určitých integrálov

Spôsob 1. Použitím príkazu `intsplin` sa vypočíta určitý integrál diskrétnej funkcie pomocou interpolačného splajnu. Syntax tohto príkazu je `v=intsplin(x,y)`, kde `x` je usporiadaný reálny vektor argumentov, teda aj premenná, podľa ktorej sa integruje, `y` je reálny vektor rovnakej dĺžky s funkčnými hodnotami, `v` je hodnota určitého integrálu na intervale $[x_0, x_n]$, t. j.

$$v = \int_{x_0}^{x_n} f(x)\mathrm{d}x$$

Príklad. Vypočítajme určitý integrál diskrétnej funkcie $y(x) = x^2$ zadanej na intervale $[1;6]$ s krokom 1. Analytické riešenie je:

$$\int_1^6 x^2\mathrm{d}x = \left[x^3/3\right]_1^6 = 6^3/3 - 1/3 \doteq 71.666667.$$

```
-->x=[1.,2.,3.,4.,5.,6.];
-->y=[1,4,9,16,25,36];
-->v=intsplin(x,y)
 v =
  71.666667
```

Spôsob 2. Integrovanie lichobežníkovou metódou sa uskutočňuje pomocou príkazu `inttrap`. Používa sa v tvare `v=inttrap(x,s)`, kde parametre sú tie isté ako v predchádzajúcom spôsobe.

Príklad. Vypočítajme integrál z predchádzajúceho príkladu lichobežníkovou metódou.

```
-->x=1:1:6;
-->y=x.^2;
-->v=inttrap(x,y)
 v =
   72.5
```

Spôsob 3. Integrovanie pomocou príkazu `integrate`. Ide o výpočet pomocou kvadratúry. Syntax príkazu je `v=integrate('fx','x',x0,xn,[ea,er])`, kde `fx` je funkcia, `x` je integračná premenná, `x0` a `xn` sú hranice integrovania. Nepovinné parametre `ea` a `er` sú nezáporné reálne čísla a znamenajú požadovanú absolútnu a relatívnu chybu. Štandardne sú v Scilabe preddefinované a ich hodnoty sú `0.` a `1.D-8`. Výpočet sa ukončí, keď bude splnená podmienka presnosti `abs(I-v)<=max(ea,er*abs(I))`, kde `I` je presná hodnota integrálu.

Príklad. Vypočítajme opäť ten istý integrál.

```
-->integrate('x.^2','x',1,6)
ans  =
    71.666667
```

Spôsob 4. Integrovanie pomocou príkazu `intg`. Integrálna funkcia musí byť externá alebo diskrétna v tvare reálneho vektora. Jeho syntax je `[v,err]= intg(a,b,f,[ea,er])`, kde `a` a `b` sú hranice integrovania, `f` je externá (alebo diskrétna) funkcia, `ea` a `er` je požadovaná absolútna a relatívna chyba, `v` je hodnota integrálu, `err` je odhad absolútnej chyby výsledku. Výpočet sa ukončí, keď je splnená podmienka ako pri predchádzajúcom spôsobe.

Príklad.

```
-->function y=f(x)
-->y=x*sin(30*x)/sqrt(1-((x/(2*%pi))^2)),endfunction
-->exact=-2.5432596188; // presná hodnota integrálu
-->I=intg(0,2*%pi,f)
 I  =
  - 2.5432596

-->abs(exact-I)
 ans  =
    9.337D-11
```

Poznámka. V prostredí Scilab existujú príkazy na výpočet určitých integrálov komplexných funkcií, pozri `help intc` a `help intl`.

5.4. Riešenie nelineárnych rovníc

5.4.1. Určovanie koreňov polynómu

Pomocou príkazu `roots`. Syntax je `[x]=roots(p)`, kde parameter `p` je polynóm s reálnymi alebo komplexnými koeficientami a `x` je komplexný vektor s koreňmi polynómu `p`. Stupeň polynómu môže byť nanjviac 100.

Príklad 1.

```
-->x=poly(0,"x");
-->p=(x-2)*(x+3)*(x-7);
-->w=roots(p)
```

Výsledok:

```
 w =
!   2. !
! - 3. !
!   7. !
```

Príklad 2.

```
-->a=1;b=2;c=3;
-->p=poly([a b c],"x","c")
```

Výsledok:

```
 p =
1 + 2x + 3x2
```

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

Výsledok:

```
 r =
! - .3333333 + .4714045i !
! - .3333333 - .4714045i !
```

Skúška správnosti:

```
-->q1=1+2*r(1)+3*r(1)^2 // r(1) = prvý koreň
 q1 =
- 1.110E-16

-->q2=1+2*r(2)+3*r(2)^2 // r(2) = druhý koreň
 q2 =
- 1.110E-16
```

Poznámka. Pri polynómoch vysokého stupňa je presnosť výpočtu dosť nízka, pričom pre rôzne korene je rôzna, preto je dobré vykonať skúšku správnosti dosadením koreňov.

5.4.2. Určovanie nulových bodov funkcií

Na riešenie nelineárnej rovnice môžeme použiť príkaz `fsolve`, ktorého povinná syntax je `fsolve(x0,fct)`, kde `x0` je začiatočná hodnota argumentu a `fct` je vonkajšia funkcia. Tento príkaz hľadá nulové body funkcie.

Príklad. Nájdime riešenie rovnice $e^x + x^2 - 5 = 0$.

```
-->deff('y=f1(x)','y=exp(x)+x^2-5');
-->xr=fsolve(5,f1)
```

Výsledok:

```
 xr  =
    1.2411428       // kladný koreň rovnice


-->xr=fsolve(-5,f1)
```

Výsledok:

```
 xr  =
  - 2.2114378       // záporný koreň rovnice
```

Príkaz `fsolve` slúži aj na riešenie sústav nelineárnych rovníc, podrobne pozri `help fsolve`.

5.5. Riešenie diferenciálnych rovníc

Uvedieme niektoré príkazy na riešenie diferenciálnych rovníc:

Spôsob 1. Pomocou príkazu `ode`, ktorý sa používa na riešenie obyčajných diferenciálnych rovníc.

Spôsob 2. Pomocou príkazu `odedc`, ktorý sa používa na riešenie zmiešaných diskrétno-spojitých systémov obyčajných diferenciálnych rovníc (pozri `help odedc`).

Spôsob 3. Pomocou príkazu `dassl`, ktorý sa používa na riešenie implicitne zadaných diferenciálnych rovníc typu $g(t, y, y') = 0$, $y(t_0) = a$ a $y'(t_0) = b$ (pozri `help dassl` alebo tiež podrobný príklad v `SCIDIR/tests/dassldasrt.tst`).

Spôsob 4. Pomocou príkazu `impl`, ktorý sa používa na riešenie implicitne zadaných lineárnych dif. rovníc typu $A(t, y)\mathrm{d}y/\mathrm{d}t = g(t, y)$, $y(t_0) = a$ (pozri `help impl`, prípadne príklad v `SCIDIR/routines/default/Ex-impl.f`).

5.5.1. Riešenie Cauchyho úlohy pre obyčajnú diferenciálnu rovnicu

V tejto časti sa bližšie zoznámime s príkazom `ode`, ktorý slúži na riešenie obyčajnej diferenciálnej rovnice s pravou stranou $\mathrm{d}y/\mathrm{d}t = f(t, y)$, $y(t_0) = y_0$. Syntax príkazu je nasledovná:

```
y=ode(y0,t0,t,f)
[y,w,iw]=ode([typ],y0,t0,t [,rtol [,atol]],f [,jac] [,w,iw])
[y,rd,w,iw]=ode("root",y0,t0,t [,rtol [,atol]],f [,jac],ng,g...
    [,w,iw])
y=ode("discrete",y0,t0,kvect,f)
```

Parametre:

`y0` je reálne číslo alebo matica (začiatočná podmienka)

`t0` je reálne číslo (začiatočný čas, čas inicializácie)

`t` je reálne číslo (udáva koniec časového intervalu výpočtu)

`f` je vonkajší parameter (funkcia alebo reťazec) `typ` je premenná, ktorá nadobúda nasledujúce hodnoty (tie budú opísané nižšie): `"adams"`, `"stiff"`, `"rk"`, `"rkf"`, `"fix"`, `"discrete"`, `"roots"`

`rtol, atol` sú buď reálne konštanty alebo reálne vektory rovnakého typu ako `y`, udávajúce požadovaná relatívnu a absolútnu chybu

`jac` je vonkajší parameter (funkcia alebo reťazec)

`w, iw` sú reálne vektory

`ng` je celé číslo

`g` je vonkajší parameter (funkcia alebo reťazec)

`kvect` je celočíselný vektor

Spôsob riešenia závisí od výberu prvého parametra, ktorý môže byť nasledovný:

`nezadaný argument` – bude sa volať podprogram balíka ODEPACK. Automaticky sa uskutoční výber medzi metódou prediktor-korektor na riešenie normálnych úloh (nonstiff predictor-corrector) a metódou BDF (Backward Differentiation Formula) Adamsa-Moultona pre tuhé (stiff) úlohy. Najprv sa použije metóda pre normálne úlohy s dynamickou kontrolou výsledkov, potom sa uskutoční výber metódy.

`"adams"` – volá sa podprogram balíka ODEPACK a použje sa Adamsova metóda.

`"stiff"` – volá sa podprogram balíka ODEPACK a použje sa metóda BDF (Backward Differentiation Formula) na riešenie tuhých úloh.

`"rk"` – použije sa metóda Rungeho-Kuttova štvrtého rádu (RK4).

`"rkf"` – použije sa program Shampine a Watts, založený na metóde Rungeho-Kutta-Fehlberga (Fehlberg-Runge-Kutta) štvrtého a piateho rádu (RKF45) s automatickou kontrolou chýb. Táto metóda sa používa na riešenie normálnych (non-stiff) a mierne tuhých (mildly stiff) úloh, ak nás prvá derivácia nezaujíma. Neodporúča sa však v prípadoch, keď je potrebná vysoká presnosť riešenia.

`"fix"` – používa sa ten istý podprogram ako pri `"rkf"`, avšak nie je potrebné zadávať také veľké množstvo parametrov. Pre začínajúceho užívateľa je to najjednoduchší spôsob.

`"root"` – používa sa podprogram balíka ODEPACK, ktorý je schopný nachádzať nulové body funkcií. Podrobne pozri `help ode_root`.

`"discrete"` – modelovanie v diskrétnom čase (Discrete Time Simulation). Podrobne pozri `help de_discrete`

Poznámka. Pojem tuhé (z anglického stiff) diferenciálne rovnice sa používa v numerickej matematike pri riešení dif. rovníc a tiež pri integrovaní (pozri

napr. (PAUL DAVIS, 1995; CLEVE MOLER, 2003)).

Príklad. Určme približné numerické riešenie diferenciálnej rovnice $\mathrm{d}y/\mathrm{d}t = ty^{(1/3)}$ so začiatočnou podmienkou $y(0) = 1$ na intervale $[t_0; t_{\max}] = [0; 5]$ s krokom $\mathrm{d}t = 0.01$.

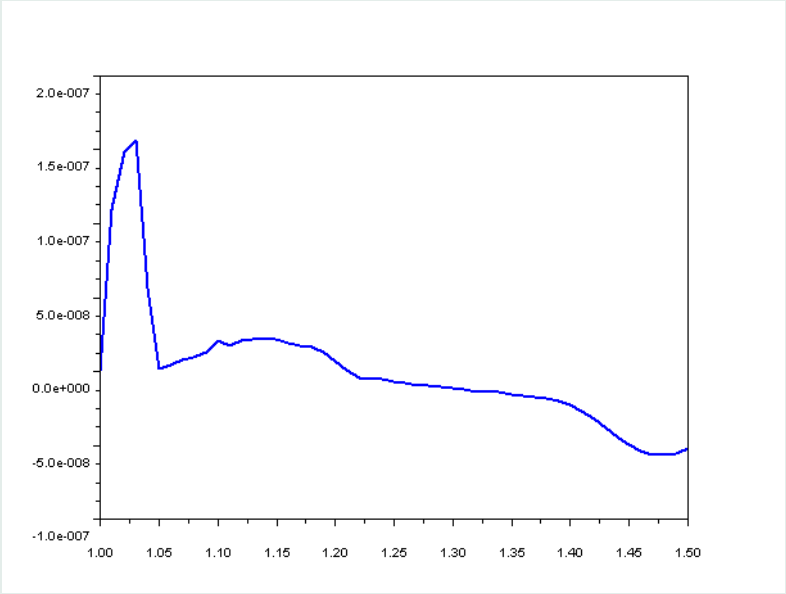
Keďže existuje analytické riešenie tejto rovnice, ktoré je

$$y(t) = ((t^2 + 2)/3)^{(3/2)},$$

možeme ho porovnať s numerickým riešením Scilabu.

```
-->y0=1;
-->t0=1;
-->t=1:0.01:1.5;
-->deff("[ydot]=f(t,y)","ydot=y^(1/3)*t")
-->y=ode(y0,t0,t,f);
//
-->y_exact=((t^2+2)/3)^(1.5); // presné riešenie rovnice
-->my_er=y-y_exact;
-->plot(t,my_er) // graf odchýlky od presného riešenia
```

Výsledok:



Príkaz `ode` môže obsahovať veľké množstvo rôznych parametrov. Preto je často výhodné zapĺňať tieto parametre v dialógovom režime. Na otvorenie dialógového okna slúži príkaz `odeoptions()`. Podrobný popis všetkých parametrov, ktoré obsahuje tento režim môžeme nájsť pomocou `help odeoptions`.

5.6. Štatistika

5.6.1. Výberový priemer

Výberový priemer (výberovú strednú hodnotu) môžeme určiť pomocou príkazu `mean`. Pre diskrétny výber veličín $(x_1, x_2, \ldots, x_n)$ sa výberový priemer rovná aritmetickému priemeru a počíta sa pomocou vzťahu

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Syntax je `y=mean(x)`, `y=mean(x,'r')`, `y=mean(x,'c')`, kde `x` je reálny vektor alebo matica, `y` je skalár alebo vektor. Ak `x` je matica s rozmerom `n_r/n_c`, výberový priemer sa počíta ako súčet všetkých prvkov matice, delený počtom prvkov matice, teda `y=sum(x_ij)/(n_r*n_c)`.

Ak `x` je matica, príkaz `mean(x,'r')`, resp. `mean(x,1)` je „riadkový“ výberový priemer a `mean(x,'c')`, resp. `mean(x,2)` je „stĺpcový“ výberový priemer. Príkaz `mean(x,'r')` (`mean(x,1)`) vracia riadkový vektor, ktorého zložky sú výberové priemery prvkov v stĺpci matice, teda `y(j)= =mean(x(:,j))`.

Príklad.

```
-->x=[1,2,10;7,7.1,7.01]
 x =
! 1.  2.  10.  !
! 7.  7.1 7.01 !
```

```
-->M=mean(x) // (1+2+10+7+7.1+7.01)/6
```

Výsledok:

```
 M =
   5.685

-->M_row=mean(x,'r')
```

Výsledok:

```
 M_row =
! 4.  4.55  8.505 ! // (1+7)/2 (2+7.1)/2 (10+7.2)/1

   -->M_col=mean(x,'c')
```

Výsledok:

```
 M_col=
! 4.3333333 ! // (1+2+10)/3
! 7.0366667 ! // (7+7.1+7.01)/3
```

5.6.2. Vážený priemer

Pomocou príkazu `meanf(x,f)` (weighted mean) sa počíta vážený aritmetický priemer, kde `x` je vektor hodnôt a `f` je vektor váh (napr. početností), obidva majú rovnakú dĺžku. Ak `x` a `f` sú matice, na výpočet

„riadkového“ a „stĺpcového“ váženého aritmetického priemeru použijeme príkazy `meanf(x,f,'r')` alebo `meanf(x,f,1)` a `meanf(x,f,'c')` alebo `meanf(x,f,2)`. Vážený priemer sa počíta podľa vzťahu

$$\overline{y} = \frac{\sum_{i=1}^{n} f_i x_i}{\sum_{i=1}^{n} f_i}.$$

5.6.3. Geometrický priemer

Geometrický priemer kladných hodnôt vypočítame pomocou vzťahu

$$y = \sqrt[n]{\prod_{i=1}^{n} x_i}.$$

V Scilabe na jeho výpočet použijeme príkaz `geomean(x)` (geometric mean), ktorého syntax a parametre sú analogické ako pri výberovom priemere. Ak `x` je matica, na výpočet „riadkového“ a „stĺpcového“ geometrického priemeru môžeme opäť použiť príkazy `geomean(x,'r')` (alebo `geomean(x,1)`) a `geomean(x,'c')` (alebo `geomean(x,2)`).

5.6.4. Medián

Ak usporiadame štatistický súbor, ktorý má počet prvkov n, potom jeho medián je v prípade nepárneho $n = 2k + 1$ hodnota prvku x_{k+1}, v prí-

pade párneho $n = 2k$ je to aritmetický priemer $(x_k + x_{k+1})/2$. Na výpočet mediánu slúžia príkazy `y=median(x)` ak `x` je vektor, `y=median(x,'r')` a `y=median(x,'c')` ak `x` je matica.

Príklad.

```
-->A=[1,2,10;7,7.1,7.01]
 A  =
! 1.    2.     10.   !
! 7.    7.1    7.01 !

-->median(A)
 ans  =
    7.005

-->median(A,'r')
 ans  =
  ! 4.    4.55    8.505 !

-->median(A,'c')
 ans  =
  ! 2.   !
  ! 7.01 !
```

5.6.5. Priemerná odchýlka

Priemerná odchýlka sa počíta ako aritmetický priemer absolútnych hodnôt odchýlok od strednej hodnoty $\overline{x}$, teda

$$s = \frac{1}{n} \sum_{i=1}^{n} |x_i - \overline{x}| .$$

V Scilabe môžeme priemernú odchýlku vypočítať pomocou príkazu `s=mad(x)` (mean absolute deviation), kde `x` je vektor. Ak `x` je matica, tak „riadkové“, resp. „stĺpcové“ priemerné odchýlky vypočítame pomocou príkazov `s=mad(x,'r')` (alebo `s=mad(x,1)`), resp. `s=mad(x,'c')` (alebo `s=mad(x,2)`).

5.6.6. Smerodajná odchýlka

Smerodajná odchýlka sa počíta ako druhá odmocnina rozptylu (disperzie), t. j. podľa vzťahu

$$s_n = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2}.$$

V Scilabe na to použijeme príkaz `s=msd(x)` (mean squared deviation), kde `x` je vektor. Ak `x` je matica, tak „riadkové“, resp. „stĺpcové“ smerodajné odchýlky vypočítame pomocou príkazov `s=msd(x,'r')` (alebo `s=msd(x,1)`), resp. `s=msd(x,'c')` (alebo `s=msd(x,2)`).

5.6.7. Regresná priamka

Na výpočet regresnej priamky[7] použijeme príkaz `rc=regress(x,y)`. Výsledkom bude vektor `rc=[c1,c2]`, pre ktorý bude priamka s rovnicou `y=c1+c2*x` aproximovať diskrétne hodnoty v zmysle metódy najmenších štvorcov.

5.6.8. Korelačný koeficient

Korelačný koeficient vypočítame pomocou príkazu `rk=correl(x,y,fre)`, kde `x` a `y` sú reálne alebo komplexné vektory a `fre` je matica frekvencií (f_{ij} je počet výskytov dvojice (x_i, y_j) delený celkovým počtom dvojíc) typu `length(x)/length(t)`.

Príklad.

```
-->x=[2.5 7.5 12.5 17.5];
-->y=[0 1 2];
-->fre=[.03 .12 .07;.02 .13 .11;.01 .13 .14;.01 .09 .14];
-->rho=correl(x,y,fre)
 rho  =
   0.2097870
```

[7] Máme na mysli tzv. 1. regresnú priamku, 2. regresnú priamku získame keď vymeníme premenné x a y.

5.6.9. Funkcie rozdelenia

Postupnosť náhodných veličín

Postupnosť náhodných veličín môžeme zostrojiť pomocou príkazu `rand`, ktorého syntax a parametre sme už ukázali v druhej kapitole v oddiele 2.5. Pomocou tohoto príkazu môžeme generovať náhodné veličiny s rovnomerným alebo normálnym rozdelením.

Existuje však príkaz, ktorý môže generovať náhodné veličiny s rôznym typom rozdelenia. Tým je príkaz `grand`, ktorého syntax je

```
Y=grand(m, n, dist_typ [,p1,...,pk])
Y=grand(X, dist_typ [,p1,...,pk])
Y=grand(n, dist_typ [,p1,...,pk])
```

Parametre:

`m, n` sú celé čísla určujúce rozmer požadovanej matice `Y`

`X` je matica (použije sa len jej rozmer)

`dist_typ` je symbolická premenná, ktorá určuje typ rozdelenia a môže mať hodnoty `'bin'`, `'nor'`, `'poi'`, `...` (ostatné pozri v nasledujúcej tabuľke)

`p1, ..., pk` sú reálne alebo celé čísla – parametre pre typ rozdelenia

`Y` je výsledná matica typu `m/n`

Podrobný opis príkazu a parametrov je možné pozrieť pomocou `help grand`.

Distribučná funkcia rozdelenia

Hodnoty distribučnej funkcie daného rozdelenia môžeme určiť pomocou príkazu `cdf*` (z anglického cumulative distribution function). V tabuľke uvádzame príkazy pre niektoré typy rozdelenia a tiež príkazy na určenie hodnôt distribučnej funkcie daného rozdelenia.

rozdelenie	skratka	distribučná funkcia
beta	`bet`	`cdfbet`
binomické	`bin`	`cdfbin`
záporné binomické	`nbn`	`cdfnbn`
χ^2 (chí-kvadrát)	`chi`	`cdfchi`
necentrálne χ^2	`chn`	`cdfchn`
exponenciálne	`exp`	`cdfexp`
Fisherovo	`f`	`cdff`
necentrálne Fisherovo	`nf`	`cdffnc`
gama	`gam`	`cdfgam`
normálne (Gaussovo)	`nor`	`cdfnor`
Poissonovo	`poi`	`cdfpoi`
Studentovo	`t`	`cdft`

Podrobný opis príkazov jednotlivých typov rozdelenia a parametrov je možné pozrieť pomocou `help grand`. Opis príkazu pre distribučnú funkciu daného rozdelenia nájdeme pomocou `help cdf*`.

Knižnice štatistiky ponúkajú samozrejme ešte ďalšie možnosti, napr. testovanie štatistických hypotéz a pod, no v porovnaní s MATLABom nie je až tak podrobne rozpracovaná. Zoznam príkazov štatistiky je možné pozrieť v systéme pomoci `Statistic basis`, ktorý nájdeme pomocou príkazu `help statistic`.

Záver

Spomenuli sme tu len malú časť toho čo Scilab dokáže. Vôbec neboli ukázané niektoré ďalšie možnosti, napr.

- rozhranie medzi funkciami jazyka C a Fortran,
- prostriedky optimalizácie,
- prostriedky grafických a sieťových aplikácii,
- balík na modelovanie dynamických systémov,
- balík na spracovanie signálov.

Špeciálne by som ešte chcel upozorniť na možnosť pracovať s PVM – Parallel Virtual Machine, čo umožňuje rozvetvovať výpočty a vykonávať ich súčasne (paralelne) na viacerých procesoroch (počítačoch). PVM je všeobecne dostupná knižnica, ktorá umožňuje riadiť procesy prostredníctvom správ. Reálizácie PVM existujú tak isto pre operačné systémy Linux aj Windows. Zoznámiť sa s PVM je možné na domovskej stránke (PARALLEL VIRTUAL MACHINE HOMEPAGE, 2006).

Literatúra

Documentation and support, On-line help, http://www.scilab.org/product/man-eng/index.html 9

Parallel Basic Linear Algebra Subprograms (PBLAS), http://www.netlib.org/scalapack/html/pblas_qref.html 7

Scalable Linear Algebra PACKage, http://www.netlib.org/scalapack/scalapack_home.html 7

Documentation and support, Matlab/Scilab functions, http://www.scilab.org/product/dic-mat-sci/M2SCI_doc.htm 9

Henrion, J. — Lasserre, J. B. 2006. *Convergent relaxations of polynomial matrix inequalities and static output feedback*, Appeared in the IEEE Transactions on Automatic Control, Vol. 51, No. 2, pp. 192-202 36

The Method of Cofactors, Online Notes, Linear Algebra (Math 2318), http://tutorial.math.lamar.edu/AllBrowsers/2318/MethodOfCofactors.asp 77

Henrion, J. — Lasserre, J. B. 1973. *A method for numerical computation of the cofactors of a matrix*, Chemical Physics Letters, Vol. 22, No. 1, p.161-163 77

Paul Davis, 1995. *CFCs and Stiff Differential Equations at DuPont*, SIAM News, Vol. 28, No. 9, http://www.siam.org/siamnews/mtc/0195111.htm 114

Cleve Moler, 2003. *Stiff Differential Equations*, MATLAB News & Notes, http://www.mathworks.com/company/newsletters/news_notes/clevescorner/may03_cleve.html 114

Pavlova, M. I. 2003. *Rukovodstvo po rabote s paketom Scilab*, http://www.csa.ru/~zebra/my_scilab/index.html 9

Parallel Virtual Machine, homepage, http://www.csm.ornl.gov/pvm/pvm_home.html 124

NÁZOV: Scilab.
AUTOR: Ján Pribiš
POČET STRÁN: 126
VYDANIE: prvé
SADZBA: elektronická, programom pdfTEX.

ISBN 80-8073-655-3