Numerické metody a programování

Lekce 2

Octave, Matlab – vyšší programovací jazyky pro numerické výpočty

základní charakteristiky

- oproti C zjednodušená syntaxe, proměnné není nutno explicitně definovat
- uzpůsobeny pro snadnou práci s vektorovými a maticovými objekty
- obsahují funkce pro grafický výstup a rozsáhlé knihovny pro numerické výpočty
- program, které plně nevyužívá maticovou aritmetiku běží ve srovnání s obdobným programem napsaným v jazyku C velmi pomalu

příklad: program vynásobí dvě matice 3x3 a výslednou matici vypíše na obrazovku

```
#include<stdio.h>
#define DIM 3  //  dimenze matice

main(){
  // reprezentuji matice jako dvourozmerne pole
  double a[DIM][DIM]={{0,0,1},{0,-1,0},{-1,0,0}};
  double b[DIM][DIM]={{0,0,-1},{0,-1,0},{1,0,0}};
  double c[DIM][DIM];
  int i,j,k;
  for(i=0;i<DIM;i++){
    for(j=0;j<DIM;j++){
      c[i][j]=0;
      for(k=0;k<DIM;k++){   // radek "a[i]" nasobi sloupec "b[j]"
     c[i][j]+=a[i][k]*b[k][j];
      }
    }
  }
  for(i=0;i<DIM;i++){
    for(j=0;j<DIM;j++)
      printf("%f ",c[i][j]); // tisk prvku matice
    printf("\n");
  }
}
```

totéž v Octave/Matlabu

```
  a=[0,0,1;0,-1,0;-1,0,0]; # strednik potlacuje vystup
  b=[0,0,-1;0,-1,0;1,0,0];
  c=a*b                    # automaticky zobrazi vysledek
```

práce s vektory/maticemi

```
a=[0,0,1;0,-1,0;-1,0,0] # matice
v=[1;0;-1]              # sloupcový vektor
dolni:krok:horni        # rozsah (range)
-1:0.5:1  # generuje: [-1,-0.5,0,0.5,1]
a(1,3)    # prvek na prvním řádku a třetím sloupci: 1
```

```
a(2,:)     # druhý řádek: [0,-1,0]
a(1:2,1:2)# submatice 2x2 v levém horním rohu: [0,0;0,-1]
reshape(a,1,9)# změna tvaru: [0,0,-1,0,-1,0,1,0,0]
a*v        # součin matice a vektoru: [-1;0;-1]
v.*v       # součin po elementech: [1;0;1]
v'         # transponovaný vektor: [1,0,-1]
v'*v       # skalární součin: 2
zeros(dim1,dim2)  # nulová matice dim1 x dim2
ones(dim1,dim2)   # matice vyplněná jednotkami
eye(dim1,dim2)    # diagonální jednotková matice
rand(dim1,dim2)   # náhodná matice, rovnoměrné rozdělení
randn(dim1,dim2)  # náhodná matice, normální rozdělení
rand("seed",n),randn("seed",n)  # nastaví náhodný generátor

v=linspace(-2,2,5)  # rozsah: [-2,-1,0,1,2]
v>0                 # vektorová relace: [0,0,0,1,1]
(-1).^(mod(v,2))    # generuje: [1,-1,1,-1,1]
v=sin(linspace(-10,50,5)) # generuje:
                    # [0.54402,-0.95892,0.91295,-0.42818,-0.26237]
v(v<0)              # výběr: [-0.95892,-0.42818,-0.26237]
idx=find(v>0)       # indexy: [1,3]
v(idx)              # výběr: [0.54402,0.91295]

inverse(a) # vypočte inverzní matici (totéž co a^-1)
trace(a)   # stopa
det(a)     # determinant
eig(a)     # vlastní čísla
[v,lam]=eig(a)  # vlastní vektory a vlastní čísla
logm(a)    # logaritmus matice
sqrtm(a)   # odmocnina matice
sum(a,dim) # součet vzhledem k dimenzi dim(1 = po sloupcích)
```

přehled základních řídících struktur

příkaz `if`

```
if (podmínka)
  tělo
endif
```

příkaz `while`

```
while (podmínka)
  tělo
endwhile
```

příkaz `for`

```
for var=výraz # postupně dosazuje sloupce
  tělo
endfor
```

```
soucet=0;  # příklad
for i=1:2:100
  soucet=soucet+i;
endfor
```

vektorizace: `soucet=sum(1:2:100)`

funkce

```
function c=prepona(a,b) # funkce vracejici hodnotu
  c=sqrt(a.*a+b.*b);    # aby fungovalo i pro vektory
endfunction
```

po uložení jako soubor "`prepona.m`"

```
x=[1,3]
y=[1,4]
z=prepona(x,y)  # vrací: [1.4142,5]

function [r,fi]=polarni(x,y)  # funkce vrací více hodnot
  r=sqrt(x.^2+y.^2);
  fi=atan2(y,x);
endfunction
```

po uložení jako soubor `polarni.m`

```
vysledek=polarni(1,2)  # vrací pouze "r": vysledek=2.2361
[ra,uhel]=polarni(1,2) # vrací obě hodnoty: ra=2.2361;uhel=1.1071
```

práce se soubory

```
save filename v1 v2  # uloží "v1", "v2" do souboru "filename"
load filename        # načte data ze souboru "filename"
who -file filename   # zobrazí obsah souboru "filename"

save -ascii "filename" var   # uloží "var" v textové podobě
load -ascii "filename"       # načte data z textového souboru
```

identifikátor načtené proměné je odvozen z názvu souboru: `xx.txt -> xx`

příklad:

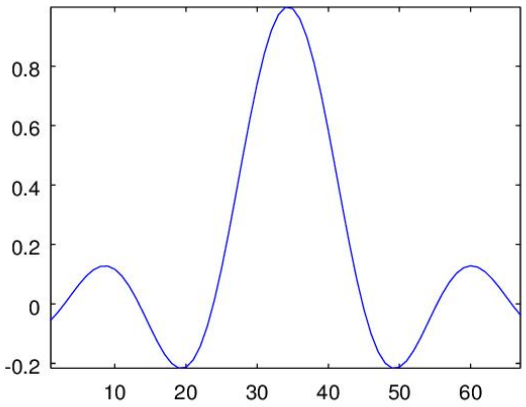
```
x=magic(3)
save -ascii ctverec.txt x
clear
x                   # generuje chybu, proměnná neexistuje
load -ascii ctverec.txt
ctverec             # zde budou uložena přečtená data
```

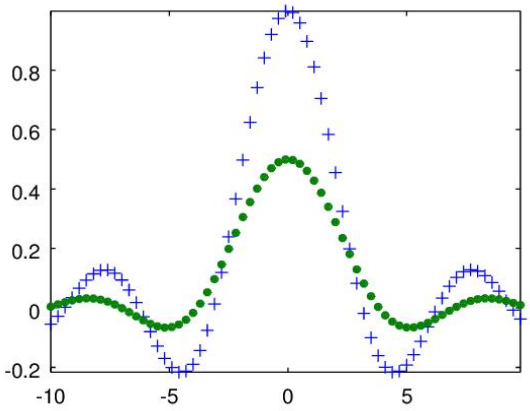
vizualizace

```
plot(x,y,format)  # 2D graf
```

příklad:

```
x=-10.001:0.3:10
y1=sin(x)./x
y2=besselj(1,x)./x
figure(1)  # budu potřebovat více oken
plot(y1)   # hodnoty y1 spojeny
figure(2)
plot(x,y1,"+",x,y2,".") # hodnoty y1 a y2 zobrazeny jako symboly
```

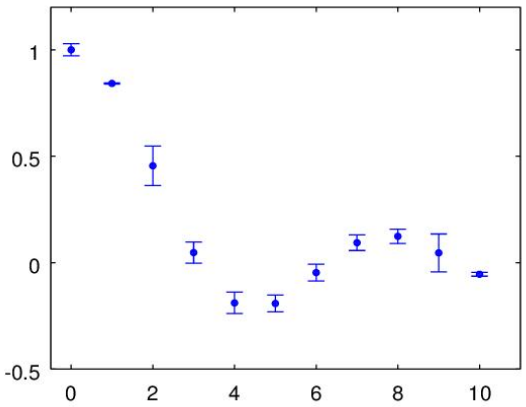





```
errorbar(x,y,er,format)  # 2D graf s chybovými intervaly
```

příklad:

```
x=-10.001:1:10
y1=sin(x)./x
chyba=rand(1,length(x))/10  # generuje "malé" chyby
errorbar(x,y1,chyba,"o")
xlim([-0.5,11])  # specifikace rozsahu na ose x
ylim('manual')
ylim([-0.5,1.2])
```



`mesh(x,y,z)/surf(x,y,z)/pcolor(x,y,z)  # 3D graf, densitogram`

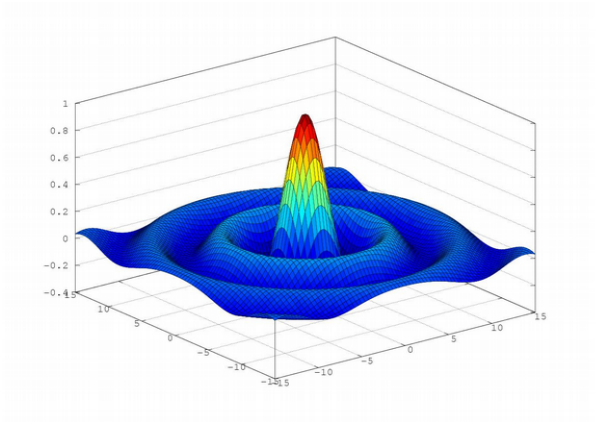
- x, y, z jsou matice
- prvek `z(i,j)` obsahuje výšku kresleného vrcholu
- odpovídající souřadnice x a y jsou uloženy v `x(i,j)` a `y(i,j)`
- matice `x,y` lze generovat z rozsahů pomocí funkce `meshgrid`

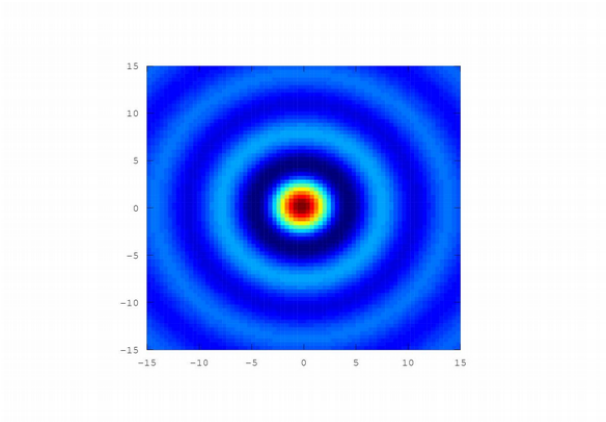
např.
`[xx,yy]=meshgrid(1:3,1:2)` generuje

```
xx =
   1   2   3
   1   2   3
yy =
   1   1   1
   2   2   2
```

příklad:

```
[x,y]=meshgrid(-15:0.3:15)
r=sqrt(x.^2+y.^2)
z=sin(r)./r
figure(1)
surf(x,y,z)
print -djpg surf.jpg   # tisk obrázku do souboru
figure(2)
pcolor(x,y,z);       # kreslí densitogram
shading("interp")    # zapne vyhlazování, potlačí hrany pixelů
axis("square")       # základna bude čtverec
```





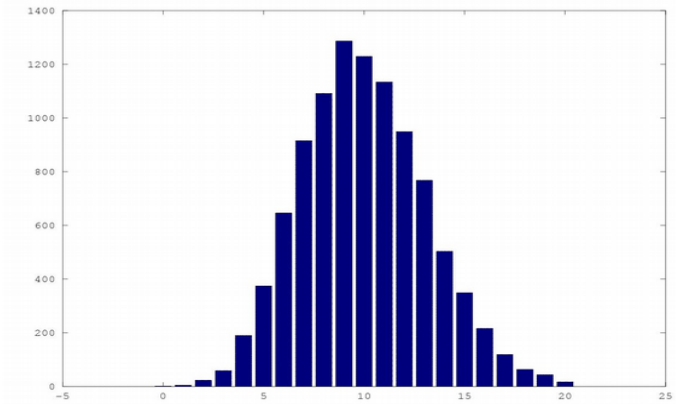
příklady práce s Octave/Matlab

výpočet histogramů pro soubor `5x10000` náhodných čísel

```
function histogramy
  MAX=20
  load -ascii random.txt  # nacte matici nahodnych cisel
  size(random)
  histogram=zeros(5,MAX+1);  # cetnosti nuly az MAX
```

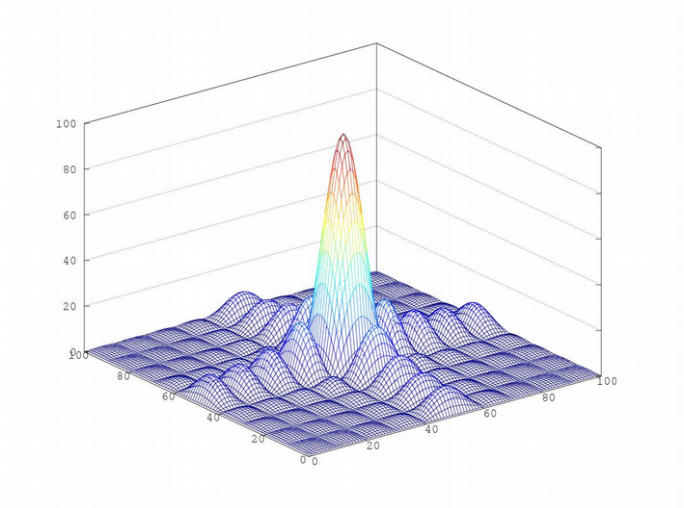
```
  for i=0:MAX                  # pocitam cetnost hodnoty "i"
    kde=(random==i);       # kde_i ma "1" na miste "i" jinak nuly
    histogram(:,i+1)=sum(kde,2);  # scita jednicky na radcich
  endfor
  for i=1:5
    bar(0:MAX,histogram(i,1:MAX+1)); # kresli histogram
    pause(2);            # ceka 2 sekundy
  endfor
endfunction
```

poslední zobrazený histogram:



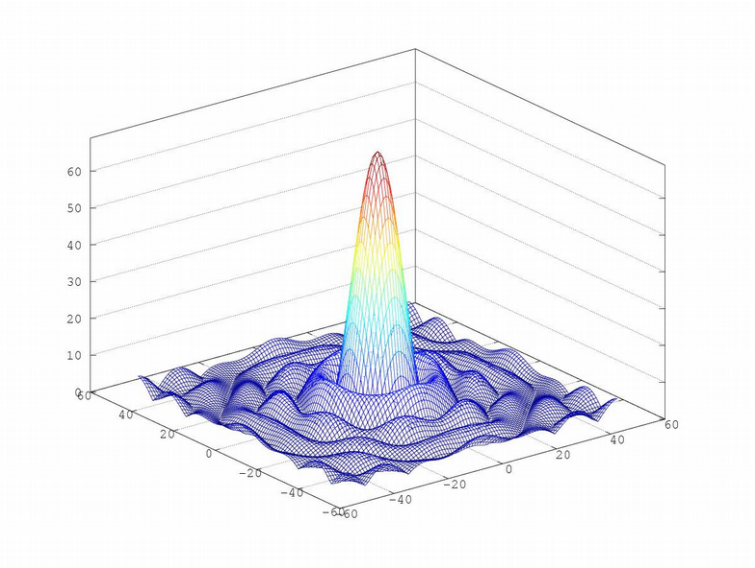
Fraunhoferova difrakce na čtvercovém otvoru

```
function difrakce
  pupila=zeros(100,100);  # propustnost 0=clona, 1=otvor
  ctverec=ones(10,10);
  pupila(1:10,1:10)=ctverec; # vkladame otvor do pupily
  vystup=fftshift(fft2(ifftshift(pupila))); # Fraunh. difrakce
  obrazec=abs(vystup);     # budeme kreslit abs. hodnotu
  [x,y]=meshgrid(1:100);  # generujeme matice souradnic x a y
  mesh(x,y,obrazec);             # kreslime plochu
  print -djpg difrakce1.jpg
endfunction
```



Fraunhoferova difrakce na kruhovém otvoru

```
function difrakce2
  [x,y]=meshgrid(-50:1:50);
  pupila=sqrt(x.^2+y.^2)<5; # uvnitr kruhu 1 jinak 0
  obrazec=abs(fftshift(fft2(ifftshift(pupila))));
  mesh(x,y,obrazec);
  print -djpg difrakce2.jpg
endfunction
```



dvoučočkový objektiv: ohniskové vzdálenosti členů jsou čteny ze souboru, program počítá ohniskovou vzdálenost pro různé vzdálenosti členů

```
function ohnisko
  close all
  load -ascii "objektiv.txt"
  mezera=linspace(0.01,1500,50); # 50 ruznych mezer
  f1=objektiv(1); f2=objektiv(2);
  f=f1*f2./(f1+f2-mezera);
  figure(1)
  h=plot(mezera,f,"b");     # kresli graf
  set(h,'linewidth',2);     # tloustka cary
  axis('tight');
  xlabel("mezera");         # popiska osy
  ylabel("ohniskova vzdalenost")
  xlim([-100,1600]);        # rozsah osy x
  ylim([400,2100]);
  grid on;                  # mrizka
  print -dpdf "graf.pdf"    # vytvari pdf
endfunction
```

např. objektiv ze dvou stejných spojných čoček

```
objektiv.txt
1000
1000
```

funkce `ohnisko` vytvoří výstup