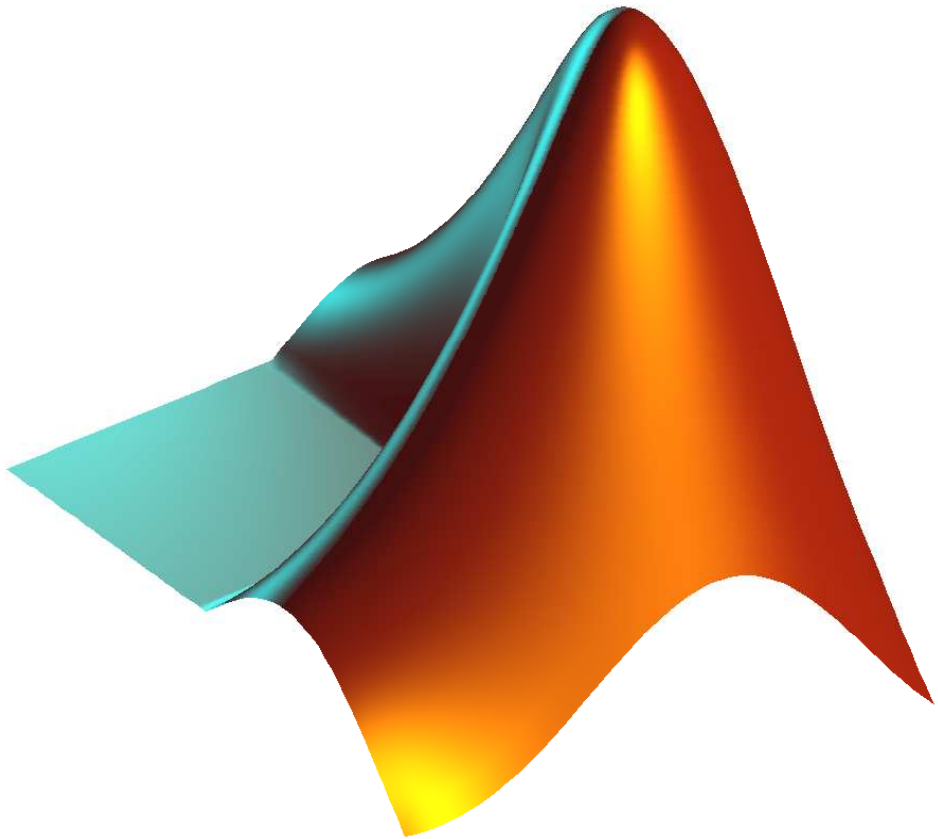
Osnove grafike in napotki za hitro in ekonomično računanje

Grafika

Zelo dober priročnik za osnove grafike dobite na naslovu
http://www.mathworks.com/access/helpdesk/help/techdoc/ref/f16-6011.html



Slika: Primer slike v Matlabu (logotip).

Nekaj osnovnih ukazov

- `plot`, `plot3d`, `plotyy`, `polar`, `contour...`, risanje dvodimenzionalnih objektov, krivulj v treh dimenzijah in polarnih grafov ter kontur
- `surf`, `mesh`, `contour3d`
- `colormap`, `axis`, `xlabel`, `ylabel`,...nadzor nad grafiko.

Primer

Primer 3D grafa

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z)
colormap hsv
axis([-3 3 -3 3 -10 5])
```

Splošni napotki

Pri programiranju v Matlabu se moramo zavedati nekaj splošnih navodil:

- Matlab je interpreterski jezik, zato je preudarno programiranje še posebej pomembno.
- Če je le mogoče uporabljajmo Matlabove vgrajene ukaze.
- Uporabljajmo vektorsko/matrični zapis in ustrezne matrične operacije.
- Pri računanju s polji uporabljajmo operator . (pika).

Nekaj poučnih primerov

Oglejmo si nekaj primerov, ki pokažejo, kako velike so lahko razlike v času izvajanja programa za različne implementacije.

- Naj bosta $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. Izračunati želimo vsoto

$$s = \sum_{i=1}^{n} x_i \, y_i.$$

Najpogostejša rešitev je

```
s=0;
for i=1:n
  s=s+x(i)*y(i);
end
```

Izkaže se, da ni najboljša, zato poskusimo z izboljšavo

```
z=x.*y
s=sum(z);
```

Najboljša rešitev pa je kar skalarni produkt

```
s=x*y'
```

- Poiskati želimo največji element v matriki $A \in \mathbb{R}^{m \times n}$. Ponovno izvedba

```
M=-inf;
for i=1:m
  for j=1:n
    if A(i,j)>M
      M=A(i,j);
    end
  end
end
```

ni najboljša. Bistveno krajša in hitrejša je

```
M=max(max(A));
```

- Izračunati želimo prvih n členov Fibonaccijevega zaporedja $f_1 = 1, f_2 = 1, f_{n+1} = f_n + f_{n-1}$. Napisali bomo dve verziji, eno brez inicializacije vektorja `f`, drugo pa z inicializacijo. Torej

```
f(1:2)=1;
for i=3:n
  f(i)=f(i-1)+f(i-2);
end
```

in

```
f=ones(1,n);
for i=3:n
  f(i)=f(i-1)+f(i-2);
end
```

- Čeprav prejšnji primeri večinoma kažejo, da se moramo izogibati nepotrebnim stavkom `for`, to včasih ne drži. Ponovno se vrnimo k Fibonaccijevemu zaporedju. Izračunati želimo n-ti člen. Verzija s stavkom `for` je očitna

```
f1=1;
f2=1;
for i=3:n
  f3=f2+f1;
  f1=f2;
  f2=f3;
end
```

Rekurzivna izvedba pa neuporabna

```
function f=fib(n)
%FIB racuna n-ti clen Fibonaccijevega zaporedja
%rekurzivno (NEUPORABNO!)
%f=FIB(n)
%n je indks clena (zacnemo z 1)
%f je rezultat

if n==1
  f=1;
elseif n==2
  f=1;
else
  f=fib(n-1)+fib(n-2);
end
```

- Koliko elementov v matriki $A \in \mathbb{R}^{m \times n}$ je večjih od c?

```
stevec=0;
for i=1:m
  for j=1:n
    if A(i,j)>c
      stevec=stevec+1;
    end
  end
end
```

Elegantna izvedba v enem stavku je

```
stevec=sum(sum(A>c));
```

Oglejmo si klasično izvedbo LU razcepa z delnim pivotiranjem

```
function [L,U,P]=lu_delno(A)
%LU_DELNO je LU razcep z celnim pivotiranjem
%[L,U,P]=LU_DELNO(A)
%A je vhodna matrika
%L je spodnja trikotna z enkami na diagonali
%U je zgornja trikotna
%P je permutacijska

n=size(A,1);
P=eye(n);
L=zeros(n);
U=zeros(n);
```

```
for i=1:n-1
  [M,maxi]=max(abs(A(i:n,i)));
  maxi=maxi+i-1;%poravimo na globalni indeks
  P([i,maxi],:)=P([maxi,i],:);
  A([i,maxi],:)=A([maxi,i],:);
  for k=i+1:n
    A(k,i)=A(k,i)/A(i,i);
  end
  for v=i+1:n
    for s=i+1:n
      A(v,s)=A(v,s)-A(v,i)*A(i,s)
    end
  end
end
L=tril(A,-1)+eye(n);
U=triu(A);
```

Glavni del funkcije lahko zamenjamo z

```
for i=1:n-1
  [M,maxi]=max(abs(A(i:n,i)));
  maxi=maxi+i-1;%poravimo na globalni indeks
  P([i,maxi],:)=P([maxi,i],:);
  A([i,maxi],:)=A([maxi,i],:);
  A(i+1:n,i)=A(i+1:n,i)/A(i,i);
  A(i+1:n,i+1:n)=A(i+1:n,i+1:n)-A(i+1:n,i)*A(i,i+1:n);
end
L=tril(A,-1)+eye(n);
U=triu(A);
```