Metody numeryczne I

Wstęp do GNU Octave

Janusz Szwabiński

szwabin@ift.uni.wroc.pl

Konsultacje

wtorek	$16^{00} - 18^{00}$
pokój	514

Plan wykładu

- Programy wspomagające obliczenia
- Dokładność w obliczeniach numerycznych
- Złożoność obliczeniowa algorytmów
- Układy równań liniowych
- Interpolacja i aproksymacja funkcji
- Całkowanie numeryczne funkcji
- Różniczkowanie numeryczne funkcji
- Równania nieliniowe
- Zagadnienia własne
- Równania różniczkowe zwyczajne

Literatura

- Å. Björck, G. Dahlquist, “Metody numeryczne”
- S. E. Koonin, “Computational Physics”
- J. Stoer, “Wstęp do metod numerycznych”
- Z. Fortuna, B. Macukow, J. Wąsowski, “Metody numeryczne”
- D. Potter, “Metody obliczeniowe fizyki”
- T. Pang, “Metody obliczeniowe w fizyce”
- W. H. Press, S. A. Teutolsky, W. T. Vetterling, B. P. Flannery, “Numerical Recipes in . . .” (http://www.nr.com)
- K. A. Ross, Ch. R. B. Wright, “Matematyka dyskretna”

GNU Octave

- Podstawy
- Zmienne i operatory
- Wektory i macierze
- Funkcje
- Pętle i wyrażenia warunkowe
- Operacje wejścia/wyjścia
- Wykresy funkcji
- Równania nieliniowe
- Całkowanie numeryczne

Podstawy

Zasoby w sieci

- `http://www.octave.org`
- John W. Eaton, "GNU Octave"
- Hubert Selhofer, "Einführung in GNU Octave"

Pomoc

- `help name` - informacja o zmiennej lub funkcji `name`
- `help` - lista wszystkich dostępnych poleceń i zmiennych

- skrypty to pliki o rozszerzeniu `*.m`
- polecenia rozdzielamy za pomocą "`;`" lub "`,`"
- "`...`" oznacza, że wyrażenie jest kontynuowane w nowej linii
- komentarze zaczynamy znakiem "%"
- Octave rozróżnia między małymi i dużymi literami

Zmienne i operatory

- przyporządkowanie wartości

```
varname = expression
```

- operatory arytmetyczne

```
+ - * / ^
```

- operatory logiczne

```
== < > <= >= <> & | ~
```

- pomocne polecenia

`whos`, `clear`, `type`

Wektory i macierze

- wektor wierszowy $\vec{v} = (123)$

```
v = [ 1 2 3 ]
```

- wektor kolumnowy $\vec{v} = (123)^{\mathrm{T}}$

```
v = [ 1; 2; 3 ]
```

- automatyczna generacja

```
v = 3:0.1:6
```

- definiowanie macierzy bezpośrednio ...

```
A = [1 2; 3 4]
```

- ...oraz za pomocą innych macierzy

```
b = [5; 6]
M = [A b]
```

- kilka przydatnych poleceń

```
eye(m,n), zeros(m,n)
ones(m,n), rand(m,n)
```

- dodawanie, odejmowanie i mnożenie macierzy

```
+, -, *
```

- transpozycja macierzy

```
A.'
```

- sprzężenie hermitowskie

```
A'
```

- działania element po elemencie

  `.*`, `./` itd.

- indeksowanie

`v(k)`	-	k–ty element wektora $\vec{v}$
`A(k,l)`	-	element A_{kl} macierzy A
`v(m:n)`	-	„wycinek” wektora $\vec{v}$
`A(k,:)`	-	k–ty wiersz macierzy A
`A(:,l)`	-	l–ta kolumna macierzy A

- długość wektora

```
length(v)
```

- rozmiar macierzy

```
[wiersze,kolumny]=size(A)
```

- zmiana postaci

```
reshape(A,m,n)
diag(A), diag(v), diag(v,k)
A(k,:)=w, A(:,l)=[]
```

- równania liniowe $\mathbf{A}\vec{x} = \vec{b}$

```
A\b
```

- przydatne funkcje macierzowe

```
B=inv(A)
E=eig(A), [V;D]=eig(A)
norm(A), any(A)
svd(A), cond(A)
```

Funkcje definiowane przez użytkownika

```
function ret-var = name(arg-list)
    body
endfunction
```

Przykład

```
function y = fun(x)
    y=cos(x)+x;
endfunction
```

Pętla `for`

```
for var = expression
    body
endfor
```

Przykład

```
fib = ones(1,10);
for i=3:10
    fib(i) = fib(i-1) + fib(i-2);
endfor
```

Pętla `while`

```
while (condition)
    body
endwhile
```

Przykład

```
fib = ones(1,10);
i = 3;
while (i<= 10)
    fib(i) = fib(i-1) + fib(i-2);
    i++;
endwhile
```

Wyrażenia warunkowe

```
if (condition)
    then-body
else
    else-body
endif
```

```
if (condition)
    then-body
elseif (condition)
    elseif-body
else
    else-body
endif
```

Przykład

```
if (rem(x,2) == 0)
    printf("x jest parzyste \n");
else
    printf("x jest nieparzyste \n");
endif
```

Operacje wejścia/wyjścia

- `save nazwa_pliku var1 [var2 ...]`
- `load nazwa_pliku`
- `fopen`, `fclose`, `fprintf` ...

Wykresy 2D

- `plot(x,y [,fmt])`
- `semilogx(x,y [,fmt])`
- `semilogy(x,y [,fmt])`
- `loglog(x,y [,fmt])`

Przykład

```
x = -10:0.1:10;
y = sin(x).*exp(-abs(x));
plot(x,y); grid; replot
```

Wykresy 3D

Przykład

```
x = -2:0.1:2;
[xx,yy] = meshgrid(x,x);
z = cos(xx.^2 -yy.^2);
mesh(x,x,z)
```

Równania nieliniowe: $f(x) = 0$

```
[x,info] = fsolve(fun,x0)
```

Przykład

$$
\begin{aligned}
-2x^2 + 3xy + 4\sin(y) - 6 &= 0 \\
3x^2 - 2xy^2 + 3\cos(x) + 4 &= 0
\end{aligned}
$$

```
function y = f(x)
    y(1) = -2*x(1)^2 + 3*x(1)*x(2) + 4*sin(x(2))-6;
    y(2) = 3*x(1)^2 - 2*x(1)*x(2)^2 + 3*cos(x(1)) +4;
endfunction
[x,info] = fsolve("f",[1;2]);
perror("fsolve",info)
```

Całkowanie numeryczne

```
[v,ier,nfun,err] = quad(f,a,b,tol,sing)
```