SOUSTAVA NELINEÁRNÍCH ROVNIC F(X) = 0

Newtonova metoda

Příklad: hledáme řešení soustavy rovnic
$x_2 = x_1^3 + 1$
$x_1^2 + x_2^2 = 4$

Soustavu napíšeme ve tvaru F(X) = 0, kde $X = [\ x_1\ ,\ x_2\ ]^T$, $F = [\ f_1(X)\ ,\ f_2(X)\ ]^T$,
$f_1(X) = x_1^2 + x_2^2 - 4$,
$f_2(X) = x_2 - x_1^3 - 1$.

V Matlabu zadáme F jako

```
>> syms x1 x2          % definujeme promenne x1, x2
>> X = [ x1 ; x2]      % vektor promennych x1, x2
>> f1 = x1^2 + x2^2 -4 % funkce f1 promennych x1, x2
>> f2 = x2-x1^3-1      % funkce f2 promennych x1, x2
>> F = [ f1 ; f2 ]     % vektorova funkce F promennych x1, x2
```

Pro Newtonovu metodu nejdřív potřebujeme spočítat Jacobiovu matici funkce F

```
>> J=jacobian(F, X)        % takhle
>> J=jacobian(F, [x1, x2]) % nebo takhle (pak nemusime vytvaret vektor X)
```

a zvolit počáteční aproximaci X0:

```
>> X0 = [1; 2]
```

Pak opakujeme následující 4 kroky:

```
>> J0 = subs( J, {x1, x2}, {X0(1), X0(2)}) % do J dosadime za X hodnotu X0
>> F0 = subs(F, {x1, x2}, {X0(1), X0(2)})  % do F dosadime za X hodnotu X0
>> D0 = J0 \ (- F0)                        % resime soustavu J0 * D0 = - F0
>> X0 = X0 + D0                            % opravime aproximaci X0
```

Pro Octave bez symb. proměnných

```
F = @(x) [x(1)^2+x(2)^2-4; x(2)-x(1)^3-1];
dF = @(x) [2*x(1) , 2*x(2); -3*x(1) , 1];  % Jacobiova matice F

X=[1; 2];
K = 3;
for k=1:K
  d = -dF(X)\F(X);
  X = X+d;
end

printf('\nVysledne X po %d iteracich: \n', K);
disp(X)
printf('\nF(X): \n');
disp(F(X))
```

Prostá iterační metoda (PIM)

Řešíme stejnou úlohu jako výše, případně pozměněnou úlohu, kde první rovnici nahradíme rovnicí
$x_2 = \sin(x_1) + 1$

V následujícím skriptu si zkuste měnit hodnoty proměnných pit (počet iterací), X (výchozí iterace) a alfa (pomocný parametr, kterým ovlivňujeme konvergenci).

```
F = @(x) [x(1)^2+x(2)^2-4; x(2)-x(1)^3-1];
%F = @(x) [x(1)^2+x(2)^2-4; x(2)-sin(x(1))-1]; % lepsi alfa = 0.25;

alfa = 0.1; % pro zlepseni konvergence - empiricky odhad
G = @(X) X - alfa*F(X);  % transformace F(x) = 0 na x = G(x)

X=[1; 2];
K = 30;  % konverguje mnohem pomaleji nez Newt. met.
for k=1:K
  X = G(X);
end

printf('\nVysledne X po %d iteracich: \n', K);
disp(X)
printf('\nF(X): \n');
disp(F(X))
```