Riešenie nelineárnych rovníc I

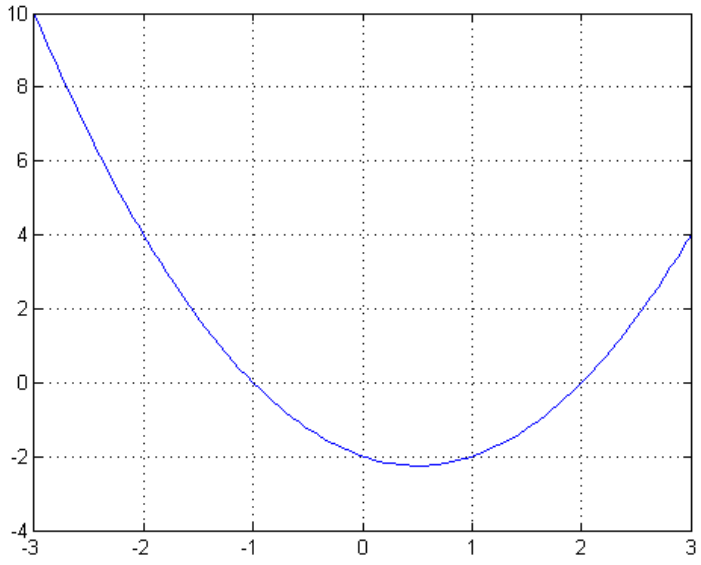
Ako je už zo samotného názvu hodiny parné budeme sa venovať spôsobom výpočtu nelineárnych rovníc. Prečo je riešenie takýchto rovníc nevyhnutné? Nielen v samotnom chemickom inžinierstve, ale takmer v každej oblasti inžinierskych výpočtov sa stretneme s problém, ktorý je zapísaný vo forme nelineárnych rovníc. Lineárne rovnice, ktorých riešenie sme si ukázali v minulom semestri, bolo pomerne jednoduché v porovnaní s riešením nelineárnych rovníc. Je to spôsobené hlavne skutočnosťou, že nelineárne rovnice môžu nadobúdať skutočne rozmanité tvary. Čo sa týka metódy na riešenie nelineárnych rovníc neexistuje jednoznačný, univerzálny a vždy použiteľný postup riešenia. Z tohto dôvodu existuje veľké množstvo metód určených na riešenie nelineárnych rovníc. Preto sa budeme tejto tematike venovať na najbližších dvoch hodinách, počas ktorých si ukážeme základné postupy riešenia nelineárnych rovníc.

Priame hľadanie riešenia – grafický spôsob

Azda najjednoduchší spôsob akým je možné približne určiť riešenie nelineárnych rovníc je priame hľadanie riešenia. Aby tento postup bol čo najpochopiteľnejší, pokúsime sa využiť grafickú závislosť funkcie, ktorú hľadáme.
Predstavme si že chceme vypočítať korene nelineárnej rovnice x^2-x-2=0.

V skutočnosti ide o kvadratickú rovnicu ktorá sa dá veľmi pohodlne vypočítať pomocou funkcie **root**. Naviac každú kvadratickú rovnicu sme schopný vypočítať aj analytický. Konkrétne táto rovnica má dva korene $x1$=2 a $x2$=-1.
No je to nelineárna rovnica, preto si postup riešenia priamym hľadaním môžeme vyskúšať aj na tejto rovnici. Kedže už samotný nadpis hovorí, že budeme používať grafické zobrazenie priebehu funkcie, v prvom kroku si danú závislosť nakreslíme:

```
x=linspace(-3,3,100);
f=x.^2-x-2;
plot(x,f);
grid on
```



Riešenie danej rovnice predstavujú dva body, v ktorých krivka pretína x–ovú os.
Takto nakreslený graf nám automatický poskytne informáciu o tom, aké sú korene danej rovnice. Má to však niekoľko nevýhod:

- presnosť, s akou sme schopný získať hodnotu koreňov je priamo úmerná počtu bodov, pomocou ktorých sme danú závislosť vykreslili
- ďalšia nevýhoda súvisí s našimi možnosťami presne z daného grafu odčítať hodnotu, pri ktorej závislosť prechádza x–ovou osou
- tretia nevýhoda súvisí so skutočnosť, že riešenie nelineárnej rovnice je často len určitou časťou riešeného problému, preto pri každom výpočte nelineárnej rovnice by bolo potrebné výpočet zastaviť, odčítať hodnotu z grafu a pokračovať vo výpočte
- azda najväčšia nevýhoda je, že takýmto spôsobom je možné riešiť jednu, maximálne dve rovnice

Práve poslednú nevýhodu si zdokumentujeme na nasledujúcom príklade, keď budeme počítať riešenie dvoch nelineárnych rovníc:

Príklad 1

Riešte sústavu nelineárných rovníc:

$$2x_1-x_2-e^{-x1}=0$$
$$-x_1+2x_2-e^{-x2}=0$$

Ako na to? Ak chceme zabezpečiť, aby riešenie uvedenej sústavy bolo správne, musíme nájsť takú dvojicu neznámych x_1 a x_2, pre ktorú ľavá strana oboch rovníc bude rovná nule.
Algoritmus bude teda nasledovný:
Budeme postupne meniť hodnoty x_1 a x_2 a v každom kroku vyhodnotíme do akej miery nami zvolené hodnoty vyhovujú riešeniu uvedeného problému. Spôsobov ako vyhodnotiť kvalitu riešenia je viacero, no mi potrebujeme zabezpečiť, aby obe rovnice mali ľavú stranu čo najbližšiu hodnote nula. Možno Vás napadá, že najjednoduchšie by bolo spočítať obe ľavé strany dokopy a požadovať, aby spolu mali čo najmenšiu hodnotu. To však ale vôbec nemusí platiť (ľavá strana prvej rovnice má hodnotu napr. 100 a ľavá strana druhej rovnice má hodnotu -100, ich súčet je síce nula, ale daný bod nemožno považovať za riešenie). Aby sme sa vyhli uvedenému problému, vytvoríme si iné kritérium kvality riešenia, tzv. **najmenšie štvorce**:

Nech:

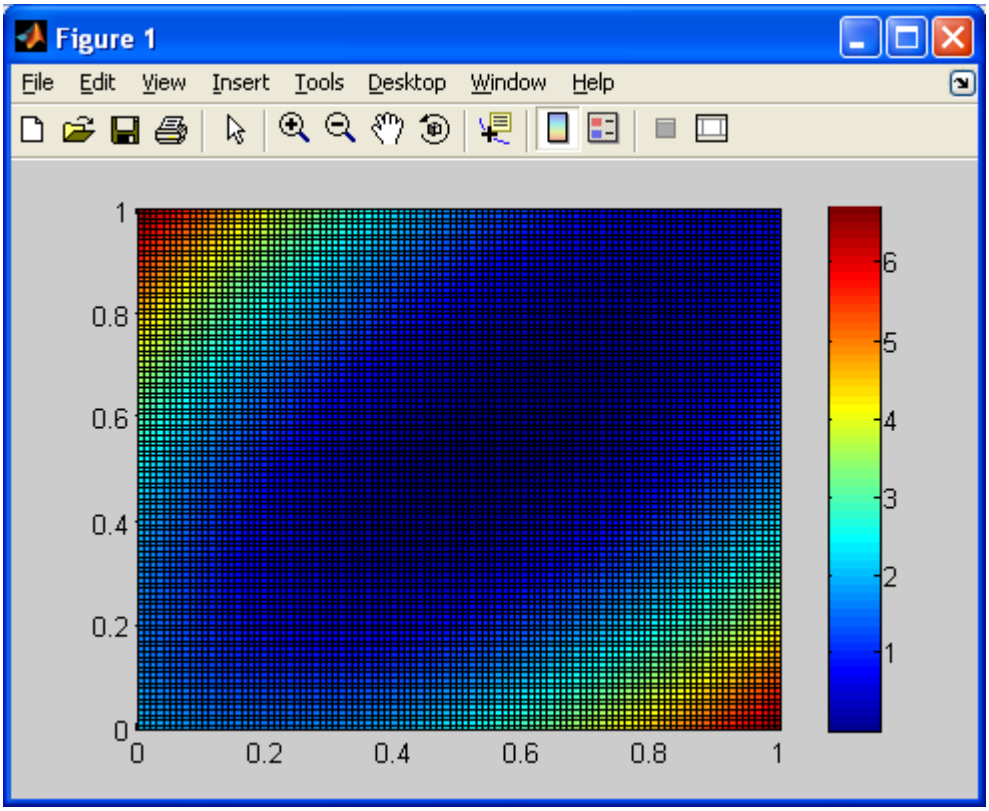
$$F_1=2x_1-x_2-e^{-x1}=0$$
$$F_2=-x_1+2x_2-e^{-x2}=0$$

Naše kritérium teda bude vyzerať nasledujúco:

$$suma\_stvorcov=(F_1)^2+(F_2)^2=0$$

Skúsme si nakresliť graf závislosti sumy štvorcov na hodnotách x_1 a x_2:

```
x1=linspace(0,1,100);
x2=linspace(0,1,100);
[x1,x2]=meshgrid(x1,x2);
for i=1:100
  for j=1:100
    F1=2*x1(i,j)-x2(i,j)-exp(-x1(i,j));
    F2=-x1(i,j)+2*x2(i,j)-exp(-x2(i,j));
    suma_stvorcov(i,j)=F1^2+F2^2;
  end
end
pcolor(x1,x2,suma_stvorcov);
colorbar
```



Z uvedeného obrázku je zjavné, že nulovej hodnote pripadá tmavomodrá farba. Ale kde presne je nulová hodnota? To je práve najväčšia nevýhoda uvedenej metódy. Ak v prípade jednej rovnice sme boli schopný koreň rovnice určiť relatívne presne, v tomto prípade by sme už potrebovali „orlí" zrak a ani to by nemuselo byť zárukou úspechu.

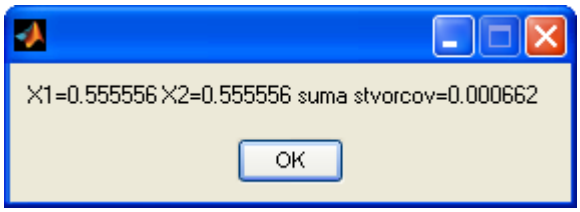
Beztak uvedená metóda je vhodná aspoň na približné určenie oblasti, v ktorej sa nachádza koreň nelineárnej rovnice.

Všeobecnejší spôsob priameho hľadania koreňov

Vyššie uvedený príklad predpokladal, že dokážeme veľmi presne odčítať z grafu hodnotu, v ktorej má závislosť sumy štvorcov nulovú hodnotu, čo bolo takmer nereálne. Pokúsme sa teraz na uvedený problém nedívať cez grafické zobrazenie, ale cez konkrétne hodnoty sumy štvorcov. Zvlášť nás budú zaujímať tie hodnoty neznámych x_1 a x_2, pre ktoré bude suma štvorcov najmenšia.

Vytvorme teda program, ktorý otestuje všetky kombinácie hodnôt x_1 a x_2 z intervalu od -5 po 5 a zapamätá si a vypíše hodnoty x_1 a x_2, v ktorých bola hodnota sumy štvorcov minimálna:

```
x1=linspace(-5,5,100);
x2=linspace(-5,5,100);
[x1,x2]=meshgrid(x1,x2);
minimalna_hodnota=1e6;
for i=1:100
  for j=1:100
    F1=2*x1(i,j)-x2(i,j)-exp(-x1(i,j));
    F2=-x1(i,i)+2*x2(i,j)-exp(-x2(i,j));
    suma_stvorcov=F1^2+F2^2;
    if minimalna_hodnota>suma_stvorcov
       minimalna_hodnota=suma_stvorcov;
       koren1=x1(i,j);
       koren2=x2(i,j);
    end
  end
end
str=sprintf('X1=%f X2=%f suma stvorcov=%f',koren1,koren2,minimalna_hodnota);
msgbox(str);
```



Dostali sme teda hodnoty koreňov x_1 a x_2, pre ktoré je hodnota sumy štvorcov 0,00062, čo v podstate nie je zlý výsledok. Čo ale ak chceme dostať presnejší výsledok?
Existujú v podstate dva spôsoby ako uvedený výsledok spresniť:

- zväčšiť počet krokov
- postupne zmenšovať interval, na ktorom hľadáme riešenie

Využitie Symbolic toolboxu na riešenie nelineárnych rovníc

MATLAB poskytuje hneď niekoľko principiálne odlišných spôsobov riešenia nelineárnych rovníc. Teraz si ukážeme ako je možné riešiť jednoduché nelineárne rovnice pomocou **Symbolic toolboxu.**

Riešenie využitím symbolic toolboxu je veľmi jednoduché, preto sa nebude veľmi venovať teórii, ale priamo sa vrhneme na príklad:

Príklad 1

Riešte nelineárnu rovnicu:

$$x^2\text{-}x\text{-}2=0$$

```
syms x
f=x^2-x-2;
korene=solve(f)
```

Podobným spôsobom je možné vyriešiť aj vyššie uvedenú sústavú dvoch nelineárnych rovníc:

Príklad 2

Riešte sústavu nelineárnych rovníc:

$$2x_1-x_2-e^{-x1}=0$$
$$-x_1+2x_2-e^{-x2}=0$$

```
syms x1 x2
F1=2*x1-x2-exp(-x1);
F2=-x1+2*x2-exp(-x2);
[k1,k2]=solve(F1,F2)
```