evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Číslo projektu	CZ.1 .07/1.1 .36/02.0066
Autor	Ladislav Kašpárek
Předmět	Programovací metody
Téma	Numerické řešení rovnic
Metodický pokyn	výkladový text s ukázkami

Funkce FindRoot

Úvod

V tomto dokumentu ukážeme několik vlastností a parametrů funkce FindRoot, která je v systému *Mathematica* zabudována. Jedním ze závěrů bude nutnost znát vnitřní algoritmus numerického výpočtu, abychom mohli funkci efektivně používat, popř ji naprogramovat v Javě. Výklad metod půlení intervalu a Newtonovy metody tečen bude v dalších souborech.

V tomto souboru bude připravena řada pokusů s funkcí FindRoot, tak aby žák mohl experimentovat a zkoušet co vydrží, kde se osvědčí a kde selže.

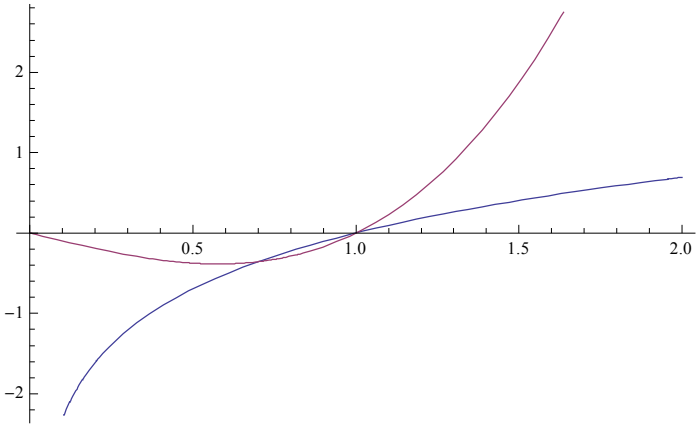
Matematická poznámka

Rovnice jsou často zadávány jako rovnost dvou výzazů a řešením je bod kde se protnou jejich funkce. Např

$$\ln x = x^3 - x$$

```
In[1]:= Plot[{Log[x], x^3 - x}, {x, 0, 2}]
```

Out[1]=



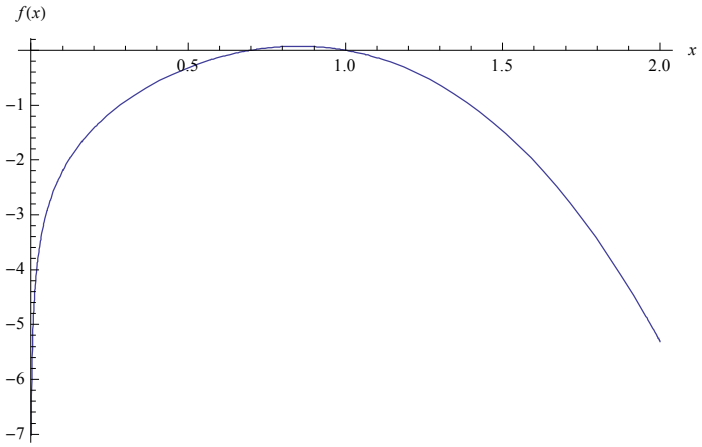
Dle potřeby lze rovnici upravit a převést neznámou na jednu stranu a na druhé ponechat nulu. Graf pak vypadá jinak, je tam jen jedna funkce a výsledek--kořen se pak hledá jako průsečík s osou x. Oba způsoby zápisu jsou ekvivalentní.

$$\ln x = x^3 - x \mid -x^3 + x$$

$$-x^3 + x + \ln x = 0$$

In[2]:=
```
Plot[{-x^3 + x + Log[x]}, {x, 0, 2}, AxesLabel → {x, f[x]}]
```

Out[2]=



Algebraické řešení rovnice pravděpodobně selže, proto bude nutné použít funkci FindRoot.

In[3]:=
```
Solve[Log[x] == x^3 - x, x]
```

Solve::nsmet : This system cannot be solved with the methods available to Solve. ≫

Out[3]=
```
Solve[Log[x] == -x + x^3, x]
```

Z grafů lze vyčíst, že rovnice má dva kořeny: $x = 1$, ten je zřejmý, ověřte zkouškou a druhý "okolo" čísla 0,7.

In[4]:=
```
FindRoot[Log[x] == x^3 - x, {x, 0.7}]
```

Out[4]=
```
{x → 0.699661}
```

Funkce FindRoot

Několik poznámek k funkci FindRoot:
1/ prostudujte si její dokumentaci (některým věcem bez dalších znalostí nebudeme rozumnět)
2/ implicitně používá Newtonovu metodu tečen, která má svá omezení
3/ zde popíšeme vybrané parametry funkce

Počet desetinných míst výsledku a přesnost výpočtu

Přesnost výpočtu lze ovlivnit celkem třemi paramtry (AccuracyGoal, PrecisionGoal, WorkingPrecision). Zde nebudeme rozebírat jemné rozdíly mezi těmito parametry, nám postačí WorkingPrecision.

In[5]:=
```
FindRoot[Log[x] == x^3 - x, {x, 0.7}, WorkingPrecision → 12]
```

Out[5]=
```
{x → 0.699661268820}
```

In[6]:=
```
FindRoot[Log[x] == x^3 - x, {x, 0.7}, WorkingPrecision → 50]
```

Out[6]=
```
{x → 0.69966126882071989973028470235168919083941680338187}
```

In[7]:=
```
FindRoot[Log[x] == x^3 - x, {x, 0.7}, WorkingPrecision → 222]
```

Out[7]=
```
{x →
  0.69966126882071989973028470235168919083941680338187211084015326899211666109815
   45675589392877901606867095083120450660865895336993207868267395959276434241895
   53229190090684112841293591351394431639492028637167642097069453453195693}
```

WorkingPrecision určuje na kolik desetinných míst budou nastavena čísla v průběhu výpočtu.

Počet iterací výpočtu

Iterace je jakýsi postupný krok algoritmu, výpočtu. V našem případě by výpočet měl po konečném počtu kroků skončit. Může se však stát, že se algoritmus zacyklí a pak by nikdy neskončil. Podobně, když je rovnice nešťastně zadaná, algoritmus spatně (pomalu) konverguje k výsledku a výpočet by trval dlouho.
Algoritmus má nastaven maximální počet iterací. Pokud najde kořen do tohoto počtu kroků je vše vpořádku, pokud dosáhne maxima, je ukončen a uživatel musí rozhodnout co dál.

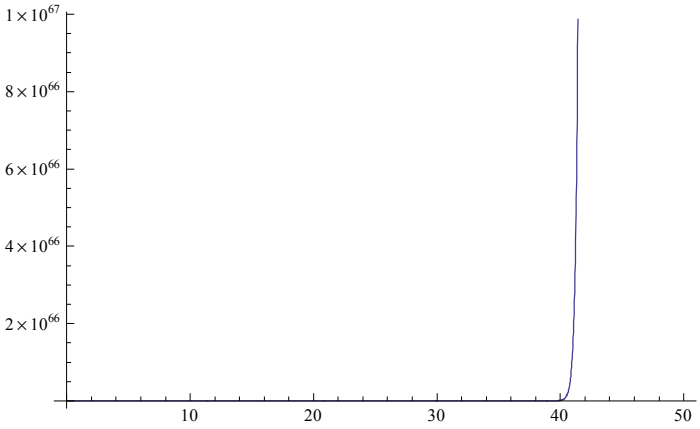
In[8]:=
```
FindRoot[x^x == 2 x, {x, 50}]
```

FindRoot::cvmit : Failed to converge to the requested accuracy or precision within 100 iterations. ≫

Out[8]=
```
{x → 28.5435}
```

In[9]:=
```
Plot[x^x - 2 x == 0, {x, 0, 50}]
```

Out[9]=



POZOR výsledek okolo 28 je zcela chybný! Proveďte zkoušku!!!
Ale *Mathematica* nás na to upozornila, že dosáhla maxima iterací, implicitně nastaveno na 100. Pokusme se toto číslo zvětšit.

In[10]:=
```
FindRoot[x^x == 2 x, {x, 50}, MaxIterations → 200]
```

Out[10]=
```
{x → 2.}
```

Úkol

Tato rovnice má dva kořeny, najděte druhý na 100 desetinných míst. (použijte graf)

Počáteční podmínka výpočtu

Newtonova metoda pro svůj výpočet potřebuje počáteční podmínku: nějaký výchozí bod výpočtu. Určit nějaké obecné pravidlo kde a jak hodnotu volit nelze, ale rovnice zpravidla popisují nějaký

reálný jev, děj, úlohu a lze tedy odhadnou přibližně výsledek, a to obvykle stačí. Newtonova metoda pak výpočet kořene oproti odhadu jen zpřesní. Na volbě počáteční hodnoty hodně záleží. Pro nevhodné počáteční hodnoty můžeme dotat zcela nevyhovující kořeny:

In[11]:=
```
FindRoot[Log[x] == -x, {x, .5}]
```

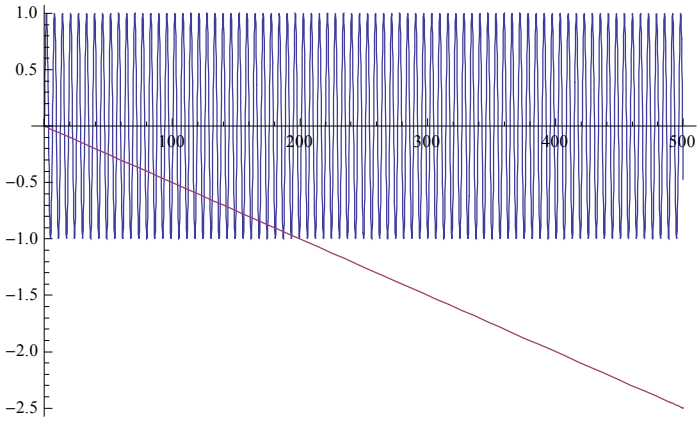
Out[11]=
```
{x → 0.567143}
```

Každá metoda tohoto typu potřebuje “starovací” hodnotu, od které spustí výpočet, ate to je slabina, je-li kořenů víc.

Rozdílné výsledky pro různá nastavení počáteční hodnoty výpočtu.

In[12]:=
```
Plot[{Sin[x], -0.005 x}, {x, 0, 500}]
```

Out[12]=



In[13]:=
```
Table[FindRoot[Sin[x] == -0.005 x, {x, a}],
 {a, {1, 2, 3, 5, 7, 10, 20, 50, 100, 200, 300, 500, 800}}]
```

Out[13]=
```
{{x → -2.52435 × 10^-29}, {x → 3.15738}, {x → 3.15738}, {x → 6.25192},
 {x → 6.25192}, {x → 9.47216}, {x → 18.7556}, {x → 50.0127}, {x → 100.007},
 {x → 199.558}, {x → 149.949}, {x → 2.52435 × 10^-29}, {x → -87.5117}}
```

Rozdílné výsledky pro různá nastavení počáteční hodnoty výpočtu.

Metoda sečen

Metoda tečen pro svůj regulerní běh potřebuje splnit poměrně přísná kriteria (zde ja zatím neb- udeme vyjmenovávat, až u výkladu metody). Jsou-li splněna, pak je metoda rychlá. Pokud ne nemusíme výsledek dostat vůbec nebo za dlouho.
Metoda sečen tolik omezení nemá, ale je trochu pomalejší. Pro metodu sečen musíme zadat celý interval, kde se nachází právě jeden kořen.

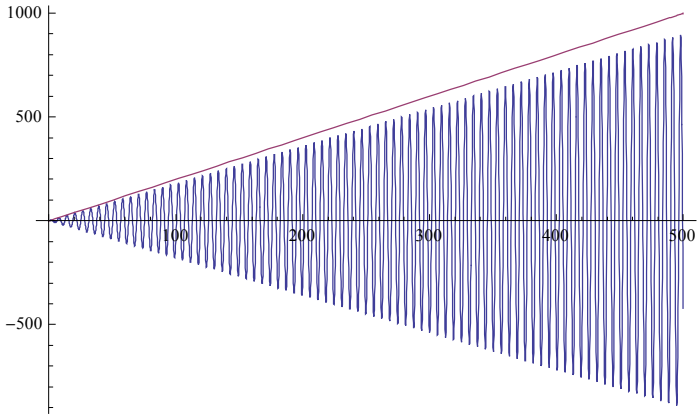
In[14]:=
```
FindRoot[x^x == 2 x, {x, 0.1, .5}]
```

Out[14]=
```
{x → 0.346323}
```

Metoda tečen selhává!

In[15]:=
```
Plot[{1.8 x * Sin[x], 2 x - 1}, {x, 0, 500}]
```

Out[15]=



In[16]:=
```
FindRoot[1.8 x * Sin[x] == 2 x - 1, {x, 500}]
```

FindRoot::lstol :
The line search decreased the step size to within tolerance specified by AccuracyGoal and PrecisionGoal but was unable to find a sufficient decrease in the merit function. You may need more than MachinePrecision digits of working precision to meet these tolerances. ≫

Out[16]=
```
{x → 0.640471}
```

Metoda sečen bez problému vyčísluje výsledek

In[17]:=
```
FindRoot[1.8 x * Sin[x] == 2 x - 1, {x, 0, 500}]
```

Out[17]=
```
{x → 2.126}
```

Shrnutí

Rovnice lze řešit numericky, různým metodami, které však mají své výhody a nevýhody. Proto je nutné metody znát, aby je uživatel mohl efektivně používat.

Úlohy

1/ Vyřeš rovnice, počáteční podmínky si stanov sám z pozorování grafů:

$$\sin x^x = 2x$$

$$\sin(\sin x) = 2x - 1$$

Zdroje

http://reference.wolfram.com/mathematica/ref/FindRoot.html

http://mathworld.wolfram.com/NewtonsMethod.html