# Puleni_intervalu

# Metoda půlení intervalu (hledání kořenů)

*Klára Drhová, 2014*

## Teoretické shrnutí

### Věta

Nechť funkce $f$ je spojitá na uzavřeném intervalu $\langle a, b\rangle$ a nechť $f(a)\cdot f(b) < 0$ (jedna z funkčních hodnot je záporná, druhá kladná). Potom existuje bod $c \in (a, b)$ takový, že $f(c) = 0$.

Tato věta udává postačující podmínky pro existenci řešení rovnice $f(x) = 0$. Navíc její důkaz udává i postup (algoritmus), jak toto řešení hledat přibližně (numericky se zadanou předepsanou přesností).

### Postup

Mějme funkci $f$ s definičním oborem $D_f$ a body $a, b \in D_f$, pro které platí, že $f(a)$ a $f(b)$ mají různá znaménka. Potom pro bod $c = \frac{a+b}{2}$ musí platit jedna z následujících možností:

- $f(c) = 0$
- znaménka $f(a)$ a $f(c)$ jsou různá
- znaménka $f(c)$ a $f(b)$ jsou různá.

Pokud nastala první možnost, tak jsme nalezli hledané řešení. Pokud nastala druhá nebo třetí možnost, tak jsme získali nový interval poloviční délky, pro který předpoklady věty také platí. Postup tedy můžeme rekurzivně opakovat, dokud nenastane první možnost (tzn. dokud nenajdeme hledané řešení), nebo dokud nenalezneme interval takový, abychom získali požadovanou přesnost přibližného řešení (tzn. $b - a < \epsilon$, kde $\epsilon$ je požadovaná přesnost).

Funkce v Pythonu, která postupně vrací intervaly, až nalezne řešení nebo k přibližné řešení s požadovanou přesností:

```
def Puleni_intervalu(funkce, od, do, presnost):
    prumer = (od + do)/2
```

```
    if funkce(prumer)==0: #nalezli jsme řešení
        return [(float(od), float(do)), float(prumer)]
    if do - od <= presnost: #máme požadovanou přesnost
        return [(float(od), float(do))]
    if funkce(prumer)*funkce(od) < 0:
        return [(float(od), float(do))] +
Puleni_intervalu(funkce, od, prumer, presnost)
    return [(float(od), float(do))] +
Puleni_intervalu(funkce, prumer, do, presnost)
```

# Ukázka

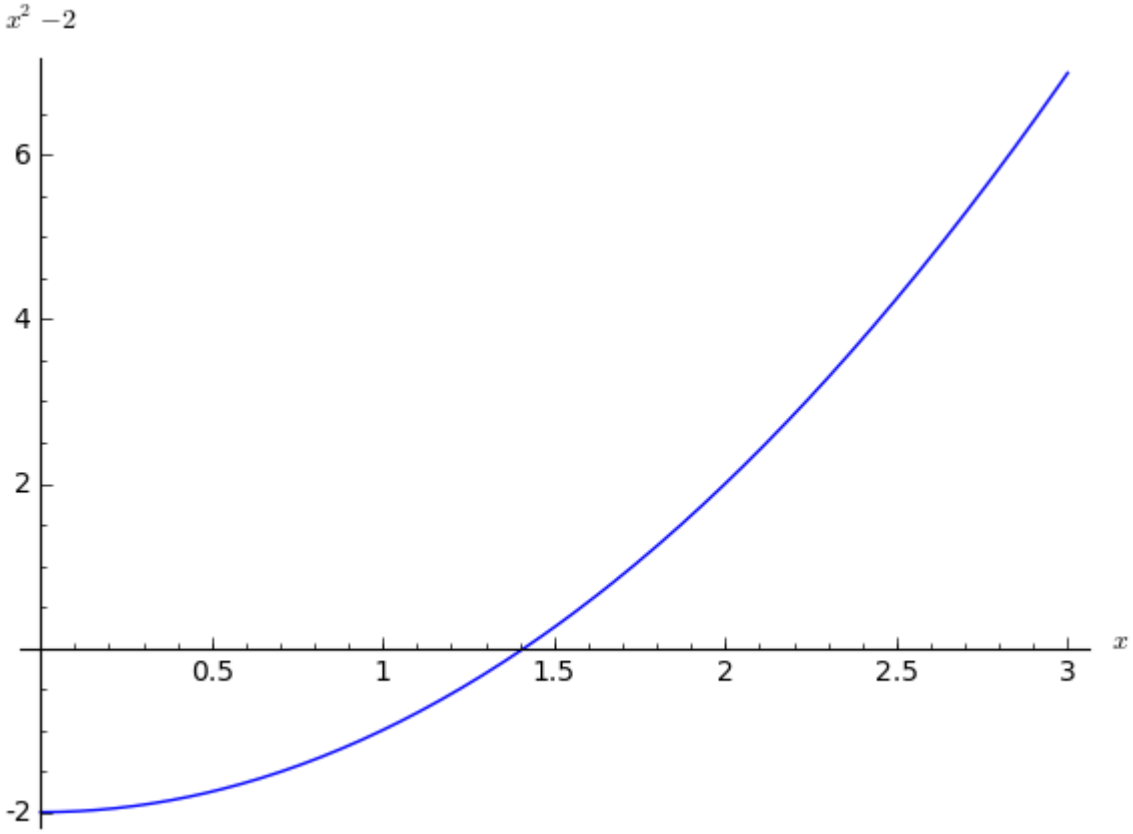
Mějme rovnici $x^2 = 2$. Pokud převedeme dvojku na druhou stranu, získáme rovnici $x^2 - 2 = 0$. Její řešení je průsečík grafu funkce $f(x) = x^2 - 2$ s osou $x$ a hledáme tedy numerickou hodnotu $\sqrt{2}$.

**Příklad**

Najděte numerickou hodnotu $\sqrt{2}$ s přesností na 0,01.

Graf:

```
plot(x^2-2, (x, 0, 3), axes_labels = ['$x$',
'$x^2-2$']).show(figsize=6)
```




Definujme si funkci $f(x)$ a použijme funkci Puleni_intervalu definovanou výše.

Jako počáteční body můžeme zvolit například 1 a 2, protože:

$1 < 2 < 4$ a tudíž $\sqrt{1} = 1 < \sqrt{2} < \sqrt{4} = 2$ (odmocnina ze dvou tedy jistě leží v intervalu $(1, 2)$)

$f(1) = -1,\ f(2) = 2,\ f(1) \cdot f(2) < 0$ (předpoklady Věty platí)

```
def f(x):
    return x*x - 2
reseni = Puleni_intervalu(f, 1, 2, 0.01)
reseni #výpis řešení
```

```
[(1.0, 2.0),
 (1.0, 1.5),
 (1.25, 1.5),
 (1.375, 1.5),
 (1.375, 1.4375),
 (1.40625, 1.4375),
 (1.40625, 1.421875),
 (1.4140625, 1.421875)]
```

```
sqrt(2).n() #odmocnina ze dvou
```

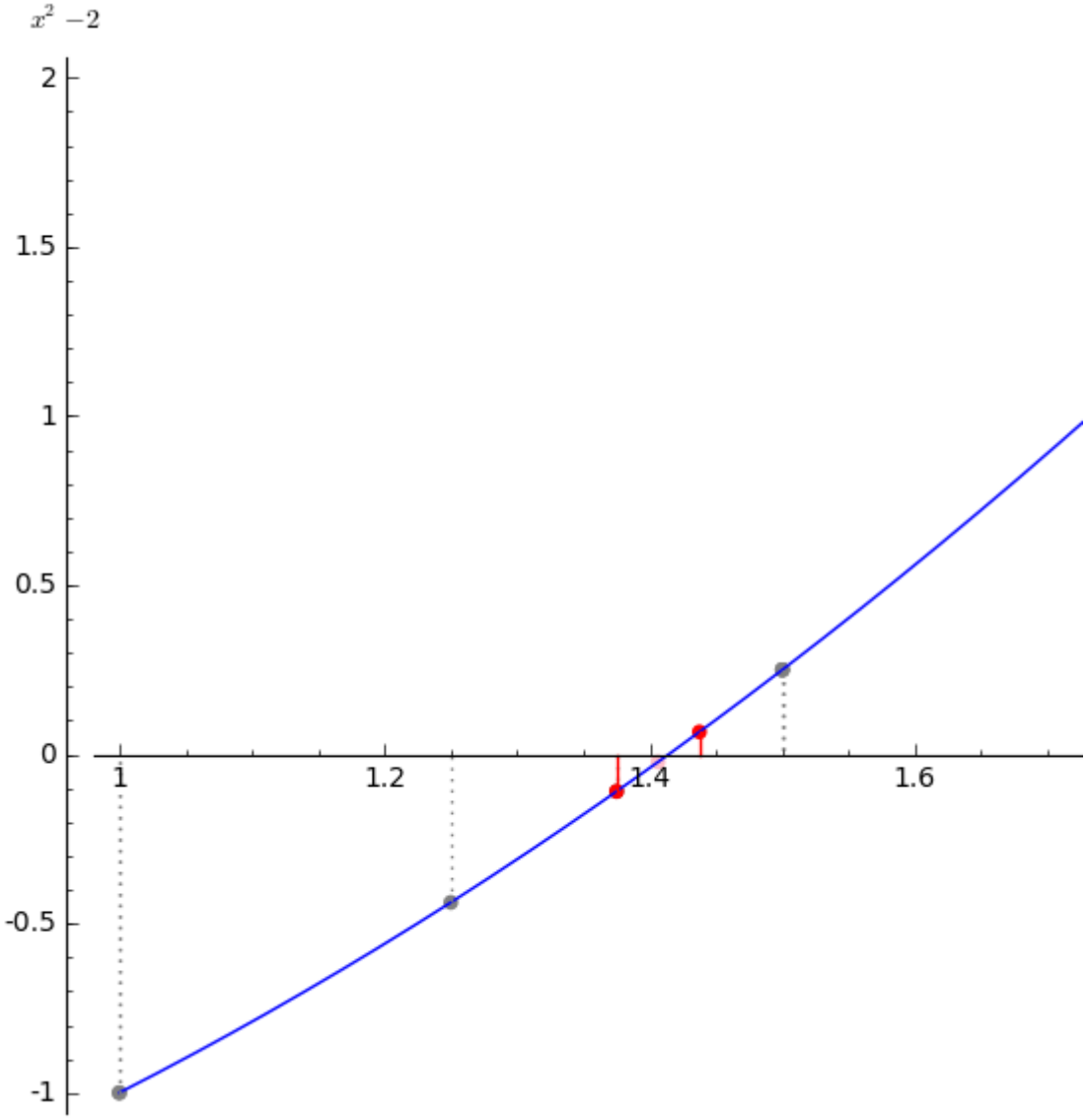
```
1.41421356237310
```

Vidíme, že $\sqrt{2}$ leží někde mezi 1,414 a 1,422. Podle Sage přibližně platí $\sqrt{2} = 1,41421356237310$ a tudíž nám naše řešení opravdu sedí. Pokud bychom potřebovali přesnější výsledek, stačilo by aplikovat funkci vícekrát.

Na následující demonstraci můžete vidět na grafu, jak se postupovalo.

```
@interact
def interv(i = slider(range(8), default = 0)):
    graf = plot(x^2-2, (x, 1, 2), axes_labels = ['$x$',
'$x^2-2$'])
    for j in [0..i-1]: #minulé intervaly
        graf+= line([(reseni[j][0],f(reseni[j][0])),(reseni[j]
[0],0)], linestyle = 'dotted', color = 'grey')
        graf+= point((reseni[j][0],f(reseni[j][0])), size =
30, color = 'grey')
        graf+= line([(reseni[j][1],f(reseni[j][1])),(reseni[j]
[1],0)], linestyle = 'dotted', color = 'grey')
        graf+= point((reseni[j][1],f(reseni[j][1])), size =
30, color = 'grey')
    graf+= line([(reseni[i][0],f(reseni[i][0])),(reseni[i]
[0],0)], color = 'red') #aktuální interval
    graf+= point((reseni[i][0],f(reseni[i][0])), size = 30,
color = 'red')
    graf+= line([(reseni[i][1],f(reseni[i][1])),(reseni[i]
[1],0)], color = 'red')
    graf+= point((reseni[i][1],f(reseni[i][1])), size = 30,
color = 'red')
    graf+= point(((reseni[i][1]+reseni[i][0])/2,f((reseni[i]
[1]+reseni[i][0])/2)), size = 30, color = 'pink') #průměr
    graf.show()
```

i 0




**Příklad 2**

Najděte numerickou hodnotu zlatého řezu s přesností na 0,0001.

Zlatý řez je kořen rovnice $\varphi^2 - \varphi - 1 = 0$.

Počáteční hodnoty můžeme například zvolit 0 a 2.

```
def Zlaty_rez_rovnice(x):
    return x*x - x - 1
Puleni_intervalu(Zlaty_rez_rovnice, 0, 2, 0.0001)
```

```
    [(0.0, 2.0),
```

```
(1.0, 2.0),
(1.5, 2.0),
(1.5, 1.75),
(1.5, 1.625),
(1.5625, 1.625),
(1.59375, 1.625),
(1.609375, 1.625),
(1.6171875, 1.625),
(1.6171875, 1.62109375),
(1.6171875, 1.619140625),
(1.6171875, 1.6181640625),
(1.61767578125, 1.6181640625),
(1.617919921875, 1.6181640625),
(1.617919921875, 1.6180419921875),
(1.61798095703125, 1.6180419921875)]
```

Zlatý řez má tedy hodnotu asi 1.618, což nám potvrdí i Sage.

```
show(golden_ratio.n())
plot(x^2-x-1, (x, 0, 2)).show(figsize=4)
```

$$1.61803398874989$$