Algoritmy pro numerické výpočty

Jitka Kreslíková, Aleš Smrčka

2023

Fakulta informačních technologií
Vysoké učení technické v Brně

IZP – Základy programování

Algoritmy pro numerické výpočty

- Výpočet hodnoty polynomu
- Řešení nelineárních rovnic
- Numerický výpočet určitého integrálu

Výpočet hodnoty polynomu

- Definice polynomu:

Nechť n je přirozené číslo a nechť $a_0, a_1, \ldots, a_n$, jsou reálná, resp. komplexní čísla. Funkce P(x), kterou lze definovat pro všechna reálná, resp. komplexní čísla s předpisem:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i, \quad (1)$$

se nazývá **mnohočlen** (jedné proměnné x s reálnými, resp. komplexními koeficienty). Místo mnohočlen se také říká **polynom** nebo **celá racionální funkce**. Čísla $a_0, a_1, \ldots, a_n$, se nazývají **koeficienty polynomu** P(x).

Výpočet hodnoty polynomu

- **Stupněm polynomu** P(x) nazýváme nejvyšší mocninu proměnné x ve výrazu (1), u níž je nenulový koeficient. Je-li v (1) $a_n \neq 0$, pak P(x) je n-tého stupně

Příklad: $P(x) = 3x^4 + 2x^3 - x^2 + x + 4 \Rightarrow n = 4$

- **Výpočet hodnoty polynomu v bodě x**

Prostá implementace funkce vyžaduje přímý výpočet pomocí funkce, která počítá x^n. Tento přístup potřebuje kvadratický čas $O(n^2)$.

Počet operací násobení: $n + n\text{-}1 + \ldots + 1 \Rightarrow O(n^2)$

Hornerovo schéma

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 =$$

n operací násobení

$$(((\ \ldots(a_n x + a_{n-1})x + \ldots + a_1)x + a_0$$

n operací sečítání

Příklad: $P(x) = 3x^4 + 2x^3 - x^2 + x + 4 = (((3x + 2)x - 1)x + 1)x + 4$
$P(2) = 66$

koeficienty	**3**	**2**	**-1**	**1**	**4**
		6	**16**	**30**	**62**
		x 2	x 2	x 2	x 2
v bodě 2	**3**	**8**	**15**	**31**	**66**

Hornerovo schéma

Příklad: výpočet polynomu v zadaném bodě.

Následující funkce předpokládá, že polynom je v paměti počítače reprezentován strukturou, která obsahuje stupeň polynomu a pole koeficientů polynomu. Funkce vrací hodnotu polynomu v bodě daném parametrem x.

```
#include <stdio.h>
#define N 4
typedef struct poly
{
  int degree;          // stupeň polynomu
  double coef[N + 1]; // koeficienty polynomu
}Tpoly;
```

Hornerovo schéma

Příklad: výpočet polynomu v zadaném bodě - pokračování.

```
double evalHorner (Tpoly *polynom, double x)
{
  double sum = 0.0;
  for (int i = 0; i <= polynom->degree; i++)
    sum = sum * x + polynom->coef[i];
  return sum;
}
```

Hornerovo schéma

Příklad: výpočet polynomu v zadaném bodě - pokračování.

```
int main (void)
{
  Tpoly myPoly = {4,{3.0, 2.0, -1.0, 1.0, 4.0} };
  double x = 2.0;  // inicializace
  printf ("Hodnota polynomu v bode %f je %f\n",x,
           evalHorner (&myPoly, x));
  return 0;
}
```

Hornerovo schéma

- vyčíslení hodnoty čísla zapsaného v obecné číselné soustavě:
  - číslice vstupního čísla figurují jako koeficienty polynomu
  - základ číselné soustavy jako bod, ve kterém se má polynom spočítat.

Příklad: vyčíslení hodnoty čísla

$$(2352)_6 = 2\times 6^3 + 3\times 6^2 + 5\times 6^1 + 2\times 6^0 = ((2\times 6+3)\times 6+5)\times 6+2 = (572)_{10}$$

Řešení nelineárních rovnic

- Formulace problému:

Hledání reálných kořenů rovnice P(x) = 0, kde P(x) je polynom:

$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

Hledáme reálné číslo α pro které platí $f(\alpha) = 0$; α je kořen rovnice $f(x) = 0$.

Při určování kořene rovnice je u některých metod požadováno, aby byl separován kořen rovnice

Řešení nelineárních rovnic

- Separaci lze provést několika způsoby:
  - Můžeme např. vyšetřit průběh funkce a z funkce *f(x)* spočítat první a druhou derivaci.
  - Z průběhů derivací lze zjistit, ve kterých intervalech je funkce rostoucí a klesající a vyšetřit pak lokální minima a maxima.
  - Je důležité si uvědomit, že reálné kořeny rovnice jsou průsečíky grafu funkce a osy x.
  - Zjištěné intervaly potom použijeme pro přibližný výpočet kořenů.

Numerické metody:
http://www.slu.cz/math/cz/knihovna/ucebni-texty/Numericke-metody/Numericke-metody.pdf
[on line, cit. 2019-11-17]

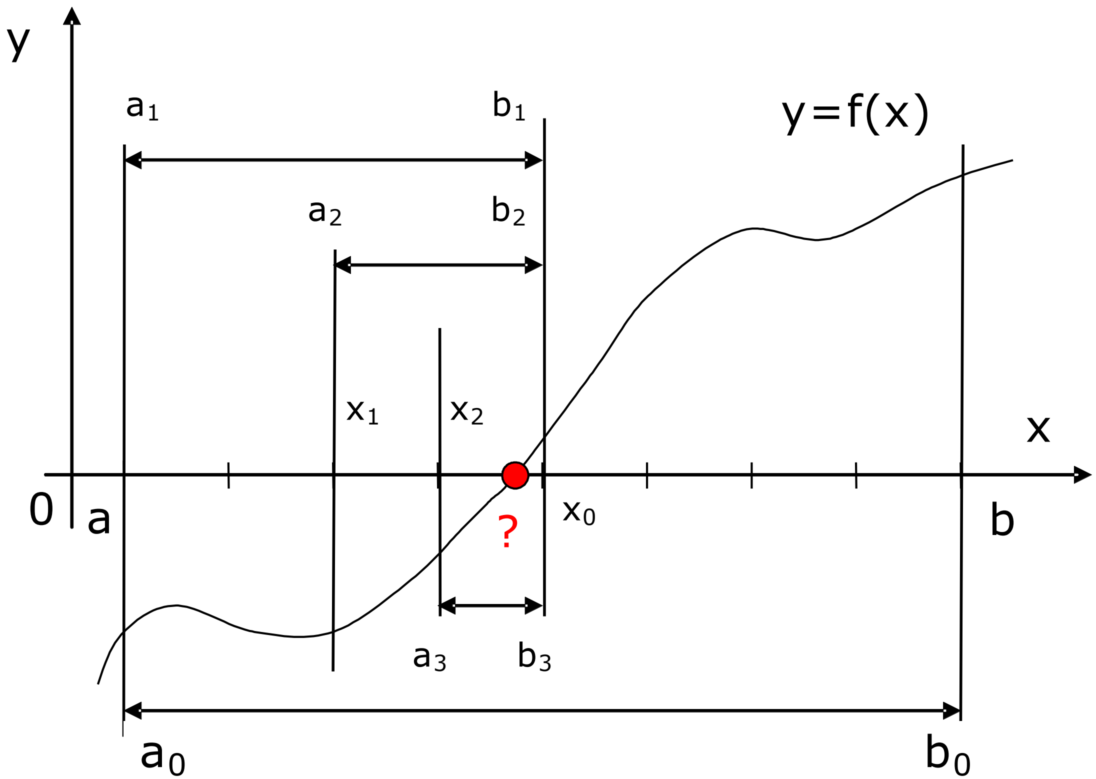
Metoda půlení intervalu (bisekce)

- konvergentní, univerzální metoda,
- musí být splněny dvě podmínky:
  1. funkce *f* musí být spojitá pro $\forall x \in I_0 = \langle a_0, b_0 \rangle$
  2. funkční hodnoty v krajních bodech zvoleného intervalu musí mít opačná znaménka tj. musí platit: $f(a_0) \times f(b_0) < 0$.

- pokud jsou obě podmínky splněny, pak tato metoda vždy konverguje

Metoda půlení intervalu (bisekce)

Princip metody půlení intervalu.



$x_i = s_i$ je střed příslušného intervalu

Metoda půlení intervalu (bisekce)

Příklad: funkce pro hledání kořene metodou půlení intervalu

```
double root_equation (double a, double b,
    double eps,double(*evalFun)(double))
{
  double middle = (a + b) / 2;
  double fmid = evalFun(middle);
```

Metoda půlení intervalu (bisekce)

Příklad: funkce pro hledání kořene metodou půlení intervalu - pokračování

```
  while (fabs(fmid) > eps) {
    if (evalFun(a) * fmid < 0)
      b = middle;
    else
      a = middle;
    if (fabs (fmid) > eps) {
      middle = (a + b) / 2;
      fmid = evalFun(middle);
    }
  }
  return middle;
}
```

Numerický výpočet určitého integrálu

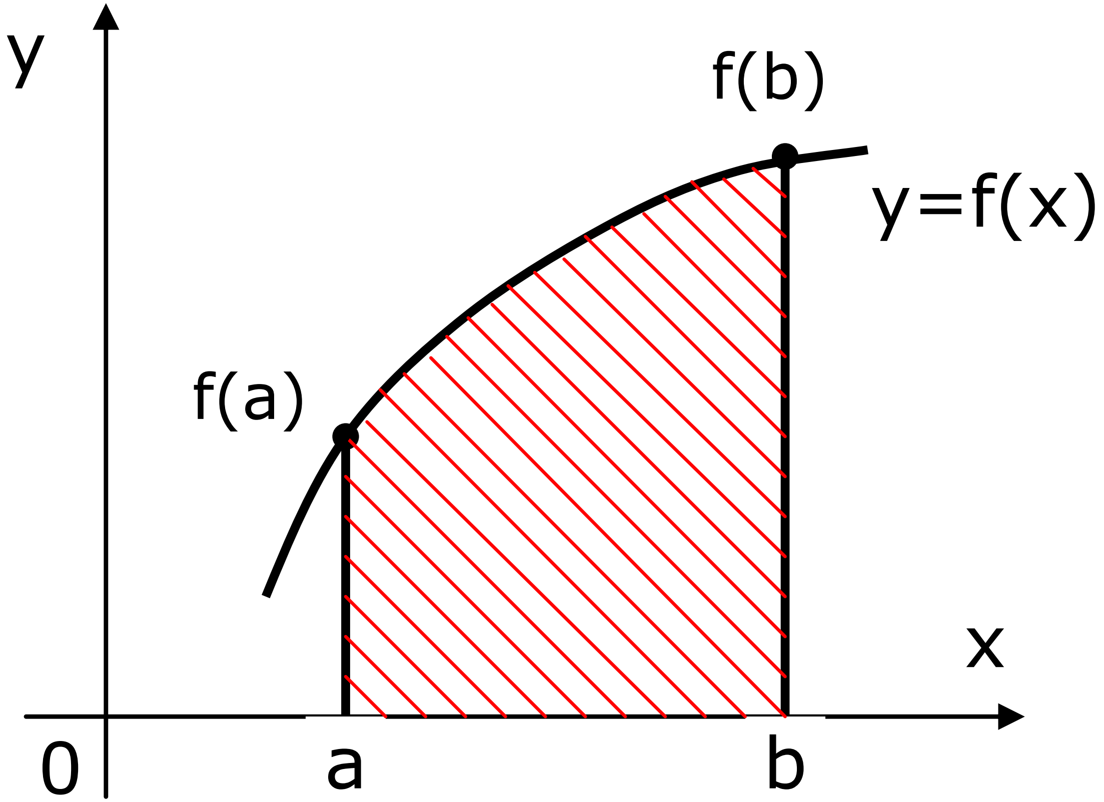
- Definice:

Jestliže je funkce f(x) spojitá v uzavřeném intervalu <a,b> a známe-li její primitivní funkci F(x), můžeme vypočítat určitý integrál funkce f(x) v mezích od a do b pomocí vztahu:

$$\int_a^b f(x)dx = F(b) - F(a)$$

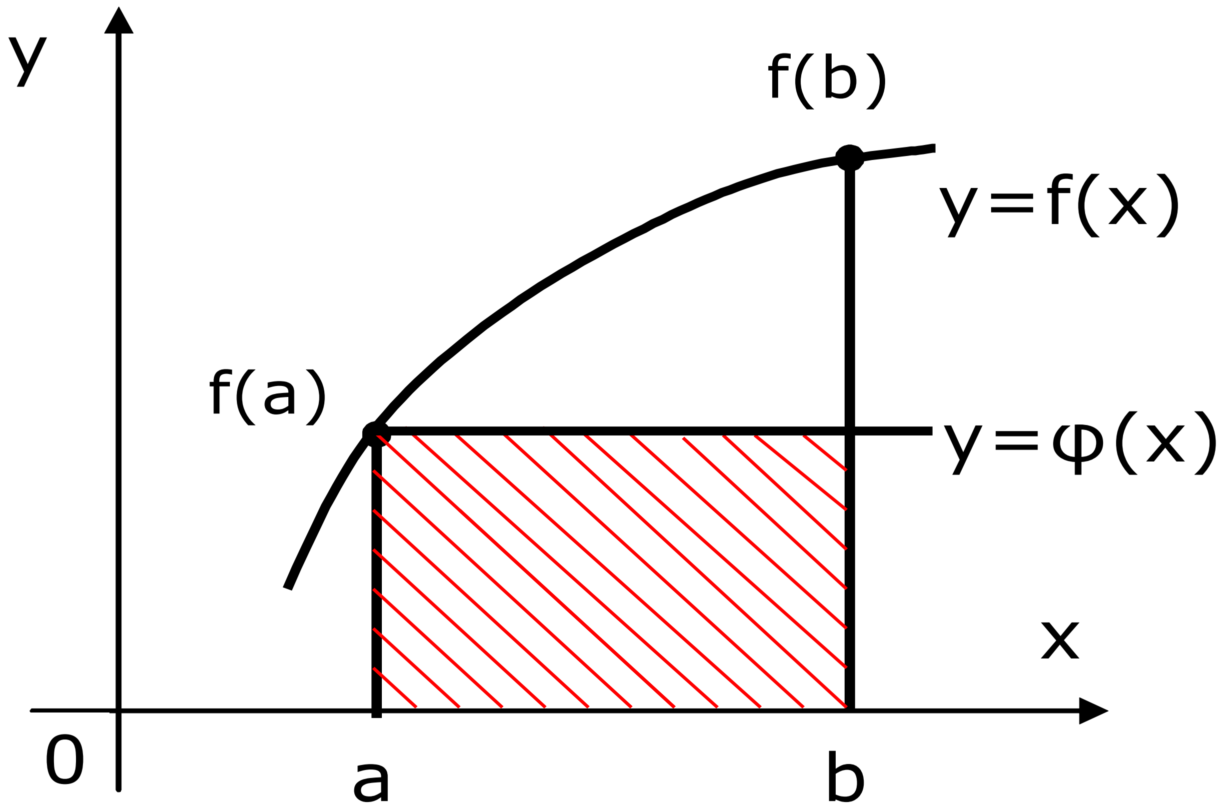
Metody numerického integrování užíváme v takových případech, kdy je obtížné najít funkci F(x), nebo v případě, že funkce f(x) je dána tabulkou.

Numerický výpočet určitého integrálu

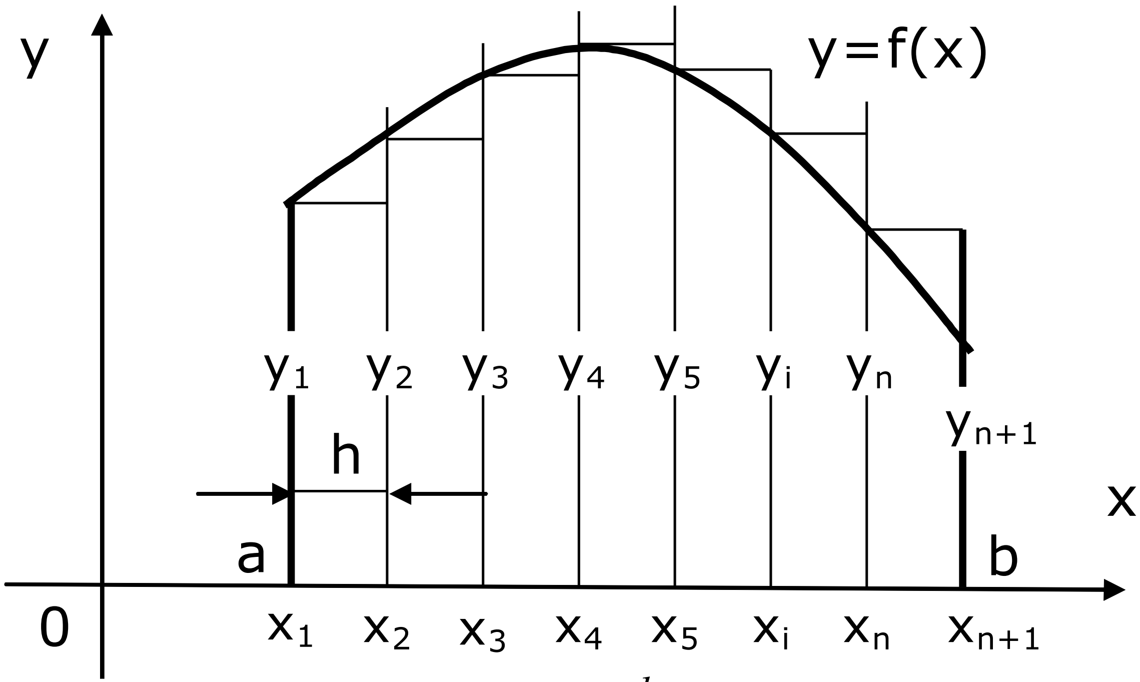


Obdélníková metoda (pravidlo, vzorec)

Riemannův součet



Obdélníková metoda



Vzorec pro výpočet: $$\int_a^b f(x)dx \approx \sum_{i=1}^{n} h_i y_i = h\sum_{i=1}^{n} y_i \quad ,$$

kde šířka dílčího intervalu $h = (b-a)/n$

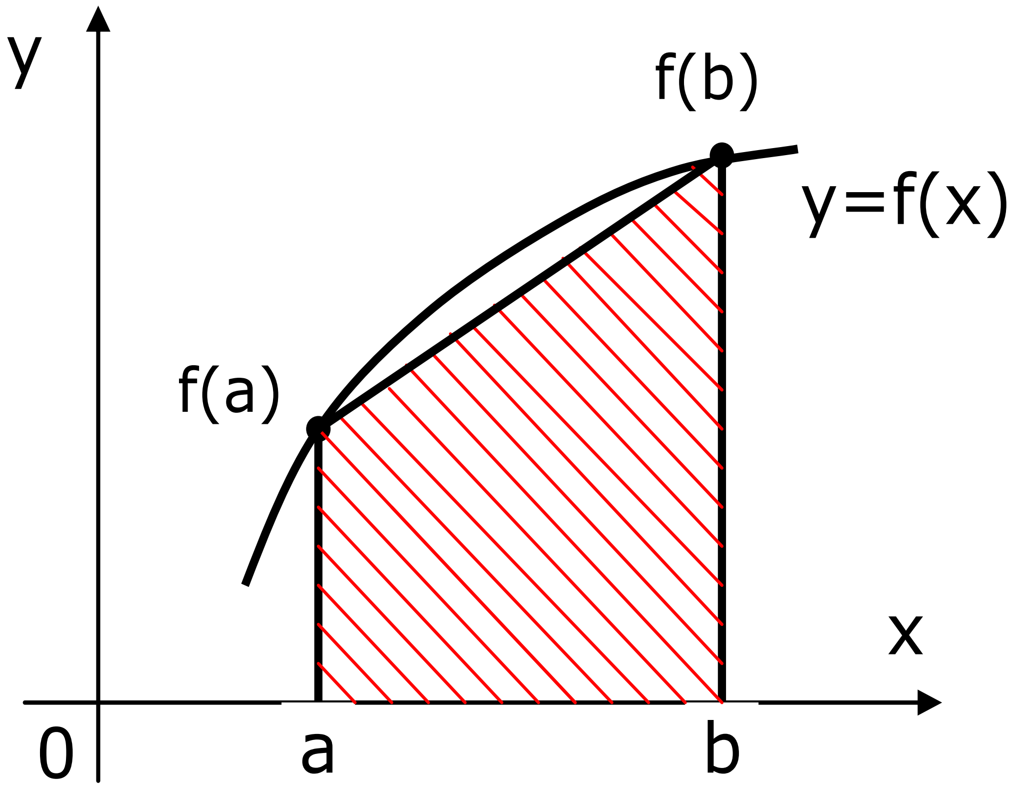
Obdélníková metoda

Příklad: funkce pro výpočet integrálu obdélníkovou metodou

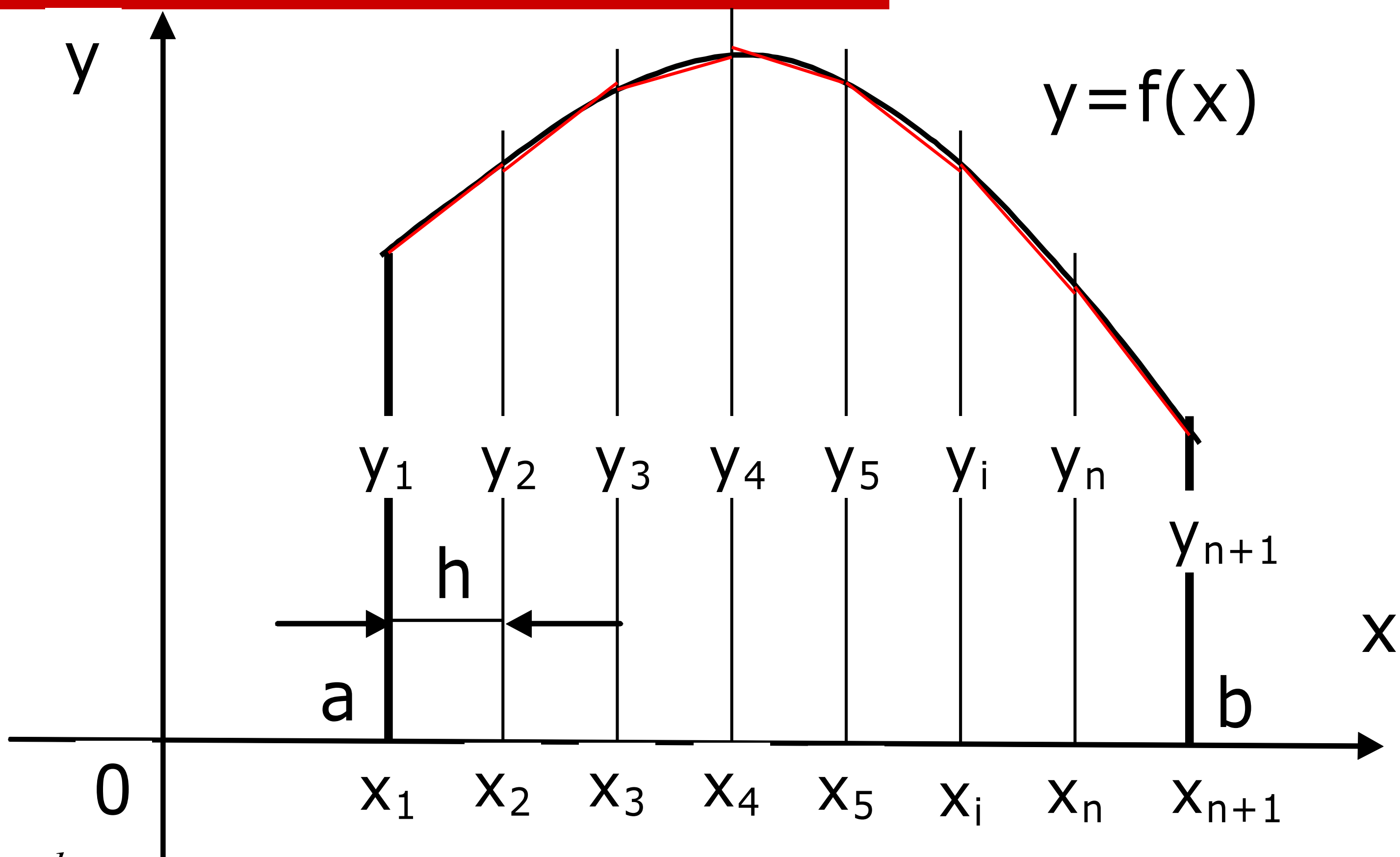
```
double integrate_rectangle (double a, double b,
                int n, double (*evalFun)(double))
{
  double step, sum = 0.0;
  step = (b - a) / n;
  for (double x = a; x < b - (step/2); x += step)
    sum += evalFun(x);
  sum *= step;
  return sum;
}
```

Lichoběžníková metoda (pravidlo, vzorec)

Přesnější integraci funkce f(x) získáme použitím lichoběžníků, kterými aproximujeme danou funkci.



Lichoběžníková metoda (pravidlo, vzorec)



$$\int_a^b f(x)dx \approx h\sum_{i=1}^{n}\frac{y_i + y_{i+1}}{2} = h(y_1/2 + y_{n+1}/2 + \sum_{i=2}^{n} y_i)$$

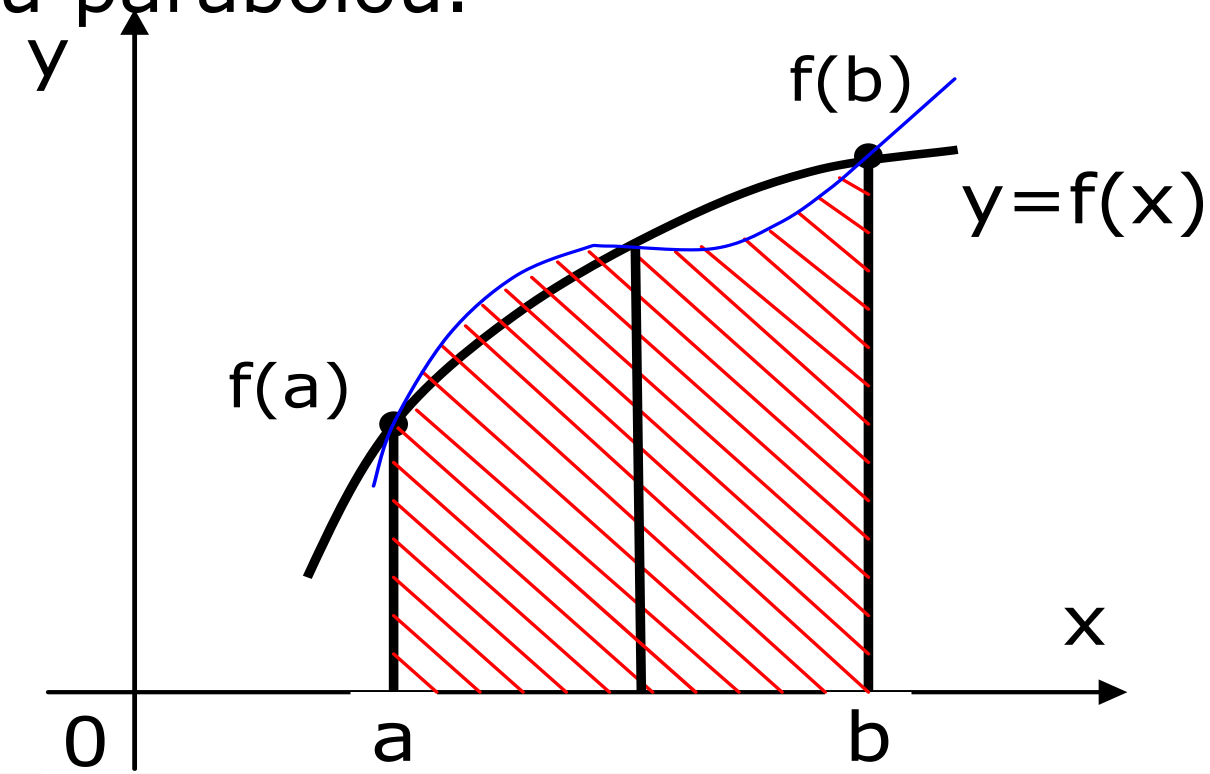
Lichoběžníková metoda

Příklad: funkce pro výpočet integrálu lichoběžníkovou metodou

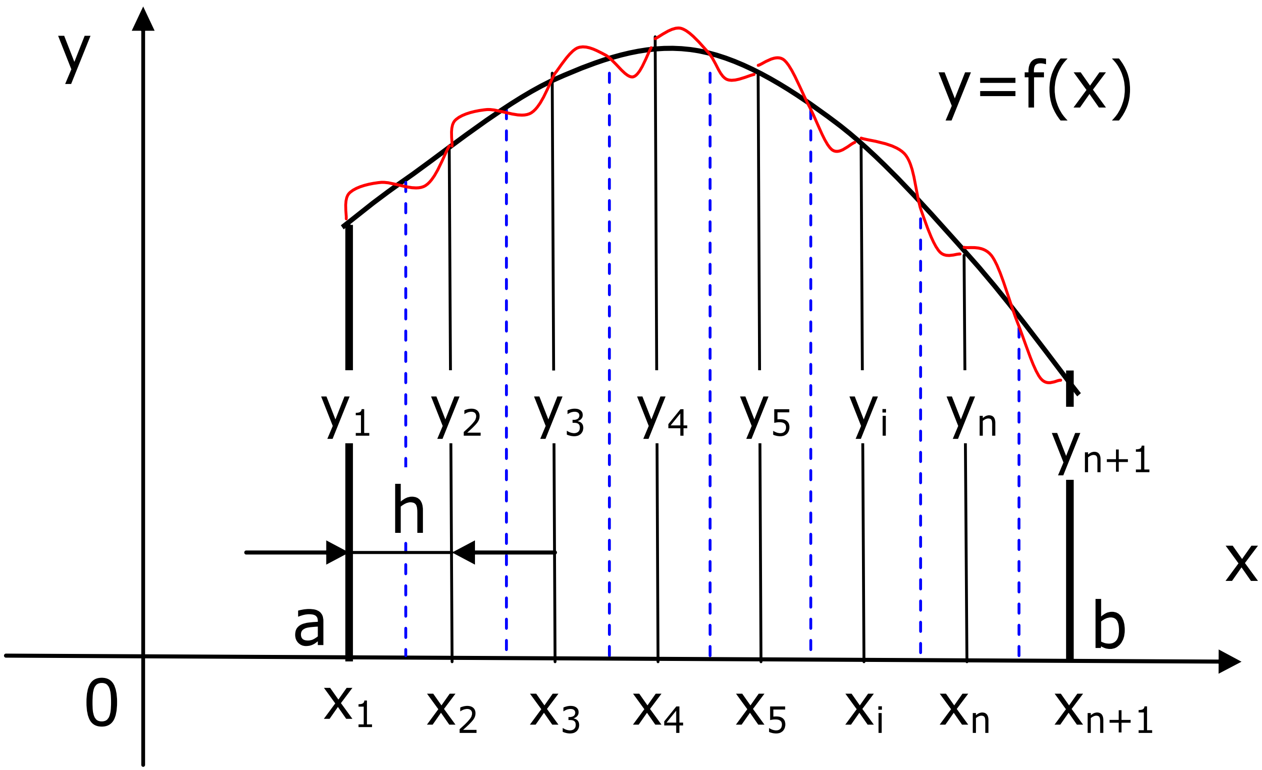
```
double integrate_trapezoid (double a, double b,
                 int n,double (*evalFun)(double))
{
  double step, sum = 0.0;
  step = (b - a) / n;
  for (double x = a+step; x < b-step; x += step)
    sum += evalFun(x);
  sum += (evalFun(a) + evalFun(b))/2;
  sum *= step;
  return sum;
}
```

Simpsonova metoda (pravidlo, vzorec)

- počet dílčích intervalů musí být sudé číslo
- tři sousední body na křivce f(x) se aproximují vhodnou parabolou.



Simpsonova metoda (pravidlo, vzorec)



$$\int_a^b f(x)\ dx \approx \frac{h}{3}(y_1 + 4y_2 + 2y_3 + 4y_4 + 2y_5 + \ldots + 2y_{n-1} + 4y_n + y_{n+1})$$

Simpsonova metoda

Příklad: funkce pro výpočet integrálu Simpsonovou metodou

```
double integrate_simpson (double a, double b,
            int n, double (*evalFun)(double))
{
  int c,tmp = 1;
  double step, sum = 0.0;
  step = (b - a) /(2 * n);
```

Simpsonova metoda

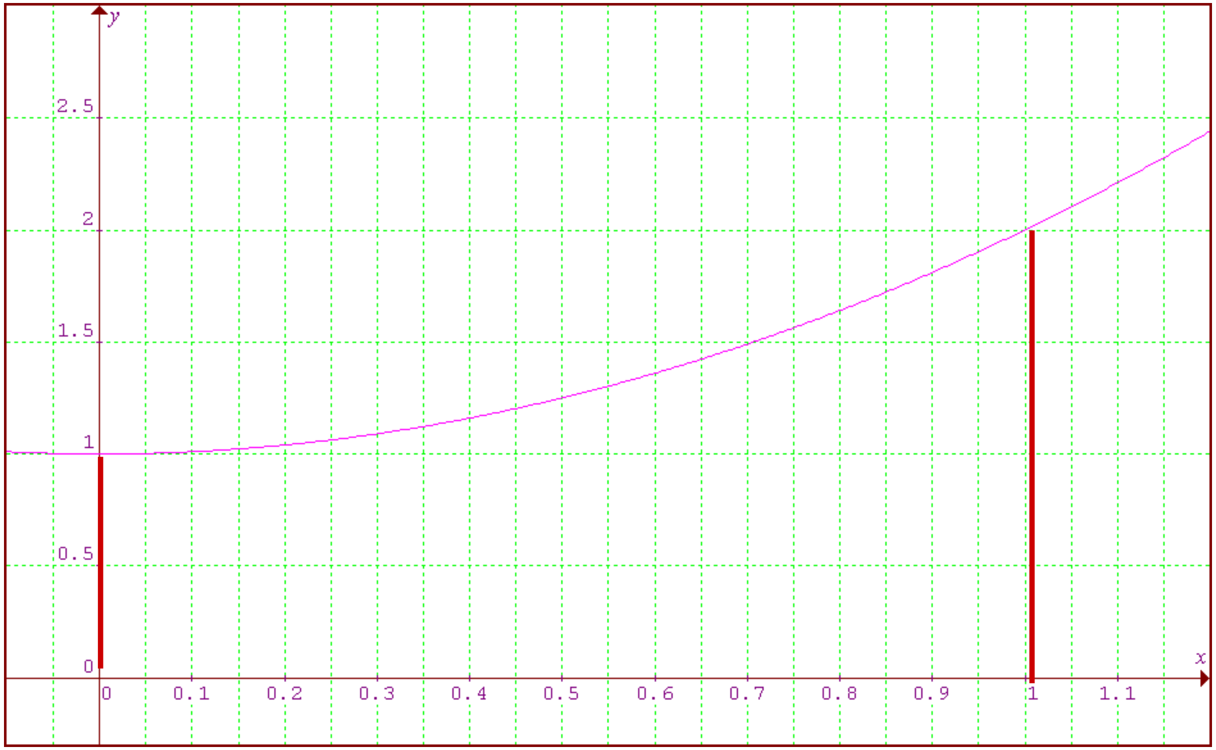
Příklad: funkce pro výpočet integrálu Simpsonovou metodou - pokračování

```
for(double x=a+step; x<b-(step/2); x+=step)
  {
    tmp++;
    c = (tmp&1)?2:4; // lichý/sudý bod?
    sum += c * evalFun (x);
  }
  sum += evalFun (a) + evalFun (b);
  sum *= step/3;
  return sum;
  }
```

Numerický výpočet určitého integrálu

Příklad: použití funkcí pro numerický výpočet určitého integrálu

$$\int_0^1 (x^2+1)\,dx$$



Numerický výpočet určitého integrálu

Příklad: použití funkcí pro numerický výpočet určitého integrálu

$$\int (x^2 + 1)dx = \frac{x^3}{3} + x + C$$

$$\int_a^b f(x)dx = F(b) - F(a)$$

primitivní funkce

$$\int_0^1 (x^2 + 1)dx = 4/3$$

Numerický výpočet určitého integrálu

Příklad: použití funkcí pro numerický výpočet určitého integrálu

```
double parabola (double argument)
{
  return argument * argument + 1;
}
int main (void)
{
  int n,factor,lower_limit=0;
  int upper_limit=1;
  double eps,integral_old,integral_new = 0.0;
  // zadání hodnoty n - počáteční počet
  // dělení úseku
```

Numerický výpočet určitého integrálu

Příklad: použití funkcí pro numerický výpočet určitého integrálu -pokr.

```
  // zadání faktoru násobení – jak se má zvyšovat
  // počet dílčích úseků
  // zadání hodnoty eps - přesnost výpočtu
  do{
    integral_old = integral_new;
    integral_new = integrate_rectangle
           (lower_limit,upper_limit,n,parabola);
    n = n * factor;
  } while(fabs(integral_new - integral_old)>eps);
  // výpočet s hodnotou integrálu
  return 0;
}
```

Numerický výpočet určitého integrálu

- Experimentální výsledky:

n	obdélníková	lichoběžníková	Simpsonova
10	1.285000	1.335000	1.333333
100	1.328350	1.333350	1.333333
1000	1.332834	1.333334	1.333333
10000	1.333283	1.333333	1.333333
100000	1.333328	1.333333	1.333333
přesná	1.333333	1.333333	1.333333

Algoritmy pro numerické výpočty

Kontrolní otázky

1. Co je Hornerovo schéma? Vysvětlete princip implementace funkce pro výpočet hodnoty polynomu pomocí Hornerova schématu.
2. Vysvětlete algoritmus pro hledání kořene rovnice metodou půlení intervalu.
3. Vysvětlete algoritmus pro výpočet integrálu obdélníkovou metodou.
4. Vysvětlete algoritmus pro výpočet integrálu lichoběžníkovou metodou.
5. Vysvětlete algoritmus pro výpočet integrálu Simpsonovou metodou.

Úkoly k procvičení

1. V jazyce C napište funkci pro výpočet hodnoty polynomu pomocí Hornerova schématu.
2. V jazyce C napište funkci pro hledání kořene rovnice metodou půlení intervalu.
3. V jazyce C napište funkci pro výpočet integrálu obdélníkovou metodou.
4. V jazyce C napište funkci pro výpočet integrálu lichoběžníkovou metodou.
5. V jazyce C napište funkci pro výpočet integrálu Simpsonovou metodou.