Integração numérica

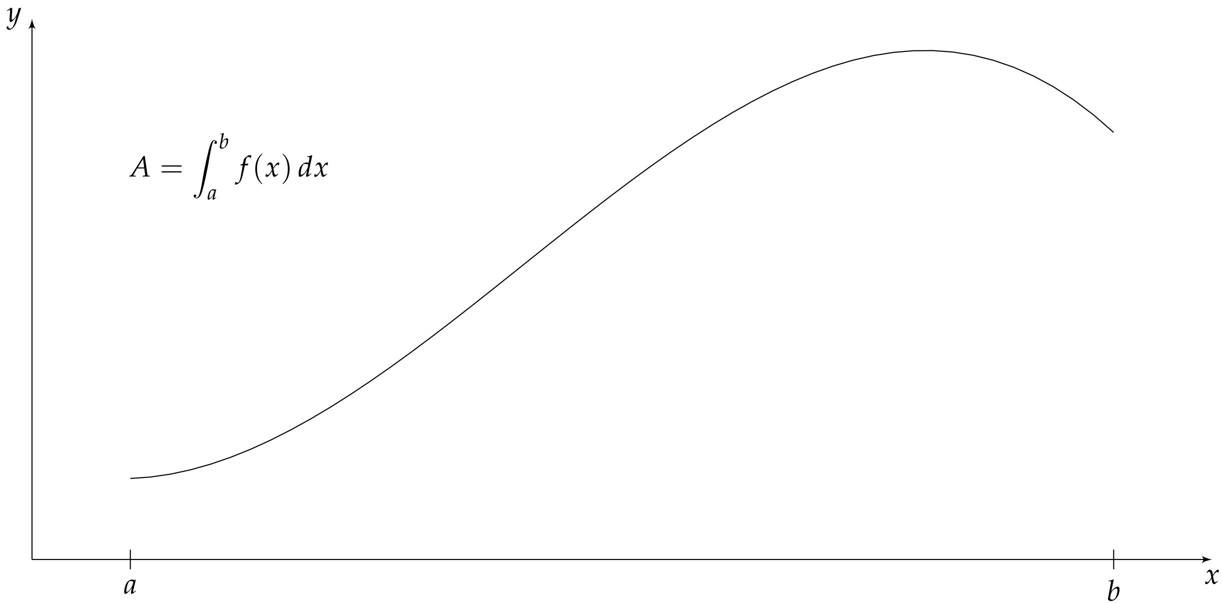
Prof. Luiz T. F. Eleno

LOM3260 — Computação Científica em Python

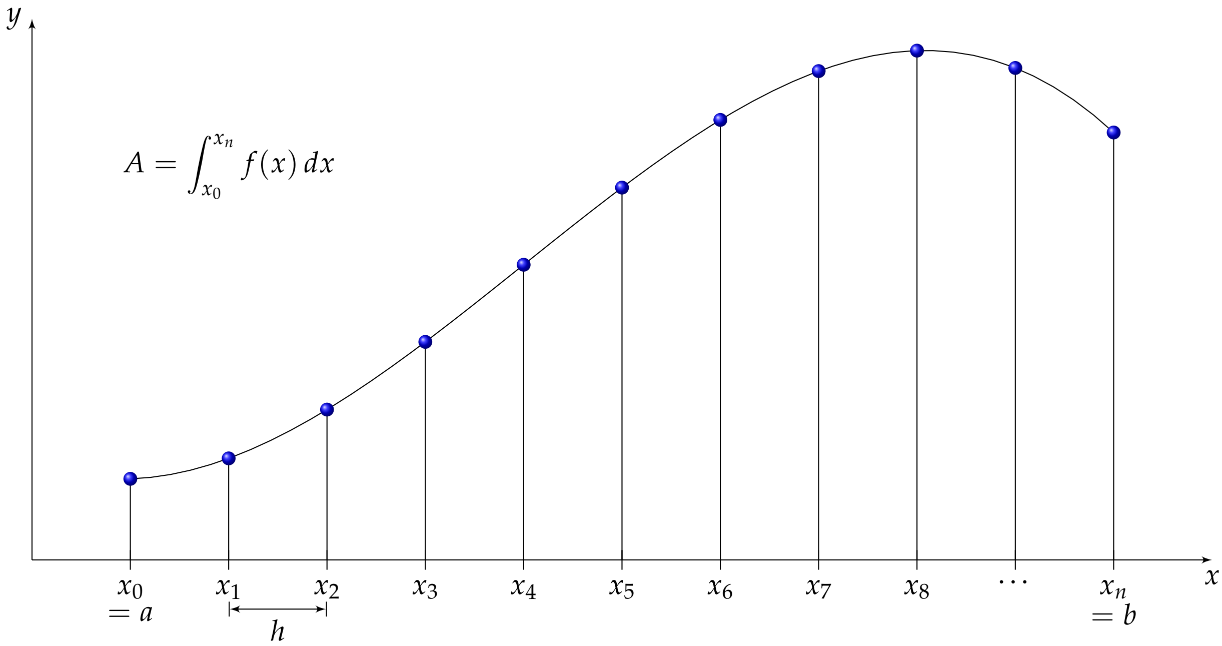
Departamento de Engenharia de Materiais
Escola de Engenharia de Lorena
Universidade de São Paulo

Plano de aula

Regra dos trapézios

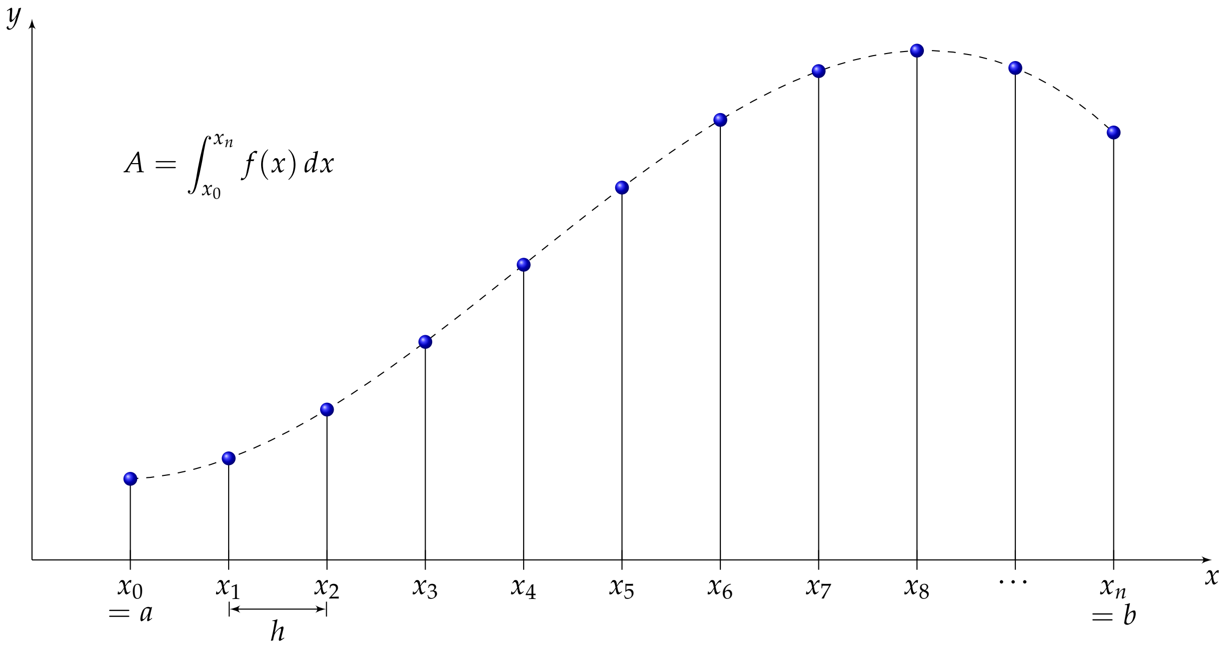


Regra dos trapézios



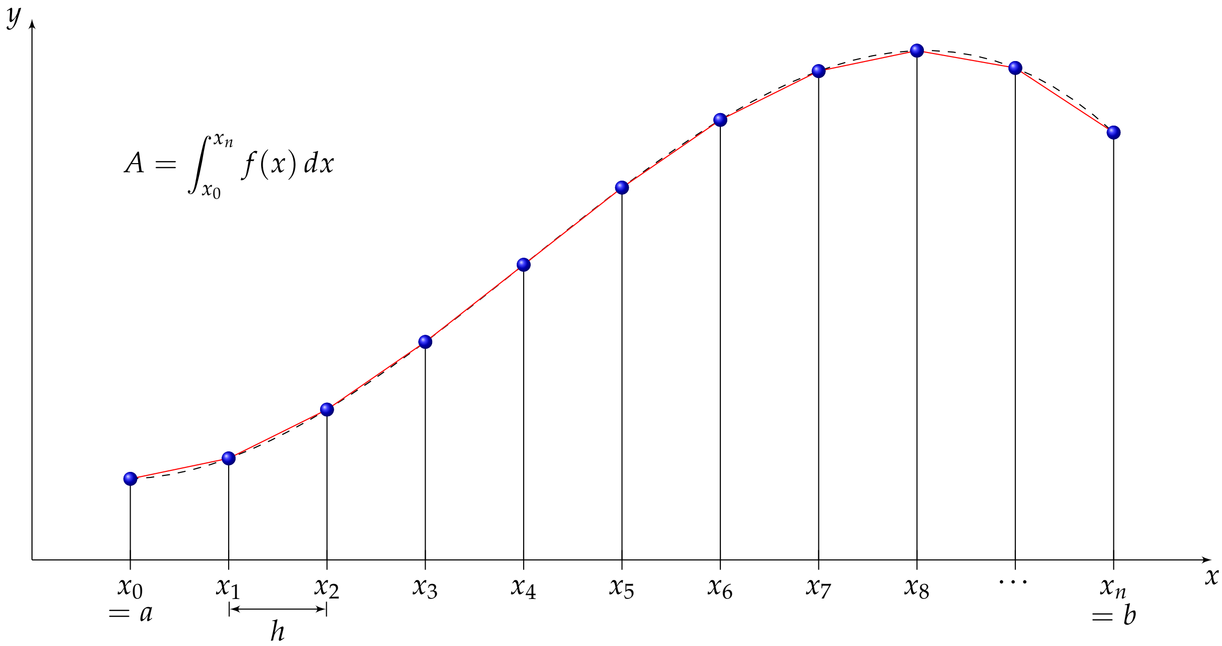
$$h = \frac{b-a}{n}, \qquad x_i = x_{i-1} + h$$

Regra dos trapézios



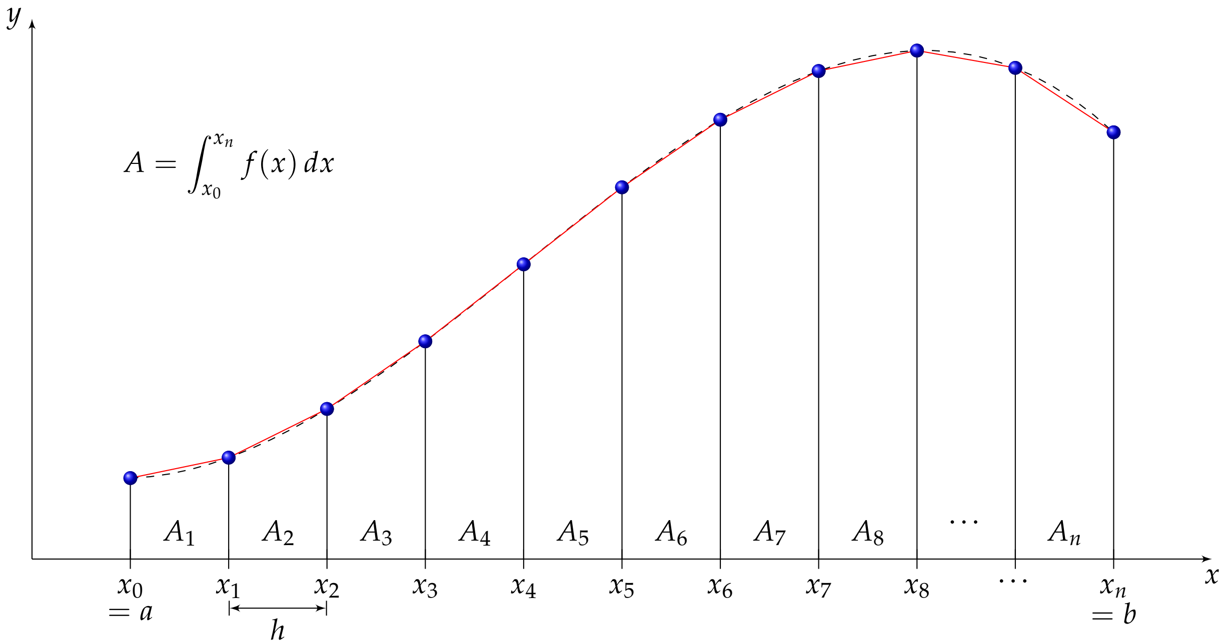
$$h = \frac{b-a}{n}, \qquad x_i = x_{i-1} + h$$

Regra dos trapézios



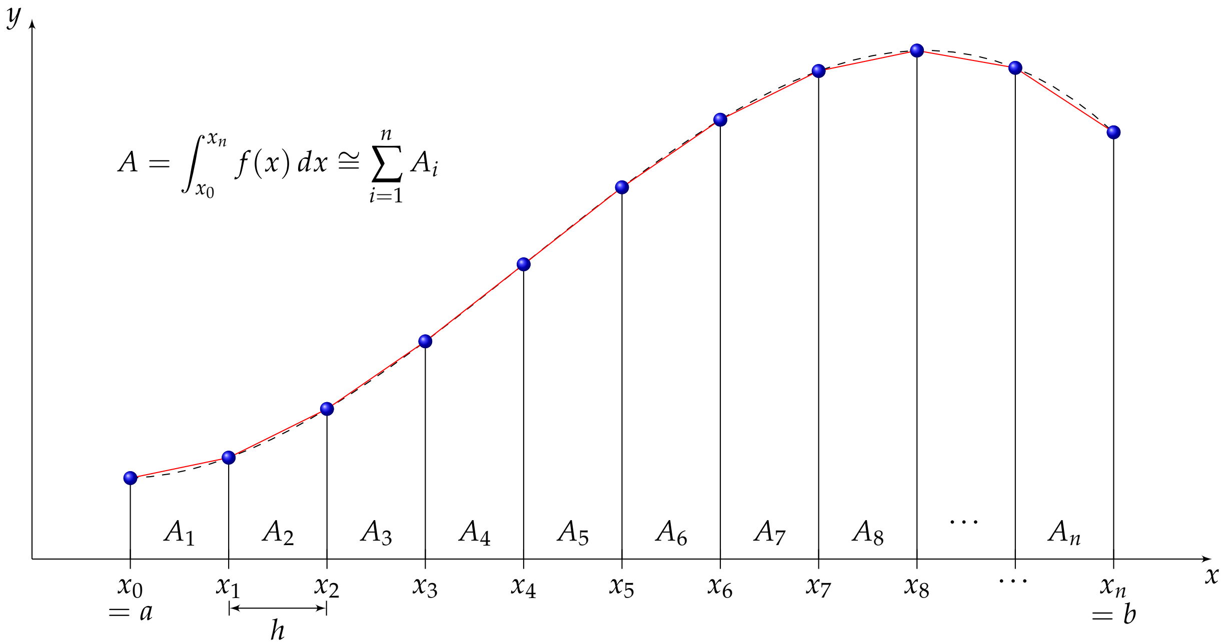
$$h = \frac{b-a}{n}, \qquad x_i = x_{i-1} + h$$

Regra dos trapézios



$$h = \frac{b-a}{n}, \qquad x_i = x_{i-1} + h$$

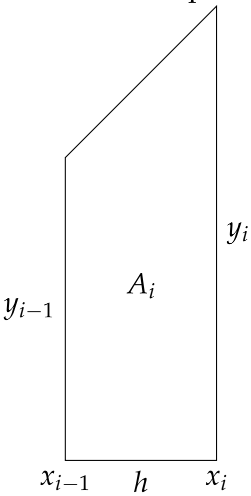
Regra dos trapézios



$$h = \frac{b-a}{n}, \qquad x_i = x_{i-1} + h$$

Regra dos trapézios

- Área de um trapézio:



- Da geometria euclidiana:

$$A_i = h \cdot \frac{y_{i-1} + y_i}{2}$$

onde

$$y_i = f(x_i)\,, \quad y_{i-1} = f(x_{i-1})$$

- Veja que também podemos achar A_i interpolando uma reta entre x_{i-1} e $x_i = x_{i-1} + h$:

$$A_i = \int_{x_{i-1}}^{x_i} P_1(x)dx = \int_{x_{i-1}}^{x_i} \left[y_{i-1} + (y_i - y_{i-1})\frac{x - x_{i-1}}{h} \right] dx$$

$$A_i = y_{i-1}h + (y_i - y_{i-1})\frac{h}{2}$$

$$A_i = h \cdot \frac{y_{i-1} + y_i}{2}$$

Regra dos trapézios

- A área completa é então

$$A \cong \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} h \cdot \frac{y_{i-1} + y_i}{2}$$

$$A \cong \frac{h}{2} \left[\sum_{i=1}^{n} (y_{i-1} + y_i) \right]$$

- Ou seja:

$$A \cong \frac{h}{2} (y_0 + y_1 + y_1 + y_2 + y_2 + y_3 + \ldots + y_{n-2} + y_{n-1} + y_{n-1} + y_n)$$

$$\boxed{A \cong \frac{h}{2} (y_0 + 2y_1 + 2y_2 + 2y_3 + \ldots + 2y_{n-1} + y_n)}$$

Exemplo em python

$\int_0^\pi \operatorname{sen} x\, dx$ pela regra dos trapézios com 100 intervalos

```
import numpy as np


def malha(func, a, b, n):
    x = np.linspace(a, b, n+1, dtype=float)
    y = func(x)
    h = (b - a) / n
    return y, h


def integral_trapezio(func, a, b, n):
    y, h = malha(func, a, b, n)
    S = y[0] + 2. * np.sum(y[1:-1]) + y[-1]
    return .5 * h * S


def f(x):
    return np.sin(x)


print(integral_trapezio(f, 0, np.pi, 100))
```

- O resultado é 1.9998355038874436 (o valor exato é 2)

Análise de erros

- O erro global da integração aproximada pela regra dos trapézios é

$$\boxed{E = -\frac{(b-a)^3}{12n^2} f''(\xi)}$$

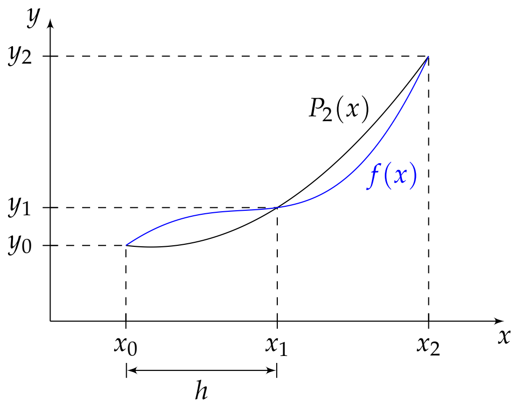
  sendo ξ uma constante (desconhecida) tal que $a \leq \xi \leq b$
- O erro é delimitado da seguinte forma:

$$\boxed{|E| \leq \frac{|b-a|^3}{12n^2} \max \left| f''(x) \right|}$$

- Não vou demonstrar essas fórmulas!

Plano de aula

Regra de Simpson



- E se, ao invés de interpolar linearmente, usarmos uma interpolação quadrática?
- Dividindo o intervalo de integração em duas partes usando x_0, x_1 e x_2, aproximamos $f(x)$ por $P_2(x)$, a paarábola que passa por esses três pontos:

$$f(x) \approx P_2(x) = y_0 + (x - x_0)\frac{\Delta f_0}{h} + (x - x_0)(x - x_1)\frac{\Delta_2 f_0}{2h^2}$$

  com $\Delta f_0 = y_1 - y_0$ e $\Delta_2 f_0 = y_2 - 2y_1 + y_0$
- Aproximamos então a integral de $f(x)$ por

$$A = \int_{x_0}^{x_2} f(x)\,dx \approx \int_{x_0}^{x_2} P_2(x)\,dx$$

Regra de Simpson

- A área é aproximada por

$$A \approx \int_{x_0}^{x_2} \left[y_0 + (x - x_0)\frac{\Delta f_0}{h} + (x - x_0)(x - x_1)\frac{\Delta_2 f_0}{2h^2} \right] dx$$

que fornece

$$A \approx 2hy_0 + 2h\Delta f_0 + \frac{h}{3}\Delta_2 f_0$$

- Mas, como

$$\Delta f_0 = y_1 - y_0$$

$$\Delta_2 f_0 = \Delta f_1 - \Delta f_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0$$

- segue que

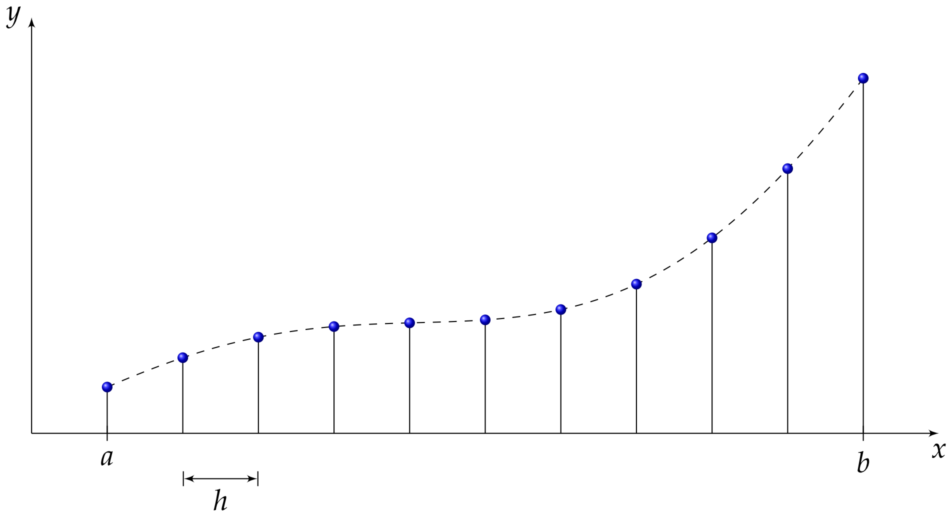
$$A \approx 2hy_0 + 2h\,(y_1 - y_0) + \frac{h}{3}\,(y_2 - 2y_1 + y_0)$$

ou, finalmente,

$$\boxed{A \approx \frac{h}{3}\,(y_0 + 4y_1 + y_2)}$$

- Essa fórmula é conhecida como **Regra de Simpson simples —** simples porque usamos apenas um polinômio em todo o intervalo de interpolação

Regra de Simpson composta



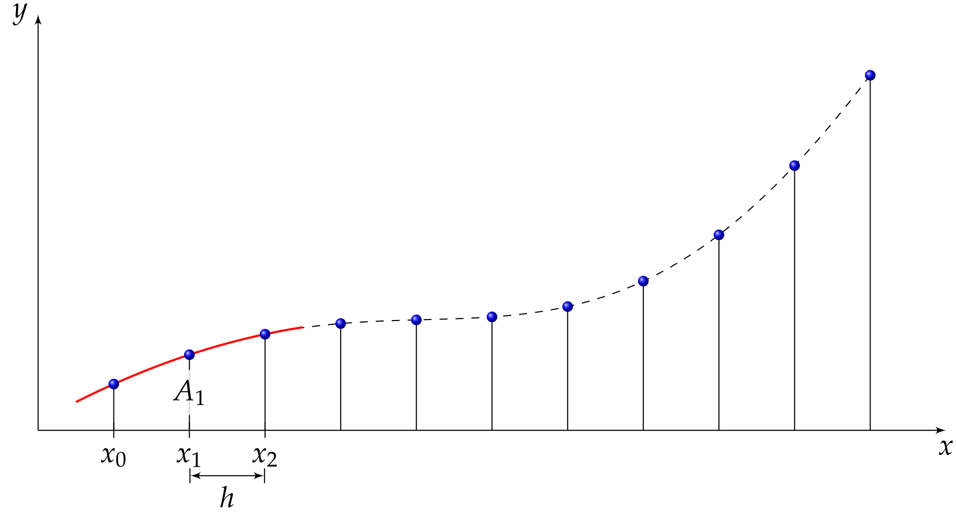
- Para aumentar a precisão, dividimos o intervalo $[a, b]$ em n intervalos iguais de largura

$$h = \frac{b-a}{n}$$

  - n precisa necessariamente ser par!
- Encontramos a parábola interpoladora em cada intervalo $[x_{2i-2}, x_{2i}]$ $(i = 1, 2, \ldots, n/2)$
- As áreas A_i, como já vimos, são dadas por

$$A_i = \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

Regra de Simpson composta



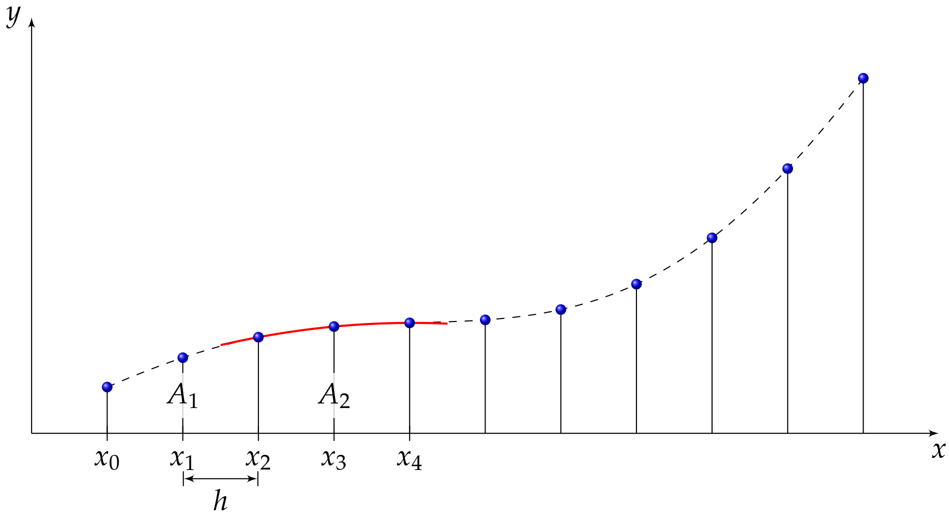
- Para aumentar a precisão, dividimos o intervalo $[a, b]$ em n intervalos iguais de largura

$$h = \frac{b-a}{n}$$

  - n precisa necessariamente ser par!
- Encontramos a parábola interpoladora em cada intervalo $[x_{2i-2}, x_{2i}]$ $(i = 1, 2, \ldots, n/2)$
- As áreas A_i, como já vimos, são dadas por

$$A_i = \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

Regra de Simpson composta



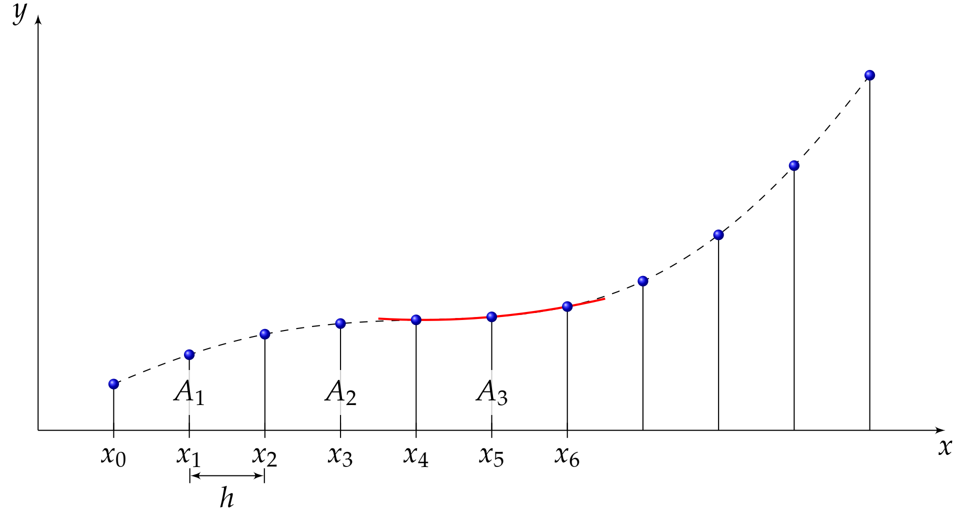
- Para aumentar a precisão, dividimos o intervalo $[a, b]$ em n intervalos iguais de largura

$$h = \frac{b-a}{n}$$

  - n precisa necessariamente ser par!
- Encontramos a parábola interpoladora em cada intervalo $[x_{2i-2}, x_{2i}]$ $(i = 1, 2, \ldots, n/2)$
- As áreas A_i, como já vimos, são dadas por

$$A_i = \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

Regra de Simpson composta



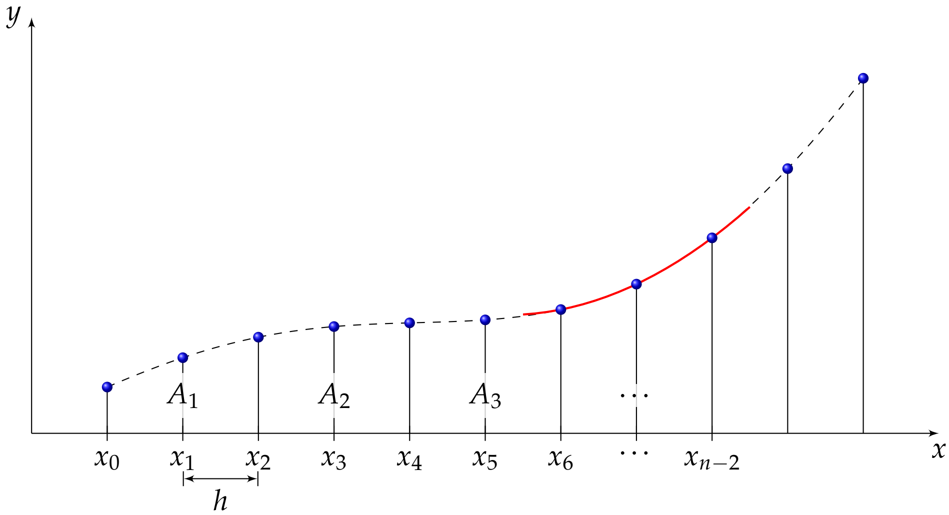
- Para aumentar a precisão, dividimos o intervalo $[a, b]$ em n intervalos iguais de largura

$$h = \frac{b-a}{n}$$

  - n precisa necessariamente ser par!
- Encontramos a parábola interpoladora em cada intervalo $[x_{2i-2}, x_{2i}]$ ($i = 1, 2, \ldots, n/2$)
- As áreas A_i, como já vimos, são dadas por

$$A_i = \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

Regra de Simpson composta



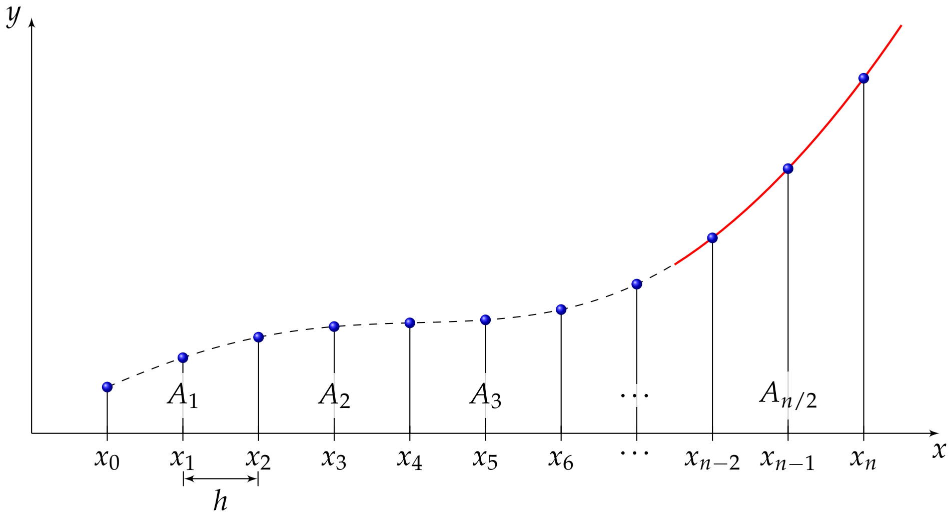
- Para aumentar a precisão, dividimos o intervalo $[a, b]$ em n intervalos iguais de largura

$$h = \frac{b-a}{n}$$

  - n precisa necessariamente ser par!
- Encontramos a parábola interpoladora em cada intervalo $[x_{2i-2}, x_{2i}]$ $(i = 1, 2, \ldots, n/2)$
- As áreas A_i, como já vimos, são dadas por

$$A_i = \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

Regra de Simpson composta



- Para aumentar a precisão, dividimos o intervalo $[a, b]$ em n intervalos iguais de largura

$$h = \frac{b-a}{n}$$

  - n precisa necessariamente ser par!
- Encontramos a parábola interpoladora em cada intervalo $[x_{2i-2}, x_{2i}]$ ($i = 1, 2, \ldots, n/2$)
- As áreas A_i, como já vimos, são dadas por

$$A_i = \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

Regra de Simpson composta

- A regra de Simpson composta então fornece

$$A \approx \sum_{i=1}^{n/2} A_i = \sum_{i=1}^{n/2} \frac{h}{3}(y_{2i-2} + 4y_{2i-1} + y_{2i})$$

ou seja,

$$A \approx \frac{h}{3}\left(y_0 + 4y_1 + y_2 + y_2 + 4y_3 + y_4 + y_4 + 4y_5 + y_6 + \ldots + y_{n-4} + 4y_{n-3} + y_{n-2} + y_{n-2} + 4y_{n-1} + y_n\right)$$

isto é,

$$\boxed{A \approx \frac{h}{3}\left(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n\right)}$$

Exemplo em python

$\int_0^\pi \operatorname{sen} x \, dx$ pela regra dos trapézios com 100 intervalos

```
import numpy as np

def malha(func, a, b, n):
    x = np.linspace(a, b, n+1, dtype=float)
    y = func(x)
    h = (b - a) / n
    return y, h

def integral_simpson(func, a, b, n):
    if int(n) % 2:  # para garantir numero par de intervalos
        n += 1
    y, h = malha(func, a, b, n)
    Si = np.sum(y[1:-1:2])  # posicoes impares
    Sp = np.sum(y[2:-1:2])  # posicoes pares
    S = y[0] + 4. * Si + 2. * Sp + y[-1]
    return h * S / 3.

def f(x):
    return np.sin(x)

print(integral_simpson(f, 0, np.pi, 100))
```

- O resultado é 2.000000010824504 (o valor exato é 2)

Análise de erros

- O erro cometido na integração de uma função $f(x)$ no intervalo $[a, b]$ usando n intervalos (sendo n um número par!) é dado por

$$\boxed{E = -\frac{(b-a)^5}{180n^4} f^{iv}(\xi)}$$

  sendo ξ tal que $a \leq \xi \leq b$.
- Como não conhecemos ξ, podemos delimitar o erro:

$$\boxed{|E| = \frac{|b-a|^5}{180n^4} \max \left| f^{iv}(x) \right|}$$

- Também não demonstrarei esse resultado!

Plano de aula

Funções da biblioteca `scipy.integrate`

- A função `quad`, da biblioteca `scipy.integrate`, calcula uma integral definida usando métodos mais sofisticados do que os vistos anteriormente
- Ajuda completa em https://docs.scipy.org/doc/scipy/reference/integrate.html

Exemplo de aplicação da função `scipy.integrate.quad`

```python
import numpy as np
import scipy.integrate as spi

f = lambda x: np.sin(x)
'''
Construção lambda: equivalente a
def f(x):
    return np.sin(x)
'''

valor, erro = spi.quad(f, 0, np.pi)
print(valor, erro)  # erro é uma estimativa!
```

- Repare no uso da construção `lambda` como alternativa para definir uma função!
  - é muito usada para criar funções simples
- o resultado retornado é 2.0 (com erro da ordem de 10^{-14})