

# Extrapolação de Séries e Aplicações à Distribuição de Tweedie

Wellington José Leite da Silva  
Orientador: Luiz Max Carvalho

December 14, 2023

## Definição

Seja uma sequência  $\{a_k\}_{k \in \mathbb{N}} \subset \mathbb{R}$ . Definimos uma série a partir dessa sequência tomando  $S_n := \sum_{k=1}^n a_k$ . Assim a sequência  $\{S_n\}_{n \in \mathbb{N}}$  é dita uma série de termos reais, e se

$$\lim_{n \rightarrow \infty} S_n = \lim_{n \rightarrow \infty} \sum_{i=0}^n a_i = L < \infty$$

dizemos que a série é convergente à  $L$ .

Algumas séries podem ter vários formatos:  $\sum_{k=1}^{\infty} \frac{1}{k^s}$ ,  $\sum_{k=0}^{\infty} \frac{x^k}{k!}$ ,  
 $\sum_{k=1}^{\infty} \frac{1}{k^3 \sin^2 k}$ ,  $\sum_{k=2}^{\infty} \frac{\sin(kx)}{\ln k}$ ,  $\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}$ ,  $\sum_{k=1}^{\infty} \left( \frac{1}{3^k} + \frac{1}{4^k} \right) \frac{1}{k}$ ,  $\dots$

# Transformação de séries

Seja uma série  $S_n = \sum_{k=1}^n a_k$ , definimos outra sequência

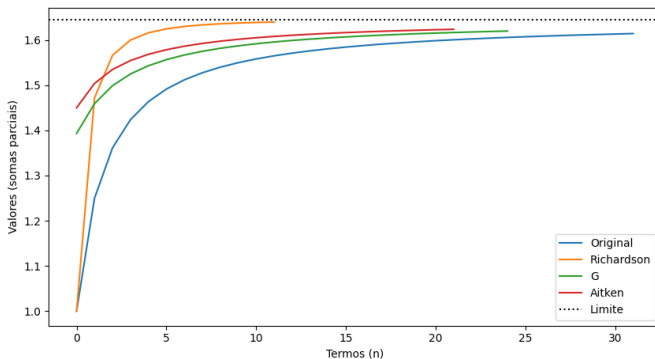
$$S'_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}.$$

esta transformação de sequência é chamada *Aitken*, tem como objetivo convergir mais rápido que a série original.

$$\begin{array}{cccc} s_1 & & & \\ s_2 & s'_2 & & \\ s_3 & s'_3 & s''_3 & \\ s_4 & s'_4 & s''_4 & s'''_4 \\ s_5 & s'_5 & s''_5 & \\ s_6 & s'_6 & & \\ s_7 & & & \end{array}$$

# Problema de Basel

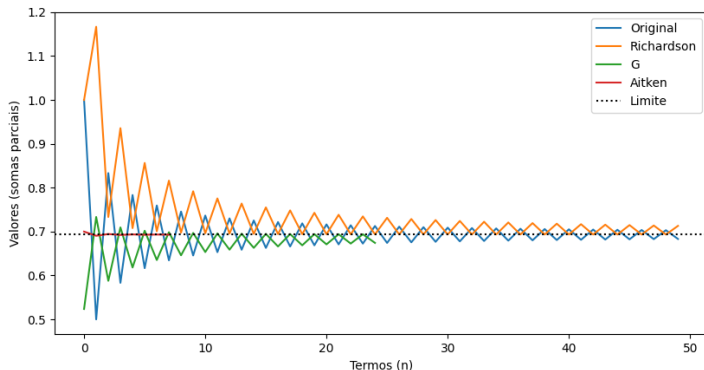
$$S_n = \sum_{k=1}^n \frac{1}{k^2}$$



DELAHAYE, J. P.; GERMAIN-BONNE, B. Résultats négatifs en accélération de la convergence. v. 35, n. 4, p. 443–457, 1980.

# Série Harmônica Alternada

$$S_n = \sum_{k=1}^n (-1)^{k+1} \frac{1}{k}$$



# Aceleração

## Definição

Seja  $\{S_n\}_{n \in \mathbb{N}}$  uma série que converge para um limite  $L < \infty$ . Considere também uma transformação  $T$  tal que mapeie  $\{S_n\}_{n \in \mathbb{N}}$  em  $\{T_n\}_{n \in \mathbb{N}}$ . A nova sequência  $\{T_n\}$  deve satisfazer:

- 1  $\{T_n\}_{n \in \mathbb{N}}$  é convergente;
- 2  $\{T_n\}_{n \in \mathbb{N}}$  converge para o mesmo limite de  $\{S_n\}_{n \in \mathbb{N}}$ ;
- 3  $\{T_n\}_{n \in \mathbb{N}}$  converge mais rapidamente que  $\{S_n\}_{n \in \mathbb{N}}$ , ou seja,

$$\lim_{n \rightarrow \infty} \frac{T_n - L}{S_n - L} = 0.$$

Se  $T$  satisfaz essas três propriedades, dizemos<sup>a</sup> que  $T$  acelera a convergência da sequência  $\{S_n\}_{n \in \mathbb{N}}$  ou que a sequência  $\{T_n\}_{n \in \mathbb{N}}$  converge mais rápido que  $\{S_n\}_{n \in \mathbb{N}}$ .

---

<sup>a</sup>BREZINSKI, C.; ZAGLIA, M. Redivo. Extrapolation Methods: Theory and Practice, 2003.

# Principais objetivos

- Há uma grande literatura sobre aplicação de métodos de extrapolação em um determinado problema, mas poucas em aplicação de forma geral;
- Criação de uma biblioteca em *Python* numericamente estável, com os métodos de aceleração considerados aqui;

Estrutura do trabalho: (i) Métodos de extrapolação selecionados (ii) Criação da biblioteca em *Python* e (iii) aplicação dos métodos na distribuição de Tweedie.

# Algoritmo E

O algoritmo E (Brezinski, 1980) dá suporte teórico aos demais métodos. Se baseia no fato de que podemos escrever a sequência  $\{S_n\}$  convergente com limite  $L$  da seguinte forma

$$S_{n+i} = L + a_1 g_1(n+i) + \dots + a_k g_k(n+i), \quad i = 0, \dots, k,$$

onde as funções  $(g_i(n))$ 's são dadas pela transformação escolhida, e  $L$  pode ser obtido resolvendo um sistema linear.



# Algoritmo E

$$E_n^{(k)} = \frac{\begin{vmatrix} S_n & S_{n+1} & \dots & S_{n+k} \\ g_1(n) & g_1(n+1) & \dots & g_1(n+k) \\ \vdots & \vdots & \ddots & \vdots \\ g_k(n) & g_k(n+1) & \dots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \dots & 1 \\ g_1(n) & g_1(n+1) & \dots & g_1(n+k) \\ \vdots & \vdots & \ddots & \vdots \\ g_k(n) & g_k(n+1) & \dots & g_k(n+k) \end{vmatrix}}.$$

## Teorema

Se  $\lim_{n \rightarrow \infty} S_n = L$ ,  $\lim_{n \rightarrow \infty} g_i(n+1)/g_i(n) = b_i \neq 1 \ \forall i$ , com  $b_i \neq b_j \ \forall i \neq j$ , então  $\lim_{n \rightarrow \infty} E_n^{(k)} = L \ \forall k$ .

**Prova:** Veja Apêndice.

# Método de Aitken

Assim como as demais transformações de sequência podemos escrever a transformação de Aitken como

$$\begin{aligned} T_n &= \frac{\begin{vmatrix} S_n & S_{n+1} \\ a_n & a_{n+1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ a_n & a_{n+1} \end{vmatrix}}, \\ &= \frac{S_n(S_n - S_{n-1}) - S_{n+1}(S_{n+1} - S_n)}{(S_n - S_{n-1}) - (S_{n+1} - S_n)}, \\ &= \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}. \end{aligned}$$

# Método de Richardson

Método para séries de termos positivos, dado por

$$T_n = S_{2n} + \frac{S_{2n} - S_n}{2^p - 1},$$

onde  $p$  é variável ajustável. Para  $p = 1$

$$T_n = 2S_{2n} - S_n.$$

# Método de Richardson

A fim de exemplo, observe a Tabela 1, aplicando a transformação de Richardson com  $p = 1$  para o problema de Basel,  $\zeta(2) = \sum_{k=1}^{\infty} \frac{1}{k^2}$ :

Table: Richardson ( $p=1$ ) aplicado à Basel

| n | $S_n$  | $T_n$  | $T'_n$ | $T''_n$ |
|---|--------|--------|--------|---------|
| 1 | 1.0000 | 1.5000 | 1.6296 | 1.6444  |
| 2 | 1.2500 | 1.5972 | 1.6425 |         |
| 4 | 1.4236 | 1.6312 |        |         |
| 8 | 1.5274 |        |        |         |

Note que, o limite da série é  $\pi^2/6 = 1.6449\dots$  Para este exemplo simples podemos provar que Richardson acelera o problema de Basel. Veja a prova no Apêndice.

# Método Epsilon

O Método de Epsilon para séries de convergência lenta. Seja a série  $\{S_n\}_{n \in \mathbb{N}}$ , primeiro definimos

$$\varepsilon_n^{(-1)} = 0 \quad \text{e} \quad \varepsilon_n^{(0)} = S_n$$

e iterativamente

$$\varepsilon_n^{(k+1)} = \varepsilon_{n+1}^{(k-1)} + \frac{1}{\varepsilon_{n+1}^{(k+1)} - \varepsilon_n^{(k)}}.$$

A sequência acelerada é definida pelos valores de  $k$  pares, os ímpares são sequências intermediárias.

## Algoritmo G

Com foco em integrais impróprias o Algoritmo G é o penúltimo método de extrapolação apresentado aqui. Seja a série  $S_n = \sum_{k=1}^n a_k$ , então definiremos duas sequências auxiliares da seguinte forma

$$s_n^{(0)} = 1, \quad r_n^{(1)} = a_n, \quad n = 0, 1, \dots,$$

e iterativamente

$$s_n^{(k+1)} = s_{n+1}^{(k)} \left( \frac{r_{n+1}^{(k+1)}}{r_n^{(k+1)}} - 1 \right), \quad k, n = 0, 1, \dots,$$
$$r_n^{(k+1)} = r_{n+1}^{(k)} \left( \frac{s_{n+1}^{(k+1)}}{s_n^{(k+1)}} - 1 \right), \quad k = 1, 2, \dots; n = 0, 1, \dots$$

# Algoritmo G

Daí podemos obter  $G_n^{(k)}$ 's pelo método iterativo

$$G_n^{(k)} = G_n^{(k-1)} - \frac{G_{n+1}^{(k-1)} - G_n^{(k-1)}}{r_{n+1}^{(k)} - r_n^{(k)}} r_n^k, \quad k = 1, 2, \dots; n = 0, 1, \dots,$$

com  $G_n^{(0)} = S_n$ .

# Método de Levin generalizado

O último método é o Método de Levin, se diferencia dos outros por ser uma versão mais simples do Algoritmo E. Sendo  $S_n = \sum_{k=1}^n a_k$ , a nova sequência extrapolada é definida como

$$W_n^{(k)} = \frac{M_n^{(k)}}{N_n^{(k)}}$$

onde

$$M_n^{(0)} = \frac{S_n}{g(n)},$$

$$M_n^{(k+1)} = \frac{M_{n+1}^{(k)} - M_n^{(k)}}{a_{n+k}^{-1} - a_{n+1}^{-1}},$$



# Método de Levin generalizado

e

$$N_n^{(0)} = \frac{1}{g(n)},$$

$$N_n^{(k+1)} = \frac{N_{n+1}^{(k)} - N_n^{(k)}}{a_{n+k}^{-1} - a_{n+1}^{-1}}.$$

Note que, o método é constituído por uma função  $g(\cdot)$  livre que Levin sugere 3 opções para  $g$

- **t - variante:**  $g(n) = a_{n+1}$ ;
- **u - variante:**  $g(n) = na_n$ ;
- **v - variante:**  $g(n) = a_n a_{n+1} / (a_{n+1} - a_n)$ .

---

LEVIN, David. Development of non-linear transformations for improving convergence of sequences. v. 3, n. 1-4, p. 371–388, 1972.

# Mpmath

Mpmath é uma biblioteca de precisão ponto flutuante em *python*, usada em bibliotecas como *Sage* e *SymPy*. Representa números com precisão de ponto flutuante

$$(-1)^s x \cdot 2^y$$

por  $(s, x, y, b)$ , sendo  $x$  e  $y$  inteiros e  $b$  (padrão  $b = 53$  bits) uma variável global que guarda a precisão de  $x$ .

A biblioteca também provem de uma função para avaliar séries

```
from mpmath import nsum, inf  
  
nsum(lambda x: 1/x**2, [1, inf])  
✓ 0.0s  
  
mpf('1.6449340668482264')
```

$$S_n = \sum_{k=2}^{\infty} \frac{1}{k(\log(k))^2}$$

```
from mpmath import nsum, log, inf  
  
nsum(lambda x: 1/(x * log(x)**2), [2, inf])  
✓ 0.0s  
  
mpf('1.9105629507008053')
```

Porém com 15 casas decimais de precisam o resultado deveria ser 2.10974280123689.

# Escala Logarítmica

Operar com números muito pequenos ou muito grandes pode trazer erros numéricos.

- multiplicação  $\rightarrow$  adição;
- divisão  $\rightarrow$  subtração;
- exponenciação  $\rightarrow$  multiplicação;
- adição  $\rightarrow$  logSumExp:

$$\text{LSE}(x_1, x_2, \dots, x_n) = x_{\max} + \log \sum_{i=1}^n \exp(x_i - x_{\max});$$

- $\vdots$

# Funções principais

A função principal da biblioteca é *esum*

- *Uma série*: Na forma de uma função  $f : \mathbb{N} \rightarrow \mathbb{R}$  que retorna os termos da série;
- *O método*: Podendo ser "Aitken", "Richardson", "Epsilon", "G", "Levin-t", "Levin-u", "Levin-v" ou "None" para soma usual;
- *O erro* ( $\varepsilon$ ): Um valor para o critério de parada  $|T_{n-1} - T_n| < \varepsilon$ ;
- *Um booleano*: Os cálculos são feitos em escala logarítmica, e é retornado em escala normal, mas caso o usuário prefira trabalhar em escala logarítmica basta passar "True" e será retornado um objeto com sinal e valor na escala logarítmica;
- *A precisão*: Se a precisão for 53 usamos o float64 padrão do Python, caso contrario a estrutura aritmética de ponto flutuante da *mpmath*.

## esum vs. nsum

|            | esum   |            | nsum   |            |
|------------|--------|------------|--------|------------|
|            | value  | time (seg) | value  | time (seg) |
| Atiken     | 1.9529 | 0.1451     | -      | -          |
| Richardson | 1.9648 | 0.0954     | 1.9105 | 0.0040     |
| Epsilon    | 1.9529 | 0.1541     | 1.9105 | 0.0359     |
| G          | 1.9450 | 0.5862     | -      | -          |
| Levin-t    | 1.9529 | 0.2057     | 1.9105 | 0.0769     |
| Levin-u    | 2.0142 | 0.1866     | 1.9105 | 0.0790     |
| Levin-v    | 2.0257 | 0.1939     | 1.9105 | 0.0764     |
| None       | 1.9350 | 0.0246     | 1.9105 | 0.0011     |

## esum vs. nsum

|            | esum   |            | nsum   |            |
|------------|--------|------------|--------|------------|
|            | value  | time (seg) | value  | time (seg) |
| Atiken     | 1.9529 | 0.1489     | -      | -          |
| Richardson | 1.9648 | 0.1737     | 1.9399 | 0.0126     |
| Epsilon    | 1.9529 | 0.1809     | 2.0110 | 0.2795     |
| G          | 1.9450 | 0.6675     | -      | -          |
| Levin-t    | 1.9529 | 0.2485     | 1.9399 | 0.7834     |
| Levin-u    | 2.0142 | 0.2129     | 1.9399 | 0.7698     |
| Levin-v    | 2.0257 | 0.2340     | 1.9399 | 0.7757     |
| None       | 1.9350 | 0.0541     | 1.9399 | 0.0043     |

A biblioteca pode ser encontrada em  
<https://pypi.org/project/extrapolation>.

Os modelos exponenciais de dispersão (EDM) têm uma função de densidade de probabilidade da forma

$$f(z; \mu, \phi) = a(z, \phi) \exp \left[ \frac{1}{\phi} \{z\theta - \kappa(\theta)\} \right],$$

onde  $\kappa()$  e  $a()$  são funções conhecidas, porém  $a()$  não tem formula fechada. A distribuição de Tweedie é um EDM, que satisfaz

$$\text{Var}(X) = \sigma(E[X])^p,$$

para qualquer  $p$  fora do intervalo  $(0, 1)$ .



## Tweedie $p > 2$

Para alguns valores específicos de  $p$  obtemos distribuições famosas como a Normal ( $p = 0$ ), Poisson ( $p = 1$ ), Gamma ( $p = 2$ ) e Gaussiana Inversa ( $p = 3$ ).

Jørgensen provou que para  $1 < p < 2$  e  $p > 2$ , que podemos escrever  $a()$  como uma série. Para  $1 < p < 2$

$$a(z, \phi) = \frac{1}{z} V(z, \phi, p),$$

com  $V(z, \phi, p) = \sum_{k=1}^{\infty} V_k$  e

$$V_k = \frac{\Gamma(1 + \alpha k) \phi^{k(\alpha-1)} (p-1)^{\alpha k}}{\Gamma(k) (p-2)^k z^{\alpha k}} (-1)^k \sin(-k\pi\alpha).$$

# Forma da Tweedie

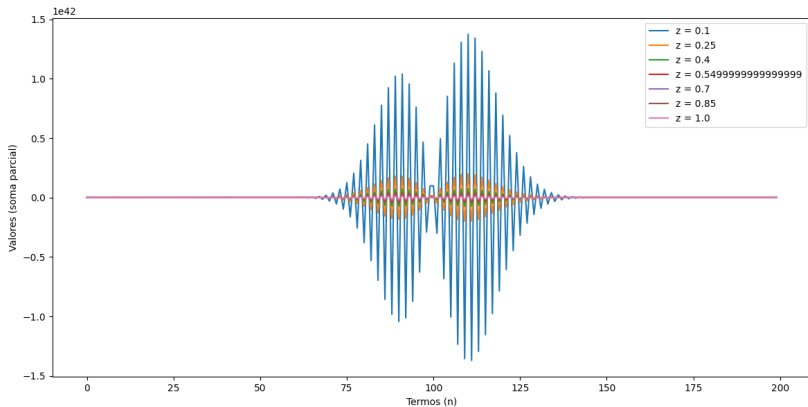


Figure:  $\alpha = 0.01$ ,  $\theta = -1/2$ ,  $z \in (0.1, 1)$ ,  $n = 200$

# Forma da Tweedie

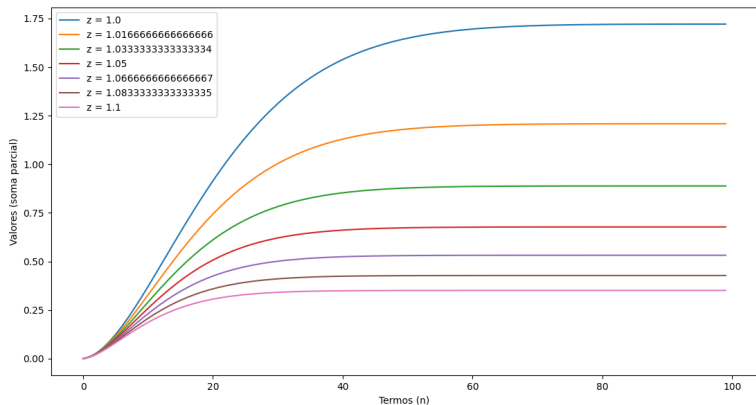


Figure:  $\alpha = 0.99$ ,  $\theta = -1/2$ ,  $z \in (1, 1.1)$ ,  $n = 100$

# Experimento de Dias & Ribeiro (2019)

É implementado em Dias uma versão para avaliação da Tweedie no intervalo  $p > 2$ , que atinge resultados melhores que os do R. Sendo  $\alpha = \frac{2-p}{1-p}$ ,

- A f.d.p. integra 1;
- 1000-bits de precisão;
- $\alpha \in [0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99]$ ;
- Quadratura com 1000 pontos (para  $\alpha = 0.99$  usasse 10000);
- Intervalo  $(1e-6, 20)$  (para  $\alpha = 0.99$  usasse  $(1e-6, 50)$ );

# Dias vs. tweedie

|          |        | Dias       |                    | tweedie     |                    |
|----------|--------|------------|--------------------|-------------|--------------------|
| $\alpha$ | pontos | I          | erro ( $ I - 1 $ ) | I           | erro ( $ I - 1 $ ) |
| 0.01     | 1000   | 1.0000e-00 | 7.0451e-07         | nan         | nan                |
| 0.1      | 1000   | 1.0000e-00 | 2.0245e-08         | 2.9811e+08  | 2.9811e+08         |
| 0.2      | 1000   | 1.0000e-00 | 1.2977e-11         | 7.2761e+15  | 7.2761e+15         |
| 0.3      | 1000   | 1.0000e-00 | 7.7060e-13         | nan         | nan                |
| 0.4      | 1000   | 1.0000e-00 | 6.0196e-13         | 1.6752e+35  | 1.6752e+35         |
| 0.5      | 1000   | 1.0000e-00 | 9.1116e-13         | 1.0000e-00  | 2.5501e-13         |
| 0.6      | 1000   | 1.0000e-00 | 3.2041e-13         | nan         | nan                |
| 0.7      | 1000   | 1.0000e-00 | 9.9475e-14         | 8.0891e+87  | 8.0891e+87         |
| 0.8      | 1000   | 1.0000e-00 | 9.6034e-13         | 2.3095e+129 | 2.3095e+129        |
| 0.9      | 1000   | 1.0000e-00 | 1.5676e-11         | 8.9866e+239 | 8.9866e+239        |
| 0.99     | 10000  | 1.0000e-00 | 3.3810e-09         | nan         | nan                |

# Dias vs. métodos de extrapolação

- Mudança na estrutura (resultando numa perda de 20% em tempo);
- Foi avaliado com todos os métodos 3 vezes para medições de tempo;
- Para analisar melhor os modelos atribuímos um 'score' que é dado por

$$\text{score} = \frac{\text{erroDias} - \text{erroMetodo}}{\min(\text{erroDias}, \text{erroMetodo})}.$$

# Dias vs. métodos de extrapolação

O método com melhores resultados foi o Levin com variante t

| $\alpha$ | erro       |                    | tempo (100/seg) |                    | score   |
|----------|------------|--------------------|-----------------|--------------------|---------|
|          | Dias       | Dias com 'Levin-t' | Dias            | Dias com 'Levin-t' |         |
| 0.01     | 7.0451e-07 | 7.0451e-07         | 17.5309         | 30.7279            | 0.0     |
| 0.1      | 2.0245e-08 | 2.0245e-08         | 1.8232          | 3.4692             | 0.0     |
| 0.2      | 1.2977e-11 | 1.2981e-11         | 1.0835          | 2.0639             | -0.0003 |
| 0.3      | 7.7060e-13 | 6.8522e-13         | 0.8759          | 1.6194             | 0.1245  |
| 0.4      | 6.0196e-13 | 5.4878e-13         | 0.9057          | 1.6796             | 0.0969  |
| 0.5      | 9.1116e-13 | 1.1095e-12         | 0.8929          | 1.6510             | -0.2177 |
| 0.6      | 3.2041e-13 | 8.3910e-13         | 1.1767          | 2.0020             | -1.6188 |
| 0.7      | 9.9475e-14 | 9.3991e-13         | 1.3225          | 2.1988             | -8.4486 |
| 0.8      | 9.6034e-13 | 8.3093e-12         | 1.4956          | 2.4455             | -7.6524 |
| 0.9      | 1.5676e-11 | 1.4293e-11         | 1.2956          | 2.2279             | 0.0968  |
| 0.99     | 3.3810e-09 | 3.3825e-09         | 1.3509          | 2.3331             | -0.0004 |

*Veja os resultados completos no Apêndice.*

# Ideias Futuras

- Criar tratamentos para diferentes tipos de problemas.
- As funções principais retornam uma sequência e não só o valor de extrapolação (permite aplicar vários métodos, mas é menos eficiente).
- Buscar outros critérios de, além do  $|T_{n-1} - T_n| < \varepsilon$ .