FUNDAÇÃO GETULIO VARGAS
ESCOLA DE MATEMÁTICA APLICADA

WELLINGTON JOSÉ LEITE DA SILVA

MÉTODOS DE EXTRAPOLAÇÃO DE SÉRIES APLICADOS À DISTRIBUIÇÃO DE TWEEDIE

Rio de Janeiro
2023

WELLINGTON JOSÉ LEITE DA SILVA

MÉTODOS DE EXTRAPOLAÇÃO DE SÉRIES APLICADOS À DISTRIBUIÇÃO DE TWEEDIE

Trabalho de conclusão de curso apresentado à Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: Análise Numérica e Estatística.

Orientador: Luiz Max Fagundes de Carvalho

Rio de Janeiro

2023

Ficha catalográfica elaborada pela BMHS/FGV
Wellington José Leite da Silva. – 2023.
Silva, Wellington 43f.
Métodos de extrapolação de séries aplicados à distribuição de Tweedie/
Trabalho de Conclusão de Curso – Escola de Matemática Aplicada.
Advisor: Luiz Max Fagundes de Carvalho.
Includes bibliography.
1. Análise Numérica 2. Extrapolação de Séries 3. Estatística I. Carvalho, Luiz Max II. Escola de Matemática Aplicada III. Métodos de extrapolação de séries

WELLINGTON JOSÉ LEITE DA SILVA

MÉTODOS DE EXTRAPOLAÇÃO DE SÉRIES APLICADOS À DISTRIBUIÇÃO DE TWEEDIE

Trabalho de conclusão de curso apresentado à Escola de Matemática Aplicada (FGV/EMAp) como requisito para o grau de bacharel em Matemática Aplicada.

Área de estudo: Análise Numérica e Estatística.

E aprovado em / /
Pela comissão organizadora

Luiz Max Fagundes de Carvalho
Orientador – Escola de Matemática Aplicada

Moacyr Alvim Horta Barbosa da Silva
Membro – Escola de Matemática Aplicada

Hugo de la Cruz Cancino
Membro – Escola de Matemática Aplicada

Resumo

Aproximar o limite de uma série pode ser problemático por diversas razões, seja, por precisão numérica ou impossibilidade de avaliar muitos termos da série. Uma estratégia para lidar com esses problemas são os métodos de extrapolação de séries. Aplicando certas transformações à uma série podemos obter uma sequência que convirja mais rápido que a série original para o mesmo limite. Este trabalho explora implementações desses métodos, desenvolvendo uma biblioteca que visa minimizar imprecisões numéricas de fácil uso. E por fim, vamos aplicar estes métodos num problema pratico, sendo ele avaliar a função de densidade de probabilidade (f.d.p.) da distribuição de Tweedie, que pode ser pode ser estendido a outros problemas envolvendo avaliação de séries.

Palavras-chave: Séries, Análise Numérica, Métodos de Extrapolação, Python, Tweedie.

Abstract

Approximating the limit of a series can be problematic for various reasons, whether due to numerical precision or the impossibility of evaluating many terms in the series. One strategy to address these issues is the use of series extrapolation methods. By applying certain transformations to a series, we can obtain a sequence that converges faster than the original series to the same limit. This work explores implementations of these methods, developing a library that aims to minimise numerical inaccuracies while being user-friendly. Finally, we apply these methods to a practical problem, namely, evaluating the probability density function (f.d.p.) of the Tweedie distribution. This approach can be extended to other problems involving series evaluation.

Keywords: Series, Numerical Analysis, Extrapolation Methods, Python, Tweedie.

Sumário

1 Introdução

O estudo de séries é essencial em diversas áreas do conhecimento, tais como matemática, estatística, ciência da computação, finanças e física. Sua relevância envolve não só suas aplicações como também lidar com questões de convergência, precisão e dificuldade em avaliar termos. Neste contexto, apresentaremos os métodos de extrapolação de séries, mas antes introduziremos algumas definições importantes:

Definição 1. *Seja* $\{a_i\}_{i\in\mathbb{N}\cup\{0\}} \subset \mathbb{R}$ *uma sequência. Podemos definir uma série a partir dessa sequência tomando* $S_n := \sum_{i=0}^{n} a_i$*. A sequência* $\{S_n\}_{n\in\mathbb{N}\cup\{0\}}$ *é dita uma série de termos reais, e podemos avaliar o limite de* $\{S_n\}_{n\in\mathbb{N}\cup\{0\}}$*, sendo,*

$$\lim_{n\to\infty} S_n = \lim_{n\to\infty} \sum_{i=0}^{n} a_i.$$

Se o limite converge para um valor real, dizemos que a série $\{S_n\}$ *converge, caso contrario dizemos que ela diverge.*

Além disso, podemos classificar a taxa de convergência de uma série convergente converge, da seguinte forma:

Definição 2. *Seguindo (SMALL, 2010), seja* $\{S_n\}_{n\in\mathbb{N}\cup\{0\}}$ *uma série convergente, com limite* $L = \lim_{n\to\infty} S_n$ *e seja* μ *tal que*

$$\mu = \lim_{n\to\infty} \frac{|L - S_{n+1}|}{|L - S_n|},$$

onde $\mu \in [0, 1]$*. Dizemos que a série* $\{S_n\}_{n\in\mathbb{N}\cup\{0\}}$ *converge com taxa:*

- *Logarítmica (ou sub-linear), se* $\mu = 1$*;*
- *Linear, se* $\mu \in (0, 1)$*;*
- *Hiper-linear (ou super-linear), se* $\mu = 0$*.*

Em situações em que a taxa de convergência é baixa ou a avaliação de termos é muito custosa, o método comum de somar os termos pode não ser suficiente. Surgem, então, os métodos de extrapolação de séries, nos quais obtemos outra sequência a partir de uma transformação da sequência original.

Definição 3. *Seguindo (BREZINSKI; ZAGLIA, 2003), seja* $\{S_n\}_{n\in\mathbb{N}}$ *uma sequência de números (reais ou complexos) que converge para um limite* L*. Considere também uma*

transformação T tal que mapeie $\{S_n\}_{n\in\mathbb{N}}$ em $\{T_n\}_{n\in\mathbb{N}}$. Para que essa transformação seja útil na extrapolação de séries, a nova sequência $\{T_n\}$ deve satisfazer as seguintes propriedades para pelo menos uma família de sequências $\{S_n\}$:

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,*

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

Se T satisfaz essas três propriedades, dizemos 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 $\{S_n\}_{n\in\mathbb{N}}$.

Como exemplo de transformação de sequência, considere

$$T_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n},\ n = 0, 1, \ldots,$$

que é a transformação de (AITKEN, 1927), abordada mais detalhadamente na Seção 2.2. É importante destacar que a definição acima sugere que T deve acelerar pelo menos uma família de sequências $\{S_n\}_{n\in\mathbb{N}}$. No entanto, (DELAHAYE; GERMAIN-BONNE, 1980) prova que não pode existir uma transformação universal T que acelere todas as famílias de sequências convergentes, destacando a importância de estudar diferentes transformações para abordar novos problemas.

A Figura 1 ilustra a aplicação de diferentes transformações à série harmônica alternada. Observa-se que transformações como Aitken e G aproximam-se do limite da série em poucos termos. Enquanto o Richardson não chega a ser mais rápido que a série original, o que não faz dele um método ruim, como vamos discutir na Seção 2.3, já que esse método funciona melhor para séries de termos positivos.

Exploraremos uma aplicação desses métodos na distribuição de Tweedie, pertencente à classe de modelos exponenciais de dispersão (EDM), cuja propriedade

$$\mathrm{Var}(X) = E[(X - E[X])^2] = \sigma(E[X])^p$$

é válida para qualquer p fora do intervalo $(0, 1)$. Conforme apresentado em (JØRGENSEN, 1987) e (JØRGENSEN, 1997), essas distribuições são conhecidas como a classe de modelos de (TWEEDIE, 1984). A classe dos modelos de Tweedie inclui importantes modelos lineares generalizados, como a distribuição normal ($p = 0$), Poisson ($p = 1$), gamma ($p = 2$) e Gaussiana inversa ($p = 3$). Segundo (JØRGENSEN, 1992), além dessas quatro

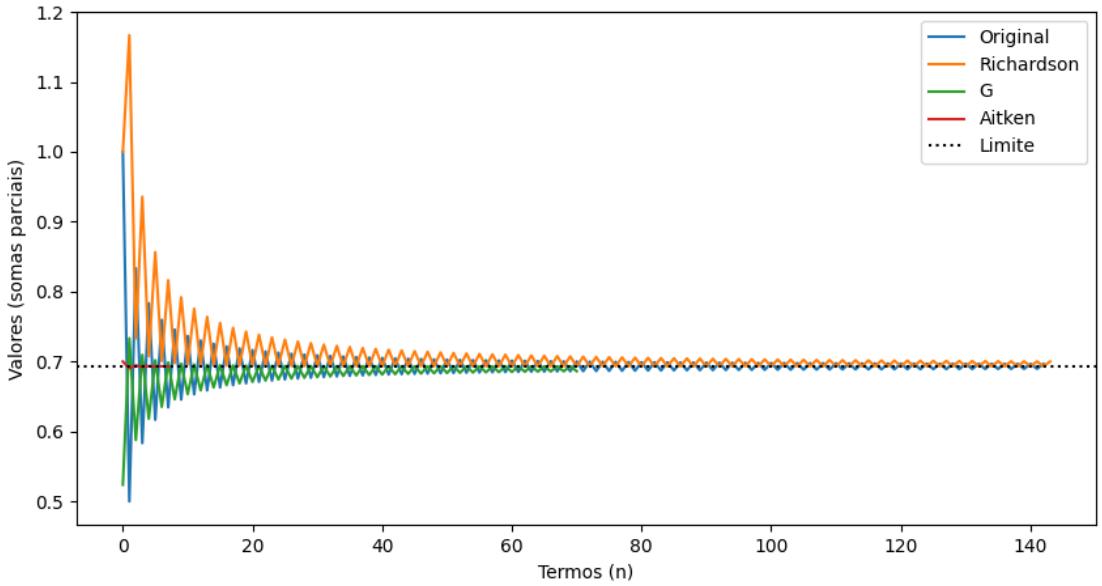

Figura 1 – Série harmônica alternada

distribuições conhecidas, nenhum dos modelos de Tweedie possui funções de densidade com forma analítica conhecida, mas sim na forma de somatórios.

Seguindo (DUNN, Peter K; SMYTH, Gordon K, 2005), os modelos de Tweedie para $p > 2$ são gerados por distribuições estáveis, com suporte nos reais positivos. As distribuições de Tweedie para $1 < p < 2$ podem ser representadas como misturas de Poisson de distribuições gamma e são distribuições mistas com suporte nos reais não negativos. (JØRGENSEN, 1987) demonstra que, para $p < 0$ a média é maior que zero, mas suporta valores em toda a reta real. Dada a limitada aplicabilidade desse último caso, não nos deteremos nele.

Além das aplicações das quatro distribuições mencionadas acima, a distribuição de Tweedie é utilizada em diversos campos, como ciências atuariais ((HABERMAN; RENSHAW, 1996), (JØRGENSEN; PAES DE SOUZA, 1994) e (SMYTH; JØRGENSEN, 2002)), 'assay analysis' ((DAVIDIAN, 1990)), analise de sobrevivência ((AALEN, 1992), (HOUGAARD, 1986) e (HOUGAARD; HARVALD; HOLM, 1992)), tempo gasto emendando cabos telefônicos ((NELDER, 1994)), custo com contratação de mão de obra externa ((JØRGENSEN, 1987)) e ecologia ((PERRY, 1981)).

O objetivo deste trabalho é fornecer uma biblioteca em Python com diferentes métodos de extrapolação de séries e apresentar uma metodologia para aplicação desses métodos. A estrutura deste trabalho é dividida em: Seção 2 descrevendo alguns dos métodos de extrapolação mais usados. Seção 3 detalha a implementação da biblioteca em Python e suas funções principais. Seção 4 apresenta o modelo de Tweedie e as abordagens para avaliação de sua função de densidade de probabilidade (f.d.p.). Seção 5 descreve a aplicação

dos métodos à Tweedie. Por fim, Seção 6 conclui com os principais resultados e implicações deste trabalho.

2 Métodos de extrapolação de séries

Como não temos um método de extrapolação que funcione sempre, é interessante observar o comportamento de diferentes métodos no problema em questão. Para escolha das transformações apresentadas aqui consideramos métodos que aparecem frequentemente em artigos relevantes e com bons resultados em famílias abrangentes de séries. Algumas dessas referencias são (SMITH; FORD, 1979), (SMITH; FORD, 1982), (SMALL, 2010), (BREZINSKI; ZAGLIA, 2003) e (JOYCE, 1971).

Os métodos escolhidos que serão apresentados nas seções seguintes são: O Algoritmo E; O método de Aitken's Δ^2; O método de Richardson; O método épsilon; O Algoritmo G e o método Levin generalizado. Vale mencionar também que existem métodos que derivam dos que estamos apresentando, sendo ou uma generalização, ou um caso específico, como nesta seção nos concentramos em apresentar os principais métodos vamos deixar casos particulares de lado, mas tenha em vista que existem outras variações que podem ser relevantes em algumas aplicações.

2.1 Algoritmo E

O algorítimo E é uma transformação de sequência que tem mais utilidade teórica do que prática, mas da suporte teórico aos demais métodos. O Algoritmo E desenvolvido por (BREZINSKI, C., 1980) é a transformação de sequência mais geral conhecida, pois contém quase todas as outras já descobertas, incluindo as que serão apresentadas aqui na sequência. Se baseia no fato de que podemos escrever a sequência (S_n) convergente com limite L da seguinte forma

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

onde as funções $(g_i(n))'s$ são dadas pela transformação escolhida, e L pode ser obtida resolvendo esse sistema linear, dependendo dos índices de k e n, onde n é o tamanho da amostra e k é a quantidade de interações do método. E podemos denotar a resolução desse sistema por $E_k^{(n)}$ (que será a aproximação do limite da série), onde

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

Mas está expressam não é muito prática, já que envolve calcular 2 determinantes de matrizes possivelmente grandes, o interessante é que podemos escolher essa função $(g_i(n))'s$, em particular, escolher de forma a obter outros métodos. Mas o que tiramos dessa seção é que resultados provados para essa transformação com $g_i(n)$ qualquer, valem para os demais métodos também. Como o seguinte teorema, que garante a convergência do método e por consequência a convergência de alguns dos métodos que veremos aqui:

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

Prova: Veja Apêndice A. ■

Algumas observações devem ser feitas antes de seguir. A primeira é que o teorema acima estabelece a convergência, não a convergência mais rápida. A segunda é que não basta que S_n convirja para um real L, uma condição a mais sobre g_i deve ser satisfeita no teorema. Não será abordado aqui, mas uma prova completa para a convergência dos métodos pode ser encontrada em (BREZINSKI; ZAGLIA, 2003).

2.2 O método de Aitken

O primeiro método prático que será introduzido aqui é o método de (AITKEN, 1927). A motivação para estudar este método se da principalmente pela sua simplicidade e eficiência em casos de convergência linear.

O método se baseia na ideia de taxa de variação da série inicial, em que dado uma série de interesse $S_n = \sum_{k=1}^{n} a_k$ que convirja e substituímos está série pela sua transformação $\{T_n\}_{n\in\mathbb{N}}$

$$T_n = \frac{S_{n-1}S_{n+1} - S_n^2}{S_{n+1} - 2S_n + S_{n-1}}. \tag{2.1}$$

Essa nova sequência de somas se aproveita da taxa de variação das somas da série anterior para obter uma nova que possa convergir mais rápido. Assim como a maioria

dos métodos que serão abordados aqui, o método de Aitken pode ser facilmente aplicado sucessivamente, podendo atingir resultados melhores. Porém, aplicações sucessivas geram mais tempo de processamento e perda de precisão numérica. Além disso, o ganho de aplicar sucessivamente é em geral diminui em relação a primeira aplicação.

2.3 O método de Richardson

A próxima transformação tem como alvo séries de termos positivos, o método de extrapolação de (RICHARDSON; GLAZEBROOK, 1911) (RICHARDSON, 1927). Mas vamos discutir uma versão um pouco diferente da original apresentada em (BREZINSKI, Claude, 2009) e (SMALL, 2010).

Para este método tomamos um valor inicial p que pode ser determinado observando a calda, mas em algumas literaturas se convenciona escolher p = 1, principalmente quando não se tem informações da série. Então, a partir da série convergente inicial $\{S_n\}_{n\geq 0}$, definimos a nova sequência como sendo: [1]

$$T_n = S_{2n} + \frac{S_{2n} - S_n}{2^p - 1}. \tag{2.2}$$

Alguns exemplos de uso deste método podem ser encontrados em (SMALL, 2010, Seção 8.7), este é um dos melhores métodos para séries de termos positivos, principalmente se pode-se avaliar p. Porém também tem problemas, como cada interação usa metade dos valores anteriores, para algumas séries com variância alta isso pode resultar em convergência lenta. A fim de exemplo, observe a Tabela 1, aplicando a transformação de Richardson com $p = 1$ para o problema de Basel, $\zeta(2) = \sum_{k=1}^{\infty} \frac{1}{k^2}$:

Tabela 1 – 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			

olhando para o valor de $T''_1 \approx 1.6444$, note que, o valor real dessa série é $\pi^2/6 \approx 1.6449$, então T''_1 está muito mais próximo do valor real do que a soma S_8 obtida inicialmente. No Apêndice B provamos que de fato a transformação de Richardson com $p = 1$ acelera o problema de Basel satisfazendo as 3 condições da Definição 3. Porém note que foi possível prosseguir com a prova por causa do Lema 2 que temos limites laterais para S_n, em outros casos isso pode não ser possível. O que torna a Definição 3 de difícil uso em problemas

[1] Note que, essa transformação usa até o termo $2n$ na série inicial para avaliar n termos na nova sequência, o que em termos de complexidade algorítmica faz desse método $O(\log(n))$.

práticos, porém, pode ser útil em problemas com limites conhecidos, vide (BRADEN, 1992) para alguns.

2.4 O método de Epsilon

Além dos métodos anteriores, um muito conhecido é o método Épsilon (ou método de Shanks) apresentado (SHANKS, 1955) e (WYNN, 1956). É provado a convergência em (GRAVES-MORRIS; ROBERTS; SALAM, 2000) que o apresenta como o melhor em geral 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

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

Os valores de k impares são apenas valores intermediários que não expressam um significado em si, mas os valores de k pares formam sequências convergentes para o mesmo limite da série original. A partir daqui é natural para cada transformação ter um indexador para aplicações sucessivas, em particular, $k = 2$ corresponde a uma única aplicação.

2.5 Algoritmo G

Introduzido por (GRAY; ATCHISON, 1967) e (ATCHISON; GRAY, 1968), inicialmente com intuito de avaliar integrais impróprias o Algoritmo G é um pouco mais complexo que os anteriores. Seguindo (PYE; ATCHISON, 1973) usaremos duas sequências auxiliares, como de costume seja $S_n = \sum_{k=1}^{n} a_k$, então definiremos duas sequências auxiliares da seguinte forma

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

e iterativamente

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

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

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

com $G_n^{(0)} = S_n$. Assim k indexa o número de aplicações sucessivas, em particular $k = 1$ descreve uma aplicação da transformação.

2.6 O método de Levin generalizado

O último método a ser considerado é o método de Levin, que se diferencia dos anteriores por ser considerado uma versão mais simples do Algoritmo E da Seção 2.1. Seguindo (LEVIN, 1972) e (SMITH; FORD, 1979) seja uma série convergente $S_n = \sum_{k=1}^{n} a_k$, a nova sequência extrapolada é definida como

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

onde

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

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

e

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

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

aqui k determina o número de vezes que o método é reaplicado. Note que, o método é constituído por uma função $g(.)$ livre que (LEVIN, 1972) sugere 3 opções para g

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

3 Biblioteca em Python

Neste capítulo, descreveremos o desenvolvimento da biblioteca https://pypi.org/project/extrapolation em Python[1]. Além disso, vamos descrever o que já existe, analizando os prós e contras. Este capítulo é dividido em 3 partes. Na Seção 3.1 falamos da biblioteca usada como base para a *extrapolation* e o que já existe em termos de soma nela. Na Seção 3.2 é descrito a implementação de operações básicas dos métodos acima em escala logarítmica guardando o sinal, assim podendo ser usado em séries de termos negativos. Por fim, na Seção 3.3 descrevemos as principais funções implementadas, para extrapolação de uma série real.

3.1 Mpmath

A *mpmath* ((THE MPMATH DEVELOPMENT TEAM, 2023)) é uma biblioteca de precisão aritmética para reais e complexos. Segundo (MEURER et al., 2017), originalmente a *mpmath* foi desenvolvida como um submódulo do *SymPy*, mas posteriormente movida para um pacote autônomo. Hoje, o *SymPy* recorre ao *mpmath* para representação de *floating-point*, assim como o *sagemath* que possui a *mpmath* por padrão.

Os números na *mpmath* são guardados em formato de string podendo assim ter precisão arbitraria, onde a precisão é controlada por variável global (por padrão `mpmath.prec` $= 53$ que representa os bits da mantissa de um float64). Porém isso tem um ponto negativo apesar de ganhar em precisão perde em tempo, por isso apesar de usar precisão arbitraria dada pelo usuário, a *extrapolation* entende que se for passado uma precisão de 53 em vez de usar a *mpmath* usa o float64 padrão do Python.

A partir da estrutura básica de controle de precisão, a *mpmath* fornece suporte a funções especiais como[2]: localização de raízes, álgebra linear, calculo numérico de limites, derivadas e integrais, séries infinitas e solução de EDOs. Dentre essas funções especiais a *mpmath* possui a função `nsum`[3] que tem como objetivo aproximar séries. Seja por exemplo, o exemplo que tomamos na Seção 2.3, do problema de Basel de avaliar $\zeta(2) = \lim_{n\to\infty} S_n$ tal que

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

[1] Um pequeno tutorial de instalação e uso da biblioteca pode ser encontrado em https://wellington36.github.io/extrapolation_tutorial

[2] Aqui temos 100 formas de aproximar π usando apenas *mpmath*: https://fredrikj.net/blog/2011/03/100-mpmath-one-liners-for-pi.

[3] Para mais detalhes vide a documentação: https://mpmath.org/doc/current/calculus/sums_limits.html

Podemos obter uma aproximação satisfatória para $\zeta(2)$ com 53 bits de precisão usando a `nsum`, passando como argumento uma função lambda que descreve os termos da série, o intervalo de soma nesse caso $[1, \infty]$ e um método dentre os descritos na documentação (temos alguns métodos de extrapolação implementados aqui, no caso, Richardson, Epsilon[4] e Levin). E de fato, escolhendo por exemplo o método de Richardson resulta em uma aproximação de 1.64493406684823 que é exatamente $\frac{\pi^2}{6}$ com 15 casas decimais exatas. Porém, esse bom resultado não ocorre sempre. Seja um outro exemplo, bem comum em provas de cursos de análise real, a série com somas parciais

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

Esta série, em comparação com o problema de Basel, converge extremamente devagar. Usando o mesmo processo do exemplo anterior, obtemos 1.91056295070081, porém essa série é famosa, uma aproximação com 15 casas decimais pode ser obtida via fórmula de Euler–Maclaurin sendo 2.10974280123689. Note que não passamos em nenhum momento uma tolerância ou erro esperado à função. Segundo a documentação, o critério de parada verifica se a última soma menos uma estimativa para o valor[5] é menor que uma tolerância, que é dada a partir da precisão. Ou seja, melhorar o resultado requer aumentar a precisão, o que também significa mais uso de processamento e memória. Para problemas que requerem uma aproximação muito boa ou com muitos termos, isso se torna inviável computacionalmente.

Para a biblioteca *extrapolation* além de ter mais métodos e funções auxiliares, algumas mudanças foram consideradas para obter resultados melhores em relação a função *nsum* da *mpmath*. Para tentar mitigar o problema acima uma das soluções foi a implementação de escala logarítmica, que não era usada na biblioteca e será melhor discutida na Seção 3.2 a seguir. Além disso, fizemos uma mudança no critério de parada, parando quando a diferença das duas últimas somas é menor que uma tolerância dada.

Outra melhoria crucial consiste na utilização eficiente do float64 padrão do Python quando a precisão é definida como 53. Adicionalmente, a *extrapolation* agora retorna não apenas o resultado final, mas também a sequência extrapolada, permitindo a aplicação de métodos adicionais, conforme será detalhado na Seção 3.3, e possibilitando a análise de resultados intermediários.

[4] Note que, na biblioteca o Epsilon aparece como Shanks, como mencionado na Seção 2.4 este método aparece com ambos os nomes

[5] Não é mencionado nem na documentação nem no código o que seria essa estimativa e como ela é calculada.

3.2 Escala logarítmica

Operar com números muito pequenos ou muito grandes pode trazer erros numéricos (*underflow* e *overflow*), uma forma famosa de reduzir esses erros é usar a escala logarítmica (HILDEBRAND, 1987). A ideia é aplicar uma função logarítmica em todos os valores, o que faz valores no intervalo $(0, 1)$ irem para o intervalo $(-\infty, 0)$, dando mais liberdade para valores pequenos. O problema é que o logaritmo não está definido para valores negativos, o que impede o uso da biblioteca para séries com algum termo negativo.

A solução para isso implementada na biblioteca *extrapolation* é implementar uma classe `LogNum` em Python que guarde o sinal e o valor absoluto do número em escala logarítmica. Sendo pela biblioteca *mpmath* no caso de precisão diferente de 53 ou pela *math* caso contrário. Essa classe também possui a implementação das operações básicas, respeitando o logaritmo. Ou seja, no lugar de exponenciação é feito produto, de produto é feito soma, de soma é aplicado a logSumExp, definida como

$$\mathrm{LSE}(x_1, x_2, \ldots, x_n) = x_{\max} + \log \sum_{i=1}^{n} \exp\left(x_i - x_{\max}\right),$$

onde $x_{\max} = \max x_1, x_2, \ldots, x_n$. Além de verificações de desigualdades e igualdades, implementadas para condizer com o novo tipo, todos aplicando as operações também nos sinais.

As funções principais da biblioteca convertem qualquer número para um objeto dessa nova classe, calculam as somas parciais, aplicam o método de extrapolação desejado e por fim retornam o número para a sua forma original. Este processo se mostrou bem eficiente como mostraremos a seguir, na Seção 3.3, onde apresentaremos as funções de extrapolação implementadas.

3.3 Funções principais

Como mencionado anteriormente a biblioteca consiste em diversas funções que podem ser úteis, mas em especial para o problema de obter melhores aproximações para uma série infinita temos 2 funções principais. A primeira consiste em dado uma série e um natural n, avaliar os n primeiros termos e aplicar uma das transformações da Seção 2, retornando uma nova sequência acelerada a partir dos n termos chamaremos ela de `acelsum`.

Já a segunda é um pouco mais prática, em vez de receber o número de termos, recebe um erro e seu objetivo é encontrar o menor natural n tal que a diferença absoluta dos dois últimos valores seja menor que o erro, isso em termos do método de extrapolação escolhido. Está segunda chamaremos de `esum` tem como ambição ser uma versão com mais opções e mais robusta em teoria à função `nsum` da *mpmath*. A seguir apresentaremos com mais detalhes ambas as funções, para na Seção 4 apresentarmos uma aplicação prática.

3.3.1 acelsum

A função `acelsum` recebe como parâmetros:

- *Uma série*: Na forma de uma função $f : \mathbb{N} \to \mathbb{R}$ que retorna os termos da série;
- *O método*: Podendo ser "Aitken", "Richardson", "Epsilon", "G", "Levin-t", "Levin-u", "Levin-v", mencionado na Seção 2 ou "None"para soma usual;
- *Um natural n*: Sendo o número de termos a serem considerados;
- *Um boleano*: Os cálculos são feitos em escala logarítmica, como descrito na Seção 3.2, 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*.

E retorna uma sequência resultante do método de extrapolação, onde, desde que o método acelere a série, o último termo se aproxima do limite mais do que o último da sequencia de somas parciais. Está função permite a partir de um critério de parada próprio obter uma aproximação para uma determinada série usando extrapolação. As diversas possibilidades de critérios de parada não serão abordados aqui, mas estão em estudo (CARVALHO; MOREIRA, 2022). No contexto do presente trabalho, vamos usar um critério de parada simples tomar a diferença em valor absoluto das últimas 2 extrapolações e verificar se é menor que um erro dado.

3.3.2 esum

A ideia desta função é encontrar o menor natural n tal que a extrapolação dos primeiros n termos atinge o critério de parada. Note que, não se trata de uma busca linear, e sim logarítmica, para encontrar este valor n percorremos as potencias de 2 até atingir o critério de parada, obtendo um intervalo em que o n está basta então particionar o intervalo no meio repetidamente até obter o valor. Esta ideia garante encontrar o menor natural que atinge o critério de parada com complexidade $\mathcal{O}(2\log(n))$. Está função recebe quase os mesmos parâmetros da `acelsum` com uma diferença, o erro:

- *Uma série*: Na forma de uma função $f : \mathbb{N} \to \mathbb{R}$ que retorna os termos da série;
- *O método*: Podendo ser "Aitken", "Richardson", "Epsilon", "G", "Levin-t", "Levin-u", "Levin-v", mencionado na Seção 2 ou "None"para soma usual;
- *O erro (ε)*: Um valor para o critério de parada $|T_{n-1} - T_n| < \varepsilon$;

- *Um boleano*: Os cálculos são feitos em escala logarítmica, como descrito na Seção 3.2, 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*.

Quando a função encontra o primeiro n que satisfaz $|T_{n-1} - T_n| < \varepsilon$ é retornado o valor de n e a sequência $\{T_k\}_{k\in\mathbb{N}}$, onde o último valor atinge o critério de parada e é portando uma aproximação para a série original. Retornando ao exemplo mencionado na Seção 3.1

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

O melhor resultado, com precisão fixa de 53 da `nsum`, é 1.9105 com o método de Richardson e tempo médio de 0.004 segundos. Já o melhor resultado da `esum` é obtido com a variante v do método de Levin, alcançando 2.02578, com tempo médio de 0.1939 segundos e erro de 10^{-4}. Esse erro pode ser diminuído para melhorar os resultados, ao custo de aumentar o tempo. Apesar do resultado ser superior, foi consideravelmente mais lento. No entanto, isso muda se aumentarmos a precisão, a única forma de aprimorar os resultados na função `nsum`.

Com precisão de 100 bits, o melhor resultado da `nsum` é 2.0110 com o método Epsilon e tempo médio 0.2795 segundo, enquanto o da `esum` se mantém como 2.0257 com o com a variante v do método de Levin e tempo médio de 0.2340 com mesmo erro, assim ganhando tando no tempo quanto no resultado. O aumento da precisão não muda o resultado da `esum` em nenhum dos métodos, o que mostra certa consistência com o erros numéricos, o que não ocorre na `nsum`. Os valores encontrados durante este teste podem ser encontrados no Apêndice C.

Para comparação e teste, até aqui trabalhamos com problemas de séries cujo limite é conhecido. Com objetivo de apresentar aplicações mais próximas da realidade, na seção seguinte introduziremos a cerca da distribuição de Tweedie.

4 Tweedie

Os modelos exponenciais de dispersão (EDM) têm uma função de densidade de probabilidade da forma (JØRGENSEN, 1997)

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

onde $\kappa()$ e $a()$ são funções conhecidas.

O parâmetro canônico θ pertence a um intervalo aberto que satisfaz $\kappa(\theta) < \infty$ e o parâmetro de dispersão ϕ é positivo. A função $\kappa()$ é chamada de função cumulante do EDM porque, quando $\phi = 1$, as derivadas de $\kappa()$ fornecem os cumulantes sucessivos da distribuição. Como mencionado na Seção 1, a distribuição de Tweedie faz parte dessa classe de modelos e é definida pela propriedade

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

para qualquer p fora do intervalo $(0,1)$. Para alguns valores específicos de p obtemos distribuições famosas como a distribuição Normal ($p = 0$), Poisson ($p = 1$), Gamma ($p = 2$) e Gaussiana Inversa ($p = 3$). No entanto, como mencionado em (JØRGENSEN, 1992), para outros valores de p não temos uma fórmula fechada para a função de densidade probabilidade.

Conforme (JØRGENSEN, 1987), os modelos de Tweedie para $p < 0$, apesar de terem suporte em toda a reta real, têm aplicações limitadas, que não será explorado aqui. O problema em avaliar a f.d.p. da Tweedie está na função $a()$, que não pode ser expressa de forma fechada, (JØRGENSEN, 1997) e (SMYTH, 1996) avaliam a função $a()$ como uma série infinita, dividindo em dois casos. Tomando $\alpha = \frac{2-p}{1-p}$ para simplificar a notação, para $1 < p < 2$ e $z > 0$, temos

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

com $W(z,\phi,p) = \sum_{j=1}^{\infty} W_j$ e

$$W_j = \frac{z^{j\alpha}(p-1)^{\alpha j}}{\phi^{j(1-\alpha)}(2-p)^j j!\Gamma(-j\alpha)}.$$

Por outro lado, para $p > 2$ e $z > 0$,

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

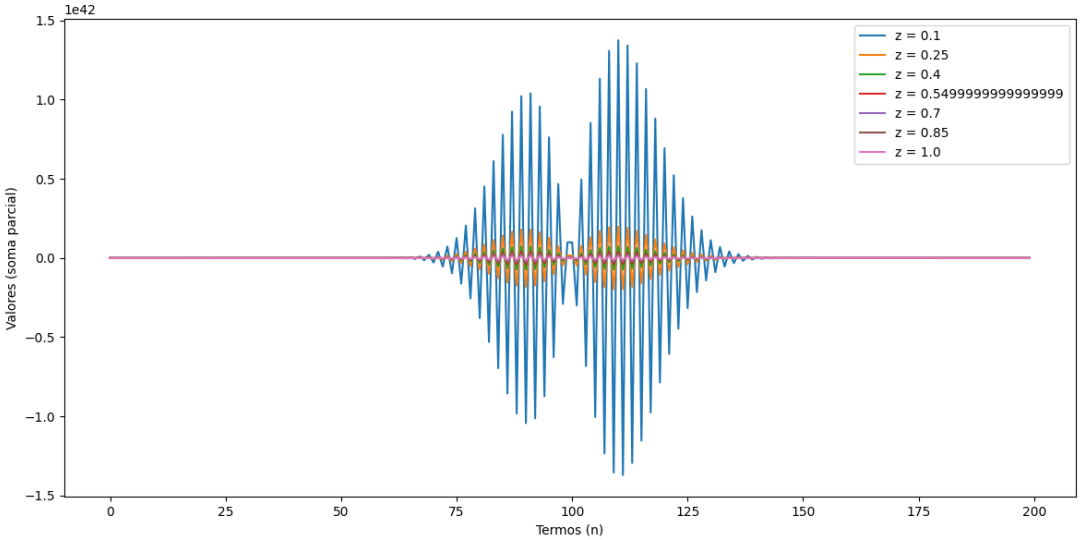

Figura 2 – $\alpha = 0.01,\ \theta = -1/2,\ z \in (0.1, 1),\ n = 200$

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

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

Cada um desses casos tem certas peculiaridades. O primeiro é positivo para todo W_j e possui um fator fatorial em seu denominador, o que torna esse caso não tão interessante se o objetivo é buscar problemas mais desafiadores que os exemplos anteriores. Já o segundo caso possui dois termos que alteram o sinal constantemente, $(-1)^k$ e $\sin(-k\pi\alpha)$, introduzindo um comportamento que ainda não observamos nos casos em que os métodos de extrapolação foram testados.

Avaliar a f.d.p. da Tweedie no intervalo $p > 2$ tem outra peculiaridade que o torna interessante para testar os métodos da Seção 2, a forma que as sequências de somas parciais tomam. Sendo, por exemplo, $\theta = -1/2$, $\alpha = 0.01$[1] e $z \in (0.1, 1)$ temos o comportamento das somas parciais representadas na Figura 2.

As somas parciais atingem valores muito altos como pode ser observado na escala da esquerda da figura, apesar disso o valor final não, isso ocorre pelo cancelamento de termos que afeta principalmente a precisão. Voltando ao nosso problema de extrapolar séries, outro ponto pode ser identificado olhando para as somas parciais quando $\theta = -1/2$, $\alpha = 0.99$ e $z \in (1, 1.1)$, vide a Figura 3.

Este caso é bem diferente do anterior principalmente em termos de extrapolação, a Figura 3 mostra somas parciais crescentes, enquanto a Figura 2 somas parciais oscilantes. Isto pode indicar que um método que funcione para alguns parâmetros da Tweedie, não vai

[1] $\alpha = \frac{2-p}{1-p}$, assim $p > 2 \Rightarrow 0 < \alpha < 1$.

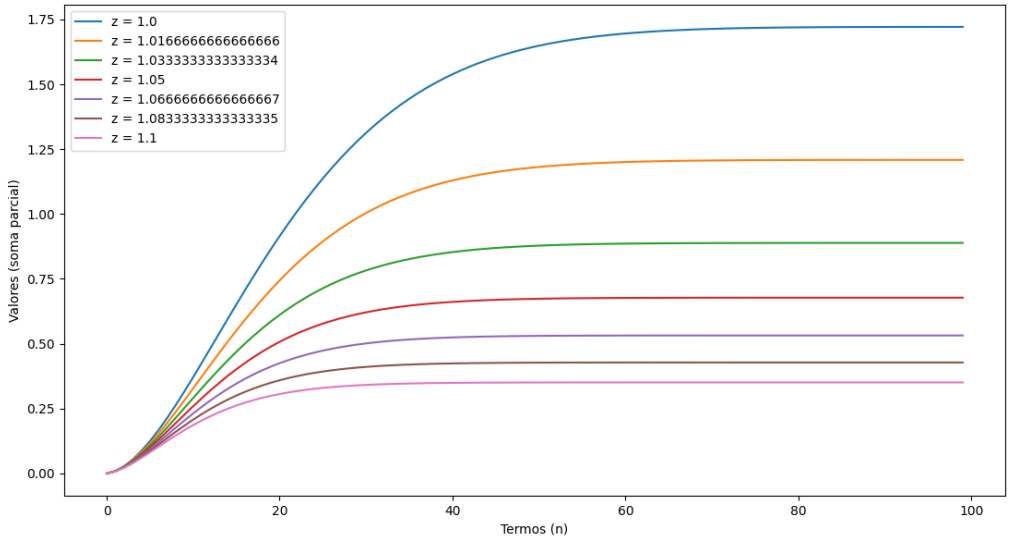

Figura 3 – $\alpha = 0.99,\ \theta = -1/2,\ z \in (1, 1.1),\ n = 100$

funcionar da mesma forma para outros. Dado que os métodos fazem uso de comportamentos prévios das séries para atingir valores melhores.

Para alguns outros gráficos, ver Apêndice D. Na seção a seguir vamos tratar de duas implementações para a Tweedie já existentes em Python. Da biblioteca *tweedie*[2] com foco no intervalo $1 < p < 2$ e a outra implementação apresentada em (DIAS; RIBEIRO JR, 2019) com intuito de obter melhores aproximações em comparação ao que está implementado no R por (DUNN, Peter K.; SMYTH, Gordon K., 2008). Daí na Seção 5 discutiremos a aplicação dos métodos de extrapolação ao problema de avaliar a f.d.p. da Tweedie e apresentaremos seus resultados.

4.1 Tweedie no Python

Como mencionado anteriormente, a p.d.f. da Tweedie para $p > 2$ tem problemas de cancelamento catastrófico e perda de precisão. (DIAS; RIBEIRO JR, 2019) apresenta algumas formas de minimizar esses e outros problemas ao avaliar essa série, uma das formas mais eficientes apresentadas é usando a biblioteca *mpmath* com precisão de 1000 bits, o que garante no artigo obter resultados melhores que as implementações da Tweedie no R.[3] A implementação da Tweedie por (DIAS; RIBEIRO JR, 2019) olha para o problema de avaliar $a(z, \phi)$ a série

$$N(z;\alpha) = \frac{1}{\pi z} \times \sum_{k=1}^{\infty} \frac{\Gamma(1+\alpha k)}{\Gamma(1+k)} (\kappa(-1/z, \alpha))^k \sin(-k\pi\alpha),$$

[2] https://pypi.org/project/tweedie

[3] A implementação feita por (DIAS; RIBEIRO JR, 2019) pode ser encontrada no Apêndice do mesmo.

onde

$$\kappa(\theta, \alpha) = \frac{\alpha - 1}{\alpha} \left(\frac{\theta}{\alpha - 1} \right)^{\alpha}.$$

A fim de manter a precisão ao computar a série, uma estratégia é quebrar em séries parciais da seguinte forma. Seja

$$B = |\kappa(-1/z, \alpha)|;$$

escrevermos

$$\begin{aligned} b_k &= \frac{\Gamma(1 + \alpha k)}{\Gamma(1 + k)} B^k, \\ c_k &= (-1)^k b_k, \\ d_k &= c_k \sin(-k\pi\alpha), \end{aligned}$$

e as respectivas séries

$$\begin{aligned} S_b(k) &= \sum_{j=1}^{k} b_j, \\ S_c(k) &= \sum_{j=1}^{k} c_j, \\ S_d(k) &= \sum_{j=1}^{k} d_j. \end{aligned}$$

Dessa forma, $N(z; \alpha) = S_d(\infty)/(\pi z)$. Muitas sutilezas, como critérios de parada e divisão dos termos pelo máximo, apresentadas por (DIAS; RIBEIRO JR, 2019) vêm de (DUNN, Peter K; SMYTH, Gordon K, 2005), (DUNN, Peter K.; SMYTH, Gordon K., 2008) e as implementações para Tweedie no R. Não vamos entrar nos detalhes da implementação, mas antes de fazer comparações com e sem os métodos de extrapolação vamos comparar a implementação em (DIAS; RIBEIRO JR, 2019) com o que já existia no Python, ou seja, a biblioteca *tweedie* do Python.

Para a metodologia de comparação de duas implementações vamos usar que a série gera uma f.d.p. – ou seja que integra 1 na reta. Este método é usado em (DIAS; RIBEIRO JR, 2019) para comparar a implementação feita com a existente no R. A ideia é avaliar a quadratura da f.d.p. num intervalo que cubra a parte significante da f.d.p., com um número de pontos fixo em ambas as implementações. Seguindo o que foi feito em (DIAS;

RIBEIRO JR, 2019) verificaremos para $\alpha \in [0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$ no intervalo $[1e{-}6, 50]$ com 1000 pontos na quadratura e para $\alpha \in [0.99$ no intervalo $[1e{-}6, 20]$ com 10000 pontos na quadratura.

		Dias		tweedie	
α	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

Tabela 2 – Dias vs. Tweedie no Python

A Tabela 2 mostra a comparação da implementação do (DIAS; RIBEIRO JR, 2019) com a biblioteca *tweedie* do Python para $p > 2$. Note que, em quase todos os casos a biblioteca se sai pior em relação a implementação do (DIAS; RIBEIRO JR, 2019), com exceção de $\alpha = 0.5$, o que ocorre nesse caso é que a biblioteca do Python não está avaliando uma série pois $\alpha = 0.5$ corresponde a $p = 3$, que é a Gama Inversa e portanto a biblioteca *tweedie* possui a fórmula fechada nesse caso.

Na seção seguinte usaremos a mesma estratégia de teste para comparar a implementação em (DIAS; RIBEIRO JR, 2019) com uma versão com cada um dos métodos disponíveis na biblioteca *extrapolation*. E por fim, na Seção 6 colocaremos as considerações finais e ideias futuras.

5 Aplicação dos métodos

Todos os testes apresentados aqui foram feitos em uma máquina com processador Intel Core i7-11370H (decima primeira geração) @3.30GHz, com 16 GB de RAM e NVIDIA GeForce GTX 1650 rodando Linux OS (Manjaro 23.1.0). Os testes de tempo foram feitos 3 vezes com mesmas configurações de máquina, guardando a média dos resultados.

Para verificar os métodos algumas mudanças na estrutura do código apresentado em (DIAS; RIBEIRO JR, 2019) foram necessárias. Portanto foi feito uma testagem comparando a performance de (DIAS; RIBEIRO JR, 2019) com a nova versão antes de aplicar qualquer método. Assim, tendo uma diferenciação entre as mudanças que os métodos de extrapolação trouxeram e o que veio da mudança na estrutura. O resultado está apresentado na Tabela 3.

α	erro		tempo (100/seg)		score
	Dias	Dias com 'None'	Dias	Dias com 'None'	
0.01	7.0451e-07	7.0451e-07	15.9783	18.0906	0.0
0.1	2.0245e-08	2.0245e-08	1.7151	2.0884	0.0
0.2	1.2977e-11	1.2977e-11	1.0333	1.2504	0.0
0.3	7.7060e-13	7.7049e-13	0.7744	0.9851	0.0001
0.4	6.0196e-13	6.0196e-13	0.7819	0.9243	0.0
0.5	9.1116e-13	9.1116e-13	0.7893	0.8937	0.0
0.6	3.2041e-13	3.2041e-13	1.0440	1.2109	0.0
0.7	9.9475e-14	9.9698e-14	1.1957	1.3951	-0.0022
0.8	9.6034e-13	9.6034e-13	1.3598	1.5240	0.0
0.9	1.5676e-11	1.5676e-11	1.1383	1.2902	0.0
0.99	3.3810e-09	3.3810e-09	1.1989	1.4394	3.2836e-08

Tabela 3 – Dias vs. Dias com estrutura para extrapolação

Está Tabela 3 apresenta na coluna 1 os valores de α verificados, nas colunas 2 e 3 os erros da quadratura da f.d.p. em relação a 1 na versão original 'Dias' e na versão modificada mas sem métodos 'Dias com 'None' (usando soma usual), nas colunas 4 e 5 os tempos em segundos de avaliação de 100 pontos da quadratura e na coluna 6 o 'score' que é dado pela fórmula

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

ou seja, se o score é negativo significa que o erro de (DIAS; RIBEIRO JR, 2019) foi menor case seja positivo o erro da versão modificada foi menor e o tamanho desse score está relacionado a diferença dos erros. Uma consideração para a aplicação dos métodos é que as mudanças feitas há estrutura em média reduzem o tempo em 20%.

Com essa nova estrutura geramos tabelas análogas a Tabela 3 e podem ser encontradas no Apêndice C[1]. Um problema encontrado ao aplicar os métodos vem de que em alguns métodos há divisões em que pode ocorrer divisões por zero, isso não era um problema nos exemplos anteriores, mas para a Tweedie é um problema constante dado que uma infinidade de termos zera por causa do fator $\sin(-k\pi\alpha)$. Não foi encontrado uma metodologia para lidar com os casos de erros nos métodos, mas o que é feito em alguns artigos de aplicações é ignorar o termo problemático da extrapolação, que foi como foi procedido aqui. A Tabela 4 apresenta o método de Levin com variante t, que obteve melhores resultados dentre os métodos aplicados.

α	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

Tabela 4 – Dias vs. Dias com Levin-t

[1] Os códigos usados para gerais as tabelas podem ser encontrados em https://github.com/wellington36/tweedie-numerical-approximation.

6 Conclusões

Neste trabalho apresentamos alguns dos métodos de extrapolação mais relevantes em diferentes aplicações criando uma biblioteca para aplicação desses métodos em problemas de somas infinitas, com precisão arbitraria e tomando precauções contra erros numéricos. Além de aplicar os métodos ao problema de avaliar a f.d.p. da distribuição de Tweedie.

O objetivo de apresentar a Tweedie como aplicação aos métodos de extrapolação vem pelas diferentes formas que a f.d.p. apresenta, mostrado nos gráficos da Seção 4 e no Apêndice D. Essas diferentes formas que as somas parciais possuem trazem dificuldades aos métodos de atingirem resultados melhores que o (DIAS; RIBEIRO JR, 2019) para todos os parâmetros possíveis. Além de que os métodos tendem a ser mais custosos em tempo, desconsiderando o tempo a mais pela estrutura na aplicação do método Levin com variante t, o método é entre 50% a 70% mais lento que o original.

Apesar dos resultados ruins para a Tweedie no geral, encontramos boas aproximações para parâmetros específicos, o que sugere que no caso da Tweedie, uma estratégia de particionar o intervalo dos parâmetros buscando em vez de um método que funciona sempre, um método que funcione bem num intervalo em específico. Outra estratégia, para casos como da Tweedie é buscar métodos mais gerais do que os que apresentamos aqui, dado sua estrutura mais complexa, o que claro pode trazer mais peso em tempo de execução.

Na literatura os métodos de extrapolação aparecem em geral em artigos para uma série especifica, mas em diferentes áreas. A proposta aqui é generalizar aplicações dos métodos para diferentes problemas, (BREZINSKI; ZAGLIA, 2003) lista algumas dessas aplicações, varias ainda não estudadas.

Como futuras atualizações estão a criação de formas de tratamento para erros dos métodos dentro da própria biblioteca, adicionando por exemplo, um parâmetro as funções principais sobre o que fazer em certos casos. Retornar a sequência permite reaplicar os métodos, uma ideia mais rápida é avaliar apenas termo da extrapolação (no caso o último termo), o que pode ser feito também passando um novo parâmetro às funções principais. Com as atualizações anteriores as funções principais da biblioteca pode ter muitos parâmetros de entrada, então outra atualização é quebrar as funções principais refatorando a estrutura da biblioteca para facilitar mais o uso.

Como mencionado há diversas outras aplicações possíveis aos métodos de extrapolação além dos exemplos matemáticos e da f.d.p. da Tweedie mostrados aqui. Um possível caminho de pesquisa à frente é explorar outras classes de problemas que envolvam somas infinitas com a metodologia apresentada aqui.

Referências

AALEN, Odd O. Modelling Heterogeneity in Survival Analysis by the Compound Poisson Distribution. **The Annals of Applied Probability**, Institute of Mathematical Statistics, v. 2, n. 4, p. 951–972, 1992. DOI: 10.1214/aoap/1177005583. Disponível em: <https://doi.org/10.1214/aoap/1177005583>.

AITKEN, A. C. On Bernoulli's Numerical Solution of Algebraic Equations. **Proceedings of the Royal Society of Edinburgh**, Royal Society of Edinburgh Scotland Foundation, v. 46, p. 289–305, 1927. DOI: 10.1017/S0370164600022070.

ATCHISON, T. A.; GRAY, H. L. Nonlinear Transformations Related to the Evaluation of Improper Integrals. II. **SIAM Journal on Numerical Analysis**, v. 5, n. 2, p. 451–459, 1968. DOI: 10.1137/0705035.

BRADEN, Bart. Calculating Sums of Infinite Series. **The American Mathematical Monthly**, Taylor & Francis, v. 99, n. 7, p. 649–655, 1992. DOI: 10.1080/00029890.1992.11995905. Disponível em: <https://doi.org/10.1080/00029890.1992.11995905>.

BREZINSKI, C. A general extrapolation algorithm. **Numerische Mathematik**, v. 35, n. 2, p. 175–187, 1980. DOI: 10.1007/bf01396314.

BREZINSKI, C.; ZAGLIA, M. Redivo. **Extrapolation Methods: Theory and Practice**. [S.l.]: North-Holland, 2003. (Studies in Computational Mathematics 2). ISBN 0444888144; 9780444888143.

BREZINSKI, Claude. Some pioneers of extrapolation methods. **The Birth of Numerical Analysis**, p. 1–22, 2009. DOI: 10.1142/9789812836267_0001.

CARVALHO, Luiz Max; MOREIRA, Guido A. Adaptive truncation of infinite sums: applications to Statistics. **arXiv preprint arXiv:2202.06121**, 2022.

DAVIDIAN, M. Estimation of Variance Functions in Assays with Possibly Unequal Replication and Nonnormal Data. **Biometrika**, [Oxford University Press, Biometrika Trust], v. 77, n. 1, p. 43–54, 1990. ISSN 00063444. Acesso em: 21 nov. 2023.

DELAHAYE, J. P.; GERMAIN-BONNE, B. Résultats négatifs en accélération de la convergence. **Numerische Mathematik**, v. 35, n. 4, p. 443–457, 1980. DOI: 10.1007/bf01399010.

DIAS, Nelson L.; RIBEIRO JR, Paulo J. Practical rules for summing the series of the Tweedie probability density function with high-precision arithmetic. **Anais da Academia Brasileira de Ciências**, Academia Brasileira de Ciências, v. 91, n. 4, e20180268, 2019. ISSN 0001-3765. DOI: 10.1590/0001-3765201920180268. Disponível em: <https://doi.org/10.1590/0001-3765201920180268>.

DUNN, Peter K; SMYTH, Gordon K. Series evaluation of Tweedie exponential dispersion model densities. **Statistics and Computing**, Springer, v. 15, n. 4, p. 267–280, 2005.

_____. Evaluation of Tweedie Exponential Dispersion Model Densities by Fourier Inversion. **Statistics and Computing**, Kluwer Academic Publishers, USA, v. 18, n. 1, p. 73–86, mar. 2008. ISSN 0960-3174. DOI: 10.1007/s11222-007-9039-6. Disponível em: <https://doi.org/10.1007/s11222-007-9039-6>.

GRAVES-MORRIS, P.R.; ROBERTS, D.E.; SALAM, A. The epsilon algorithm and related topics. **Journal of Computational and Applied Mathematics**, v. 122, n. 1, p. 51–80, 2000. Numerical Analysis in the 20th Century Vol. II: Interpolation and Extrapolation. ISSN 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(00)00355-1. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377042700003551>.

GRAY, H. L.; ATCHISON, T. A. Nonlinear Transformations Related to the Evaluation of Improper Integrals. I. **SIAM Journal on Numerical Analysis**, v. 4, n. 3, p. 363–371, 1967. DOI: 10.1137/0704032.

HABERMAN, Steven; RENSHAW, Arthur. Generalized Linear Models and Actuarial Science. **The Statistician**, v. 45, p. 407, jan. 1996. DOI: 10.2307/2988543.

HILDEBRAND, F. B. **Introduction to numerical analysis**. 2nd ed. [S.l.]: Dover Publications, 1987. (Dover Books on Advanced Mathematics). ISBN 9780486653631; 0486653633.

HOUGAARD, Philip. Survival Models for Heterogeneous Populations Derived from Stable Distributions. **Biometrika**, [Oxford University Press, Biometrika Trust], v. 73, n. 2, p. 387–396, 1986. ISSN 00063444. Acesso em: 21 nov. 2023.

HOUGAARD, Philip; HARVALD, Bent; HOLM, Niels V. Measuring the Similarities Between the Lifetimes of Adult Danish Twins Born Between 1881-1930. **Journal of the American Statistical Association**, [American Statistical Association, Taylor & Francis, Ltd.], v. 87, n. 417, p. 17–24, 1992. ISSN 01621459. Acesso em: 21 nov. 2023.

JØRGENSEN, Bent. Exponential Dispersion Models. **Journal of the Royal Statistical Society. Series B (Methodological)**, [Royal Statistical Society, Wiley], v. 49, n. 2, p. 127–162, 1987. ISSN 00359246. Disponível em: <http://www.jstor.org/stable/2345415>. Acesso em: 4 jul. 2023.

_____. **The Theory of Dispersion Models**. [S.l.]: Taylor & Francis, 1997. (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). ISBN 9780412997112. Disponível em: <https://books.google.com.br/books?id=0gO7bgs_eSYC>.

_____. The Theory of Exponential Dispersion Models and Analysis of Deviance. In. Disponível em: <https://api.semanticscholar.org/CorpusID:221993554>.

JØRGENSEN, Bent; PAES DE SOUZA, Marta C. Fitting Tweedie's compound poisson model to insurance claims data. **Scandinavian Acturial Journal**, p. 69–93, jan. 1994. DOI: 10.1080/03461238.1994.10413930.

JOYCE, D. C. Survey of Extrapolation Process in Numerical Analysis. **SIAM Review**, Society for Industrial e Applied Mathematics, v. 13, n. 4, p. 435–490, 1971. ISSN 00361445. Disponível em: <http://www.jstor.org/stable/2029188>. Acesso em: 17 jan. 2023.

LAVRIK, A. F. The principal term of the divisor problem and the power series of the Riemann zeta-function in a neighborhood of a pole. **Proc. Steklov Inst. Math.**, v. 142, p. 165–173, 1976.

LEVIN, David. Development of non-linear transformations for improving convergence of sequences. **International Journal of Computer Mathematics**, Taylor & Francis, v. 3, n. 1-4, p. 371–388, 1972. DOI: 10.1080/00207167308803075. Disponível em: <https://doi.org/10.1080/00207167308803075>.

MEURER, Aaron et al. SymPy: Symbolic computing in Python. **PeerJ Computer Science**, v. 3, e103, jan. 2017. DOI: 10.7717/peerj-cs.103.

THE MPMATH DEVELOPMENT TEAM. **mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.3.0)**. [S.l.], 2023. http://mpmath.org/.

NELDER, J. A. An Alternative View of the Splicing Data. **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, [Wiley, Royal Statistical Society], v. 43, n. 3, p. 469–476, 1994. ISSN 00359254, 14679876. Acesso em: 21 nov. 2023.

PERRY, J. N. Taylor's Power Law for Dependence of Variance on Mean in Animal Populations. **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, [Wiley, Royal Statistical Society], v. 30, n. 3, p. 254–263, 1981. ISSN 00359254, 14679876. Acesso em: 21 nov. 2023.

PYE, W. C.; ATCHISON, T. A. An Algorithm for the Computation of the Higher Order G-Transformation. **SIAM Journal on Numerical Analysis**, v. 10, n. 1, p. 1–7, 1973. DOI: 10.1137/0710001.

RICHARDSON, Lewis F. Viii. The deferred approach to the limit. **Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character**, v. 226, n. 636-646, p. 299–361, 1927. DOI: 10.1098/rsta.1927.0008.

RICHARDSON, Lewis Fry; GLAZEBROOK, Richard Tetley. IX. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. **Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a**

Mathematical or Physical Character, v. 210, n. 459-470, p. 307–357, 1911. DOI: 10.1098/rsta.1911.0009.

SHANKS, Daniel. Non-linear transformations of divergent and slowly convergent sequences. **Journal of Mathematics and Physics**, v. 34, n. 1-4, p. 1–42, 1955. DOI: 10.1002/sapm19553411.

SMALL, Christopher G. **Expansions and asymptotics for statistics**. [S.l.]: CRC, 2010. (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). ISBN 9781584885900; 1584885904.

SMITH, David A.; FORD, William F. Acceleration of Linear and Logarithmic Convergence. **SIAM Journal on Numerical Analysis**, v. 16, n. 2, p. 223–240, 1979. DOI: 10.1137/0716017. Disponível em: <https://doi.org/10.1137/0716017>.

_____. Numerical Comparisons of Nonlinear Convergence Accelerators. **Mathematics of Computation**, American Mathematical Society, v. 38, n. 158, p. 481–499, 1982. ISSN 00255718, 10886842. Disponível em: <http://www.jstor.org/stable/2007284>. Acesso em: 17 jan. 2023.

SMYTH, Gordon K. Regression Analysis of Quantity Data with Exact Zeroes. **Proceedings of the Second Australia-Japan Workshop on Stochastic Models in Engineering**, p. 572–580, 1996. Disponível em: <https://api.semanticscholar.org/CorpusID:971001>.

SMYTH, Gordon K.; JØRGENSEN, Bent. Fitting Tweedie's Compound Poisson Model to Insurance Claims Data: Dispersion Modelling. **ASTIN Bulletin: The Journal of the IAA**, Cambridge University Press, v. 32, n. 1, p. 143–157, 2002. DOI: 10.2143/AST.32.1.1020.

TWEEDIE, M. C. K. An index which distinguishes between some important exponential families. **Statistics: Applications and New Directions. Proceedings of the Indian Statistical Institute Golden Jubilee International Conference**, p. 579–604, 1984.

WYNN, P. On a Device for Computing the em(Sn) Transformation. **Mathematical Tables and Other Aids to Computation**, American Mathematical Society, v. 10, n. 54, p. 91–96, 1956. ISSN 08916837. Disponível em: <http://www.jstor.org/stable/2002183>. Acesso em: 25 dez. 2022.

Apêndices

APÊNDICE A – Provas

Prova do Teorema 1:

Seguindo (BREZINSKI, C., 1980) teorema 5. Por consequência do teorema de Toeplitz em solução de sistemas lineares. Temos que, se $\lim_{n\to\infty} E_{k-1}^{(n)} = L$, $\exists \alpha, \beta$ tais que $0 < \alpha < 1 < \beta$ e $\frac{g_{k-1,k}(n+1)}{g_{k-1,k}(n)} \in [\alpha, \beta] \forall n > N$, então $\lim_{n\to\infty} E_k^{(n)} = L$. Provemos com as hipóteses do teorema que $\lim_{n\to\infty} \frac{g_{k,i}(n+1)}{g_{k,i}(n)} = b_i \neq 1 \forall i$ (onde $g_{k,i}$ é a função g_i do passo k). Pelo resultado acima temos que todas as colunas da matriz $E_k^{(n)}$ convergem a L. Então seguimos a prova por indução. Assumindo valido para k-1, temos:

$$\frac{g_{k,i}(n+1)}{g_{k,i}(n)} = \frac{g_{k-1,k}(n+2)/g_{k-1,k}(n+1) - g_{k-1,i}(n+2)/g_{k-1,i}(n+1)}{g_{k-1,k}(n+1)/g_{k-1,k}(n) - g_{k-1,i}(n+1)/g_{k-1,i}(n)} \cdot \frac{g_{k-1,k}(n+1)/g_{k-1,k}(n) - 1}{g_{k-1,k}(n+2)/g_{k-1,k}(n+1) - 1} \frac{g_{k-1,i}(n+1)}{g_{k-1,i}(n)}$$

e

$$\lim_{n\to\infty} \frac{g_{k,i}(n+1)}{g_{k,i}(n)} = \frac{b_k - b_i}{b_k - b_i} \frac{b_k - 1}{b_k - 1} b_i = b_i$$

desde que $b_i \neq 1 \forall i$ e $b_k \neq b_i \forall i > k$. ■

Definição 4 (Constantes de Stieltjes)**.** *Constantes de Stieltjes são os números γ_k que aparecem na expansão da série de Laurent da função zeta de Riemann:*

$$\zeta(s) = \frac{1}{s-1} + \sum_{k=0}^{\infty} \frac{(-1)^k \gamma_k}{k!} (s-1)^k.$$

As constantes podem ser escritas como:

$$\gamma_n = \lim_{m\to\infty} \left(\sum_{k=1}^{m} \frac{(\ln k)^n}{k} - \frac{(\ln m)^{n+1}}{n+1} \right),$$

onde $\gamma_0 = \gamma$ (Euler-Mascheroni) por definição.

Lema 1. *Podemos escrever*

$$\zeta\left(1 + \frac{1}{n}\right) = n + \gamma + r_n,$$

onde $r_n \in (-\frac{1}{4n-2}, \frac{1}{4n-2}),\ \forall n \in \mathbb{N}$.

Prova: Pela expansão de Laurent e da definição acima (pondo $s = 1 + \frac{1}{n}$)

$$\begin{aligned}\zeta\left(1+\frac{1}{n}\right) &= \frac{1}{1+\frac{1}{n}-1} + \sum_{k=0}^{\infty}\frac{(-1)^k\gamma_k}{k!}\left(1+\frac{1}{n}-1\right)^k, \\ &= n + \sum_{k=0}^{\infty}\frac{(-1)^k\gamma_k}{k!n^k}, \\ &= n + \gamma + \sum_{k=1}^{\infty}\frac{(-1)^k\gamma_k}{k!n^k}.\end{aligned}$$

Agora basta mostrar que $r_n = \sum_{k=0}^{\infty}\frac{(-1)^k\gamma_k}{k!n^k} \in \left(-\frac{1}{4n-2}, \frac{1}{4n-2}\right),\ \forall n \in \mathbb{N}$. Usaremos o seguinte resultado provado em (LAVRIK, 1976):

$$|\gamma_k| \leq \frac{k!}{2^{k+1}}.$$

Pelo resultado,

$$\begin{aligned}|r_n| &= \left|\sum_{k=1}^{\infty}\frac{(-1)^k\gamma_k}{k!n^k}\right|, \\ &\leq \sum_{k=1}^{\infty}\frac{|\gamma_k|}{k!n^k}, \\ &= 2\sum_{k=1}^{\infty}\frac{1}{n^k 2^k} = \frac{1}{4n-2}.\end{aligned}$$

A última igualdade vale pois $\frac{1}{n^k 2^k}$ é uma progressão geométrica com razão $\frac{1}{2n}$. E pelo módulo:

$$r_n \in \left(-\frac{1}{4n-2}, \frac{1}{4n-2}\right),\ \forall n.$$

Em particular $\lim_{n\to\infty} r_n = 0$. ■

Lema 2. *Sendo $S_n = \sum_{i=1}^{n}\frac{1}{i^2}$ (soma parcial da série $\zeta(2)$), vale que*

$$|S_n - \zeta(2)| < \frac{1}{n},$$

e

$$|S_{n-1} - \zeta(2)| > \frac{1}{n}.$$

Prova: Vamos mostrar as 2 desigualdades,

$$
\begin{aligned}
|S_n - \zeta(2)| &\overset{(1)}{=} \zeta(2) - S_n, \\
&= \sum_{i=n+1}^{\infty} \frac{1}{i^2}, \\
&= \frac{1}{(n+1)^2} + \frac{1}{(n+2)^2} + \frac{1}{(n+3)^2} + \dots \\
&\overset{(2)}{<} \frac{1}{(n)(n+1)} + \frac{1}{(n+1)(n+2)} + \frac{1}{(n+2)(n+3)} + \dots \\
&\overset{(3)}{=} \left(\frac{1}{n} - \frac{1}{n+1}\right) + \left(\frac{1}{n+1} - \frac{1}{n+2}\right) + \left(\frac{1}{n+2} - \frac{1}{n+3}\right) + \dots \\
&= \frac{1}{n} + \left(\frac{1}{n+1} - \frac{1}{n+1}\right) + \left(\frac{1}{n+2} - \frac{1}{n+2}\right) + \left(\frac{1}{n+3} - \frac{1}{n+3}\right) + \dots \\
&= \frac{1}{n}.
\end{aligned}
$$

(1) S_n é monótona crescente.

(2) $\frac{1}{a^2} < \frac{1}{(a-1)(a)},\ \forall a \notin \{0, 1\}$.

(3) $\frac{1}{a(a+1)} = \frac{1}{a} - \frac{1}{a+1},\ \forall a \notin \{0, 1\}$.

De forma análoga à feita acima,

$$
\begin{aligned}
|S_{n-1} - \zeta(2)| &\overset{(1)}{=} \zeta(2) - S_{n-1}, \\
&= \sum_{i=n}^{\infty} \frac{1}{i^2}, \\
&= \frac{1}{n^2} + \frac{1}{(n+1)^2} + \frac{1}{(n+2)^2} + \dots \\
&\overset{(2)}{>} \frac{1}{(n)(n+1)} + \frac{1}{(n+1)(n+2)} + \frac{1}{(n+2)(n+3)} + \dots \\
&= \frac{1}{n} + \left(\frac{1}{n+1} - \frac{1}{n+1}\right) + \left(\frac{1}{n+2} - \frac{1}{n+2}\right) + \left(\frac{1}{n+3} - \frac{1}{n+3}\right) + \dots \\
&= \frac{1}{n}.
\end{aligned}
$$

(1) S_n é monótona crescente.

(2) $\frac{1}{a^2} > \frac{1}{(a)(a+1)},\ \forall a \notin \{-1, 0\}$.

(3) $\frac{1}{a(a+1)} = \frac{1}{a} - \frac{1}{a+1},\ \forall a \notin \{0, 1\}$.

Daí, segue que, $|S_n - \zeta(2)| < \frac{1}{n}$ e $|S_{n-1} - \zeta(2)| > \frac{1}{n}$. ■

APÊNDICE B – Verificando se uma transformação acelera uma série

Vamos provar usando o Lema 2 que a transformação de Richardson com p = 1 (correspondente a $T_n = 2S_{2n} - S_n$) acelera o problema de Basel. Sendo $L = \lim_{n\to\infty} S_n = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{1}{i^2}$, temos que satisfazer as 3 condições da Definição 3:

1. T_n *é convergente*. Note que,

$$\begin{aligned}\lim_{n\to\infty} T_n &= \lim_{n\to\infty} 2S_{2n} - S_n \\ &= 2\lim_{n\to\infty} S_{2n} - \lim_{n\to\infty} S_n \\ &= 2L - L = L\end{aligned}$$

2. T_n *converge para o mesmo limite de* S_n. Isto sai diretamente do item anterior.

3. T_n *converge mais rápido que* S_n, ou seja,

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

Usando o Lema 2

$$\begin{aligned}\lim_{n\to\infty} \frac{T_n - L}{S_n - L} &= \lim_{n\to\infty} \frac{2S_{2n} - S_n - L}{S_n - L} \\ &\leq \lim_{n\to\infty} \frac{2\left(L - \frac{1}{2n+1}\right) - \left(L - \frac{1}{n}\right) - L}{\left(L - \frac{1}{n}\right) - L} \\ &= \lim_{n\to\infty} \frac{\frac{2}{2n+1} - \frac{1}{n}}{\frac{1}{n}} \\ &= \lim_{n\to\infty} \frac{2n}{2n+1} - 1 = 0.\end{aligned}$$

Por outro lado,

$$\begin{aligned}
\lim_{n\to\infty}\frac{T_n-L}{S_n-L} &= \lim_{n\to\infty}\frac{2S_{2n}-S_n-L}{S_n-L}\\
&\geq \lim_{n\to\infty}\frac{2\left(L-\frac{1}{2n}\right)-\left(L-\frac{1}{n+1}\right)-L}{\left(L-\frac{1}{n+1}\right)-L}\\
&= \lim_{n\to\infty}\frac{\frac{2}{2n}-\frac{1}{n+1}}{\frac{1}{n+1}}\\
&= \lim_{n\to\infty}\frac{2(n+1)}{2n}-1=0.
\end{aligned}$$

Logo como $0 \leq \lim_{n\to\infty}\frac{T_n-L}{S_n-L} \leq 0$, então $\lim_{n\to\infty}\frac{T_n-L}{S_n-L}=0$. O que completa a prova. ■

APÊNDICE C – Tabelas adicionais

Tabelas referentes a Seção 3.3.2

	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

Tabela 5 – esum versos nsum com precisão 53

	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

Tabela 6 – esum versos nsum com precisão 100

Tabelas referentes a Seção 5

	erro		tempo (100/seg)		score
	Dias	Dias com 'Atiken'	Dias	Dias com 'Atiken'	
0.01	7.04510e-07	7.0451e-07	15.5832	25.9766	-1.2354e-07
0.1	2.02455e-08	2.0245e-08	1.7305	3.1145	2.7337e-05
0.2	1.2977e-11	8.6424e-12	1.0226	1.7721	0.5015
0.3	7.7060e-13	5.1159e-13	0.8142	1.4049	0.5062
0.4	6.0196e-13	1.6134e-12	0.8158	1.3689	-1.6803
0.5	9.1116e-13	2.0964e-12	0.8392	1.4035	-1.3008
0.6	3.2041e-13	1.7471e-11	1.0531	1.7558	-53.5284
0.7	9.9475e-14	2.5415e-10	1.2510	1.9312	-2553.9486
0.8	9.6034e-13	3.6559e-11	1.3458	2.1880	-37.0690
0.9	1.5676e-11	1.6644e-10	1.1886	1.8971	-9.6171
0.99	3.3810e-09	4.1836e-09	11.9989	19.6568	-0.2373

Tabela 7 – Dias vs. Dias com Aitken

α	erro		tempo (100/seg)		score
	Dias	Dias com 'Richardson'	Dias	Dias com 'Richardson'	
0.01	7.0451e-07	5.9007e+86	15.9217	9.9546	-8.3757e+92
0.1	2.0245e-08	7.7284e+11	1.5992	2.1811	-3.8173e+19
0.2	1.2977e-11	3.6531e+15	1.0278	1.3603	-2.8150e+26
0.3	7.7060e-13	1.1890e+42	0.8055	1.0983	-1.5430e+54
0.4	6.0196e-13	2.1741e+72	0.8610	1.0034	-3.6117e+84
0.5	9.1116e-13	1.5334e+120	0.8610	0.8578	-1.6830e+132
0.6	3.2041e-13	9.4682e+87	1.0690	1.1013	-2.9550e+100
0.7	9.9475e-14	2.7384e+81	1.1872	1.1731	-2.7529e+94
0.8	9.6034e-13	3.0522e+68	1.3307	1.2932	-3.1782e+80
0.9	1.5676e-11	6.5519e+36	1.2034	1.2464	-4.1793e+47
0.99	3.3810e-09	3.2959e+07	1.2202	1.6181	-9.7482e+15

Tabela 8 – Dias vs. Dias com Richardson

α	erro		tempo (100/seg)		score
	Dias	Dias com 'Epsilon'	Dias	Dias com 'Epsilon'	
0.01	7.0451e-07	7.0451e-07	15.4136	26.3696	-1.2449e-07
0.1	2.0245e-08	2.0245e-08	1.6451	3.0288	-1.1636e-05
0.2	1.2977e-11	1.4479e-11	0.9937	1.7145	-0.1157
0.3	7.7060e-13	1.3897e-12	0.8270	1.3802	-0.8034
0.4	6.0196e-13	5.9940e-13	0.7889	1.2993	0.0042
0.5	9.1116e-13	9.1116e-13	0.8367	1.2149	0.0
0.6	3.2041e-13	6.9101e-12	1.0564	1.6660	-20.5665
0.7	9.9475e-14	2.5602e-10	1.1857	1.9593	-2572.7790
0.8	9.6034e-13	3.6123e-11	1.3468	2.1109	-36.6152
0.9	1.5676e-11	1.6544e-10	1.1469	1.9220	-9.5538
0.99	3.3810e-09	4.1836e-09	1.2231	2.0508	-0.2373

Tabela 9 – Dias vs. Dias com Epsilon

α	erro		tempo (100/seg)		score
	Dias	Dias com 'G'	Dias	Dias com 'G'	
0.01	7.0451e-07	7.0483e-07	16.9232	50.1981	-0.0004
0.1	2.0245e-08	1.9856e-08	1.72033	6.2364	0.0195
0.2	1.2977e-11	2.9474e-10	1.0170	3.6232	-21.7125
0.3	7.7060e-13	4.5383e-10	0.8231	2.6347	-587.9376
0.4	6.0196e-13	3.1846e-10	0.7764	2.5704	-528.0450
0.5	9.1116e-13	6.4327e-11	0.7799	2.8271	-69.5992
0.6	3.2041e-13	1.2484e-10	1.0090	3.0941	-388.6500
0.7	9.9475e-14	2.1210e-09	1.1469	3.4435	-21320.7544
0.8	9.6034e-13	1.2298e-10	1.3709	4.0643	-127.0628
0.9	1.5676e-11	2.7182e-11	1.2125	3.6502	-0.7339
0.99	3.3810e-09	2.7865e-09	1.2857	3.7781	0.2133

Tabela 10 – Dias vs. Dias com G

α	erro		tempo (100/seg)		score
	Dias	Dias com 'Levin-u'	Dias	Dias com 'Levin-u'	
0.01	7.0451e-07	7.0451e-07	17.2020	30.7493	0.0
0.1	2.0245e-08	2.0245e-08	1.7990	3.5177	1.6451e-08
0.2	1.2977e-11	1.3048e-11	1.0820	2.0964	-0.0055
0.3	7.7060e-13	1.2749e-12	0.8953	1.6424	-0.6545
0.4	6.0196e-13	5.7753e-13	0.9094	1.6982	0.0422
0.5	9.1116e-13	5.9420e-12	0.9228	1.6906	-5.5213
0.6	3.2041e-13	2.1305e-11	1.2132	2.0285	-65.4948
0.7	9.9475e-14	1.4566e-11	1.3135	2.2582	-145.4308
0.8	9.6034e-13	3.4876e-10	1.4687	2.4981	-362.1701
0.9	1.5676e-11	9.4013e-12	1.2725	2.2461	0.6675
0.99	3.3810e-09	5.1945e-09	1.3758	2.4003	-0.5363

Tabela 11 – Dias vs. Dias com Levin-u

α	erro		tempo (100/seg)		score
	Dias	Dias com 'Levin-v'	Dias	Dias com 'Levin-v'	
0.01	7.0451e-07	7.0451e-07	16.7859	35.5557	0.0
0.1	2.0245e-08	2.0245e-08	1.7957	4.1192	0.0
0.2	1.2977e-11	1.3016e-11	1.0622	2.4981	-0.0030
0.3	7.7060e-13	5.8253e-13	0.8531	1.9118	0.3228
0.4	6.0196e-13	1.0935e-13	0.8695	1.9290	4.5045
0.5	9.1116e-13	9.1116e-13	0.8883	2.0706	0.0
0.6	3.2041e-13	9.0911e-12	1.1173	2.3041	-27.3735
0.7	9.9475e-14	4.0766e-10	1.2700	2.5697	-4097.1238
0.8	9.6034e-13	4.3417e-11	1.4258	2.8658	-44.2099
0.9	1.5676e-11	1.9412e-10	1.2430	2.5477	-11.3828
0.99	3.3810e-09	2.2994e-09	1.3130	2.7173	0.4703

Tabela 12 – Dias vs. Dias com Levin-v

APÊNDICE D – Gráficos adicionais

Gráficos adicionais da Seção 4

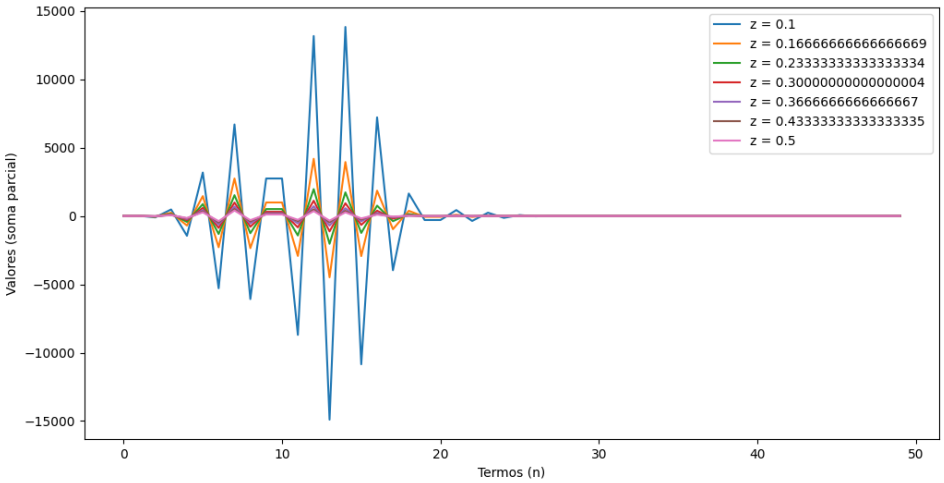

Figura 4 – $\alpha = 0.1,\ \theta = -1/2,\ z \in (0.1, 0.5),\ n = 50$

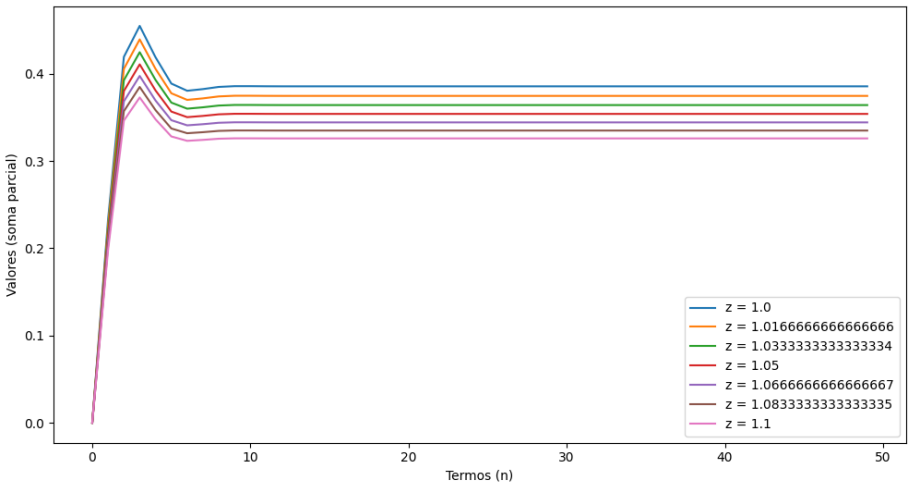

Figura 5 – $\alpha = 0.7,\ \theta = -1/2,\ z \in (1, 1.1),\ n = 50$

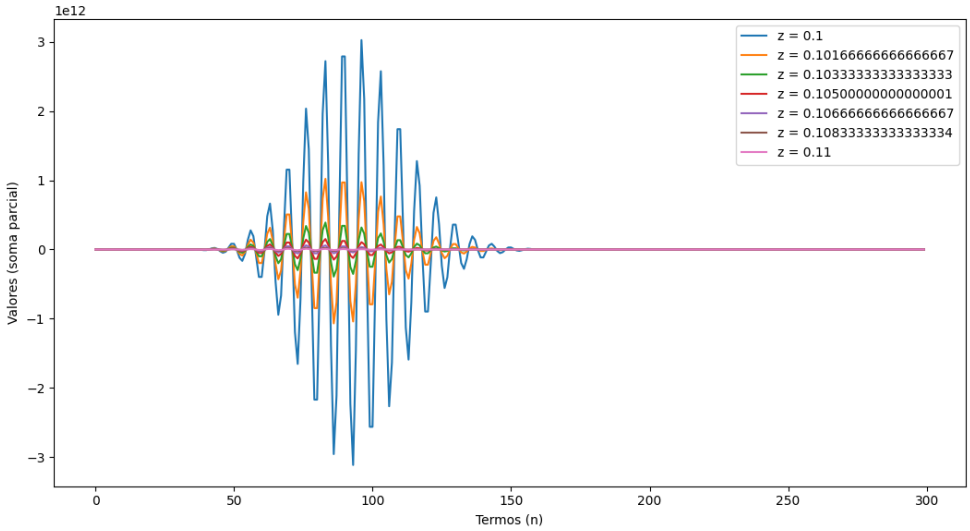

Figura 6 – $\alpha = 0.7,\ \theta = -1/2,\ z \in (0.1, 0.11),\ n = 300$