**Análise de Erros em Séries**

FCUP

Análise Numérica (M2018) 2018/2019

Trabalho 1

Ângelo Gomes – 201703990 – MIERSI

Eduardo Morgado – 201706894 – MIERSI

Simão Cardoso – 201604595 – MIERSI

Sónia Rocha – 201704679 – MIERSI

1. **Considerações Iniciais:**

Na realização deste trabalho, utilizamos como linguagem de implementação o Python, como tal, tendo em conta que, pontos de virgula flutuante em Python são sempre de precisão dupla, a evolução dos dados não é tão notória. Uma das soluções para este problema seria, através de bibliotecas como *numpy*, forçar o programa a trabalhar apenas em precisão simples, no entanto, todos os cálculos seriam realizados em precisão dupla e só depois convertidos/arredondados para precisão simples, o que acabaria por corromper na mesma os resultados. Sendo assim, durante todo este trabalho, iremos sempre ter em atenção esse fator, fator esse que influencia no calculo do *epsilon* máquina.

Uma nota importante para utilização dos próximos métodos em Python é a necessidade de configurar a divisão inteira como sendo uma divisão de pontos flutuantes, caso contrário, operações como 1/4=0 enão 0.25, para isso é necessário importar um parâmetro de uma biblioteca, *from \_\_future\_\_ import division*

1. **Cálculo do *Epsilon* Máquina:**

Antes de qualquer cálculo, é necessário perceber que o *epsilon* máquina é o menor número que, quando somado a 1, produza um resultado diferente de 1, ou seja, é o primeiro valor que, quando somado a 1, não leve a arredondamento, representando a exatidão relativa de um computador. Este valor surge provém da precisão finita de pontos flutuantes, uma vez que, em qualquer computador existem um número limitado de *bits* para representar um número, normalmente 32 *bits* ou 64 *bits*.

Para este trabalho, procuramos diferentes formas de calcular o *epsilon* (eps), as próximas implementações foram as mais simples de realizar:

* eps= numpy.finfo(float).eps
* eps=numpy.spacing(1.0)
* eps=2\*\*(-23)

Tanto a primeira como a segunda implementação, utilizam funções pré-definidas pela biblioteca *numpy*, a primeira produz um cálculo em precisão dupla (), enquanto que a segunda produz um cálculo em precisão simples(). A terceira implementação, provem da limitação de representação para pontos de vírgula flutuante. Em qualquer computador com implementação em precisão simples, dispomos apenas de 23 *bits* para a mantissa de um qualquer número, como tal, o menor valor possível a representar é. Para o resto deste trabalho e, tendo em conta, restrições de linguagem, iremos considerar como valor de *epsilon*.

**3. Resolução de Série de Termos Positivos (Exercício 2)**

Qualquer série pode ser decomposta em onde , pelo que, , logo, aproximar a série será encontrar um valor n tal que, , , uma vez que, a série tratada é uma série de termos positivos, podemos aplicar o critério de D’Alembert, o que permite concluir que , onde =L.

Para resolver este problema, primeiro é necessário verificar a convergência e determinar L:

L=1/4<1 pelo que a série converge, logo, basta encontrar n tal que, , é também importante referir que, uma vez que, este resultado irá facilitar o cálculo dos termos da série.

**4 Resolução de Série Alternada (Exercício 3)**

**5 Aplicação para valores Exatos (Exercício 4)**

**5.1 Implementação de Exercício 2 para Valor de Série Conhecido**

**5.2 Implementação de Exercício 3 para Valor de Série Conhecido**