

**Instituto Superior de Engenharia de Coimbra**

**Análise Matemática II**

**Atividade03Trabalho**

**Máquina para Derivação e Integração**

**Licenciatura em Engenharia Informática**

**2019 / 2020**

**TheForgotten**

[**someone@isec.pt**](mailto:someone@isec.pt)

**Conteúdo**

[1. Introdução 2](#_Toc42302328)

[2. Métodos Numéricos para Derivação 2](#_Toc42302329)

[2.1. Fórmulas 2](#_Toc42302330)

[2.1.1. Diferenças finitas em 2 pontos 2](#_Toc42302331)

[2.1.2. Diferenças finitas em 3 pontos 2](#_Toc42302332)

[2.2. Algoritmos / Funções 3](#_Toc42302333)

[2.2.1. Diferenças finitas em 2 pontos 3](#_Toc42302334)

[2.2.2. Diferenças finitas em 3 pontos 4](#_Toc42302335)

[3. Derivação Simbólica no MatLab 6](#_Toc42302336)

[3.1. diff 6](#_Toc42302337)

[4. Métodos Numéricos para Integração 7](#_Toc42302338)

[4.1. Fórmulas 7](#_Toc42302339)

[4.1.1. Regra dos Trapézios 7](#_Toc42302340)

[4.1.2. Regra de Simpson 7](#_Toc42302341)

[4.2. Algoritmos / Funções 8](#_Toc42302342)

[4.2.1. Regra dos Trapézios 8](#_Toc42302343)

[4.2.2. Regra de Simpson 8](#_Toc42302344)

[5. Integração Simbólica no MatLab 9](#_Toc42302345)

[5.1 int 9](#_Toc42302346)

[6. Exemplos de Aplicação e Teste dos Métodos 9](#_Toc42302347)

[6.1. Métodos Numéricos para Derivação 10](#_Toc42302348)

[6.2. Métodos Numéricos para Integração 11](#_Toc42302349)

[7. Conclusão 15](#_Toc42302350)

1. Introdução

Tal como sugerido, neste trabalho foi realizada uma máquina para derivação e integração. O objetivo desta máquina é ser capaz de calcular analiticamente e numericamente derivadas e integradas (definidas).

Esta máquina é baseada em 3 GUIs: uma GUI “mãe” e duas GUIs “filhas”. A primeira é responsável por obter as soluções analíticas para a derivada ou integrada duma dada função, as outras duas são responsáveis por obter as soluções aproximadas para o mesmo utilizando para isso os métodos numéricos sugeridos no enunciado.

2. Métodos Numéricos para Derivação

2.1. Fórmulas

Para todos os métodos, a fórmula para obter , que se trata do passo de discretização, para o intervalo e iterações é a seguinte:

2.1.1. Diferenças finitas em 2 pontos

Progressivas:

Regressivas:

2.1.2. Diferenças finitas em 3 pontos

Progressivas:

Regressivas:

Centradas:

2ª Derivada:

2.2. Algoritmos / Funções

Todas as funções começam por calcular um vetor x que vai conter os valores de x que vamos usar para obter a aproximação e o n que corresponde ao número de iterações. Após isso todas elas verificam se recebem 4 argumentos e, no caso de ser verdade, cria um vetor y que vai conter os valores da função f para os x obtidos anteriormente. De seguida, alocam espaço para o vetor dydx que vai conter as aproximações.

Por fim, aplicam o método e, antes de acabarem, colocam para os valores incalculáveis o valor possível mais próximo, de modo a que o gráfico fique mais belo.

2.2.1. Diferenças finitas em 2 pontos

Progressivas:

function [x, y, dydx] = NDerivacaoDFP2(f, a, b, h, y)

x = a: h: b;

n = length(x);

if nargin == 4

y = f(x);

end

dydx = zeros(1, n);

for i = 1: (n - 1)

dydx(i) = (y(i + 1) - y(i)) / h;

end

dydx(n) = (y(n) - y(n - 1)) / h;

end

Por se tratar dum método progressivo, o ciclo for começa no valor 1 e vai até ao valor n-1.

Regressivas:

function [x, y, dydx] = NDerivacaoDFR2(f, a, b, h, y)

x = a: h: b;

n = length(x);

if nargin == 4

y = f(x);

end

dydx = zeros(1, n);

for i = n: -1: 2

dydx(i) = (y(i) - y(i - 1)) / h;

end

dydx(1) = (y(2) - y(1)) / h;

end

Por se tratar dum método regressivo, o ciclo for começa no valor n e vai até ao valor 2.

2.2.2. Diferenças finitas em 3 pontos

Progressivas:

function [x, y, dydx] = NDerivacaoDFP3(f, a, b, h, y)

x = a: h: b;

n = length(x);

if nargin == 4

y = f(x);

end

dydx = zeros(1, n);

for i = 1: (n - 2)

dydx(i) = (-3 \* y(i) + 4 \* y(i + 1) - y(i + 2)) / (2 \* h);

end

dydx(n - 1) = (-3 \* y(n - 2) + 4 \* y(n - 1) - y(n)) / (2 \* h);

dydx(n) = (-3 \* y(n - 2) + 4 \* y(n - 1) - y(n)) / (2 \* h);

end

Por se tratar dum método progressivo o, o ciclo for começa no valor 1 e, por causa de ser em 3 pontos, vai até ao valor n-2.

Regressivas:

function [x, y, dydx] = NDerivacaoDFR3(f, a, b, h, y)

x = a: h: b;

n = length(x);

if nargin == 4

y = f(x);

end

dydx = zeros(1, n);

for i = n: -1: 3

dydx(i) = (y(i - 2) -4 \* y(i - 1) + 3 \* y(i)) / (2 \* h);

end

dydx(2) = (y(1) -4 \* y(2) + 3 \* y(3)) / (2 \* h);

dydx(1) = (y(1) -4 \* y(2) + 3 \* y(3)) / (2 \* h);

end

Por se tratar dum método regressivo, o ciclo for começa no valor n e, por causa de ser em 3 pontos, vai até ao valor 3.

Centradas:

function [x, y, dydx] = NDerivacaoDFC3(f, a, b, h, y)

x = a: h: b;

n = length(x);

if nargin == 4

y = f(x);

end

dydx = zeros(1, n);

for i = 2: (n - 1)

dydx(i) = (y(i + 1) - y(i - 1)) / (2 \* h);

end

dydx(1) = (y(3) - y(1)) / (2 \* h);

dydx(n) = (y(n) - y(n - 2)) / (2 \* h);

end

Por se tratar dum método centrado, o ciclo for começa no valor 2 e vai até ao valor n-1.

2ª Derivada:

function [x, y, dydx] = NDerivacaoDF2D(f, a, b, h, y)

x = a: h: b;

n = length(x);

if nargin == 4

y = f(x);

end

dydx = zeros(1, n);

for i = 2: (n - 1)

dydx(i) = (y(i + 1) - 2 \* y(i) + y(i - 1)) / h^2;

end

dydx(1) = (y(3) - 2 \* y(2) + y(1)) / h^2;

dydx(n) = (y(n) - 2 \* y(n - 1) + y(n - 2)) / h^2;

end

Por se tratar dum método centrado, o ciclo for começa no valor 2 e vai até ao valor n-1.

3. Derivação Simbólica no MatLab

3.1. diff

Apesar de poder ser usada de mais maneiras, neste trabalho apenas utilizamos a função diff() com a sintaxe diff(f(x)) pois apenas estamos interessados em calcular a primeira derivada da função fornecida. Se, por exemplo, quiséssemos calcular a terceira derivada, poderíamos chamar a função com um argumento extra que indica o número de vezes que derivamos, ou seja, diff(f(x), 3).

Com isto, na derivação, há a pequena chance de a derivada ser uma constante e, nesse caso, o vetor com as soluções exatas passaria a ser um valor e isso cria problemas para o resto do programa (deixa de funcionar corretamente). Para contornar isto incluí o seguinte pedaço de código:

if size(sExata) == 1

sExata = zeros(length(x), 1) + sExata;

sExata = sExata.';

end

Basicamente, isto verifica se o tamanho do “vetor” sExata é 1 e, se isso se verificar, o que ele faz preencher esse vetor com 0 de modo a ficar com o mesmo tamanho que o vetor x (vetor que guarda os valores para os quais calculamos a diferencial) e em seguida soma a cada um desses 0 a constante. Por fim, transforma esse vetor (que fica na vertical) no seu transposto de modo a conseguir funcionar corretamente com o restante código.

4. Métodos Numéricos para Integração

4.1. Fórmulas

Para todos os métodos, a fórmula para obter , que se trata do passo de discretização, para o intervalo e iterações é a seguinte:

4.1.1. Regra dos Trapézios

4.1.2. Regra de Simpson

4.2. Algoritmos / Funções

Ambas começam por calcular h e, de seguida, por fazer x = a e s = 0. Isto é necessário para que x tome o valor inicial que para que s esteja nulo de modo a se poder constantemente somar os valores do “interior” da fórmula sem adicionar lixo.

Como o algoritmo para estes métodos é fornecido pelo professor penso que dispensa grandes explicações.

4.2.1. Regra dos Trapézios

function sol = Trapezios(f, a, b, n)

h = (b - a) / n;

x = a;

s = 0;

for i = 1: (n - 1)

x = x + h;

s = s + (2 \* f(x));

end

sol = (h / 2) \* (f(a) + s + f(b));

end

4.2.2. Regra de Simpson

function sol = Simpson(f, a, b, n)

h = (b - a) / n;

x = a;

s = 0;

for i = 1: (n - 1)

x = x + h;

if (mod(i, 2) == 0)

s = s + (2 \* f(x));

else

s = s + (4 \* f(x));

end

end

sol = (h / 3) \* (f(a) + s + f(b));

end

Como para este método é necessário fazer a distinção entre i ser par e i ser ímpar, a minha solução foi usar a função mod(). Da maneira que é chamada, esta função vai devolver o resto da divisão inteira de i por 2. Se o resto for 0 significa que i é par, se não significa que i é ímpar.

5. Integração Simbólica no MatLab

5.1 int

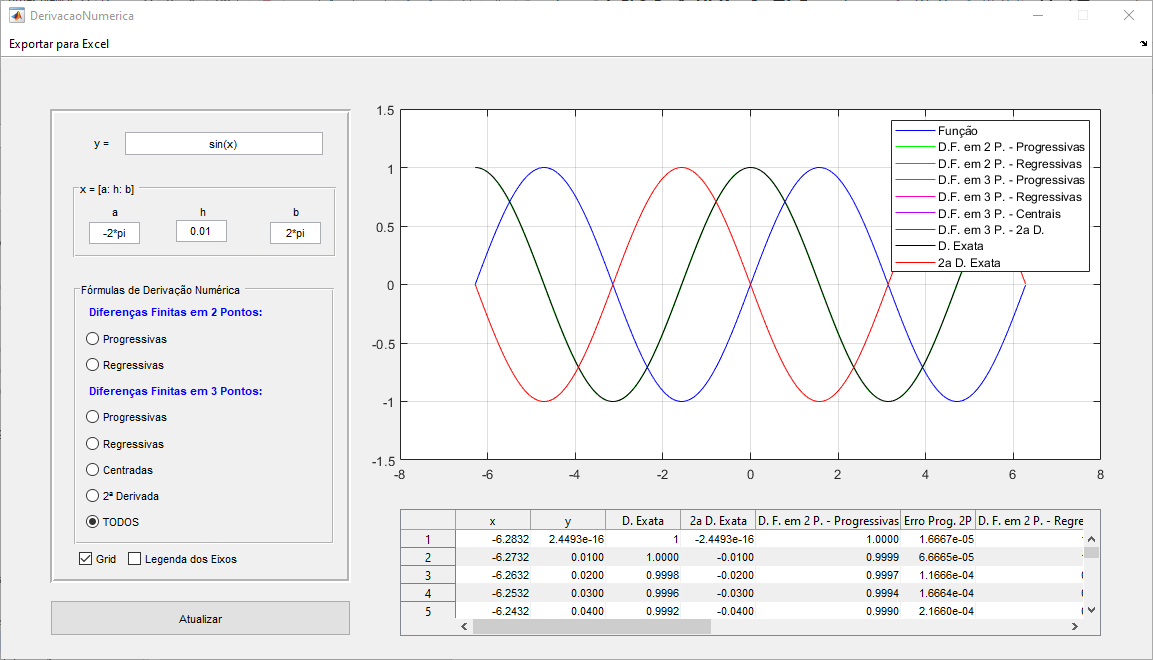
Esta função é utilizada para obter o valor exato dum dado determinado definido. Para isso, esta função deve ser invocada int(f, x, a, b) onde f é a função a integrar, x a variável escalar a respeitar e [a, b] o intervalo do integral.

Isto vai devolver uma expressão simbólica o que para o nosso caso é um problema pois para imprimir para a tabela os valores é necessário que estes sejam convertidos para string e não há maneira fácil de converter uma expressão simbólica para string em MatLab. Assim, o que podemos fazer é transformar a expressão simbólica num número usando double() e, em seguida, converter esse número para string usando num2str().

6. Exemplos de Aplicação e Teste dos Métodos

6.1. Métodos Numéricos para Derivação

Seja . Escrevendo esta função na nossa GUI e aplicando todos os métodos, podemos visualizar o seguinte:



Devido às limitações da GUI do MatLab, torna-se muito complicado gerir e manipular informação diretamente nela. Assim, comecemos por exportar os dados para Excel usando o botão no menu.

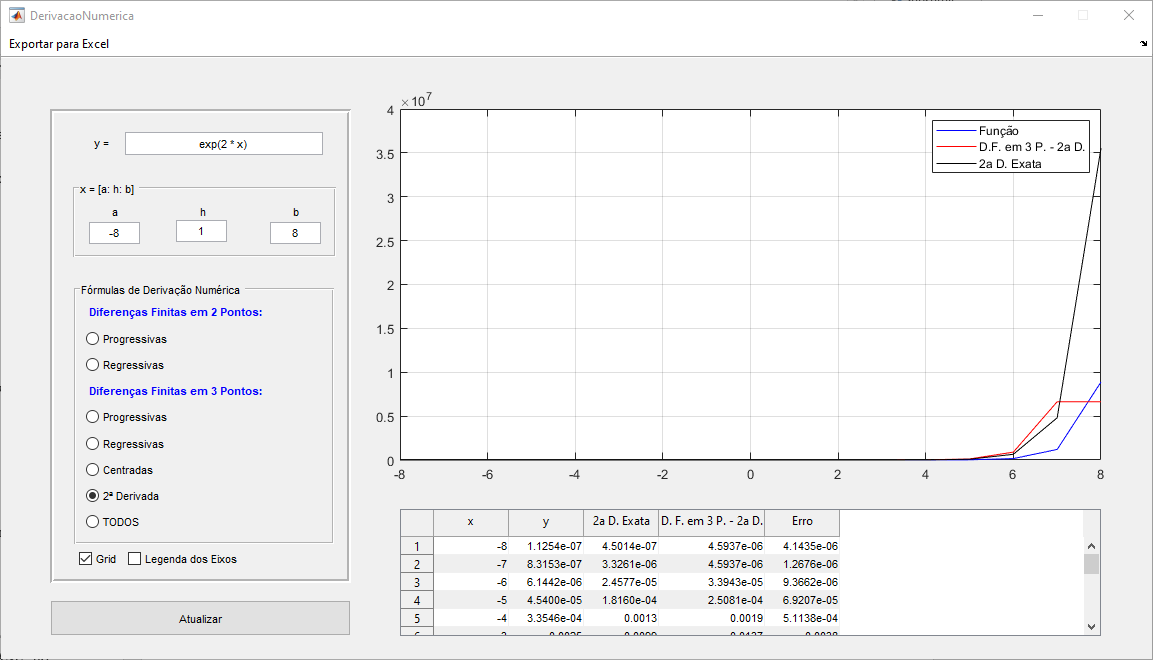
Já com os dados no Excel, para este caso, temos três colunas que não nos interessam de momento, que são as colunas para a segunda derivada, que serão apagadas. Dando uso às ferramentas disponibilizadas pelo Excel, facilmente obtemos a média do erro cometido por cada método numérico de derivação.

|  |  |
| --- | --- |
| Método | Média Erro |
| Prog. 2P | 0,003182191 |
| Regr. 2P | 0,003182188 |
| Prog. 3P | 2,15485E-05 |
| Regr. 3P | 0,000288465 |
| Centrais 3P | 1,07423E-05 |

Deste modo, podemos concluir que o método que oferece menor erro (em média) é o método centrado das diferenças finitas em 3 pontos.

Seja . Qual o valor de

Usando a GUI programada, inserindo a função no respetivo campo e selecionando o método para a segunda derivada, obtemos o seguinte:



O intervalo escolhido foi [-8, 8] e o de modo a ter contido o valor que pretendemos obter.

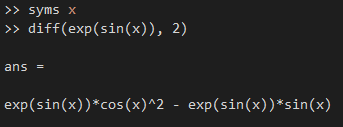
Podemos assim concluir que e que a solução aproximada dada pela fórmula da diferença finita em 3 pontos para a segunda derivada nos diz que .

6.2. Métodos Numéricos para Integração

Os exercícios foram tirados do PDF “Cap5\_IntegracaoNumerica.pdf” que pode ser encontrado na secção **Derivação e Integração Numérica** da página da disciplina (AM2 no Moodle).

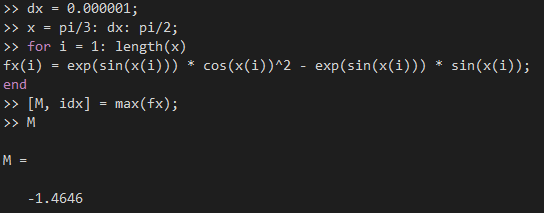
Exercício 1:

* a)



Deste modo, ficamos a saber que .

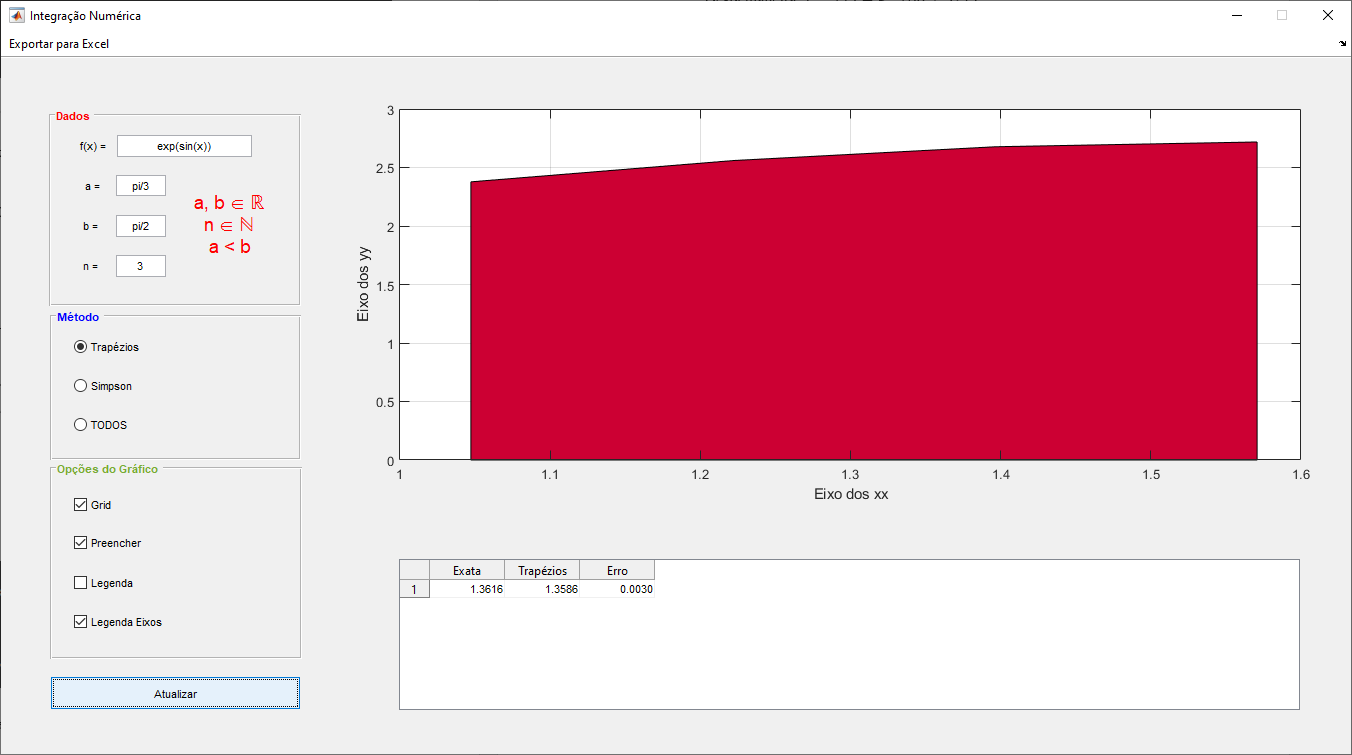
Utilizando a função max() do MatLab, conseguimos obter o máximo local para o intervalo de iteração sem ser preciso manipular a derivada da expressão.



Deste modo, sabemos que .

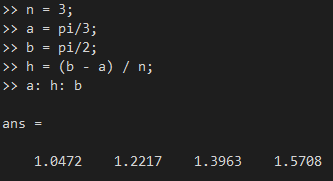
Assim, aplicando a fórmula do erro:

Assim, façamos . Para obter n façamos . Deste modo, para manter o erro inferior a 0.0031, consideremos .



* b)

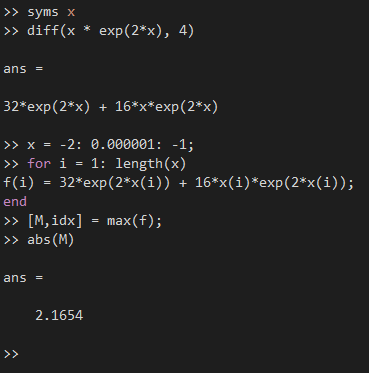
Para obter os pontos que precisamos de conhecer da integranda basta-nos apenas calcular um vetor que contenha os valores do intervalo [a, b] de em passos.



Assim, como podemos visualizar, os pontos que precisamos de conhecer da função integranda são .

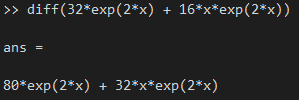
Exercício 4:

* a)



Deste modo, ficamos a saber que .

Para calcularmos o máximo de precisamos da sua derivada.



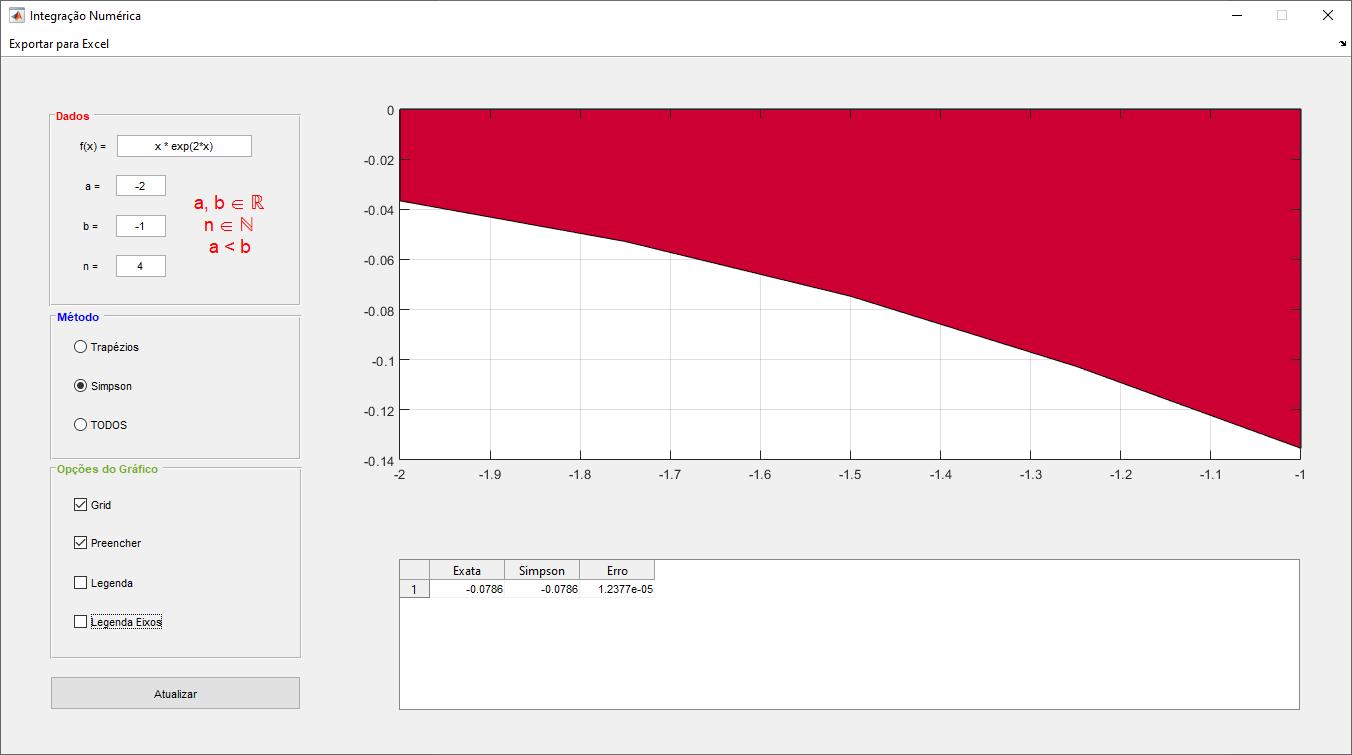
Desta maneira, .

Assim, calculando os zeros da função para posteriormente podermos obter o máximo no intervalo de integração:

|  |  |  |  |
| --- | --- | --- | --- |
|  | -2 |  | -1 |
|  | + | + | + |
|  | mínimo local | + | Máximo local |

Deste modo, aplicando a fórmula do erro:

* b)

 Assim, façamos . Para obter n façamos . Deste modo, para manter o erro inferior a 0.0005, e como n tem de ser par para a regra de Simpson, consideremos .

7. Conclusão

Após a realização deste trabalho sinto que estou muito mais à vontade e entendo melhor o conceito de derivadas e integrais e os métodos numéricos dos mesmos. Assim, fico grato por ter a oportunidade de ter realizado este trabalho.