Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

SME0104 – Cálculo Numérico

Trabalho 2 – Solução de uma EDO de 2ª Ordem pelo Método de Euler Modificado

Elias Italiano Rodrigues – 7987251

São Carlos, 24 de junho de 2014

1 Introdução

1.1 Solução Numérica de EDO

A maioria das Equações Diferenciais Ordinárias (EDO) encontradas na prática na área de engenharia, matemática, física, computação etc, não podem ser resolvidas analiticamente: o único recurso é a aplicação de um método numérico iterativo para encontrar uma solução aproximada que satisfaça o problema. No final dessa seção, veremos a fórmula iterativa do Método de Euler Modificado que foi usada para programar o problema que descrito no enunciado do trabalho.

1.2 Transformação de um PVI de Ordem $m > 1$ em um Sistema de EDO de 1ª Ordem

Muitas vezes, para resolver um Problema de Valor Inicial (PVI) de ordem maior que um, precisamos transformá-lo em um sistema de EDO de 1ª Ordem e então aplicar um método numérico conhecido para obter a solução.

Um PVI de ordem $m > 1$, tem a seguinte forma geral:

$$\begin{cases} y^{(m)}(x) &= f(x, y(x), y'(x), \ldots, y^{(m-1)}(x)) \\ y(a) &= y_a \\ y'(a) &= y'_a \\ \vdots &= \vdots \\ y^{(m-1)}(a) &= y_a^{(m-1)} \end{cases}$$

Esse PVI pode ser transformado em um sistema de equações de 1ª ordem com as seguintes m-equações fazendo a transformação:

$$\begin{cases} y_1(x) &= y(x) \\ y_2(x) &= y'(x) \\ \vdots &= \vdots \\ y_m(x) &= y^{(m-1)}(x) \end{cases}$$

E então, derivando os dois dados, vem:

$$\begin{cases} y'_1(x) &= y_2(x) &, \quad y_1(a) = y_a \\ y'_2(x) &= y_3(x) &, \quad y_2(a) = y'_a \\ \vdots &= \vdots &, \quad \vdots \\ y'_m(x) &= f(x, y_1(x), y_2(x), \ldots, y_m(x)) &, \quad y_m(a) = y_a^{(m-1)} \end{cases}$$

Escrevendo esse sistema na forma vetorial, temos:

$$\begin{cases} \mathbf{y}' &= \mathbf{F}(x, \mathbf{y}) \ , \ a \leq x \leq b \\ \mathbf{y}(a) &= \mathbf{y}_a \end{cases}$$

onde,

$$\mathbf{y} = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_m(x) \end{bmatrix}, \mathbf{y}' = \begin{bmatrix} y_1'(x) \\ y_2'(x) \\ \vdots \\ y_m'(x) \end{bmatrix}, \mathbf{F}(x, \mathbf{y}) = \begin{bmatrix} y_2(x) \\ y_3(x) \\ \vdots \\ f(x, y_1(x), y_2(x), \ldots, y_m(x)) \end{bmatrix} \text{ e } \mathbf{y}_a = \begin{bmatrix} y_a \\ y_a' \\ \vdots \\ y_a^{(m-1)} \end{bmatrix}$$

1.3 O Método de Euler Modificado

Dado um PVI da forma

$$\begin{cases} y' &= f(x, y),\ a \leq x \leq b \\ y(a) &= a \end{cases}$$

No Método de Euler Modificado, divide-se o intervalo $[a, b]$ do PVI em intervalos h de modo que $h = (b - a)/N$ e define-se os pontos $x_j = a + j \cdot h$, para $j = 0, 1, 2, \ldots, N$

E então, a solução é encontrada a partir da iteração da seguinte fórmula:

$$y_{j+1} = y_j + \frac{h}{2}\Big[f(x_j, y_j) + f(x_{j+1}, \overline{y}_{j+1})\Big]$$

onde $x_{j+1} = x_j + h$ e $\overline{y}_{j+1} = y_j + h \cdot f(x_j, y_j)$

Um algoritmo para execução desse método consiste na aplicação dos seguintes passos:

1. Calcular $f(x_j, y_j)$
2. Calcular $\overline{y}_{j+1}$
3. Calcular $f(x_{j+1}, \overline{y}_{j+1})$
4. Calcular y_{j+1}

De maneira análoga, o Método de Euler Modificado pode ser aplicado a um sistema de EDO de 1ª ordem escrito na forma vetorial como visto no tópico anterior. Logo, dado um PVI de ordem maior que um, pode-se transformá-lo em um sistema de EDO de 1ª ordem, escrevê-lo na forma vetorial e aplicar este método iterativo para se obter a solução.

1.4 O Trabalho

Este trabalho consiste em implementar um programa de computador que execute o Método de Euler Modificado para a resolução do seguinte PVI de ordem 2:

$$\begin{cases} y'' &= y + e^x\ ,\ x \in [0, 2] \\ y(0) &= 1 \\ y'(0) &= 0 \end{cases}$$

Que tem a solução analítica conhecida e dada pelo enunciado:

$$y(x) = \frac{1}{4}\Big[e^x(1+2x) + 3e^{-x}\Big]$$

O programa é iterado para os valores de $h_k = 0.2/(2^k)$, $k = 1, 2, 3, 4$ de modo que se observe a convergência do método de Euler Modificado quando $h \to 0$.

Além disso, para cada k é calculada a ordem de convergência com $n_k = \log_2(E_{h_k}/E_{h_{k+1}})$, $k = 1, 2, 3$ onde E_{h_k} é o erro relativo dado por $E_{h_k} = \frac{\sqrt{\sum_{i=1}^{N_k}\left[y(x_i) - y_i^{h_k}\right]^2}}{\sqrt{\sum_{i=1}^{N_k} y(x_i)^2}}$, em que N_k é o número de pontos na malha h_k e $y_i^{h_k}$ representa a solução obtida pelo Método de Euler Modificado com $h = h_k$.

2 Desenvolvimento

2.1 Transformação

Antes de ser implementado no código, o PVI de segunda ordem a ser resolvido foi transformado no seguinte sistema de equações seguindo as transformações mostradas na seção 1:

$$\begin{cases} \mathbf{y}' &= \mathbf{F}(x, \mathbf{y}) \ , \ 0 \leq x \leq 2 \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{cases}$$

onde,

$$\mathbf{y} = \begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} \ , \ \mathbf{y}' = \begin{bmatrix} y_1'(x) \\ y_2'(x) \end{bmatrix} \ , \ \mathbf{F}(x, \mathbf{y}) = \begin{bmatrix} y_2(x) \\ y_1(x) + e^x \end{bmatrix} \text{ e } \mathbf{y}(0) = \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \mathbf{y}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

2.2 Implementação

O programa foi desenvolvido em linguagem `C` em sistema operacional `Linux`. Além das bibliotecas convencionais `stdlib.h` e `stdio.h`, foi utilizada também a `math.h` para usar a função `log()` que retorna o valor do logaritmo natural de um numero, `exp()` que retorna o valor da exponencial e elevado a um número e `sqrt()` que retorna a raiz quadrada de um número real.

Para representar os números reais no programa foi utilizado o tipo de dado de ponto flutuante `double` que é próprio da linguagem `C`.

O código do programa consiste nas seguintes funções:

- `int main(int argc, char **argv)`
  Função principal do programa que é responsável por iterar k vezes a função `calcular()`.

- `double euler_mod(double x, double h)`
  Retorna o valor da solução de Euler Modificado para um dado `x` e considerando intervalo de tamanho `h`.

- `double analitica(double x)`
  Retorna o valor da solução analítica para um dado `x`.

- `void calcular(int k)`
  Efetua os cálculos iterativos chamando as devidas funções e imprime o resultado na tela.

O vetores com os valores do PVI e com os valores dos erros foram criados globais e estão declarados e documentados no começo do código-fonte.

2.3 Código-fonte

main.c

```c
/**
 * Elias Italiano Rodrigues, 7987251
 *
 * SME0104 Calculo Numerico: Trabalho 2
 * 2014/06/24
 */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

// Vetores do PVI
double  y[2] = {1.0, 0.0}; // solucao
double  F[2] = {0.0, 0.0}; // F
double _F[2] = {0.0, 0.0}; // F calculado com y barra
double _y[2] = {0.0, 0.0}; // y barra

// Erros
double erros[4][3] = {
// resultado, soma do numerador, soma do denominador
    {0.0, 0.0, 0.0},
    {0.0, 0.0, 0.0},
    {0.0, 0.0, 0.0},
    {0.0, 0.0, 0.0}
};

/**
 * Resposta da solucao de Euler Modificado para um dado x
 * e considerando intervalo de tamanho h
 */
double euler_mod(double x, double h)
{
    // Passo 1
    F[0] = y[1];
    F[1] = y[0] + exp(x - h);

    // Passo 2
    _y[0] = y[0] + h * F[0];
    _y[1] = y[1] + h * F[1];
```

```
    // Passo 3
    _F[0] = _y[1];
    _F[1] = _y[0] + exp(x);

    // Passo 4
    y[0] = y[0] + (h * (F[0] + _F[0])) / 2.0;
    y[1] = y[1] + (h * (F[1] + _F[1])) / 2.0;

    return y[0];
}

/**
 * Resposta da solucao analitica para um dado x
 */
double analitica(double x)
{
    return 0.25 * (exp(x) * (1.0 + 2.0 * x) + 3.0 * exp(-x));
}

/**
 * Efetua todos os calculos iterativos chamando as devidas funcoes
 * e imprime os resultados na tela
 */
void calcular(int k)
{
    int j;
    double h = 0.2 / pow(2.0, (double)k); // calcula h
    int N = (int)(2.0 / h); // calcula N
    double x, res1, res2;

    // Reseta os valores
    y[0] = 1.0;
    y[1] = 0.0;

    printf("\n Resultados para k = %d\n", k);
    printf("---------------------------------\n");
    printf(" x        Euler     Analitica\n");
    printf("---------------------------------\n");
    // Iterando em cada um dos pontos
    for (j = 0; j < N + 2; j++) {
        // Obtem valores
        x = (double)j * h;
        res1 = j > 0 ? euler_mod(x, h) : y[0];
        res2 = analitica(x);

        // Imprime
        printf(" %lf  %lf   %lf\n", x, res1, res2);

        // Acumula erros
        erros[k-1][1] += pow(res2 - res1, 2.0);
        erros[k-1][2] += pow(res2, 2.0);
    }
```

```
    // Calcula e imprime o erro relativo
    erros[k-1][0] = sqrt(erros[k-1][1] / erros[k-1][2]);
    printf("---------------------------------\n");
    printf(" Erro relativo: %lf\n", erros[k-1][0]);
    printf("---------------------------------\n");
}

/**
 * Programa principal
 */
int main(int argc, char **argv)
{
    int k;

    // Imprimindo informacoes do problema
    printf("SME0104 Calculo Numerico\nTrabalho 2: Metodo de Euler Modificado\n\n");
    printf(" EDO alvo : y'' = y + e^x\n");
    printf("Intervalo : x em [0,2]\n");
    printf(" Valores : y(0) = 1 e y'(0) = 0\n");

    // Resultados para k = 1, 2, 3, 4
    for (k = 1; k < 5; k++)
        calcular(k);

    // Ordens de convergencia
    printf("\n k  Convergencia\n");
    printf("------------------\n");
    for (k = 0; k < 3; k++)
        printf(" %d  %lf\n", k + 1, log(erros[k][0] / erros[k+1][0]) / log(2.0));

    return EXIT_SUCCESS;
}
```

2.4 Compilação e Execução

Para compilar o código-fonte foi utilizado o `gcc` (GNU project C and C++ compiler):
`gcc -o t2_euler-mod main.c -lm`

A execução do programa, se este estiver no diretório atual, dá-se por:
`./t2_euler-mod`

O programa não pede por dados de entrada.

3 Resultados

Executando o programa para os casos com $k = 1, 2, 3, 4$ obteve-se os seguintes resultados:

```
SME0104 Calculo Numerico
Trabalho 2: Metodo de Euler Modificado

 EDO alvo : y'' = y + e^x
Intervalo : x em [0,2]
  Valores : y(0) = 1 e y'(0) = 0

 Resultados para k = 1
--------------------------------
 x          Euler      Analitica
--------------------------------
 0.000000   1.000000   1.000000
 0.100000   1.010000   1.010179
 0.200000   1.041102   1.041539
 0.300000   1.094776   1.095557
 0.400000   1.172839   1.174061
 0.500000   1.277488   1.279259
 0.600000   1.411333   1.413774
 0.700000   1.577439   1.580691
 0.800000   1.779376   1.783598
 0.900000   2.021273   2.026649
 1.000000   2.307878   2.314621
 1.100000   2.644633   2.652986
 1.200000   3.037750   3.047995
 1.300000   3.494304   3.506766
 1.400000   4.022336   4.037388
 1.500000   4.630963   4.649037
 1.600000   5.330516   5.352106
 1.700000   6.132676   6.158355
 1.800000   7.050645   7.081069
 1.900000   8.099327   8.135250
 2.000000   9.295534   9.337822
--------------------------------
 Erro relativo: 0.004119
--------------------------------

 Resultados para k = 2
--------------------------------
 x          Euler      Analitica
--------------------------------
 0.000000   1.000000   1.000000
 0.050000   1.002500   1.002522
 0.100000   1.010131   1.010179
 0.150000   1.023048   1.023127
 0.200000   1.041423   1.041539
 0.250000   1.065452   1.065610
```

```
 0.300000   1.095351   1.095557
 0.350000   1.131359   1.131620
 0.400000   1.173740   1.174061
 0.450000   1.222781   1.223169
 0.500000   1.278795   1.279259
 0.550000   1.342123   1.342670
 0.600000   1.413136   1.413774
 0.650000   1.492231   1.492970
 0.700000   1.579842   1.580691
 0.750000   1.676430   1.677400
 0.800000   1.782497   1.783598
 0.850000   1.898578   1.899823
 0.900000   2.025248   2.026649
 0.950000   2.163124   2.164695
 1.000000   2.312865   2.314621
 1.050000   2.475176   2.477133
 1.100000   2.650812   2.652986
 1.150000   2.840576   2.842987
 1.200000   3.045329   3.047995
 1.250000   3.265986   3.268929
 1.300000   3.503524   3.506766
 1.350000   3.758983   3.762549
 1.400000   4.033472   4.037388
 1.450000   4.328171   4.332464
 1.500000   4.644335   4.649037
 1.550000   4.983301   4.988443
 1.600000   5.346490   5.352106
 1.650000   5.735413   5.741541
 1.700000   6.151675   6.158355
 1.750000   6.596984   6.604258
 1.800000   7.073154   7.081069
 1.850000   7.582112   7.590716
 1.900000   8.125904   8.135250
 1.950000   8.706704   8.716848
 2.000000   9.326819   9.337822
---------------------------------
 Erro relativo: 0.001064
---------------------------------

 Resultados para k = 3
---------------------------------
 x          Euler      Analitica
---------------------------------
 0.000000   1.000000   1.000000
 0.025000   1.000625   1.000628
 0.050000   1.002516   1.002522
 0.075000   1.005690   1.005699
 0.100000   1.010167   1.010179
 0.125000   1.015965   1.015982
 0.150000   1.023107   1.023127
```

0.175000	1.031613	1.031638
0.200000	1.041509	1.041539
0.225000	1.052819	1.052854
0.250000	1.065569	1.065610
0.275000	1.079788	1.079835
0.300000	1.095504	1.095557
0.325000	1.112748	1.112808
0.350000	1.131553	1.131620
0.375000	1.151951	1.152026
0.400000	1.173979	1.174061
0.425000	1.197672	1.197763
0.450000	1.223070	1.223169
0.475000	1.250212	1.250321
0.500000	1.279140	1.279259
0.525000	1.309898	1.310027
0.550000	1.342530	1.342670
0.575000	1.377085	1.377236
0.600000	1.413611	1.413774
0.625000	1.452159	1.452334
0.650000	1.492781	1.492970
0.675000	1.535534	1.535737
0.700000	1.580474	1.580691
0.725000	1.627659	1.627891
0.750000	1.677152	1.677400
0.775000	1.729016	1.729280
0.800000	1.783317	1.783598
0.825000	1.840123	1.840422
0.850000	1.899505	1.899823
0.875000	1.961536	1.961873
0.900000	2.026292	2.026649
0.925000	2.093851	2.094230
0.950000	2.164294	2.164695
0.975000	2.237706	2.238130
1.000000	2.314173	2.314621
1.025000	2.393785	2.394258
1.050000	2.476634	2.477133
1.075000	2.562816	2.563343
1.100000	2.652431	2.652986
1.125000	2.745581	2.746166
1.150000	2.842372	2.842987
1.175000	2.942912	2.943559
1.200000	3.047315	3.047995
1.225000	3.155697	3.156412
1.250000	3.268178	3.268929
1.275000	3.384883	3.385671
1.300000	3.505939	3.506766
1.325000	3.631479	3.632346
1.350000	3.761639	3.762549
1.375000	3.896561	3.897514
1.400000	4.036389	4.037388

```
 1.425000   4.181274   4.182320
 1.450000   4.331370   4.332464
 1.475000   4.486836   4.487982
 1.500000   4.647838   4.649037
 1.525000   4.814545   4.815799
 1.550000   4.987132   4.988443
 1.575000   5.165780   5.167150
 1.600000   5.350674   5.352106
 1.625000   5.542008   5.543504
 1.650000   5.739978   5.741541
 1.675000   5.944789   5.946421
 1.700000   6.156651   6.158355
 1.725000   6.375782   6.377559
 1.750000   6.602404   6.604258
 1.775000   6.836747   6.838682
 1.800000   7.079051   7.081069
 1.825000   7.329558   7.331662
 1.850000   7.588522   7.590716
 1.875000   7.856202   7.858489
 1.900000   8.132867   8.135250
 1.925000   8.418791   8.421275
 1.950000   8.714261   8.716848
 1.975000   9.019569   9.022263
 2.000000   9.335016   9.337822
---------------------------------
 Erro relativo: 0.000270
---------------------------------

 Resultados para k = 4
---------------------------------
 x          Euler      Analitica
---------------------------------
 0.000000   1.000000   1.000000
 0.012500   1.000156   1.000157
 0.025000   1.000627   1.000628
 0.037500   1.001414   1.001415
 0.050000   1.002520   1.002522
 0.062500   1.003947   1.003949
 0.075000   1.005697   1.005699
 0.087500   1.007773   1.007775
 0.100000   1.010176   1.010179
 0.112500   1.012910   1.012914
 0.125000   1.015977   1.015982
 0.137500   1.019380   1.019385
 0.150000   1.023122   1.023127
 0.162500   1.027205   1.027211
 0.175000   1.031632   1.031638
 0.187500   1.036407   1.036413
 0.200000   1.041531   1.041539
 0.212500   1.047010   1.047018
```

0.225000	1.052845	1.052854
0.237500	1.059041	1.059050
0.250000	1.065600	1.065610
0.262500	1.072526	1.072537
0.275000	1.079823	1.079835
0.287500	1.087494	1.087507
0.300000	1.095544	1.095557
0.312500	1.103975	1.103990
0.325000	1.112793	1.112808
0.337500	1.122001	1.122017
0.350000	1.131603	1.131620
0.362500	1.141603	1.141621
0.375000	1.152007	1.152026
0.387500	1.162818	1.162837
0.400000	1.174040	1.174061
0.412500	1.185679	1.185701
0.425000	1.197740	1.197763
0.437500	1.210227	1.210251
0.450000	1.223144	1.223169
0.462500	1.236498	1.236524
0.475000	1.250293	1.250321
0.487500	1.264535	1.264564
0.500000	1.279229	1.279259
0.512500	1.294380	1.294411
0.525000	1.309994	1.310027
0.537500	1.326077	1.326111
0.550000	1.342635	1.342670
0.562500	1.359673	1.359710
0.575000	1.377198	1.377236
0.587500	1.395216	1.395256
0.600000	1.413733	1.413774
0.612500	1.432755	1.432798
0.625000	1.452290	1.452334
0.637500	1.472343	1.472390
0.650000	1.492923	1.492970
0.662500	1.514034	1.514084
0.675000	1.535686	1.535737
0.687500	1.557884	1.557937
0.700000	1.580636	1.580691
0.712500	1.603950	1.604006
0.725000	1.627833	1.627891
0.737500	1.652293	1.652353
0.750000	1.677337	1.677400
0.762500	1.702975	1.703040
0.775000	1.729214	1.729280
0.787500	1.756062	1.756130
0.800000	1.783527	1.783598
0.812500	1.811619	1.811693
0.825000	1.840347	1.840422
0.837500	1.869718	1.869796

0.850000	1.899743	1.899823
0.862500	1.930429	1.930512
0.875000	1.961788	1.961873
0.887500	1.993828	1.993916
0.900000	2.026559	2.026649
0.912500	2.059991	2.060084
0.925000	2.094134	2.094230
0.937500	2.128998	2.129097
0.950000	2.164594	2.164695
0.962500	2.200932	2.201036
0.975000	2.238023	2.238130
0.987500	2.275878	2.275988
1.000000	2.314508	2.314621
1.012500	2.353924	2.354040
1.025000	2.394138	2.394258
1.037500	2.435162	2.435285
1.050000	2.477007	2.477133
1.062500	2.519685	2.519815
1.075000	2.563210	2.563343
1.087500	2.607592	2.607729
1.100000	2.652846	2.652986
1.112500	2.698984	2.699127
1.125000	2.746018	2.746166
1.137500	2.793963	2.794114
1.150000	2.842831	2.842987
1.162500	2.892638	2.892797
1.175000	2.943396	2.943559
1.187500	2.995119	2.995287
1.200000	3.047823	3.047995
1.212500	3.101522	3.101698
1.225000	3.156231	3.156412
1.237500	3.211965	3.212150
1.250000	3.268739	3.268929
1.262500	3.326570	3.326764
1.275000	3.385472	3.385671
1.287500	3.445462	3.445666
1.300000	3.506557	3.506766
1.312500	3.568773	3.568987
1.325000	3.632127	3.632346
1.337500	3.696637	3.696861
1.350000	3.762319	3.762549
1.362500	3.829192	3.829427
1.375000	3.897273	3.897514
1.387500	3.966582	3.966828
1.400000	4.037136	4.037388
1.412500	4.108954	4.109212
1.425000	4.182055	4.182320
1.437500	4.256460	4.256730
1.450000	4.332188	4.332464
1.462500	4.409259	4.409541

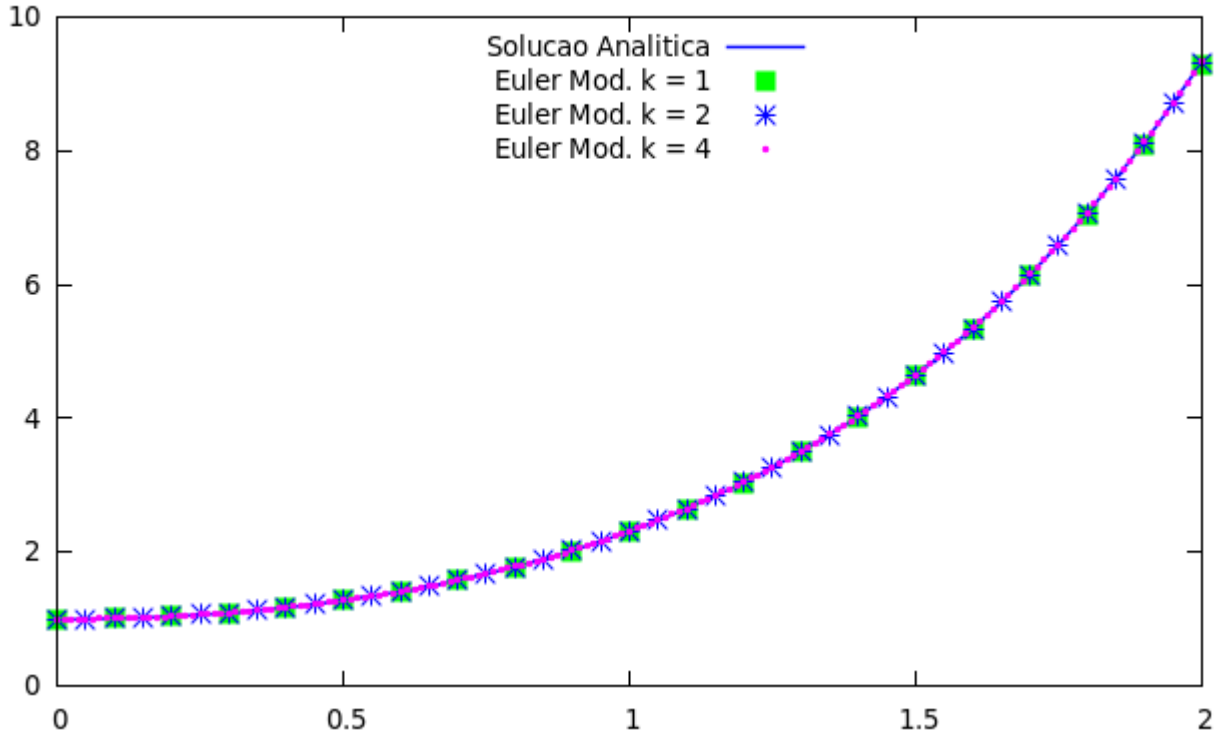
```
 1.475000   4.487693   4.487982
 1.487500   4.567511   4.567807
 1.500000   4.648734   4.649037
 1.512500   4.731384   4.731693
 1.525000   4.815482   4.815799
 1.537500   4.901051   4.901374
 1.550000   4.988112   4.988443
 1.562500   5.076689   5.077027
 1.575000   5.166804   5.167150
 1.587500   5.258482   5.258835
 1.600000   5.351745   5.352106
 1.612500   5.446618   5.446988
 1.625000   5.543126   5.543504
 1.637500   5.641294   5.641680
 1.650000   5.741146   5.741541
 1.662500   5.842709   5.843112
 1.675000   5.946009   5.946421
 1.687500   6.051072   6.051493
 1.700000   6.157925   6.158355
 1.712500   6.266595   6.267034
 1.725000   6.377111   6.377559
 1.737500   6.489499   6.489958
 1.750000   6.603790   6.604258
 1.762500   6.720012   6.720490
 1.775000   6.838194   6.838682
 1.787500   6.958366   6.958865
 1.800000   7.080559   7.081069
 1.812500   7.204804   7.205324
 1.825000   7.331131   7.331662
 1.837500   7.459573   7.460116
 1.850000   7.590162   7.590716
 1.862500   7.722930   7.723496
 1.875000   7.857912   7.858489
 1.887500   7.995139   7.995729
 1.900000   8.134648   8.135250
 1.912500   8.276472   8.277087
 1.925000   8.420648   8.421275
 1.937500   8.567210   8.567850
 1.950000   8.716195   8.716848
 1.962500   8.867640   8.868306
 1.975000   9.021583   9.022263
 1.987500   9.178061   9.178755
 2.000000   9.337113   9.337822
--------------------------------
 Erro relativo: 0.000068
--------------------------------
```

```
 k   Convergencia
------------------
 1   1.953112
 2   1.976958
 3   1.988590
```

4 Conclusão

Observando-se os erros relativos calculados para cada grupo de soluções k, conclui-se que este erro está diminuindo à medida que $h \to 0$ como era previsto.

Isso fica mais evidente se colocarmos num mesmo gráfico a curva da solução analítica e os pontos da solução de Euler Modificado:



5 Referências

Anotações pessoais de aula da disciplina SME0104 Cálculo Numérico ministrada pela professor Murilo Francisco Tomé no 1º semestre de 2014.