Instituto Superior Técnico

Matemática Computacional

3º Quarter - 2021/2022

Licenciatura Em Matemática Aplicada e Computação

Projeto Computacional 3

Abril 2022

Clara Pereira 99405
Marta Sereno 99432
Samuel Pearson 99441

Conteúdo

Exercício 1

1. a)

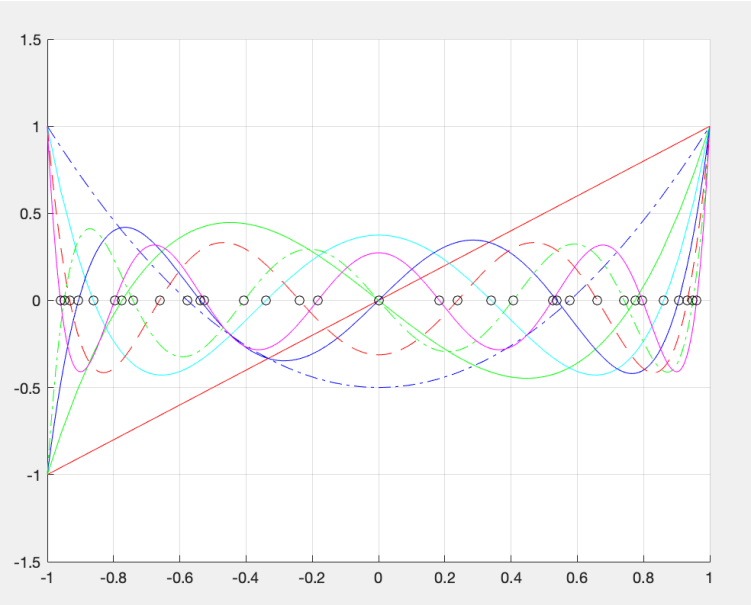
Nesta alínea temos dois objetivos principais.
Em primeiro lugar, é-nos pedido que tracemos os gráficos dos polinómios de Legendre P_k de graus $k = 1, \ldots, 8$, definidos recurssivamente pelas fórmulas

$$P_{n+1}(x) = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x) \, , \; n = 1, 2, \ldots \, , \quad P_0(x) = 1 \, , \; P_1(x) = x \, .$$

Criámos então a função *legendrepol*, que dado um valor de n, recorre a estas fórmulas e devolve o respetivo polinómio de Legendre de grau n.

Por exemplo, depois de simplificado, $legendrepol(3) = \frac{5x^3 - 3x}{2}$

De seguida, recorrendo ao comando *fplot*, para os vários polinómios de graus 1 até 8, foram desenhados os seus gráficos no intervalo $[-1, 1]$. Além disso, através do comando *vpasolve* determinaram-se os zeros de cada polinómio, que são também representados graficamente. Mediante a análise do gráfico, podemos observar que cada polinómio de grau k tem exatamente k zeros reais e distintos em $(-1, 1)$, distribuídos simetricamente em relação à origem.
(Ver *script "ex1apt1".*)



Em segundo lugar, pretendemos verificar que

$$\int_{-1}^{1} P_k(x)P_j(x)\,dx = \begin{cases} \frac{2}{2k+1} & , \quad k = j \\ 0 & , \quad k \neq j \end{cases}$$

Deste modo, determinámos, para cada $k, j \leq 8$ o valor de $\int_{-1}^{1} P_k(x)P_j(x)\,dx$, recorrendo ao comando *integral*, assim como, o designado "valor teórico", isto é, quando $k = j$, este valor será $\frac{2}{2k+1}$, e, caso contrário, será igual a 0. Caso o módulo da diferença entre cada par de valores seja menor que uma tolerância de 10^{-10}, o programa atribuirá o valor *true*, e, caso contrário, o valor *false*. Basta, então, verificar que todos os elementos devolvem *true*, provando o pedido.
(Ver *script "ex1apt2"*.)

1. b)

Nesta alínea, pretendemos criar uma função que receba um inteiro n e devolva os n zeros do polinómio de Legendre $P_n(x)$. Assim, definimos a função *zerospol* que, após verificar que $n \geq 1$ é inteiro, recorre ao comando *vpasolve* para determinar os zeros do polinómio, devolvendo-os numa lista.

Temos, por exemplo, que
$zerospol(3) = \{-0.7745966692, 0, 0.7745966692\}$
$zerospol(4) = \{-0.8611363116, -0.3399810436, 0.3399810436, 0.8611363116\}$

1. c)

Nesta alínea tencionamos implementar um código que devolva os pesos A_j , $j = 0, \ldots, n$, da quadratura $Q_n(f) = \sum_{j=0}^{n} A_j f(x_j)$, através do método dos coeficientes indeterminados.
Assim, dados os nós de integração x_j , $j = 0, \ldots, n$, queremos determinar A_j tais que a quadratura é exata para polinómios de grau o mais elevado possível, ou seja, queremos determinar A_j tais que $\sum_{j=0}^{n} A_j p(x_j) = \int_{-1}^{1} p(x)\,dx$ para todos os polinómios $p(x)$ com o grau mais elevado possível.
Notemos que $\{1, x, x^2, x^3, \ldots\}$ forma uma base dos polinómios.
Queremos, então, que a igualdade acima se verifique para o maior número de potências x^k , $k = 0, 1, 2, \ldots$.

Resulta o seguinte sistema linear

$$\begin{array}{ccccccccc} A_0 & + & \dots & + & A_n & = & \int_{-1}^{1} dx & = & 2 \\ A_0x_0 & + & \dots & + & A_nx_n & = & \int_{-1}^{1} x\,dx & = & 0 \\ A_0x_0^2 & + & \dots & + & A_nx_n^2 & = & \int_{-1}^{1} x^2\,dx & = & \frac{2}{3} \\ \vdots & & \dots & & \vdots & & \vdots & & \vdots \\ A_0x_0^{2n+1} & + & \dots & + & A_nx_n^{2n+1} & = & \int_{-1}^{1} x^n\,dx & = & \frac{(1-(-1)^{n+1})}{n+1} \end{array}$$

O qual pode ser escrito na forma

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ x_0 & x_1 & \dots & x_n \\ x_0^2 & x_1^2 & \dots & x_n^2 \\ \vdots & \vdots & \dots & \vdots \\ x_0^{2n+1} & x_1^{2n+1} & \dots & x_n^{2n+1} \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ A_1 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \frac{2}{3} \\ \vdots \\ \frac{(1-(-1)^{n+1})}{n+1} \end{bmatrix}$$

onde a matriz da esquerda é a transposta da matriz Vandermonde do vetor $[x_0, x_1, \dots, x_n]$.

Deste modo, foi definida a função *pesos*, que recebe um inteiro $n \geq 1$, e começa por calcular a matriz Vandermonde recorrendo aos zeros do polinómio P_{n+1}, determinados pela função *zerospol*. Para além disso, determina um vetor de dimensões $(n+1) \times 1$ com os respetivos valores de $\int_{-1}^{1} x^k\,dx$, $k = 0, \dots, n$. Por fim, resolve o sistema acima, multiplicando a inversa da transposta da matriz Vandermonde pelo vetor determinado, e devolve uma lista com os valores dos pesos obtidos.

Por exemplo,
$pesos(2) = \{0.5555555556, 0.8888888888, 0.5555555556\}$
$pesos(3) = \{0.3478548451, 0.6521451549, 0.6521451549, 0.3478548451\}$

1. d)

Nesta alínea é-nos pedido para implementarmos um programa *Matlab* que recebe um inteiro $n \geq 1$ e uma função f, que devolva um valor aproximado do integral $\int_{-1}^{1} f(x)\,dx$, calculado pela quadratura de Gauss-Legendre $Q_n(f)$.
Para tal, definimos a função *aprox* que, recorrendo às funções definidas anteriormente, calcula os valores da função f nos nós de interpolação, $f(zerospol(n+1))$, e os respetivos pesos, $pesos(n)$. Por fim, como estes valores foram guardados em listas, ambas com dimensões $n \times 1$, basta multiplicar a transposta do primeiro vetor pelo segundo, obtendo-se, assim, a soma pretendida $Q_n(f) = \sum_{j=0}^{n} A_j f(x_j)$, isto é, a aproximação do integral $I(f)$.

Exemplificando,
$aprox(10, @(x)\ x^2) = 0.6666666667$

1. e)

Nesta última alínea, utilizaremos o nosso programa para aproximar os integrais

$$1) \quad \int_{-1}^{1} \frac{x^2}{1+25x^2}\, dx\,, \qquad 2) \quad \int_{0}^{\pi} sin(sin(5x))\, dx\,.$$

Consideramos $n = 1, 2, \ldots, 8$ e notamos que o segundo integral deve ser alterado para o intervalo $[-1, 1]$ recorrendo à mudança de variável dada. Temos, então, $t \in [0, \pi]$, $t = \frac{\pi}{2}(x+1)$, $\frac{dt}{dx} = \frac{\pi}{2}$, com $x \in [-1, 1]$. Ou seja, o integral que devemos aproximar é

$$\int_{-1}^{1} \frac{\pi}{2} sin(sin(\frac{5\pi}{2}(x+1)))\, dx.$$

Por conseguinte, recorrendo à função *aprox* definida na alínea anterior, calculamos os valores de cada integral para cada $n = 1, 2, \ldots, 8$.
De seguida, queremos comparar os valores aproximados com os valores obtidos através do comando *integral*. Prosseguimos, então, ao cálculo dos erros absoluto e relativo de cada aproximação.
Obtivemos os resultados apresentados na tabela abaixo.

	integral 1)	erro relativo 1)	integral 2)	erro relativo 2)
integral	0.0580255877	-	0.3572974964	-
$n = 1$	0.0714285714	0.2309840232	-0.5530201764	2.547786318
$n = 2$	0.04166666667	0.2819259865	2.6245643219	6.345599531
$n = 3$	0.0651629073	0.1230029685	-0.7461795641	3.088398524
$n = 4$	0.0517220832	0.1086331874	0.5414805116	0.5154892409
$n = 5$	0.0615319777	0.0604283398	0.3338664776	0.0655784579
$n = 6$	0.0553551168	0.0460222988	0.3844338917	0.0759490216
$n = 7$	0.0596751233	0.028427727	0.3143350947	0.1202426608
$n = 8$	0.0568518921	0.0202272077	0.3267852077	0.0853974320

Através da análise do erro relativo, podemos concluir que quanto maior o valor de n, ou seja, quanto maior o grau do polinómio, menor será o erro e portanto, melhor será a aproximação segundo a quadratura. É, no entanto, importante notar que, ao contrário do que acontece com o integral 1), relativamente à aproximção do integral 2), esta proporcionalidade não é tão linear.

Exercício 2

2. a)

Para obter a aproximação $\tilde{y}(x)$, criámos a função $a2$, que recebe como argumentos os extremos do integral a e b, o número n para o qual teremos os nodos t_j, com $j = 0, ..., n$, a função $K(x,t)$ e a função $f(x)$, e devolve $\tilde{y}(x)$, usando o método descrito. De seguida, utilizámos os métodos i), ii) e iii), para aproximar o integral $I(K,y)$, utilizando a aproximação $y \approx \tilde{y}$, ou seja, $I(K,y) \approx I(K,\tilde{y})$.

i)

Criou-se a função *comptrapz* que recebe os mesmos argumentos que a função $a2$, e realiza a regra dos Trapézios composta para $I(K,\tilde{y})$ para os $n+1$ nodos no intervalo $[a,b]$ - através da função *trapz* do $Matlab$; devolvendo a função aproximada $I(K,y)(x)$.

ii)

Criou-se a função *simpcomp* que recebe os mesmos argumentos que a função $a2$, e realiza a regra de Simpson composta para $I(K,\tilde{y})$ para os $n+1$ nodos no intervalo $[a,b]$; devolvendo a função aproximada $I(K,y)(x)$.

iii)

Para este método utilizámos o código do exercício 1, nomeadamente a função *aprox*. No entanto, a regra de Gauss-Legendre apenas realiza integrais de extremos -1 e 1, pelo que tivemos de reescrever o integral.

Seja $h(t) = \sqrt{t} \cdot y(t)$ definida no intervalo $[0,1]$ e considere-se $\tilde{h}$, o seu prolongamento par (em t) ao intervalo $[-1,1]$.
Podemos então definir $\tilde{h}$ no $Matlab$ e temos que:

$$2\int_0^1 \frac{\sqrt{x}}{(1+xt^2)^2} h(t)\,dt = \int_{-1}^1 \frac{\sqrt{x}}{(1+xt^2)^2} \tilde{h}(t)\,dt$$

Usando este resultado, definiu-se então a função gl, que recebe como argumento n e devolve a aproximação do integral $I(K,y)$ pela regra de quadratura de Gauss-Legendre.

2. b)

Queremos mostrar que $\int_0^1 \frac{\sqrt{x}t}{(1+xt^2)^2} \cdot \sqrt{t}\, dt = \frac{\sqrt{x}}{2} - \frac{x\sqrt{x}}{2(1+x)} = \frac{\sqrt{x}}{2(1+x)}$.

Vamos então calcular o integral do lado esquerdo da equação.
Efetuando a substituição $1 + xt^2 = s \Leftrightarrow t = \sqrt{\frac{s-1}{x}}$, temos que $\frac{ds}{dt} = 2xt$.
Calculando os novos extremos do intervalo de integração:

$$\sqrt{\frac{s-1}{x}} = 0 \Leftrightarrow s = 1 \text{ e } \sqrt{\frac{s-1}{x}} = 1 \Leftrightarrow s = x + 1$$

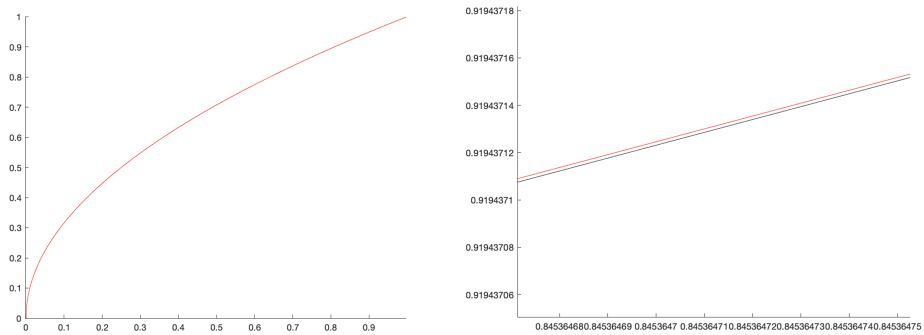
Obtemos então:

$$\int_0^1 \frac{\sqrt{x}t}{(1+xt^2)^2}\, dt = \int_1^{x+1} \frac{\sqrt{x}}{s^2}\frac{1}{2x}\, ds = \frac{1}{2\sqrt{x}} \int_1^{x+1} s^{-2}\, ds = -\frac{1}{2\sqrt{x}} s^{-1}\Big|_1^{x+1}$$

$$= \frac{1}{2\sqrt{x}}(1 - \frac{1}{x+1}) = \frac{1}{2\sqrt{x}}(\frac{x}{x+1}) = \frac{\sqrt{x}}{2(1+x)}$$

Está então mostrado que $y(x) = \sqrt{x}$ satisfaz a equação (4) do enunciado.
De seguida apresenta-se o gráfico da solução exata (a preto) sobreposto ao gráfico da solução aproximada (a vermelho).

Considerou-se também pertinente mostrar um fragmento do gráfico obtido resultante de uma ampliação, de modo a poder distinguir os dois gráficos:



Pela observação dos gráficos, podemos concluir que a aproximação obtida é muito próxima da solução exata, pelo que o método devolve uma boa aproximação.

2. c)

Para construir a tabela, considerámos, para a proximação de $y(x)$, $\tilde{y}(x) = a2(0, 1, K, f, 15)$ e avaliámo-la nos pontos t_j , $j = 0, ..., 15$. Avaliámos também a solução exta $y(x) = \sqrt{x}$, e calculámos o módulo da diferença para o erro. Os valores obtidos apresentam-se em baixo:

t_j	$y(t_j)$	$\tilde{y}(t_j)$	$\lvert y(t_j) - \tilde{y}(t_j)\rvert$
0	0	0	0
0.0667	0.1674	0.2582	$2.0867{\cdot}10^{-10}$
0.1333	0.2325	0.3651	$2.7378{\cdot}10^{-10}$
0.2000	0.2816	0.4472	$3.1250{\cdot}10^{-10}$
0.2667	0.3233	0.5164	$3.3747 \cdot 10^{-10}$
0.3333	0.3608	0.5774	$3.5339 \cdot 10^{-10}$
0.4000	0.3958	0.6325	$3.6225{\cdot}10^{-10}$
0.4667	0.4291	0.6831	$3.6595 \cdot 10^{-10}$
0.5333	0.4611	0.7303	$3.6916{\cdot}10^{-10}$
0.6000	0.4922	0.7746	$3.8232{\cdot}10^{-10}$
0.6667	0.5226	0.8165	$4.2393{\cdot}10^{-10}$
0.7333	0.5523	0.8563	$5.2120{\cdot}10^{-10}$
0.8000	0.5816	0.8944	$7.0855 \cdot 10^{-10}$
0.8667	0.6103	0.9309	$1.0238 \cdot 10^{-10}$
0.9333	0.6387	0.9661	$1.5023 \cdot 10^{-10}$
1.0000	0.6667	1.0000	$2.1702 \cdot 10^{-10}$

Analisando os erros, cuja ordem de grandeza é de $10^{-}10$, temos a confirmação do que tinhamos concluido graficamente: o método utilizado permite obter uma aproximação muito boa.

2. d)

Para calcular a ordem de convergência, notou-se o seguinte: quando n aumenta num fator de k, o erro diminui num fator de k^{ord} onde ord é a ordem de convergência do método.
Assim sendo, implementou-se um código $Matlab$ que devolve uma matriz 2x5 onde as entradas da coluna j correspondem, respetivamente, ao logaritmo na base k do quociente do erro absoluto para n_j pelo erro absoluto obtido para $n_{j+1} = nk$, para o método i e ii.
Consideraram-se os valores de n mencionados no enunciado. (No caso do método i, n aumenta sempre num fator de 2).
Estudando o comportamento assintótico das duas sucessões obtidas, vemos que no caso do método i os valores parecem convergir para 2, enquanto que no caso do método ii parecem convergir para 4. Desta forma, concluímos que a ordem de con-

vergência destes métodos é, respetivamente, quadrática e quártica.
Na tabela abaixo apresentam-se os resultados:

Trapézios	4.1474	2.1369	2.2064	2.0957	-
Simpson	6.7892	5.4256	4.3842	4.3743	4.27965

Exercício 3

3. a)

$$\begin{cases} \begin{bmatrix} \text{y(t)} \\ \text{y'(t)} \\ \text{y"(t)} \end{bmatrix}' = \begin{bmatrix} \text{y'(t)} \\ \text{y"(t)} \\ \text{f(t,y(t),y'(t),y"(t))} \end{bmatrix}, \text{ t>a} \\ \begin{bmatrix} \text{y(a)} \\ \text{y'(a)} \\ \text{y"(a)} \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \end{cases}$$

3. b)

i)

Tendo em conta que $y(t) = (1+t^2)^{-1}$ temos:

- $y'(t) = -\frac{2t}{(1+t^2)^2}$
- $y''(t) = -\frac{2}{(1+t^2)^2} + \frac{8t^2}{(1+t^2)^3} = \frac{-2+6t^2}{(1+t^2)^3}$
- $y'''(t) = \frac{(1+t^2)^2(12t+12t^3+12t-36t^3)}{(1+t^2)^6} = \frac{24t-24t^3}{(1+t^2)^4}$
- $f(t, y(t), y'(t), y''(t)) = \frac{-8t}{(1+t^2)^3} \cdot \frac{-2t}{(1+t^2)^2} - \frac{4t}{(1+t^2)} \cdot \frac{-2+6t^2}{(1+t^2)^3} = \frac{24t-24t^3}{(1+t^2)^4}$
- $y(0) = 1$, $y'(0) = 0$, $y''(0) = -2$

Conclui-se então que $y'''(t) = f(t, y(t), y'(t), y''(t))$ e as condições iniciais são satisfeitas, pelo que $y(t) = (1+t^2)^{-1}$ satisfaz o problema (5).

ii)

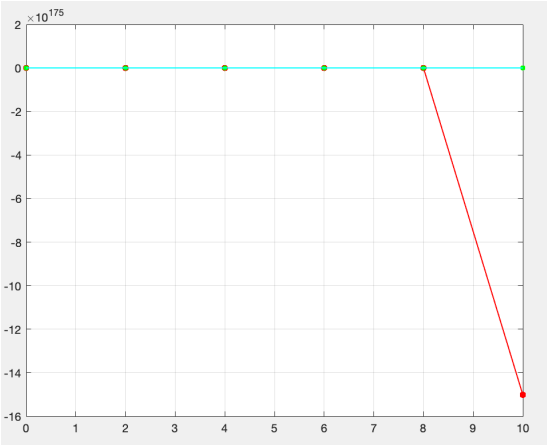
Para esta alínea criou-se a função RK que, como pedido, recebe os parâmetros $f, a, \alpha, \beta, \gamma, n$ e devolve uma aproximação numérica da solução do problema (5) no intervalo $[0, T]$ (mais especificamente, nos $n+1$ pontos de discretização $t_j = jh, j = 0, ..., n$), considerando $T = 10$. A função devolve, portanto, n+1 valores, onde o j-ésimo valor corresponde à aproximação y_j do valor $y(t_j)(j = 0, ..., n)$

3. c)

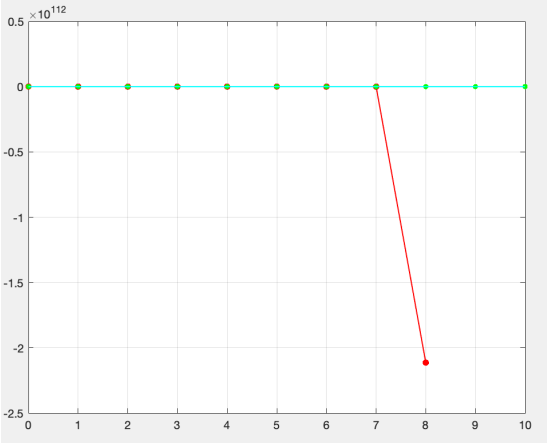
Para esta alínea foram criadas duas funções:

- *grafico*, que recebe como argumento n e devolve o gráfico em que a solução aproximada aparece a vermelho e a solução exata a verde;
- *tabela*, que recebe como argumento n e devolve uma tabela em que as colunas correspondem, respetivamente, aos pontos t_k, aos valores aproximados y_k, aos valores exatos $y(t_k)$ e aos erros $|y(t_k) - y_k|$, com $k = 1, ..., n$.

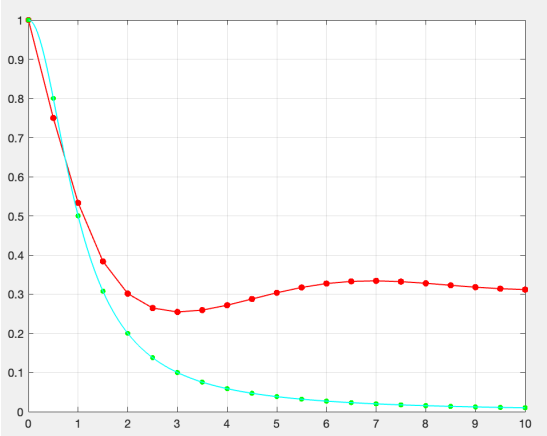
Os resultados obtidos apresentam-se de seguida.
Gráficos:
$n = 5$:



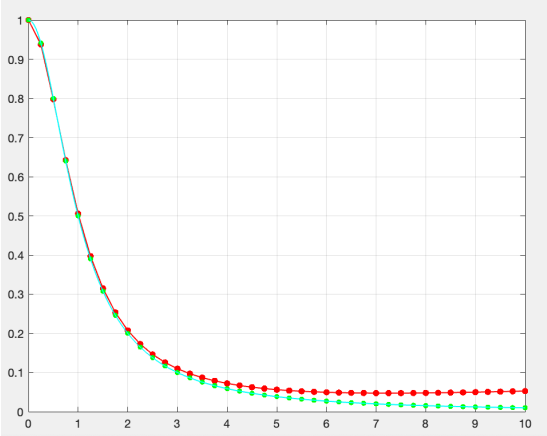
$n = 10$:



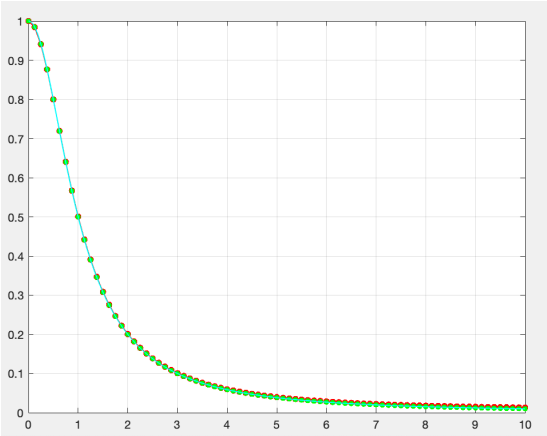
$n = 20$:


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8 9 10

$n = 40$:


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8 9 10

$n = 80$:


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8 9 10

Tabelas:

$n = 5$:

t_k	y_k	$y(t_k)$	$\lvert y(t_k) - y_k \rvert$
2	-3	0.2	-3.2
4	-2003.4	0.058824	2003.5
6	2.6652e+13	0.027027	2.6652e+13
8	2.4141e+49	0.015385	2.4141e+49
10	-1.5014e+176	0.009901	1.5014e+176

$n = 10$:

t_k	y_k	$y(t_k)$	$\lvert y(t_k) - y_k \rvert$
1	0	0.5	0.5
2	1.2883	0.2	1.0883
3	1.8725	0.1	1.7725
4	-1.7356	0.058824	1.7944
5	-87.318	0.038462	87.356
6	3.074e+08	0.027027	3.074e+08
7	4.007e+31	0.02	4.007e+31
8	-2.1137e+112	0.015385	2.1137e+112
9	-Inf	0.012195	Inf
10	NaN	0.009901	NaN

De seguida, apresentam-se apenas 5 linhas das tabelas para $n = 20, 40, 80$, correspondentes a $t_k = 2, 4, 6, 8, 10$, uma vez que estas tabelas teriam dimensões muito grandes para serem apresentadas na sua totalidade.

$n = 20$:

t_k	y_k	$y(t_k)$	$\lvert y(t_k) - y_k \rvert$
2	0.30138	0.2	0.10138
4	0.27186	0.058824	0.21303
6	0.32717	0.027027	0.30014
8	0.32784	0.015385	0.31246
10	0.31175	0.009901	0.30185

$n = 40$:

t_k	y_k	$y(t_k)$	$\lvert y(t_k) - y_k \rvert$
2	0.20738	0.2	0.0073807
4	0.072227	0.058824	0.013403
6	0.049564	0.027027	0.022537
8	0.047966	0.015385	0.032581
10	0.052751	0.009901	0.04285

$n = 80$:

t_k	y_k	$y(t_k)$	$\lvert y(t_k) - y_k \rvert$
2	0.20063	0.2	0.00063409
4	0.059822	0.058824	0.00099885
6	0.02871	0.027027	0.0016826
8	0.017841	0.015385	0.0024559
10	0.013169	0.009901	0.0032684

Podemos observar que para $n = 5, 10$, o método diverge muito rapidamente (repare-se na escala usada nos gráficos). No caso de $n = 10$ o valor de y_9 obtido, calculado pelo $Matlab$, é $-Inf$, uma vez que o cálculo deste valor leva a um resultado demasiado elevado para ser representado pelo sistema de ponto flutuante do Matlab. Deste modo, o $Matlab$ considera que o ponto y_{10} também não toma um valor numérico (NaN), pelo que y_9 e y_{10} não são representados no gráfico.
Notou-se também, usando a função $grafico$, que a partir de $n = 11$, todas as aproximações y_k se mantêm no intervalo $[0, 1]$ (Repare-se na diferença de resultados para $n = 10$ e $n = 11$!).

3. d)

De modo a estudar a convergência do método com os resultados obtidos, criou-se a função *convergencia*, que recebe como argumento um inteiro nit e devolve $nit + 1$ valores, em que o k-ésimo valor corresponde ao quociente do erro absoluto da última iterada do método com $n = 20 \cdot 2^k$ pelo erro absoluto da última iterada do método com $n = 20 \cdot 2^{k+1}$ $(k = 0, ..., nit)$.
Deste modo, podemos estudar experimentalmente o comportamento assintótico da sucessão produzida pela função *convergencia*.
Fazendo $convergencia(9)$, obtemos:

7.0444 13.1102 10.9360 9.6469 8.8921 8.4675 8.2398 8.1215 8.0605 8.0337

Deste modo, observamos que quando n aumenta num fator de 2, o erro tende a diminuir num fator de, aproximadamente, $8 = 2^3$, o que nos leva a concluir que a convergência do método é cúbica.

Bibliografia

[1] Videman, J., Matemática Computacional, (2021-2022), *Apontamentos das Aulas*
[2] Graça, M.M. e P.L. Lima, (2022), *Apontamentos de Matemática Computacional*