Lenguajes de Programación

2° Cuatrimestre 2016

992850 - Agustín Pivetta

Índice

1. Introducción

El objetivo del presente informe consiste en mostrar los algoritmos desarrollados a lo largo del cuatrimestre con sus correspondientes resultados de ejecución. Todos los algoritmos, junto a este informe, se encuentran en `https://github.com/APivetta/metodos-numericos`

2. Aproximación numérica y errores

Se prueban dos métodos para resolver la expresión: $I_n = \int_0^1 \frac{x^n}{x+10} dx$

2.1. Método con mayor error

```
format long
I = [log(11/10)];
for n = 2 : 25
        I(n) = 1/n -10*I(n-1);
endfor
I
```

Resultado:

```
I =

 Columns 1 through 3:

   9.53101798043249e-02  -4.53101798043249e-01   4.86435131376583e+00

 Columns 4 through 6:

  -4.83935131376583e+01   4.84135131376583e+02  -4.84118464709916e+03

 Columns 7 through 9:

   4.84119893281344e+04  -4.84119768281344e+05   4.84119779392456e+06

 Columns 10 through 12:

  -4.84119778392456e+07   4.84119778483365e+08  -4.84119778475031e+09

 Columns 13 through 15:

   4.84119778475801e+10  -4.84119778475729e+11   4.84119778475736e+12

 Columns 16 through 18:

  -4.84119778475735e+13   4.84119778475735e+14  -4.84119778475735e+15

 Columns 19 through 21:
```

```
   4.84119778475735e+16  -4.84119778475735e+17   4.84119778475735e+18

 Columns 22 through 24:

  -4.84119778475735e+19   4.84119778475735e+20  -4.84119778475735e+21

 Column 25:

   4.84119778475735e+22
```

2.2. Método con menor error

```
format long
I=[];
I(24) = 1/(25*11);
for n = 23:-1:1
        I(n) = (1/(n+1) - I(n+1))/10;
endfor
I
```

Resultado:

```
I =

 Columns 1 through 3:

   0.04689820195675140   0.03101798043248600   0.02315352900847329

 Columns 4 through 6:

   0.01846470991526711   0.01535290084732894   0.01313765819337729

 Columns 7 through 9:

   0.01148056092337000   0.01019439076629997   0.00916720344811137

 Columns 10 through 12:

   0.00832796551888631   0.00762943572022779   0.00703897613105545

 Columns 13 through 15:

   0.00653331561252245   0.00609541530334689   0.00571251363319779

 Columns 16 through 18:

   0.00537486366802208   0.00507489273154395   0.00480662824011610

 Columns 19 through 21:

   0.00456529654620742   0.00434703453792584   0.00414870223978920
```

```
Columns 22 through 24:

   0.00396752305665349    0.00380303030303030    0.00363636363636364
```

3. Ceros de funciones

3.1. Punto Fijo

```
function [x,t] = puntoFijo(x0,eps,n,g)
        g = inline(g);
        i = 1;
        x = g(x0);
        t = [x0, x];
        while abs(x-x0) > eps && i < n
                x0 = x;
                x = g(x);
                t = [t,x];
                i = i + 1;
        endwhile
endfunction

cos = puntoFijo(0,0.001,20,"cos(x)")
```

Resultado:

```
cos =  0.73876
```

3.2. Regula Falsi

```
function [x,t] = regulaFalsi(a,b,f,eps,n)
        f = inline(f);
        i = 1;
        t = [];
        while i <= n
                x = b - ((f(b)*(a-b))/(f(a)-f(b)));
                t = [t,x];
                if (abs(f(x)) <= eps)
                        break;
                elseif (f(x)*f(a) < 0)
                        b = x;
                else
                        a = x;
                endif
                i = i + 1;
        endwhile
endfunction

[x1,t] = regulaFalsi(1,2,"x^3 + 4*x^2 - 10",10^-6,20);
[x2,t] = regulaFalsi(0,1,"x - cos(x)",10^-6,20);

[x1,x2]
```

Resultado:

```
ans =

    1.36523    0.73908
```

3.3. Secante

```
function [x,t] = secante(x0,x1,f,eps,n)
        f = inline(f);
        i = 2;
        t = [x0,x1];
        while i <= n
                x = x1 - ((f(x1) * (x0 - x1)) / (f(x0) - f(x1)));
                if (abs(f(x)) < eps)
                        break;
                endif
                x0 = x1;
                x1 = x;
                i = i + 1;
                t = [t,x];
        endwhile
endfunction

[x,t] = secante(0,1,"exp(-x) - x",10^-6,20);
x
```

Resultado:

```
x =  0.56714
```

3.4. Newton Raphson

```
function [x,t] = newtonRaphson(x0,f,f1,eps,n)
        f = inline(f);
        f1 = inline(f1);
        i = 1;
        t = [x0];
        while i <= n
                x = x0 - (f(x0)/f1(x0));
                if (abs(f(x)) < eps)
                        break;
                endif
                x0 = x;
                i = i + 1;
                t = [t,x];
        endwhile
endfunction

[x1,t1] = newtonRaphson(15,"(e/3*x*exp(-x/3))-0.25","e/3*(1-x/3)*exp(-x/3)",
[x2,t2] = newtonRaphson(4,"80*exp(-2*x)+20*exp(-x/2)-7","-160*exp(-2*x)-10*e

[x1,x2]
```

Resultado:

```
ans =

    11.0779    2.3291
```

4. Interpolación

4.1. Lagrange

```
function s = lagrange(x,y,r)
        n = length(x);
        s = 0;
        for k = 1 : n
                p = 1;
                for i = 1 : n
                        if (k != i)
                                p = p * (r - x(i))/(x(k)-x(i));
                        endif
                endfor
                s = s + (p * y(k));
        endfor
endfunction

s = lagrange([0,1,2],[0,1,32],0.5)
```

Resultado:

```
s = -3.2500
```

5. Aproximación de funciones

5.1. Regresion lineal

```
function [a,b,R2] = regresionLineal(x,y)
        z = sum(x.*y);
        t = sum(x);
        q = sum(y);
        w = sum(x.*x);
        n = length(x);

        a = ((n*z)-(t*q))/((n*w)-(t^2));
        b = ((w*q)-(t*z))/((n*w)-(t^2));

        y2 = mean(y);
        v1 = arrayfun(@(xi) (a*xi+b-y2)^2,x);
        v2 = arrayfun(@(yi) (yi-y2)^2,y);

        R2 = sum(v1)/sum(v2);
endfunction
```

```
[a,b,r] = regresionLineal([1,2,3,4,5],[7.14,8.58,7.96,-1.51,-1.37])
[a2,b2,r2] = regresionLineal([1,2,3,4,5,6,7],[0.5,2.5,2.0,4.0,3.5,6.0,5.5])
```

Resultado:

```
a = -2.7110
b =  12.293
r =  0.69607
a2 =  0.83929
b2 =  0.071429
r2 =  0.86832
```

6. Integración numérica

6.1. Trapecios

```
function s = trapecios(n,a,b,f)
        x = linspace(a,b,n+1);
        y = arrayfun(@(xi) (f(xi)),x);
        h = (b-a)/(n);
        s = (f(a) + f(b) + sum(y(2:n))*2 ) * h/2;
endfunction

s1 = trapecios(1,1,2,@(x) (4))
s2 = trapecios(1,1,2,@(x) (x))
s3 = trapecios(4,1,2,@(x) (x^2))
```

Resultado:

```
s1 =  4
s2 =  1.5000
s3 =  2.3438
```

6.2. Romberg

```
function s = romberg(j,a,b,f)
        h = (b-a)/(2^j);
        s = rombergIt(j,h,a,b,f);
endfunction

function s = trapeciosH(h,a,b,f)
        n = (b-a)/h;
        x = linspace(a,b,n+1);
        y = arrayfun(@(xi) (f(xi)),x);
        s = (f(a) + f(b) + sum(y(2:n))*2 ) * h/2;
endfunction

function s = rombergIt(j,h,a,b,f)
        if (j == 0)
                s = trapeciosH(h,a,b,f);
        else
                s = ( 4^j * rombergIt(j-1,h,a,b,f) - rombergIt(j-1,2*h,a,b,f
```

```
        endif
endfunction

x1 = romberg(0,0,4,@(x) (x))
x2 = romberg(1,0,4,@(x) (x^2))
x3 = romberg(2,0,4,@(x) (x^2))
x4 = romberg(3,0,10,@(x) (x^3))
x5 = romberg(3,0,2*pi,@(x) (e^(2-(0.5*sin(x)))));
x5 = x5 * pi/2
```

Resultado:

```
x1 =  8
x2 =  21.333
x3 =  21.333
x4 =  2500
x5 =  77.410
```

6.3. Simpson

```
function s = simpson(n,a,b,f)
        x = linspace(a,b,n+1);
        y = arrayfun(@(xi) (f(xi)),x);
        h = (b-a)/(n);
        s = (f(a) + f(b) + 4*sum(y(2:2:n)) + 2*sum(y(3:2:n))) * h/3;
endfunction

s1 = simpson(2,1,2,@(x) (4))
s2 = simpson(2,1,2,@(x) (x))
s3 = simpson(4,2,4,@(x) (x^2))
```

Resultado:

```
s1 =  4
s2 =  1.5000
s3 =  18.667
```

7. Ecuaciones diferenciales

7.1. Euler

```
function tabla = euler(f,g,a,b,n,y0)
        tabla = [];
        x = a;
        y = y0;
        h = (b-a)/n;
        for i = 1:n
                [x,y] = euler_i(f,x,y,h);
                u(i) = x;
                v(i)  = y;
                tabla = [tabla;i,g(x),y,g(x)-y];
        endfor
```

```
        plot(u,v)
        grid on
        hold on
        fplot(g,[a,b])
        pause()
endfunction

function [x,y] = euler_i(f,x,y,h)
        y = y + h * f(x,y);
        x = x + h;
endfunction

euler(@(x,y) (x+1-y), @(x) (exp(-x)+x),0,0.5,5,1)
```

Resultado:

```
ans =

   1.0000000   1.0048374   1.0000000   0.0048374
   2.0000000   1.0187308   1.0100000   0.0087308
   3.0000000   1.0408182   1.0290000   0.0118182
   4.0000000   1.0703200   1.0561000   0.0142200
   5.0000000   1.1065307   1.0904900   0.0160407
```

7.2. Euler Mejorado

```
function tabla = euler(f,g,a,b,n,y0)
        tabla = [];
        x = a;
        y = y0;
        h = (b-a)/n;
        for i = 1:n
                [x,y] = euler_i(f,x,y,h);
                u(i) = x;
                v(i)  = y;
                tabla = [tabla;i,g(x),y,g(x)-y];
        endfor
        plot(u,v,'r')
        grid on
        hold on
        fplot(g,[a,b])
        pause()
endfunction

function [x,y] = euler_i(f,x,y,h)
        k = f(x,y);
        x = x + h;
        q = f(x,y);
        y = y + h/2 * (k+q);
endfunction
```

```
euler(@(x,y) (x+1-y), @(x) (exp(-x)+x),0,0.5,5,1)
```

Resultado:

```
ans =

    1.0000e+00   1.0048e+00   1.0050e+00  -1.6258e-04
    2.0000e+00   1.0187e+00   1.0195e+00  -7.6925e-04
    3.0000e+00   1.0408e+00   1.0425e+00  -1.7318e-03
    4.0000e+00   1.0703e+00   1.0733e+00  -2.9750e-03
    5.0000e+00   1.1065e+00   1.1110e+00  -4.4348e-03
```

7.3. Runge-Kutta

```
function [x,y] = rungekutta(n,a,b,y0,f)
        h = (b-a)/n;
        x = [a];
        y = [y0];
        for i = 1:n
                k1 = f(x(i),y(i));
                k2 = f(x(i)+h/2,y(i)+k1*h/2);
                k3 = f(x(i)+h/2,y(i)+k2*h/2);
                k4 = f(x(i)+h,y(i)+k3*h);
                x = [x,x(i)+h];
                y = [y,y(i)+(k1+2*k2+2*k3+k4)*h/6];
        endfor
endfunction
```

```
[x,y] = rungekutta(4,0,0.4,1,@(x,y) (x^2-3*y))
```

Resultado:

```
x =

   0.00000   0.10000   0.20000   0.30000   0.40000

y =

   1.00000   0.74115   0.55115   0.41389   0.31744
```

8. Sistemas de ecuaciones

8.1. Gauss

```
function x = gauss(A,b)
        Adiag = descendente(A,b)
        x = ascendente(Adiag);
endfunction

function x = ascendente(A)
        n = rows(A);
        x(n) = A(n,n+1);
```

```
        for i = n-1:-1:1;
                s = 0;
                for j = i+1:n;
                        s += A(i,j)*x(j);
                endfor
                x(i) = A(i,n+1) - s;
        endfor
endfunction

function Aext = descendente(A,b)
        Aext = [A,transpose(b)]
        n = rows(A);
        for p = 1:n-1;
                for j = p+1:n+1;
                        Aext(p,j) = Aext(p,j)/Aext(p,p);
                endfor
                for i = p+1:n;
                        for j = p+1:n+1;
                                Aext(i,j) = Aext(i,j) - Aext(i,p)*Aext(p,j);
                        endfor
                endfor
        endfor
        Aext(n,n+1) = Aext(n,n+1)/Aext(n,n);
endfunction

x = gauss([-2,0,-2;2,2,4;0,1,0],[-10,16,0])
```

Resultado:

```
Aext =

   -2    0   -2  -10
    2    2    4   16
    0    1    0    0

Adiag =

  -2  -0   1   5
   2   2   1   3
   0   1  -1   3

x =

   2   0   3
```