EXAMEN FINAL CÁLCULO NUMÉRICO II

ALUMNO: PAUCAR, SEBASTIÁN. 20190521A.

EJERCICIO 1: MÉTODO DEL DISPARO LINEAL

```
clearvars
clear all
close all
clc
```

Introducimos los datos del problema de valor de frontera:

```
x0 = 0; xf = 2;
alpha = 1; beta = 0;
```

Proponemos una cantida de intervalos, así como su correspondiente longitud de paso:

```
n = 20; h = (xf-x0)/n;
```

El método de disparo lineal se basa en resolver 2 sistemas (reemplaza el PVF por dos PVI), numéricamente. Al combinar soluciones, tendremos lo querido.

SISTEMA 1:

```
w1 = @(x,u) [u(2);u(2)-2*x*u(1)];
```

Por teoría, esta sometido al valor inicial.

```
v0w1 = [alpha; 0];
```

Resolvemos por RK4:

```
[x1SL,w1SL] = RK4(w1,x0,xf,v0w1,h);
```

SISTEMA 2:

```
w2 = @(x,u) [u(2);u(2)-2*x*u(1)];
```

Por teoría, esta sometido al valor inicial.

```
v0w2 = [0; 1];
```

Notese que se tratan de la misma EDO, pero diferentes valores iniciales. Resolvemos por RK4:

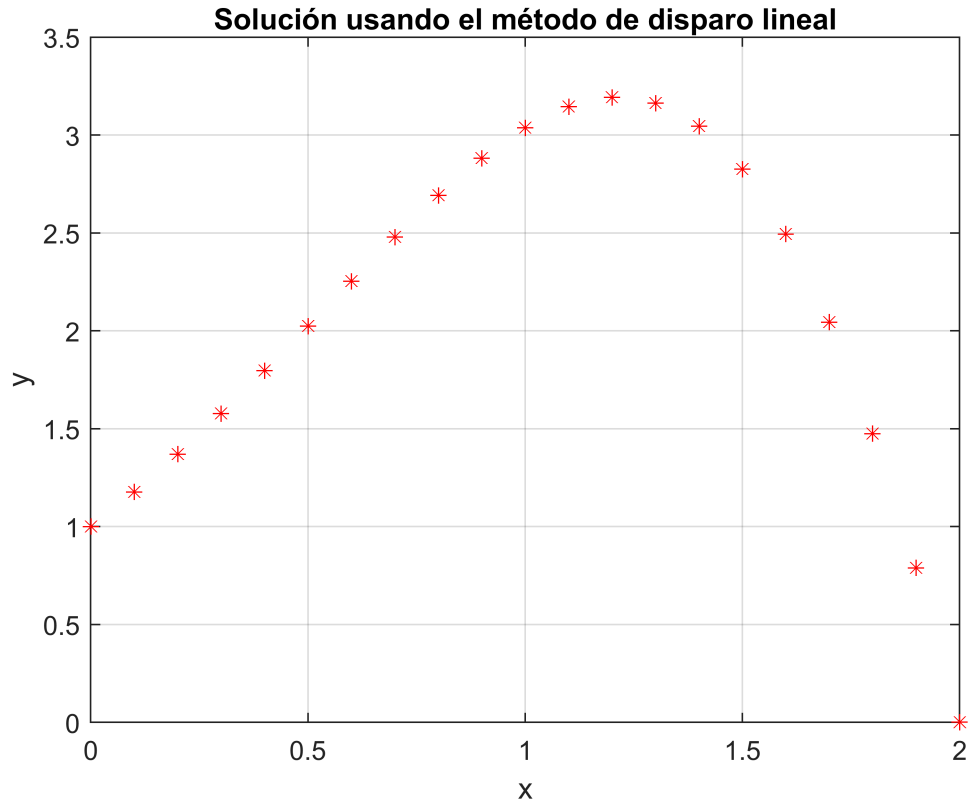
```
[x2SL,w2SL] = RK4(w2,x0,xf,v0w2,h);
```

Combinamos soluciones:

```
y = w1SL(1,:)+((beta-w1SL(1,end))./(w2SL(1,end))).*w2SL(1,:);
x = x1SL;
```

Graficamos la solución obtenida:

```
plot(x,y,'*r')
xlabel('x')
ylabel('y')
grid on
title('Solución usando el método de disparo lineal')
```



EJERCICIO 2: MÉTODO DEL DISPARO NO LINEAL

```
clearvars
clear all
close all
clc
```

Introducimos los datos del problema de valor de frontera:

```
x0 = 0; xf = 0.5;
alpha = 200; beta = 100;
```

Introducimos los parámetros del método iterativo:

```
l = 1; N = 20; M = 10;
```

Donde "l", indica el numero de iteraciones hasta determinar un "tk" adecuado. Tomamos de tolerancia:

```
tol = 10^(-5);
```

El método de disparo NO lineal se basa en resolver 2 sistemas (reemplaza el PVF por dos PVI), numéricamente, donde uno requiere de las soluciones del otro. La solución correspondiente al primer sistema será lo querido.

SISTEMA 1:

Por teoría, nuestro primer sistema sería:

```
WS1 = @(t,w) [w(2);(w(1)+273)^4*10^(-7)+4*(150-w(1))];
```

SISTEMA 2:

```
fT = @(x,T,dT) 4*((T+273)^3*10^(-7)-1);
fdT = @(x,T,dT) 0;
```

Nuestro segundo sistema contiene a las derivadas parciales de la función que define al primer sistema. Este es:

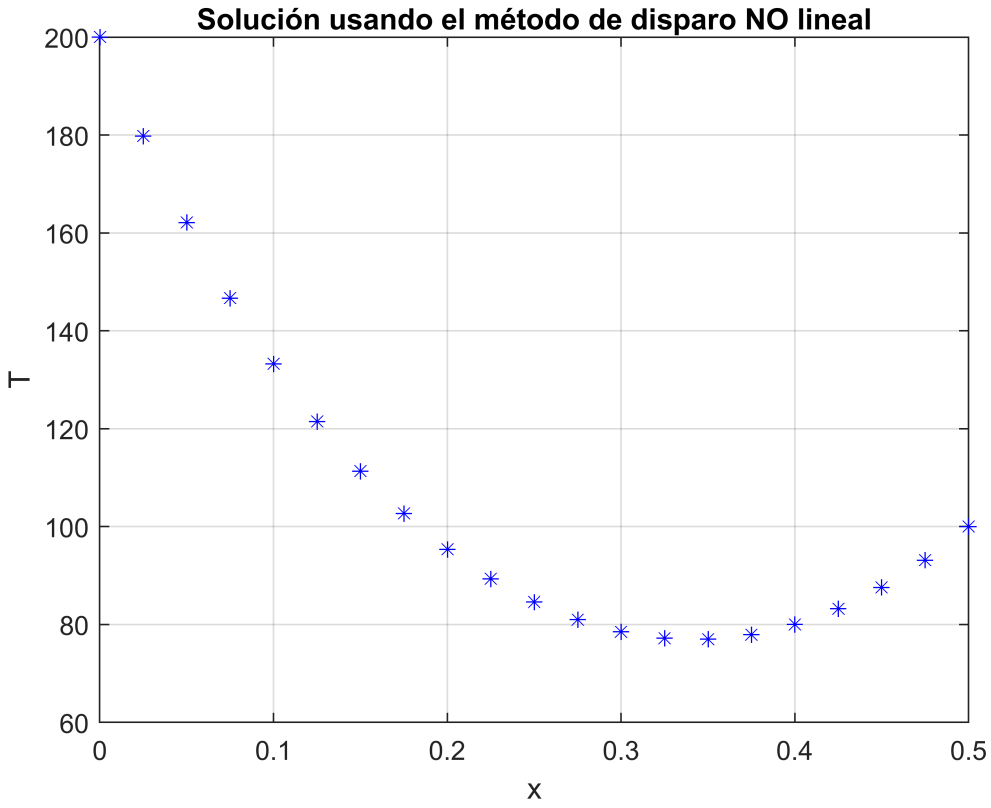
```
WS2 = @(t,w,u) [u(2);fT(t,w(1),w(2))*u(1)+fdT(t,w(1),w(2))*u(2)];
```

Hacemos el cómputo según la teoría de disparo no lineal:

```
[x,solS1] = DisparoNLporNR(x0,xf,beta,alpha,N,M,WS1,WS2,tol);
```

Graficamos la solución:

```
plot(x,solS1(1,:),'b*')
xlabel('x')
ylabel('T')
title('Solución usando el método de disparo NO lineal')
grid on
```



EJERCICIO 3: DIFERENCIAS FINITAS LINEALES

```
clearvars
clear all
close all
clc
```

Los datos del problema son los siguientes:

```
l = 120; Q = 100; E = 3*10^7; S = 1000; I = 625;
```

Identificamos las funciones que definen a la EDO lineal:

```
p = @(x) 0;
q = @(x) -S/(E*I);
r = @(x) Q*x.*(x-l)/(2*E*I);
```

La frontera se define por:

```
x0 = 0; xf = l;
alpha = 0; beta = 0;
```

a)

Piden que hagamos el cómputo con pasos de 6, de ahí que:

```
h = 6;
```

```
x = [x0:h:xf];
```

Como comentamos en el pseudocódigo, debemos resolver un sistema de ecuaciones asociado al tratamiento de las diferencias finitas:

```
[A,b] = DifFinitasEDO(p,q,r,alpha,beta,x,h)
```

```
A = 19×19
   -0.0556    0.0278         0         0         0         0         0         0 ···
    0.0278   -0.0556    0.0278         0         0         0         0         0
         0    0.0278   -0.0556    0.0278         0         0         0         0
         0         0    0.0278   -0.0556    0.0278         0         0         0
         0         0         0    0.0278   -0.0556    0.0278         0         0
         0         0         0         0    0.0278   -0.0556    0.0278         0
         0         0         0         0         0    0.0278   -0.0556    0.0278
         0         0         0         0         0         0    0.0278   -0.0556
         0         0         0         0         0         0         0    0.0278
         0         0         0         0         0         0         0         0
      ⋮
      ⋮
b = 19×1
10^-5 ×
   -0.1824
   -0.3456
   -0.4896
   -0.6144
   -0.7200
   -0.8064
   -0.8736
   -0.9216
   -0.9504
   -0.9600
     ⋮
     ⋮
```

La solución son los puntos interiores de la partición:

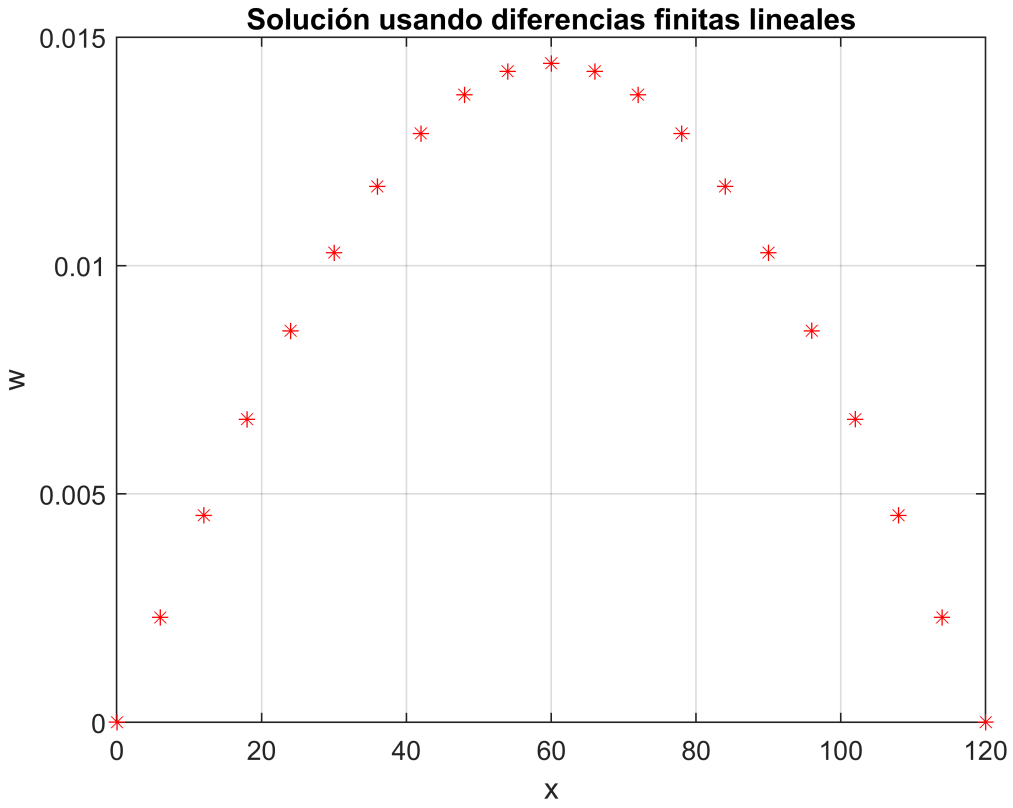
```
wint = A\b;
```

Esto es, la solución completa es:

```
w = [alpha wint' beta];
```

Graficamos la solución numérica:

```
plot(x,w,'*r')
xlabel('x')
ylabel('w')
grid on
title('Solución usando diferencias finitas lineales')
```



b)

Los parámetros que definen a la solución analítica es:

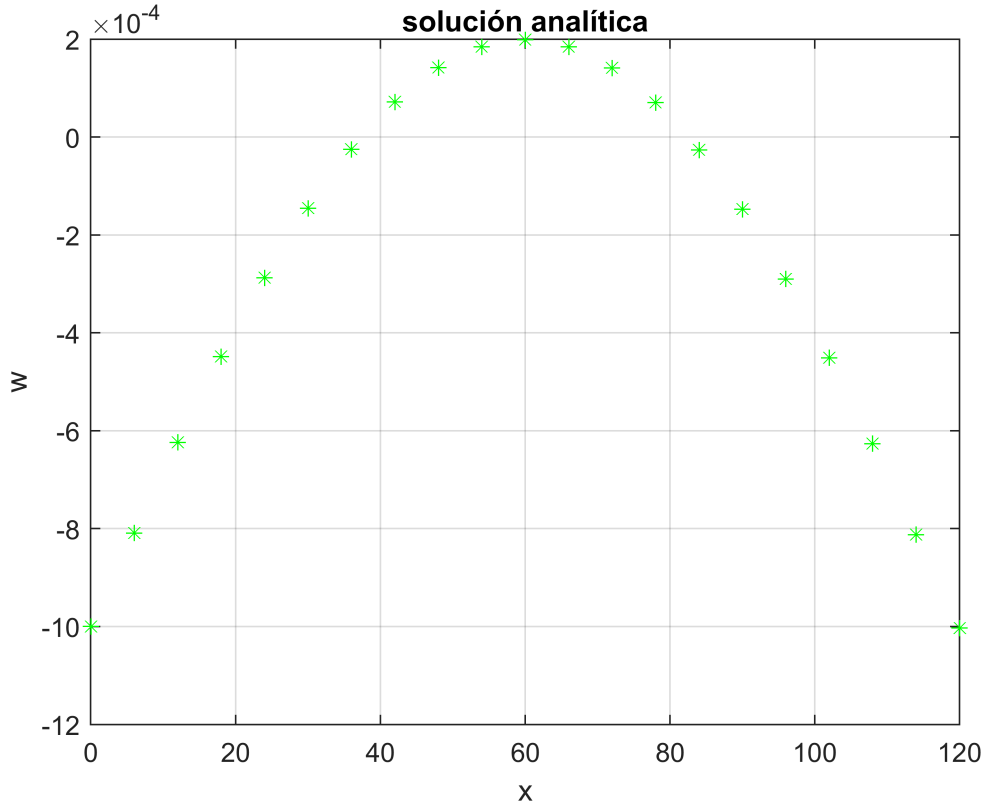
```
c1 = 7.7042537*10^4;
c2 = 7.9207462*10^4;
a = 2.3094010*10^(-4);
b = -4.1666666*10^(-3);
c = -1.5625*10^5;
```

El enunciado del problema muestra:

```
wx = @(x) c1*exp(a*x)+c2*exp(-a*x)+b*x.*(x-l)+c;
```

Graficamos la solución analítica correspondiente:

```
plot(x,wx(x),'g*')
xlabel('x')
ylabel('w')
grid on
title('solución analítica')
```



El error máximo que piden verificar es:

```
ErrMax = max(w-wx(x));
```

El requerimiento del problema es:

```
req = 0.2;
if ErrMax < req
    fprintf('La solución numérica cumple el requerimiento.\n')
else
    fprintf('La solución numérica NO cumple el requerimiento.\n')
end
```

```
La solución numérica cumple el requerimiento.
```

c)

Para este inciso el requerimiento del problema es:

```
req = 1/300;
if max(w) < req
    fprintf('La solución numérica cumple el requerimiento.\n')
else
    fprintf('La solución numérica NO cumple el requerimiento.\n')
end
```

```
La solución numérica NO cumple el requerimiento.
```

```
if max(wx(linspace(0,l))) < req
```

```
    fprintf('La solución analítica cumple el requerimiento.\n')
else
    fprintf('La solución analítica NO cumple el requerimiento.\n')
end
```

```
La solución analítica cumple el requerimiento.
```

EJERCICIO 4: DIFERENCIAS FINITAS LINEALES

```
clearvars
clear all
close all
clc
```

Los datos del problema son los siguientes:

```
Q  = 200; S = 100; D = 8.8*10^7; l = 50;
```

Identificamos las funciones que definen a la EDO lineal:

```
p = @(x) 0;
q = @(x) -S/D;
r = @(x) -Q*l*x/(2*D) + Q*x.^2/(2*D);
```

La frontera se define por:

```
x0 = 0; xf = l;
alpha = 0; beta = 0;
```

Piden tomar la unidad como longitud de paso:

```
h = 1;
x = [x0:h:xf];
```

El sistema de ecuaciones a resolver para hallar la aproximación numérica interior es:

```
[A,b] = DifFinitasEDO(p,q,r,alpha,beta,x,h)
```

```
A = 49×49
   -2.0000    1.0000         0         0         0         0         0         0 ···
    1.0000   -2.0000    1.0000         0         0         0         0         0
         0    1.0000   -2.0000    1.0000         0         0         0         0
         0         0    1.0000   -2.0000    1.0000         0         0         0
         0         0         0    1.0000   -2.0000    1.0000         0         0
         0         0         0         0    1.0000   -2.0000    1.0000         0
         0         0         0         0         0    1.0000   -2.0000    1.0000
         0         0         0         0         0         0    1.0000   -2.0000
         0         0         0         0         0         0         0    1.0000
         0         0         0         0         0         0         0         0
      ⋮
b = 49×1
10^-3 ×
   -0.0557
   -0.1091
   -0.1602
```

```
-0.2091
-0.2557
-0.3000
-0.3420
-0.3818
-0.4193
-0.4545
   ⋮
   ⋮
```

Esto es:

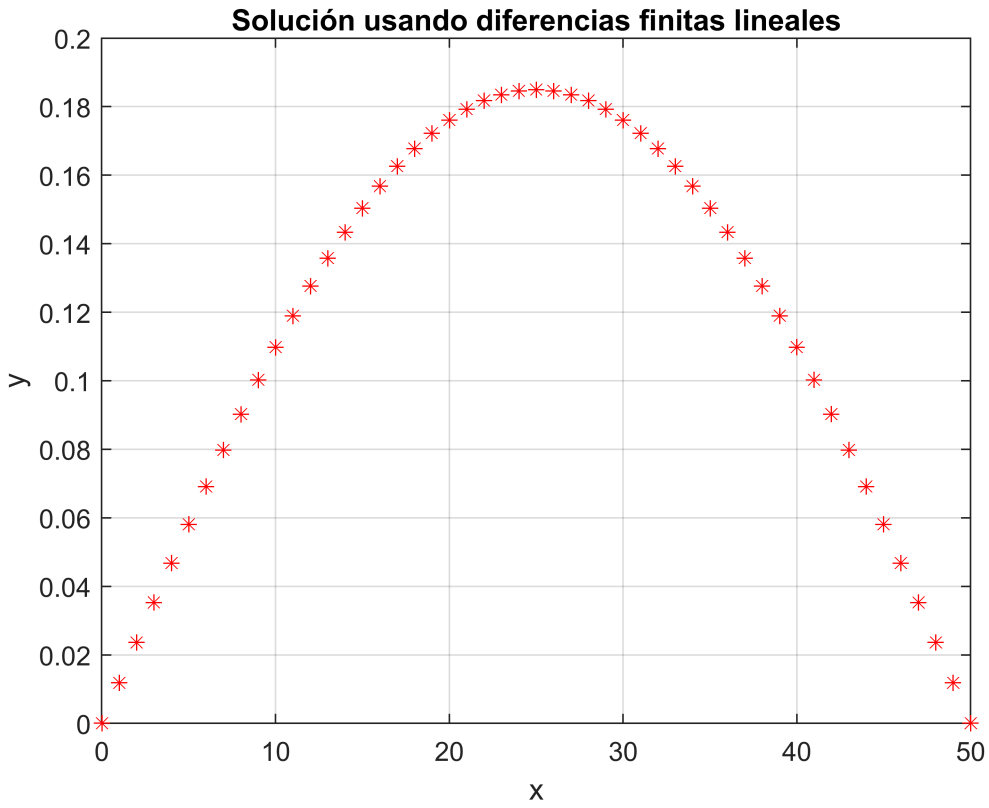
```
wint = A\b;
```

La solución total:

```
w = [alpha wint' beta];
```

Graficamos:

```
plot(x,w,'*r')
xlabel('x')
ylabel('y')
grid on
title('Solución usando diferencias finitas lineales')
```



LAS FUNCIONES USADAS FUERON:

```
function [x,wj] = RK4(w,x0,xf,w0,h)
```

```
    x = [x0:h:xf];
    n = (xf-x0)/h;
    wj(:,1) = w0;
    for i = 1:n
        k1 = w(x(i),wj(:,i));
        k2 = w(x(i)+(h/2),wj(:,i)+(h/2)*k1);
        k3 = w(x(i)+(h/2),wj(:,i)+(h/2)*k2);
        k4 = w(x(i)+h,wj(:,i)+h*k3);
        wj(:,i+1) = wj(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
end
function [x,solS1] = DisparoNLporNR(a,b,bt,ap,N,M,PVIsist1,PVIsist2,tol)
    h = (b-a)/N;
    l = 1;
    tl = (bt-ap)/(b-a);
    w10 = ap;
    while (1)
```

Resolvemos los dos sistemas:

```
        solS1_0 = [w10;tl];
        solS2_0 = [0;1];
        [solS1] = RK4S1(PVIsist1,a,b,solS1_0,h,N);
```

Note que el siguiente sistema usa la solución obtenida del anterior:

```
        [solS2] = RK4S2(PVIsist2,a,b,solS2_0,solS1,h,N);
```

Los criterios de paro son:

```
        if l > M
            break;
        elseif abs(solS1(1,N+1)-bt) < tol
            x = [a:h:b];
            break;
        else
```

Observe que seguimos la lógica de Newton Raphson:

```
            l = l+1;
            tl = tl-((solS1(1,N+1)-bt)/(solS2(1,N+1)));
        end
    end
end
function [A,b] = DifFinitasEDO(p,q,r,alpha,beta,x,h)
    vones = ones(1,length(x(2:end-1)));
    dvones = ones(1,length(x(2:end-2)));
```

La matriz del sistema viene a ser:

```
    dA = (-2/h^2)+vones.*q(x(2:end-1));
    dAI = (1/h^2)-dvones.*(p(x(3:end-1))/(2*h));
    dAF = (1/h^2)+dvones.*(p(x(2:end-2))/(2*h));
```

```
    A = diag(dA)+diag(dAI,-1)+diag(dAF,1);
```

Que se puede identificar de inmediato a partir de la teoría. El vector de términos independientes es:

```
    bI = r(x(2))-((1/h^2)-(p(x(2))/(2*h))).*alpha;
    bF = r(x(end-1))-((1/h^2)+(p(x(end-1))/(2*h))).*beta;
    b = [bI;r(x(3:end-2))';bF];
```

Que también se puede verificar.

```
end
```

Los siguientes métodos RK los hemos usado en el código del disparo NO lineal.

```
function [z] = RK4S1(df,t0,tf,z0,h,n)
    x = [t0:h:tf];
    z(:,1) = z0;
    for i = 1:n
        k1 = df(x(i),z(:,i));
        k2 = df(x(i)+(h/2),z(:,i)+(h/2)*k1);
        k3 = df(x(i)+(h/2),z(:,i)+(h/2)*k2);
        k4 = df(x(i)+h,z(:,i)+h*k3);
        z(:,i+1) = z(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
end
function [z] = RK4S2(df,t0,tf,z0,w,h,n)
    x = [t0:h:tf];
    z(:,1) = z0;
    for i = 1:n
        k1 = df(x(i),w(:,i),z(:,i));
        k2 = df(x(i)+(h/2),w(:,i),z(:,i)+(h/2)*k1);
        k3 = df(x(i)+(h/2),w(:,i),z(:,i)+(h/2)*k2);
        k4 = df(x(i)+h,w(:,i),z(:,i)+h*k3);
        z(:,i+1) = z(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
end
```