Práctica

▼ Unidad 5

▼ Mi intento de Practica Guía 5

▼ Ejercicio2

```
# Ejericicio 2 Nico

close all;
clear all;
clc;

% Datos
x = [3 5 7 9]';
y = [1.2 1.7 2.0 2.1]';

% Obtener coeficientes de los polinomios
[Coef1, L1] = Lagrange(x, y);     % Polinomio de Lagrange
[Coef2, dv] = dif_div(x, y);     % Polinomio de Newton (diferencias divididas)

% Obtener polinomios como strings simbólicos
PL = polyout(Coef1, 'x');  % Lagrange
z = linspace(0, 9, 900);
PN = eval_newton(Coef2, x, z);

% Crear una figura con 3 subgráficos
figure(1)

% Subplot 1: Ambos polinomios
subplot(3,1,1)
plot(x, y, '*k', 'MarkerSize', 8)
hold on
ezplot(PL, [3, 9])
plot(z, PN, 'b-', 'LineWidth', 1);
grid on; grid minor;
legend('Datos', 'Lagrange', 'Newton')
title('Ambos polinomios')

% Subplot 2: Solo Lagrange
subplot(3,1,2)
plot(x, y, '*k', 'MarkerSize', 8)
hold on
ezplot(PL, [3, 9])
grid on; grid minor;
legend('Datos', 'Lagrange')
title('Polinomio de Lagrange')

% Subplot 3: Solo Newton
subplot(3,1,3)
plot(x, y, '*k', 'MarkerSize', 8)
hold on
plot(z, PN, 'b-', 'LineWidth', 1);
grid on; grid minor;
```

```
legend('Datos', 'Newton')
title('Polinomio de Newton')
```

```
#Ejercicio2:

x = [3 5 7 9]';
y = [1.2 1.7 2.0 2.1]';


[P, L] = Lagrange(x,y);
Polinomio_Lagrange = @(x) P(4)*x.^3 + P(3)*x.^2 + P(2)*x + P(1);
[c, dv] = dif_div(x,y);
Polinomio_Newton = @(x) c(4)*x.^3 + c(3)*x.^2 + c(2)*x + c(1);

%???
#Polinomio_CI = @(x)

% Graficar ambos
z = linspace(3, 9, 300);  % puntos para graficar
PL = Polinomio_Lagrange(z);
PN = Polinomio_Newton(z);
%jsdjskd grafique con chatgpt
figure;
plot(x, y, 'or', 'MarkerSize', 8, 'LineWidth', 1.5); hold on;
plot(z, PL, 'b-', 'LineWidth', 2);
plot(z, PN, 'm--', 'LineWidth', 2);
grid on; grid minor;
legend('Datos', 'Pol. Lagrange', 'Pol. Newton');
title('Interpolación polinómica: Lagrange vs Newton');
xlabel('x'); ylabel('P(x)');
```

▼ Ejercicio3

```
# Ejercicio 3 Nico
clear all;
close all;
clc;

# Datos del ejercicio
x = [0 1 3/2 2]';
y = [3 3 13/4 5/3]';

# Obtener coeficientes de Newton
[c, dv] = dif_div(x, y);

# Evaluar el polinomio en puntos z
z = linspace(0, 3, 300);
PN = eval_newton(c, x, z);

# Mostrar el polinomio (opcional: simbólico si querés)
disp('Coeficientes de Newton:');
disp(c);
```

```
# Graficar
figure;
plot(x, y, 'ro', 'MarkerSize', 8, 'LineWidth', 1.5); hold on;
plot(z, PN, 'b-', 'LineWidth', 2);
title('Polinomio Interpolante de Newton');
xlabel('x'); ylabel('P(x)');
grid on; grid minor;
legend('Puntos dados', 'Polinomio interpolante');
```

```
#Ejercicio3
x = [0 1 3/2 2]';
y = [3 3 13/4 5/3]';

[c, dv] = dif_div(x,y);
Polinomio_Newton = @(x) c(4)*x.^3 + c(3)*x.^2 + c(2)*x + c(1);

% Graficar
z = linspace(0, 3, 300);  % puntos para graficar
PN = Polinomio_Newton(z);
figure;
plot(x, y, 'or', 'MarkerSize', 8, 'LineWidth', 1.5); hold on;
plot(z, PN, 'm--', 'LineWidth', 2);
grid on; grid minor;
xlabel('x'); ylabel('P(x)');
```

▼ Ejercicio5

```
#Ejercicio5
x=linspace(-1,1,21);
f_sp=sin(2*pi*x);

%Datos perturbados
f_p=sin(2*pi*x)+(-1).^[1:21]*1e-4;

# Polinomio perturbado:
p=polyfit(x,f_p,20);
xx=linspace(-1,1,200);
z=polyval(p,xx);

#Spline pertubado:
[a,b,c,d] = trazador_cubico_natural(x,f_p);
n = length(a);
x_aux = x;
M = [d c b a];
S = @(x) a(1)*(x == x_aux (1));
for i=1:n
 S = @(x) S(x) + ...
 polyval(M(i,:), (x - x_aux(i))).*(x>x_aux(i)).*(x<=x_aux(i+1)); %%Evalúa el polinomio cúbico corres
pondiente al subintervalo actual en los puntos
 %%polyval([d(i), c(i), b(i), a(i)], xx(idx) - x(i)) calcula ese polinomio en todos los puntos xx que es
tán dentro del intervalo actual.
end

#Graficar Lagrange, Spline y F_sp:
```

```
figure;
plot(xx, sin(2*pi*xx), 'k', 'LineWidth', 2); hold on;          % f_sp exacta
plot(xx, z, 'r--', 'LineWidth', 1.5);                          % Polinomio de grado 20
plot(xx, S_vals, 'b-.', 'LineWidth', 1.5);                     % Spline cúbico
legend('f\_sp (exacta)', 'Polinomio grado 20', 'Spline cúbico', 'Location', 'Best');
title('Comparación: función exacta, polinomio perturbado y spline perturbado');
xlabel('x'); ylabel('f(x)');
grid on;
```

▼ Ejercicio6 (de drive)

```
% Ejercicio 6  - Ej6TP052025.m
clear all; clc; clf;
x1 = [ 1 2 3]';
y1 = [ 0 4 22/3]';

df1=3;
df2=3;

[S,dS,ddS]=funcion_spline(x1,y1,df1,df2);

figure(1)
x=linspace(1,3,51);
plot(x, S(x), 'b-')
hold on
plot(x1,y1,'ko')
legend('S(x)','Ptos.Intervalo','location','north')
title('Ejercicio 6- Spline Cubica Sujeta')
grid on; grid minor
hold off
disp('La Spline Cubica Sujeta será:')
disp(S)
```

▼ Ejercicio7 (de drive)

```
% Ej7Tp052025.m
#brazoRobotico.m

#Primera parte
t1=[0 1 2];
x1=[0 2 6];
y1=[0 4 6];
#Segunda parte
t2=[2 3 4];
x2=[6 3 0];
y2=[6 2 0];

% curva t vs x (tiempo coordenada x)
% PRIMERA ETAPA (Tiempo vs Posicion X) IDA
df1x0=0; df1xn=0;
[Sx1,dSx1,ddSx1]=funcion_spline(t1,x1,df1x0,df1xn);
figure(1)

% GRAFICAMOS LOS PUNTOS DADOS.
plot(t1,x1,'bo', 'MarkerEdgeColor','k',...
```

```
'MarkerFaceColor',[0.49 1 0.63],'MarkerSize',10)
hold on % para graficar el Spline en la misma grafica.
tt1=linspace(0,2,101);
% Grafica de la función Spline Cúbica Natural
plot(tt1,Sx1(tt1),'k-','linewidth',2)
grid on; grid minor;
xlabel('Datos Tiempo')
ylabel('Posicion X')
title('Primera y Segunda Etapa brazo robotico-Tiempo vs Posicion X')

% SEGUNDA ETAPA (Tiempo vs Posicion X) REGRESO
df2x0=0; df2xn=0;
[Sx2,dSx2,ddSx2]=funcion_spline(t2,x2,df2x0,df2xn);

plot(t2,x2,'bo', 'MarkerEdgeColor','k',...
'MarkerFaceColor',[0.49 1 0.63],'MarkerSize',10)
tt2=linspace(2,4,101);
% Grafica de la función Spline Cúbica Natural
plot(tt2,Sx2(tt2),'r-','linewidth',2)
hold off;

%=================================================================
% curva t vs y (tiempo coordenada y)
% PRIMERA ETAPA (Tiempo vs Posicion y) IDA
df1y0=0; df1yn=0;
[Sy1,dSy1,ddSy1]=funcion_spline(t1,y1,df1y0,df1yn);

figure(2)
% GRAFICAMOS LOS PUNTOS DADOS.
plot(t1,y1,'bo', 'MarkerEdgeColor','k',...
'MarkerFaceColor',[0.49 1 0.63],'MarkerSize',10)
hold on % para graficar el Spline en la misma grafica.
tt1=linspace(0,2,101);
% Grafica de la función Spline Cúbica Natural
plot(tt1,Sy1(tt1),'k-','linewidth',2)
grid on; grid minor;
xlabel('Datos Tiempo')
ylabel('Posicion Y')
title('Primera y Segunda Etapa brazo robotico-Tiempo vs Posicion Y')

% SEGUNDA ETAPA (Tiempo vs Posicion X) REGRESO
df2y0=0; df2yn=0;
[Sy2,dSy2,ddSy2]=funcion_spline(t2,y2,df2y0,df2yn);

plot(t2,y2,'bo', 'MarkerEdgeColor','k',...
'MarkerFaceColor',[0.49 1 0.63],'MarkerSize',10)
tt2=linspace(2,4,101);
% Grafica de la función Spline Cúbica Natural
plot(tt2,Sy2(tt2),'r-','linewidth',2)
hold off;
%=================================================================

% Curva X vs Y  TRAYECTORIA DEL BRAZO EN EL PLANO XY
figure(3)
%Graficamos los puntos IDA
```

```
plot(x1,y1,'bo', 'MarkerEdgeColor','k',...
'MarkerFaceColor',[0.49 1 0.63],'MarkerSize',10)
hold on;
plot(x2,y2,'bo', 'MarkerEdgeColor','k',...
'MarkerFaceColor',[0.5 0.7 0.3],'MarkerSize',10)
plot(Sx1(tt1),Sy1(tt1),'k-','linewidth',2)
plot(Sx2(tt2),Sy2(tt2),'r-','linewidth',2)

grid on; grid minor;
xlabel('posicion X');
ylabel('posicion Y');
legend('Puntos Ida', 'Puntos Regreso','Trayectoria Ida', 'Trayectoria Regreso','location','south')
title('Trayectoria del brazo ida y regreso plano XY SPLINE SUJETA')
hold off;
```

▼ Ejercicio8 (de drive)

```
# Ej8TP052025.m

# datos
x = [0 1 2 3 4 5 6]'
y = [432 599 1012 1909 2977 4190 5961]'

xx=linspace(-1,10.5,201);


# interpolamos con lagrange
[PL, L] = Lagrange(x,y);
plag = @(x) polyval(PL,x);

figure(1)
 plot(x,y,'ro', xx,plag(xx),'k')
 grid on
 hold on
 xlabel ('X')
 ylabel('Y')
 legend('datos', 'Lagrange')

# Ajuste de datos
# Lineal
cp = polyfit(x,y,1);
p1 = @(x) polyval(cp,x);

figure(1)
hold on;
grid on; grid minor
plot(xx,p1(xx),'b')

legend('datos', 'Lagrange', 'Lineal')
%=============================================================
# Cuadratico
cp2 = polyfit(x,y,2);
p2 = @(x) polyval(cp2,x);

plot(xx,p2(xx),'g-x')
```

```
legend('datos', 'Lagrange', 'Lineal','Cuadratico')

%============================================================
cp3 = polyfit(x,y,3);
p3 = @(x) polyval(cp3,x);

plot(xx,p3(xx),'r-o')

legend('datos', 'Lagrange', 'Lineal','Cuadratico','p3')
%============================================================
cp4 = polyfit(x,y,4);
p4 = @(x) polyval(cp4,x);

plot(xx,p4(xx),'m-')

legend('datos', 'Lagrange', 'Lineal','Cuadratico','p3', 'Cuartico')

%============================================================
cp5 = polyfit(x,y,5);
p5 = @(x) polyval(cp5,x);

plot(xx,p5(xx),'r-')

legend('datos', 'Lagrange', 'Lineal','Cuadratico','p3', 'Cuartico','p5')

%=====================================================
cp6 = polyfit(x,y,6);
p6 = @(x) polyval(cp6,x);

plot(xx,p6(xx),'c')

legend('datos', 'Lagrange', 'Lineal','Cuadratico','p3','Cuartico','p5','p6')
%=====================================================
# Newton
[PN, c, N] = PolIntNewton(x, y);
pnew = @(x) polyval(PN,x);

plot(xx,pnew(xx),'k')
legend('datos', 'Lagrange', 'Lineal','Cuadratico','p3','Cuartico','p5','p6','Newton')


% valor a las 10 semanas
x10=10;
y10=14900;

plot(x10,y10,'*k',"linewidth",2);
legend('datos', 'Lagrange', 'Lineal','Cuadratico','p3','Cuartico','p5','p6','Newton','y10', 'location','nor
th')

%=====================================================
% CALCULO DE ERRORES
disp('Calculo de Errores')

%=======================================
```

```
ErrLag = norm(y10-plag(10))
ErrNew = norm(y10-pnew(10))

%=======================================
ErrLineal = norm(y10-p1(10))
ErrorRelativo_Lineal = abs((y10-p1(10))/y10)
%Errorlineal= norm(y - p1(x))
%=======================================
Error_y10p2 = norm(y10-p2(10))
ErrorRelativo_p2 = abs((y10-p2(10))/y10)
%ErrorCuadratico= norm(y - p2(x))
%=======================================

Error_y10p3 = norm(y10-p3(10))
ErrorRelativo_p3 = abs((y10-p3(10))/y10)
%ErrorCuadratico= norm(y - p2(x))
%=======================================
Error_y10p4 = norm(y10-p4(10))
ErrorRelativo_p4 = abs((y10-p4(10))/y10)
%ErrorCuartico= norm(y - p4(x))
%=======================================
%=======================================
Error_y10p5 = norm(y10-p5(10))
ErrorRelativo_p5 = abs((y10-p5(10))/y10)
%ErrorCuartico= norm(y - p4(x))
%=======================================
Error_y10p6 = norm(y10-p6(10))
ErrorRelativo_p6 = abs((y10-p6(10))/y10)
%ErrorCuartico= norm(y - p4(x))
```

▼ Ejercicio9 (de drive)

```
#Ej9TP52025.m
% carga de datos
x = [-1:2]'
y = [-1.1 -0.4 -0.9 -2.7]'
% El ejercicio nos dice que transformemos los datos y en -log(y)
disp('Usamos polyfit, para ajustar ')
c = polyfit(x,log(-y),2) # polinomio de ajuste de orden cuadratico.
# y = -exp(ax^2+bx+c); lo ponemos de esta manera para generar un
# polinomio con el exponente: log(-y) = ax^2 + bx+c; de grado 2

disp('Transformamos c en un polonimo como función')
f = @(x) -exp(polyval(c,x));

# lo vemos graficamente:
xx = linspace(-1.5,2.5,101);
plot(x,y,'o',xx,f(xx))
grid on; grid minor
title('Ajuste por minimos cuadrados con una función exponencial Supuesta')
legend('puntos','funcion: y=-e^((ax^2+bx+c))', 'location','south')
xlabel('valores de x')
ylabel('Valores de la funcion exponencial')

# Medimos el error
```

```
disp('medimos el error')
errf = norm(y - f(x)); # error cuadratico usando la funcion norm
disp('Error cuadratico')
disp(errf)

ErrRelat = (f(0) + 0.4)/(-0.4);
# error realtivo en el punto 0
disp('Error relativo en el punto x=0')
disp(ErrRelat)
```

▼ Ejercicio10

```
#Ejer10
t = [4 8 12 16 20 24]
c = [1590 1320 1000 900 650 560]
# c(t) = b * e^(-k*t)
# ln(c) = -k*t + ln(b) = a*t + d ;
# (d = ln(b))→ b = e^d;
# a = -k → k = -a

cL = polyfit(t,log(c),1)

b=e^cL(2)
k = -cL(1)

c = @(t) b * e.^(-k.*t);

xx=linspace(0,50,201);
figure(1)
plot(xx,c(xx),'r',[4 8 12 16 20 24], [1590 1320 1000 900 650 560] ,'*k')

#c(t)-200 = 0?

c200 = @(t) c(t) -200;
[x] = biseccion(c200,0,100,1000,1e-6)
grid on;
hold on;
plot(x,200,'*m')
```

▼ Ejercicio11 (https://www.youtube.com/watch?v=Eog7Z7lFkxI&t=6s min8) **(JJ Dijo que uno de estos puede entrar el jueves)**

```
v = load('datos_velocidades.txt');
t=0:12:300;%min
t=t'/60; %h

# Determine la funcion de la forma v(t) = c1 sen(2t) + c2t^2 + c3 2^t + c4
# que mejor aproxima a los datos en el sentido de cuadrados minimos.

%OJO EL ORDEN!
f1 = @(t) sin(2*t);
f2 = @(t) t.^2;
f3 = @(t) 2.^t;
f4 = @(t) ones(size(t));
```

```
M=[f1(t) f2(t) f3(t) f4(t)];

A=M'*M;
b=M'*v;

disp('Coeficientes');
c=gauss(A,b)

vf= @(t) c(1)*f1(t) + c(2)*f2(t) + c(3)*f3(t) +c(4)*f4(t);

#Velocidad a las 6hs
vf6=vf(6);

#Generar polinomio de grado 6 y evaluarlo
p6=polyfit(t,v,6);
pol6=polyval(p6,6);

#Calcular el error
erfun= nomr(v-vf(t));
erpol=  nomr(v-polyval(p6,t));
```

▼ Unidad 6

▼ Práctica

▼ Ejercicio4

```
% Ej4TP
format long
fa = @(x) sin(pi*x);
fb = @(x) 1./(1+x.^2);
% Soluciones analíticas (evaluando la integral definida);
Ia=2/pi;
Ib=atan(5)-atan(-5);
% Ponemos el vector error en cero
err=[];
% Generamos puntos para la graficación
xa=linspace(0,5,201);
xb=linspace(-5,5,201);
% Calculamos por integracion numérica compuesta, pero con L=1
% es decir un solo intervalo, con diferentes cuadraturas.
for n=2:13
  Q1 = intNCcompuesta(fa,0,5,1,n);
  Q2 = intNCcompuesta(fb,-5,5,1,n);
  err=[err; abs(Ia-Q1) abs(Ib-Q2)];

  % -
  subplot(2,1,1)
  xia = linspace(0,5,n);
  % Graficamos la funcion y el polinomio que la aproxima y calcula
  % por integración numérica.
  plot(xa,fa(xa),xa,polyval(polyfit(xia,fa(xia),n-1),xa))
  grid on, grid minor
  title('Integracion funcion sin(pi*x), Analitica-Numericas,L=1');
  legend('Funcion analitica', 'Polinomio Interpolante gr <= n','location', 'north')
  txt = text(1, 0.5, ''); % crea el objeto de texto en (1, 0.5)
```

```
  set(txt, 'String', sprintf('n = % 3i', n)); % actualiza el n utilizado
  xlabel('ptos x')
  ylabel('f(x) Pn(x)')
 % -
  subplot(2,1,2)
  xib = linspace(-5,5,n);
  plot(xb,fb(xb),xb,polyval(polyfit(xib,fb(xib),n-1),xb))
  title('Integracion funcion: 1/(1+x^2), Analitica-Numericas,L=1');
  grid on, grid minor
  %legend('Funcion analitica', 'Polinomio Interpolante gr <= n')
  txt = text(1, 0.5, ''); % crea el objeto de texto en (1, 0.5)
  set(txt, 'String', sprintf('n = % 3i', n)); % actualiza el n utilizado
  xlabel('ptos x')
  ylabel('f(x) Pn(x)')
  pause(1)
endfor
disp('Error:');
err

disp('Error1:');
E1=err(:,1)
disp('Error2:');
E2=err(:,2)
```

▼ Ejercicio5

```
% Ej5TP062025.m
% Funciones
fa = @(x) sin(pi*x);
fb = @(x) 1./(1+x.^2);
fc = @(x) abs(x).^(3/2);

% Solucion Analita de la Integral
Ia = 2/pi;
Ib = atan(5)-atan(-5);
Ic = 10*sqrt(5);

Ta=[]; Tb=[]; Tc=[];

xa=linspace(0,5,201);
xb=linspace(-5,5,201);
%Mismo procedimiento que ejercicio anterior, ahora son 3 funciones y L = 20.
L=20;
for n=1:12
  QL2=intNCcompuesta(fa,0,5,L,2);
  QL3=intNCcompuesta(fa,0,5,L,3);
  Ea=abs([QL2 QL3]-Ia);
  Ta=[Ta; L QL2 Ea(1) QL3 Ea(2)];
  %================================
  QL2=intNCcompuesta(fb,-5,5,L,2);
  QL3=intNCcompuesta(fb,-5,5,L,3);
  Eb=abs([QL2 QL3]-Ib);
  Tb=[Tb; L QL2 Eb(1) QL3 Eb(2)];
  %================================
  QL2=intNCcompuesta(fc,0,5,L,2);
```

```
  QL3=intNCcompuesta(fb,0,5,L,3);
  Ec=abs([QL2 QL3]-Ic);
  Tc=[Tc; L QL2 Ec(1) QL3 Ec(2)];
  %================================
  L*=2; % Duplico el valor de L
endfor
% visualizamos solo uno de ellos (caso c)
Tc
Ct(:,1) = Ta(1:end-1,3)./Ta(2:end,3);
Cs(:,1) = Ta(1:end-1,5)./Ta(2:end,5);
%================================
Ct(:,1) = Tb(1:end-1,3)./Tb(2:end,3);
Cs(:,1) = Tb(1:end-1,5)./Tb(2:end,5);
%================================
Ct(:,1) = Tc(1:end-1,3)./Tc(2:end,3);
Cs(:,1) = Tc(1:end-1,5)./Tc(2:end,5);
```

▼ Ejercicio7

```
#Ejer7
format long;
x = [0;0.2;0.4;0.6;0.8;1;1.2]; #cm /1000 ordenados
p = [4;3.95;3.89;3.80;3.60;3.41;3.30]; #g/cm^3
A = [100;103;106;110;120;133;149.6]; #cm^2

integrando = p.*A #g/cm

disp('Trapezoidal')
It = trapcomp(x,integrando) #en kg

disp('Simpson')
Is = simpsoncomp(x,integrando) #en kg

% Diferencia relativa
rel_diff = abs(Is - It)/abs(Is)
fprintf('Diferencia relativa porcentual → ', rel_diff*100);
```

▼ Ejercicio9

```
#Ejer9

disp('Con n=2');
n=2;
f = @(x) x.^2.*exp(-x);
a = 0; b = 1; L=1;
Q=cuad_gauss_c(f,a,b,L,n)


disp('Con n=3');
n=3;
f = @(x) x.^2.*exp(-x);
a = 0; b = 1; L=1;
```

```
Q=cuad_gauss_c(f,a,b,L,n)
```

▼ Ejercicio10

```
#Ejer10
format long;
#Datos:
f = @(x) 20*x-((x.^3)/5);
df = @(x) 20-(3/5)*(x.^2);
integrando = @(x) 2*pi*f(x).*sqrt(1+df(x).^2);

#Calculo integrales

disp('Para un solo intervalo: (L=1)');
Integral_cuad_gauss_n3 = cuad_gauss_c(integrando,0,2,1,3)
Integral_simp_1L = intNCcompuesta(integrando,0,2,1,3)

disp('Para 5 subintervalos: (L=5)');
Integral_simp_5L = intNCcompuesta(integrando,0,2,5,3)
Integral_trap_5L = intNCcompuesta(integrando,0,2,5,2)

disp('Según octave:');

Int_gauss_octave = quad(integrando,0,2)
```

▼ Ejercicio12 [PREGUNTAR / NO MISMA FUNCION Q DRIVE DEL PROFE?]

```
#Ejer12

%Funcion
f = @(r, t) 10 + r.^3.*cos(3.*t) +2.*r.^2.*sin(2.*t);

%Intervalo
a = -pi;
b = pi;
c = 0;
d = 1;
%Datos cuad:
n=2;
m=2;
L1=1;
L2=1;

% Calculo de la Energia Total sobre la placa, consideramos rho*c=1
% Función integración multiple
disp('resultado x cuad y newton cotes:');
I1=cuad_gauss_doble(f,a,b,c,d,m,n)
I2 = intNCcompuesta2(f,a,b,c,d,L1,L2,n)

disp('resultado x octave:');
% Funcion de Octave
% integramos sobre un rectangulo de 2×2
Q=quad2d(f,a,b,c,d)
Q1=dblquad(f,a,b,c,d)
```

```
# I1 = 91.98099809591146
# I2 = 119.2197082734794
# Q = 92.10481349440387
# Q1 = 92.10481349440390
```

▼ Unidad 7

▼ Práctica

▼ Ejercicio2

▼ Ejercicio4

▼ Ejercicio7

```
#Ejer7
f = @(t,x) [-t.*x(2);t.*x(1)-t.*x(2)];
t0 = [1;-1];
inter = [0 20];
h = 0.05;
L = 20/h;


[t,y] = euler(f,inter,t0,L)

#[t,y]=rk4(f, inter, t0, L);

x1 = y(:,1);
x2 = y(:,2);
plot(x1,x2)
```

▼ Ejercicio8

```
#Ejer8
f = @(t,x)  [x(1).*(3-0.002.*x(2));(-1).*x(2).*(0.5-0.0006.*x(1))];
y0 = [1600 800];
h = 0.05;
inter = [0 24];
L = 24/h;

#x(1) es la presa y x(2) es el depredador
[t,y]=rk4(f, inter, y0, L)

plot(t,y(:,1),'-r');
hold on;grid on;
plot(t,y(:,2),'-b');
```

▼ Ejercicio9

```
#Ejer9
format long;
# y''' + 4y''+5y'+2y = -4sent(t) - 20cos(t)
#cambio de variables:
#y1 = y
```

```
#y2 = y' = dy1/dt
#y3 = y'' = dy2/dt2
#dy3/dt = y'''
# Entonces:
# dy3/dt + 4y3 + 5y2 + 2y1 =  -4sent(t) - 20cos(t)
# dy3/dt =  -4sent(t) - 2cos(t) -4y3 - 5y2 - 2y1
# y1(0) = y(0) = 1
# y2(0) = y'(0) = 0
# y3(0) = y''(0) = -1

f = @(t, y) [y(2); y(3); -4.*sin(t) - 2.*cos(t) - 4.*y(3) - 5.*y(2) - 2.*y(1)];

y0 = [1; 0; -1];
inter1 = [0, 2.5];
L1 = 10000;
[t1,y1]=rk4(f, inter1, y0, L1);
y1(end,1)
plot(t1, y1(:,1), '-r');
hold on, grid on, grid minor;

inter2 = [0,15];
L2 = 10000;
[t2,y2]=rk4(f, inter2, y0, L2);
plot(t2, y2(:,2), '-b');
hold on, grid on, grid minor;
plot(t2, y2(:,3), '-g');
y2(end,1)
```

▼ Ejercicio10

```
#Ejer10
f =@(t,y) [y(2); -sin(y(1))];
y0 = [3, 0];
y1 = [0, 2];
inter = [0,20];
h = 0.05;
L = 20/h;
[t,y]=rk4(f, inter, y0, L)
y(1,end);
```

▼ Ejercicio11

```
% Ej11TP07
clear all; clc; close all;
format long;

% Ecuación: t^2 * y'' − 2ty' +2y = t^3*ln(t).
% Cambio de variable:
% x1 = y
% x2 = y'
% x3 = y''
% Entonces:
% x1' = x2
% x2' = (t^3*ln(t) + 2t*x2 - 2*x1)t^(-2)
```

```
f = @(t,x) [x(2); (t.^3.*log(t) + 2.*t.*x(2) - 2.*x(1)).*t.^(-2)];

y_exacta = @(t) (7/4)*t + (1/2)*t.^3 .* log(t) - (3/4)*t.^3;

inter = [1 2];
% Condiciones iniciales
Y0 = [1; 0];

% Valores de h a probar
hs = [0.2, 0.1, 0.05];
errores_max = zeros(length(hs), 1);

for i = 1:length(hs)
  h = hs(i);
  L = round((inter(2) - inter(1))/h);

  [t, y] = adams_rashford(f, inter, Y0, L);

  y_real = y_exacta(t);
  errores = abs(y(:,1) - y_real);
  errores_max(i) = max(errores);

  % Graficar
  figure;
  plot(t, y(:,1), 'b', t, y_real, 'r--');
  legend('Aproximación', 'Exacta');
  title(['Solución con h = ', num2str(h)]);
  xlabel('t');
  ylabel('y(t)');
  grid on;
endfor

% Tabla de errores
disp('h       Error máximo')
for i = 1:length(hs)
  fprintf('%.2f     %.5e\n', hs(i), errores_max(i));
end
```

▼ Unidad 8

▼ Práctica

```
clear all; clc; close all;
format long;

% y'' = p(x)y' + q(x)y + r(x) para x en [a,b]
% y(a)=alpha , Ay'(b) + By(b) = C

W1 = 2;
L = 6;
L1 = 4;
D = 0.2;
K = 2.04;
H = 6e-3;
```

```
y0 = 200;
uE = 40;
inter = [0 6];

p = @(x) 1./L-x;
q = @(x) (H./K).*((2./D)+2.*L./W1.*(L-x));
r = @(x) -(H./K).*((2./D)+2.*L./W1.*(L-x)).*uE;

f =@(x) [p(x) q(x) r(x)];
rob = [K H H*uE];
ycd = y0;
h = 0.01;
L = abs(inter(2)-inter(1))./h;

[x,y] = dif_fin_rob(f,inter,ycd,rob,L);

plot(x,y,'-b');
```

▼ Errores

▼ Error cuadrático absoluto

```
% Error cuadrático absoluto
err = abs(valor_real-valor_calculado) / valor_real
```

▼ Error absoluto y relativo

```
% Error absoluto
error_abs = abs(valor_real - valor_medido);
error_abs = norm(A*x-b);

% Error relativo
error_rel = abs(valor_real - valor_medido) / abs(valor_real);
error_rel = norm(x-x0)/norm(x);

% error abs inf:
error_abs_inf =  norm(x-x0, inf);
% norma infinito del error relativo
norm_inf = norm(x - x0, inf) / norm(x, inf);
```