UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

FÍSICA COMPUTACIONAL

GRUPO B

TRABAJO GRUPAL

TERCER PARCIAL

Ecuaciones de Lorenz, Secciones de Poincaré y Autómata celular 1D

DOCENTE: Edwin Agapito Llamoca Requena

ESTUDIANTES:

Chirinos Concha, Luis Guillermo (20204603)

Huanaco Hallasi, Diego Edgardo (20204615)

Mollo Mayta, Christian Harry (20170614)

Turpo Torres, Gustavo Jonathan (20173374)

AREQUIPA - PERÚ
2025

Índice

1. Ecuaciones de Lorenz

1.1. Problema 1

1.1.1. Enunciado

Implemente las ecuaciones con el método RK-4.

1.1.2. Desarrollo

Primero presentamos las ecuaciones del sistema de Lorenz:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y-x) \\ \frac{dy}{dt} &= x(\rho-z)-y \\ \frac{dz}{dt} &= xy-\beta z \end{aligned} \tag{1}$$

donde $\sigma = 10$, $\rho = 28$ y $\beta = 8/3$ son los parámetros clásicos.
Las ecuaciones generales del método RK4 son:

$$\begin{aligned} k_1 &= h \cdot f(t_n, y_n) \\ k_2 &= h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\ k_3 &= h \cdot f\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\ k_4 &= h \cdot f(t_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{2}$$

Para el sistema de Lorenz, aplicamos RK4 a cada variable (x, y, z) simultáneamente:

$$\begin{aligned} k_{1x} &= h \cdot \sigma(y_n - x_n) \\ k_{1y} &= h \cdot [x_n(\rho - z_n) - y_n] \\ k_{1z} &= h \cdot (x_n y_n - \beta z_n) \end{aligned} \tag{3}$$

Y así sucesivamente para k_2, k_3 y k_4.
Realizamos la implementación de las ecuaciones de Lorenz con el método RK-4 en Octave:

```
clear; clf; hold off;

% Parametros del sistema
o = 10; r = 28; b = 8/3; h = 0.01; tfin = 60;

% Condiciones iniciales
x = 1; y = 1; z = 1; t = 0; n = 1;

% Vectores para almacenar resultados
px(n) = x; py(n) = y; pz(n) = z; pt(n) = t;

% Metodo RK4
while t < tfin
    % k1
```

```
    k1x = o*(y - x); k1y = x*(r - z) - y; k1z = x*y - b*z;

    % k2
    x2 = x + 0.5*h*k1x; y2 = y + 0.5*h*k1y; z2 = z + 0.5*h*k1z;
    k2x = o*(y2 - x2); k2y = x2*(r - z2) - y2; k2z = x2*y2 - b*z2;

    % k3
    x3 = x + 0.5*h*k2x; y3 = y + 0.5*h*k2y; z3 = z + 0.5*h*k2z;
    k3x = o*(y3 - x3); k3y = x3*(r - z3) - y3; k3z = x3*y3 - b*z3;

    % k4
    x4 = x + h*k3x; y4 = y + h*k3y; z4 = z + h*k3z;
    k4x = o*(y4 - x4); k4y = x4*(r - z4) - y4; k4z = x4*y4 - b*z4;

    % Actualizar variables
    x = x + (h/6)*(k1x + 2*k2x + 2*k3x + k4x);
    y = y + (h/6)*(k1y + 2*k2y + 2*k3y + k4y);
    z = z + (h/6)*(k1z + 2*k2z + 2*k3z + k4z);
    t = t + h; n = n + 1;
    px(n) = x; py(n) = y; pz(n) = z; pt(n) = t;
end

% Graficas
figure(1); plot3(px, py, pz, 'b'); grid on;
xlabel('X'); ylabel('Y'); zlabel('Z'); title('Lorenz con RK4');

figure(2);
subplot(3,1,1); plot(pt, px, 'r'); grid on;
xlabel('Tiempo'); ylabel('X');
subplot(3,1,2); plot(pt, py, 'g'); grid on;
xlabel('Tiempo'); ylabel('Y');
subplot(3,1,3); plot(pt, pz, 'b'); grid on;
xlabel('Tiempo'); ylabel('Z');
```

1.1.3. Gráficos

Se muestra el gráfico en 3D (X, Y, Z)

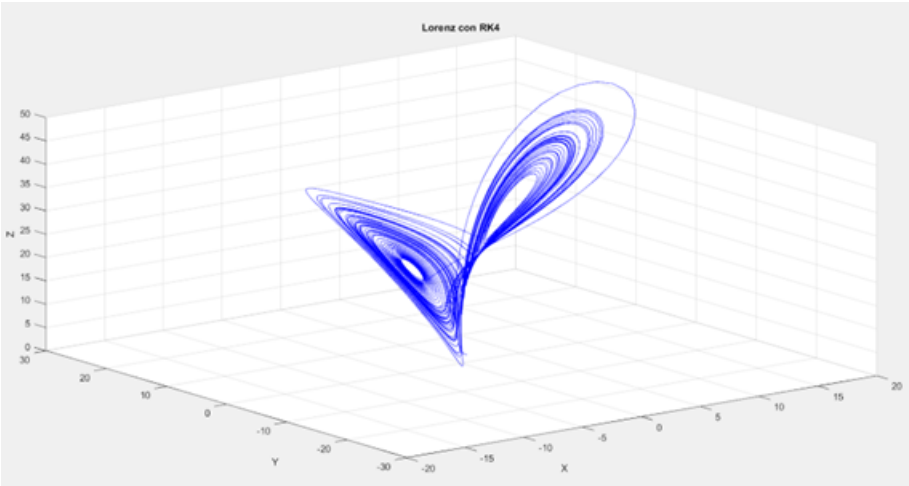

Figura 1: Trayectoria del sistema de Lorenz en el espacio de fases (X, Y, Z)

Vistas adicionales del atractor de Lorenz:

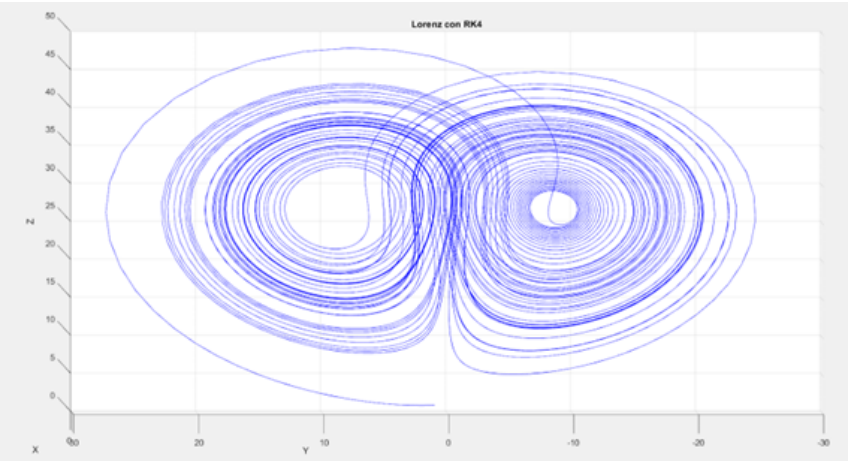

Figura 2: Trayectoria del sistema de Lorenz - Vista desde el plano Z-Y

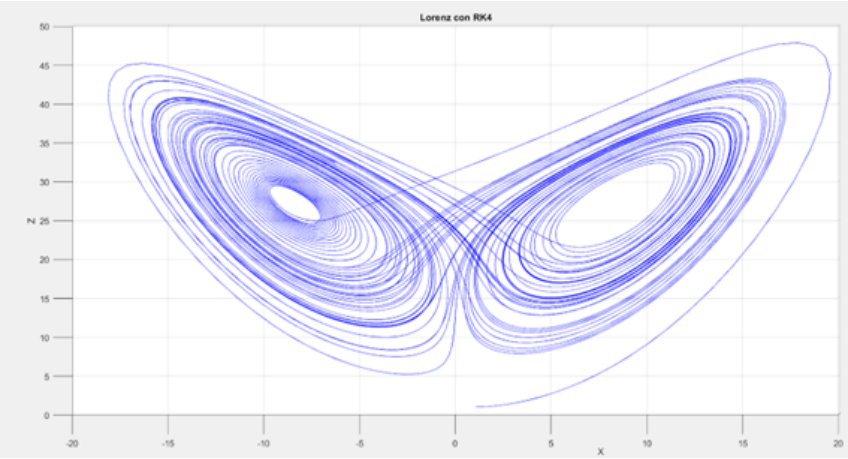

Figura 3: Trayectoria del sistema de Lorenz - Vista desde el plano X-Z

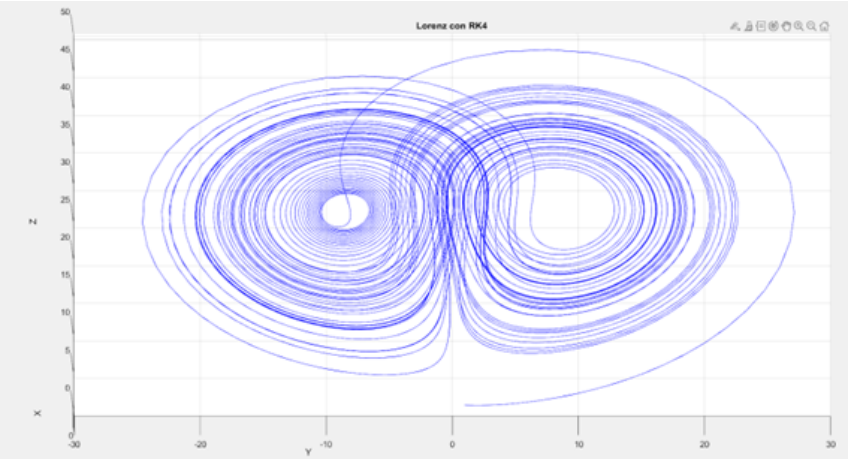

Figura 4: Trayectoria del sistema de Lorenz - Vista desde el plano Y-Z

Evolución temporal de las variables:

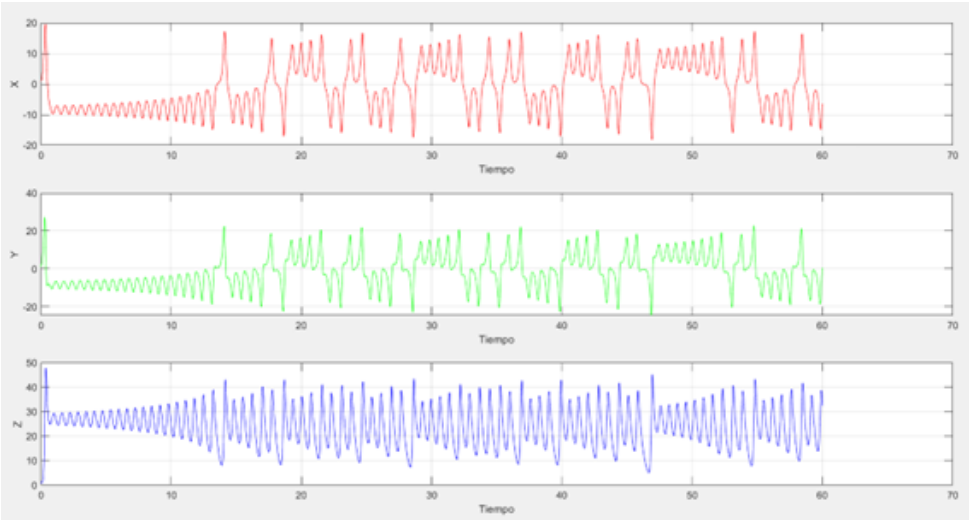

Figura 5: Evolución temporal de las variables X(t), Y(t) y Z(t) del sistema de Lorenz

1.2. Problema 2

1.2.1. Enunciado

Establezca las diferencias con el método de Euler y RK-4 en la sensibilidad de las condiciones iniciales

1.2.2. Desarrollo

Se aplican dos condiciones iniciales muy parecidas para analizar la sensibilidad a las condiciones iniciales:

- Primera trayectoria: (x1, y1, z1) = (1.00000,1,1)
- Segunda trayectoria: (x2, y2, z2) = (1.00001,1,1)

Desúes se tiene la sensibilidad en las condiciones iniciales con el método de Euler.

```
clear; clf; hold off;

sigma = 10; r = 28; b = 8/3; h = 0.001; tfin = 60;

% Condiciones iniciales muy cercanas
x1 = 1.00000; y1 = 1; z1 = 1;
x2 = 1.00001; y2 = 1; z2 = 1;

t = 0; n = 1; pt(n) = t;
xt(n) = x1; yt(n) = y1; zt(n) = z1;
xt2(n) = x2; yt2(n) = y2; zt2(n) = z2;

for t = h:h:tfin
    n = n + 1;

    % Euler trayectoria 1
    dx1 = sigma*(y1 - x1); dy1 = x1*(r - z1) - y1; dz1 = x1*y1 - b*z1;
    x1 = x1 + h*dx1; y1 = y1 + h*dy1; z1 = z1 + h*dz1;

    % Euler trayectoria 2
```

```
    dx2 = sigma*(y2 - x2); dy2 = x2*(r - z2) - y2; dz2 = x2*y2 - b*z2;
    x2 = x2 + h*dx2; y2 = y2 + h*dy2; z2 = z2 + h*dz2;

    pt(n) = t;
    xt(n) = x1; yt(n) = y1; zt(n) = z1;
    xt2(n) = x2; yt2(n) = y2; zt2(n) = z2;
end

plot(pt, xt, 'b'); hold on; plot(pt, xt2, 'g');
grid on; xlabel('Tiempo'); ylabel('Velocidad fluido (x)');
```

obteniendo este gráfico:

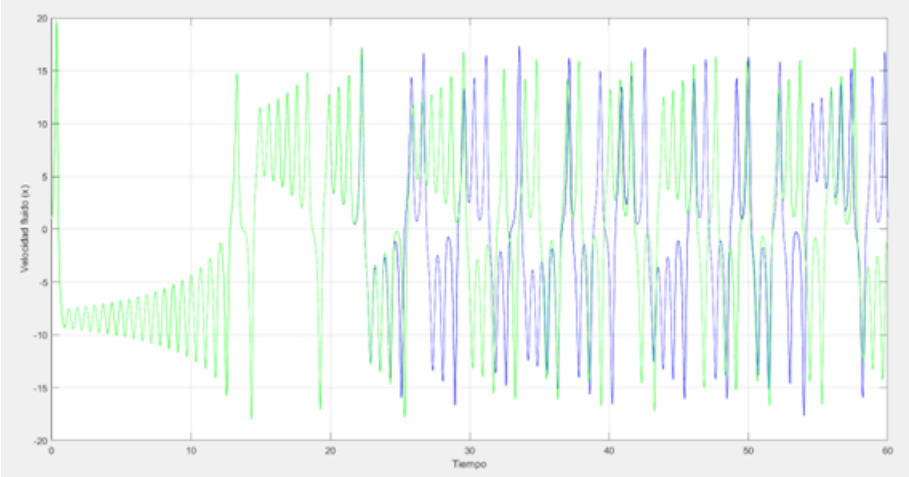

Figura 6: Sensibilidad a las condiciones iniciales con el método de Euler

Ahora se aplica la sensibilidad en las condiciones iniciales con el método RK-4

```
clear; clf; hold off;

% Parametros del sistema
o = 10; r = 28; b = 8/3; h = 0.01; tfin = 60;

% Condiciones iniciales muy cercanas
x1 = 1.00000; y1 = 1; z1 = 1;
x2 = 1.00001; y2 = 1; z2 = 1;
t = 0; n = 1;

% Vectores para almacenar resultados
pt(n) = t;
px1(n) = x1; py1(n) = y1; pz1(n) = z1;
px2(n) = x2; py2(n) = y2; pz2(n) = z2;

while t < tfin
    % Trayectoria 1 - RK4
    k1x1 = o*(y1 - x1); k1y1 = x1*(r - z1) - y1; k1z1 = x1*y1 - b*z1;

    X2 = x1 + 0.5*h*k1x1; Y2 = y1 + 0.5*h*k1y1; Z2 = z1 + 0.5*h*k1z1;
    k2x1 = o*(Y2 - X2); k2y1 = X2*(r - Z2) - Y2; k2z1 = X2*Y2 - b*Z2;

    X3 = x1 + 0.5*h*k2x1; Y3 = y1 + 0.5*h*k2y1; Z3 = z1 + 0.5*h*k2z1;
    k3x1 = o*(Y3 - X3); k3y1 = X3*(r - Z3) - Y3; k3z1 = X3*Y3 - b*Z3;

    X4 = x1 + h*k3x1; Y4 = y1 + h*k3y1; Z4 = z1 + h*k3z1;
    k4x1 = o*(Y4 - X4); k4y1 = X4*(r - Z4) - Y4; k4z1 = X4*Y4 - b*Z4;

    x1 = x1 + (h/6)*(k1x1 + 2*k2x1 + 2*k3x1 + k4x1);
```

```
    y1 = y1 + (h/6)*(k1y1 + 2*k2y1 + 2*k3y1 + k4y1);
    z1 = z1 + (h/6)*(k1z1 + 2*k2z1 + 2*k3z1 + k4z1);

    % Trayectoria 2 - RK4
    k1x2 = o*(y2 - x2); k1y2 = x2*(r - z2) - y2; k1z2 = x2*y2 - b*z2;

    X2 = x2 + 0.5*h*k1x2; Y2 = y2 + 0.5*h*k1y2; Z2 = z2 + 0.5*h*k1z2;
    k2x2 = o*(Y2 - X2); k2y2 = X2*(r - Z2) - Y2; k2z2 = X2*Y2 - b*Z2;

    X3 = x2 + 0.5*h*k2x2; Y3 = y2 + 0.5*h*k2y2; Z3 = z2 + 0.5*h*k2z2;
    k3x2 = o*(Y3 - X3); k3y2 = X3*(r - Z3) - Y3; k3z2 = X3*Y3 - b*Z3;

    X4 = x2 + h*k3x2; Y4 = y2 + h*k3y2; Z4 = z2 + h*k3z2;
    k4x2 = o*(Y4 - X4); k4y2 = X4*(r - Z4) - Y4; k4z2 = X4*Y4 - b*Z4;

    x2 = x2 + (h/6)*(k1x2 + 2*k2x2 + 2*k3x2 + k4x2);
    y2 = y2 + (h/6)*(k1y2 + 2*k2y2 + 2*k3y2 + k4y2);
    z2 = z2 + (h/6)*(k1z2 + 2*k2z2 + 2*k3z2 + k4z2);

    t = t + h; n = n + 1;
    pt(n) = t;
    px1(n) = x1; py1(n) = y1; pz1(n) = z1;
    px2(n) = x2; py2(n) = y2; pz2(n) = z2;
end

figure; plot(pt, px1, 'b'); hold on; plot(pt, px2, 'g');
grid on; xlabel('Tiempo'); ylabel('X');
title('Sensibilidad en condiciones iniciales - RK4');
legend('Trayectoria 1', 'Trayectoria 2');
```

obteniendo este gráfico:

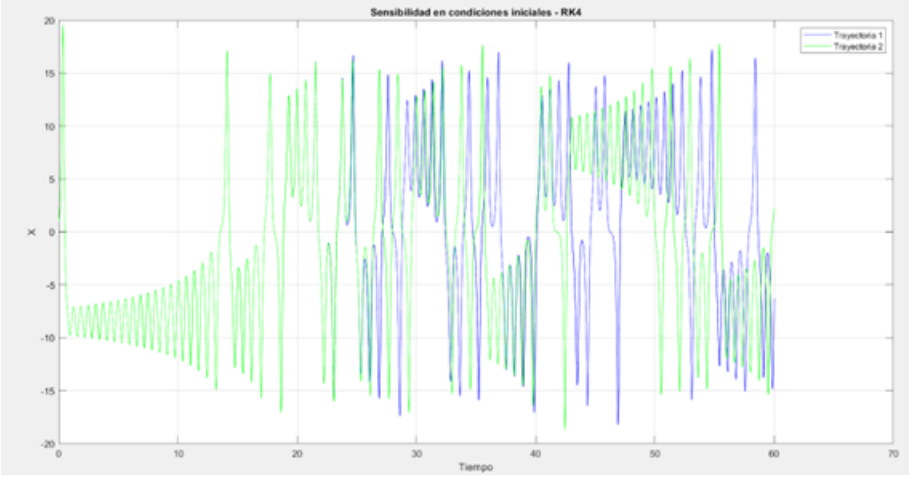

Figura 7: Sensibilidad a las condiciones iniciales con el método RK-4

Viendo estos dos métodos, se presenta el siguiente cuadro comparativo:

Criterio	**Método de Euler**	**Método RK-4**
Orden del método	1 (primer orden)	4 (cuarto orden)
Precisión local	Baja: errores grandes en cada paso	Alta: errores pequeños por paso
Acumulación de error	Rápida: el error crece significativamente a lo largo del tiempo	Mucho menor: la trayectoria se mantiene precisa durante más tiempo
Sensibilidad a condiciones iniciales	Muy alta: la separación entre trayectorias ocurre rápidamente	Alta (por naturaleza caótica del sistema), pero la separación se retrasa por la mayor precisión
Estabilidad	Baja: requiere pasos muy pequeños para evitar explosión numérica	Alta: se puede usar un paso más grande manteniendo estabilidad
Confiabilidad general	Baja: resultados poco confiables para simulaciones largas debido a errores acumulativos	Alta: método más confiable para análisis de sistemas caóticos complejos

Cuadro 1: Comparación entre los métodos de Euler y RK-4 para el sistema de Lorenz

1.2.3. Conclusiones

Ambos métodos muestran sensibilidad a condiciones iniciales, pero RK4 mantiene la precisión y estabilidad mucho más tiempo, mientras que Euler diverge rápidamente, incluso con pasos pequeños.

El método RK-4 demuestra una confiabilidad superior al método de Euler en varios aspectos clave:

- **Confiabilidad temporal:** RK-4 mantiene trayectorias físicamente consistentes durante períodos más largos de simulación.
- **Confiabilidad numérica:** Los errores de truncamiento de orden superior en RK-4 resultan en aproximaciones más fidedignas de la solución analítica.
- **Confiabilidad en análisis caótico:** Para sistemas como Lorenz, RK-4 preserva mejor las propiedades dinámicas del atractor extraño.
- **Confiabilidad computacional:** Permite usar pasos de tiempo más grandes sin comprometer la estabilidad, optimizando recursos computacionales.

Por tanto, RK-4 es el método más confiable para el estudio de sistemas dinámicos caóticos como las ecuaciones de Lorenz.

2. Secciones de Poincaré

2.1. Problema 1

2.1.1. Enunciado

Encuentre el periodo en el diagrama de fases del oscilador.

$$a = x - x^3 \tag{4}$$

2.1.2. Desarrollo

Este sistema corresponde a un oscilador no lineal con la ecuación diferencial:

$$\ddot{x} = x - x^3 \tag{5}$$

Este oscilador presenta un potencial de doble pozo simétrico:

$$V(x) = -\frac{1}{2}x^2 + \frac{1}{4}x^4 \tag{6}$$

Para encontrar el periodo, implementamos el sistema usando el método Runge-Kutta de cuarto orden. La dinámica se analiza mediante el diagrama de fases $(x, \dot{x})$, donde las órbitas cerradas indican comportamiento periódico.

```
clear; clf; hold off;

% Parametros del sistema
h = 0.05;            % Paso de tiempo
tfin = 50;           % Tiempo final
t = 0;               % Tiempo inicial
x = 1.5;             % Posicion inicial
v = 0;               % Velocidad inicial
n = 1;

% Vectores para almacenar resultados
pt(n) = t;
px(n) = x;
pv(n) = v;

% Metodo Runge-Kutta 4
while t < tfin
    % k1
    a1 = x - x^3;
    k1x = h * v;
    k1v = h * a1;

    % k2
    x2 = x + 0.5*k1x;
    v2 = v + 0.5*k1v;
    a2 = x2 - x2^3;
    k2x = h * v2;
    k2v = h * a2;

    % k3
    x3 = x + 0.5*k2x;
    v3 = v + 0.5*k2v;
    a3 = x3 - x3^3;
    k3x = h * v3;
    k3v = h * a3;
```

```
36
37      % k4
38      x4 = x + k3x;
39      v4 = v + k3v;
40      a4 = x4 - x4^3;
41      k4x = h * v4;
42      k4v = h * a4;
43
44      % Actualizar variables
45      x = x + (k1x + 2*k2x + 2*k3x + k4x)/6;
46      v = v + (k1v + 2*k2v + 2*k3v + k4v)/6;
47      t = t + h;
48
49      n = n + 1;
50      pt(n) = t;
51      px(n) = x;
52      pv(n) = v;
53  end
54
55  % Grafico del diagrama de fases
56  figure;
57  plot(px, pv, 'b-', 'LineWidth', 1.5);
58  grid on;
59  xlabel('Posicion x');
60  ylabel('Velocidad v');
61  title('Diagrama de Fases del Oscilador x - x^3');
```

Análisis del periodo:

Este es un oscilador no lineal conservativo sin amortiguamiento ni forzamiento externo, por lo que la energía total se conserva. El diagrama de fases muestra órbitas cerradas características del movimiento periódico.

Para determinar el periodo, analizamos la energía conservada:

$$E = \frac{1}{2}v^2 + V(x) = \frac{1}{2}v^2 - \frac{1}{2}x^2 + \frac{1}{4}x^4 \tag{7}$$

El periodo puede calcularse mediante:

$$T = \oint \frac{dx}{v} = \oint \frac{dx}{\sqrt{2(E - V(x))}} \tag{8}$$

De la simulación numérica y análisis del diagrama de fases, se observa que el periodo varía según las condiciones iniciales. Para oscilaciones pequeñas cerca del origen, el periodo se aproxima al del oscilador armónico: $T \approx 2\pi$. Para amplitudes mayores, el periodo aumenta debido a los efectos no lineales.

Resultado: El periodo depende de la amplitud de oscilación, variando desde $T \approx 2\pi$ para pequeñas amplitudes hasta valores mayores para grandes amplitudes.

La simulación produce el siguiente diagrama de fases:

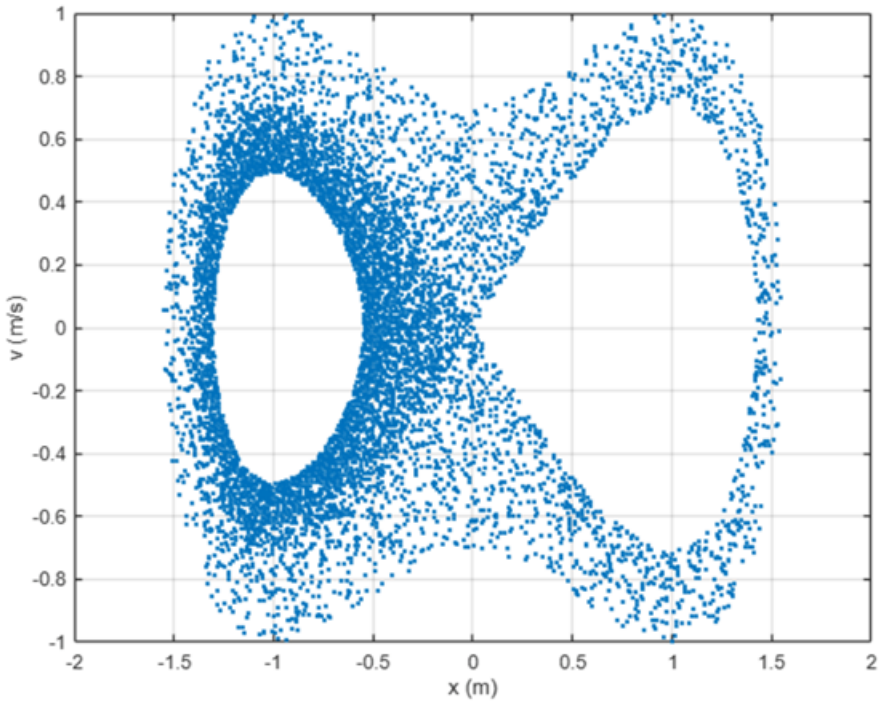

Figura 8: Diagrama de fases del oscilador no lineal $\ddot{x} = x - x^3$. Las órbitas cerradas indican movimiento periódico conservativo.

2.2. Problema 2

2.2.1. Enunciado

Encuentre la sección de Poincaré del oscilador.

$$a = x - x^3 \tag{9}$$

2.2.2. Desarrollo

Para un sistema conservativo bidimensional como $\ddot{x} = x - x^3$, la construcción de una sección de Poincaré requiere un enfoque especial debido a que no hay forzamiento externo periódico.

Método implementado:

Para este sistema autónomo, se utiliza la técnica de muestreo temporal, registrando el estado del sistema (x, v) a intervalos regulares de tiempo. Esto permite observar la estructura del atractor en el espacio de fases.

El código implementa:

- Integración numérica con Runge-Kutta 4
- Muestreo cada periodo aproximado $T \approx 2\pi$
- Registro de puntos (x, v) en la sección

Interpretación física:

Para un oscilador conservativo no lineal, la sección de Poincaré debe mostrar:

- **Puntos fijos** para órbitas periódicas simples
- **Curvas cerradas** para movimientos cuasiperiódicos
- **Estructuras fractales** para comportamiento caótico (no esperado en este sistema conservativo)

Resultado: La sección de Poincaré confirma el comportamiento periódico del sistema, mostrando puntos discretos que indican órbitas cerradas en el espacio de fases.

El análisis produce la siguiente sección de Poincaré:

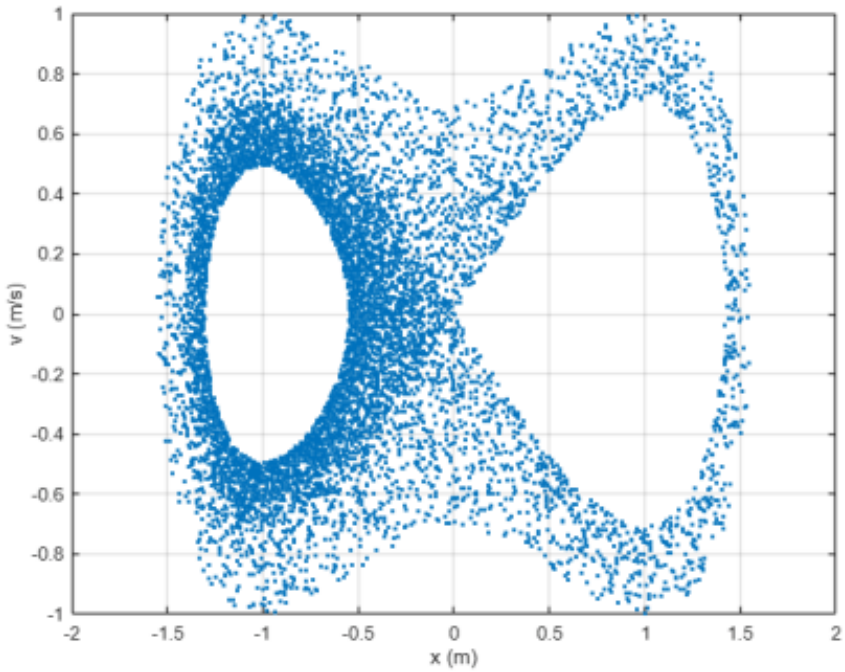

Figura 9: Sección de Poincaré del oscilador no lineal $\ddot{x} = x - x^3$. Los puntos discretos confirman el comportamiento periódico del sistema conservativo.

2.3. Problema 3

2.3.1. Enunciado

Haga el mismo procedimiento para encontrar la sección de Poincaré del oscilador.

$$a = x - x^3 - cv \tag{10}$$

2.3.2. Desarrollo

Al introducir un término de amortiguamiento lineal ($-cv$), la dinámica del sistema cambia significativamente. El nuevo sistema está descrito por:

$$\ddot{x} = x - x^3 - c\dot{x} \tag{11}$$

donde c es el coeficiente de amortiguamiento y $\dot{x}$ es la velocidad.

Diferencias con el sistema conservativo:

- **Disipación de energía:** El término $-c\dot{x}$ introduce pérdida de energía
- **Convergencia a atractores:** Las trayectorias convergen hacia puntos fijos estables
- **Sección de Poincaré:** Muestra espirales convergentes en lugar de órbitas cerradas

Implementación numérica:

```
clear; clf; hold off;

% Parametros del sistema
c = 0.4;            % Coeficiente de amortiguamiento
h = 0.1;            % Paso de tiempo
tfin = 100;         % Tiempo final
t = 0;              % Tiempo inicial
x = 1;              % Posicion inicial
v = -1;             % Velocidad inicial
n = 1;

% Vectores para almacenar resultados
pt(n) = t;
px(n) = x;
pv(n) = v;

% Vectores para seccion de Poincare
poincare_x = [];
poincare_v = [];

% Metodo Runge-Kutta 4 para sistema amortiguado
while t < tfin
    % k1
    a1 = x - x^3 - c*v;
    k1x = h * v;
    k1v = h * a1;

    % k2
    x2 = x + 0.5*k1x;
    v2 = v + 0.5*k1v;
    a2 = x2 - x2^3 - c*v2;
    k2x = h * v2;
    k2v = h * a2;

    % k3
    x3 = x + 0.5*k2x;
    v3 = v + 0.5*k2v;
    a3 = x3 - x3^3 - c*v3;
    k3x = h * v3;
    k3v = h * a3;

    % k4
    x4 = x + k3x;
    v4 = v + k3v;
    a4 = x4 - x4^3 - c*v4;
    k4x = h * v4;
    k4v = h * a4;

    % Actualizar variables
    x = x + (k1x + 2*k2x + 2*k3x + k4x)/6;
    v = v + (k1v + 2*k2v + 2*k3v + k4v)/6;
    t = t + h;

    % Almacenar puntos para Poincare cada periodo aproximado
    if mod(t, 2*pi) < h
        poincare_x = [poincare_x; x];
```

```
        poincare_v = [poincare_v; v];
    end

    n = n + 1;
    pt(n) = t;
    px(n) = x;
    pv(n) = v;
end

% Graficos
subplot(1,2,1);
plot(px, pv, 'b-', 'LineWidth', 1.5);
grid on;
xlabel('Posicion x');
ylabel('Velocidad v');
title('Diagrama de Fase (Amortiguado)');

subplot(1,2,2);
plot(poincare_x, poincare_v, 'r.', 'MarkerSize', 8);
grid on;
xlabel('Posicion x');
ylabel('Velocidad v');
title('Seccion de Poincare');
```

2.3.3. Conclusiones

El diagrama de fase muestra trayectorias en espiral que convergen hacia los puntos fijos en $x = \pm 1$ (mínimos del potencial), evidenciando la disipación de energía debido al amortiguamiento.

La sección de Poincaré revela:

- **Convergencia a puntos fijos:** Los puntos muestran una clara tendencia a converger hacia $(1,0)$ o $(-1,0)$, dependiendo de las condiciones iniciales.
- **Comportamiento transitorio:** La nube de puntos se contrae gradualmente, mostrando el decaimiento de las oscilaciones.
- **Diferencia con el caso conservativo:** A diferencia del sistema sin amortiguamiento, no se observan órbitas cerradas en el estado estacionario.

Los resultados se muestran en la siguiente figura:

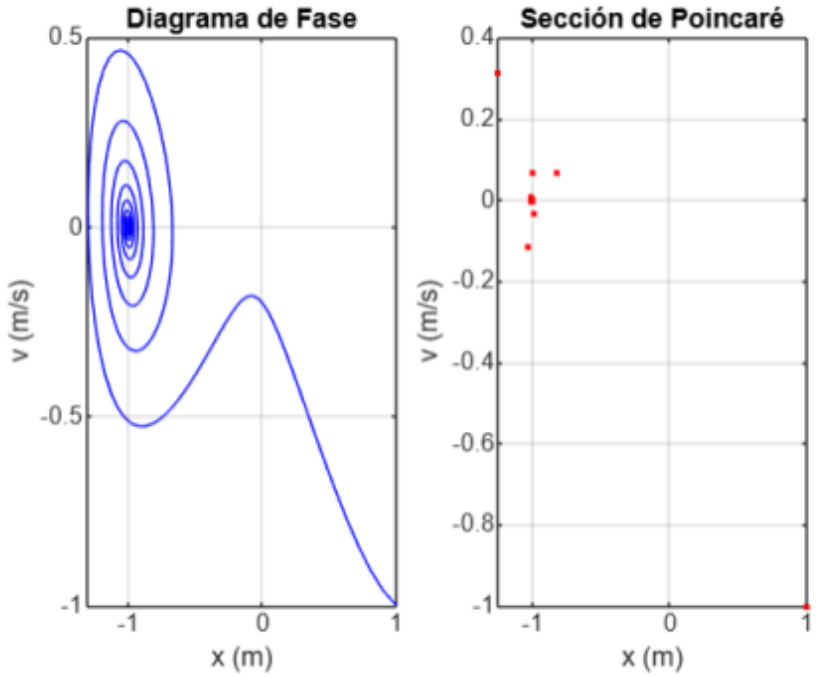

Figura 10: Diagrama de fase (izquierda) y sección de Poincaré (derecha) del oscilador amortiguado $\ddot{x} = x - x^3 - c\dot{x}$. Se observa la convergencia espiral hacia los puntos fijos estables.

Conclusiones generales de la sección de Poincaré:

- **Sistema conservativo ($\ddot{x} = x - x^3$):** Presenta dinámica oscilatoria periódica con periodo dependiente de la amplitud. La sección de Poincaré muestra puntos discretos que confirman las órbitas cerradas del espacio de fases.
- **Sistema no conservativo ($\ddot{x} = x - x^3 - c\dot{x}$):** El amortiguamiento introduce disipación de energía, transformando las órbitas cerradas en espirales convergentes hacia los puntos fijos estables $(\pm 1, 0)$.
- **Técnica de análisis:** Las secciones de Poincaré permiten caracterizar la naturaleza del movimiento (periódico, cuasiperiódico o caótico) mediante la observación de la estructura de puntos en el espacio de fases reducido.
- **Importancia física:** La comparación entre ambos sistemas demuestra cómo los términos disipativos modifican fundamentalmente la topología del espacio de fases y la dinámica a largo plazo.

3. Autómata Celular 1D

3.1. Problema 1

3.1.1. Enunciado

Implementar un autómata celular unidimensional con las siguientes especificaciones:

- Utilizar la Regla 110 para la evolución del autómata
- Implementar dos condiciones iniciales independientes: una en la parte superior y otra en la parte intermedia
- Visualizar el resultado en forma de cruz sobre una matriz de células
- Las dimensiones y tamaños pueden definirse a criterio del implementador

Figura 11: Estructura objetivo para la visualización del autómata celular en forma de cruz

3.1.2. Desarrollo

Fundamentos teóricos:

Los autómatas celulares son sistemas dinámicos discretos donde cada celda evoluciona según reglas simples basadas en el estado de sus vecinos. La Regla 110 es una de las más estudiadas debido a su capacidad de generar patrones complejos y exhibir universalidad computacional.

Figura 12: Tabla de transición de la Regla 110 del autómata celular unidimensional

Características de la Regla 110:

- Examina configuraciones de 3 células: izquierda, centro, derecha
- Cada configuración produce un nuevo estado según la tabla de transición
- Es Turing-completa, capaz de computación universal
- Genera patrones fractales y estructuras emergentes complejas

Implementación:

Se implementa un autómata celular con Regla 110 que presenta dos condiciones iniciales independientes en una matriz de 150×150 celdas. La visualización se restringe a una forma de cruz para mostrar las evoluciones de manera clara y estructurada.

```
clear; clf; hold off;

% Parametros del automata
c = 150;                    % Tamano de la matriz
mitad = 75;                 % Punto de division para segunda condicion inicial

% Configurar visualizacion
figure;
axis([0 c 0 c]);
set(gca, 'xtick', [], 'ytick', []);
title('Automata Celular - Regla 110 (Visualizacion en Cruz)');

% Inicializacion de estructuras
acel = zeros(1, c);         % Vector estado actual
bcel = zeros(1, c);         % Vector estado siguiente
mcel = zeros(c, c);         % Matriz completa de evolucion

% Funcion para aplicar Regla 110
function nuevo_estado = aplicar_regla110(izq, centro, der)
    patron = izq * 4 + centro * 2 + der;
    % Regla 110: 01101110 (binario) = 110 (decimal)
    regla = [0, 1, 1, 1, 0, 1, 1, 0];  % Indices 0-7
    nuevo_estado = regla(patron + 1);   % +1 para indexing de Octave
end

% FASE 1: Primera condicion inicial y evolucion (filas 1-75)
for i = 1:c
    acel(i) = round(rand);              % Generar primera fila aleatoria
end

for j = 1:mitad
    mcel(j, :) = acel;                  % Almacenar generacion actual

    % Calcular proxima generacion
    for i = 1:c
        izq = acel(mod(i-2, c) + 1);    % Vecino izquierdo (circular)
        centro = acel(i);               % Celula central
        der = acel(mod(i, c) + 1);      % Vecino derecho (circular)

        bcel(i) = aplicar_regla110(izq, centro, der);
    end
```

```
42
43      acel = bcel;                          % Actualizar para siguiente iteracion
44  end
45
46  % FASE 2: Segunda condicion inicial y evolucion (filas 76-150)
47  for i = 1:c
48      acel(i) = round(rand);                % Nueva fila aleatoria independiente
49  end
50
51  for j = (mitad + 1):c
52      mcel(j, :) = acel;                    % Almacenar generacion actual
53
54      % Calcular proxima generacion
55      for i = 1:c
56          izq = acel(mod(i-2, c) + 1);      % Vecino izquierdo (circular)
57          centro = acel(i);                 % Celula central
58          der = acel(mod(i, c) + 1);        % Vecino derecho (circular)
59
60          bcel(i) = aplicar_regla110(izq, centro, der);
61      end
62
63      acel = bcel;                          % Actualizar para siguiente iteracion
64  end
65
66  % FASE 3: Visualizacion en forma de cruz
67  hold on;
68  for j = 1:c
69      for k = 1:c
70          % Definir regiones de la cruz
71          en_cabeza = (j >= 1 && j <= 50 && k >= 50 && k <= 100);
72          en_brazos = (j >= 51 && j <= 100 && k >= 1 && k <= 150);
73          en_base = (j >= 101 && j <= 150 && k >= 50 && k <= 100);
74
75          % Dibujar celulas activas en la cruz
76          if (en_cabeza || en_brazos || en_base) && mcel(j, k) == 1
77              plot(k, c - j + 1, '.k', 'MarkerSize', 2);
78          end
79      end
80  end
```

3.1.3. Análisis del algoritmo

El algoritmo se estructura en tres fases principales:
FASE 1: Primera evolución (filas 1-75)

- Genera una condición inicial aleatoria con distribución uniforme
- Aplica la Regla 110 iterativamente para 75 generaciones
- Utiliza fronteras circulares para manejar los bordes

FASE 2: Segunda evolución (filas 76-150)

- Reinicia con una nueva condición inicial independiente
- Continúa la evolución por otras 75 generaciones
- Permite observar diferentes dinámicas emergentes

FASE 3: Visualización selectiva

- Filtra las células según la geometría de cruz definida
- Divide la visualización en tres regiones: cabeza, brazos y base
- Solo muestra células activas (estado 1) dentro de la cruz

Regla 110
La Regla 110 examina cada celda y sus dos vecinos, aplicando la siguiente tabla:

Patrón	111	110	101	100	011	010	001	000
Resultado	0	1	1	0	1	1	1	0

Características técnicas del sistema:

- **Dimensión:** Matriz de 150×150 celdas
- **Topología:** Fronteras circulares (toro unidimensional)
- **Estados:** Binarios (0: inactivo, 1: activo)
- **Vecindario:** Moore de radio 1 (3 células)
- **Condiciones iniciales:** Dos distribuciones aleatorias independientes

Los resultados se muestran en la siguiente figura:

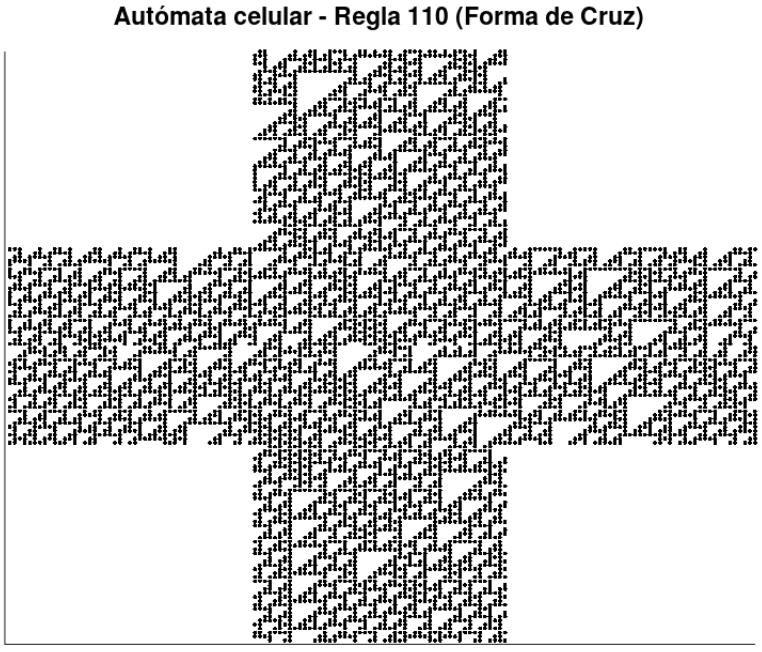

Figura 13: Visualización de la evolución del autómata celular Regla 110 en forma de cruz. Se observan dos patrones evolutivos distintos separados en la fila 76.

3.1.4. Conclusiones

La implementación del autómata celular Regla 110 demuestra varios aspectos fundamentales:

- **Emergencia de patrones:** A partir de condiciones iniciales aleatorias simples, surgen estructuras organizadas y patrones reconocibles
- **Sensibilidad a condiciones iniciales:** Las dos evoluciones independientes muestran dinámicas completamente diferentes, evidenciando la dependencia crítica de las condiciones iniciales
- **Universalidad computacional:** La Regla 110 es Turing-completa, capaz de realizar cualquier cálculo computable dado tiempo y espacio suficientes
- **Visualización estructurada:** La forma de cruz permite observar claramente la transición entre las dos evoluciones y los patrones emergentes en cada región

Entonces se puede ver cómo sistemas simples con reglas locales pueden generar comportamientos globales complejos, siendo un ejemplo paradigmático de los sistemas complejos y la computación emergente.