ACTIVIDAD 6

Nombre: Alexander Fermin Dario Chicalla Garcia

Código: 2024-119042

1. Mostrar resultados (solución) por cada ejercicio de la Práctica 6 (captura de pantalla).
2. Genere tabla de iteraciones de cada ejercicio.
3. Realice 2 pseudocódigos (Punto Fijo y Newton-Raphson Modificado)
4. Pegar código generado en Python de todos los ejercicios (captura de pantalla).

subir la ACTIVIDAD 6, debidamente terminado, hora límite hasta las 00.00 horas del 01OCT2025

ATTE. El Docente.

A- Resultados de cada ejercicio de la practica 6

```
===== PUNTO FIJO =====
PF1: cos/sen: solución = [0.76816916 0.69481969],

PF2: sqrt-symmetric: solución = [1. 1.],
 PF3: sen/cos scaled: solución = [0.8668091  0.82363108],
PF4: 3-variable: solución = [0.98509902 0.61110894 0.40050071],
===== NEWTON-RAPHSON MODIFICADO =====
NRM1: circle+exp: error -> math range error
NRM2: 3-var: error -> math range error
NRM3: symmetric quad: solución = [1.61803399 1.61803399],

NRM4: sin/cos mix: solución = [-0.02181769  1.02181596],
```

Resultados 9 y 10

```
 PF9: 3-variable (cos-sin-mix): solución = [0.66491439 0.40424794 0.25375805],
PF10: 3-variable (sqrt-symmetric): solución = [0.80901699 0.80901699 0.80901699
```

B- Tabla de iteraciones de cada ejercicio
Para los ejercicios solo se muestran las ultimas 5 iteraciones por cada ejercicio

Punto fijo

EJERCICIO 1

iter	x0	x1	||x_{k+1}-x_k||_inf
54	0.768169	0.69482	4.559227e-10
55	0.768169	0.69482	2.919032e-10
56	0.768169	0.69482	2.099321e-10
57	0.768169	0.69482	1.344084e-10
58	0.768169	0.69482	9.666445e-11

EJERCICIO 2

iter	x0	x1	||x_{k+1}-x_k||_inf
13	1.0	1.0	2.451143e-08
14	1.0	1.0	6.127857e-09
15	1.0	1.0	1.531964e-09
16	1.0	1.0	3.829911e-10
17	1.0	1.0	9.574774e-11

EJERCICIO 3

iter	x0	x1	||x_{k+1}-x_k||_inf
20	0.866809	0.823631	2.463377e-09
21	0.866809	0.823631	9.388759e-10
22	0.866809	0.823631	3.190122e-10
23	0.866809	0.823631	1.215863e-10
24	0.866809	0.823631	4.131262e-11

EJERCICIO 4

iter	x0	x1	x2	||x_{k+1}-x_k||_inf
10	0.985099	0.611109	0.400501	2.053883e-07
11	0.985099	0.611109	0.400501	1.992409e-08
12	0.985099	0.611109	0.400501	4.249789e-09
13	0.985099	0.611109	0.400501	1.092723e-09
14	0.985099	0.611109	0.400501	9.194223e-11

NEWTON-RAPHSON MODIFICADO

EJERCICIO 1

```
NRM1: circle+exp: error -> math range error
```

EJERCICIO 2

```
NRM2: 3-var: error -> math range error
```

EJERCICIO 3

NRM3: symmetric quad: solución = [1.61803399 1.61803399], iteraciones = 12, error final

	iter	x0	x1	delta0	delta1	||delta||_inf	||f(x)||_2
7	8	1.618034	1.618034	-2.161301e-08	-2.161301e-08	2.161301e-08	6.113083e-08
8	9	1.618034	1.618034	2.551070e-09	2.551070e-09	2.551070e-09	7.215515e-09
9	10	1.618034	1.618034	-3.011128e-10	-3.011128e-10	3.011128e-10	8.516756e-10
10	11	1.618034	1.618034	3.554135e-11	3.554135e-11	3.554135e-11	1.005261e-10
11	12	1.618034	1.618034	-4.195089e-12	-4.195089e-12	4.195089e-12	1.186550e-11

EJERCICIO 4

	iter	x0	x1	delta0	delta1	||delta||_inf	||f(x)||_2
9	10	-0.021818	1.021816	8.752078e-08	-5.879846e-08	8.752078e-08	1.171032e-07
10	11	-0.021818	1.021816	-1.907151e-08	6.043583e-09	1.907151e-08	2.443318e-08
11	12	-0.021818	1.021816	2.375674e-09	2.452962e-10	2.375674e-09	3.244761e-09
12	13	-0.021818	1.021816	-3.343822e-11	-2.609137e-10	2.609137e-10	3.043843e-10
13	14	-0.021818	1.021816	-6.723766e-11	6.309200e-11	6.723766e-11	9.757114e-11

EJERCICIO 9

	iter	x0	x1	x2	||x_{k+1}-x_k||_inf
6	7	0.664914	0.404248	0.253758	1.691655e-07
7	8	0.664914	0.404248	0.253758	2.732625e-08
8	9	0.664914	0.404248	0.253758	1.287094e-09
9	10	0.664914	0.404248	0.253758	1.280202e-10
10	11	0.664914	0.404248	0.253758	1.349443e-11

EJERCICIO 10

	iter	x0	x1	x2	||x_{k+1}-x_k||_inf
15	16	0.809017	0.809017	0.809017	5.248575e-09
16	17	0.809017	0.809017	0.809017	1.621899e-09
17	18	0.809017	0.809017	0.809017	5.011943e-10
18	19	0.809017	0.809017	0.809017	1.548776e-10
19	20	0.809017	0.809017	0.809017	4.785983e-11

C- Pseudocodigos

```
FUNCIÓN PuntoFijoMultivariable(g, x0, tol, maxiter):

  x ← x0
  historial ← lista vacía

  PARA k = 1 HASTA maxiter HACER:
    x_new ← g(x)                  # evaluar función de iteración
    err ← norma_infinito(x_new - x)   # error entre iteraciones

    guardar (k, x, x_new, err) en historial

    x ← x_new                     # actualizar

    SI err < tol ENTONCES:
      SALIR DEL BUCLE

  RETORNAR (x, err, k, historial)
```

Punto Fijo

Newton-Raphson Modificado

```
FUNCIÓN NewtonModificado(f, J, x0, tol, maxiter):

  x ← x0
  J0 ← J(x0)                      # Jacobiano congelado en x0
  historial ← lista vacía

  PARA k = 1 HASTA maxiter HACER:
    fx ← f(x)                     # evaluar funciones
    delta ← resolver(J0 * delta = -fx) # resolver sistema lineal
    x_new ← x + delta             # actualizar

    err ← norma_infinito(delta)       # tamaño del paso
    resnorm ← norma_2(fx)             # residuo ||f(x)||

    guardar (k, x, delta, err, resnorm) en historial

    x ← x_new

    SI (err < tol) Y (resnorm < tol) ENTONCES:
      SALIR DEL BUCLE

  RETORNAR (x, err, k, historial)
```

D- Código generado en Python de todos los ejercicios

```python
import numpy as np
import pandas as pd
from math import sin, cos, sqrt, exp

# ------------------------------------------------------------
# MODELAMIENTO COMPUTACIONAL PARA INGENIERIA - Métodos numéricos
# ------------------------------------------------------------

def punto_fijo_multivariable(g, x0, tol=1e-8, maxiter=100):
    """
    Método de punto fijo multivariable: x_{k+1} = g(x_k)
    g: función que recibe vector y devuelve vector (numpy array)
    x0: vector inicial
    """
    x = x0.astype(float).copy()
    history = []
    for k in range(1, maxiter+1):
        x_new = np.asarray(g(x))
        err = np.linalg.norm(x_new - x, ord=np.inf)
        history.append((k, x.copy(), x_new.copy(), err))
        x = x_new
        if err < tol:
            break
    return x, err, k, history

def newton_modificado(f, J, x0, tol=1e-8, maxiter=50):
    """
    Newton-Raphson modificado (Jacobiano congelado en x0):
    - Evalúa J0 = J(x0) una vez y lo usa para todas las iteraciones.
    - Cada iteración resuelve J0 * delta = -f(x_k)
    """
    x = x0.astype(float).copy()
    J0 = np.asarray(J(x0))
    history = []
    try:
        for k in range(1, maxiter+1):
            fx = np.asarray(f(x))
            delta = np.linalg.solve(J0, -fx)
            x_new = x + delta
            err = np.linalg.norm(delta, ord=np.inf)
            resnorm = np.linalg.norm(fx, ord=2)
            history.append((k, x.copy(), delta.copy(), err, resnorm))
            x = x_new
            if err < tol and resnorm < tol:
                break
        return x, err, k, history
    except Exception as e:
        return None, str(e), 0, history
```

```python
def mostrar_historial_puntofijo(history, names=None):
    rows = []
    for k, x_old, x_new, err in history:
        row = {"iter": k}
        if names is None:
            for i, val in enumerate(x_new):
                row[f"x{i}"] = val
        else:
            for i, name in enumerate(names):
                row[name] = x_new[i]
        row["||x_{k+1}-x_k||_inf"] = err
        rows.append(row)
    return pd.DataFrame(rows)

def mostrar_historial_newton(history, names=None):
    rows = []
    for k, x_old, delta, err, resnorm in history:
        row = {"iter": k}
        if names is None:
            for i, val in enumerate(x_old):
                row[f"x{i}"] = val
        else:
            for i, name in enumerate(names):
                row[name] = x_old[i]
        for i, d in enumerate(delta):
            row[f"delta{i}"] = d
        row["||delta||_inf"] = err
        row["||f(x)||_2"] = resnorm
        rows.append(row)
    return pd.DataFrame(rows)

#EJERCICIO 1
#Ejercicio PF1: x = cos(y), y = sin(x)
def g1(v):
    x, y = v
    return np.array([cos(y), sin(x)])
x0_1 = np.array([0.5, 0.5])

# Ejercicio PF2: x = sqrt((1+y)/2), y = sqrt((1+x)/2)
def g2(v):
    x, y = v
    return np.array([sqrt((1+y)/2), sqrt((1+x)/2)])
x0_2 = np.array([0.5, 0.5])

# Ejercicio PF3: x = (sin(y)+1)/2, y = (cos(x)+1)/2
def g3(v):
    x, y = v
    return np.array([(sin(y)+1)/2, (cos(x)+1)/2])
x0_3 = np.array([0.3, 0.3])
```

```
# Ejercicio PF4: sistema 3 variables
def g4(v):
    x, y, z = v
    return np.array([(cos(y*z)+1)/2, (sin(x)+1)/3, (x*y + 1)/4])
x0_4 = np.array([0.2, 0.2, 0.2])
#EJERCICIOS DE LA PRACTICA
# Ejercicio 9: Punto fijo de 3 variables
def g9(v):
    x, y, z = v
    return np.array([(cos(y*z)+1)/3,
                     (sin(x)+1)/4,
                     (x*y+1)/5])
x0_9 = np.array([0.2, 0.2, 0.2])

# Ejercicio 10: Punto fijo de 3 variables (simétrico)
def g10(v):
    x, y, z = v
    return np.array([sqrt((1+y+z)/4),
                     sqrt((1+x+z)/4),
                     sqrt((1+x+y)/4)])
x0_10 = np.array([0.5, 0.5, 0.5])
# ----------------------------
# EJERCICIOS NEWTON-RAPHSON MODIFICADO
# ----------------------------

# Ejercicio NRM1: f1 = x^2+y^2-4, f2 = exp(x)+y-1
def f_nr1(v):
    x, y = v
    return np.array([x**2 + y**2 - 4, exp(x) + y - 1])
def J_nr1(v):
    x, y = v
    return np.array([[2*x, 2*y],[exp(x), 1.0]])
x0_nr1 = np.array([1.0, 1.0])

# Ejercicio NRM2: sistema 3 variables
def f_nr2(v):
    x, y, z = v
    return np.array([x + y + z - 3, x**2 + y**2 + z**2 - 5, exp(x) + y - z -
1])
def J_nr2(v):
    x, y, z = v
    return np.array([[1,1,1],[2*x,2*y,2*z],[exp(x),1,-1]])
x0_nr2 = np.array([0.5, 1.0, 1.5])

# Ejercicio NRM3: f1 = x^2 - y -1, f2 = y^2 - x -1
def f_nr3(v):
    x, y = v
    return np.array([x**2 - y - 1, y**2 - x - 1])
def J_nr3(v):
    x, y = v
    return np.array([[2*x, -1],[-1, 2*y]])
x0_nr3 = np.array([1.5, 1.5])
```

```
# Ejercicio NRM4: f1 = sin(x)+y-1, f2 = x+cos(y)-0.5
def f_nr4(v):
    x, y = v
    return np.array([sin(x) + y - 1, x + cos(y) - 0.5])
def J_nr4(v):
    x, y = v
    return np.array([[cos(x), 1], [1, -sin(y)]])
x0_nr4 = np.array([0.5, 0.5])

# ----------------------------
# MAIN (PRINCIPAL): resolver todos los ejercicios
# ----------------------------

if __name__ == "__main__":

    print("===== PUNTO FIJO =====")
    for i, (g, x0, name) in enumerate([
    (g1, x0_1, "PF1: cos/sen"),
    (g2, x0_2, "PF2: sqrt-symmetric"),
    (g3, x0_3, "PF3: sen/cos scaled"),
    (g4, x0_4, "PF4: 3-variable"),
    (g9, x0_9, "PF9: 3-variable (cos-sin-mix)"),
    (g10, x0_10, "PF10: 3-variable (sqrt-symmetric)")]):

        sol, err, iters, hist = punto_fijo_multivariable(g, x0, tol=1e-10, maxiter=500)
        df = mostrar_historial_puntofijo(hist)
        print(f"{name}: solución = {sol}, iteraciones = {iters}, error final = {err}")
        print(df.tail(), "\n")

    print("===== NEWTON-RAPHSON MODIFICADO =====")
    for i, (f, J, x0, name) in enumerate([
        (f_nr1,J_nr1,x0_nr1,"NRM1: circle+exp"),
        (f_nr2,J_nr2,x0_nr2,"NRM2: 3-var"),
        (f_nr3,J_nr3,x0_nr3,"NRM3: symmetric quad"),
        (f_nr4,J_nr4,x0_nr4,"NRM4: sin/cos mix")]):

        sol, err, iters, hist = newton_modificado(f, J, x0, tol=1e-10, maxiter=50)
        if sol is None:
            print(f"{name}: error → {err}")
        else:
            df = mostrar_historial_newton(hist)
            print(f"{name}: solución = {sol}, iteraciones = {iters}, error final = {err}")
            print(df.tail(), "\n")
```