Taller

Lina Mercedes Espitia Cuadrado - Cod. 201921194
Jeimmy Patricia Valderrama Vásquez - Cod. 201821062
Ingeniería de Sistemas y Computación, Universidad Pedagógica y Tecnológica de Colombia

Método de Broyden

A la aplicación del ejercicio asignado (para nuestro caso el ejercicio número 13. Problema de análisis de redes eléctricas)

Ejercicio 13: Problema de análisis de redes eléctricas

Considera el problema de un sistema eléctrico con tres nodos y flujos de corriente no lineales representados por:

$$f_1(V_1, V_2, V_3) = V_1^2 - V_2 + V_3 - 3 = 0$$
$$f_2(V_1, V_2, V_3) = V_1 - V_2^3 + \cos(V_3) - 1 = 0$$
$$f_3(V_1, V_2, V_3) = \sin(V_1) + V_2 - V_3^2 = 0$$

1. Resolver el sistema usando el método de Broyden para obtener las tensiones V_1, V_2, V_3.
2. Realizar un análisis de convergencia en función de las condiciones iniciales.
3. Evaluar la estabilidad y eficiencia del método en comparación con el método de Newton-Raphson.

2. Realizar un análisis de convergencia en función de las condiciones iniciales

Para realizar el análisis de la convergencia, probamos con diferentes condiciones iniciales, en este caso variamos 5 veces las condiciones, con los siguientes resultados:

valores_iniciales = [0, 0, 0]

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
  Iteración 1: V = [3. 0. 0.], Norma(función) = 3.0
  Iteración 2: V = [ 1.20039821 -0.8998009  -0.04232664], Norma(función) = 6.709688134084535
  Iteración 3: V = [ 1.70486848 -2.37230954 -0.06610356], Norma(función) = 2.051924169935079
  Iteración 4: V = [ 1.44816308 -0.58637323 -0.29851586], Norma(función) = 15.278419624910976
  Iteración 5: V = [ 1.70522345 -0.32620317 -0.8742209 ], Norma(función) = 1.7482770632126574
  Iteración 6: V = [ 1.94653027 -0.04054216 -1.00554719], Norma(función) = 1.5259145257636484
  Iteración 7: V = [ 1.95543919 -0.01928797 -1.01482814], Norma(función) = 1.4975697330016093
  Iteración 8: V = [ 0.23184723 -4.99545503  0.9692624 ], Norma(función) = 1.498123392240795
  Iteración 9: V = [ 2.15232334  0.44443613 -1.14336218], Norma(función) = 124.62444631141324
  Iteración 10: V = [ 2.20458528  0.55306866 -1.23099245], Norma(función) = 1.4799994111495869
  Iteración 11: V = [  9.35889959  12.20553487 -12.43244264], Norma(función) = 1.379730654075495
  Iteración 12: V = [ 1.66707775 -0.4140397  -0.42462647], Norma(función) = 1815.5485222726395
  Iteración 13: V = [ 2.30649698  0.75259891 -1.34982376], Norma(función) = 1.7130100286648005
  Iteración 14: V = [ 6.3864412   8.18676707 -7.2873503 ], Norma(función) = 1.167739219764976
  Iteración 15: V = [ 1.34537459 -1.00029073  0.04416513], Norma(función) = 545.0835895366728
  Iteración 16: V = [ 1.12703507 -1.33944197  0.4730715 ], Norma(función) = 2.349943060869446
  Iteración 17: V = [ 3.25071301  2.38198447 -2.89545155], Norma(función) = 3.484403984538738
  Iteración 18: V = [-3.16270723 -8.96623897  7.06906002], Norma(función) = 13.865661070732
  Iteración 19: V = [ -8.72768571 -18.95036005  15.45467763], Norma(función) = 720.1547723329919
  Iteración 20: V = [ -64.88622171 -118.77512191  101.84331426], Norma(función) = 6800.448596492282
  Iteración 21: V = [-415.67524509 -746.88024     632.79563925], Norma(función) = 1675596.6131893813
  Iteración 22: V = [-16959.04379905 -30353.61776184  25701.22620479], Norma(función) = 416632086.3048117
  Iteración 23: V = [ -750510.16950292 -1345195.98350593  1133371.54432426], Norma(función) = 27966066264548.008
  Iteración 24: V = [-1.23908303e+09 -2.22120323e+09  1.87059565e+09], Norma(función) = 2.434202397175101e+18
  Iteración 25: V = [-2.46838512e+12 -4.42630449e+12  3.72370301e+12], Norma(función) = 1.0958847596358756e+28
  Iteración 26: V = [-6.72600819e+18 -1.20614082e+19  1.01459377e+19], Norma(función) = 8.672091637643845e+37
  Iteración 27: V = [-2.31985793e+27 -2.09397961e+29  1.19317294e+29], Norma(función) = 1.7546643416646443e+57
  Iteración 28: V = [-4.72558631e+52 -2.85085906e+52  8.14091997e+51], Norma(función) = 9.181578336095621e+87
  c:\Users\valde\Documents\Ejercicio13.py:13: RuntimeWarning: overflow encountered in scalar power
    V1 - V2**3 + np.cos(V3) - 1,
  Iteración 29: V = [-3.13588592e+105 -6.41471374e+105  3.22618386e+105], Norma(función) = inf
  Iteración 30: V = [nan nan nan], Norma(función) = inf
  Iteración 31: V = [nan nan nan], Norma(función) = nan
  Iteración 32: V = [nan nan nan], Norma(función) = nan
  Iteración 33: V = [nan nan nan], Norma(función) = nan
  Iteración 34: V = [nan nan nan], Norma(función) = nan
  Iteración 35: V = [nan nan nan], Norma(función) = nan
  Iteración 36: V = [nan nan nan], Norma(función) = nan
  Iteración 37: V = [nan nan nan], Norma(función) = nan
  Iteración 38: V = [nan nan nan], Norma(función) = nan
  Iteración 39: V = [nan nan nan], Norma(función) = nan
  Iteración 40: V = [nan nan nan], Norma(función) = nan
```

valores_iniciales = [0.5, 0.5, 0.5]

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
  Iteración 1: V = [ 3.25        0.24741744 -0.22942554], Norma(función) = 2.8562842237668042
  Iteración 2: V = [ 1.45868714 -0.56375626 -0.25131525], Norma(función) = 7.7787837555200126
  Iteración 3: V = [ 1.86906142 -1.92504719 -0.55710532], Norma(función) = 1.7402829679232457
  Iteración 4: V = [ 1.6394555  -0.40543875 -0.84680879], Norma(función) = 9.13534342817717
  Iteración 5: V = [ 1.92603896 -0.07794784 -0.95939217], Norma(función) = 1.567219217017719
  Iteración 6: V = [ 2.04913082  0.19029358 -1.14375493], Norma(función) = 1.5115594124678169
  Iteración 7: V = [ 2.52227157  1.42679475 -1.30118635], Norma(función) = 1.480664766139993
  Iteración 8: V = [ 2.19624999  0.80152733 -1.21344609], Norma(función) = 1.321317188389167
  Iteración 9: V = [ 2.36294978  1.25060278 -1.40391596], Norma(función) = 1.058003246697058
  Iteración 10: V = [ 2.39671891  1.24862019 -1.41977797], Norma(función) = 0.43313390789100986
  Iteración 11: V = [ 2.10607899  0.59450389 -1.08035399], Norma(función) = 0.41632318443347516
  Iteración 12: V = [ 2.32612285  1.09193769 -1.33748343], Norma(función) = 1.4172226198297877
  Iteración 13: V = [ 2.34349113  1.14743532 -1.35912901], Norma(función) = 0.25793614524304725
  Iteración 14: V = [ 2.36177825  1.18519409 -1.38060451], Norma(función) = 0.04810470074148373
  Iteración 15: V = [ 2.35736073  1.17439318 -1.37530377], Norma(función) = 0.11601200762465111
  Iteración 16: V = [ 2.35027374  1.15735169 -1.36679123], Norma(función) = 0.06936203763128236
  Iteración 17: V = [ 2.35063287  1.15817275 -1.36722884], Norma(función) = 0.0027102637950007077
  Iteración 18: V = [ 2.35058113  1.15804452 -1.36716556], Norma(función) = 0.0007464042551949859
  Iteración 19: V = [ 2.35055991  1.1579927  -1.36713994], Norma(función) = 0.0002114722566826489 7
  Iteración 20: V = [ 2.35056062  1.15799425 -1.36714075], Norma(función) = 4.595932149058309e-06
  Iteración 21: V = [ 2.35056047  1.15799389 -1.36714057], Norma(función) = 1.888800594703142e-06
  Convergencia alcanzada en la iteración 22
  Solución final: V1 = 2.350560467209131, V2 = 1.1579938887919552, V3 = -1.3671405731433426
```

valores_iniciales = [1, 1, 1]

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
 Iteración 1: V = [3.         1.45969769 0.15852902], Norma(función) = 2.2179710070836633
 Iteración 2: V = [ 1.66500709  1.49457031 -0.28914187], Norma(función) = 4.957506299563232
 Iteración 3: V = [ 1.89613714  2.03800976 -1.16945015], Norma(función) = 3.574711595835767
 Iteración 4: V = [ 1.91388141  1.52785129 -1.26842988], Norma(función) = 7.808024738258819
 Iteración 5: V = [ 2.42835856  1.08078912 -1.42351052], Norma(función) = 3.2919861033417765
 Iteración 6: V = [ 2.36490674  1.11456963 -1.34980966], Norma(función) = 0.5803089266826555
 Iteración 7: V = [ 2.3237872   1.16389351 -1.36663655], Norma(función) = 0.23734635647697994
 Iteración 8: V = [ 2.57613743  0.97203009 -1.2371569 ], Norma(función) = 0.1422073564547437
 Iteración 9: V = [ 2.39029525  1.12048953 -1.34116624], Norma(función) = 1.734452171865683
 Iteración 10: V = [ 2.40794503  1.11677893 -1.33787728], Norma(función) = 0.32868060275233063
 Iteración 11: V = [ 2.45874915  1.06837102 -1.304517  ], Norma(función) = 0.42250720153090715
 Iteración 12: V = [ 2.31706155  1.1929065  -1.39099924], Norma(función) = 0.839514251246109
 Iteración 13: V = [ 2.29405299  1.20021186 -1.39708691], Norma(función) = 0.29496165102111044
 Iteración 14: V = [ 2.2317625   1.25345937 -1.43444477], Norma(función) = 0.42500765070802016
 Iteración 15: V = [ 2.44720734  1.08878907 -1.31790156], Norma(función) = 0.9286016863049391
 Iteración 16: V = [ 2.26057697  1.22053817 -1.41163268], Norma(función) = 0.7101749070134382
 Iteración 17: V = [ 2.31758853  1.17807772 -1.38149372], Norma(función) = 0.6571091053080186
 Iteración 18: V = [ 2.40201175  1.12033213 -1.34051641], Norma(función) = 0.22846346118248656
 Iteración 19: V = [ 2.3588514   1.15025298 -1.36173383], Norma(función) = 0.3815565299022131
 Iteración 20: V = [ 2.36594714  1.14615105 -1.35879772], Norma(función) = 0.06860820233153891
 Iteración 21: V = [ 2.37008651  1.14267848 -1.35636483], Norma(función) = 0.11663283753017611
 Iteración 22: V = [ 2.34503895  1.16263169 -1.37038424], Norma(función) = 0.1491438331488405
 Iteración 23: V = [ 2.34809208  1.15983894 -1.36844501], Norma(función) = 0.04353880023658028
 Iteración 24: V = [ 2.35333647  1.15581952 -1.36560937], Norma(función) = 0.01850664162900915
 Iteración 25: V = [ 2.3507953   1.15780254 -1.36700628], Norma(función) = 0.021217220129982686
 Iteración 26: V = [ 2.35061065  1.15795469 -1.36711298], Norma(función) = 0.001825878175150244
 Iteración 27: V = [ 2.35054662  1.1580047  -1.3671482 ], Norma(función) = 0.0003829859414070598 6
 Iteración 28: V = [ 2.35055967  1.15799439 -1.36714095], Norma(función) = 0.0001059196803538826 3
 Iteración 29: V = [ 2.35056043  1.15799379 -1.36714052], Norma(función) = 5.810142253121051e-06
 Convergencia alcanzada en la iteración 30
 Solución final: V1 = 2.35056043499653, V2 = 1.157993788818548, V3 = -1.367140521195237
```

valores_iniciales = [1.5, 1.5, 1.5]

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
 Iteración 1: V = [2.25       4.3042628  1.25250501], Norma(función) = 2.9133560734150965
 Iteración 2: V = [2.21309391 1.38758514 1.38358515], Norma(función) = 78.26591279141516
 Iteración 3: V = [0.23812437 1.33494748 1.16182423], Norma(función) = 2.2979513370993665
 Iteración 4: V = [1.41373227 1.28989022 1.12552308], Norma(función) = 4.157649723894387
 Iteración 5: V = [1.85284264 1.22129795 0.20369217], Norma(función) = 2.01867032920974
 Iteración 6: V = [1.85546938 1.22374857 0.30036701], Norma(función) = 2.2187177898719566
 Iteración 7: V = [0.70135798 1.3071546  5.2040548 ], Norma(función) = 2.1478595383973342
 Iteración 8: V = [1.90877933 1.21823617 0.47325715], Norma(función) = 25.252289089053182
 Iteración 9: V = [1.81537983 1.21919349 0.8320383 ], Norma(función) = 1.9403693272879559
 Iteración 10: V = [1.52129985 1.17804607 2.00243267], Norma(función) = 1.53442937435762
 Iteración 11: V = [1.6797244  1.17413343 1.34752767], Norma(función) = 2.392837483313602
 Iteración 12: V = [1.64857601 1.14923597 1.44507754], Norma(función) = 0.799380115585164
 Iteración 13: V = [1.63648405 1.12083546 1.45848922], Norma(función) = 0.7462614452137837
 Iteración 14: V = [1.62158553 1.05197422 1.43104287], Norma(función) = 0.6597616835302105
 Iteración 15: V = [1.59595576 0.93916329 1.39273952], Norma(función) = 0.40338410906781186
 Iteración 16: V = [1.59234437 0.92140854 1.38578999], Norma(función) = 0.055305613119465795
 Iteración 17: V = [1.59182001 0.91907641 1.38517152], Norma(función) = 0.006021654558678989
 Iteración 18: V = [1.59180513 0.91906176 1.38521783], Norma(función) = 0.0001599070559120781 2
 Iteración 19: V = [1.59180427 0.91906428 1.38522362], Norma(función) = 1.7586377185518076e-05
 Iteración 20: V = [1.59180428 0.91906418 1.38522336], Norma(función) = 1.1235560267627587e-06
 Iteración 21: V = [1.59180428 0.91906413 1.38522325], Norma(función) = 3.10426335447464e-07
 Iteración 22: V = [1.59180428 0.91906414 1.38522326], Norma(función) = 4.171221786380735e-08
 Convergencia alcanzada en la iteración 23
 Solución final: V1 = 1.5918042839479905, V2 = 0.9190641380083905, V3 = 1.3852232596080156
PS C:\Users\valde>
```

valores_iniciales = [2, 2, 2]

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
Iteración 1: V = [1.         9.41614684 3.09070257], Norma(función) = 7.562332048009877
Iteración 2: V = [0.92561903 1.94833627 3.09700273], Norma(función) = 835.9120233401729
Iteración 3: V = [ 1.8516558   1.87602543 10.07287969], Norma(función) = 10.934342289890422
Iteración 4: V = [2.47801465 1.85795813 2.81465401], Norma(función) = 99.21883879619078
Iteración 5: V = [-1.42567824  1.84705593  1.9754287 ], Norma(función) = 9.004360336201094
Iteración 6: V = [-0.72689958  1.7784718   1.92188051], Norma(función) = 9.65212990897802
Iteración 7: V = [-2.94504271  1.82808549  1.7240428 ], Norma(función) = 8.44421124116983
Iteración 8: V = [-2.38635341  1.76157047  1.69074978], Norma(función) = 11.704391512548915
Iteración 9: V = [-2.85253696  1.67380066  1.48957097], Norma(función) = 9.516620235502465
Iteración 10: V = [-3.13023748  1.86333205  1.73624351], Norma(función) = 9.838834433186687
Iteración 11: V = [-2.01300022  1.02330533  0.68069994], Norma(función) = 12.717341006472207
Iteración 12: V = [-1.86519664  0.72477499  0.30741077], Norma(función) = 3.4001155624986406
Iteración 13: V = [-1.83153074  0.24011474 -0.32052116], Norma(función) = 2.3167778344737844
Iteración 14: V = [-1.89423404 -0.10267212 -0.80539016], Norma(función) = 2.0797592425269635
Iteración 15: V = [-1.66318414  0.84513455  0.63546638], Norma(función) = 2.782581419053245
Iteración 16: V = [-0.89654745  3.69474528  5.21824498], Norma(función) = 2.562351405267799
Iteración 17: V = [-1.65685191  1.10747928  1.03291285], Norma(función) = 57.27228346843995
Iteración 18: V = [-1.65665588  1.15584706  1.13291195], Norma(función) = 3.645820470150164
Iteración 19: V = [-1.59756331  1.37212587  1.18563585], Norma(función) = 3.9503411623132463
Iteración 20: V = [-2.1449206  -1.54888259 -1.47023108], Norma(función) = 4.955788000633943
Iteración 21: V = [ 1.6260625  15.5310112  19.44025854], Norma(función) = 4.896376313698343
Iteración 22: V = [-0.68035386  6.3874842   7.59453779], Norma(función) = 3762.21393706691
Iteración 23: V = [ 8.55311921 72.44674934 86.45045232], Norma(función) = 267.1301864420159
Iteración 24: V = [ 15.95350728 122.1915706  145.60895151], Norma(función) = 380303.45584185777
Iteración 25: V = [ 1368.00369375  9350.95991885 11112.25000469], Norma(función) = 1824521.839651251
Iteración 26: V = [ 7552.20396301 51448.88629235 61118.79787644], Norma(función) = 817652164324.4568
Iteración 27: V = [3.68250339e+07 2.50528355e+08 2.97564947e+08], Norma(función) = 136184579565208.36
Iteración 28: V = [1.35291725e+09 9.20332312e+09 1.09313166e+10], Norma(función) = 1.5724276135202098e+25
Iteración 29: V = [3.12026729e+16 2.12260130e+17 2.52108667e+17], Norma(función) = 7.795321104120713e+29
Iteración 30: V = [ 7.69237050e+24 -7.83208050e+25 -6.57057523e+26], Norma(función) = 9.56324495750497e+51
Iteración 31: V = [-9.89548261e+43  9.85782617e+44  1.02739029e+45], Norma(función) = 4.80431447809169e+77
C:\Users\valde\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-pac
rflow encountered in multiply
  return multiply(a.ravel()[:, newaxis], b.ravel()[newaxis, :], out)
c:\Users\valde\Documents\Ejercicio13.py:40: RuntimeWarning: invalid value encountered in divide
  inversa_jacobiano += np.outer(diferencia_V - np.dot(inversa_jacobiano, diferencia_funcion),
Iteración 32: V = [3.82522896e+89 2.61020166e+90 3.10165179e+90], Norma(función) = 9.579513782353782e+134
Iteración 33: V = [nan nan nan], Norma(función) = inf
Iteración 34: V = [nan nan nan], Norma(función) = nan
Iteración 35: V = [nan nan nan], Norma(función) = nan
Iteración 36: V = [nan nan nan], Norma(función) = nan
Iteración 37: V = [nan nan nan], Norma(función) = nan
```

Análisis de cada una de las condiciones iniciales:

1.Condiciones iniciales = [0, 0, 0]: estas condiciones hicieron que el sistema actuara de manera impredecible desde el principio, con la norma cambiando mucho A partir de la novena vez, los números de las variables comenzaron a aumentar muy rápidamente, lo que provocó desbordamientos numéricos y resultados indefinidos (nan). Esto indica que el método de Broyden no pudo corregir el curso y converger desde esta posición inicial.

2.Condiciones iniciales = [0.5, 0.5, 0.5]: Estas condiciones iniciales tuvieron una convergencia estable. Aunque las primeras iteraciones mostraron fluctuaciones, el método rápidamente corrigió el curso y alcanzó una solución en 23 iteraciones. Las normas disminuyeron de manera consistente, y el método de Broyden logró converger adecuadamente sin mayores problemas.

3.Condiciones iniciales = [1, 1, 1]: Estas condiciones también permitieron que el método de Broyden convergiera con éxito, aunque necesito un poco más de iteraciones (31). Hubo algunas fluctuaciones en las primeras iteraciones, pero el método mostró una buena corrección y las normas disminuyeron de forma progresiva hasta alcanzar una solución estable.

4. Condiciones iniciales = [1.5, 1.5, 1.5]: Con estas condiciones iniciales, el método de Broyden alcanzó la convergencia en 23 iteraciones. Aunque hubo más fluctuaciones en la norma al inicio, el método logró estabilizarse y llegar a una solución.

5. Condiciones iniciales = [2, 2, 2]: Estas condiciones iniciales resultaron en una divergencia extrema. A partir de la segunda iteración, las normas comenzaron a crecer exponencialmente, y el método rápidamente perdió estabilidad. En la iteración 33, los valores de las variables se volvieron indefinidos (nan), lo que indica que el método no pudo encontrar una solución adecuada desde este punto de partida.

Según los resultados obtenidos, el intervalo recomendado para las condiciones iniciales se encuentran entre **[0.5, 0.5, 0.5] y [1.5, 1.5, 1.5],** se observó que con estos valores iniciales se obtuvo una convergencia mucho más estable, con un número de iteraciones relativamente bajo (las dos tuvieron 22 iteraciones), mientras que con los otros valores, como **[0, 0, 0] y [2, 2, 2],** provocaron la divergencia del método, hasta el punto de obtener valores indefinidos; entonces podemos concluir que la asignación de la asignación adecuada de valores iniciales puede marcar una diferencia en la estabilidad y convergencia del método de Broyden.

3. Evaluar la estabilidad y eficiencia del método en comparación del método de Newton - Raphson

Para evaluar la estabilidad y la eficiencia de ambos métodos se tomaron los resultados del ejercicio propuesto bajo el mismo valor de las condiciones iniciales tomando en cuenta:

- Número de iteraciones de cada método y comparando la cantidad de iteraciones realizadas en cada uno de los métodos.
- Valores en la solución final, se observará el resultado de cada uno de los métodos en comparación para dar con la eficiencia de estos.

Condiciones iniciales = [1.5, 1.5, 1.5]

Método de Newton- Raphson

```
main.py
20  def newton_raphson_system(f, jacobian, V0, tol, max_iter):  1 usage
35      print("No se alcanzó la convergencia en el número máximo de iteraciones")
36      return V
37
38  V0 = np.array([1.5, 1.5, 1.5])
39  tol = 1e-6
40  max_iter = 100
41
42  V_sol = newton_raphson_system(f_system, jacobian, V0, tol, max_iter)
43  print("Solución aproximada con Newton-Raphson:", V_sol)

Run  main
C:\Users\linae\PycharmProjects\ExercicesMetodos\.venv\Scripts\python.exe C:\Users\linae\PycharmProjects\Exercices
Iteración 1: V = [1.63500657 1.11105204 1.45603234], |delta_V| = 0.4140536701761497
Iteración 2: V = [1.59775744 0.94877949 1.39733813], |delta_V| = 0.1765358032984636
Iteración 3: V = [1.59196916 0.91991855 1.38558625], |delta_V| = 0.03169487092186939
Iteración 4: V = [1.59180442 0.91906487 1.38522357], |delta_V| = 0.00094203841020781
Iteración 5: V = [1.59180428 0.91906414 1.38522326], |delta_V| = 8.079972498656726e-07
Convergencia alcanzada en 5 iteraciones
Solución aproximada con Newton-Raphson: [1.59180428 0.91906414 1.38522326]

Process finished with exit code 0
```

Método de Broyden:

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
Iteración 1: V = [2.25       4.3042628  1.25250501], Norma(función) = 2.9133560734150965
Iteración 2: V = [2.21309391 1.38758514 1.38358515], Norma(función) = 78.26591279141516
Iteración 3: V = [0.23812437 1.33494748 1.16182423], Norma(función) = 2.2979513370993665
Iteración 4: V = [1.41373227 1.28989022 1.12552308], Norma(función) = 4.157649723894387
Iteración 5: V = [1.85284264 1.22129795 0.20369217], Norma(función) = 2.01867032920974
Iteración 6: V = [1.85546938 1.22374857 0.30036701], Norma(función) = 2.2187177898719566
Iteración 7: V = [0.70135798 1.3071546  5.2040548 ], Norma(función) = 2.1478595383973342
Iteración 8: V = [1.90877933 1.21823617 0.47325715], Norma(función) = 25.252289089053182
Iteración 9: V = [1.81537983 1.21919349 0.8320383 ], Norma(función) = 1.9403693272879559
Iteración 10: V = [1.52129985 1.17804607 2.00243267], Norma(función) = 1.53442937435762
Iteración 11: V = [1.6797244  1.17413343 1.34752767], Norma(función) = 2.392837483313602
Iteración 12: V = [1.64857601 1.14923597 1.44507754], Norma(función) = 0.799380115585164
Iteración 13: V = [1.63648405 1.12083546 1.45848922], Norma(función) = 0.7462614452137837
Iteración 14: V = [1.62158553 1.05197422 1.43104287], Norma(función) = 0.6597616835302105
Iteración 15: V = [1.59595576 0.93916329 1.39273952], Norma(función) = 0.40338410906781186
Iteración 16: V = [1.59234437 0.92140854 1.38578999], Norma(función) = 0.055305613119465795
Iteración 17: V = [1.59182001 0.91907641 1.38517152], Norma(función) = 0.006021654558678989
Iteración 18: V = [1.59180513 0.91906176 1.38521783], Norma(función) = 0.00015990705591207812
Iteración 19: V = [1.59180427 0.91906428 1.38522362], Norma(función) = 1.7586377185518076e-05
Iteración 20: V = [1.59180428 0.91906418 1.38522336], Norma(función) = 1.1235560267627587e-06
Convergencia alcanzada en la iteración 21
Solución final: V1 = 1.5918042773251118, V2 = 0.9190641785687516, V3 = 1.3852233573105013
```

Condiciones iniciales = [0, 0, 0]

Método de Newton- Raphson

```
EM ExercicesMetodos  Version control   main
main.py
20  def newton_raphson_system(f, jacobian, V0, tol, max_iter):  1 usage   20 15
36      return V
37
38  V0 = np.array([0,0, 0])
39  tol = 1e-6
40  max_iter = 100
41
42  V_sol = newton_raphson_system(f_system, jacobian, V0, tol, max_iter)
43  print("Solución aproximada con Newton-Raphson:", V_sol)

main
C:\Users\linae\PycharmProjects\ExercicesMetodos\.venv\Scripts\python.exe C:\Users\linae\PycharmProjects\Exercice
Iteración 1: V = [0. 0. 3.], |delta_V| = 3.0
Iteración 2: V = [ 1.78639572 -1.44272086  1.55727914], |delta_V| = 2.711843763856582
Iteración 3: V = [ 1.26371682 -0.80720324  0.8690098 ], |delta_V| = 1.0727490651460105
Iteración 4: V = [ 1.41261874 -0.04198013  0.98469995], |delta_V| = 0.7881131950841581
Iteración 5: V = [2.01926382 3.55871325 2.84930513], |delta_V| = 4.099971165187312
Iteración 6: V = [1.84226911 2.3757817  2.01315335], |delta_V| = 1.4593847949346908
Iteración 7: V = [1.72568615 1.62048779 1.65608671], |delta_V| = 0.8435384227048288
Iteración 8: V = [1.64177676 1.17270212 1.48431197], |delta_V| = 0.48688740977957734
Iteración 9: V = [1.60130446 0.96761704 1.40507906], |delta_V| = 0.22355256565370102
Iteración 10: V = [1.59223334 0.92129272 1.386168  ], |delta_V| = 0.05085132029211019
Iteración 11: V = [1.59180523 0.9190691  1.3852254 ], |delta_V| = 0.0024528021562128482
Iteración 12: V = [1.59180428 0.91906414 1.38522326], |delta_V| = 5.4873972098466515e-06
Iteración 13: V = [1.59180428 0.91906414 1.38522326], |delta_V| = 2.730057777226927e-11
Convergencia alcanzada en 13 iteraciones
Solución aproximada con Newton-Raphson: [1.59180428 0.91906414 1.38522326]
```

Método de Broyden:

```
PS C:\Users\valde> & C:/Users/valde/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/valde/Documents/Ejercicio13.py
Iteración 1: V = [3. 0. 0.], Norma(función) = 3.0
Iteración 2: V = [ 1.20039821 -0.8998009  -0.04232664], Norma(función) = 6.709688134084535
Iteración 3: V = [ 1.70486848 -2.37230954 -0.06610356], Norma(función) = 2.051924169935079
Iteración 4: V = [ 1.44816308 -0.58637323 -0.29851586], Norma(función) = 15.278419624910976
Iteración 5: V = [ 1.70522345 -0.32620317 -0.8742209 ], Norma(función) = 1.7482770632126574
Iteración 6: V = [ 1.94653027 -0.04054216 -1.00554719], Norma(función) = 1.5259145257636484
Iteración 7: V = [ 1.95543919 -0.01928797 -1.01482814], Norma(función) = 1.4975697330016093
Iteración 8: V = [ 0.23184723 -4.99545503  0.9692624 ], Norma(función) = 1.498123392240795
Iteración 9: V = [ 2.15232334  0.44443613 -1.14336218], Norma(función) = 124.62444631141324
Iteración 10: V = [ 2.20458528  0.55306866 -1.23099245], Norma(función) = 1.4799994111495869
Iteración 11: V = [  9.35889959  12.20553487 -12.43244264], Norma(función) = 1.379730654075495
Iteración 12: V = [ 1.66707775 -0.4140397  -0.42462647], Norma(función) = 1815.5485222726395
Iteración 13: V = [ 2.30649698  0.75259891 -1.34982376], Norma(función) = 1.7130100286648005
Iteración 14: V = [ 6.3864412   8.18676707 -7.2873503 ], Norma(función) = 1.167739219764976
Iteración 15: V = [ 1.34537459 -1.00029073  0.04416513], Norma(función) = 545.0835895366728
Iteración 16: V = [ 1.12703507 -1.33944197  0.4730715 ], Norma(función) = 2.349943060869446
Iteración 17: V = [ 3.25071301  2.38198447 -2.89545155], Norma(función) = 3.484403984538738
Iteración 18: V = [-3.16270723 -8.96623897  7.06906002], Norma(función) = 13.865661070732
Iteración 19: V = [ -8.72768571 -18.95036005  15.45467763], Norma(función) = 720.1547723329919
Iteración 20: V = [ -64.88622171 -118.77512191  101.84331426], Norma(función) = 6800.448596492282
Iteración 21: V = [-415.67524509 -746.88024     632.79563925], Norma(función) = 1675596.6131893813
Iteración 22: V = [-16959.04379905 -30353.61776184  25701.22620479], Norma(función) = 416632086.3048117
Iteración 23: V = [ -750510.16950292 -1345195.98350593  1133371.54432426], Norma(función) = 27966066264548.008
Iteración 24: V = [-1.23908303e+09 -2.22120323e+09  1.87059565e+09], Norma(función) = 2.434202397175101e+18
Iteración 25: V = [-2.46838512e+12 -4.42630449e+12  3.72370301e+12], Norma(función) = 1.0958847596358756e+28
Iteración 26: V = [-6.72600819e+18 -1.20614082e+19  1.01459377e+19], Norma(función) = 8.672091637643845e+37
Iteración 27: V = [-2.31985793e+27 -2.09397961e+29  1.19317294e+29], Norma(función) = 1.7546643416646443e+57
Iteración 28: V = [-4.72558631e+52 -2.85085906e+52  8.14091997e+51], Norma(función) = 9.181578336095621e+87
c:\Users\valde\Documents\Ejercicio13.py:13: RuntimeWarning: overflow encountered in scalar power
  V1 - V2**3 + np.cos(V3) - 1,
Iteración 29: V = [-3.13588592e+105 -6.41471374e+105  3.22618386e+105], Norma(función) = inf
Iteración 30: V = [nan nan nan], Norma(función) = inf
Iteración 31: V = [nan nan nan], Norma(función) = nan
Iteración 32: V = [nan nan nan], Norma(función) = nan
Iteración 33: V = [nan nan nan], Norma(función) = nan
Iteración 34: V = [nan nan nan], Norma(función) = nan
Iteración 35: V = [nan nan nan], Norma(función) = nan
Iteración 36: V = [nan nan nan], Norma(función) = nan
Iteración 37: V = [nan nan nan], Norma(función) = nan
Iteración 38: V = [nan nan nan], Norma(función) = nan
Iteración 39: V = [nan nan nan], Norma(función) = nan
Iteración 40: V = [nan nan nan], Norma(función) = nan
```

Eficiencia:

Condiciones iniciales = [1.5, 1.5, 1.5]

Bajo estas condiciones iniciales se puede observar que el método de Newton- Raphson requirio menor número de iteraciones a comparación del método de Broyden. Siendo este método más eficiente que el anterior a comparación de estos valores iniciales.

Condiciones iniciales = [0, 0, 0]

En estas condiciones iniciales el método de Broyden después de una cantidad de iteraciones los números de las variables aumentaron demasiado y provocaron desbordamiento numérico mientras que el método de Newton- Raphson si concluyó con la solución aproximada en 13 iteraciones.

Por lo tanto el método más eficiente es el método de Newton- Raphson requiriendo un número menor de iteraciones en ambas condiciones iniciales.

Estabilidad: Como se muestra en la ejecución de los ejemplos anteriores la estabilidad del método de Broyden es menor que la del método de Newton- Raphson ya que bajo algunas condiciones iniciales este método no da una solución aproximada.