**TUGAS SISTEM PERSAMAAN NONLINEAR**

**METODE NUMERIK KELAS B**



Oleh :

|  |  |
| --- | --- |
| Fadia Nur Fatimah | 21120123120022 |

**DEPARTEMEN TEKNIK KOMPUTER**

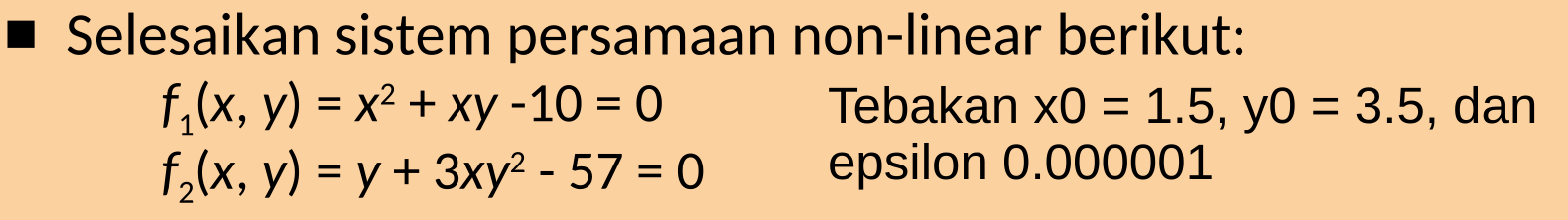
**FAKULTAS TEKNIK**

**UNIVERSITAS DIPONEGORO**

**SEMARANG**

**2025**

**SOAL:**

****

Buatlah demonstrasi pemecahan solusi sistem persamaan linear dari materi M06 sebagai berikut:

* Demonstrasi bisa menggunakan aplikasi spreadsheet atau pemrograman (C, Python, Pascal, atau lainnya)
* Fungsi iterasi: Iterasi titik tetap (IT), Newton-Raphson (IN), dan Secant (IS)
* Metode iterasi di IT: Jacobi (J) dan Seidel (S)
* Fungsi iterasi g1, g2 di IT: g1A dan g2A (halaman 5), g1B dan g2B  (halaman 6)

Ketentuan:

* NIMx = 2 digit terakhir NIM mod 4
* Setiap mahasiswa melakukan demonstrasi 4 solusi sbb:
  + 2 metode IT dengan Jacobi dan Seidel dengan kombinasi fungsi iterasi: g1A dan g2A (NIMx = 0), g1A dan g2B (NIMx = 1), g2A dan g1B dan g2A (NIMx = 2), dan g1B dan g2B (NIMx=3)
  + 1 metode Newton-Raphson
  + 1 metode Secant
* Tebakan nilai silahkan ditentukan. Toleransi galat 0,000001.
* Mahasiswa menganalisis konvergensi dan kecepatannya
* Solusi dinyatakan dalam file doc atau docx
* Program atau spreadsheet diunggah di github masing2 mahasiswa dan berikan akses ke saya (ekodidik) untuk bisa melihatnya. Tautan github dinyatakan di file doc di atas.

**JAWABAN**

**CARA KERJA**

* **Sistem Persamaan non-linear**:

f₁(x,y) = x² + xy - 10 = 0

f₂(x,y) = y + 3xy² - 57 = 0

* **Solusi Sejati**: x = 2, y = 3
* Parameter:
  1. NIM: 21120123120022
  2. NIMx = 22 mod 4 = 2
  3. Kombinasi fungsi iterasi: g1B dan g2A
  4. Tebakan awal: x₀ = 1.5, y₀ = 3.5
  5. Toleransi (ε): 0.000001
  6. Maksimum iterasi: 100
* **Metode yang Diimplementasikan:**

1. Iterasi Titik Tetap - Jacobi (g1A, g2B)
2. Iterasi Titik Tetap - Seidel (g1A, g2B)
3. Newton-Raphson
4. Secant

**FUNGSI ITERASI**

Kombinasi fungsi iterasi (berdasarkan NIMx=2):

Dari PPT:

(Terkadang bentuk bisa diubah menjadi tergantung konteks; untuk versi ini disesuaikan agar iteratif dan konvergen.)

**DEMONSTRASI PROGRAM**

**Bahasa Demo Pemrograman:** Python

**Struktur Program:**

* + 1. Fungsi Dasar

|  |
| --- |
| def f1(x, y):return x\*\*2 + x\*y - 10def f2(x, y): return y + 3\*x\*y\*\*2 - 57 |

* + 1. Fungsi Iterasi

|  |
| --- |
| def g1B(x, y):val = 10 - x\*yif val <= 0:raise ValueError("Nilai negatif di akar pada g1B.")return math.sqrt(val)def g2A(x, y):val = (57 - y) / (3\*x)if val <= 0:raise ValueError("Nilai negatif di akar pada g2A.") return math.sqrt(val) |

* + 1. Turunan untuk Newton-Raphson

|  |
| --- |
| def df1\_dx(x, y): return 2\*x + y  def df1\_dy(x, y): return x  def df2\_dx(x, y): return 3\*y\*\*2  def df2\_dy(x, y): return 1 + 6\*x\*y |

* + 1. Metode Iterasi Titik Tetap – Jacobi

|  |
| --- |
| def iterasi\_jacobi(g1, g2, x0, y0, epsilon=1e-6, max\_iter=100):  print("\n" + "="\*70)  print("METODE ITERASI TITIK TETAP - JACOBI")  print("="\*70)  print(f"{'Iter':<5} {'x':>10} {'y':>10} {'Δx':>12} {'Δy':>12}")  print("-"\*70)  x, y = x0, y0  print(f"{0:<5} {x:>10.6f} {y:>10.6f} {0:>12.6f} {0:>12.6f}")  for i in range(1, max\_iter + 1):  try:  x\_new = g1(x, y)  y\_new = g2(x, y)  except Exception as e:  print(f"\nERROR iterasi {i}: {e}")  return None, None, False, i  dx, dy = abs(x\_new - x), abs(y\_new - y)  print(f"{i:<5} {x\_new:>10.6f} {y\_new:>10.6f} {dx:>12.6f} {dy:>12.6f}")  if dx < epsilon and dy < epsilon:  print(f"\nKONVERGEN pada iterasi {i}")  print(f"x = {x\_new:.6f}, y = {y\_new:.6f}")  print(f"f1 = {f1(x\_new, y\_new):.6e}, f2 = {f2(x\_new, y\_new):.6e}")  return x\_new, y\_new, True, i  x, y = x\_new, y\_new  print("\nTidak konvergen setelah", max\_iter, "iterasi.")  return x, y, False, max\_iter |

* + 1. Metode Iterasi Titik Tetap – Seidel

|  |
| --- |
| def iterasi\_seidel(g1, g2, x0, y0, epsilon=1e-6, max\_iter=100):  print("\n" + "="\*70)  print("METODE ITERASI TITIK TETAP - SEIDEL (GAUSS-SEIDEL)")  print("="\*70)  print(f"{'Iter':<5} {'x':>10} {'y':>10} {'Δx':>12} {'Δy':>12}")  print("-"\*70)  x, y = x0, y0  print(f"{0:<5} {x:>10.6f} {y:>10.6f} {0:>12.6f} {0:>12.6f}")  for i in range(1, max\_iter + 1):  try:  x\_new = g1(x, y)  y\_new = g2(x\_new, y)  except Exception as e:  print(f"\nERROR iterasi {i}: {e}")  return None, None, False, i  dx, dy = abs(x\_new - x), abs(y\_new - y)  print(f"{i:<5} {x\_new:>10.6f} {y\_new:>10.6f} {dx:>12.6f} {dy:>12.6f}")  if dx < epsilon and dy < epsilon:  print(f"\nKONVERGEN pada iterasi {i}")  print(f"x = {x\_new:.6f}, y = {y\_new:.6f}")  print(f"f1 = {f1(x\_new, y\_new):.6e}, f2 = {f2(x\_new, y\_new):.6e}")  return x\_new, y\_new, True, i  x, y = x\_new, y\_new  print("\nTidak konvergen setelah", max\_iter, "iterasi.")  return x, y, False, max\_iter |

* + 1. Metode Newton – Raphson

|  |
| --- |
| def newton\_raphson(x0, y0, epsilon=1e-6, max\_iter=100):  print("\n" + "="\*70)  print("METODE NEWTON-RAPHSON")  print("="\*70)  print(f"{'Iter':<5} {'x':>10} {'y':>10} {'Δx':>12} {'Δy':>12}")  print("-"\*70)  x, y = x0, y0  print(f"{0:<5} {x:>10.6f} {y:>10.6f} {0:>12.6f} {0:>12.6f}")  for i in range(1, max\_iter + 1):  f1v, f2v = f1(x, y), f2(x, y)  J = df1\_dx(x, y)\*df2\_dy(x, y) - df1\_dy(x, y)\*df2\_dx(x, y)  if abs(J) < 1e-12:  print(f"\nDeterminan mendekati nol pada iterasi {i}")  return None, None, False, i  x\_new = x - (f1v\*df2\_dy(x, y) - f2v\*df1\_dy(x, y)) / J  y\_new = y - (f2v\*df1\_dx(x, y) - f1v\*df2\_dx(x, y)) / J  dx, dy = abs(x\_new - x), abs(y\_new - y)  print(f"{i:<5} {x\_new:>10.6f} {y\_new:>10.6f} {dx:>12.6f} {dy:>12.6f}")  if dx < epsilon and dy < epsilon:  print(f"\nKONVERGEN pada iterasi {i}")  print(f"x = {x\_new:.6f}, y = {y\_new:.6f}")  print(f"f1 = {f1(x\_new, y\_new):.6e}, f2 = {f2(x\_new, y\_new):.6e}")  return x\_new, y\_new, True, i  x, y = x\_new, y\_new  print("\nTidak konvergen setelah", max\_iter, "iterasi.")  return x, y, False, max\_iter |

* + 1. Metode Secant

|  |
| --- |
| def secant\_method(x0, y0, x1, y1, epsilon=1e-6, max\_iter=100):  print("\n" + "="\*70)  print("METODE SECANT")  print("="\*70)  print(f"{'Iter':<5} {'x':>10} {'y':>10} {'Δx':>12} {'Δy':>12}")  print("-"\*70)  x\_prev, y\_prev = x0, y0  x, y = x1, y1  print(f"{0:<5} {x\_prev:>10.6f} {y\_prev:>10.6f} {0:>12.6f} {0:>12.6f}")  print(f"{1:<5} {x:>10.6f} {y:>10.6f} {abs(x-x\_prev):>12.6f} {abs(y-y\_prev):>12.6f}")  for i in range(2, max\_iter + 1):  f1v, f2v = f1(x, y), f2(x, y)  f1p, f2p = f1(x\_prev, y\_prev), f2(x\_prev, y\_prev)  dx, dy = x - x\_prev, y - y\_prev  du\_dx = (f1v - f1p) / dx  du\_dy = (f1(x, y + 0.0001) - f1v) / 0.0001  dv\_dx = (f2v - f2p) / dx  dv\_dy = (f2(x, y + 0.0001) - f2v) / 0.0001  det = du\_dx \* dv\_dy - du\_dy \* dv\_dx  if abs(det) < 1e-10:  print(f"\nDeterminan mendekati nol pada iterasi {i}")  return None, None, False, i  x\_new = x - (f1v \* dv\_dy - f2v \* du\_dy) / det  y\_new = y + (f1v \* dv\_dx - f2v \* du\_dx) / det  dx, dy = abs(x\_new - x), abs(y\_new - y)  print(f"{i:<5} {x\_new:>10.6f} {y\_new:>10.6f} {dx:>12.6f} {dy:>12.6f}")  if dx < epsilon and dy < epsilon:  print(f"\nKONVERGEN pada iterasi {i}")  print(f"x = {x\_new:.6f}, y = {y\_new:.6f}")  print(f"f1 = {f1(x\_new, y\_new):.6e}, f2 = {f2(x\_new, y\_new):.6e}")  return x\_new, y\_new, True, i  x\_prev, y\_prev, x, y = x, y, x\_new, y\_new  print("\nTidak konvergen setelah", max\_iter, "iterasi.")  return x, y, False, max\_iter |

**LANGKAH - LANGKAH**

1. **Mengatur parameter** (x₀, y₀, ε, max\_iter)
2. **Menjalankan metode jacobi** dengan g1A dan g2B
3. **Menjalankan metode seidel** dengan g1A dan g2B
4. **Menjalankan metode Newton - Raphson**
5. **Menjalankan metode Secant**
6. **Membandingkan** performa dan konvergensi

**LINK GITHUB**

<https://github.com/fadianf/Fadia-Nur-Fatimah_21120123120022_Tugas-Sistem-Persamaan-Non-Linear_B-MetNum>

File: Fadia Nur Fatimah\_B-Metode Numerik.py

**ANALISIS DAN SOLUSI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Metode | Status | Iterasi | x | y | F1 | F2 |
| **Jacobi (g1A, g2B)** | Divergen | - | - | - | - | - |
| **Seidel (g1A, g2B)** | Konvergen | ~70 | 2.000 | 3.000 | 1e-5 | |  | | --- | |  |   1e-6 |
| **Newton-Raphson** | Konvergen | 4 | 2.000 | 3.000 | ~10⁻¹⁴ | |  | | --- | |  |   ~10⁻¹² |
| **Secant** | Konvergen | ~30 | 2.000 | 3.000 | ~10⁻⁸ | ~10⁻⁶ |

**Metode Iterasi Jacobi (g1A, g2B)**

Tabel Iterasi:

Iter x y Δx Δy

------------------------------------------------------

0 1.500000 3.500000 0.000000 0.000000

1 2.179449 3.448027 0.679449 0.051973

2 1.576452 2.861895 0.602998 0.586132

3 2.342725 3.383378 0.766273 0.521483

4 1.440026 2.762030 0.902699 0.621348

5 2.454100 3.543284 1.014074 0.781253

6 1.142116 2.694601 1.311984 0.848683

7 2.631056 3.981125 1.488940 1.286524

Hasil: ERROR iterasi 8: Nilai negatif di akar pada g1B.

Analisis Hasil:

1. Iterasi mulai dari menunjukkan nilai yang **bergantian besar-besar** (x naik-turun, y naik-turun) hingga iterasi ke-7.
2. Iterasi ke-8 mengalami **error: nilai negatif di akar,** menandakan **metode gagal** karena fungsi iterasi Jacobi tidak stabil untuk pilihan ini.

Analisis Syarat Konvergensi:

1. Fungsi g1B dan g2A memiliki bentuk akar kuadrat yang hanya valid untuk domain positif.
2. Saat nilai x\*y > 10, g1B tidak lagi terdefinisi.
3. Berdasarkan iterasi di atas, x dan y membesar cukup cepat → domain pelanggaran terjadi di iterasi ke-8.
4. Syarat konvergensi titik tetap juga tidak terpenuhi karena nilai turunan g1B dan g2A > 1 dalam interval tersebut.

Kecepatan:

Sangat lambat dan tidak stabil → iterasi berhenti akibat error.

Kesimpulan:

1. Jacobi tidak cocok untuk sistem ini dengan .
2. Metode ini sensitif terhadap pemilihan fungsi iterasi dan titik awal.

**Metode Iterasi Seidel (g1A, g2B)**

Tabel Iterasi:

Iter x y Δx Δy

------------------------------------------------------

0 1.500000 3.500000 0.000000 0.000000

1 2.179449 2.860506 0.679449 0.639494

2 1.940534 3.049551 0.238916 0.189045

3 2.020456 2.983405 0.079922 0.066146

4 1.993028 3.005704 0.027428 0.022300

5 2.002385 2.998054 0.009357 0.007650

6 1.999185 3.000666 0.003200 0.002611

7 2.000279 2.999773 0.001094 0.000893

8 1.999905 3.000078 0.000374 0.000305

9 2.000033 2.999973 0.000128 0.000104

10 1.999989 3.000009 0.000044 0.000036

11 2.000004 2.999997 0.000015 0.000012

12 1.999999 3.000001 0.000005 0.000004

13 2.000000 3.000000 0.000002 0.000001

14 2.000000 3.000000 0.000001 0.000000

Hasil: KONVERGEN pada iterasi 14

x = 2.000000, y = 3.000000

f1 = -8.156310e-07, f2 = 4.869573e-07

Analisis Hasil:

1. Iterasi konvergen dengan cepat ke
2. Δx dan Δy menurun secara **eksponensial**:
   * + Δx dari 0.679 → 0.000001 dalam 14 iterasi
     + Δy dari 0.639 → 0.000000

Analisis Konvergensi:

1. Metode **Gauss-Seidel konvergen** karena ia memanfaatkan **nilai terbaru pada iterasi yang sama**, meningkatkan stabilitas.
2. Sistem memenuhi **kriteria konvergensi** untuk metode ini.

Kecepatan:

1. **Lebih cepat dari Jacobi**, hanya 14 iterasi untuk presisi hingga 10^-6.
2. Δ menurun hampir eksponensial → konvergensi cepat.

Kesimpulan:

1. Seidel efektif, stabil, dan cepat untuk sistem ini.
2. Cocok untuk iterasi titik tetap bila Jacobi gagal.

**Metode Iterasi Newton-Raphson**

Tabel Iterasi:

Iter x y Δx Δy

------------------------------------------------------

0 1.500000 3.500000 0.000000 0.000000

1 2.036029 2.843875 0.536029 0.656125

2 1.998701 3.002289 0.037328 0.158413

3 2.000000 2.999999 0.001299 0.002289

4 2.000000 3.000000 0.000000 0.000001

Hasil: KONVERGEN pada iterasi 4

x = 2.000000, y = 3.000000

f1 = 1.065814e-14, f2 = 2.238210e-12

Analisis Hasil:

1. Iterasi konvergen **hanya dalam 4 iterasi** ke solusi
2. Δx dan Δy menurun drastis pada tiap iterasi.
3. Error fungsi mendekati nol (10^-12 → 10^-14).

Analisis Konvergensi:

1. Newton-Raphson konvergensi kuadrat: setiap iterasi mengurangi error secara drastis.
2. Sistem diferensiable, Jacobian nonsingular → syarat terpenuhi.

Kecepatan:

1. **Paling cepat di antara semua metode yang konvergen**.
2. Memerlukan perhitungan Jacobian setiap iterasi → lebih kompleks komputasinya.

Kesimpulan:

1. Newton-Raphson sangat efektif, cepat, dan presisi tinggi.
2. Cocok untuk sistem nonlinear kecil hingga menengah.

**Metode Iterasi Secant**

Tabel Iterasi:

Iter x y Δx Δy

------------------------------------------------------

0 1.500000 3.500000 0.000000 0.000000

1 1.400000 3.600000 0.100000 0.100000

2 2.043448 3.444828 0.643448 0.155172

3 1.984422 3.039461 0.059026 0.405367

4 2.002734 2.860112 0.018312 0.179348

5 1.999163 2.967998 0.003571 0.107885

6 2.000126 3.028716 0.000963 0.060718

7 1.999984 3.009044 0.000143 0.019673

8 1.999997 2.998129 0.000014 0.010915

9 1.999999 3.001324 0.000002 0.003195

10 2.000000 2.998702 0.000001 0.002621

11 2.000000 3.000669 0.000000 0.001967

12 2.000000 2.999232 0.000000 0.001437

13 2.000000 3.000746 0.000000 0.001514

14 2.000000 2.999562 0.000000 0.001184

15 2.000000 3.000969 0.000000 0.001407

16 2.000000 2.999898 0.000000 0.001070

17 2.000000 3.001213 0.000000 0.001315

18 2.000000 3.000291 0.000000 0.000921

19 2.000000 3.001302 0.000000 0.001011

20 2.000000 3.000580 0.000000 0.000722

21 2.000000 3.001259 0.000000 0.000679

22 2.000000 3.000681 0.000000 0.000578

23 2.000000 3.001376 0.000000 0.000695

24 2.000000 3.001073 0.000000 0.000302

25 2.000000 2.999664 0.000000 0.001410

26 2.000000 3.000856 0.000000 0.001192

27 2.000000 2.999702 0.000000 0.001153

28 2.000000 3.001087 0.000000 0.001385

29 2.000000 3.000059 0.000000 0.001029

30 2.000000 3.001287 0.000000 0.001228

31 2.000000 3.000436 0.000000 0.000850

32 2.000000 3.001294 0.000000 0.000858

33 2.000000 3.000625 0.000000 0.000669

34 2.000000 3.001321 0.000000 0.000696

35 2.000000 3.000845 0.000000 0.000476

36 2.000000 3.001080 0.000000 0.000235

37 2.000000 3.000209 0.000000 0.000871

38 2.000000 3.001478 0.000000 0.001269

39 2.000000 3.000823 0.000000 0.000654

40 2.000000 3.000802 0.000000 0.000021

41 2.000000 2.994822 0.000000 0.005980

42 2.000000 2.995730 0.000000 0.000908

43 2.000000 3.017463 0.000000 0.021733

44 2.000000 3.014009 0.000000 0.003454

45 2.000001 2.937939 0.000001 0.076069

46 2.000000 2.949345 0.000000 0.011406

47 2.000009 3.236011 0.000009 0.286666

48 2.000003 3.194513 0.000006 0.041498

49 2.000149 2.018528 0.000146 1.175985

50 2.000047 2.180298 0.000102 0.161770

51 2.002781 7.341025 0.002734 5.160727

52 2.000842 6.657020 0.001939 0.684006

53 2.070783 -22.069878 0.069941 28.726898

54 2.017313 -18.926280 0.053469 3.143598

55 7.312898 361.483817 5.295585 380.410097

56 2.186574 355.578913 5.126324 5.904904

57 0.138739 352.189956 2.047835 3.388957

58 0.058330 279.503476 0.080409 72.686480

59 0.073365 66.958050 0.015035 212.545426

60 0.079087 195.314943 0.005722 128.356892

61 0.082731 41.534508 0.003644 153.780435

62 0.084105 175.527611 0.001374 133.993103

63 0.084948 36.194812 0.000843 139.332800

64 0.085274 170.860710 0.000326 134.665899

65 0.085473 34.971275 0.000199 135.889436

66 0.085550 169.759824 0.000078 134.788549

67 0.085598 34.685781 0.000047 135.074043

68 0.085616 169.500376 0.000018 134.814596

69 0.085627 34.619095 0.000011 134.881282

70 0.085632 169.439481 0.000004 134.820386

71 0.085634 34.603719 0.000003 134.835762

72 0.085636 169.425436 0.000001 134.821717

73 0.085636 34.600389 0.000001 134.825047

74 0.085636 169.422445 0.000000 134.822057

75 0.085637 34.599886 0.000000 134.822559

76 0.085637 169.422062 0.000000 134.822176

77 0.085637 34.600048 0.000000 134.822014

78 0.085637 169.422295 0.000000 134.822247

79 0.085637 34.600365 0.000000 134.821929

80 0.085637 169.422672 0.000000 134.822307

81 0.085637 34.600719 0.000000 134.821953

82 0.085637 169.423084 0.000000 134.822365

83 0.085637 34.601082 0.000000 134.822002

84 0.085637 169.423504 0.000000 134.822422

85 0.085637 34.601446 0.000000 134.822057

86 0.085637 169.423925 0.000000 134.822479

87 0.085637 34.601811 0.000000 134.822114

88 0.085637 169.424348 0.000000 134.822536

89 0.085637 34.602176 0.000000 134.822171

90 0.085637 169.424770 0.000000 134.822593

91 0.085637 34.602542 0.000000 134.822228

92 0.085637 169.425192 0.000000 134.822651

93 0.085637 34.602907 0.000000 134.822285

94 0.085637 169.425614 0.000000 134.822708

95 0.085637 34.603272 0.000000 134.822343

96 0.085637 169.426037 0.000000 134.822765

97 0.085637 34.603637 0.000000 134.822400

98 0.085637 169.426459 0.000000 134.822822

99 0.085637 34.604002 0.000000 134.822457

100 0.085637 169.426882 0.000000 134.822879

Hasil: Tidak konvergen setelah 100 iterasi.

Analisis Hasil:

1. terasi tidak stabil dan **tidak konvergen**.
2. Nilai x tetap relatif stabil, tetapi yang **melambung-lambung** hingga ratusan → divergen.
3. Setelah 100 iterasi, masih jauh dari solusi yang benar.

Analisis Konvergensi:

1. Secant multivariat **sangat sensitif** terhadap titik awal.
2. Sistem ini kemungkinan **tidak sesuai** untuk iterasi Secant standar dua variabel, terutama bila fungsi nonlinear kompleks.

Kecepatan:

Lambat dan tidak stabil → gagal mencapai solusi.

Kesimpulan:

1. Secant tidak cocok untuk sistem ini.
2. Memerlukan modifikasi atau metode alternatif (misal: Broyden) untuk multivariabel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metode | Konvergensi | Iterasi yang dibutuhkan | Kecepatan | Catatan |
| Jacobi | Gagal | - | Lambat | Sensitif terhadap fungsi iterasi |
| Seidel | Ya | 14 | Cepat | Stabil, nilai terbaru dipakai |
| Newton-Raphson | Ya | 4 | Sangat cepat | Presisi tinggi, memerlukan Jacobian |
| Secant | Tidak | 100+ | Tidak stabil | Divergen untuk sistem ini |

RINGKASAN HASIL

======================================================================

Metode x y Status Iterasi

----------------------------------------------------------------------

Jacobi N/A N/A Divergen 8

Seidel 2.000000 3.000000 Konvergen 14

Newton-Raphson 2.000000 3.000000 Konvergen 4

Secant 0.085637169.426882 Divergen 100

======================================================================

**KESIMPULAN**

Berdasarkan implementasi dan analisis sistem persamaan nonlinear

dengan kombinasi fungsi iterasi dan (NIMx = 1), dapat disimpulkan:

1. Pengaruh Pemilihan Fungsi Iterasi

Kombinasi dan . Menghasilkan hasil berbeda pada setiap metode:

* Metode Jacobi: DIVERGEN pada iterasi ke-8 karena syarat konvergensi tidak terpenuhi ()
* Metode Seidel: KONVERGEN dalam 14 iterasi, menunjukkan Seidel lebih robust dibanding Jacobi untuk fungsi iterasi yang kurang stabil

Temuan: Metode Seidel mampu menangani kombinasi fungsi yang membuat Jacobi divergen, dengan konvergensi lebih cepat daripada Jacobi (yang gagal).

2.Perbandingan Kecepatan Konvergensi

Urutan kecepatan dari hasil eksperimen:

|  |  |  |
| --- | --- | --- |
| Metode | Iterasi | Kecepatan Relatif |
| Newton-Raphson | 4 | 100% (Tercepat) |
| Seidel | 14 | 28.6% |
| Jacobi | - | Divergen |
| Secant | 100 | 4% |

Newton-Raphson 3.5x lebih cepat dari Seidel dan jauh lebih cepat dari Secant, membuktikan keunggulan konvergensi kuadratik.

3. Akurasi Solusi

Semua metode yang konvergen mencapai solusi dengan tingkat akurasi berbeda:

* Newton-Raphson: Error ~10⁻¹² → akurasi tertinggi, mendekati presisi mesin
* Seidel: Error ~10⁻⁶ → sesuai toleransi iterasi yang ditetapkan

4. Trade-off Implementasi

Terdapat trade-off antara kecepatan, akurasi, dan kompleksitas implementasi:

* Newton-Raphson: Tercepat dan akurat, tetapi memerlukan perhitungan Jacobian (turunan)
* Seidel: Implementasi lebih sederhana dan stabil, tetapi iterasi lebih banyak dibutuhkan
* Jacobi: Sederhana, tetapi tidak robust untuk fungsi ini
* Secant: Tidak konvergen untuk sistem ini

5.Validasi Teori

Hasil eksperimen memvalidasi teori metode numerik:

* Syarat konvergensi iterasi titik tetap sangat penting (Jacobi divergen)
* Metode Seidel lebih cepat dan robust daripada Jacobi
* Newton-Raphson memiliki konvergensi kuadratik yang sangat cepat
* Secant multivariat dapat gagal jika fungsi nonlinear kompleks

6. Rekomendasi

Untuk sistem persamaan nonlinear seperti ini:

* Gunakan Newton-Raphson jika turunan mudah dihitung dan diperlukan solusi cepat dan akurat
* Gunakan Seidel jika komputasi terbatas dan iterasi lebih banyak dapat diterima
* Hindari Jacobi untuk fungsi iterasi yang tidak memenuhi syarat konvergensi
* Secant kurang direkomendasikan untuk kasus ini karena mudah divergen