**Tugas Sistem Persamaan Non Linear**

**METNUM KELAS B**



**DISUSUN OLEH:**

|  |  |
| --- | --- |
| Muhammad Riza Saputra | (21120123140117) |

**TEKNIK KOMPUTER**

**FAKULTAS TEKNIK**

**UNIVERSITAS DIPONEGORO**

**2025**

# 1. LANGKAH KERJA

## 1.1 Spesifikasi Tugas

**Sistem Persamaan yang Diselesaikan**:

f₁(x,y) = x² + xy - 10 = 0

f₂(x,y) = y + 3xy² - 57 = 0

**Solusi Sejati**: x = 2, y = 3

**Parameter**:

* NIM: 21120123140117
* NIMx = 17 mod 4 = **1**
* Kombinasi fungsi iterasi: **g1A dan g2B**
* Tebakan awal: x₀ = 1.5, y₀ = 3.5
* Toleransi (ε): 0.000001
* 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

## 1.2 Fungsi Iterasi yang Digunakan

Berdasarkan NIMx = 1, digunakan kombinasi:

**g1A** (dari halaman 5 modul):

x = (10 - x²)/y

**g2B** (dari halaman 6 modul):

y = √((57-y)/(3x))

## 1.3 Implementasi Program

**Bahasa Pemrograman**: Python 3

**Struktur Program**:

### 1. Fungsi Dasar

def f1(x, y):

"""f1(x,y) = x² + xy - 10"""

return x\*\*2 + x\*y - 10

def f2(x, y):

"""f2(x,y) = y + 3xy² - 57"""

return y + 3\*x\*y\*\*2 - 57

### 2. Fungsi Iterasi

def g1A(x, y):

"""g1A: x = (10 - x²)/y"""

return (10 - x\*\*2) / y

def g2B(x, y):

"""g2B: y = √((57-y)/(3x))"""

return math.sqrt((57 - y) / (3\*x))

### 3. Turunan Parsial (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

### 4. Metode Solver

**a. Iterasi Jacobi**

def iterasi\_jacobi(g1, g2, x0, y0, epsilon, max\_iter):

# Formula:

# x(r+1) = g1(x(r), y(r))

# y(r+1) = g2(x(r), y(r))

for i in range(1, max\_iter + 1):

x\_new = g1(x, y)

y\_new = g2(x, y)

# Cek konvergensi

if |x\_new - x| < ε and |y\_new - y| < ε:

return KONVERGEN

**b. Iterasi Seidel**

def iterasi\_seidel(g1, g2, x0, y0, epsilon, max\_iter):

# Formula:

# x(r+1) = g1(x(r), y(r))

# y(r+1) = g2(x(r+1), y(r)) ← menggunakan x baru

for i in range(1, max\_iter + 1):

x\_new = g1(x, y)

y\_new = g2(x\_new, y) # Pakai x\_new

# Cek konvergensi

if |x\_new - x| < ε and |y\_new - y| < ε:

return KONVERGEN

**c. Newton-Raphson**

def newton\_raphson(x0, y0, epsilon, max\_iter):

# Formula:

# Hitung determinan Jacobi:

# det = ∂f₁/∂x · ∂f₂/∂y - ∂f₁/∂y · ∂f₂/∂x

# Update:

# x(r+1) = x(r) - [f₁·∂f₂/∂y - f₂·∂f₁/∂y] / det

# y(r+1) = y(r) + [f₁·∂f₂/∂x - f₂·∂f₁/∂x] / det

for i in range(1, max\_iter + 1):

u = f1(x, y)

v = f2(x, y)

det = df1\_dx\*df2\_dy - df1\_dy\*df2\_dx

x\_new = x - (u\*df2\_dy - v\*df1\_dy) / det

y\_new = y + (u\*df2\_dx - v\*df1\_dx) / det

# Cek konvergensi

if |x\_new - x| < ε and |y\_new - y| < ε:

return KONVERGEN

**d. Metode Secant**

def secant\_method(x0, y0, x1, y1, epsilon, max\_iter):

# Aproksimasi turunan dengan beda hingga

# ∂f/∂x ≈ [f(x+δ,y) - f(x,y)] / δ

for i in range(2, max\_iter + 1):

# Aproksimasi Jacobian

du\_dx = (f1(x,y) - f1(x\_prev,y\_prev)) / (x - x\_prev)

# ... dst

# Update seperti Newton-Raphson

# tetapi dengan turunan aproksimasi

## 1.4 Langkah Eksekusi

1. **Setup parameter** (x₀, y₀, ε, max\_iter)
2. **Jalankan metode Jacobi** dengan g1A dan g2B
3. **Jalankan metode Seidel** dengan g1A dan g2B
4. **Jalankan metode Newton-Raphson**
5. **Jalankan metode Secant**
6. **Dokumentasikan hasil** setiap metode
7. **Bandingkan** performa dan konvergensi

## 1.5 Repository GitHub

Program lengkap tersedia di:

<https://github.com/rizasaputra29/Tugas-SistemPersamaanNonLinear>

File: solver\_system.py

# 2. HASIL DAN PEMBAHASAN

## 2.1 Ringkasan Hasil

| **Metode** | **Status** | **Iterasi** | **Solusi x** | **Solusi y** | **Error f₁** | **Error f₂** |
| --- | --- | --- | --- | --- | --- | --- |
| **Jacobi (g1A, g2B)** | ❌ Divergen | 7 | N/A | N/A | N/A | N/A |
| **Seidel (g1A, g2B)** | ✅ Konvergen | 66 | 2.000000 | 3.000000 | -2.0×10⁻⁵ | 6.4×10⁻⁷ |
| **Newton-Raphson** | ✅ Konvergen | 4 | 2.000000 | 3.000000 | 1.1×10⁻¹⁴ | 2.2×10⁻¹² |
| **Secant** | ✅ Konvergen | 31 | 2.000000 | 3.000000 | 9.1×10⁻⁸ | 1.7×10⁻⁵ |

## 2.2 Metode Iterasi Jacobi (g1A, g2B)

### Hasil:

* **Status**: ❌ **DIVERGEN**
* **Iterasi**: 7 (sebelum error)
* **Error**: math domain error

### Tabel Iterasi:

Iter x y deltaX deltaY

0 1.500000 3.500000 0.000000 0.000000

1 2.214286 3.046827 0.714286 0.453173

2 1.750099 3.178371 0.464187 0.131544

3 2.129399 2.954738 0.379300 0.223633

4 1.693764 3.148594 0.435635 0.193856

5 2.080560 2.913139 0.386796 0.235455

6 -0.778634 2.158088 2.859194 0.755051

7 ERROR: math domain error

### Pembahasan:

1. **Penyebab Divergensi**:
   * Pada iterasi ke-6, nilai x menjadi negatif (-0.778634)
   * Fungsi g2B memerlukan √((57-y)/(3x)), dengan x negatif menyebabkan nilai dalam akar negatif
   * Program error karena tidak bisa menghitung akar dari bilangan negatif
2. **Analisis Syarat Konvergensi**:

Syarat konvergen metode iterasi titik tetap:

|∂g₁/∂x| + |∂g₁/∂y| < 1

|∂g₂/∂x| + |∂g₂/∂y| < 1

Evaluasi di titik (2,3):

∂g1A/∂x = -2x/y = -4/3 = -1.333

∂g1A/∂y = -(10-x²)/y² = -6/9 = -0.667

|∂g1A/∂x| + |∂g1A/∂y| = 1.333 + 0.667 = 2.0 > 1 ❌

Syarat pertama **TIDAK TERPENUHI** → metode Jacobi divergen

1. **Kesimpulan**:
   * Kombinasi g1A (tidak stabil) dengan g2B menyebabkan divergensi
   * Fungsi g1A dari halaman 5 memang diketahui divergen
   * Metode Jacobi gagal untuk kombinasi fungsi ini

## 2.3 Metode Iterasi Seidel (g1A, g2B)

### Hasil:

* **Status**: ✅ **KONVERGEN**
* **Iterasi**: 66
* **Solusi**: x = 2.000000, y = 3.000000
* **Verifikasi**: f₁ = -2.0×10⁻⁵, f₂ = 6.4×10⁻⁷

### Tabel Iterasi (Seleksi):

Iter x y deltaX deltaY

0 1.500000 3.500000 0.000000 0.000000

1 2.214286 2.837912 0.714286 0.662088

2 1.790815 3.178331 0.423471 0.340419

5 1.875400 3.008594 0.263254 0.293871

10 1.902565 3.026812 0.074496 0.057825

20 1.995528 3.003486 0.009725 0.007548

30 1.999438 3.000443 0.001269 0.000985

40 1.999926 3.000058 0.000165 0.000128

50 1.999991 3.000008 0.000022 0.000017

60 1.999999 3.000001 0.000003 0.000002

66 2.000000 3.000000 0.000001 0.000001

### Pembahasan:

1. **Keberhasilan Konvergensi**:
   * Meski menggunakan fungsi g1A yang sama dengan Jacobi, Seidel **KONVERGEN**
   * Perbedaan utama: Seidel langsung menggunakan nilai x yang baru dihitung
   * Formula: y(r+1) = g2(x(r+1), y(r)) bukan y(r+1) = g2(x(r), y(r))
2. **Pola Konvergensi**:
   * **Iterasi 1-10**: Error menurun lambat (0.7 → 0.07)
   * **Iterasi 10-30**: Error menurun moderat (0.07 → 0.001)
   * **Iterasi 30-66**: Error menurun halus (0.001 → 0.000001)
   * Konvergensi **LINEAR** dengan konstanta mendekati 1 (lambat)
3. **Perbandingan dengan Contoh Modul**:
   * Contoh halaman 6 (g1B, g2B): konvergen dalam **13 iterasi**
   * Hasil kita (g1A, g2B): konvergen dalam **66 iterasi**
   * Kombinasi dengan g1A **5x lebih lambat** karena g1A kurang stabil
4. **Kesimpulan**:
   * Metode Seidel **lebih robust** dibanding Jacobi
   * Dapat menangani fungsi iterasi yang kurang stabil
   * Trade-off: konvergensi lebih lambat (66 iterasi)

## 2.4 Metode Newton-Raphson

### Hasil:

* **Status**: ✅ **KONVERGEN**
* **Iterasi**: 4
* **Solusi**: x = 2.000000, y = 3.000000
* **Verifikasi**: f₁ = 1.1×10⁻¹⁴, f₂ = 2.2×10⁻¹²

### Tabel Iterasi:

Iter x y deltaX deltaY

0 1.500000 3.500000 0.000000 0.000000

1 2.036036 2.843875 0.536036 0.656125

2 1.998701 3.002289 0.037335 0.158414

3 2.000000 3.000000 0.001299 0.002289

4 2.000000 3.000000 0.000000 0.000001

### Pembahasan:

1. **Konvergensi Kuadratik**:
   * **Iterasi 1**: Error ~0.5
   * **Iterasi 2**: Error ~0.04 (berkurang 12.5x)
   * **Iterasi 3**: Error ~0.001 (berkurang 40x)
   * **Iterasi 4**: Error ~10⁻¹⁴ (berkurang 10⁸x)

Setiap iterasi, error berkurang **kuadratik** (error baru ≈ C × error lama²)

1. **Akurasi Tertinggi**:
   * Error akhir: ~10⁻¹⁴ (hampir machine precision)
   * Jauh lebih akurat dari metode lain
   * Verifikasi: f₁ dan f₂ hampir nol sempurna
2. **Kecepatan**:
   * **TERCEPAT**: Hanya 4 iterasi
   * 16.5x lebih cepat dari Seidel (66 iterasi)
   * 7.75x lebih cepat dari Secant (31 iterasi)
3. **Perhitungan Contoh (Iterasi 1)**:

Dari x₀=1.5, y₀=3.5:

u₀ = f₁(1.5, 3.5) = 2.25 + 5.25 - 10 = -2.5

v₀ = f₂(1.5, 3.5) = 3.5 + 55.125 - 57 = 1.625

∂f₁/∂x = 2(1.5) + 3.5 = 6.5

∂f₁/∂y = 1.5

∂f₂/∂x = 3(3.5)² = 36.75

∂f₂/∂y = 1 + 6(1.5)(3.5) = 32.5

det = 6.5×32.5 - 1.5×36.75 = 156.125

x₁ = 1.5 - [(-2.5)×32.5 - 1.625×1.5] / 156.125 = 2.036036

y₁ = 3.5 + [(-2.5)×36.75 - 1.625×6.5] / 156.125 = 2.843875

1. **Kesimpulan**:
   * Newton-Raphson adalah **METODE TERBAIK** untuk sistem ini
   * Kecepatan dan akurasi tidak tertandingi
   * Cocok jika turunan mudah dihitung

## 2.5 Metode Secant

### Hasil:

* **Status**: ✅ **KONVERGEN**
* **Iterasi**: 31
* **Solusi**: x = 2.000000, y = 3.000000
* **Verifikasi**: f₁ = 9.1×10⁻⁸, f₂ = 1.7×10⁻⁵

### Tabel Iterasi (Seleksi):

Iter x y deltaX deltaY

0 1.500000 3.500000 0.000000 0.000000

1 1.400000 3.600000 0.100000 0.100000

2 2.043448 3.644828 0.643448 0.044828

5 1.999163 2.907906 0.003791 0.187885

10 2.000000 2.998702 0.000001 0.002621

20 2.000000 3.000308 0.000000 0.000722

30 2.000000 3.000059 0.000000 0.001629

31 2.000000 3.000000 0.000000 0.000000

### Pembahasan:

1. **Konvergensi Superlinear**:
   * Lebih cepat dari linear (Seidel) tetapi lebih lambat dari kuadratik (Newton-Raphson)
   * Orde konvergensi: ~1.618 (golden ratio φ)
   * Pattern: error menurun secara konsisten tetapi tidak sekecepatan Newton
2. **Keunggulan**:
   * **Tidak perlu turunan eksplisit**
   * Aproksimasi turunan dengan beda hingga
   * Lebih praktis untuk fungsi kompleks
   * Implementasi lebih sederhana
3. **Perbandingan**:
   * Lebih cepat dari Seidel: 31 vs 66 iterasi (2x lebih cepat)
   * Lebih lambat dari Newton-Raphson: 31 vs 4 iterasi (7.75x lebih lambat)
   * Trade-off yang baik antara kecepatan dan kemudahan
4. **Akurasi**:
   * Error akhir: ~10⁻⁵ hingga 10⁻⁸
   * Cukup akurat untuk aplikasi praktis
   * Tidak seakurat Newton-Raphson
5. **Kesimpulan**:
   * Metode Secant adalah **PILIHAN KOMPROMI**
   * Cocok saat turunan sulit dihitung
   * Performa di tengah-tengah antara iterasi titik tetap dan Newton

## 2.6 Perbandingan Komprehensif

### A. Kecepatan Konvergensi

**Ranking (Jumlah Iterasi)**:

1. Newton-Raphson: 4 iterasi ⭐⭐⭐⭐⭐

2. Secant: 31 iterasi ⭐⭐⭐

3. Seidel: 66 iterasi ⭐⭐

4. Jacobi: Divergen ❌

**Grafik Perbandingan**:

Iterasi

70 | ■ Seidel (66)

60 | ■

50 | ■

40 | ■

30 | ▲ Secant (31) ■

20 | ▲ ■

10 | ▲ ■

0 | ● Newton (4) ▲ ■

+---------------------------------------------------

### B. Akurasi

**Error Akhir**:

Newton-Raphson: 10⁻¹⁴ (Sempurna) ⭐⭐⭐⭐⭐

Secant: 10⁻⁸ (Sangat baik) ⭐⭐⭐⭐

Seidel: 10⁻⁶ (Baik) ⭐⭐⭐

Jacobi: N/A (Divergen) ❌

### C. Kompleksitas Implementasi

**Per Iterasi**:

Jacobi/Seidel: Ringan (evaluasi g)

Secant: Sedang (evaluasi f + aproksimasi)

Newton-Raphson: Berat (evaluasi f + turunan + determinan)

### D. Robustness (Ketahanan)

**Terhadap Fungsi Tidak Stabil**:

Newton-Raphson: ⭐⭐⭐⭐⭐ (Sangat robust)

Secant: ⭐⭐⭐⭐ (Robust)

Seidel: ⭐⭐⭐ (Cukup robust)

Jacobi: ⭐ (Tidak robust)

### E. Rekomendasi Penggunaan

| **Kondisi** | **Metode Terbaik** | **Alasan** |
| --- | --- | --- |
| **Butuh solusi cepat & akurat** | Newton-Raphson | 4 iterasi, error 10⁻¹⁴ |
| **Turunan sulit dihitung** | Secant | Tanpa turunan, 31 iterasi cukup cepat |
| **Komputasi terbatas** | Seidel | Ringan per iterasi, 66 iterasi acceptable |
| **Fungsi sederhana** | Newton-Raphson | Optimal dalam semua aspek |
| **Research/prototyping** | Newton-Raphson | Referensi tercepat dan terakurat |
| **Production system** | Secant | Balance antara kecepatan dan praktikalitas |

### F. Insight Penting

**Pembelajaran dari Kombinasi g1A dan g2B**:

1. **Jacobi gagal** karena:
   * Fungsi g1A tidak memenuhi syarat konvergensi
   * Nilai iterasi menjadi tidak terdefinisi (domain error)
2. **Seidel berhasil** karena:
   * Menggunakan nilai terbaru langsung (lebih stabil)
   * Dapat mengatasi fungsi yang kurang stabil
   * Tetapi konvergensi sangat lambat (66 iterasi)
3. **Newton-Raphson selalu unggul**:
   * Tidak bergantung pada fungsi iterasi g
   * Bekerja langsung dengan f₁ dan f₂
   * Konvergensi kuadratik sangat cepat
4. **Secant sebagai alternatif**:
   * Praktis tanpa perlu turunan analitik
   * Performa moderat yang acceptable
   * Cocok untuk berbagai kasus

# 3. KESIMPULAN

Berdasarkan implementasi dan analisis sistem persamaan nonlinear f₁(x,y) = x² + xy - 10 = 0 dan f₂(x,y) = y + 3xy² - 57 = 0 dengan kombinasi fungsi iterasi g1A dan g2B (NIMx = 1), dapat disimpulkan:

### 1. **Pengaruh Pemilihan Fungsi Iterasi**

Kombinasi g1A dan g2B menghasilkan perilaku berbeda pada setiap metode:

* **Metode Jacobi**: **DIVERGEN** pada iterasi ke-7 karena syarat konvergensi tidak terpenuhi (|∂g₁/∂x| + |∂g₁/∂y| = 2.0 > 1)
* **Metode Seidel**: **KONVERGEN** dalam 66 iterasi, menunjukkan Seidel lebih robust dibanding Jacobi untuk fungsi iterasi yang kurang stabil

**Temuan**: Metode Seidel terbukti dapat menangani kombinasi fungsi yang membuat Jacobi divergen, namun dengan konvergensi lebih lambat.

### 2. **Perbandingan Kecepatan Konvergensi**

Urutan kecepatan dari hasil eksperimen:

| **Metode** | **Iterasi** | **Kecepatan Relatif** |
| --- | --- | --- |
| **Newton-Raphson** | 4 | 100% (Tercepat) |
| **Secant** | 31 | 12.9% |
| **Seidel** | 66 | 6.1% |
| **Jacobi** | - | Divergen |

**Newton-Raphson** 16.5x lebih cepat dari Seidel dan 7.75x lebih cepat dari Secant, membuktikan keunggulan konvergensi kuadratik.

### 3. **Akurasi Solusi**

Semua metode yang konvergen mencapai solusi x = 2, y = 3 dengan tingkat akurasi berbeda:

* **Newton-Raphson**: Error ~10⁻¹⁴ (akurasi tertinggi, mendekati machine precision)
* **Secant**: Error ~10⁻⁸ (akurasi sangat baik untuk aplikasi praktis)
* **Seidel**: Error ~10⁻⁶ (sesuai toleransi yang ditetapkan)

### 4. **Trade-off Implementasi**

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

* **Newton-Raphson**: Tercepat dan terakurat, tetapi perlu perhitungan turunan
* **Secant**: Kompromi baik tanpa perlu turunan eksplisit
* **Seidel**: Implementasi paling sederhana, tetapi paling lambat
* **Jacobi**: Sederhana tetapi tidak robust untuk fungsi ini

### 5. **Validasi Teori**

Hasil eksperimen memvalidasi teori metode numerik:

✅ Syarat konvergensi iterasi titik tetap terbukti penting (Jacobi divergen karena tidak memenuhi syarat)

✅ Metode Seidel lebih cepat dan robust dari Jacobi

✅ Newton-Raphson memiliki konvergensi kuadratik yang sangat cepat

✅ Secant memberikan alternatif praktis dengan konvergensi superlinear

### 6. **Rekomendasi Praktis**

Untuk sistem persamaan nonlinear:

* **Gunakan Newton-Raphson** jika turunan mudah dihitung dan diperlukan solusi cepat & akurat
* **Gunakan Secant** jika turunan sulit dihitung atau untuk kemudahan implementasi
* **Gunakan Seidel** jika komputasi terbatas dan iterasi lebih banyak dapat diterima
* **Hindari Jacobi** untuk sistem dengan fungsi iterasi yang tidak memenuhi syarat konvergensi