LAPORAN TUGAS BESAR CODING
MATA KULIAH METODE NUMERIS
SEMESTER GASAL 2023/2024

Disusun oleh:
Ryan Krishandi Lukito
22/497249/TK/54488
Teknologi Informasi

NEWTON RHAPSON

A. Pendahuluan

1. Permasalahan yang Diangkat

Pada studi keinsinyuran, ada banyak sekali permasalahan yang tidak bisa dipecahkan menggunakan metode tradisional. Bahkan, beberapa persamaan matematis memerlukan bantuan komputasi komputer untuk menemukan jawaban yang konvergen. Salah satu metode yang bisa digunakan adalah Newton-Rhapson. Newton-Rhapson merupakan salah satu metode perhitungan matematis untuk menyelesaikan persamaan tak linier secara numerik. Metode ini akan memberikan hasil penghitungan yang mendekati hasil asli atau bahkan sama dengan hasil asli. Perlu diperhatikan bahwa toleransi eror memegang peranan penting dalam hasil yang akan didapatkan.

Untuk menunjukkan pemanfaatan Newton-Rhapson secara riil, pada laporan ini Newton-Rhapson akan dimanfaatkan untuk menghitung volume molal ideal yang didasari pada persamaan Van Der Waals. Dalam sebuah proses kimia, jumlah molal ideal dari sebuah zat bisa dihitung mrenggunakan pertsamaan sebagai berikut

$$f(x) = \left(P + \frac{a}{x^2}\right)(x - b) - RT$$

Persamaan di atas umumnya digunakan untuk mencari nilai molal ideal dari zat oksigen maupun karbondioksida pada suhu T dalam satuan Kelvin dan tekanan P dalam satuan atmosfer (atm). Sementara untuk a dan b berisi konstanta dari mol oksigen dan karbondioksida itu sendiri. Ide dari permasalahan ini didapatkan dari buku *Numerical Methods for Engineers* oleh Steven C. Chapra dan Raymond P. Canale.

2. Tujuan Simulasi Numeris

2.1. Mencari solusi optimal dengan cara tercepat dan mengidentifikasi kemungkinan hasil tak konvergen dari sebuah persamaan tak linier

2.2. Mencari solusi akhir dari sistem persamaan tak linier dari berapapun titik terkaan awal yang dimasukan

2.3. Memberikan hasil dari sebuah masukan pada sistem persamaan tak linier dengan memperhatikan nilai eror yang ditetapkan oleh pengguna

2.4. Mengimplementasikan Metode Newton-Rhapson ke dalam formula Van Der Waals untuk mencari jumlah mol ideal

3. **Kekurangan Simulasi**

3.1. Kurangnya kemampuan menyajikan data yang empiris karena tidak ada proses penelitian yang runtut di awal

3.2. Ketidakmampuan penulis untuk menciptakan sebuah fungsi yang lebih akurat karena keterbatasan pengetahuan

3.3. Kurangnya kemampuan gawai penulis dalam pemrosesan data dengan jumlah yang sangat besar.

B. Metode

1. Metode Numeris yang Digunakan

Metode Newton-Rhapson merupakan salah satu metode numerik yang mencari akar persamaan dari sebuah sistem permsaaan tak linier dengan memanfaatkan turunan dari sistem persamaan itu sendiri. Formula dari Metode Newton-Rhapson adalah sebagai berikut

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Pada mulanya, nilai terkaan (x_n) dimasukkan ke dalam persamaan dan nilainya dikurangi dari hasil pembagian antara fungsi $f(x_n)$ dan turunan fungsi $f'(x_n)$. Kemudian hasilnya disimpan ke dalam x_{n+1}. Setelahnya, proses yang sama dilakukan secara terus menerus (iterasi) sampai nilai n yang terakhir.

2. Estimasi Awal

Ada beberapa ketentuan untuk menentukan terkaan awal (x_0) agar nilai yang dihasilkan oleh Metode Newton-Rhapson konvergen. Pertama, tes nilai terkaan awal dengan memasukkannya ke dalam fungsi derivatif. Hal ini dilakukan untuk menghindari nilai dari fungsi derivatif mendekati nol. Apabila mendekati nol, dikhawatirkan terjadi pembagian suatu bilangan dengan angka nol (akan menghasilkan eror). Kedua, usahakan agar nilai terkaan awal dekat dengan akar persamaan. Hal ini bertujuan untuk mengurangi jumlah iterasi yang dibutuhkan agar perhitungan dengan Metode Newton-Rhapson bisa lebih efisieen, terutama dari segi waktu.

3. Kriteria Pemberhentian Simulasi

Ada dua factor pemberhentian simulasi pada Metode Newton-Rhapson. Keduanya dicek secara simultan selama proses penghitungan berlangsung. Pertama, nilai eror. Nilai eror pada penghitungan menggunakan Metode Newton-Rhapson ditentukan oleh pengguna. Maka dari itu, ketelitian dari hasil akhir akan memengaruhi hasil akhir dari akar persamaan. Pada kasus Newton-Rhapson, apabila nilai mutlak dari f(x) kurang dari batas eror, maka iterasi akan berhenti. Kedua, jumlah iterasi maksimal. Sama halnya dengan batas nilai eror, jumlah iterasi maksimal juga ditentukan oleh pengguna. Apabila iterasi sudah menyamai jumlah iterasi maksimal, maka proses iterasi akan berhenti. Ketika proses iterasi berhenti, akan diketahui apakah akar persamaannya konvergen atau divergen.

4. Pseudocode

```
main_func(input)
    PROGRAM START
        ASSIGN ((1 + (3.592/(input^2))) * (input-0.04267)-(0.082054 * 300))
    PROGRAM END

deriv_func(input)
    PROGRAM START
        ASSIGN (2.5 - (14.09/input**2) + (2.801092/input**3))
    PROGRAM END

Newton_rhapson(start, maximal_iteration, error_score)
    PROGRAM START

        DECLARE result = 0, sentinel = True, current_iteration = 0

        WHILE LOOP sentinel
            IF deriv_func(start) EQUAL TO 0
                BREAK THE LOOP
            ENDIF

            result = start - (main_func(start)/deriv_func(start))
            start = result

            current_iteration = current_iteration + 1
```

```
            DISPLAY "Iteration no. ",current_iteration , " current result: ",result

            IF current_iteration > maximal_iteration
                DISPLAY "The convergence root cannot be found"
                BREAK THE LOOP
            ENDIF

           IF xn > 1e20 or xn < -(1e20)
                DISPLAY “The convergence root cannot be found”
                BREAK THE LOOP

            sentinel = WILL BE TRUE IF ABSOLUTE(main_func(start)) >= error_score
        END WHILE LOOP
        DISPLAY "Loop stop."
        DISPLAY "final result: %0.5f" %(xn), " iteration: ", cur_iteration"
    PROGRAM END
```

5. Code

```python
# f(x) and f'(x)
def main_func(suhu, tekanan, x): #f(x)
    return ((tekanan + (3.592/(x**2)))*(x-0.04267)-(0.082054 * suhu))

def deriv_func(tekanan, x): #df(x)
    return (tekanan - (3.592/x**2) + (2 * 3.592 * 0.04267/x**3))

#Newton-Rhapson
def newton_rhapson(suhu, tekanan, x0, max_iteration, error):
    xn = 0

    sentinel = True
    cur_iteration = 0

    while sentinel:
        if deriv_func(tekanan, x0) == 0: #Division by zero will cause
an error
            print("Error. f'(x) equal to zero")
            break

        xn = x0 - main_func(suhu, tekanan, x0)/deriv_func(tekanan, x0)
        x0 = xn

        cur_iteration += 1
        print("iteration no.", cur_iteration, " current result: ", xn)
```

```
        if cur_iteration > max_iteration:
            print("The convergence root cannot be found")
            break

        if xn > 1e20 or xn < -(1e20):
            print("The convergence root cannot be found")
            break

        sentinel = abs(main_func(suhu, tekanan, xn)) >= error

    print("Loop stop.")
    print('\nfinal result: %0.5f' %(xn), 'mol' '\niteration: ',
cur_iteration)

##### MAIN PROGRAM #####
suhu = float(input("Masukkan nilai suhu dalam satuan kelvin: "))
tekanan = float(input("Masukkan nilai tekanan dalam satuan atm: "))
x0 = float(input("Masukkan nilai terkaan awal: "))
max_iteration = int(input("Masukkan batas iterasi: "))
error = float(input("Masukkan batas nilai eror: "))

newton_rhapson(suhu, tekanan, x0, max_iteration, error)
```

Penjelasan Kode:

Kode dimulai dengan mendefinisikan main_func() dan deriv_func(). Main_func() menampung hasil dari perhitungan f(x), sementara deriv_func() menampung hasil perhitungan dari f(x). Setelahnya, sebuah fungsi lain dibuat dengan nama newton_rhapson(). Fungsi ini yang akan menjalankan operasi Newton-Rhapson.

Perhitungan Newton-Rhapson dimulai dengan mengambil masukan dari pengguna untuk nilai terkaan awal, batas iterasi, dan batas nilai eror. Selanjutnya, ketiga masukan ini akan digunakan untuk menghitung akar persamaan dari fungsi f(x). Pada awal loop, nilai deriv_func() akan dicek apakah menghasilkan nilai nol. Apabila menghasilkan nilai nol, maka loop akan berhenti dan fungsi tidak dijalankan. Apabila tidak menghasilkan nilai nol, proses berlanjut dengan menghitung nilai x_n dari x_0. Itetasi akan terus dilakukan sampai salah satu dari 2 faktor terpenuhi. Antara nilai iterasi melebihi batas iterasi atau nilai mutlak dari main_func() lebih besar sama dengan batas eror. Setelah loop berhenti, maka akan ditampilkan nilai akhir dari perhitungan Newton-

Rhapson beserta dengan jumlah loop yang dibutuhkan untuk mendapatkan akar tersebut.

Berikut adalah persamaan yang digunakan pada perhitungan di atas

$$f(x) = \left(P + \frac{a}{x^2}\right)(x - b) - RT$$

$$f'(x) = P - \frac{a}{x^2} + \frac{2ab}{x^3}$$

Dengan nilai a = 3.592, b = 0.04267, R = 0.0802, T = masukkan user, dan P = masukkan user

C. Analisis

1. Hasil Awal Simulasi

1.1. Percobaan 1

```
Masukkan nilai suhu dalam satuan kelvin: 300
Masukkan nilai tekanan dalam satuan atm: 1
Masukkan nilai terkaan awal: 0.02
Masukkan batas iterasi: 100
Masukkan batas nilai eror: 1e-10
iteration no. 1  current result:  0.02777866030691245
iteration no. 2  current result:  0.03751768179537956
iteration no. 3  current result:  0.04912547402749588
iteration no. 4  current result:  0.06278500416444706
iteration no. 5  current result:  0.08187090053744032
iteration no. 6  current result:  0.232438988399022
iteration no. 7  current result:  -0.05508223253232636
iteration no. 8  current result:  -0.101630595440746
iteration no. 9  current result:  -0.2189517542817273
iteration no. 10  current result:  -0.650250618383613
iteration no. 11  current result:  -4.273373794340446
iteration no. 12  current result:  32.98215527307602
iteration no. 13  current result:  24.522241086716534
iteration no. 14  current result:  24.512588151164586
iteration no. 15  current result:  24.512588128441504
Loop stop.

final result: 24.51259 mol
iteration:  15
```

1.2.Percobaan 2

```
Masukkan nilai suhu dalam satuan kelvin: 250
Masukkan nilai tekanan dalam satuan atm: 2
Masukkan nilai terkaan awal: 0.01
Masukkan batas iterasi: 100
Masukkan batas nilai eror: 1e-10
iteration no. 1  current result:  0.014412352255873923
iteration no. 2  current result:  0.020395813534197945
iteration no. 3  current result:  0.028138191794259464
iteration no. 4  current result:  0.037511907940482536
iteration no. 5  current result:  0.04785696586454402
iteration no. 6  current result:  0.057909153541102765
iteration no. 7  current result:  0.06607577343906093
iteration no. 8  current result:  0.07107987467232167
iteration no. 9  current result:  0.07286712749385824
iteration no. 10  current result:  0.07307513354090571
iteration no. 11  current result:  0.07307776586269778
iteration no. 12  current result:  0.07307776628012173
Loop stop.

final result: 0.07308 mol
iteration:  12
```

1.3. Percobaan 3

```
Masukkan nilai suhu dalam satuan kelvin: 500
Masukkan nilai tekanan dalam satuan atm: 3
Masukkan nilai terkaan awal: 0.004
Masukkan batas iterasi: 100
Masukkan batas nilai eror: 1e-10
iteration no. 1  current result:  0.005910658448279657
iteration no. 2  current result:  0.008675836895374621
iteration no. 3  current result:  0.012620352397197502
iteration no. 4  current result:  0.01815169155738639
iteration no. 5  current result:  0.025793544050536688
iteration no. 6  current result:  0.03639699993262183
iteration no. 7  current result:  0.05230631283240614
iteration no. 8  current result:  0.08637085884699316
iteration no. 9  current result:  -7.1412849203267115
iteration no. 10  current result:  14.398764236242656
iteration no. 11  current result:  13.631020237116896
iteration no. 12  current result:  13.630771174971287
Loop stop.

final result: 13.63077 mol
iteration:  12
```

2. Variasi Ukuran Matriks

Pada Metode Newton-Rhapson tidak ditemui penggunaan matriks. Sehingga, tidak terdapat variasi ukuran matriks yang digunakan dalam proses kalkulasi.

3. Analisis Hasil

Tiga pengujian pada subbab Hasil Awal Simulasi menunjukkan bahwa fungsi berhasil menemukan hasil yang tidak berbeda jauh untuk setiap masukan. Hal ini membuktikan bahwa kode yang dibuat sudah benar dan memenuhi prinsip Newton-Rhapson di mana iterasi bisa dihentikan ketika nilai mutlak dari main_func() atau f(x) lebih dari sama dengan batas nilai eror.

Hal selanjutnya yang harus diperhatikan adalah tidak ditemukan nilai yang divergen. Pada hasil simulasi di atas, seluruh hasil dari tiap masukan ada dan bernilai konvergen. Hal ini menunjukkan bahwa fungsi pada main_func() dan deriv_func() valid untuk dibawa ke dalam bentuk Newton-Rhapson dan dilakukan proses iterasi.

Kode di atas menunjukkan proses pencarian nilai volume mol ideal yang dibutuhkan pada sebuah sistem kimia karbondioksida. Nilai a dan b merupakan konstanta dari fungsi itu sendiri. Nilai R merupakan konstanta mol universal. Nilai T merupakan suhu pada satuan Kelvin dengan nilai x K. Nilai P merupakan tekanan yang dialami oleh sistem (umumnya menggunakan tekanan udara normal yaitu 1 atm). Dengan memasukkan nilai terkaan untuk volume mol ideal, didapatkan nilai y mol/L sebagai nilai mol ideal untuk sistem persamaan tersebut.

D. Kesimpulan

Metode Newton-Rhapson merupakan salah satu cara dalam metode numeris yang dapat digunakan untuk menemukan akar persamaan dari sebuah sistem persamaan tak linier. Metode ini menggunakan prinsip iterasi untuk setiap penghitungannya dan nilai eror yang juga menjadi parameter apakah perhitungan harus dilanjutkan atau berhenti karena akar persamaannya sudah ditemukan.

Untuk menggunakan metode Newton-Rhapson dibutuhkan sebuah fungsi yang akan dicari akar persamaannya. Umumnya, fungsi yang digunakan adalah persamaan tak linier. Kemudian, dari fungsi tersebut, dicari dungsi derivatifnya. Setelahnya, nilai dari terkaan awal dikurangi dengan hasil pembagian sebuah fungsi dan nilai derivatifnya. Proses ini dilakukan berulang terus-menerus sampai memenuhi salah satu dari dua kondisi: nilai mutlak fungsi pada saat n melebihi/sama dengan nilai eror atau jumlah iterasi melebihi batas iterasi maksimal.

SECANT METHOD

A. Pendahuluan

1. Pendahuluan

Pada studi keinsinyuran, ada banyak sekali permasalahan yang tidak bisa dipecahkan menggunakan metode tradisional. Bahkan, beberapa persamaan matematis memerlukan bantuan komputasi komputer untuk menemukan jawaban yang konvergen. Salah satu metode yang bisa digunakan adalah Secant. Secant merupakan salah satu metode perhitungan matematis untuk menyelesaikan persamaan tak linier secara numerik. Metode ini akan memberikan hasil penghitungan yang mendekati hasil asli atau bahkan sama dengan hasil asli. Perlu diperhatikan bahwa toleransi eror memegang peranan penting dalam hasil yang akan didapatkan.

Untuk menunjukkan pemanfaatan Newton-Rhapson secara riil, pada laporan ini Newton-Rhapson akan dimanfaatkan untuk menghitung volume molal ideal yang didasari pada persamaan Van Der Waals. Dalam sebuah proses kimia, jumlah molal dari sebuah zat bisa dihitung mrenggunakan pertsamaan sebagai berikut

$$f(x) = \left(P + \frac{a}{x^2}\right)(x - b) - RT$$

Persamaan di atas umumnya digunakan untuk mencari nilai mol ideal dari zat oksigen maupun karbondioksida pada suhu T dalam satuan Kelvin dan tekanan P dalam satuan atmosfer (atm). Sementara untuk a dan b berisi konstanta dari mol oksigen dan karbondioksida itu sendiri. Ide dari permasalahan ini didapatkan dari buku

2. Tujuan Simulasi Numeris

2.1. Mencari solusi optimal dengan cara tercepat dan mengidentifikasi kemungkinan hasil tak konvergen dari sebuah persamaan tak linier

2.2. Mencari solusi akhir dari sistem persamaan tak linier dari berapapun titik terkaan awal yang dimasukan

2.3. Memberikan hasil dari sebuah masukan pada sistem persamaan tak linier dengan memperhatikan nilai eror yang ditetapkan oleh pengguna

2.4. Mengimplementasikan Metode Secant ke dalam formula Van Der Waals untuk mencari jumlah mol ideal

3. **Kekurangan Simulasi**

3.1. Kurangnya kemampuan menyajikan data yang empiris karena tidak ada proses penelitian yang runtut di awal

3.2. Ketidakmampuan penulis untuk menciptakan sebuah fungsi yang lebih akurat karena keterbatasan pengetahuan

3.3. Kurangnya kemampuan gawai penulis dalam pemrosesan data dengan jumlah yang sangat besar.

B. Metode

1. Metode Numeris yang Digunakan

Metode Secant merupakan salah satu metode numerik yang mencari akar persamaan dari sebuah sistem permsaaan tak linier dengan memanfaatkan turunan dari sistem persamaan itu sendiri. Formula dari Metode Secant adalah sebagai berikut

$$x_{n+1} = x_n - \frac{f(x_n)(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}$$

Pada mulanya, nilai terkaan awal (x_{n-1}) dan nilai terkaan kedua (x_n) dimasukkan ke dalam persamaan. Nilai terkaan kedua (x_n) dikurangi dengan hasil pembagian antara perkalian $f(x_n)$ dengan ($x_{n-1} - x_n$) dan $f(x_{n-1}) - f(x_n)$. Proses ini terus diiterasi sampai jumlah iterasi memenuhi batas maksimal iterasi atau nilai mutlak dari $f(x_{n+1})$ lebih besar sama dengan batas eror.

Untuk menunjukkan pemanfaatan Secant secara riil, pada laporan ini Secant akan dimanfaatkan untuk menghitung volume molal ideal yang didasari pada persamaan Van Der Waals. Ide dari permasalahan ini didapatkan dari buku *Numerical Methods for Engineers* oleh Steven C. Chapra dan Raymond P. Canale.

2. Estimasi Awal

Ada beberapa ketentuan untuk menentukan terkaan awal (x_{n-1}) agar nilai yang dihasilkan oleh Metode Secant konvergen. Pertama, tes nilai terkaan awal dan kedua dengan memasukkannya ke dalam $f(x_{n-1}) - f(x_n)$. Hal ini dilakukan untuk menghindari nilai dari fungsi $f(x_{n-1}) - f(x_n)$ mendekati nol. Apabila mendekati nol, dikhawatirkan terjadi pembagian suatu bilangan dengan angka nol (akan menghasilkan eror). Kedua, usahakan agar nilai terkaan awal dan kedua dekat dengan akar persamaan. Hal ini

bertujuan untuk mengurangi jumlah iterasi yang dibutuhkan agar perhitungan dengan Metode Secant bisa lebih efisien, terutama dari segi waktu.

3. Kriteria Pemberhentian Simulasi

Ada dua factor pemberhentian simulasi pada Metode Secant. Keduanya dicek secara simultan selama proses penghitungan berlangsung. Pertama, nilai eror. Nilai eror pada penghitungan menggunakan Metode Secant ditentukan oleh pengguna. Maka dari itu, ketelitian dari hasil akhir akan memengaruhi hasil akhir dari akar persamaan. Pada kasus Secant, apabila nilai mutlak dari $f(x_{n+1})$ kurang dari batas eror, maka iterasi akan berhenti. Kedua, jumlah iterasi maksimal. Sama halnya dengan batas nilai eror, jumlah iterasi maksimal juga ditentukan oleh pengguna. Apabila iterasi sudah menyamai jumlah iterasi maksimal, maka proses iterasi akan berhenti. Ketika proses iterasi berhenti, akan diketahui apakah akar persamaannya konvergen atau divergen.

4. Pseudocode

```
main_func(tekanan, suhu, input)
    PROGRAM START
        ASSIGN ((tekanan + (3.592/(input^2))) * (input-0.04267)-(0.082054 * suhu))
    PROGRAM END
secant_method(tekanan, suhu, x0,x1, iteration, error)
PROGRAM START
        DECLARE result = 0, sentinel = True, current_iteration = 0

        WHILE LOOP sentinel
            IF main_func(tekanan, suhu, start) - main_func(tekanan, suhu, end) EQUAL
TO 0
                BREAK THE LOOP
            ENDIF

            result = end - (main_func(tekanan, suhu, end) * (start - end) /
(main_func(tekanan, suhu, start) - main_func(tekanan, suhu, end)))
            start = end
            end = result

            current_iteration = current_iteration + 1

            DISPLAY "Iteration no. ", current_iteration, " current result: ", result

            IF current_iteration > maximal_iteration
```

```
                DISPLAY "The convergence root cannot be found"
                BREAK THE LOOP
            ENDIF
`
            IF xn > 1e20 or xn < -(1e20)
                DISPLAY “The convergence root cannot be found”
                BREAK THE LOOP

            sentinel = WILL BE TRUE IF ABSOLUTE(main_func(start)) >= error_score
        END WHILE LOOP

        DISPLAY "Loop stop."
        DISPLAY "final result: %0.5f" %(xn), " iteration: ", cur_iteration"
PROGRAM END
```

5. Code

```python
def main_func(suhu, tekanan, x): #f(x)
    return ((tekanan + (3.592/(x**2)))*(x-0.04267)-(0.082054 * suhu))

def secant_method(suhu, tekanan, x0, x1, iteration, error):

    xn = 0

    sentinel = True
    cur_iteration = 0

    while sentinel:
        if (main_func(suhu, tekanan, x0) - main_func(suhu, tekanan,
x1)) == 0: #Division by zero will cause an error
            break

        xn = x1 - ((main_func(suhu, tekanan, x1) * (x0-x1) /
(main_func(suhu, tekanan, x0)-main_func(suhu, tekanan, x1))))
        x0 = x1
        x1 = xn

        cur_iteration += 1
        print("iteration no.", cur_iteration, " current result: ", xn)

        if cur_iteration > iteration:
            print("The convergence root cannot be found")
            break
```

```
        if xn > 1e20 or xn < -(1e20):
            print("The convergence root cannot be found")
            break

        sentinel = abs(main_func(suhu, tekanan, xn)) > error

    print("Loop stop.")
    print("\nfinal result: %0.5f" %(xn), 'mol', "\niteration: ",
cur_iteration)
```

Penjelasan Kode:

Kode dimulai dengan deklarasi fungsi main_func() yang menampung sistem persamaan tak linier yang akan digunakan untuk penghitungan dengan metode Secant. Setelahnya, sebuah fungsi lain dibuat dengan nama secant_method(). Fungsi ini akan menjalankan operasi Secant.

Perhitungan Secant dimulai dengan mengambil masukan dari pengguna untuk nilai terkaan awal, nilai terkaan kedua, batas maksimal iterasi, dan batas nilai eror. Selanjutnya, keempat masukan akan digunakan untuk menghitung persamaan dari fungsi f(x). Pada awal loop, nilai $f(x_{n-1}) - f(x_n)$ akan dicek apakah menghasilkan nilai nol. Apabila menghasilkan nilai nol, maka niai loop akan berhenti dan fungsi tidak dijalankan. Apabila tidak menghasilkan nilai nol, proses berlanjut dengan menghitung nilai x_{n+1} dari x_n dan x_{n-1}. Iterasi akan terus dilakukan sampai salah satu dari dua faktor terpenuhi. Antara nilai iterasi melebihi batas iterasi atau nilai mutlak dri main_func() lebih besar sama dengan batas eror. Setelah loop berhenti, maka akan ditampilkan nilai akhir dari perhitungan Secant beserta dengan jumlah loop yang dibutuhkan untuk mendapatkan akar persamaan tersebut.

Berikut adalah persamaan yang digunakan pada perhitungan di atas

$$f(x) = \left(P + \frac{a}{x^2}\right)(x - b) - RT$$

Dengan nilai a = 3.592, b = 0.04267, R = 0.0802, T = masukkan user, dan P = masukkan user.

C. Analisis

1. Hasil Awal Simulasi

1.1.Percobaan 1

```
Masukkan nilai tekanan dalam satuan atm: 1
Masukkan nilai suhu dalam satuan kelvin: 300
Masukkan nilai terkaan awal: 0.01
Masukkan nilai terkaan akhir: 27
Masukkan batas iterasi: 100
Masukkan batas nilai eror: 1e-10
iteration no. 1  current result:  26.944385750195085
iteration no. 2  current result:  24.513806550367118
iteration no. 3  current result:  24.512588786021045
iteration no. 4  current result:  24.5125881284417
Loop stop.

final result: 24.51259 mol
iteration:  4
```

1.2.Percobaan 2

```
Masukkan nilai tekanan dalam satuan atm: 2
Masukkan nilai suhu dalam satuan kelvin: 700
Masukkan nilai terkaan awal: 3
Masukkan nilai terkaan akhir: 30
Masukkan batas iterasi: 100
Masukkan batas nilai eror: 1e-10
iteration no. 1  current result:  28.67577631072211
iteration no. 2  current result:  28.699080447002036
iteration no. 3  current result:  28.69908264748232
Loop stop.

final result: 28.69908 mol
iteration:  3
```

1.3. Percobaan 3

```
Masukkan nilai tekanan dalam satuan atm: 2.5
Masukkan nilai suhu dalam satuan kelvin: 210
Masukkan nilai terkaan awal: 0.01
Masukkan nilai terkaan akhir: 40
Masukkan batas iterasi: 100
Masukkan batas nilai eror: 1e-10
iteration no. 1  current result:  37.40160557945019
iteration no. 2  current result:  6.867574627043922
iteration no. 3  current result:  6.726507562052854
iteration no. 4  current result:  6.722845867675953
iteration no. 5  current result:  6.7228433843113296
iteration no. 6  current result:  6.722843384267776
Loop stop.

final result: 6.72284 mol
iteration:  6
```

2. **Variasi Ukuran Matriks**

Pada Metode Secant tidak ditemui penggunaan matriks. Sehingga, tidak terdapat variasi ukuran matriks yang digunakan dalam proses kalkulasi.

3. **Analisis Hasil**

Tiga pengujian pada subbab Hasil Awal Simulasi menunjukkan bahwa fungsi berhasil menemukan hasil yang tidak berbeda jauh untuk setiap masukan. Jal ini membuktikan bahwa kode yang dibuat sudah benar dan memenuhi prinsip Secant di mana iterasi bisa dihentikan ketika nilai mutlak dari main_func() atau f(x) lebih dari sama dengan batas nilai eror.

Hal selanjutnya yang harus diperhatikan adalah tidak ditemukan nilai yang divergen. Pada hasil simulasi di atas, seluruh hasil dari tiap masukan ada dan bernilai konvergen. Hal ini menunjukkan bahwa fungsi pada main_func() valid untuk dibawa ke dalam bentuk Secant dan dilakukan proses iterasi.

Kode di atas menunjukkan proses pencarian nilai volume mol ideal yang dibutuhkan pada sebuah sistem kimia karbondioksida. Nilai a dan b merupakan konstanta dari fungsi itu sendiri. Nilai R merupakan konstanta mol universal. Nilai T merupakan suhu pada satuan Kelvin dengan nilai x K. Nilai P merupakan tekanan yang dialami oleh sistem (umumnya menggunakan tekanan udara normal yaitu 1 atm). Dengan memasukkan nilai terkaan untuk volume mol ideal, didapatkan nilai y mol/L sebagai nilai mol ideal untuk sistem persamaan tersebut.

D. Kesimpulan

Metode Secant merupakan salah satu cara dalam metode numeris yang dapat digunakan untuk menemukan akar persamaan dari sebuah sistem persamaan tak linier. Metode ini menggunakan prinsip iterasi untuk setiap penghitungannya dan nilai eror yang juga menjadi parameter apakah perhitungan harus dilanjutkan atau berhenti karena akar persamaannya sudah ditemukan.

Untuk menggunakan metode Secant dibutuhkan sebuah fungsi yang akan dicari akar persamaannya. Umumnya, fungsi yang digunakan adalah persamaan tak linier. Kemudian, dari fungsi tersebut akan dicari nilai dari terkaan awal dan kedua. Setelahnya, nilai terkaan kedua akan dikurangi dengan hasil pembagian dari perkalian antara fungsi yang dimasukkan terkaan kedua dan hasil pengurangan antara terkaan awal serta akhir dan hasil pengurangan antara fungsi yang dimasukkan terkaan awal dan dan akhir. Proses ini dilakukan berulang terus-menerus sampai memenuhi salah satu dari dua kondisi: nilai mutlak fungsi pada saat n melebihi/sama dengan nilai eror atau jumlah iterasi melebihi batas iterasi maksimal.

GAUSS SEIDEL METHOD

A. Pendahuluan

1. Permasalahan yang Diangkat

Pada studi keinsinyuran, ada banyak sekali permasalahan yang tidak bisa dipecahkan menggunakan metode tradisional. Bahkan, beberapa persamaan matematis memerlukan bantuan komputasi komputer untuk menemukan jawaban yang konvergen. Salah satu metode yang bisa digunakan adalah Gauss-Seidel. Gauss-Seidel merupakan salah satu metode perhitungan matematis untuk menyelesaikan persamaan linier secara numerik. Metode ini akan memberikan hasil penghitungan yang mendekati hasil asli atau bahkan sama dengan hasil asli. Perlu diperhatikan bahwa toleransi eror memegang peranan penting dalam hasil yang akan didapatkan.

Untuk menunjukkan pemanfaatan Gauss-Seidel secara riil, pada laporan ini Gauss-Seidel akan dimanfaatkan untuk menghitung jumlah debit air pada setiap selang yang mengisi ember dengan air dalam satu waktu yang sama. Ide dari permasalahan ini didapatkan dari buku *Numerical Methods for Engineers* oleh Steven C. Chapra dan Raymond P. Canale.

2. Tujuan Simulasi Numeris

2.1. Mencari solusi optimal dengan cara tercepat dan mengidentifikasi kemungkinan hasil tak konvergen dari sebuah persamaan linier

2.2. Mencari solusi akhir dari sistem persamaan linier dari berapapun titik terkaan awal yang dimasukan

2.3. Memberikan hasil dari sebuah masukan pada sistem persamaan linier dengan memperhatikan nilai eror yang ditetapkan oleh pengguna

2.4. Mengimplementasikan Metode Gauss-Seidel ke dalam studi kasus pada kehidupan sehari-hari

3. Kekurangan Simulasi

3.1. Kurangnya kemampuan menyajikan data yang empiris karena tidak ada proses penelitian yang runtut di awal

3.2. Ketidakmampuan penulis untuk menciptakan sebuah fungsi yang lebih akurat karena keterbatasan pengetahuan

3.3. Kurangnya kemampuan gawai penulis dalam pemrosesan data dengan jumlah yang sangat besar.

B. Metode

1. Metode Numeris yang Digunakan

Metode Gauss-Seidel merupakan salah satu metode numerik yang mencari akar persamaan dari sebuah sistem permsaaan linier dengan memanfaatkan iterasi berulang yang dikombinasikan dengan pengurangan setiap komponen matriks dengan komponen matriks lainnya pada sebuah *square matrix*. Metode Gauss-Seidel tidak jauh berbeda dengan Jacobi. Hanya saja, untuk setiap nilai x_1, x_2, …, x_n, pada proses iterasi penghitungan menggunakan nilai x terbaru untuk masing-masing indeks.

2. Estimasi-Awal

Berbeda dengan metode Newton-Rhapson dan Secant, tidak ada estimasi awal pada proses iterasi Gauss-Seidel. Pengguna cukup memasukkan nilai dari matriks yang diambil dari persamaan linier yang umumnya berjumlah lebih dari satu persamaan.

3. Kriteria Pemberhentian Simulasi

Ada dua faktor pemberhentian simulasi pada Metode Gauss-Seidel. Keduanya dicek secara simultan selama proses penghitungan berlangsung. Pertama, nilai eror. Nilai eror pada penghitungan menggunakan Metode Gauss-Seidel ditentukan oleh pengguna. Maka dari itu, ketelitian dari hasil akhir akan memengaruhi hasil akhir dari akar persamaan. Pada kasus Gauss-Seidel, apabila nilai mutlak dari matriks hasil pada saat n dikurangi dengan matriks hasil saat n-1 (result[n] – result[n-1] kurang dari batas eror, maka iterasi akan berhenti. Kedua, jumlah iterasi maksimal. Sama halnya dengan batas nilai eror, jumlah iterasi maksimal juga ditentukan oleh pengguna. Apabila iterasi sudah menyamai jumlah iterasi maksimal, maka proses iterasi akan berhenti. Ketika proses iterasi berhenti, akan diketahui apakah akar persamaannya konvergen atau divergen.

4. Pseudocode

```
gauss_seidel(matrix_a,matrix_b,result,error)
    PROGRAM START
        DECLARE matrix_length = LENGTH OF matrix_a
        DECLARE count1 = 0 AND count2 = 0

        WHILE LOOP count1 < matrix_length
            DECLARE temp = matrix_b AT count1

            WHILE LOOP
                IF (count1 =/= count1)
                    temp = temp - matrix_a AT count1,count2 * result AT count2

                count2 + 1

            count2 = 0

            DECLARE vessel = result at count1

            result AT count1 = temp / matrix_a AT count1,count1

            IF (ABSOLUTE(result AT count1 - vessel) < error)
                exit the function

            count1 + 1

        DISPLAY result

    PROGRAM END
```

5. Code

```
def gauss_seidel(matrix_a, matrix_b, result, matrix_length, error):
    count1 = count2 = 0

    for count1 in range(matrix_length):

        temp = matrix_b[count1]

        for count2 in range(matrix_length):

            if (count2 != count1):
                temp -= (matrix_a[count1][count2] * result[count2])

            count2 += 1

        count2 = 0

        vessel = result[count1]

        result[count1] = temp / matrix_a[count1][count1]

        if result[count1] > 1e20 or result[count1] < -(1e20):
            print("Hasil Penghitungan Divergen. Loop Dihentikan.")
            exit()

        if (abs(result[count1] - vessel) < error):
            print("Hasil Penghitungan Konvergen. Loop Dihentikan.")
            print("Hasil Akhir: ", result)
            exit()

        count1 += 1

    return result

##### MAIN PROGRAM #####
n = int(input("Masukkan ukuran matrix nxn: "))

a = [[0 for i in range(n)] for j in range(n)]
b = [0 for i in range(n)]
c = [0 for i in range(n)]

error = 1e-10
max_iteration = 100
cur_iteration = 0
```

```
for i in range(n):
    for j in range(n):
        print("Masukkan angka untuk matriks indeks [",i,"][",j,"]")
        a[i][j] = int(input())

for i in range(n):
    print("Masukkan angka untuk matriks hasil indeks [",i,"]")
    b[i] = int(input())

while cur_iteration < max_iteration:
    final_result = gauss_seidel(a, b, c, n, error)
    print("Hasil iterasi ke-",cur_iteration + 1,": ",final_result)
    cur_iteration += 1
```

Penjelasan Kode:

Kode dimulai dengan deklarasi fungsi Gauss-Seidel yang menerima parameter berupa matrix A, matrix B, matriks hasil akhir, dan batas eror. Setelah menerima ketiga parameter tersebut, proses iterasi penghitungan dimulai. Pertama, ukuran dari matriks dicari untuk menjadi batasan iterasi. Setelahnya, sebuah loop dibuat dengan batasan ukuran matriks a. Kemudian, nilai dari matriks b pada indeks count1 disimpan ke dalam sebuah variabel bernama temp. Setelahnya, dibuat sebuah loop lagi yang akan menghitung nilai baru untuk matriks hasil akhir dengan memanfaatkan matriks b dan matriks a. Pada setiap penghitungan, dilakukan pengecekan apakah nilai mutlak dari matriks hasil akhir pada indeks count1 yang dikurangi dengan matriks hasil pada indeks count1-1 |(hasil_akhir[count1] – hasil_akhir[count-1])| kurang dari batas eror. Apabila memenuhi, maka program akan berhenti berjalan. Namun, apabila tidak terpenuhi, maka program akan terus berjalan. Demikian seterusnya hingga ditemukan apakah akar persamaan dari persamaan linier tersebut konvergen atau divergen.

C. Analisis

1. Hasil Awal Simulasi

1.1. Percobaan 1

```
Masukkan ukuran matrix nxn: 3
Masukkan angka untuk matriks indeks [ 0 ][ 0 ]
4
Masukkan angka untuk matriks indeks [ 0 ][ 1 ]
1
Masukkan angka untuk matriks indeks [ 0 ][ 2 ]
2
Masukkan angka untuk matriks indeks [ 1 ][ 0 ]
3
Masukkan angka untuk matriks indeks [ 1 ][ 1 ]
5
Masukkan angka untuk matriks indeks [ 1 ][ 2 ]
1
Masukkan angka untuk matriks indeks [ 2 ][ 0 ]
1
Masukkan angka untuk matriks indeks [ 2 ][ 1 ]
1
Masukkan angka untuk matriks indeks [ 2 ][ 2 ]
3
Masukkan angka untuk matriks hasil indeks [ 0 ]
4
Masukkan angka untuk matriks hasil indeks [ 1 ]
7
Masukkan angka untuk matriks hasil indeks [ 2 ]
3
Hasil iterasi ke- 1 :  [1.0, 0.8, 0.39999999999999997]
Hasil iterasi ke- 2 :  [0.6000000000000001, 0.9599999999999997, 0.48000000000000004]
Hasil iterasi ke- 3 :  [0.52, 0.9919999999999998, 0.49600000000000005]
Hasil iterasi ke- 4 :  [0.504, 0.9983999999999998, 0.4992000000000001]
Hasil iterasi ke- 5 :  [0.5008, 0.99968, 0.49984]
Hasil iterasi ke- 6 :  [0.5001599999999999, 0.9999360000000002, 0.4999679999999999]
Hasil iterasi ke- 7 :  [0.500032, 0.9999872, 0.4999936]
Hasil iterasi ke- 8 :  [0.5000064, 0.9999974400000001, 0.49999871999999995]
Hasil iterasi ke- 9 :  [0.50000128, 0.999999488, 0.4999997439999999]
Hasil iterasi ke- 10 :  [0.500000256, 0.9999998976000001, 0.49999994880000004]
Hasil iterasi ke- 11 :  [0.5000000512, 0.9999999795199999, 0.4999999897600001]
Hasil iterasi ke- 12 :  [0.50000001024, 0.999999995904, 0.499999997952]
Hasil iterasi ke- 13 :  [0.500000002048, 0.9999999991808, 0.49999999959040003]
Hasil iterasi ke- 14 :  [0.5000000004095999, 0.9999999998361601, 0.49999999991808003]
Hasil Penghitungan Konvergen. Loop Dihentikan.
Hasil Akhir:  [0.50000000008192, 0.9999999999672321, 0.49999999998361594]
```

1.2. Percobaan 2

```
Masukkan ukuran matrix nxn: 3
Masukkan angka untuk matriks indeks [ 0 ][ 0 ]
2
Masukkan angka untuk matriks indeks [ 0 ][ 1 ]
-2
Masukkan angka untuk matriks indeks [ 0 ][ 2 ]
-2
Masukkan angka untuk matriks indeks [ 1 ][ 0 ]
1
Masukkan angka untuk matriks indeks [ 1 ][ 1 ]
-6
Masukkan angka untuk matriks indeks [ 1 ][ 2 ]
-3
Masukkan angka untuk matriks indeks [ 2 ][ 0 ]
3
Masukkan angka untuk matriks indeks [ 2 ][ 1 ]
2
Masukkan angka untuk matriks indeks [ 2 ][ 2 ]
1
Masukkan angka untuk matriks hasil indeks [ 0 ]
-9
Masukkan angka untuk matriks hasil indeks [ 1 ]
-28
Masukkan angka untuk matriks hasil indeks [ 2 ]
-16
```

```
Hasil iterasi ke- 1 :  [-4.5, 3.9166666666666665, -10.333333333333332]
Hasil iterasi ke- 2 :  [-10.916666666666666, 8.013888888888888, 0.722222222222225]
Hasil iterasi ke- 3 :  [4.2361111111111125, 5.0115740740740735, -38.73148148148148]
Hasil iterasi ke- 4 :  [-38.219907407407405, 17.662422839506174, 63.334876543209866]
Hasil iterasi ke- 5 :  [76.49729938271604, -14.251221707818926, -216.98945473251024]
Hasil iterasi ke- 6 :  [-235.7406764032917, 73.87128129286693, 543.4794667352537]
Hasil iterasi ke- 7 :  [612.8507480281206, -164.93127536294008, -1524.6896933584817]
Hasil iterasi ke- 8 :  [-1694.1209687214218, 484.65801855900395, 4097.0468690462585]
Hasil iterasi ke- 9 :  [4577.204887605262, -1280.989286588919, -11185.63608963795]
Hasil iterasi ke- 10 :  [-12471.12537622687, 3518.9638154478307, 30359.44849778495]
Hasil iterasi ke- 11 :  [33873.91231323278, -9529.405530020345, -82578.92587965765]
Hasil iterasi ke- 12 :  [-92112.831409678, 25941.991038215823, 224438.51215260234]
Hasil iterasi ke- 13 :  [250376.00319081816, -70485.25554449816, -610173.4984834582]
Hasil iterasi ke- 14 :  [-680663.2540279564, 191647.5402370697, 1658678.6816097298]
Hasil iterasi ke- 15 :  [1850321.7218467994, -520947.72049706505, -4509085.724546268]
Hasil iterasi ke- 16 :  [-5030037.945043333, 1416207.8714325782, 12257682.092264842]
Hasil iterasi ke- 17 :  [13673885.46369742, -3849855.4688495174, -33321961.45339323]
Hasil iterasi ke- 18 :  [-37171821.422242746, 10465681.822989492, 90584084.62074925]
Hasil iterasi ke- 19 :  [101049761.94373874, -28450410.653084833, -246248480.52504656]
Hasil iterasi ke- 20 :  [-274698895.6781314, 77341095.64950137, 669414479.7353915]
Hasil iterasi ke- 21 :  [746755570.8848928, -210247973.38688025, -1819770781.8809178]
Hasil iterasi ke- 22 :  [-2030018759.767798, 571548935.645826, 4946958392.011743]
Hasil iterasi ke- 23 :  [5518507323.157569, -1553727970.8129432, -13448066043.846819]
Hasil iterasi ke- 24 :  [-15001794019.159761, 4223734023.3967834, 36557913994.68572]
Hasil iterasi ke- 25 :  [40781648013.582504, -11482015657.07911, -99380912742.5893]
Hasil iterasi ke- 26 :  [-110862928404.1684, 31213301641.933247, 270162181912.6387]
Hasil iterasi ke- 27 :  [301375483550.07196, -84851843693.30736, -734422763279.6011]
Hasil iterasi ke- 28 :  [-819274606977.4084, 230665613814.89914, 1996492593286.4272]
Hasil iterasi ke- 29 :  [2227158207096.826, -627053262122.4092, -5427368097061.66]
Hasil iterasi ke- 30 :  [-6054421359188.569, 1704613822004.0684, 14754036433541.57]
Hasil iterasi ke- 31 :  [16458650255541.139, -4633909840842.595, -40108131084954.23]
Hasil iterasi ke- 32 :  [-44742040925801.32, 12597058721514.896, 109032005334358.17]
Hasil iterasi ke- 33 :  [121629064055868.56, -34244491991196.324, -296398208185229.06]
Hasil iterasi ke- 34 :  [-330642700176429.9, 93091987396547.56, 805744125736178.5]
Hasil iterasi ke- 35 :  [898836113132721.5, -253066044012631.0, -2190376251372918.5]
Hasil iterasi ke- 36 :  [-2443442295385554.0, 687947743122205.0, 5954431399912236.0]
Hasil iterasi ke- 37 :  [6642379143034436.0, -1870152509450374.0, -1.6186832410202576e+16]
Hasil iterasi ke- 38 :  [-1.8056984919652956e+16, 5083918718492467.0, 4.400311732197391e+16]
Hasil iterasi ke- 39 :  [4.908703604046637 6e+16, -1.3820385987575888e+16, -1.196203361462474e+17]
Hasil iterasi ke- 40 :  [-1.3344072213382328e+17, 3.757004771748649e+16, 3.2518207096649683e+17]
Hasil iterasi ke- 41 :  [3.627521186839833e+17, -1.0213234903591787e+17, -8.83991657980114e+17]
Hasil iterasi ke- 42 :  [-9.861240070160319e+17, 2.7764182782071843e+17, 2.40308836540665 9e+18]
Hasil iterasi ke- 43 :  [2.6807301932273777e+18, -7.547558171654333e+17, -6.532678945351266e+18]
Hasil iterasi ke- 44 :  [-7.287434762516699e+18, 2.0517670122561833e+18, 1.7758770263037729e+19]
Hasil iterasi ke- 45 :  [1.9810537275293913e+19, -5.577628918969879e+18, -4.827635398794199e+19]
Hasil Penghitungan Divergen. Loop Dihentikan.
```

1.3. Percobaan 3

```
Masukkan ukuran matrix nxn: 3
Masukkan angka untuk matriks indeks [ 0 ][ 0 ]
3
Masukkan angka untuk matriks indeks [ 0 ][ 1 ]
1
Masukkan angka untuk matriks indeks [ 0 ][ 2 ]
-1
Masukkan angka untuk matriks indeks [ 1 ][ 0 ]
4
Masukkan angka untuk matriks indeks [ 1 ][ 1 ]
7
Masukkan angka untuk matriks indeks [ 1 ][ 2 ]
-3
Masukkan angka untuk matriks indeks [ 2 ][ 0 ]
2
Masukkan angka untuk matriks indeks [ 2 ][ 1 ]
-2
Masukkan angka untuk matriks indeks [ 2 ][ 2 ]
5
Masukkan angka untuk matriks hasil indeks [ 0 ]
5
Masukkan angka untuk matriks hasil indeks [ 1 ]
20
Masukkan angka untuk matriks hasil indeks [ 2 ]
10
Hasil iterasi ke- 1 :  [1.6666666666666667, 1.9047619047619047, 2.095238095238095]
Hasil iterasi ke- 2 :  [1.73015873015873, 2.766439909297052, 2.414512471655329]
Hasil iterasi ke- 3 :  [1.5493575207860921, 3.0065867616888022, 2.582891696361084]
Hasil iterasi ke- 4 :  [1.5254349782240937, 3.09241931088384, 2.6267937330638986]
Hasil iterasi ke- 5 :  [1.5114581407266863, 3.1192212337549927, 2.6431052372113224]
Hasil iterasi ke- 6 :  [1.507961334485443, 3.128210053384599, 2.6480994875596626]
Hasil iterasi ke- 7 :  [1.5066298113916876, 3.13111131673032, 2.649792602135453]
Hasil iterasi ke- 8 :  [1.5062270951350445, 3.132067060838026, 2.650335986281193]
Hasil iterasi ke- 9 :  [1.5060896418143888, 3.1323784845122895, 2.6505155370791607]
Hasil iterasi ke- 10 :  [1.5060456841889571, 3.132480553497379, 2.6505739477233687]
Hasil iterasi ke- 11 :  [1.506031131408663, 3.132513902505065, 2.650593108438561]
Hasil iterasi ke- 12 :  [1.506026401977832, 3.132524816772051, 2.650599365917688]
Hasil iterasi ke- 13 :  [1.5060248497152122, 3.132528385556031, 2.6506014143363275]
Hasil iterasi ke- 14 :  [1.5060243429267655, 3.1325295530431316, 2.6506020840465467]
Hasil iterasi ke- 15 :  [1.5060241770011384, 3.132529934876441, 2.650602303150121]
Hasil iterasi ke- 16 :  [1.5060241227578934, 3.132530059774113, 2.6506023748064877]
Hasil iterasi ke- 17 :  [1.5060241050107919, 3.132530100625185, 2.650602398245757]
Hasil iterasi ke- 18 :  [1.5060240992068572, 3.1325301139871202, 2.6506024059121054]
Hasil iterasi ke- 19 :  [1.5060240973083285, 3.132530118357572, 2.6506024084196973]
Hasil iterasi ke- 20 :  [1.5060240966873752, 3.1325301197870843, 2.6506024092398834]
Hasil iterasi ke- 21 :  [1.5060240964842666, 3.1325301202546547, 2.650602409508155]
Hasil Penghitungan Konvergen. Loop Dihentikan.
Hasil Akhir:  [1.5060240964178335, 3.1325301202546547, 2.650602409508155]
```

2. Variasi Ukuran Matriks

2.1. Matriks Ukuran 2x2

```
Masukkan ukuran matrix nxn: 2
Masukkan angka untuk matriks indeks [ 0 ][ 0 ]
3
Masukkan angka untuk matriks indeks [ 0 ][ 1 ]
1
Masukkan angka untuk matriks indeks [ 1 ][ 0 ]
4
Masukkan angka untuk matriks indeks [ 1 ][ 1 ]
7
Masukkan angka untuk matriks hasil indeks [ 0 ]
5
Masukkan angka untuk matriks hasil indeks [ 1 ]
14
Hasil iterasi ke- 1 :  [1.6666666666666667, 1.0476190476190477]
Hasil iterasi ke- 2 :  [1.3174603174603174, 1.2471655328798188]
Hasil iterasi ke- 3 :  [1.2509448223733937, 1.2851743872152035]
Hasil iterasi ke- 4 :  [1.2382752042615988, 1.2924141689933724]
Hasil iterasi ke- 5 :  [1.235861943668876, 1.2937931750463565]
Hasil iterasi ke- 6 :  [1.235402274984548, 1.2940558428659727]
Hasil iterasi ke- 7 :  [1.2353147190446758, 1.2941058748316137]
Hasil iterasi ke- 8 :  [1.2352980417227954, 1.2941154047298313]
Hasil iterasi ke- 9 :  [1.2352948650900561, 1.2941172199485393]
Hasil iterasi ke- 10 :  [1.2352942600171535, 1.2941175657044839]
Hasil iterasi ke- 11 :  [1.235294144765172, 1.294117631562759]
Hasil iterasi ke- 12 :  [1.2352941228124137, 1.2941176441071922]
Hasil iterasi ke- 13 :  [1.2352941186309359, 1.294117646496608]
Hasil iterasi ke- 14 :  [1.2352941178344639, 1.294117646951735]
Hasil Penghitungan Konvergen. Loop Dihentikan.
Hasil Akhir:  [1.235294117682755, 1.294117647038426]
```

2.2. Matriks Ukuran 4x4

```
Masukkan ukuran matrix nxn: 4
Masukkan angka untuk matriks indeks [ 0 ][ 0 ]
3
Masukkan angka untuk matriks indeks [ 0 ][ 1 ]
1
Masukkan angka untuk matriks indeks [ 0 ][ 2 ]
-1
Masukkan angka untuk matriks indeks [ 0 ][ 3 ]
2
Masukkan angka untuk matriks indeks [ 1 ][ 0 ]
4
Masukkan angka untuk matriks indeks [ 1 ][ 1 ]
7
Masukkan angka untuk matriks indeks [ 1 ][ 2 ]
-3
Masukkan angka untuk matriks indeks [ 1 ][ 3 ]
1
Masukkan angka untuk matriks indeks [ 2 ][ 0 ]
2
Masukkan angka untuk matriks indeks [ 2 ][ 1 ]
-2
Masukkan angka untuk matriks indeks [ 2 ][ 2 ]
5
Masukkan angka untuk matriks indeks [ 2 ][ 3 ]
3
Masukkan angka untuk matriks indeks [ 3 ][ 0 ]
3
Masukkan angka untuk matriks indeks [ 3 ][ 1 ]
2
Masukkan angka untuk matriks indeks [ 3 ][ 2 ]
4
Masukkan angka untuk matriks indeks [ 3 ][ 3 ]
-2
Masukkan angka untuk matriks hasil indeks [ 0 ]
5
Masukkan angka untuk matriks hasil indeks [ 1 ]
20
Masukkan angka untuk matriks hasil indeks [ 2 ]
10
Masukkan angka untuk matriks hasil indeks [ 3 ]
12
```

```
Hasil iterasi ke- 1 :  [1.6666666666666667, 1.9047619047619047, 2.095238095238095, 2.5952380952380945]
Hasil iterasi ke- 2 :  [2.9605947323337506e-16, 3.3843537414965987, 1.7965986394557827, 0.9775510204081641]
Hasil iterasi ke- 3 :  [0.48571428571428515, 3.209912536443149, 2.5031486880466476, 2.944781341107872]
Hasil iterasi ke- 4 :  [-0.5321088435374154, 3.8133000138831044, 1.9712947383034845, 0.9577262251839507]
Hasil iterasi ke- 5 :  [0.41418075801749293, 3.328490708236647, 2.5910882449772914, 3.1319383352174692]
Hasil iterasi ke- 6 :  [-0.6670930445647647, 3.901385511139066, 1.9482284211510508, 0.797202786594021]
Hasil iterasi ke- 7 :  [0.4841457789413142, 3.30155705158484, 2.6486428371009976, 3.3250613941988063]
Hasil iterasi ke- 8 :  [-0.7676790009604849, 3.9559404458494467, 1.8944109422046893, 0.5932438288180979]
Hasil iterasi ke- 9 :  [0.5839942795730156, 3.2505731256434154, 2.710685241137301, 3.5479350272775414]
Hasil iterasi ke- 10 :  [-0.8785859796870659, 4.014066373554661, 1.8282999249301652, 0.3527872538843928]
Hasil iterasi ke- 11 :  [0.7028863478689064, 3.1886524470614965, 2.7826340873464006, 3.8082501435576575]
Hasil iterasi ke- 12 :  [-1.007506215610137, 4.081382425846014, 1.750605370447866, 0.07133384332654025]
Hasil iterasi ke- 13 :  [0.8421850859829237, 3.1159631320121943, 2.866710912415784, 4.112662585818148]
Hasil iterasi ke- 14 :  [-1.1581924637442356, 4.160034286629449, 1.6596931486585849, -0.257868111669735]
Hasil iterasi ke- 15 :  [1.0051316951228688, 3.030917253879145, 2.9650350905043514, 4.468684977572151]
Hasil iterasi ke- 16 :  [-1.3344173728396982, 4.252012826471385, 1.5533610931811428, -0.6428910464258757]
Hasil iterasi ke- 17 :  [1.1957101198538362, 2.9314476923648516, 3.0800296568599315, 4.88507218586547]
Hasil iterasi ke- 18 :  [-1.5405208024119534, 4.359585713480306, 1.428999294837622, -1.09319690046238]
Hasil iterasi ke- 19 :  [1.418602460760692, 2.8151121345617827, 3.2145220097978644, 5.3720598452985495]
Hasil iterasi ke- 20 :  [-1.7815699384536725, 4.485397991129962, 1.2835512646543243, -1.6198543872418982]
Hasil iterasi ke- 21 :  [1.679287349336053, 2.679051254837237, 3.3718181945456123, 5.941618667932541]
Hasil iterasi ke- 22 :  [-2.0634901320522356, 4.632542349130462, 1.1134417917135544, -2.2358092655207815]
Hasil iterasi ke- 23 :  [1.984172657874885, 2.5199205727374148, 3.5557847252574803, 6.607749010064703]
Hasil iterasi ke- 24 :  [-2.3932112892031134, 4.804635760360028, 0.9144894137864341, -2.956202345871774]
Hasil iterasi ke- 25 :  [2.3407527817233182, 2.333808494333972, 3.770943692567326, 7.386825052053601]
Hasil iterasi ke- 26 :  [-2.7788383019579492, 5.005908461925739, 0.681803674321315, -3.798741642368555]
Hasil iterasi ke- 27 :  [2.7577928323775622, 2.11614019083175, 4.022583928802808, 8.29799729700371]
Hasil iterasi ke- 28 :  [-3.2298502853454534, 5.241307947255217, 0.4096649148380429, -4.784137651086878]
Hasil iterasi ke- 29 :  [3.2455440899188606, 1.8615651479893665, 4.316891013880329, 9.363663310628315]
Hasil iterasi ke- 30 :  [-3.757333585121889, 5.516620581642891, 0.09138368032892359, -5.936612435382096]
Hasil iterasi ke- 31 :  [3.8159959898167415, 1.5638256453002717, 4.66109932342267, 10.610018276870724]
Hasil iterasi ke- 32 :  [-4.374254291873016, 5.838613837269906, -0.280863714465265, -7.284495029470148]
Hasil iterasi ke- 33 :  [4.4831708357350415, 1.2156029347334554, 5.063669857281455, 12.067698902898927]
Hasil iterasi ke- 34 :  [-5.095776961083285, 6.215202644754082, -0.7162274994044097, -8.860917795679665]
Hasil iterasi ke- 35 :  [5.263468482400279, 0.8083373382664741, 5.534498219754277, 13.772536501375448]
Hasil iterasi ke- 36 :  [-5.939637373754365, 6.655643950414978, -1.2254093711575322, -10.704630852531633]
Hasil iterasi ke- 37 :  [6.176069461163586, 0.3320178420578134, 6.08515786387667, 15.766437761556531]
Hasil iterasi ke- 38 :  [-6.926578500431401, 7.170764261685583, -1.8209255520871253, -12.86095459313577]
Hasil iterasi ke- 39 :  [7.24340645749961, -0.22506398473200662, 6.729184578988814, 18.098414859495037]
Hasil iterasi ke- 40 :  [-8.080860385089752, 7.773225773975773, -2.5174144520708124, -15.382893707800479]
Hasil iterasi ke- 41 :  [8.491715729851457, -0.8766017954025406, 7.48240921457869, 20.825790228532025]
Hasil iterasi ke- 42 :  [-9.430856482360939, 8.477837620663973, -3.331996495909249, -18.332440094695933]
Hasil iterasi ke- 43 :  [9.951682024272882, -1.6386110700176206, 8.363346819101357, 24.015605604594416]
Hasil iterasi ke- 44 :  [-11.009751106689952, 9.301919897067068, -4.284694961253843, -21.782096685475544]
Hasil iterasi ke- 45 :  [11.659192837543392, -2.5298227926370793, 9.393651759213137, 27.74626998210428]
Hasil iterasi ke- 46 :  [-12.85635513745278, 10.265729406478036, -5.398928171690241, -25.816659643081618]
Hasil iterasi ke- 47 :  [13.656220569331651, -3.5721438784736734, 10.598650006726839, 32.10948698897748]
Hasil iterasi ke- 48 :  [-15.01606003091815, 11.39295759355366, -6.702085143597765, -30.535302740019095]
Hasil iterasi ke- 49 :  [15.991854247628922, -4.791195668755699, 12.00796167745761, 37.2125090576029]
Hasil iterasi ke- 50 :  [-17.541953589664164, 12.711312904775227, -8.226198836785986, -36.054015153292994]
Hasil iterasi ke- 51 :  [18.72350618834159, -6.21694373006162, 13.656229124614512, 43.18077380167979]
Hasil iterasi ke- 52 :  [-20.496124916227817, 14.253201891010717, -10.008733558112464, -42.50845259955594]
Hasil iterasi ke- 53 :  [21.918323249996234, -7.884434439252341, 15.58396848403413, 50.16098740381027]
Hasil iterasi ke- 54 :  [-23.951190628111352, 16.056525794391074, -12.093505873285187, -50.05727189434633]
Hasil iterasi ke- 55 :  [25.654837373672134, -9.834656460028253, 17.83856560312764, 58.32473080673523]
```

```
Hasil iterasi ke- 56 :  [-27.992079850104858, 18.165612200438158, -14.531761663823932, -58.886030902367]
Hasil iterasi ke- 57 :  [30.0248959801573, -12.11554828710486, 20.475440834515332, 67.87267735216176]
Hasil iterasi ke- 58 :  [-32.71812186090111, 20.632304656426953, -17.38343580436583, -69.21174974365637]
Hasil iterasi ke- 59 :  [35.13591967550665, -14.783176624495388, 23.559411326193008, 79.0395255411506]
Hasil iterasi ke- 60 :  [-38.245487710537596, 23.51723703993983, -20.71862542449939, -81.28824537486534]
Hasil iterasi ke- 61 :  [41.11354276176382, -17.903114563669728, 27.166284294745786, 92.09976816846756]
Hasil iterasi ke- 62 :  [-44.7100458261732, 26.891324002923227, -24.619312969441967, -95.4123706752205]
Hasil iterasi ke- 63 :  [48.1047014593586, -21.552053438648546, 31.384720445929446, 107.37443964224825]
Hasil iterasi ke- 64 :  [-52.270701799972834, 30.837504127918773, -29.181381414192305, -111.93131140042507]
Hasil iterasi ke- 65 :  [56.281245752913016, -25.81968797911485, 36.318413347443894, 125.23900734514247]
Hasil iterasi ke- 66 :  [-61.1133044545754, 35.45277864507012, -34.51697116722728, -131.25112037124754]
Hasil iterasi ke- 67 :  [65.84416364339923, -30.81092110057589, 42.08863832515847, 146.1326010148399]
Hasil iterasi ke- 68 :  [-71.45521420131514, 40.85059582369944, -40.75723659889811, -153.84669867606948]
Hasil iterasi ke- 69 :  [77.02852164318047, -36.64844252762096, 48.83723353732112, 170.568807011792]
Hasil iterasi ke- 70 :  [-83.55064598621396, 47.16363964928961, -48.05556995287376, -180.27346923577886]
Hasil iterasi ke- 71 :  [90.1092429564648, -43.475744635528805, 56.73008650466987, 199.14829280850813]
Hasil iterasi ke- 72 :  [-97.69691815893918, 54.54709133446545, -56.59137188774302, -211.18102967942937]
Hasil iterasi ke- 73 :  [105.40786537888341, -51.460649642761915, 65.9612117989995, 232.5735720235622]
Hasil iterasi ke- 74 :  [-114.24176086845432, 63.18244383532193, -66.57446133262684, -247.32912013261324]
Hasil iterasi ke- 75 :  [123.3004450324259, -60.79943485642441, 76.75752012402782, 271.66627294027006]
Hasil iterasi ke- 76 :  [-133.59186363336264, 73.28196313789486, -78.25023305565904, -289.60629842346714]
Hasil iterasi ke- 77 :  [144.22680021779345, -71.72165737352626, 89.3843960175524, 317.3873349882688]
Hasil iterasi ke- 78 :  [-156.22287219515297, 85.09390597785719, -91.9056897237572, -339.05178176238667]
Hasil iterasi ke- 79 :  [168.70132260771965, -84.49579683425193, 104.15222128064337, 370.8606296386143]
Hasil iterasi ke- 80 :  [-182.69108038744443, 98.90862225044195, -107.87649672801403, -396.8809917867527]
Hasil iterasi ke- 81 :  [197.32562153168314, -99.43585493200314, 121.4240044865771, 433.4005863386758]
Hasil iterasi ke- 82 :  [-213.64710441959042, 115.06569211420245, -126.55523318968832, -464.5154308945598]
Hasil iterasi ke- 83 :  [230.80331216174295, -116.90907390306815, 141.62430411081147, 506.5445025611692]
Hasil iterasi ke- 84 :  [-249.8518757028196, 133.96227322607766, -148.4010419651426, -543.617624258437]
Hasil iterasi ke- 85 :  [269.9573111085512, -137.3449637244565, 165.24966462185913, 592.0903321820886]
Hasil iterasi ke- 86 :  [-292.19534533928714, 156.06286329151962, -173.95091585693044, -636.131986431272]
Hasil iterasi ke- 87 :  [315.75006457136465, -161.24585991785398, 192.88082206307575, 692.1408810653445]
Hasil iterasi ke- 88 :  [-341.71836004991974, 181.91071790336596, -203.8328974578924, -744.3326170872984]
Hasil iterasi ke- 89 :  [369.3072062711129, -189.19927148154719, 225.19697915131505, 809.1554962277523]
Hasil iterasi ke- 90 :  [-399.63824727421417, 212.14120433186417, -238.78151709422008, -870.8792007678973]
Hasil iterasi ke- 91 :  [431.94522670323676, -221.89232247538718, 262.9925007892888, 946.0105191580456]
Hasil iterasi ke- 92 :  [-467.3787383504717, 247.49741951595823, -279.65584834825535, -1018.88238470626]
Hasil iterasi ke- 93 :  [505.2038338494355, -260.1286422480354, 307.1964403847677, 1106.0699892956532]
Hasil iterasi ke- 94 :  [-546.604965319501, 288.84845616237914, -327.4606249846398, -1191.9802417861522]
Hasil iterasi ke- 95 :  [590.8838008084284, -304.84811948592585, 358.89537695394966, 1293.268335634616]
Hasil iterasi ke- 96 :  [-639.2643916097855, 337.21076595233933, -383.3709383559197, -1394.4276981741784]
Hasil iterasi ke- 97 :  [691.0912306800325, -357.1500056591015, 419.3601243688534, 1512.2070890986543]
Hasil iterasi ke- 98 :  [-747.6346827231179, 393.7731449857682, -448.76112237563814, -1631.201123850185]
Hasil iterasi ke- 99 :  [808.2893267796545, -418.3199357707639, 490.0769692899436, 1768.267992978605]
Hasil iterasi ke- 100 :  [-874.3796936321675, 459.9259556314136, -525.2385360817307, -1908.120656980299]
```

3. Analisis Hasil

Tiga pengujian pada subbab Hasil Awal Simulasi menunjukkan 3 kondisi yang berbeda. Proses iterasi pada percobaan 1 dan 3 menunjukkan hasil yang konvergen. Sementara itu, pada proses iterasi percobaan 2 menunjukkan hasil yang tak konvergen. Hal ini bisa disebabkan oleh banyak hal, salah satunya adalah bentuk matriks yang memang tidak memiliki penyelesaian rill.

Kode di atas menunjukkan proses kegiatan sehari-hari, yaitu mengisi air dengan beberapa selang sekaligus. Tidak semua selang bisa memiliki debit air yang sama. Oleh karena itu, pada percobaan 1 dan 3 penulis mencoba menemukan debit air dari masing-masing selang untuk mengisi ember dalam waktu tertentu. Dapat dilihat pada percobaan 1 dan 3 ditemukan debit air yang mengalir dari tiap selang untuk mengisi air pada sebuah ember dengan volume tertntu. Apabila dinyatakan ke dalam sebuah persamaan linier, akan dijumpai bentuk umum persamaan sederhana yaitu

$$Ax + By + Cz = v_1$$

$$Dx + Ey + Fz = v_2$$

$$Gx + Hy + Iz = v_3$$

Persamaan inilah yang kemudian akan dinyatakan ke dalam bentuk matriks seperti demikian

$$\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Setelah diubah ke dalam bentuk matriks Ax = b, maka metode Gauss-Seidel bisa digunakan. Berikut merupakan bentuk umum dari metode Gauss-Seidel

$$x = \frac{v_1 - By - Cz}{A}$$

$$y = \frac{v_2 - Ax - Cz}{B}$$

$$z = \frac{v_3 - Ax - By}{C}$$

D. Kesimpulan

Metode Gauss-Seidel merupakan salah satu cara dalam metode numeris yang dapat digunakan untuk menemukan akar persamaan dari sistem persamaan linier. Metode ini menggunakan prinsip iterasi untuk setiap penghitungannya dan nilai eror yang juga menjadi parameter apakah perhitungan harus dilanjutkan atau berhenti karena akar persamaannya sudah ditemukan.

Untuk menggunakan metode Gauss-Seidel dibutuhkan beberapa fungsi yang akan dicari akar persamaannya. Umumnya, fungsi yang digunakan adalah persamaan linier. Persamaan linier ini nantinya akan diubah ke dalam bentuk matriks $Ax = b$ kemudian proses penghitungan Gauss Seidel dimulai. Salah stau kelebihan metode Gauss-Seidel apabila dibandingkan dengan metode Jacobi adalah untuk setiap nilai x_1, x_2, …, x_n, pada proses iterasi penghitungan menggunakan nilai x terbaru untuk masing-masing indeks. Hal inilah yang membuat proses penghitungan untuk mencapai kesimpulan konvergen/divergen pada metode Gauss-Seidel tergolong lebih cepat daripada metode Jacobi.