

SK5001 ANALISIS NUMERIK LANJUT

Tugas Kuliah
Sistem Persamaan Non Linear

Mohammad Rizka Fadhli
NIM: 20921004

03 October 2021

Contents

1 PENDAHULUAN	5
1.1 Penulisan Tugas	5
1.2 Bahasa Pemrograman yang Dipakai	5
1.3 <i>Libraries R</i> yang Digunakan	5
2 METODE	6
2.1 Program Menggambar Grafik	6
2.1.1 2 Peubah (Soal 3a dan 3b)	6
2.1.2 3 Peubah (Soal 4a dan 4b)	7
2.2 Program Metode Newton untuk SPNL	8
2.3 Program Metode Broyden untuk SPNL	9
2.4 Norm ∞	10
2.5 Mencari Turunan Parsial Fungsi	10
3 TASK 3a	11
Soal	11
Jawab	11
Grafik Sistem Persamaan Non Linear	11
Penyelesaian dengan Metode Newton	13
Kesimpulan	19
4 TASK 3b	20
Soal	20
Jawab	20
Grafik Sistem Persamaan Non Linear	20
Penyelesaian dengan Metode Newton	21
Kesimpulan	30
5 TASK 4a	31
Soal	31
Jawab	31
Grafik Sistem Persamaan Non Linear	31

Penyelesaian dengan Metode Broyden	33
Kesimpulan	41
6 TASK 4b	42
Soal	42
Jawab	42
Grafik Sistem Persamaan Non Linear	42
6.0.1 Penyelesaian dengan Metode Broyden	44
Kesimpulan	52

List of Figures

1	Definisi Matriks Jacobi	8
2	Definisi Matriks Jacobi	9
3	Grafik Soal 3a	12
4	Initial Points Soal 3a	13
5	Solusi Pertama dari Soal 3a	16
6	Solusi Kedua dari Soal 3a	18
7	Solusi Pertama dan Kedua dari Soal 3a	19
8	Grafik Soal 3b	21
9	Initial Points Soal 3b	22
10	Solusi Pertama Soal 3b	25
11	Solusi Kedua Soal 3b	26
12	Solusi Ketiga Soal 3b	28
13	Solusi Keempat Soal 3b	29
14	Semua Titik Solusi dari Soal 3b	30
15	Grafik Soal 4a	33
16	Grafik Soal 4b	44

1 PENDAHULUAN

1.1 Penulisan Tugas

Di dalam tugas ini disertakan juga *source code* dan grafik yang diperlukan untuk menjawab soal yang ada.

1.2 Bahasa Pemrograman yang Dipakai

Bahasa pemrograman yang digunakan pada tugas ini adalah **R** dengan versi **R** yang digunakan adalah 4.1.1. Format tugas ini ditulis menggunakan *LaTex R Markdown* di *software R Studio*. Semua perhitungan numerik (metode Newton dan Broyden) diselesaikan dengan menulis *program* berdasarkan buku **Numerical Analysis**¹ dan materi *handout* perkuliahan yang ada². Sedangkan untuk menggambar grafik dibuat program tersendiri.

1.3 *Libraries R* yang Digunakan

Berikut adalah beberapa *libraries* yang digunakan dalam mengerjakan dan menuliskan tugas ini:

1. **dplyr**: untuk *data carpentry*.
2. **ggplot2**: sebagai visualisasi data (grafik).
3. **matlib**: sebagai dasar perhitungan aljabar (invers matriks).
4. **Ryacas**: digunakan untuk mencari turunan parsial dari fungsi $f(x_1, x_2)$ atau $f(x_1, x_2, x_3)$.

¹Richard L. Burden dan J. Douglas Faires

²Prof. Kuntjoro Adji Sidarto

2 METODE

2.1 Program Menggambar Grafik

2.1.1 2 Peubah (Soal 3a dan 3b)

Oleh karena fungsi yang ada pada soal tidak bisa dituliskan secara eksplisit menjadi hubungan $y \sim x$, maka grafik dari soal digambar dengan pendekatan sebagai berikut:

- **STEP 1** Tentukan selang di sumbu x dan y , misalkan $x, y \in [a, b]$. Lalu generate sebanyak-banyaknya titik di $[a, b]$ dengan Δ yang **kecil** (kita akan definisikan deltanya). Semua titik yang di-*generate* dinotasikan sebagai (x', y') .
- **STEP 2** dari semua titik (x', y') tersebut, kita akan hitung nilai hampiran fungsinya $f(x, y)$.
- **STEP 3** kita hanya akan menggambar (x', y') yang memenuhi $f(x, y) = 0$.

Berikut adalah program yang saya buat di R:

```
data_ke_grafik = function(a,b,delta,f){  
  # generate selang  
  selang = seq(a,b,by = delta)  
  # menghitung (x,y) yang memenuhi f(x,y) = 0  
  df =  
    # mengeluarkan semua kombinasi yang mungkin dari selang  
    expand.grid(selang,selang) %>%  
    as.data.frame() %>%  
    # mengubah nama variabel menjadi x,y  
    rename(x = Var1,  
           y = Var2) %>%  
    # menghitung nilai f(x,y)  
    mutate(f = f(x,y)) %>%  
    # hanya mengambil (x,y) yang memenuhi f(x,y) = 0  
    filter(round(f,1) == 0)  
  # output berupa data frame yang siap diplot  
  return(df)  
}
```

Program tersebut berfungsi untuk men-*generate* titik. Grafik digambar dengan perintah tambahan ggplot2 di masing-masing soal.

2.1.2 3 Peubah (Soal 4a dan 4b)

Berbeda dengan soal 3 yang hanya memiliki 2 peubah pada fungsinya. Soal 4 memiliki 3 peubah pada fungsinya. Untuk menggambar grafiknya, saya akan menggunakan pendekatan 2 dimensi saja. Yakni melakukan plot antara $(x_1, x_2 + x_3)$. Selain itu, pada soal 4, selang x_1, x_2, x_3 sudah ditetapkan.

Oleh karena itu, program di R-nya harus dimodifikasi dari bagian 2.1.1 sebagai berikut:

```
data_ke_grafik3d = function(x1_lower,x1_upper, # selang x1
                            x2_lower,x2_upper, # selang x2
                            x3_lower,x3_upper, # selang x3
                            delta,f){
  # generate selang
  selang_x1 = seq(x1_lower,x2_upper,by = delta)
  selang_x2 = seq(x2_lower,x2_upper,by = delta)
  selang_x3 = seq(x3_lower,x3_upper,by = delta)

  # menghitung (x1,x2,x3) yang memenuhi f(x1,x2,x3) = 0
  df =
    # mengeluarkan semua kombinasi yang mungkin dari selang
    expand.grid(selang_x1,selang_x2,selang_x3) %>%
    as.data.frame() %>%
    # mengubah nama variabel menjadi x,y
    rename(x1 = Var1,
           x2 = Var2,
           x3 = Var3) %>%
    # menghitung nilai f(x1,x2,x3)
    mutate(f = f(x1,x2,x3)) %>%
    # hanya mengambil (x1,x2,x3) yang memenuhi f(x1,x2,x3) = 0
    filter(round(f,2) == 0) %>%
    mutate(x2_x3 = x2 + x3)
  # output berupa data frame yang siap diplot
  return(df)
}
```

Kemudian grafik akan dibuat dengan perintah di ggplot2.

2.2 Program Metode Newton untuk SPNL

Misalkan suatu sistem persamaan non linear memiliki bentuk sebagai berikut:

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n, K) &= 0 \\ f_2(x_1, x_2, \dots, x_n, K) &= 0 \\ &\dots \\ f_n(x_1, x_2, \dots, x_n, K) &= 0 \end{aligned}$$

dengan $f_i, i = 1, 2, \dots, n$ merupakan persamaan non linear.

Penyelesaian dengan metode Newton menggunakan prinsip deret Taylor. Skema iterasi³ yang akan saya gunakan adalah sebagai berikut:

$$x^{(k+1)} = x^{(k)} - J(x^{(k)})^{-1} F(x^{(k)})$$

Dengan J merupakan matriks Jacobi yang didefinisikan sebagai:

Define the matrix $J(\mathbf{x})$ by

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{bmatrix}.$$

Figure 1: Definisi Matriks Jacobi

Kelak program yang ditulis akan memanfaatkan skema iterasi tersebut. Oleh karena **R** bisa melakukan operasi matriks, saya akan memanfaatkan operasi tersebut dalam membuat programnya.

Langkah yang kritis pada metode ini adalah penentuan titik *initial* iterasi. Sebisa mungkin kita akan pilih titik *initial* yang dekat dengan solusi berdasarkan grafik yang ada.

Selain itu, kita perlu pertimbangkan juga persamaan yang ada di matriks Jacobi-nya. Jangan sampai ada pembagian dengan **nol** akibat kita salah memilih titik *initial*.

Metode ini akan dipakai untuk menyelesaikan soal 3a dan 3b.

³Persamaan 10.9 dari buku Numerical Analysis 9th Edition hal. 640

2.3 Program Metode Broyden untuk SPNL

Metode Broyden merupakan metode modifikasi dari metode Newton. Perbedaan mendasar dari metode ini adalah kita menggunakan hampiran matriks Jacobi. Berikut adalah algoritma 10.2 dari buku:

**ALGORITHM
10.2**

Broyden

To approximate the solution of the nonlinear system $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ given an initial approximation \mathbf{x} :

INPUT number n of equations and unknowns; initial approximation $\mathbf{x} = (x_1, \dots, x_n)^t$; tolerance TOL ; maximum number of iterations N .

OUTPUT approximate solution $\mathbf{x} = (x_1, \dots, x_n)^t$ or a message that the number of iterations was exceeded.

Step 1 Set $A_0 = J(\mathbf{x})$ where $J(\mathbf{x})_{i,j} = \frac{\partial f_i}{\partial x_j}(\mathbf{x})$ for $1 \leq i, j \leq n$;
 $\mathbf{v} = \mathbf{F}(\mathbf{x})$. (Note: $\mathbf{v} = \mathbf{F}(\mathbf{x}^{(0)})$.)

Step 2 Set $A = A_0^{-1}$. (Use Gaussian elimination.)

Step 3 Set $\mathbf{s} = -A\mathbf{v}$; (Note: $\mathbf{s} = \mathbf{s}_1$.)
 $\mathbf{x} = \mathbf{x} + \mathbf{s}$; (Note: $\mathbf{x} = \mathbf{x}^{(1)}$.)
 $k = 2$.

Step 4 While ($k \leq N$) do Steps 5–13.

Step 5 Set $\mathbf{w} = \mathbf{v}$; (Save \mathbf{v} .)
 $\mathbf{v} = \mathbf{F}(\mathbf{x})$; (Note: $\mathbf{v} = \mathbf{F}(\mathbf{x}^{(k)})$.)
 $\mathbf{y} = \mathbf{v} - \mathbf{w}$. (Note: $\mathbf{y} = \mathbf{y}_k$.)

Step 6 Set $\mathbf{z} = -A\mathbf{y}$. (Note: $\mathbf{z} = -A_{k-1}^{-1}\mathbf{y}_k$.)

Step 7 Set $p = -\mathbf{s}'\mathbf{z}$. (Note: $p = \mathbf{s}_k^t A_{k-1}^{-1} \mathbf{y}_k$.)

Step 8 Set $\mathbf{u}' = \mathbf{s}'A$.

Step 9 Set $A = A + \frac{1}{p}(\mathbf{s} + \mathbf{z})\mathbf{u}'$. (Note: $A = A_k^{-1}$.)

Step 10 Set $\mathbf{s} = -A\mathbf{v}$. (Note: $\mathbf{s} = -A_k^{-1}\mathbf{F}(\mathbf{x}^{(k)})$.)

Step 11 Set $\mathbf{x} = \mathbf{x} + \mathbf{s}$. (Note: $\mathbf{x} = \mathbf{x}^{(k+1)}$.)

Step 12 If $\|\mathbf{s}\| < TOL$ then OUTPUT (\mathbf{x});
(The procedure was successful.)
STOP.

Step 13 Set $k = k + 1$.

Step 14 OUTPUT ('Maximum number of iterations exceeded');
(The procedure was unsuccessful.)
STOP.

Figure 2: Definisi Matriks Jacobi

Langkah yang kritis pada metode ini adalah penentuan titik *initial* iterasi. Sebisa mungkin kita akan pilih titik *initial* yang dekat dengan solusi berdasarkan grafik yang ada.

Metode ini akan dipakai untuk menyelesaikan soal 4a dan 4b.

2.4 Norm ∞

Secara *default*, **R** memiliki *function* khusus untuk menghitung **norm**, yakni dengan perintah **norm()**. Namun perlu diperhatikan bahwa input dari **norm()** adalah berupa matriks. Sedangkan norm yang akan saya gunakan sebagai kriteria penghentian iterasi adalah norm vektor dari $x^{(k)}$.

Berikut adalah **R function** untuk menghitung norm vektor:

```
# menghitung norm infinity dari vektor v
norm_vec_inf = function(x)max(abs(x))
```

2.5 Mencari Turunan Parsial Fungsi

Untuk mencari turunan parsial dari suatu fungsi peubah banyak, saya menggunakan bantuan **library(Ryacas)**.

3 TASK 3a

Soal

Cari aproksimasi solusi dari sistem persamaan non linear berikut:

$$4x_1^2 - 20x_1 + \frac{1}{4}x_2^2 + 8 = 0$$

$$\frac{1}{2}x_1x_2^2 + 2x_1 - 5x_2 + 8 = 0$$

Jawab

Grafik Sistem Persamaan Non Linear

Untuk membantu kita menjawab soal tersebut, pertama-tama kita perlu membuat grafik fungsi dari SPNL tersebut:

```
# initial condition yang dibutuhkan untuk menggambar
a = -10
b = 10
delta = 0.005
# fungsi dari soal
f1 = function(x1,x2){4 * x1^2 - 20* x1 + (1/4) * x2^2 + 8}
f2 = function(x1,x2){(1/2) * x1 * x2^2 + 2 * x1 - 5 * x2 + 8}
# proses generator data ke grafik
# menggunakan program yang dibuat di bagian 2.1.1
df1 = data_ke_grafik(a,b,delta,f1)
df2 = data_ke_grafik(a,b,delta,f2)

# proses menggambar plot
# membuat canvas
plot_soal_3a =
  ggplot() +
  # memplot f1
  geom_point(data = df1,
             aes(x,y),
             size = .1,
             color = "steelblue") +
  # memplot f2
  geom_point(data = df2,
             aes(x,y),
             size = .1,
```

```

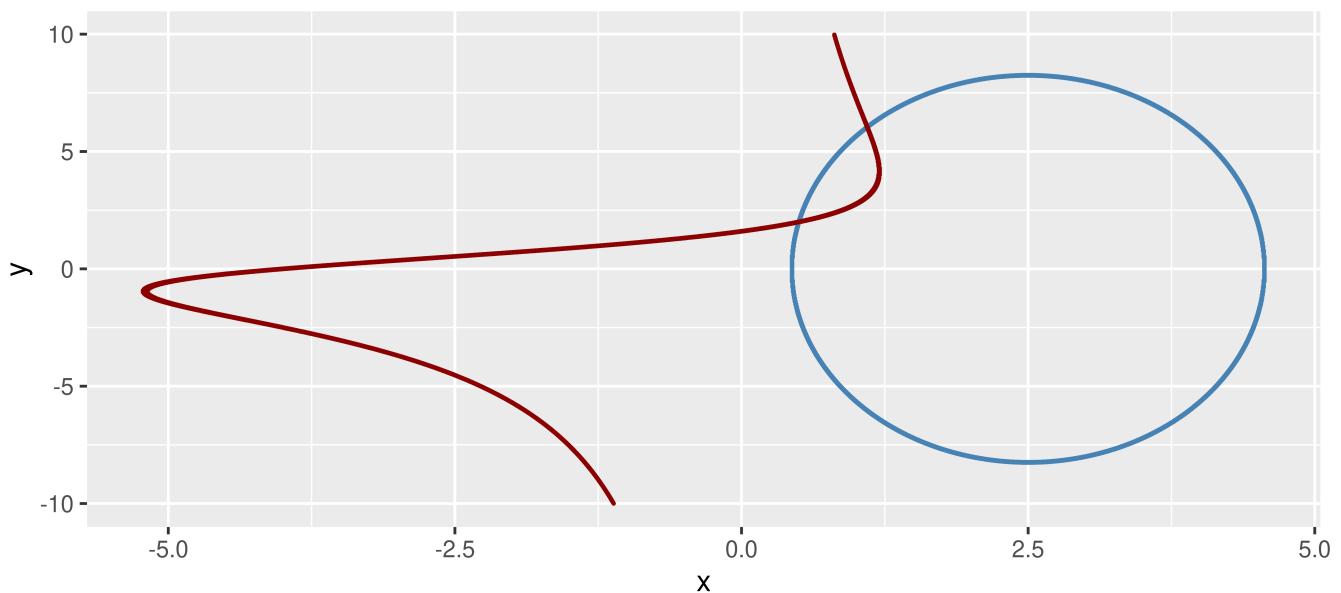
        color = "darkred") +
# menampilkan label
labs(subtitle = "Grafik f1(x1,x2) dan f2(x1,x2)",
     title = "Soal 3a",
     caption = caption)

# menampilkan grafik
plot_soal_3a

```

Soal 3a

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 3: Grafik Soal 3a

Jika kita lihat grafik di atas, ada dua titik solusi yang akan kita cari. Oleh karena kita akan gunakan metode Newton, maka diperlukan dua sembarang *initial points*. Diharapkan iterasi dari dua *initial points* tersebut akan konvergen ke dua titik solusi yang dicari.

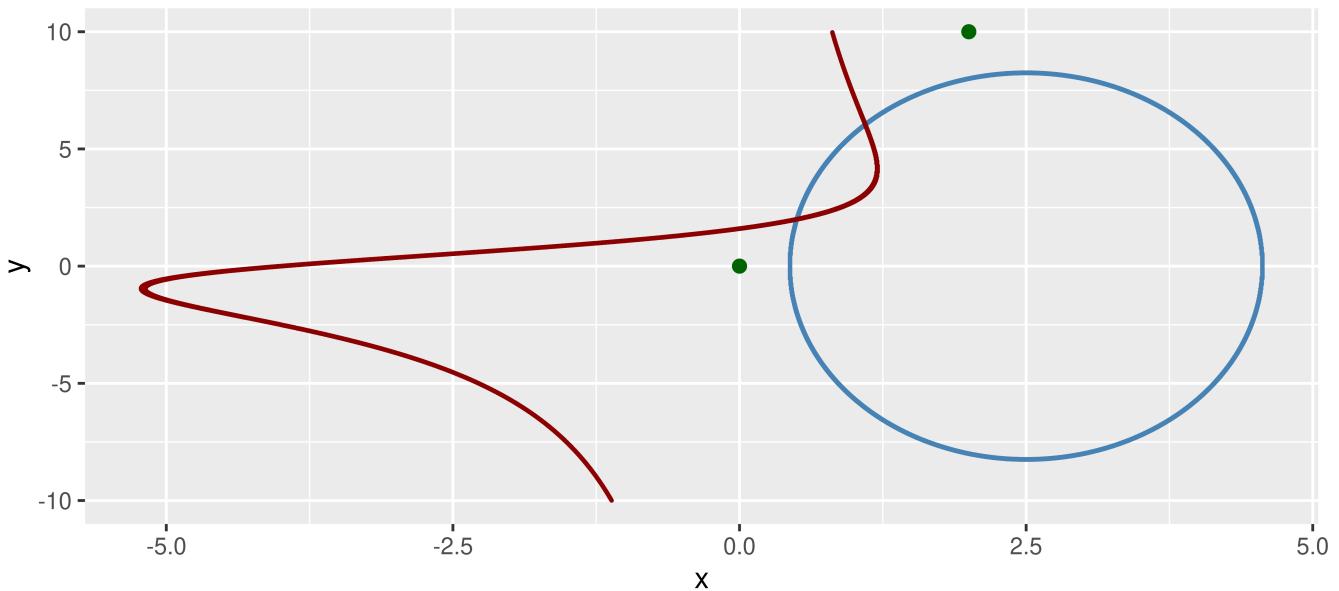
Penyelesaian dengan Metode Newton

Kita akan mengambil dua titik *initial* berikut: $(0, 0)$ dan $(2, 10)$ secara sembarang. Perlu diperhatikan bahwa perbedaan initial points yang diambil akan mempengaruhi seberapa banyak iterasi yang diperlukan menuju konvergen ke titik solusi.

Berikut adalah grafiknya:

Soal 3a

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 4: Initial Points Soal 3a

Titik hijau adalah *initial points* yang kita pilih.

Untuk menyelesaikan dengan metode Newton, kita perlu mencari persamaan di matriks Jacobi terlebih dahulu. Untuk membantu kita mencari turunan parsial dari kedua fungsi soal, kita akan gunakan `library(Ryacas)` di **R** berikut:

```
# f1
f1 = "4*x1^2 - 20*x1 + (1/4)*x2^2 + 8"
# turunan parsial f1 thd x1
f1 %>% y_fn("D(x1)") %>% yac_str()
```

```
## [1] "8*x1-20"
```

```

# turunan parsial f1 thd x2
f1 %>% y_fn("D(x2)") %>% yac_str()

## [1] "x2/2"

# f2
f2 = "(1/2) * x1 * x2^2 + 2 * x1 - 5 * x2 + 8"
# turunan parsial f2 thd x1
f2 %>% y_fn("D(x1)") %>% yac_str()

## [1] "x2^2/2+2"

# turunan parsial f2 thd x2
f2 %>% y_fn("D(x2)") %>% yac_str()

## [1] "x2*x1-5"

```

Berikut adalah matriks Jacobi yang sudah kita dapatkan:

$$J(x) = \begin{bmatrix} 8x_1 - 20 & \frac{x_2}{2} \\ \frac{x_2^2}{2} + 2 & x_2x_1 - 5 \end{bmatrix}$$

Sekarang kita tinggal melakukan iterasi dengan skema berikut:

$$x^{(k+1)} = x^{(k)} - J(x^{(k)})^{-1}F(x^{(k)})$$

Titik Pertama Mari kita coba selesaikan titik pertama terlebih dahulu.

```

# initial
x0 = c(0,0)

# bikin fungsi F(x1,x2)
F_x_k = function(x){
  f1 = 4 * x[1]^2 - 20* x[1] + (1/4) * x[2]^2 + 8
  f2 = (1/2) * x[1] * x[2]^2 + 2 * x[1] - 5 * x[2] + 8
  xk = c(f1,f2)
  return(xk)
}

# bikin matriks jacobi
jax = function(x){

```

```

a11 = 8*x[1]-20
a12 = x[2]/2
a21 = x[2]^2/2+2
a22 = x[2]*x[1]-5
J = matrix(c(a11,a12,a21,a22),ncol = 2,byrow = T)
J_inv = matlib::inv(J)
return(J_inv)
}

# set toleransi max yang diinginkan
tol_max = 0.000001

# set max iterasi yang diperbolehkan
iter_max = 40

# kita mulai iterasinya
iter = 0
while(norm_vec_inf(F_x_k(x0)) > tol_max && iter <= iter_max){
  xk_new = x0 - jax(x0) %*% F_x_k(x0)
  x0 = xk_new
  iter = iter + 1
  pesan = paste0("Iterasi ke-",iter,
                 " menghasilkan: x1 = ",x0[1],
                 " dan x2 = ",x0[2])
  print(pesan)
}

## [1] "Iterasi ke-1 menghasilkan: x1 = 0.4 dan x2 = 1.76"
## [1] "Iterasi ke-2 menghasilkan: x1 = 0.4958936141312 dan x2 = 1.9834234744192"
## [1] "Iterasi ke-3 menghasilkan: x1 = 0.499987614677558 dan x2 = 1.99993704911406"
## [1] "Iterasi ke-4 menghasilkan: x1 = 0.49999999849583 dan x2 = 1.99999999921371"

list("Solusi Final: " = x0,
     "Banyak iterasi: " = iter)

## $`Solusi Final: `
##      [,1]
## [1,]  0.5
## [2,]  2.0
##
## $`Banyak iterasi: `
## [1] 4

```

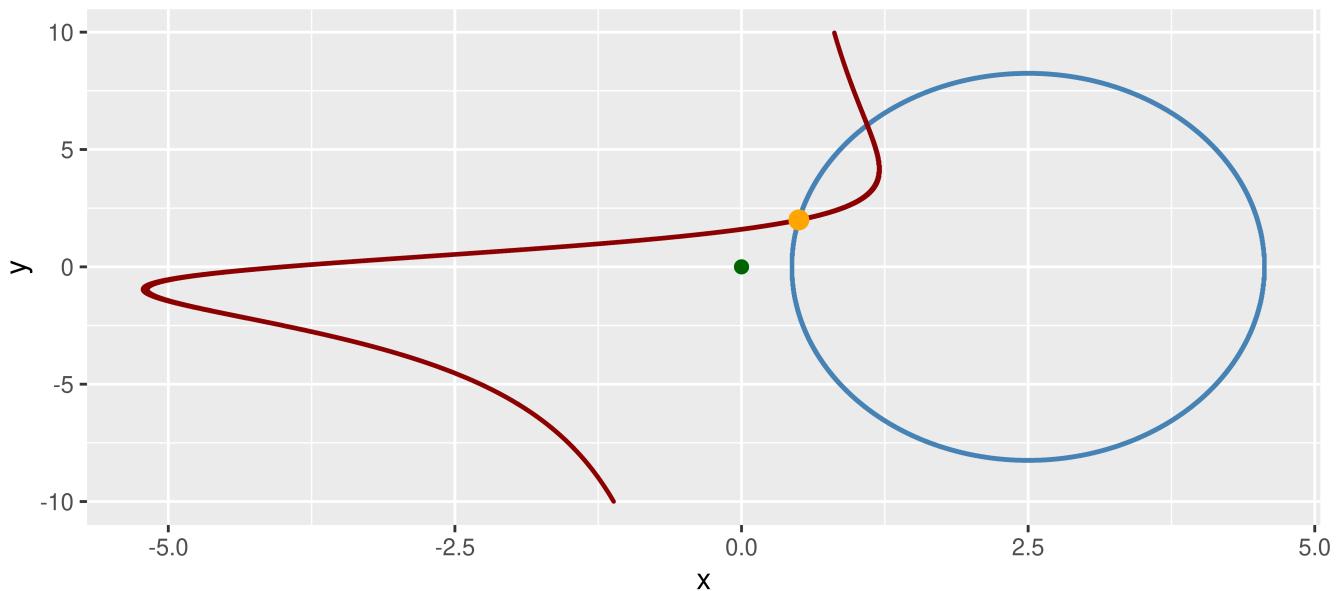
Didapatkan pada iterasi ke 4, kita mendapatkan solusi finalnya, yakni titik $(0.5, 2)$.

Saya gambarkan kembali ke grafik fungsinya sebagai berikut:

```
# menampilkan grafik
plot_soal_3a +
  geom_point(aes(x = 0,
                 y = 0),
             color = "darkgreen",
             size = 2) +
  geom_point(aes(x = 0.5,
                 y = 2.0),
             color = "orange",
             size = 3)
```

Soal 3a

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 5: Solusi Pertama dari Soal 3a

Titik **hijau** adalah *initial point* sedangkan titik **oranye** adalah solusinya.

Titik Kedua Mari kita selesaikan titik berikutnya.

```
# initial
x0 = c(2,10)

# kita mulai iterasinya
iter = 0
while(norm_vec_inf(F_x_k(x0)) > tol_max && iter <= iter_max){
  xk_new = x0 - jax(x0) %*% F_x_k(x0)
  x0 = xk_new
  iter = iter + 1
  pesan = paste0("Iterasi ke-",iter,
                 " menghasilkan: x1 = ",x0[1],
                 " dan x2 = ",x0[2])
  print(pesan)
}

## [1] "Iterasi ke-1 menghasilkan: x1 = 1.453125 dan x2 = 7.7625"
## [1] "Iterasi ke-2 menghasilkan: x1 = 1.19232471149026 dan x2 = 6.56904606464121"
## [1] "Iterasi ke-3 menghasilkan: x1 = 1.10877439050756 dan x2 = 6.11168783886561"
## [1] "Iterasi ke-4 menghasilkan: x1 = 1.09697360950515 dan x2 = 6.04245746101651"
## [1] "Iterasi ke-5 menghasilkan: x1 = 1.09671982786698 dan x2 = 6.04093366034848"
## [1] "Iterasi ke-6 menghasilkan: x1 = 1.09671970770767 dan x2 = 6.04093293624624"

list("Solusi Final: " = x0,
     "Banyak iterasi: " = iter)

## $`Solusi Final: `
##      [,1]
## [1,] 1.096720
## [2,] 6.040933
##
## $`Banyak iterasi: `
## [1] 6
```

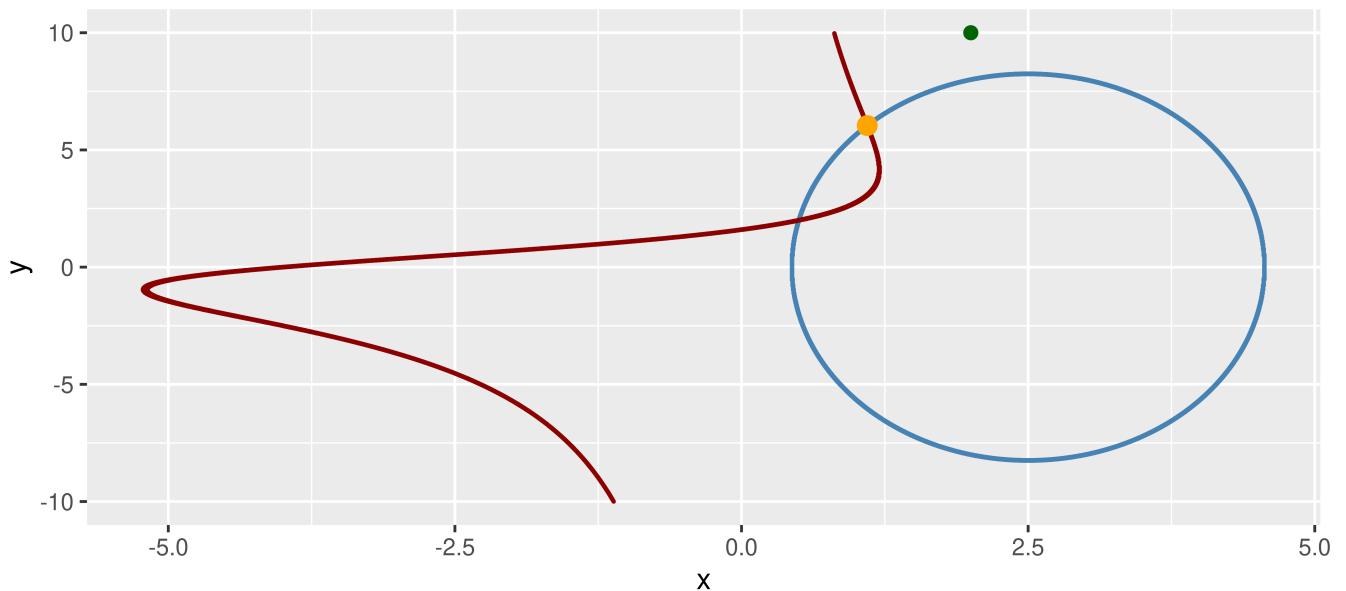
Didapatkan pada iterasi ke 6, kita mendapatkan solusi finalnya, yakni titik $(1.096720, 6.040933)$.

Saya gambarkan kembali ke grafik fungsinya sebagai berikut:

```
# menampilkan grafik
plot_soal_3a +
  geom_point(aes(x = 2,
                 y = 10),
             color = "darkgreen",
             size = 2) +
  geom_point(aes(x = 1.096720,
                 y = 6.040933),
             color = "orange",
             size = 3)
```

Soal 3a

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 6: Solusi Kedua dari Soal 3a

Titik **hijau** adalah *initial point* sedangkan titik **oranye** adalah solusinya.

Kesimpulan

Kita dapatkan ada dua solusi dari SPNL ini, yakni:

1. $(0.5, 2)$
2. $(1.096720, 6.040933)$

Berikut adalah grafiknya:

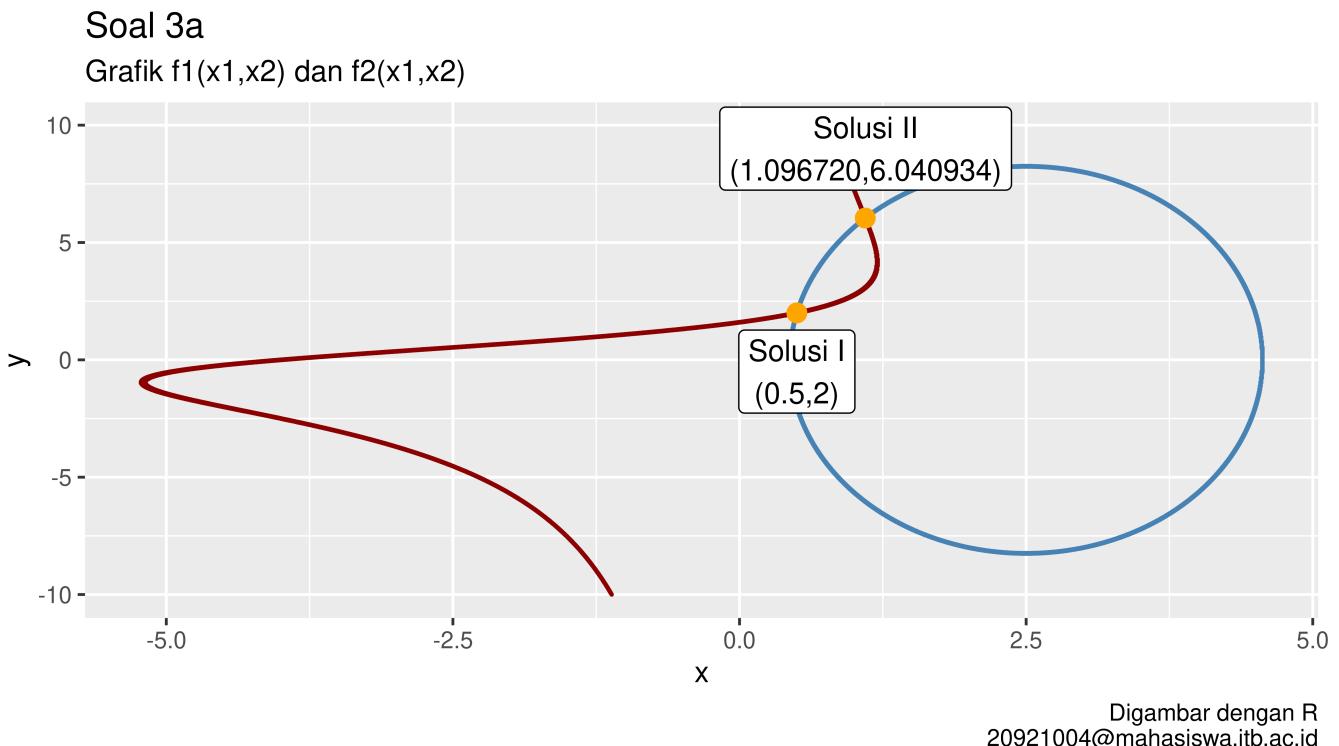


Figure 7: Solusi Pertama dan Kedua dari Soal 3a

4 TASK 3b

Soal

Cari aproksimasi solusi dari sistem persamaan non linear berikut:

$$\sin(4\pi x_1 x_2) - 2x_2 - x_1 = 0$$

$$\left(\frac{4\pi - 1}{4\pi}\right)(e^{2x_1} - e) + 4ex_2^2 - 2ex_1 = 0$$

Jawab

Grafik Sistem Persamaan Non Linear

Untuk membantu kita menjawab soal tersebut, pertama-tama kita perlu membuat grafik fungsi dari SPNL tersebut:

```
# initial condition yang dibutuhkan untuk menggambar
a = -2
b = 2
delta = 0.001
# fungsi dari soal
f1 = function(x1,x2){sin(4*pi*x1*x2) - 2*x2 - x1}
f2 = function(x1,x2){((4*pi - 1)/(4*pi))*(exp(2*x1) - exp(1)) + 4*exp(1)*x2^2 - 2*exp(1)
# proses generator data ke grafik
# menggunakan program yang dibuat di bagian 2.1.1
df1 = data_ke_grafik(a,b,delta,f1)
df2 = data_ke_grafik(a,b,delta,f2)

# proses menggambar plot
# membuat canvas
plot_soal_3b =
  ggplot() +
  # memplot f1
  geom_point(data = df1,
             aes(x,y),
             size = .1,
             color = "green") +
  # memplot f2
  geom_point(data = df2,
             aes(x,y),
             size = .1,
```

```

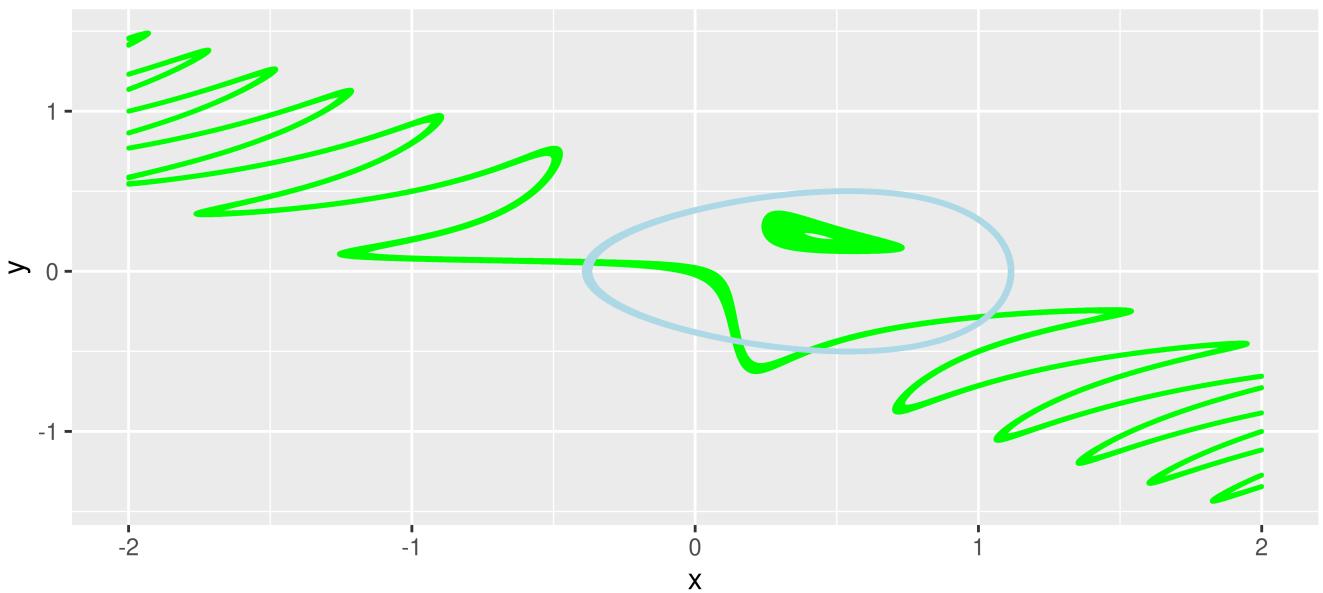
        color = "lightblue") +
# menampilkan label
labs(subtitle = "Grafik f1(x1,x2) dan f2(x1,x2)",
     title = "Soal 3b",
     caption = caption)

# menampilkan grafik
plot_soal_3b

```

Soal 3b

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 8: Grafik Soal 3b

Terlihat ada 4 solusi titik dari grafik di atas.

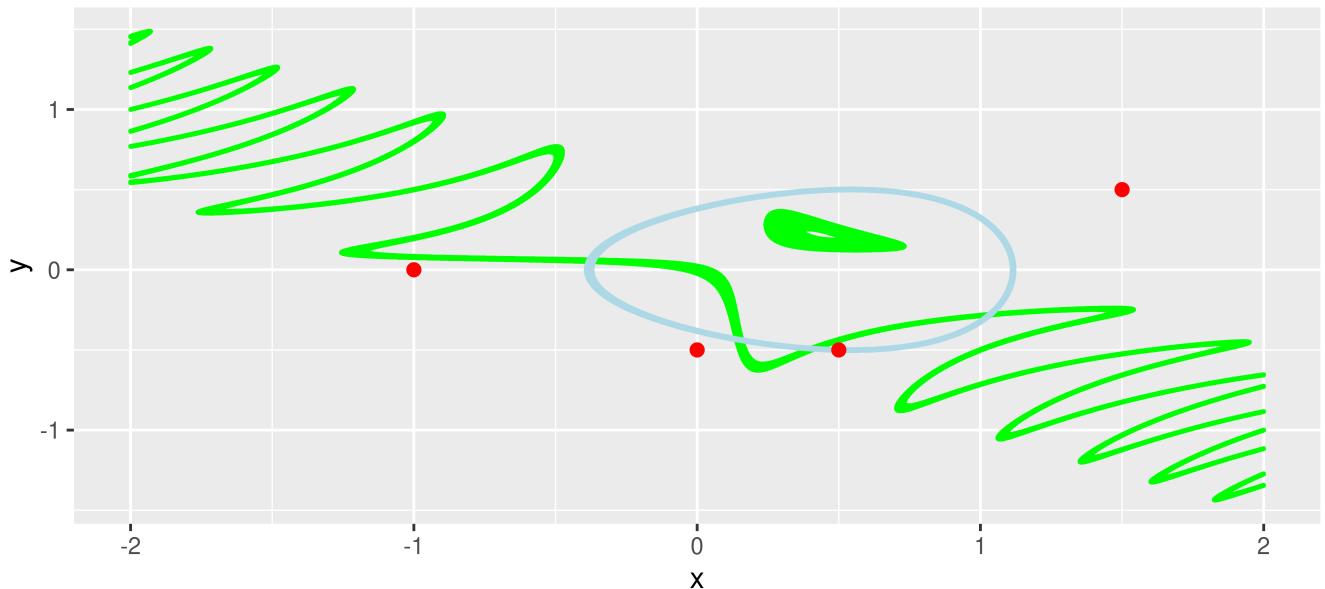
Penyelesaian dengan Metode Newton

Sama halnya dengan soal sebelumnya, kita perlu mengambil empat titik *initial* sembarang. Kita akan pilih titik-titik berikut:

1. $(-1, 0)$
2. $(0, -0.5)$
3. $(0.5, -0.5)$
4. $(1.5, 0.5)$

Soal 3b

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 9: Initial Points Soal 3b

Berikut adalah grafiknya:

Titik merah adalah *initial points* yang kita pilih.

Untuk menyelesaikan dengan metode Newton, kita perlu mencari persamaan di matriks Jacobi terlebih dahulu. Untuk membantu kita mencari turunan parsial dari kedua fungsi soal, kita akan gunakan **library(Ryacas)** di **R** berikut:

```
# f1
f1 = "Sin(4*pi*x1*x2) - 2*x2 - x1"
# turunan parsial f1 thd x1
f1 %>% y_fn("D(x1)") %>% yac_str()

## [1] "4*pi*x2*Cos(4*pi*x1*x2)-1"

# turunan parsial f1 thd x2
f1 %>% y_fn("D(x2)") %>% yac_str()

## [1] "4*pi*x1*Cos(4*pi*x1*x2)-2"
```

```

# f2
f2 = "((4*pi - 1)/(4*pi))*(Exp(2*x1) - Exp(1)) + 4*Exp(1)*x2^2 - 2*Exp(1)*x1"
# turunan parsial f2 thd x1
f2 %>% y_fn("D(x1)") %>% yac_str()

## [1] "(Exp(2*x1)*(4*pi-1))/(2*pi)-2*Exp(1)"

# turunan parsial f2 thd x2
f2 %>% y_fn("D(x2)") %>% yac_str()

## [1] "4*Exp(1)*2*x2"

```

Berikut adalah matriks Jacobi yang sudah kita dapatkan:

$$J(x) = \begin{bmatrix} 4\pi x_2 \cos(4\pi x_1 x_2) - 1 & 4\pi x_1 \cos(4\pi x_1 x_2) - 2 \\ \frac{e^{(2x_1)(4\pi-1)}}{2\pi} - 2e & 4e2x_2 \end{bmatrix}$$

Sekarang kita tinggal melakukan iterasi dengan skema berikut:

$$x^{(k+1)} = x^{(k)} - J(x^{(k)})^{-1} F(x^{(k)})$$

Titik Pertama Mari kita coba selesaikan titik pertama terlebih dahulu.

```

# initial
x0 = c(-1,0)

# bikin fungsi F(x1,x2)
F_x_k = function(x){
  f1 = sin(4*pi*x[1]*x[2]) - 2*x[2] - x[1]
  f2 = ((4*pi - 1)/(4*pi))*(exp(2*x[1]) - exp(1)) + 4*exp(1)*x[2]^2 - 2*exp(1)*x[1]
  xk = c(f1,f2)
  return(xk)
}

# bikin matriks jacobi
jax = function(x){
  a11 = 4*pi*x[2]*cos(4*pi*x[1]*x[2])-1
  a12 = 4*pi*x[1]*cos(4*pi*x[1]*x[2])-2
  a21 = (exp(2*x[1])*(4*pi-1))/(2*pi)-2*exp(1)
  a22 = 4*exp(1)*2*x[2]
  J = matrix(c(a11,a12,a21,a22), ncol = 2, byrow = T)
}

```

```

J_inv = matlab::inv(J)
return(J_inv)
}

# set toleransi max yang diinginkan
tol_max = 0.000005

# set max iterasi yang diperbolehkan
iter_max = 20

# kita mulai iterasinya
iter = 0
while(norm_vec_inf(F_x_k(x0)) > tol_max && iter <= iter_max){
  xk_new = x0 - jax(x0) %*% F_x_k(x0)
  x0 = xk_new
  iter = iter + 1
  pesan = paste0("Iterasi ke-",iter,
                 " menghasilkan: x1 = ",x0[1],
                 " dan x2 = ",x0[2])
  print(pesan)
}

## [1] "Iterasi ke-1 menghasilkan: x1 = -0.410274431043867 dan x2 = 0.0281658782727883"
## [1] "Iterasi ke-2 menghasilkan: x1 = -0.376028580583334 dan x2 = 0.0544968132160582"
## [1] "Iterasi ke-3 menghasilkan: x1 = -0.373708769582047 dan x2 = 0.0562585222917004"
## [1] "Iterasi ke-4 menghasilkan: x1 = -0.373698216978757 dan x2 = 0.0562664892652393"

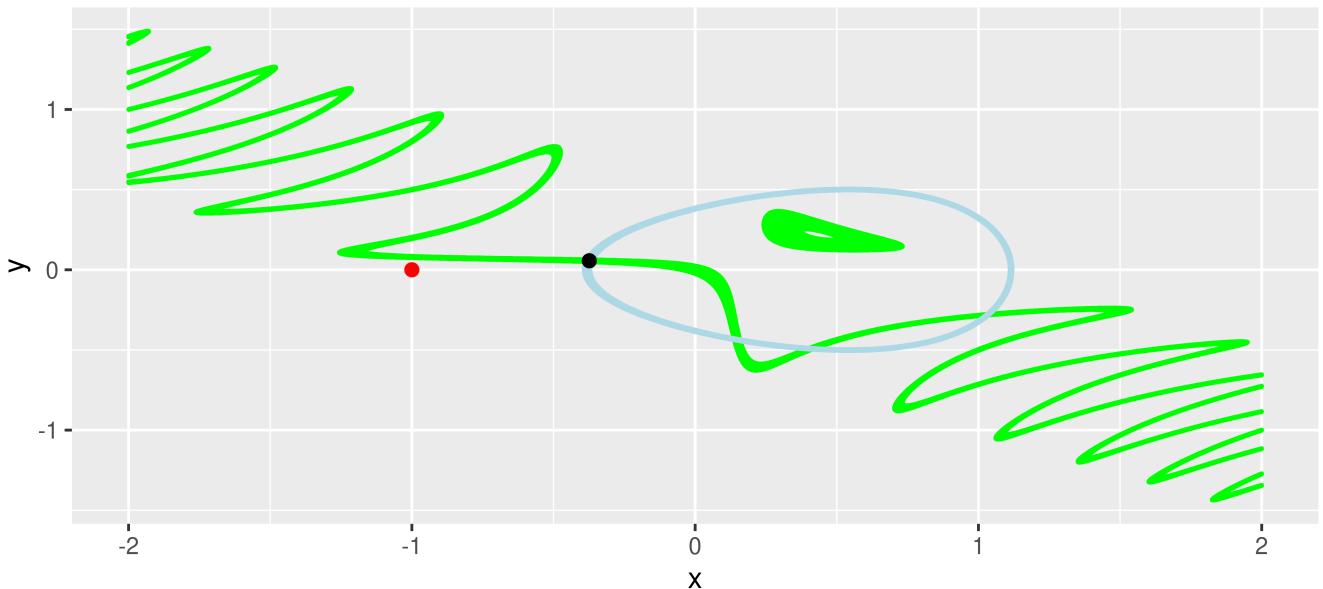
list("Solusi Final: " = x0,
     "Banyak iterasi: " = iter)

## $`Solusi Final: `
##      [,1]
## [1,] -0.37369822
## [2,]  0.05626649
##
## $`Banyak iterasi: `
## [1] 4

```

Didapatkan pada iterasi 4 solusi yang didapatkan adalah $(-0.37369822, 0.05626649)$. Berikut jika saya gambarkan dalam grafik:

Soal 3b

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$ 

Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 10: Solusi Pertama Soal 3b

Titik **merah** adalah *initial point* sedangkan titik **hitam** adalah solusi yang didapatkan.

Titik Kedua Mari kita coba selesaikan titik kedua.

```
# initial
x0 = c(0,-0.5)

# kita mulai iterasinya
iter = 0
while(norm_vec_inf(F_x_k(x0)) > tol_max && iter <= iter_max){
  xk_new = x0 - jax(x0) %*% F_x_k(x0)
  x0 = xk_new
  iter = iter + 1
  pesan = paste0("Iterasi ke-",iter,
                 " menghasilkan: x1 = ",x0[1],
                 " dan x2 = ",x0[2])
  print(pesan)
}

## [1] "Iterasi ke-1 menghasilkan: x1 = 0.119440359692262 dan x2 = -0.4349531657173"
## [1] "Iterasi ke-2 menghasilkan: x1 = 0.146357647348952 dan x2 = -0.435897060176411"
```

```

## [1] "Iterasi ke-3 menghasilkan: x1 = 0.147834271094862 dan x2 = -0.43617672848989"
## [1] "Iterasi ke-4 menghasilkan: x1 = 0.147839237183089 dan x2 = -0.43617762220478"

list("Solusi Final: " = x0,
     "Banyak iterasi: " = iter)

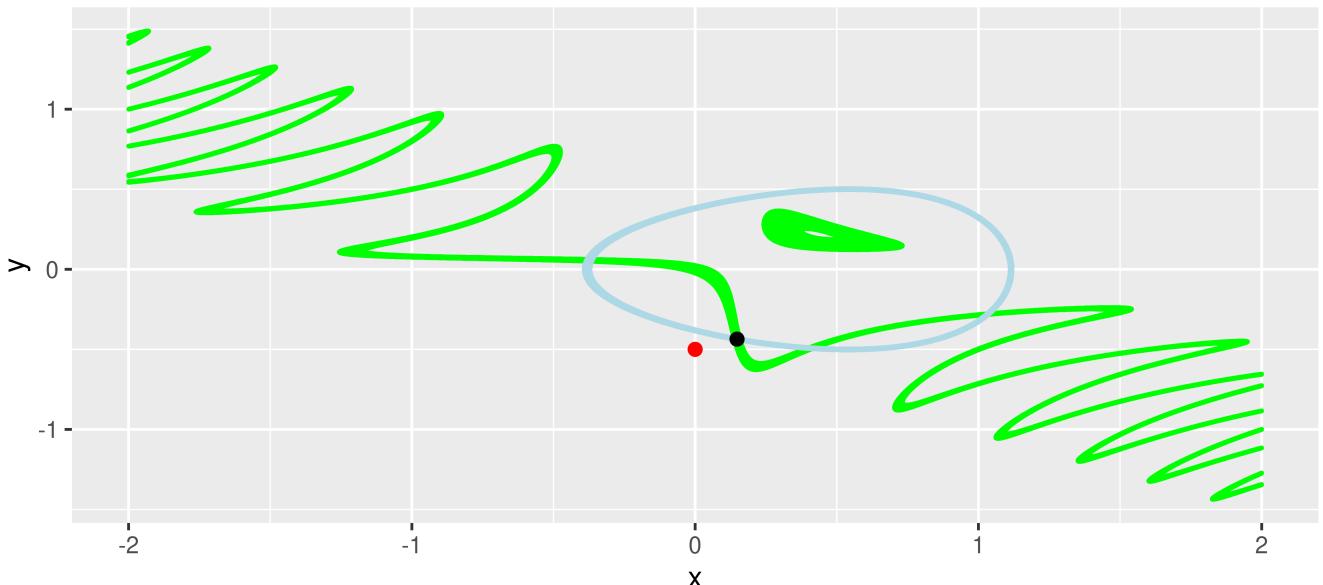
## $`Solusi Final: `
##      [,1]
## [1,] 0.1478392
## [2,] -0.4361776
##
## $`Banyak iterasi: `
## [1] 4

```

Didapatkan pada iterasi 4 solusi yang didapatkan adalah $(0.1478392, -0.4361776)$. Berikut jika saya gambarkan dalam grafik:

Soal 3b

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$



Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 11: Solusi Kedua Soal 3b

Titik **merah** adalah *initial point* sedangkan titik **hitam** adalah solusi yang didapatkan.

Titik Ketiga Mari kita coba selesaikan titik ketiga.

```
# initial
x0 = c(0.5,-0.5)

# kita mulai iterasinya
iter = 0
while(norm_vec_inf(F_x_k(x0)) > tol_max && iter <= iter_max){
  xk_new = x0 - jax(x0) %*% F_x_k(x0)
  x0 = xk_new
  iter = iter + 1
  pesan = paste0("Iterasi ke-",iter,
                 " menghasilkan: x1 = ",x0[1],
                 " dan x2 = ",x0[2])
  print(pesan)
}

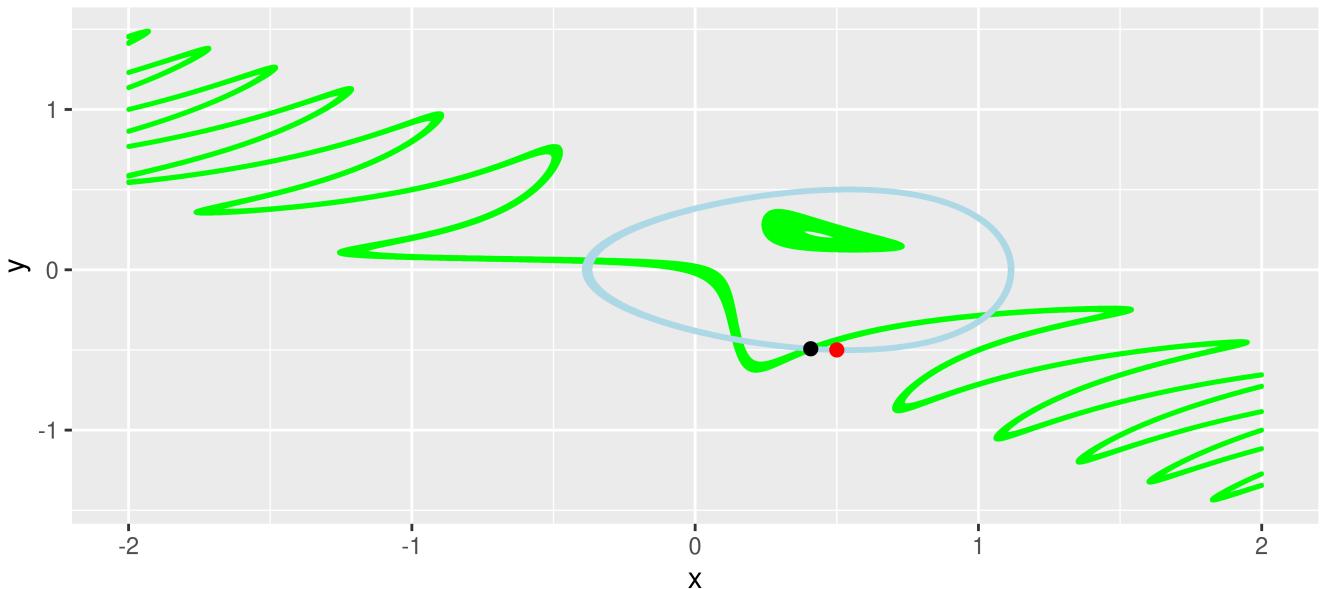
## [1] "Iterasi ke-1 menghasilkan: x1 = 0.41091731 dan x2 = -0.49645551"
## [1] "Iterasi ke-2 menghasilkan: x1 = 0.408173649698787 dan x2 = -0.492656244115729"
## [1] "Iterasi ke-3 menghasilkan: x1 = 0.408095686115349 dan x2 = -0.492629399967217"

list("Solusi Final: " = x0,
     "Banyak iterasi: " = iter)

## $`Solusi Final: `
##           [,1]
## [1,]  0.4080957
## [2,] -0.4926294
##
## $`Banyak iterasi: `
## [1] 3
```

Didapatkan pada iterasi 3 solusi yang didapatkan adalah (0.4080957, -0.4926294). Berikut jika saya gambarkan dalam grafik:

Soal 3b

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$ 

Digambar dengan R
20921004@mahasiswa.itb.ac.id

Figure 12: Solusi Ketiga Soal 3b

Titik Keempat Mari kita coba selesaikan titik keempat.

```
# initial
x0 = c(1.5, 0.5)

# kita mulai iterasinya
iter = 0
while(norm_vec_inf(F_x_k(x0)) > tol_max && iter <= iter_max){
  xk_new = x0 - jax(x0) %*% F_x_k(x0)
  x0 = xk_new
  iter = iter + 1
  pesan = paste0("Iterasi ke-", iter,
    " menghasilkan: x1 = ", x0[1],
    " dan x2 = ", x0[2])
  print(pesan)
}

## [1] "Iterasi ke-1 menghasilkan: x1 = 1.16672581311773 dan x2 = 0.496513067921227"
## [1] "Iterasi ke-2 menghasilkan: x1 = 0.546804449339741 dan x2 = 0.966165130417719"
## [1] "Iterasi ke-3 menghasilkan: x1 = 0.903632144848856 dan x2 = 0.611893363670833"
## [1] "Iterasi ke-4 menghasilkan: x1 = 2.22440765616215 dan x2 = -0.132168644738223"
```

```

## [1] "Iterasi ke-5 menghasilkan: x1 = 1.80011311956021 dan x2 = -0.194613562740547"
## [1] "Iterasi ke-6 menghasilkan: x1 = 1.44511827998176 dan x2 = -0.236005146150176"
## [1] "Iterasi ke-7 menghasilkan: x1 = 1.19680665079379 dan x2 = -0.248523389123324"
## [1] "Iterasi ke-8 menghasilkan: x1 = 1.06971570219983 dan x2 = -0.272040660183923"
## [1] "Iterasi ke-9 menghasilkan: x1 = 1.035489144902 dan x2 = -0.279372621377857"
## [1] "Iterasi ke-10 menghasilkan: x1 = 1.03308317009573 dan x2 = -0.279958834382974"
## [1] "Iterasi ke-11 menghasilkan: x1 = 1.03307147215505 dan x2 = -0.279961835824941"

list("Solusi Final: " = x0,
     "Banyak iterasi: " = iter)

## $`Solusi Final: `
##      [,1]
## [1,] 1.0330715
## [2,] -0.2799618
##
## $`Banyak iterasi: `
## [1] 11

```

Didapatkan pada iterasi 11 solusi yang didapatkan adalah $(1.0330715, -0.2799618)$. Berikut jika saya gambarkan dalam grafik:

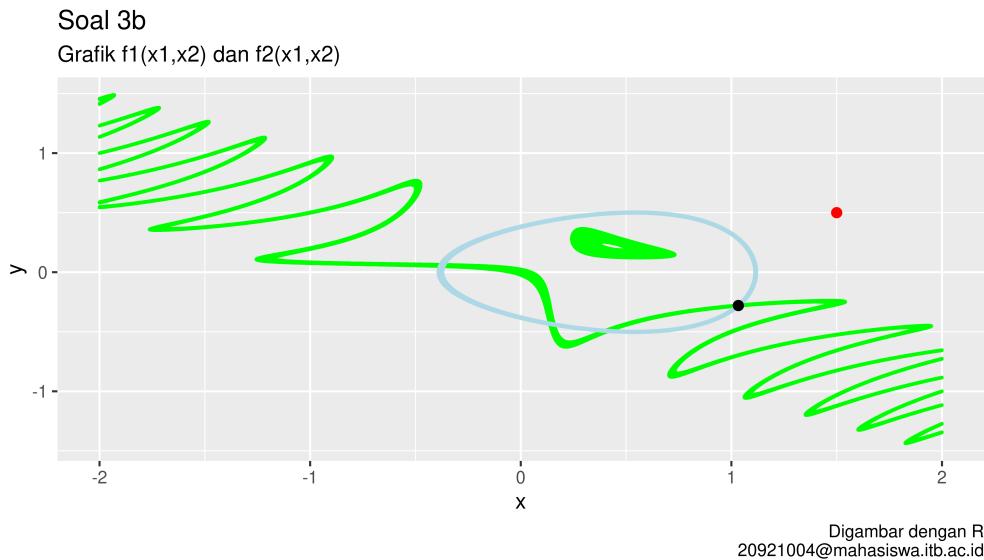


Figure 13: Solusi Keempat Soal 3b

Kesimpulan

Kita dapatkan 4 titik solusi SPNL berikut:

1. $(-0.37369822, 0.05626649)$
2. $(0.1478392, -0.4361776)$
3. $(0.4080957, -0.4926294)$
4. $(1.0330715, -0.2799618)$

Dalam bentuk grafik:

Soal 3b

Grafik $f_1(x_1, x_2)$ dan $f_2(x_1, x_2)$

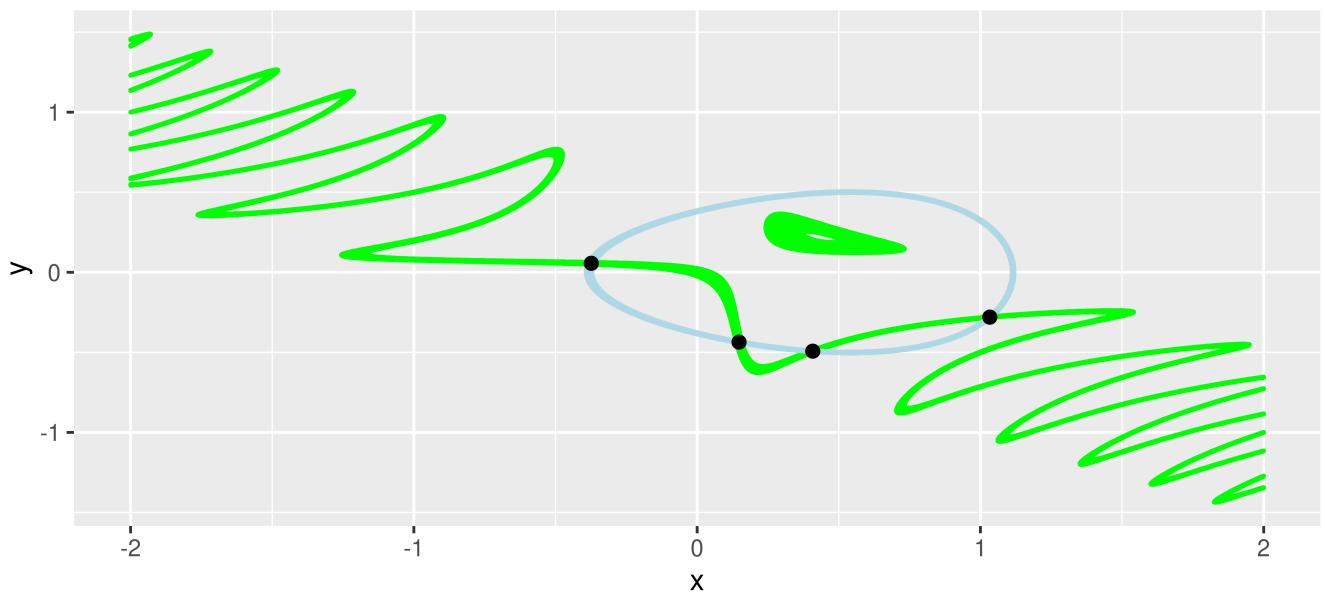


Figure 14: Semua Titik Solusi dari Soal 3b

5 TASK 4a

Soal

Aproksimasi solusi dari SPNL berikut ini:

$$3x_1 - \cos(x_2 x_3) - \frac{1}{2} = 0$$

$$4x_1^2 - 625x_2^2 + 2x_2 - 1 = 0$$

$$e^{-x_1 x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0$$

$$-1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1, -1 \leq x_3 \leq 1$$

Jawab

Grafik Sistem Persamaan Non Linear

Untuk membantu kita menjawab soal tersebut, pertama-tama kita perlu membuat grafik fungsi dari SPNL tersebut. Namun karena ada tiga peubah yang terlibat (x_1, x_2, x_3), saya akan buat penyederhanaan dengan membuat grafik sumbu x_1 dan sumbu $x_2 + x_3$.

Berikut adalah grafiknya:

```
# initial condition yang dibutuhkan untuk menggambar
# selang x1
x1_lower = -1; x1_upper = 1
# selang x2
x2_lower = -1; x2_upper = 1
# selang x3
x3_lower = -1; x3_upper = 1
delta = 0.005

# fungsi dari soal
f1 = function(x1,x2,x3){3 * x1 - cos(x2 * x3) - (1/2)}
f2 = function(x1,x2,x3){4 * x1^2 - 625 * x2^2 + 2 * x2 - 1}
f3 = function(x1,x2,x3){exp(-x1 * x2) + 20 * x3 + (10 * pi - 3)/(3)}

# proses generator data ke grafik
# menggunakan program yang dibuat di bagian 2.1.2
df1 = data_ke_grafik3d(x1_lower,x1_upper,x2_lower,x2_upper,x3_lower,x3_upper,delta,f1)
```

```
df2 = data_ke_grafik3d(x1_lower,x1_upper,x2_lower,x2_upper,x3_lower,x3_upper,delta,f2)
df3 = data_ke_grafik3d(x1_lower,x1_upper,x2_lower,x2_upper,x3_lower,x3_upper,delta,f3)

# proses menggambar plot
# membuat canvas
plot_soal_4a =
  ggplot() +
  # memplot f1
  geom_point(data = df1,aes(x1,x2_x3),size = .1,
             color = "steelblue") +
  # memplot f2
  geom_point(data = df2,aes(x1,x2_x3),size = .1,
             color = "darkgreen") +
  # memplot f3
  geom_point(data = df3,aes(x1,x2_x3),size = .1,
             color = "darkred") +
  labs(title = "Grafik Soal 4a",
       subtitle = "f1(x1,x2,x3), f2(x1,x2,x3), dan f3(x1,x2,x3)",
       x = "x1",
       y = "x2 + x3")

# menampilkan grafik
plot_soal_4a
```

Grafik Soal 4a

$f_1(x_1, x_2, x_3)$, $f_2(x_1, x_2, x_3)$, dan $f_3(x_1, x_2, x_3)$

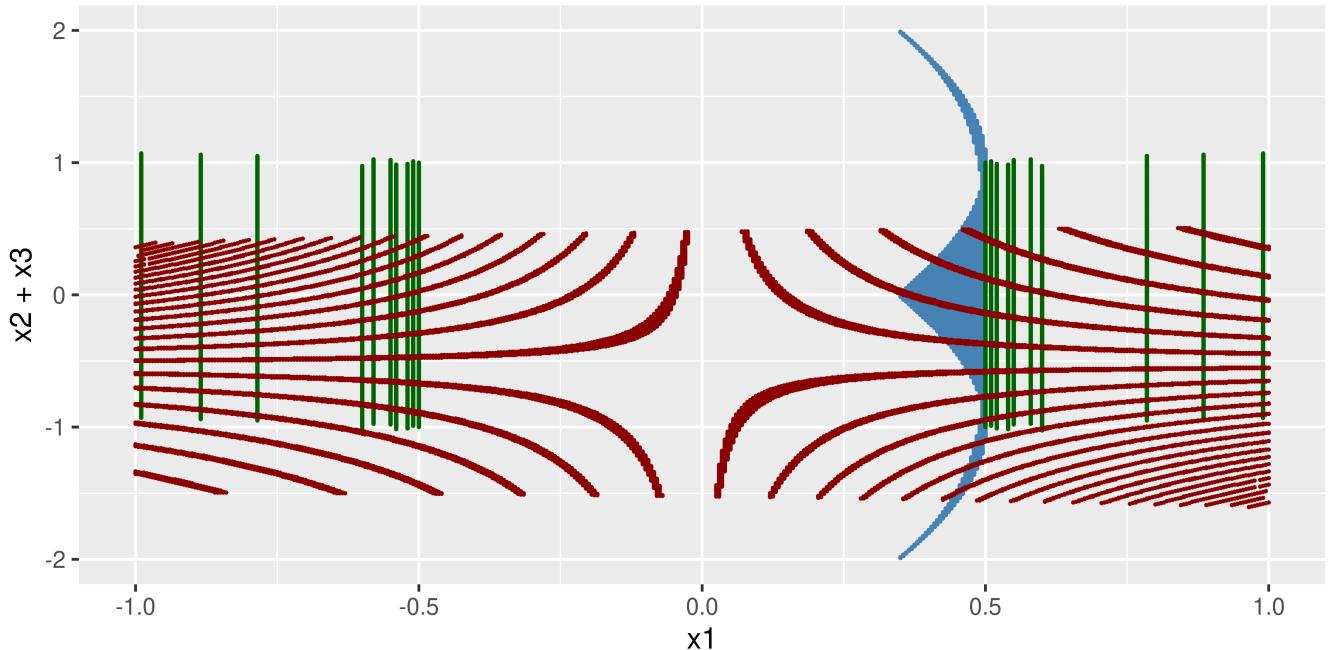


Figure 15: Grafik Soal 4a

Dari grafik di atas, kita tidak bisa **menebak** *initial points* sebagai bahan iterasi metode Broyden. Kita hanya bisa melihat bahwa titik solusi berada pada selang $x_1 > 0$.

Oleh karena itu, saya akan membuat beberapa titik *random* untuk menerka di mana saja solusi SPNL tersebut berada.

Penyelesaian dengan Metode Broyden

Metode Broyden tetap menggunakan matriks Jacobi sebagai langkah iterasinya. Oleh karena itu, kita akan bentuk matriks Jacobinya. Pertama-tama akan saya lakukan turunan parsial sebagai berikut:

```
# f1
f1 = "3 * x1 - Cos(x2 * x3) - (1/2)"
# turunan parsial f1 thd x1
f1 %>% y_fn("D(x1)") %>% yac_str()

## [1] "3"
```

```

# turunan parsial f1 thd x2
f1 %>% y_fn("D(x2)") %>% yac_str()

## [1] "x3*Sin(x2*x3)"

# turunan parsial f1 thd x3
f1 %>% y_fn("D(x3)") %>% yac_str()

## [1] "x2*Sin(x2*x3)"

# f2
f2 = "4 * x1^2 - 625 * x2^2 + 2 * x2 - 1"
# turunan parsial f2 thd x1
f2 %>% y_fn("D(x1)") %>% yac_str()

## [1] "8*x1"

# turunan parsial f2 thd x2
f2 %>% y_fn("D(x2)") %>% yac_str()

## [1] "2-1250*x2"

# turunan parsial f2 thd x3
f2 %>% y_fn("D(x3)") %>% yac_str()

## [1] "0"

# f3
f3 = "Exp(-x1 * x2) + 20 * x3 + (10 * pi - 3)/(3)"
# turunan parsial f3 thd x1
f3 %>% y_fn("D(x1)") %>% yac_str()

## [1] "-x2*Exp(-x1*x2)"

# turunan parsial f3 thd x2
f3 %>% y_fn("D(x2)") %>% yac_str()

## [1] "-x1*Exp(-x1*x2)"

```

```
# turunan parsial f3 thd x3
f3 %>% y_fn("D(x3)") %>% yac_str()
```

```
## [1] "20"
```

Berikut adalah matriks Jacobi yang sudah kita dapatkan:

$$J(x) = \begin{bmatrix} 3 & x_3 \sin(x_2 x_3) & x_2 \sin(x_2 x_3) \\ 8x_1 & 2 - 1250x_2 & 0 \\ -x_2 e^{-x_1 x_2} & -x_1 e^{-x_1 x_2} & 20 \end{bmatrix}$$

Sekarang kita akan buat program iterasi metode Broyden berdasarkan **algoritma 10.2**.

Percobaan Titik Random Pertama Sebagai percobaan, saya akan coba dekati dengan *initial point* (0, 1, 1). Berikut adalah programnya:

```
# initial
x0 = c(0,1,1)

# bikin fungsi F(x1,x2,x3)
F_x_k = function(x){
  f1 = 3 * x[1] - cos(x[2] * x[3]) - (1/2)
  f2 = 4 * x[1]^2 - 625 * x[2]^2 + 2 * x[2] - 1
  f3 = exp(-x[1] * x[2]) + 20 * x[3] + (10 * pi - 3)/(3)
  xk = c(f1,f2,f3)
  return(xk)
}

# set toleransi max yang diinginkan
tol_max = 0.00001

# set max iterasi yang diperbolehkan
iter_max = 30

# kita mulai metode Broyden-nya sesuai dengan algoritma 10.2
# step 1
v = F_x_k(x0)

# step 2
# bikin matriks jacobi
jax = function(x){
  a11 = 3
  a12 = x[3]*sin(x[2]*x[3])
```

```

a13 = x[2]*sin(x[2]*x[3])
a21 = 8*x[1]
a22 = 2-1250*x[2]
a23 = 0
a31 = -x[2]*exp(-x[1]*x[2])
a32 = -x[1]*exp(-x[1]*x[2])
a33 = 20
J = matrix(c(a11,a12,a13,a21,a22,a23,a31,a32,a33),ncol = 3,byrow = T)
J_inv = matlib:::inv(J)
return(J_inv)
}
A = jax(x0)

# step 3
s = -A %*% v
x = x0 + s
iter = 2

# step 4
while(norm_vec_inf(F_x_k(x)) > tol_max && iter <= iter_max){
  # step 5
  w = v
  v = F_x_k(x)
  y = v - w
  # step 6
  z = -A %*% y
  # step 7
  p = t(-s) %*% z
  p = as.numeric(p)
  # step 8
  ut = t(s) %*% A
  # step 9
  A = A + ((s+z)/p) %*% ut
  # step 10
  s = -A %*% v
  # step 11
  x = x + s
  iter = iter + 1
  # output
  pesan = paste(x,collapse = ",")
  pesan = paste0("Iterasi ke-",iter,": (",pesan,")")
  print(pesan)
}

```

```

## [1] "Iterasi ke- 3: (0.55870313441244,0.384526425554869,-0.521039363565993)"
## [1] "Iterasi ke- 4: (0.504105244142248,0.295993234283086,-0.515011910975756)"
## [1] "Iterasi ke- 5: (0.503684883990201,0.218476442512751,-0.517594755777091)"
## [1] "Iterasi ke- 6: (0.499701145159007,0.121503884227141,-0.520651709860875)"
## [1] "Iterasi ke- 7: (0.500066244614932,0.0832760831831942,-0.521479562313426)"
## [1] "Iterasi ke- 8: (0.499978491672467,0.0485256205883664,-0.522398202149168)"
## [1] "Iterasi ke- 9: (0.499995430456617,0.0318071979563444,-0.522802574900423)"
## [1] "Iterasi ke- 10: (0.499997277766899,0.019890100224307,-0.5231023073911)"
## [1] "Iterasi ke- 11: (0.499997915455446,0.0130427904879877,-0.523272971494865)"
## [1] "Iterasi ke- 12: (0.499998762820181,0.00873495136195977,-0.523380567873776)"
## [1] "Iterasi ke- 13: (0.499999094192144,0.00612470312295164,-0.523445786073152)"
## [1] "Iterasi ke- 14: (0.499999330131454,0.00459084262475086,-0.52348409519593)"
## [1] "Iterasi ke- 15: (0.499999453369484,0.00373992713209928,-0.523505352515376)"
## [1] "Iterasi ke- 16: (0.499999511018899,0.00334586391596685,-0.52351519591893)"
## [1] "Iterasi ke- 17: (0.499999529549029,0.00321949686308399,-0.523518352588722)"
## [1] "Iterasi ke- 18: (0.499999532392799,0.00319995093587444,-0.523518840846399)"

list("Solusi Final: " = x,
     "Banyak iterasi: " = iter)

## $`Solusi Final: `
## [1]
## [1,] 0.499999532
## [2,] 0.003199951
## [3,] -0.523518841
##
## $`Banyak iterasi: `
## [1] 18

```

Ternyata didapatkan pada iterasi ke 18 solusinya adalah $(0.499999532, 0.003199951, -0.523518841)$.

Sebelum saya membuat grafiknya kembali, saya akan coba untuk beberapa *initial points* lainnya, seperti:

Percobaan Titik *Random* Kedua Sebagai percobaan kedua, saya akan coba dekati dengan *initial point* $(0, 0.5, 0.5)$. Berikut adalah hasil dari programnya:

```
## [1] "Iterasi ke- 3: (0.500138512071758,0.187921073073109,-0.51832971418813)"
## [1] "Iterasi ke- 4: (0.499257499752291,0.129695780875714,-0.519728416260833)"
## [1] "Iterasi ke- 5: (0.49996530537344,0.0784032056089546,-0.521615855994639)"
## [1] "Iterasi ke- 6: (0.499991899907985,0.0493673906222215,-0.52237179942345)"
## [1] "Iterasi ke- 7: (0.499994891019868,0.0310675447369457,-0.522822980978204)"
## [1] "Iterasi ke- 8: (0.499997019553619,0.0198536472932781,-0.523102871213623)"
## [1] "Iterasi ke- 9: (0.499998090082306,0.0129234399772708,-0.523275961245467)"
## [1] "Iterasi ke- 10: (0.499998726604025,0.00867442048054626,-0.523382090632463)"
## [1] "Iterasi ke- 11: (0.499999108365255,0.00609287730679065,-0.523446576063504)"
## [1] "Iterasi ke- 12: (0.49999933209539,0.00456869548387359,-0.523484649738696)"
## [1] "Iterasi ke- 13: (0.49999945485577,0.00373016860324986,-0.523505596036748)"
## [1] "Iterasi ke- 14: (0.499999511671606,0.0033416978790166,-0.523515300012303)"
## [1] "Iterasi ke- 15: (0.499999529664754,0.00321862453088159,-0.523518374379973)"
## [1] "Iterasi ke- 16: (0.499999532402615,0.00319989474990266,-0.523518842249339)"

## $`Solusi Final: `
##           [,1]
## [1,] 0.499999532
## [2,] 0.003199895
## [3,] -0.523518842
##
## $`Banyak iterasi: `
## [1] 16
```

Ternyata masih didapatkan solusi yang sama dengan percobaan pertama.

Percobaan Titik Random Ketiga Sebagai percobaan ketiga, saya akan coba dekati dengan *initial point* $(0, -1, -1)$. Berikut adalah hasil dari programnya:

```
## [1] "Iterasi ke- 3: (0.525172323422725,-0.371284188088303,-0.536803347604719)"
## [1] "Iterasi ke- 4: (0.498949853459116,-0.238331922148286,-0.530269076596453)"
## [1] "Iterasi ke- 5: (0.499857445094232,-0.156151181920434,-0.528030658033799)"
## [1] "Iterasi ke- 6: (0.500109433075832,-0.0928714124634234,-0.525895701268534)"
## [1] "Iterasi ke- 7: (0.499946636075515,-0.0573674761377058,-0.525019866577174)"
## [1] "Iterasi ke- 8: (0.500004445874358,-0.0351518282002837,-0.524484552473061)"
## [1] "Iterasi ke- 9: (0.5000075069485,-0.0208655182932894,-0.524118040342847)"
## [1] "Iterasi ke- 10: (0.500000330557309,-0.0123602756232268,-0.523907137922644)"
## [1] "Iterasi ke- 11: (0.500001148054446,-0.00713425622987882,-0.523777252230746)"
## [1] "Iterasi ke- 12: (0.500000708134359,-0.00386052703301208,-0.523695154015544)"
## [1] "Iterasi ke- 13: (0.500000234958534,-0.00193905645616293,-0.523647204348698)"
## [1] "Iterasi ke- 14: (0.500000125893857,-0.000838582487974159,-0.523619731679316)"
## [1] "Iterasi ke- 15: (0.500000042998354,-0.000269477134702759,-0.52360550554963)"
## [1] "Iterasi ke- 16: (0.500000006585768,-0.0000523594893545703,-0.523600083341387)"
## [1] "Iterasi ke- 17: (0.500000000634258,-0.00000424234655551592,-0.523598881682726)"

## $`Solusi Final: `
## [1]
## [1,] 0.500000000634
## [2,] -0.000004242347
## [3,] -0.523598881683
##
## $`Banyak iterasi: `
## [1] 17
```

Ternyata masih didapatkan solusi yang sama dengan percobaan pertama.

Percobaan Titik *Random* Keempat Sebagai percobaan keempat, saya akan coba dekati dengan *initial point* $(0, 0, 0)$. Berikut adalah hasil dari programnya:

```
## [1] "Iterasi ke- 3: (0.500231223602292,-1.08029906948258,-0.523823935607034)"
## [1] "Iterasi ke- 4: (0.49958675682186,0.918911559678893,-0.500284220444249)"
## [1] "Iterasi ke- 5: (0.509706765482204,5.97137208852088,-0.368570359399117)"
## [1] "Iterasi ke- 6: (0.477463957247958,0.797293949659266,-0.506029803455345)"
## [1] "Iterasi ke- 7: (0.482309588904067,0.703550495523158,-0.508307701605321)"
## [1] "Iterasi ke- 8: (0.49882331548596,0.374830579914373,-0.514288422831914)"
## [1] "Iterasi ke- 9: (0.499649238083715,0.247629638250854,-0.517370053424252)"
## [1] "Iterasi ke- 10: (0.499909324852429,0.150146441164084,-0.519856381575055)"
## [1] "Iterasi ke- 11: (0.499969019540054,0.0943483021226307,-0.521245222058785)"
## [1] "Iterasi ke- 12: (0.49998760941274,0.0587462719888686,-0.52213208836409)"
## [1] "Iterasi ke- 13: (0.499993694512109,0.0369923470236705,-0.52267491076115)"
## [1] "Iterasi ke- 14: (0.499996358998789,0.023489900693098,-0.523012070851276)"
## [1] "Iterasi ke- 15: (0.49999773410785,0.0151710317778526,-0.523219815519864)"
## [1] "Iterasi ke- 16: (0.499998519990226,0.0100498991838438,-0.523347732086681)"
## [1] "Iterasi ke- 17: (0.499998985589558,0.00692373461730263,-0.523425821436866)"
## [1] "Iterasi ke- 18: (0.499999261207118,0.00505164559694329,-0.523472585681624)"
## [1] "Iterasi ke- 19: (0.499999417498798,0.00398533512646891,-0.523499221987533)"
## [1] "Iterasi ke- 20: (0.499999496038068,0.00344858664586372,-0.523512629933626)"
## [1] "Iterasi ke- 21: (0.499999525750802,0.00324539456698509,-0.523517705664997)"
## [1] "Iterasi ke- 22: (0.499999532040365,0.00320237271098719,-0.523518780349916)"

## $`Solusi Final: `
##           [,1]
## [1,]  0.499999532
## [2,]  0.003202373
## [3,] -0.523518780
##
## $`Banyak iterasi: `
## [1] 22
```

Ternyata masih didapatkan solusi yang sama dengan percobaan pertama.

Percobaan Titik Random Kelima Sebagai percobaan kelima, saya akan coba dekati dengan *initial point* (0.22,-0.94,0.59). Berikut adalah hasil dari programnya:

```
## [1] "Iterasi ke- 3: (0.503542859851317,-0.356808538988173,-0.530988817500403)"
## [1] "Iterasi ke- 4: (0.496639550508741,-0.270488046630992,-0.532172680266234)"
## [1] "Iterasi ke- 5: (0.499275748133898,-0.162821448754116,-0.52805251118321)"
## [1] "Iterasi ke- 6: (0.500091391213763,-0.0999965699643949,-0.526071885811444)"
## [1] "Iterasi ke- 7: (0.500048321695095,-0.0600907874758882,-0.525073282947385)"
## [1] "Iterasi ke- 8: (0.500008144179464,-0.036742818240559,-0.524515300788515)"
## [1] "Iterasi ke- 9: (0.500002111858565,-0.0221753483364851,-0.524153664239053)"
## [1] "Iterasi ke- 10: (0.500001747085126,-0.013146165899483,-0.523927335818555)"
## [1] "Iterasi ke- 11: (0.500001146449676,-0.00756260055278121,-0.523787665188065)"
## [1] "Iterasi ke- 12: (0.500000622430966,-0.00415480631947994,-0.523702552853562)"
## [1] "Iterasi ke- 13: (0.500000308548503,-0.00210650441100514,-0.523651395927789)"
## [1] "Iterasi ke- 14: (0.50000013489309,-0.000925552692354546,-0.52362189627031)"
## [1] "Iterasi ke- 15: (0.500000045702438,-0.000312970913391868,-0.523606593652092)"
## [1] "Iterasi ke- 16: (0.500000009548568,-0.0000652655526320863,-0.52360040592515)"
## [1] "Iterasi ke- 17: (0.500000000835757,-0.00000570688563034466,-0.523598918155043)"
## [1] "Iterasi ke- 18: (0.50000000016693,-0.000000113759125230608,-0.523598778439951)"

## $`Solusi Final: ` 
## [1] [,1]
## [1,] 0.5000000000167
## [2,] -0.0000001137591
## [3,] -0.5235987784400
##
## $`Banyak iterasi: ` 
## [1] 18
```

Ternyata masih didapatkan solusi yang sama dengan percobaan pertama.

Kesimpulan

Hanya ada satu solusi dari SPNL tersebut, yakni: (0.5, 0, -0.5).

6 **TASK 4b**

Soal

Aproksimasi solusi dari SPNL berikut ini:

$$x_1^2 + x_2 - 37 = 0$$

$$x_1 - x_2^2 - 5 = 0$$

$$x_1 + x_2 + x_3 - 3 = 0$$

$$-4 \leq x_1 \leq 8$$

$$-2 \leq x_2 \leq 2$$

$$-6 \leq x_3 \leq 0$$

Jawab

Grafik Sistem Persamaan Non Linear

Sama dengan soal sebelumnya, kita akan menggunakan pendekatan grafik sumbu x_1 dan sumbu $x_2 + x_3$. Berikut adalah grafiknya:

```
# initial condition yang dibutuhkan untuk menggambar
# selang x1
x1_lower = -4; x1_upper = 8
# selang x2
x2_lower = -2; x2_upper = 2
# selang x3
x3_lower = -6; x3_upper = 0
delta = 0.01

# fungsi dari soal
f1 = function(x1,x2,x3){x1^2 + x2 - 37}
f2 = function(x1,x2,x3){x1 - x2^2 - 5}
f3 = function(x1,x2,x3){x1 + x2 + x3 - 3}
```

```
# proses generator data ke grafik
# menggunakan program yang dibuat di bagian 2.1.2
df1 = data_ke_grafik(x1_lower,x1_upper,delta,f1)
df2 = data_ke_grafik(x1_lower,x1_upper,delta,f2)
df3 = data_ke_grafik3d(x1_lower,x1_upper,x2_lower,x2_upper,x3_lower,x3_upper,delta,f3)

# proses menggambar plot
# membuat canvas
plot_soal_4b =
  ggplot() +
  # memplot f1
  geom_point(data = df1,aes(x,y),size = .1,
             color = "steelblue") +
  # memplot f2
  geom_point(data = df2,aes(x,y),size = .1,
             color = "darkgreen") +
  # memplot f3
  geom_point(data = df3,aes(x1,x2_x3),size = .1,
             color = "darkred") +
  labs(title = "Grafik Soal 4b",
       subtitle = "f1(x1,x2,x3), f2(x1,x2,x3), dan f3(x1,x2,x3)",
       x = "x1",
       y = "x2 + x3")

# menampilkan grafik
plot_soal_4b
```

Grafik Soal 4b

$f_1(x_1, x_2, x_3)$, $f_2(x_1, x_2, x_3)$, dan $f_3(x_1, x_2, x_3)$

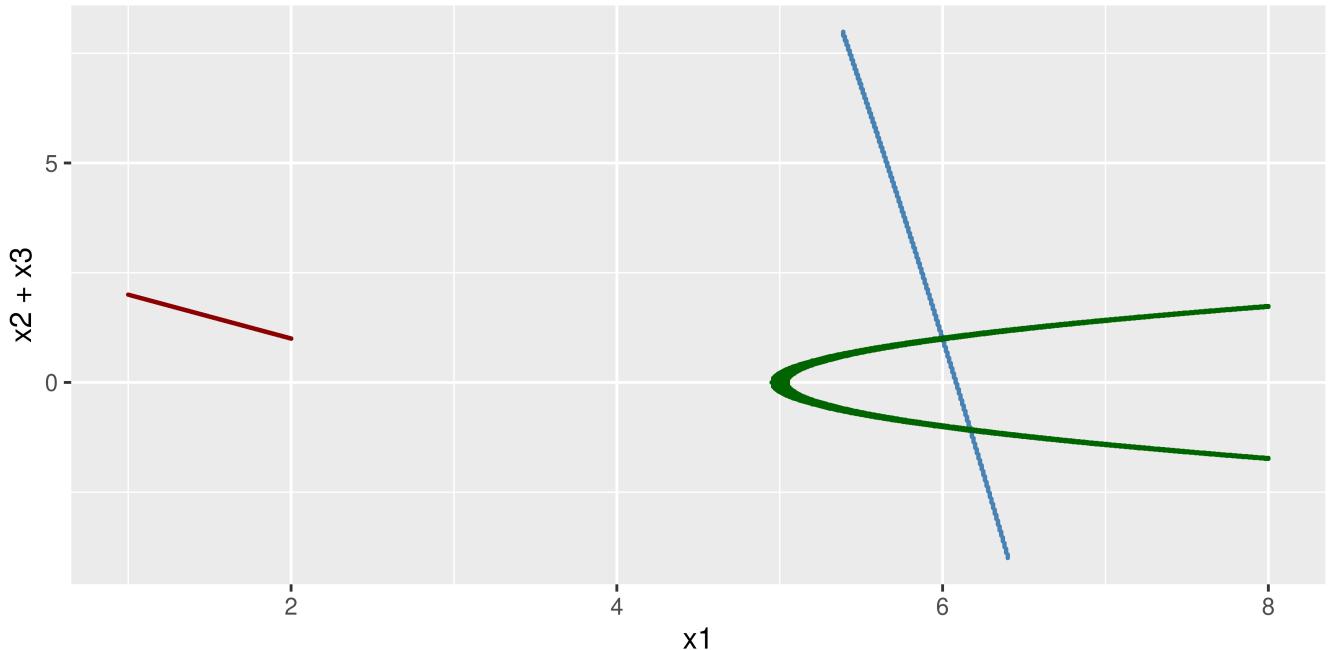


Figure 16: Grafik Soal 4b

Ternyata grafik di atas tidak konklusif. Oleh karena itu, kita akan lakukan metode Broyden dengan beberapa titik yang *di-random* di sekitar domain x_1, x_2, x_3 .

6.0.1 Penyelesaian dengan Metode Broyden

Penentuan matriks Jacobi dilakukan sama dengan soal-soal sebelumnya, yakni:

```
# f1
f1 = "x1^2 + x2 - 37"
# turunan parsial f1 thd x1
f1 %>% y_fn("D(x1)") %>% yac_str()

## [1] "2*x1"

# turunan parsial f1 thd x2
f1 %>% y_fn("D(x2)") %>% yac_str()

## [1] "1"
```

```

# turunan parsial f1 thd x3
f1 %>% y_fn("D(x3)") %>% yac_str()

## [1] "0"

# f2
f2 = "x1 - x2^2 - 5"
# turunan parsial f2 thd x1
f2 %>% y_fn("D(x1)") %>% yac_str()

## [1] "1"

# turunan parsial f2 thd x2
f2 %>% y_fn("D(x2)") %>% yac_str()

## [1] "-2*x2"

# turunan parsial f2 thd x3
f2 %>% y_fn("D(x3)") %>% yac_str()

## [1] "0"

# f3
f3 = "x1 + x2 + x3 - 3"
# turunan parsial f3 thd x1
f3 %>% y_fn("D(x1)") %>% yac_str()

## [1] "1"

# turunan parsial f3 thd x2
f3 %>% y_fn("D(x2)") %>% yac_str()

## [1] "1"

# turunan parsial f3 thd x3
f3 %>% y_fn("D(x3)") %>% yac_str()

## [1] "1"

```

Berikut adalah matriks Jacobi yang sudah kita dapatkan:

$$J(x) = \begin{bmatrix} 2x_1 & 1 & 0 \\ 1 & -2x_2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sekarang kita akan buat program iterasi metode Broyden berdasarkan **algoritma 10.2**.

Percobaan Titik Random Pertama Sebagai percobaan, saya akan coba dekati dengan *initial point* (3.8,-0.6,-1.1). Berikut adalah programnya:

```
# bikin fungsi F(x1,x2,x3)
F_x_k = function(x){
  f1 = x[1]^2 + x[2] - 37
  f2 = x[1] - x[2]^2 - 5
  f3 = x[1] + x[2] + x[3] - 3
  xk = c(f1,f2,f3)
  return(xk)
}

# set toleransi max yang diinginkan
tol_max = 0.00001

# set max iterasi yang diperbolehkan
iter_max = 70

# kita mulai metode Broyden-nya sesuai dengan algoritma 10.2
# step 1
v = F_x_k(x0)

# step 2
# bikin matriks jacobi
jax = function(x){
  a11 = 2*x[1]
  a12 = 1
  a13 = 0
  a21 = 1
  a22 = -2*x[2]
  a23 = 0
  a31 = 1
  a32 = 1
  a33 = 1
  J = matrix(c(a11,a12,a13,a21,a22,a23,a31,a32,a33),ncol = 3,byrow = T)
  J_inv = matlib::inv(J)
  return(J_inv)
}
A = jax(x0)

# step 3
s = -A %*% v
x = x0 + s
iter = 2
```

```

# step 4
while(norm_vec_inf(F_x_k(x)) > tol_max && iter <= iter_max){
  # step 5
  w = v
  v = F_x_k(x)
  y = v - w
  # step 6
  z = -A %*% y
  # step 7
  p = t(-s) %*% z
  p = as.numeric(p)
  # step 8
  ut = t(s) %*% A
  # step 9
  A = A + ((s+z)/p) %*% ut
  # step 10
  s = -A %*% v
  # step 11
  x = x + s
  iter = iter + 1
  # output
  pesan = paste(x,collapse = ",")
  pesan = paste0("Iterasi ke-",iter,": (",pesan,")")
  print(pesan)
}

```

```

## [1] "Iterasi ke- 3: (5.9616687995722,-0.132143369148719,-2.82952543042348)"
## [1] "Iterasi ke- 4: (6.09822761042486,-0.699818412527538,-2.39840919789732)"
## [1] "Iterasi ke- 5: (6.21735954518513,-1.72868638622326,-1.48867315896187)"
## [1] "Iterasi ke- 6: (6.16495459877093,-0.974063185820892,-2.19089141295004)"
## [1] "Iterasi ke- 7: (6.16873759356432,-1.05553010555274,-2.11320748801158)"
## [1] "Iterasi ke- 8: (6.17127950361903,-1.08340520712049,-2.08787429649855)"
## [1] "Iterasi ke- 9: (6.17106609436509,-1.08216169885684,-2.08890439550824)"
## [1] "Iterasi ke- 10: (6.1710753294772,-1.0821605446551,-2.0889147848221)"
## [1] "Iterasi ke- 11: (6.17107466173692,-1.08216193401242,-2.0889127277245)"

```

```

if(iter <= iter_max){
  list("Titik initial: " = x0,
       "Solusi Final: " = x,
       "Banyak iterasi: " = iter)
} else if(iter > iter_max){
  list("Titik initial: " = x0,
       "Solusi Final: " = "Tidak Konvergen atau melebihi batas iterasi")
}

```

```
## $`Titik initial: `  
## [1] 3.8 -0.6 -1.1  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6.171075  
## [2,] -1.082162  
## [3,] -2.088913  
##  
## $`Banyak iterasi: `  
## [1] 11
```

Percobaan Titik *Random* Kedua Kita bisa mengulang prosedur di atas dengan cara men-generate *random number* di domain x_1, x_2, x_3 sebagai berikut:

```
## $`Titik initial: `  
## [1] 4.6 0.2 -5.2  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6.0000005  
## [2,] 0.9999978  
## [3,] -3.9999982  
##  
## $`Banyak iterasi: `  
## [1] 10
```

Percobaan Titik *Random* Ketiga

```
## $`Titik initial: `  
## [1] 3.5 2.0 -5.5  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6  
## [2,] 1  
## [3,] -4  
##  
## $`Banyak iterasi: `  
## [1] 8
```

Percobaan Titik *Random* Keempat

```
## $`Titik initial: `  
## [1] 3.1 1.3 -0.3  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6  
## [2,] 1  
## [3,] -4  
##  
## $`Banyak iterasi: `  
## [1] 9
```

Percobaan Titik *Random* Kelima

```
## $`Titik initial: `  
## [1] 5.1 -0.9 -4.9  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6.171075  
## [2,] -1.082162  
## [3,] -2.088913  
##  
## $`Banyak iterasi: `  
## [1] 7
```

Percobaan Titik *Random* Keenam

```
## $`Titik initial: `  
## [1] 4.6 -1.7 -1.4  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6.171075  
## [2,] -1.082162  
## [3,] -2.088913  
##  
## $`Banyak iterasi: `  
## [1] 7
```

Percobaan Titik *Random* Ketujuh

```
## $`Titik initial: `  
## [1] 4.0 0.5 -3.4  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6.0000000  
## [2,] 0.9999997  
## [3,] -3.9999998  
##  
## $`Banyak iterasi: `  
## [1] 8
```

Percobaan Titik *Random* Kedelapan

```
## $`Titik initial: `  
## [1] 7.2 1.3 -2.6  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6  
## [2,] 1  
## [3,] -4  
##  
## $`Banyak iterasi: `  
## [1] 7
```

Percobaan Titik *Random* Kesembilan

```
## $`Titik initial: `  
## [1] 7.9 -0.7 -2.2  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6.171075  
## [2,] -1.082162  
## [3,] -2.088913  
##  
## $`Banyak iterasi: `  
## [1] 8
```

Percobaan Titik *Random* Kesepuluh

```
## $`Titik initial: `  
## [1] 5.0 0.6 -0.4  
##  
## $`Solusi Final: `  
## [,1]  
## [1,] 6  
## [2,] 1  
## [3,] -4  
##  
## $`Banyak iterasi: `  
## [1] 8
```

Kesimpulan

Dari berbagai titik *random* di atas, hanya ada dua titik solusi pada SPNL ini, yakni:

1. $(6, 1, -4)$
 2. $(6, -1, -2)$
-