

SPIRAL OPTIMIZATION ALGORITHM

Tugas Kuliah
SK5001 Analisis Numerik Lanjut

Mohammad Rizka Fadhli
NIM: 20921004

17 October 2021

PENDAHULUAN

Bahasa yang Digunakan

Untuk membuat program *spiral optimization algorithm*, saya menggunakan bahasa **R** yang bisa dieksekusi pada versi minimal 3.5.3.

Spiral Optimization Algorithm

Spiral Optimization Algorithm adalah salah satu metode *meta heuristic* yang digunakan untuk mencari minimum global dari suatu sistem persamaan.

Algoritmanya mudah dipahami dan intuitif tanpa harus memiliki latar keilmuan tertentu. Proses kerjanya adalah dengan melakukan *random number generating* pada suatu selang dan melakukan rotasi sekaligus kontraksi dengan titik paling minimum pada setiap iterasi sebagai pusatnya.

Berikut adalah algoritmanya:

```
INPUT
m >= 2 # jumlah titik
theta # sudut rotasi (0 <= theta <= 2pi)
r      # konstraksi
k_max # iterasi maksimum

PROCESS
1 generate m buah titik secara acak
    x_i

2 initial condition
    k = 0 # untuk keperluan iterasi

3 cari x_* yang memenuhi
    min(f(x_*))

4 lakukan rotasi dan konstraksi semua x_i
    x_* sebagai pusat rotasi
    k = k + 1

5 ulangi proses 3 dan 4
```

```

6 hentikan proses saat k = k_max
    output x_*

```

Berdasarkan algoritma di atas, salah satu proses yang penting adalah melakukan **rotasi** dan **konstraksi** terhadap semua titik yang telah *di-generate*.

Agar memudahkan, saya akan memberikan ilustrasi geometri beserta operasi matriks aljabar terkait kedua hal tersebut.

Membuat Program

Untuk menyelesaikan tugas soal yang diberikan, pertama-tama saya harus membuat program *spiral optimization algorithm*. Untuk membuatnya, saya akan melakukannya perlahan-lahan dengan bantuan ilustrasi geometri. Berikut adalah langkah-langkah yang ditempuh:

1. **Pertama** saya akan membuat program yang bisa merotasi suatu titik berdasarkan suatu θ tertentu.
2. **Kedua** saya akan memodifikasi program tersebut untuk melakukan rotasi sekaligus konstraksi dengan rasio r tertentu.
3. **Ketiga** saya akan memodifikasi program tersebut untuk melakukan rotasi sekaligus konstraksi dengan **titik pusat rotasi tertentu**.

Dari program yang terakhir, akan saya pakai untuk **membangun program spiral optimization algorithm** yang sebenarnya.

Langkah dan Ilustrasi Geometri

Operasi Matriks Rotasi

Misalkan saya memiliki titik $x \in \mathbb{R}^2$. Untuk melakukan rotasi sebesar θ , saya bisa menggunakan suatu matriks $A_{2 \times 2}$ berisi fungsi-fungsi trigonometri sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

Berdasarkan operasi matriks di atas, saya membuat **program** di **R** dengan beberapa modifikasi. Sebagai contoh, saya akan membuat program yang bertujuan untuk melakukan rotasi suatu titik $x \in \mathbb{R}$ sebanyak n kali:

```

# mendefinisikan program
rotasi_kan = function(x0,rot){
  # menghitung theta
  theta = 2*pi/rot

  # definisi matriks rotasi
  A = matrix(c(cos(theta),-sin(theta),
              sin(theta),cos(theta)),
             ncol = 2,byrow = T)

  # membuat template
  temp = vector("list")
  temp[[1]] = x0

  # proses rotasi
  for(i in 2:rot){
    xk = A %*% x0
    temp[[i]] = xk
    x0 = xk
  }
}

```

```

}

# membuat template data frame
final = data.frame(x = rep(NA,rot),
                    y = rep(NA,rot))

# gabung data dari list
for(i in 1:rot){
  tempura = temp[[i]]
  final$x[i] = tempura[1]
  final$y[i] = tempura[2]
}

# membuat plot
plot =
  ggplot() +
  geom_point(aes(x,y),data = final) +
  geom_point(aes(x[1],y[1]),
             data = final,
             color = "red") +
  coord_equal() +
  labs(title = "titik merah adalah titik initial")

# enrich dengan garis panah
panah = data.frame(
  x_start = final$x[1:(rot-1)],
  x_end = final$x[2:rot],
  y_start = final$y[1:(rot-1)],
  y_end = final$y[2:rot]
)
# menambahkan garis panah ke plot
plot =
  plot +
  geom_segment(aes(x = x_start,
                   xend = x_end,
                   y = y_start,
                   yend = y_end),
               data = panah,
               arrow = arrow(length = unit(.3,"cm"))
  )

# menyiapkan output
list("Grafik rotasi" = plot,
     "Titik-titik rotasi" = final)
}

```

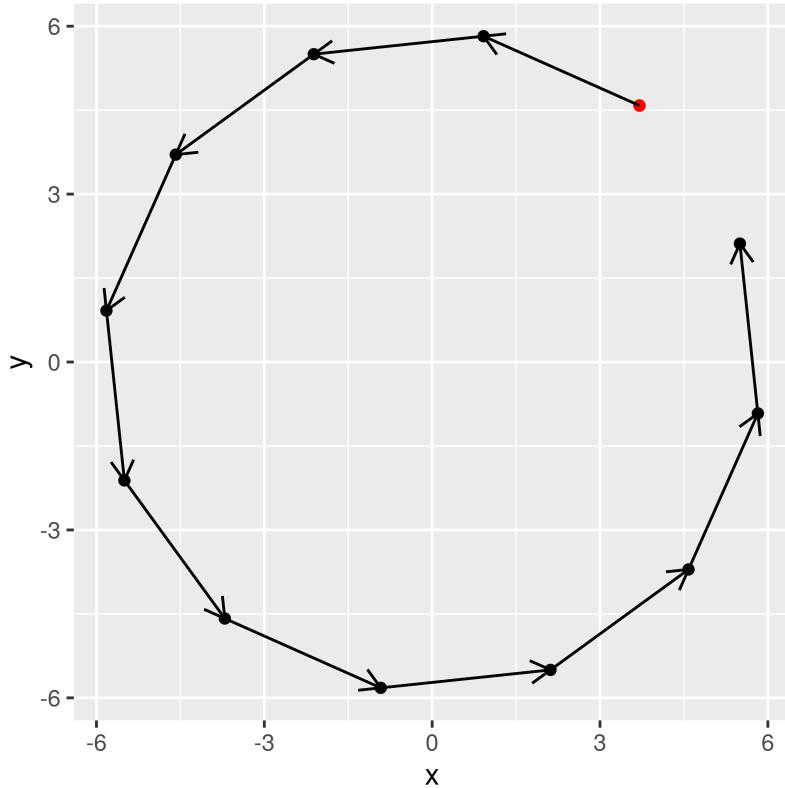
Berikut adalah uji coba dengan titik sembarang berikut ini:

```
# uji coba
rot = 12 # berapa banyak rotasi
x0 = rand_titik(0,10) # generate random titik

rotasi_kan(x0,rot)

## $`Grafik rotasi`
```

titik merah adalah titik initial



```
##
## $`Titik-titik rotasi`  

##          x      y  

## 1  3.705807 4.582290  

## 2  0.918178 5.821282  

## 3 -2.115476 5.500467  

## 4 -4.582290 3.705807  

## 5 -5.821282 0.918178  

## 6 -5.500467 -2.115476  

## 7 -3.705807 -4.582290  

## 8 -0.918178 -5.821282  

## 9  2.115476 -5.500467  

## 10 4.582290 -3.705807  

## 11 5.821282 -0.918178  

## 12 5.500467 2.115476
```

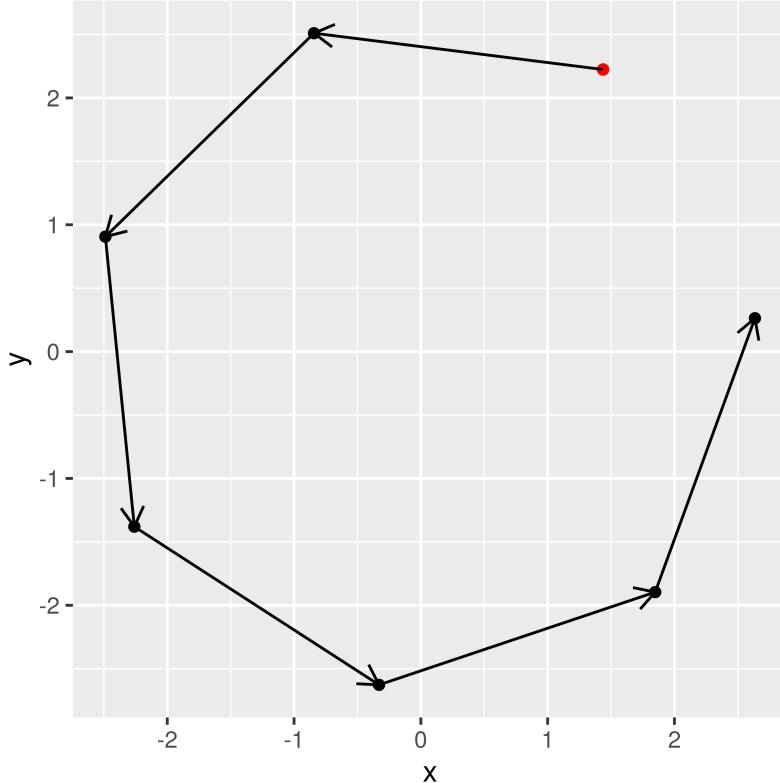
Uji coba kembali dengan titik sembarang lainnya berikut ini:

```
# uji coba
rot = 7 # berapa banyak rotasi
x0 = rand_titik(0,10) # generate random titik

rotasi_kan(x0,rot)

## $`Grafik rotasi`
```

titik merah adalah titik initial



```
##
## $`Titik-titik rotasi`  
##      x      y  
## 1  1.437130  2.223893  
## 2 -0.842674  2.510168  
## 3 -2.487927  0.906235  
## 4 -2.259720 -1.380111  
## 5 -0.329898 -2.627205  
## 6  1.848344 -1.895961  
## 7  2.634745  0.262981
```

Operasi Matriks Rotasi dan Kontraksi

Jika pada sebelumnya saya **hanya melakukan rotasi**, kali ini saya akan memodifikasi operasi matriks agar melakukan rotasi dan konstraksi secara bersamaan. Untuk melakukan hal tersebut, saya akan definisikan $r, 0 < r < 1$ dan melakukan operasi matriks sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} r \\ r \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

Oleh karena itu saya akan modifikasi program **R** sebelumnya menjadi sebagai berikut:

```
# mendefinisikan program
rotasi_konstraksi_kan = function(x0,rot,r){
  # menghitung theta
  theta = 2*pi/rot

  # definisi matriks rotasi
  A = matrix(c(cos(theta),-sin(theta),
              sin(theta),cos(theta)),
             ncol = 2,byrow = T)

  # membuat template
  temp = vector("list")
  temp[[1]] = x0

  # proses rotasi dan konstraksi
  for(i in 2:rot){
    xk = A %*% x0
    xk = r * xk
    temp[[i]] = xk
    x0 = xk
  }

  # membuat template data frame
  final = data.frame(x = rep(NA,rot),
                      y = rep(NA,rot))

  # gabung data dari list
  for(i in 1:rot){
    tempura = temp[[i]]
    final$x[i] = tempura[1]
    final$y[i] = tempura[2]
  }

  # membuat plot
  plot =
    ggplot() +
    geom_point(aes(x,y),data = final) +
    geom_point(aes(x[1],y[1]),
               data = final,
               color = "red") +
    coord_equal() +
    labs(title = "titik merah adalah titik initial")

  # enrich dengan garis panah
```

```

panah = data.frame(
  x_start = final$x[1:(rot-1)],
  x_end = final$x[2:rot],
  y_start = final$y[1:(rot-1)],
  y_end = final$y[2:rot]
)
# menambahkan garis panah ke plot
plot =
  plot +
  geom_segment(aes(x = x_start,
                    xend = x_end,
                    y = y_start,
                    yend = y_end),
               data = panah,
               arrow = arrow(length = unit(.3,"cm")))
)

# menyiapkan output
list("Grafik rotasi" = plot,
     "Titik-titik rotasi" = final)
}

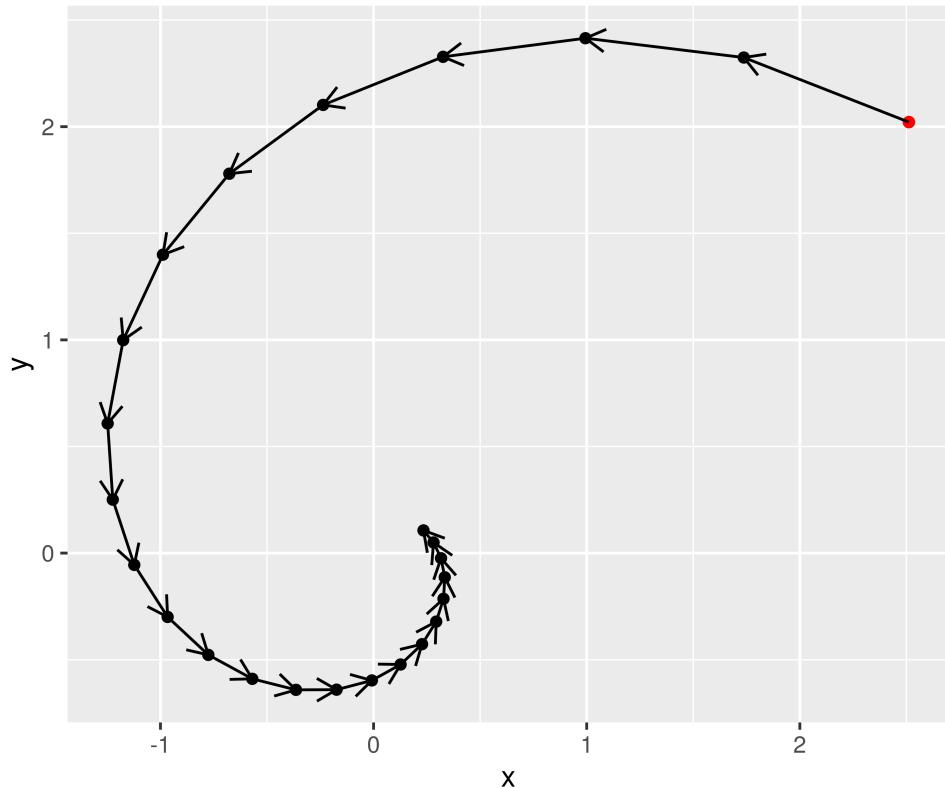
```

Saya akan uji coba untuk sembarang titik berikut ini:

```
# uji coba
rot = 25 # berapa banyak rotasi
x0 = rand_titik(0,4) # generate random titik
r = .9
rotasi_konstraksi_kan(x0,rot,r)
```

```
## $`Grafik rotasi`
```

titik merah adalah titik initial



```
##
## $`Titik-titik rotasi`  
##          x      y  
## 1  2.51174437  2.0213656  
## 2  1.73712612  2.3242555  
## 3  0.99407905  2.4149164  
## 4  0.32605466  2.3276383  
## 5  -0.23674414 2.1020379  
## 6  -0.67685577 1.7794104  
## 7  -0.98830123 1.3996618  
## 8  -1.17480030 0.9989175  
## 9  -1.24768122 0.6078363  
## 10 -1.22368119 0.2506089  
## 11 -1.12280481 -0.0554234  
## 12 -0.96637193 -0.2996211  
## 13 -0.77534895 -0.4774814  
## 14 -0.56902062 -0.5897717
```

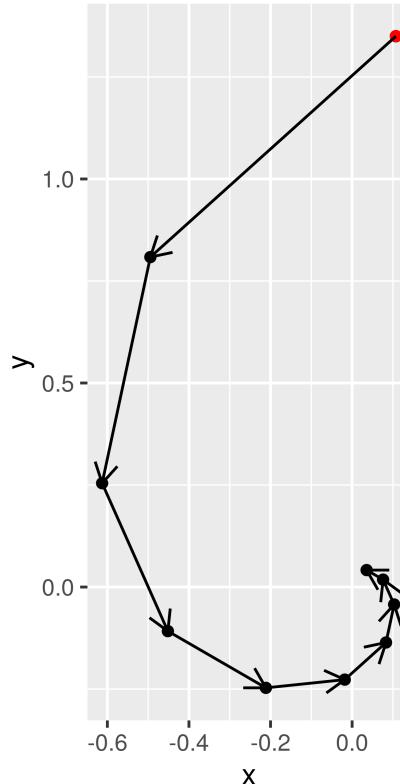
```
## 15 -0.36402617 -0.6414774
## 16 -0.17375462 -0.6406684
## 17 -0.00807123 -0.5973765
## 18  0.12666945 -0.5225544
## 19  0.22737951 -0.4271724
## 20  0.29382249 -0.3214845
## 21  0.32808732 -0.2144824
## 22  0.33400752 -0.1135369
## 23  0.31657458 -0.0242150
## 24  0.28138576  0.0497471
## 25  0.23415651  0.1063458
```

Saya akan uji coba kembali untuk sembarang titik lainnya berikut ini:

```
# uji coba
rot = 10 # berapa banyak rotasi
x0 = rand_titik(0,4) # generate random titik
r = .7
rotasi_konstraksi_kan(x0,rot,r)
```

```
## $`Grafik rotasi`
```

titik merah adalah titik initial



```
##
## $`Titik-titik rotasi`  
##           x          y  
## 1  0.1073273  1.3498148  
## 2  -0.4946001  0.8085760  
## 3  -0.6127862  0.2544031  
## 4  -0.4517022 -0.1080592  
## 5  -0.2113434 -0.2470479  
## 6  -0.0180385 -0.2268634  
## 7   0.0831274 -0.1358974  
## 8   0.1029910 -0.0427575  
## 9   0.0759176  0.0181615  
## 10  0.0355205  0.0415213
```

Catatan penting:

Terlihat bahwa semakin banyak rotasi dan konstraksi yang dilakukan akan membuat titik *initial* menuju **pusat** (0,0).

Operasi Matriks Rotasi dan Kontraksi dengan Titik x^* Sebagai Pusatnya

Salah satu prinsip utama dari *spiral optimization algorithm* adalah menjadikan titik x^* sebagai pusat rotasi di setiap iterasinya. Operasi matriksnya adalah sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} + \begin{bmatrix} r \\ r \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \left(\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} - \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} \right)$$

Oleh karena itu kita akan modifikasi program bagian sebelumnya menjadi seperti ini:

```
# mendefinisikan program
rotasi_konstraksi_pusat_kan = function(x0,rot,r,x_bin){
  # pusat rotasi
  pusat = x_bin

  # menghitung theta
  theta = 2*pi/rot

  # definisi matriks rotasi
  A = matrix(c(cos(theta),-sin(theta),
              sin(theta),cos(theta)),
             ncol = 2,byrow = T)

  # membuat template
  temp = vector("list")
  temp[[1]] = x0

  # proses rotasi dan kontraksi
  for(i in 2:rot){
    xk = A %*% (x0-pusat) # diputar dengan x_bin sebagai pusat
    xk = pusat + (r * xk)
    temp[[i]] = xk
    x0 = xk
  }

  # membuat template data frame
  final = data.frame(x = rep(NA,rot),
                      y = rep(NA,rot))

  # gabung data dari list
  for(i in 1:rot){
    tempura = temp[[i]]
    final$x[i] = tempura[1]
    final$y[i] = tempura[2]
  }

  # membuat plot
  plot =
    ggplot() +
    geom_point(aes(x,y),data = final) +
    geom_point(aes(x[1],y[1]),
               data = final,
               color = "red") +
    geom_point(aes(x = pusat[1],
                  y = pusat[2]),
```

```

        color = "blue") +
  labs(title = "titik merah adalah titik initial\ntitik biru adalah pusat rotasi")

# enrich dengan garis panah
panah = data.frame(
  x_start = final$x[1:(rot-1)],
  x_end = final$x[2:rot],
  y_start = final$y[1:(rot-1)],
  y_end = final$y[2:rot]
)
# menambahkan garis panah ke plot
plot =
  plot +
  geom_segment(aes(x = x_start,
                    xend = x_end,
                    y = y_start,
                    yend = y_end),
               data = panah,
               arrow = arrow(length = unit(.3,"cm")))
)

# menyiapkan output
list("Grafik rotasi" = plot,
     "Titik-titik rotasi" = final)
}

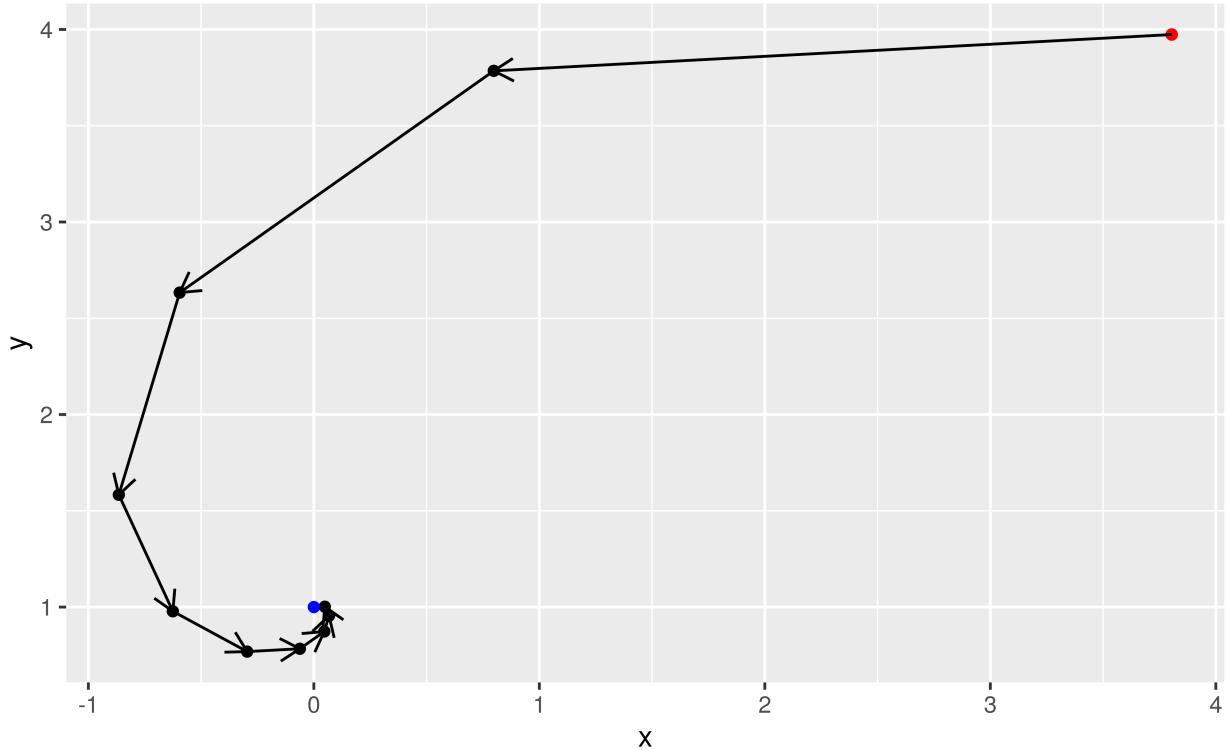
```

Saya akan coba dengan sembarang titik berikut:

```
# uji coba
rot = 10 # berapa banyak rotasi
x0 = rand_titik(0,4) # generate random titik
x_bintang = c(0,1) # contoh pusat rotasi
r = .6
rotasi_konstraksi_pusat_kan(x0,rot,r,x_bintang)
```

```
## $`Grafik rotasi`
```

titik merah adalah titik initial
titik biru adalah pusat rotasi



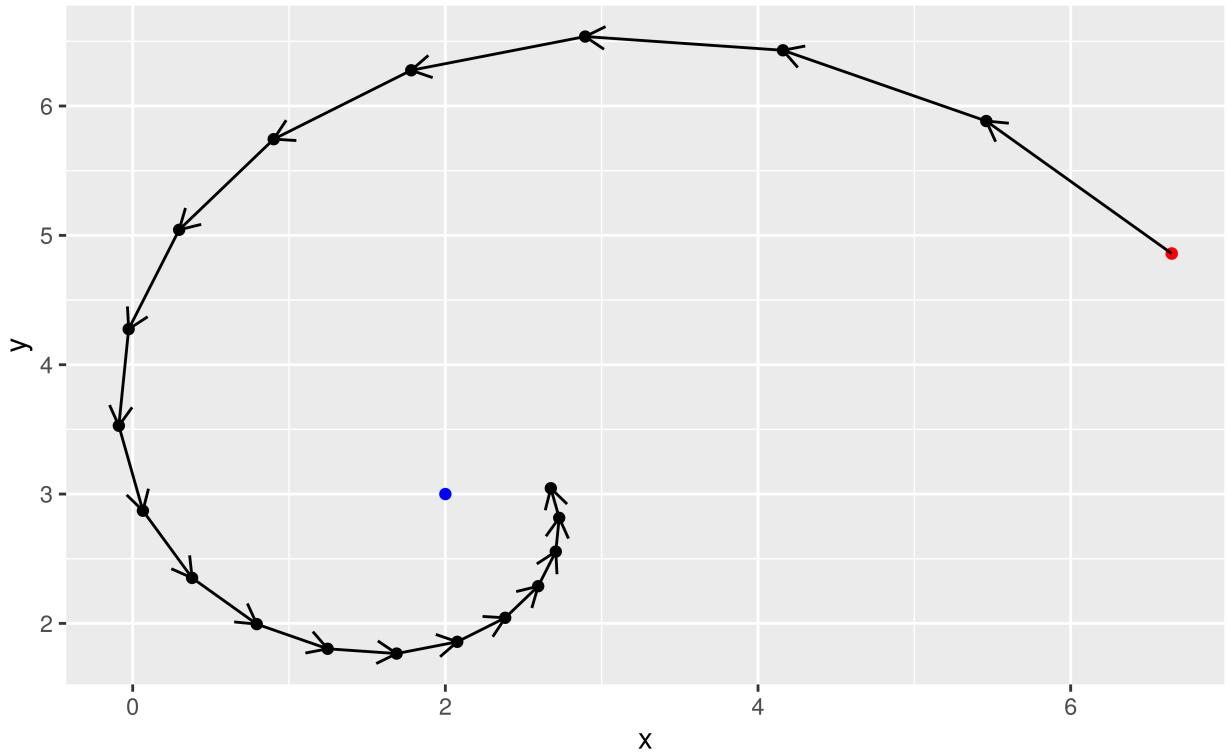
```
##
## $`Titik-titik rotasi`#
##          x      y
## 1   3.8044444 3.973729
## 2   0.7979677 3.785196
## 3  -0.5949167 2.633383
## 4  -0.8648256 1.583051
## 5  -0.6254203 0.978020
## 6  -0.2958336 0.768763
## 7  -0.0620500 0.783423
## 8   0.0462607 0.872988
## 9   0.0672488 0.954662
## 10  0.0486327 1.001709
```

Saya akan coba kembali dengan sembarang titik lainnya:

```
# uji coba
rot = 20 # berapa banyak rotasi
x0 = rand_titik(0,10) # generate random titik
x_bintang = c(2,3) # contoh pusat rotasi
r = .9
rotasi_konstraksi_pusat_kan(x0,rot,r,x_bintang)
```

```
## $`Grafik rotasi`
```

titik merah adalah titik initial
titik biru adalah pusat rotasi



```
##
## $`Titik-titik rotasi`  

##          x      y
## 1   6.6464312 4.85957
## 2   5.4599412 5.88395
## 3   4.1594701 6.43078
## 4   2.8942482 6.53716
## 5   1.7816942 6.27634
## 6   0.9019410 5.74367
## 7   0.2970585 5.04306
## 8  -0.0258407 4.27515
## 9  -0.0886575 3.52804
## 10  0.0653545 2.87109
## 11  0.3798896 2.35161
## 12  0.7935931 1.99443
## 13  1.2470393 1.80376
```

```
## 14 1.6881949 1.76667
## 15 2.0761185 1.85761
## 16 2.3828695 2.04334
## 17 2.5937790 2.28763
## 18 2.7063670 2.55538
## 19 2.7282699 2.81588
## 20 2.6745692 3.04495
```

Program *Spiral Optimization Algorithm*

Berbekal program yang telah dituliskan di bagian sebelumnya, kita akan sempurnakan program untuk melakukan *spiral optimization* sebagai berikut:

```
soa_mrf = function(N,      # banyak titik
                    x1_d,    # batas bawah x1
                    x1_u,    # batas atas x1
                    x2_d,    # batas bawah x2
                    x2_u,    # batas atas x2
                    rot,     # berapa banyak rotasi
                    k_max,   # iterasi maks
                    r){      # berapa rate konstraksi

  # N pasang titik random di selang [a,b] di R2
  x1 = runif(N,x1_d,x1_u)
  x2 = runif(N,x2_d,x2_u)

  # hitung theta
  theta = 2*pi / rot

  # definisi matriks rotasi
  A = matrix(c(cos(theta),-sin(theta),
              sin(theta),cos(theta)),
             ncol = 2,byrow = T)

  # bikin data frame
  temp = data.frame(x1,x2) %>% mutate(f = f(x1,x2))

  # proses iterasi
  for(i in 1:k_max){
    # mencari titik x* dengan min(f)
    f_min =
      temp %>%
      filter(f == min(f))
    pusat = c(f_min$x1,f_min$x2)

    for(j in 1:N){
      # kita akan ambil titiknya satu persatu
      x0 = c(temp$x1[j],temp$x2[j])

      # proses rotasi dan konstraksi terhadap pusat x*
      xk = A %*% (x0-pusat) # diputar dengan x_bin sebagai pusat
      xk = pusat + (r * xk)

      # proses mengembalikan nilai ke temp
      temp$x1[j] = xk[1]
      temp$x2[j] = xk[2]
    }

    # hitung kembali nilai f(x1,x2)
    temp = temp %>% mutate(f = f(x1,x2))
  }

  # proses output hasil
```

```

output = temp %>% filter(f == min(f))
return(output)
}

```

Contoh Penggunaan Program

Kita akan coba performa program tersebut untuk menyelesaikan fungsi berikut:

$$f(x_1, x_2) = \frac{x_1^4 - 16x_1^2 + 5x_1}{2} + \frac{x_2^4 - 16x_2^2 + 5x_2}{2}$$

$$-4 \leq x_1, x_2 \leq 4$$

Dengan $r = 0.8, N = 50, rot = 20, k_{max} = 60$.

```

# definisi
N = 50
a = -4 # x1 dan x2 punya batas bawah yang sama
b = 4 # x1 dan x2 punya batas atas yang sama
k_max = 70
r = .75
rot = 30
f = function(x1,x2){
  ((x1^4 - 16 * x1^2 + 5 * x1)/2) + ((x2^4 - 16 * x2^2 + 5*x x2)/2)
}

# solving
soa_mrf(N,a,b,a,b,rot,k_max,r)

##           x1          x2          f
## 1 -2.92184 -3.01932 -78.0856

```

Catatan

Pada algoritma ini, penentuan θ, r, x menjadi penentu hasil perhitungan.

Mengubah Optimisasi Menjadi Pencarian Akar

Spiral optimization algorithm adalah suatu metode untuk mencari solusi minimum global. Jika kita hendak memakainya untuk mencari suatu akar persamaan (atau sistem persamaan), kita bisa melakukan modifikasi pada fungsi-fungsi yang terlibat (membuat fungsi *merit*).

Misalkan suatu sistem persamaan non linear:

$$g_1(x_1, x_2, \dots, x_n) = 0$$

$$g_2(x_1, x_2, \dots, x_n) = 0$$

$$g_n(x_1, x_2, \dots, x_n) = 0$$

dengan $(x_1, x_2, \dots, x_n)^T \in D$

$$D = a_1, b_1 \times a_2, b_2 \times \dots \times a_n, b_n \subset \mathbb{R}^n$$

Pencarian Akar

Sistem di atas memiliki solusi $x = (x_1, x_2, \dots, x_n)^T$ jika $F(x)$ yang kita definisikan sebagai:

$$F(x) = \frac{1}{1 + \sum_{i=1}^n |g_i(x)|}$$

memiliki nilai maksimum sama dengan 1. **Akibatnya algoritma yang sebelumnya adalah mencari $\min F(x)$ diubah menjadi $\max F(x)$.** Kenapa demikian?

Karena jika $F(x) = 1$ artinya $\sum_{i=1}^n |g_i(x)| = 0$ yang merupakan akar dari $g_i, i = 1, 2, \dots, n$.

Kelak $F(x)$ akan digunakan untuk menjawab soal-soal yang ada dalam tugas ini.

SOAL 1

Tentukanlah akar-akar sistem persamaan berikut dengan **SOA**. Buatlah terlebih dahulu *contour plot*-nya:

$$f_1(x_1, x_2) = \cos(2x_1) - \cos(2x_2) - 0.4 = 0$$

$$f_2(x_1, x_2) = 2(x_2 - x_1) + \sin(x_2) - \sin(x_1) - 1.2 = 0$$

dengan $-10 \leq x_1, x_2 \leq 10$

JAWAB

Contour Plot

Pertama-tama, saya akan buat *contour plot* dari $f_1(x_1, x_2)$ sebagai berikut:

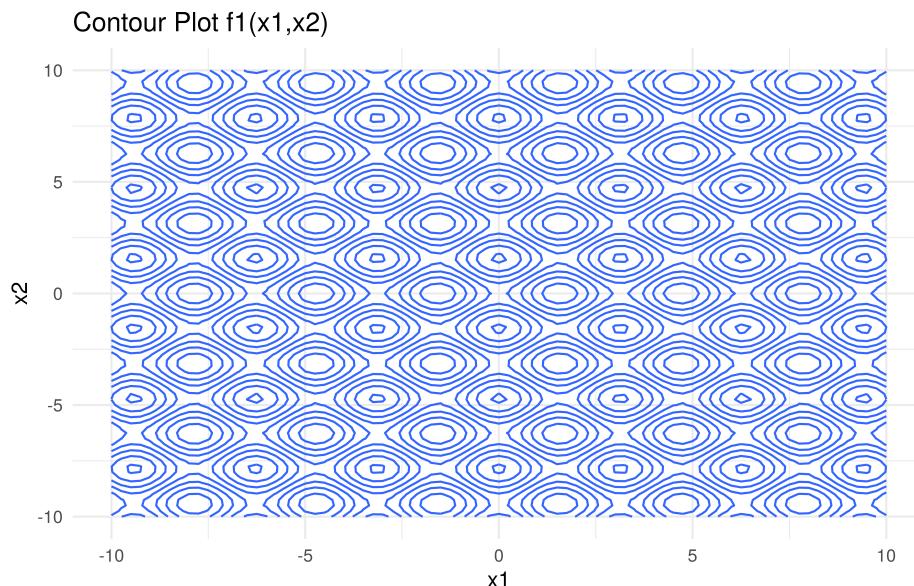


Figure 1: Contour Plot Soal 1: f1

Selanjutnya, saya akan buat *contour plot* dari $f_2(x_1, x_2)$ sebagai berikut:

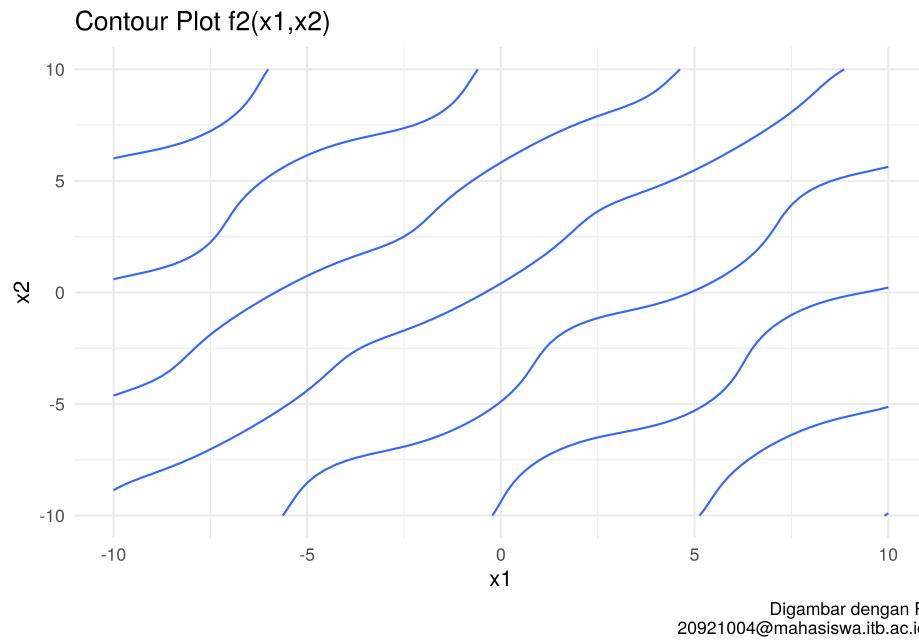


Figure 2: Contour Plot Soal 1: f_2

Grafik Sistem Persamaan

Kita akan mencari akar-akar sistem persamaan saat $f_1 = 0$ dan $f_2 = 0$ dengan bantuan grafik sebagai berikut:

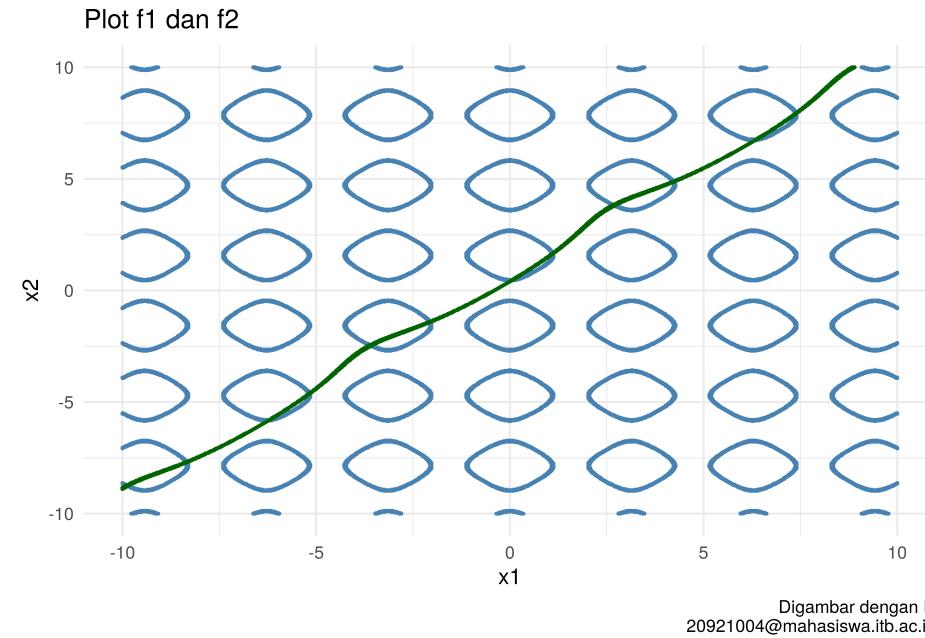


Figure 3: Plot Soal 1: f1 dan f2

Terlihat bahwa ada beberapa titik solusi (persinggungan antara $f_1(x_1, x_2)$ dengan $f_2(x_1, x_2)$).

Mencari Akar Sistem Persamaan

Untuk mencari akarnya kita perlu membentuk $F(x)$ sebagaimana yang telah dijelaskan pada bagian sebelumnya.

```
# fungsi f1 dan f2 dari soal
f1 = function(x1,x2){cos(2*x1) - cos(2*x2) - 0.4}
f2 = function(x1,x2){2*(x2 - x1) + sin(x2) - sin(x1) - 1.2}

# membuat F(x)
# saya notasikan sebagai f kecil
f = function(x1,x2){
  sum = abs(f1(x1,x2)) + abs(f2(x1,x2))
  bawah = 1 + sum
  hasil = 1/bawah
  return(hasil)
}
```

Oleh karena solusi dari grafik ada banyak, maka kita akan *run* program yang telah dibuat sebelumnya berulang kali:

```
# solving
N = 50
a = -10 # x1 dan x2 punya batas yang sama
b = 10 # x1 dan x2 punya batas yang sama
rot = 20
k_max = 60
r = .65
# run I
soa_mrf_2(N,a,b,a,b,rot,k_max,r)
```

```
##           x1      x2      f
## 1 0.0629407 0.46858 0.999991
```

```
# run II
soa_mrf_2(N,a,b,a,b,rot,k_max,r)
```

```
##           x1      x2 f
## 1 1.09995 1.64699 1
```

```
# run III
soa_mrf_2(N,a,b,a,b,rot,k_max,r)
```

```
##           x1      x2      f
## 1 -5.18231 -4.63439 0.99676
```

```
# run IV
soa_mrf_2(N,a,b,a,b,rot,k_max,r)
```

```
##           x1      x2      f
## 1 4.24122 4.87605 0.937282
```

```
# run V
soa_mrf_2(N,a,b,a,b,rot,k_max,r)
```

```
##           x1      x2      f
## 1 1.10065 1.64801 0.998305
```

Saya coba *run* sebanyak **100 kali**, berikut adalah rekap akar yang saya dapatkan:

```
##          x1          x2          f
## 1 -6.2202507 -5.814611 1.000000
## 2 -2.0651615 -1.411618 1.000000
## 3 -6.2258074 -5.821061 0.989107
## 4  4.2180238  4.871568 1.000000
## 5 -8.4439331 -7.834121 0.731311
## 6 -3.5971060 -2.463483 1.000000
## 7 -3.5971059 -2.463483 1.000000
## 8 -2.0651623 -1.411616 0.999994
## 9  4.2180238  4.871568 1.000000
## 10 1.0983290  1.653737 0.982891
## 11 -5.1830740 -4.636155 0.999394
## 12 4.2151492  4.869324 0.993378
## 13 4.2181408  4.871654 0.999743
## 14 6.3445433  6.750371 0.997451
## 15 6.3461194  6.751759 0.999999
## 16 1.1604812  1.725451 0.884780
## 17 4.2180238  4.871568 1.000000
## 18 -3.5977632 -2.463034 0.998567
## 19 4.2180246  4.871569 0.999995
## 20 -3.6053137 -2.472277 0.969126
## 21 -5.1832392 -4.636197 0.999998
## 22 -3.5160072 -2.574317 0.763512
## 23 -3.6495200 -2.489611 0.859496
## 24 1.1000048  1.646918 0.999645
## 25 -8.4873965 -7.722650 0.708963
## 26 1.1178240  1.651209 0.938563
## 27 1.1086326  1.618530 0.923055
## 28 -8.3369710 -7.686657 0.976286
## 29 0.0721901  0.479637 0.980678
## 30 -6.2985658 -5.897778 0.888948
## 31 -5.1861474 -4.644763 0.983727
## 32 0.0595642  0.465045 0.995070
## 33 2.6958042  3.846928 0.916144
## 34 9.0026268 10.141386 0.878682
## 35 2.6860794  3.819702 1.000000
## 36 1.1008538  1.648148 0.998177
## 37 0.0617748  0.467355 0.998280
## 38 -5.1827983 -4.648079 0.974066
## 39 0.0629939  0.468644 0.999877
## 40 1.0999452  1.646987 1.000000
## 41 -6.2202507 -5.814611 1.000000
## 42 1.0929512  1.633264 0.976112
## 43 6.3677504  6.774259 0.970291
## 44 -6.2202507 -5.814611 1.000000
## 45 -6.2394220 -5.834797 0.971871
## 46 0.0629346  0.468574 1.000000
## 47 2.6860794  3.819702 1.000000
## 48 4.2173296  4.871687 0.997604
## 49 4.2180209  4.871575 0.999979
## 50 -2.0630014 -1.408400 0.990794
## 51 0.0629495  0.468590 0.999976
```

```

## 52   1.0999452  1.646987 1.000000
## 53   0.0630169  0.468529 0.999529
## 54   -6.2261214 -5.821183 0.989561
## 55   -5.1828646 -4.635312 0.998342
## 56   -5.1999990 -4.718615 0.862335
## 57    6.4505583  6.860882 0.876360
## 58    6.3580306  6.765244 0.978539
## 59    6.5000779  6.913234 0.831977
## 60   -5.1774704 -4.671312 0.923031
## 61   -2.0684062 -1.407766 0.984001
## 62   -2.0651610 -1.411618 0.999999
## 63   -2.0651615 -1.411618 1.000000
## 64   -3.5445243 -2.473858 0.885488
## 65   -2.0651615 -1.411618 1.000000
## 66    0.0575901  0.471329 0.971132
## 67   -5.2288905 -4.696984 0.914795
## 68    4.2180334  4.871576 0.999975
## 69   -2.0651649 -1.411650 0.999909
## 70   -2.0651615 -1.411618 1.000000
## 71    0.0623167  0.467740 0.998256
## 72    1.0997230  1.647162 0.998813
## 73   -5.2446087 -4.720744 0.885853
## 74    2.6860794  3.819702 1.000000
## 75   -6.2597166 -5.839952 0.927488
## 76   -3.5971060 -2.463483 1.000000
## 77    2.6799640  3.848249 0.918898
## 78    0.0626612  0.468290 0.999610
## 79    4.2180238  4.871568 1.000000
## 80    1.0958598  1.640783 0.989750
## 81    7.3831605  7.930157 0.999853
## 82    4.2468317  4.899789 0.923059
## 83    0.0629346  0.468574 1.000000
## 84   -2.0651615 -1.411618 1.000000
## 85    2.7918686  3.815032 0.791680
## 86    0.0913805  0.498422 0.960376
## 87    4.2180238  4.871568 1.000000
## 88    6.3513342  6.759555 0.982150
## 89    0.1553023  0.564977 0.888481
## 90   -5.1811665 -4.639581 0.986248
## 91   -2.0806234 -1.420394 0.965237
## 92   -3.5971059 -2.463483 1.000000
## 93    2.6874752  3.818592 0.997081
## 94   -2.0651622 -1.411616 0.999995
## 95   -0.0253125  0.377479 0.884869
## 96    4.2114526  4.888048 0.956108
## 97   -2.0674660 -1.405829 0.984064
## 98    2.6860795  3.819702 1.000000
## 99   -5.1832402 -4.636198 1.000000
## 100   -3.5971060 -2.463484 0.999999

```

SOAL 2

Tentukanlah akar-akar sistem persamaan berikut dengan **SOA**. Buatlah terlebih dahulu *contour plot*-nya:

$$f_1(x_1, x_2) = \sin(x_1) \cos(x_2) + 2 \cos(x_1) \sin(x_2) = 0$$

$$f_2(x_1, x_2) = \cos(x_1) \sin(x_2) + 2 \sin(x_1) \cos(x_2) = 0$$

dengan $0 \leq x_1, x_2 \leq 2\pi$

Contour Plot

Pertama-tama, saya akan buat *contour plot* dari $f_1(x_1, x_2)$ sebagai berikut:

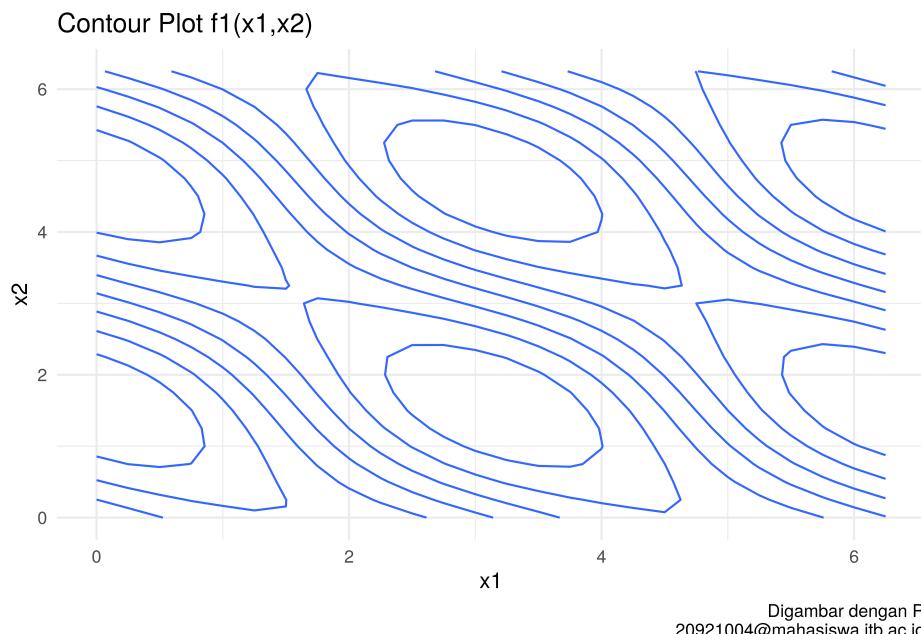


Figure 4: Contour Plot Soal 2: f_1

Berikutnya adalah *contour plot* dari $f_2(x_1, x_2)$ sebagai berikut:

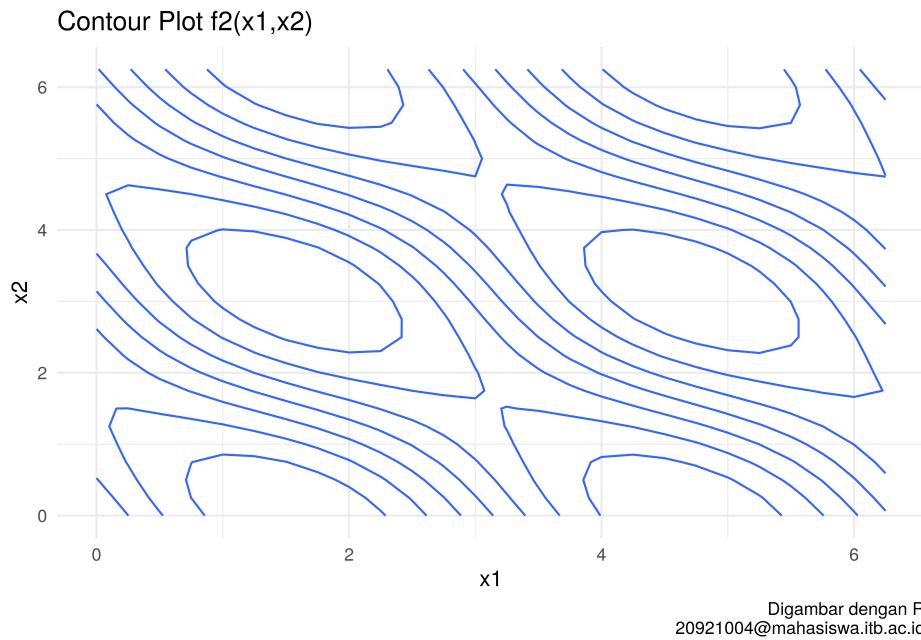


Figure 5: Contour Plot Soal 2: f_2

Grafik Sistem Persamaan

Kita akan mencari akar-akar sistem persamaan saat $f_1 = 0$ dan $f_2 = 0$ dengan bantuan grafik sebagai berikut:

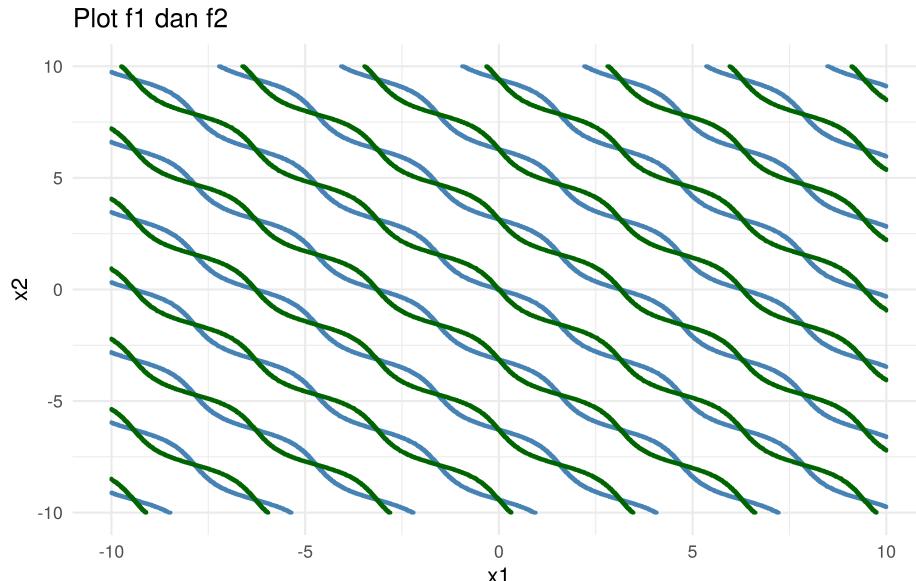


Figure 6: Plot Soal 1: f1 dan f2

SOAL 3

Tentukanlah akar-akar sistem persamaan berikut dengan **SOA**. Buatlah terlebih dahulu *contour plot*-nya:

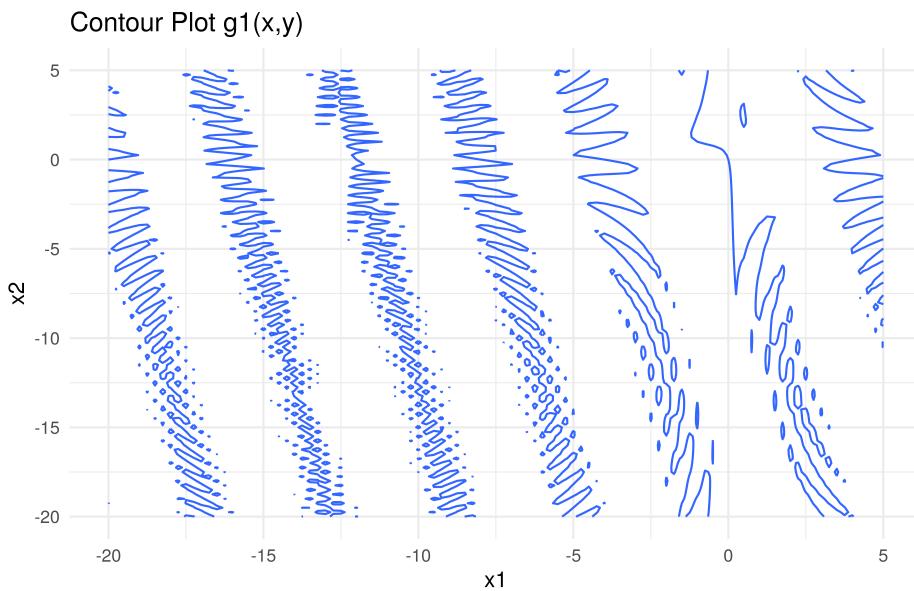
$$g_1(x, y) = 0.5 \sin(xy) - 0.25 \frac{y}{\pi} - 0.5x = 0$$

$$g_2(x, y) = \left(1 - \frac{0.25}{\pi}\right)(e^{2x} - e) + e \frac{y}{\pi} - 2ex = 0$$

dengan $-1 \leq x \leq 3, -20 \leq y \leq 5$

Contour Plot

Pertama-tama, saya akan buat *contour plot* dari $g_1(x, y)$ sebagai berikut:



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

Figure 7: Contour Plot Soal 3: g_1

Berikutnya adalah *contour plot* dari $g_2(x, y)$ sebagai berikut:

Grafik Sistem Persamaan

Kita akan mencari akar-akar sistem persamaan saat $f_1 = 0$ dan $f_2 = 0$ dengan bantuan grafik sebagai berikut:

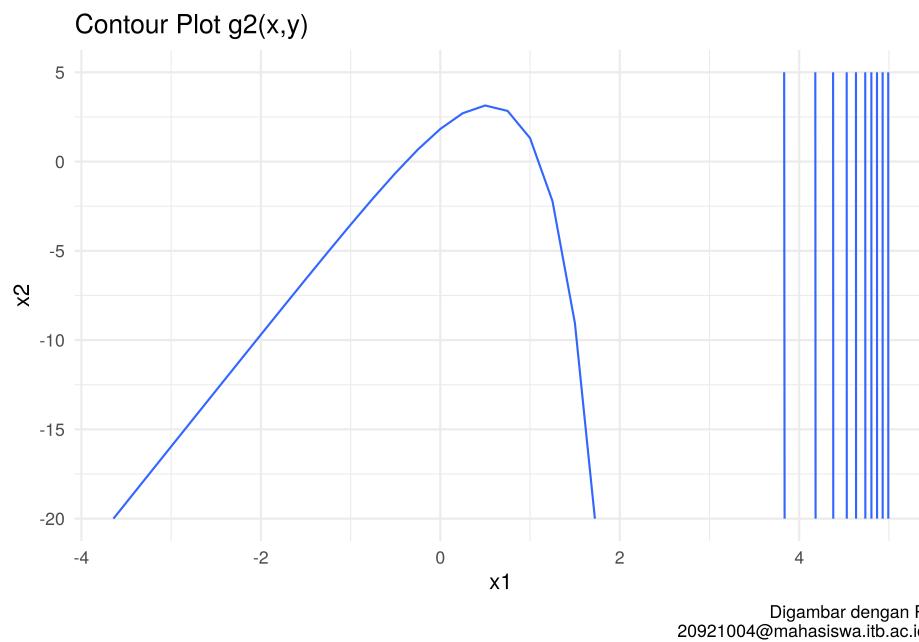


Figure 8: Contour Plot Soal 3: g2

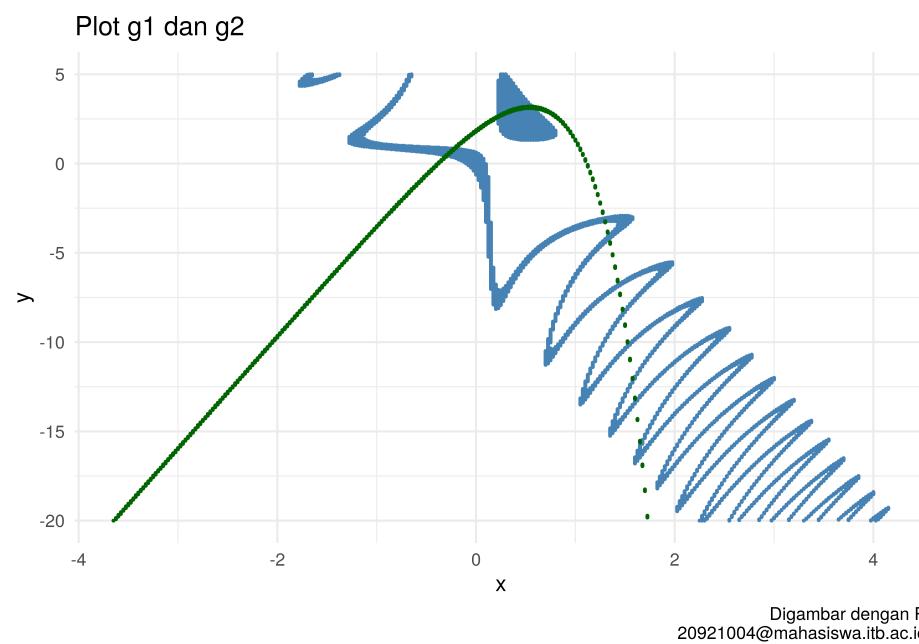


Figure 9: Plot Soal 1: f1 dan f2