Achmad Iqbal Al Faizin
24060119140113
Lab A1

BAB I
PENDAHULUAN

1.1 Tujuan

Dapat menentukan penyelesaian Optimasi tak berkendala multivariabel secara Numerik dengan metode Newton.

1.2 Rumusan Permasalahan

1. Diketahui: A = [1 1 1 2], b = [−1 −1], c = 0, x = [$x1$ $x2$]

Carilah minimum dari fungsi kuadratik (x) yang diperoleh dari matriks A, b, x, dan konstanta 0 di atas, dengan toleransi α = 0.01 dan dimulai dari titik awal $x0$ = [0 0].

Atau dapat ditulis

Min ($x1$, $x2$) = 1 2 x tAx + b t x+ c

= 1 2 $x1$ 2 + $x2$ 2 + $x1x2$ − $x1$ − $x2$

Tentukan nilai minimum dari ($x1$, $x2$)

2. Tentukan Min ($x1$, $x2$) = ($x1$ − 0.5) 2 ($x1$ + 1) 2 + ($x2$ + 1) 2 ($x2$ − 1) 2 , dengan

toleransi α = 0.01 dan dimulai dari titik awal $x0$ = [0 0.5]

Achmad Iqbal Al Faizin
24060119140113
Lab A1

BAB II
DASAR TEORI

2.1 Pendahuluan Teori

Dalam kehidupan sehari-hari baik disadari maupun tidak, sebenarnya orang selalu melakukan optimasi untuk memenuhi kebutuhannya. Tetapi optimasi yang dilakukan masyarakat awam lebih banyak dilandasi oleh intuisi daripada teori optimasi. Dalam masalah optimasi terdapat dua bentuk optimasi yaitu fungsi optimasi tanpa kendala dan dengan kendala. Banyak aplikasi dari pemodelan matematika yang berbentuk linear atau nonlinear dalam optimasi fungsi yang mensyaratkan beberapa kendala ataupun tanpa kendala untuk diperoleh suatu solusi optimal. Permasalahan ini berfungsi untuk mencari nilai ektrim dari fungsi tujuan. Persoalan dengan model nonlinear yang tidak disertai kendala dinamakan optimasi nonlinear tanpa kendala. Optimasi merupakan masalah yang berhubungan dengan keputusan yang terbaik, maksimum, minimum dan memberikan cara penentuan solusi yang memuaskan. Terdapat beberapa metode dalam mencari optimasi multivariable tanpa kendala. Diantaranya metode Univariate, Steepest Descent, dan Newton. Dalam metode Univariate membutuhkan iterasi yang banyak dalam mencari nilai optimasinya. Begitu pula dengan metode Steepest Descent yang nilai iterasinya tergantung pada pemilihan nilai , semaki kecil semakin banyak iterasi yang dibutuhkan. Dari berbagai masalah itu metode Newton memberikan alternatif yang lebih bagus. Dalam makalah ini akan dijelaskan tentang metode Newton beserta contoh kasusnya.

METODE NEWTON

Fungsi Multivariable tanpa Kendala Cara analitis yang diterapkan pada permaslahan optimasi satu variable dapat diterapkan pada permasalahan multivariable. Dalam makalah ini akan dibahas masalah optimasi untuk fungsi lebih dari satu variabel. Seperti masalah minimisasi pada satu variabel; min $z = (x1, x2, \ldots, xn)$

Sedangkan bentuk umum dari fungsi multivariable:

$$\min z = f(x) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in R^n$$

Nashirudin Baqiy
24060119130045
Lab A1

Merupakan fungsi dari n variabel $f: R^n \rightarrow R$ tranpose dari matriks x ditulis dengan

$$x^t = [x_1, x_2, \dots, x_n]$$

Penggunaan turunan sebagai syarat perlu dan cukup: $\partial f(x) = 0$ dan $\partial^2 f(x)$ definit positif.

Dalam analisis numerik, metode Newton juga dikenal dengan metode Newton Rapshon yang mendapat nama dari Isaac Newton (1669) dan Joseph Rapshon (1690), merupakan suatu metode yang cukup dikenal untuk mencari pendekatan terhadap akar fungsi riil.

Metode Newton Rapshon sering konvergen dengan cepat, terutama bila dimulai cukup dekat dengan akar yang diinginkan. Namun, bila iterasi dimulai jauh dari akar yang dicari, metode ini dapat melesat tanpa peringatan. Implementasi metode ini biasanya mendeteksi dan mengatasi kegagalan konvergensi. Gagasan awal metode Newton Rapshon adalah metode yang digunakan untuk mencari akar dari sebuah fungsi riil. Metode ini dimulai dengan memperkirakan satu titik awal dan mendekatinya dengan memperlihatkan slope atau gradient pada titik tersebut. Diharapkan dari titik awal tersebut akan diperoleh pendekatan terhadap akar fungsi yang dimaksud.

Diberikan fungsi $f: R^n \rightarrow R$

$\nabla^2 f(x)$ definit positif untuk semua x , jadi f juga konveks. Misal x_k suatu vektor tertentu (merupakan estimasi untuk (x^*). Untuk menemukan estimasi baru x_{k+1} sebagai berikut. Perhatikan pendekatan kuadratis lewat ekspansi taylor dari fungsi f di titik x_k.

$$P(x) = f(x_k) + (x - x_k).\nabla f(x_k) + \frac{1}{2}\nabla^2 f(x_k)(x - x_k)$$

$$\nabla P(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) \cong 0$$

$$\nabla^2 P(x) = \nabla^2 f(x_k)$$

Sehingga $\nabla^2 P(x)$ definit positif dan $P(x)$ adalah konveks. Estimasi baru untuk x^* dapat dengan mencari minimum dari $P(x)$ diperoleh

ALGORITMA

Algoritma Input

Input $f: R^n \rightarrow R$; ∇f , $\nabla^2(f)$, definit positif, estimasi nilai awal $x_0 \in R^n$, toleransi (α), $k = 0$

Step : 1. $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} . \nabla f(x_k)$

2. jika $\|\nabla\text{^}2\, f(x\_k)\| < \alpha$, STOP. $x^* = x_{k+1}$

3. $k = k + 1$, ulangi langkah 1.

Nashirudin Baqiy
24060119130045
Lab A1

BAB III
PEMBAHASAN

3.1 Penyelesaian dengan Metode Eliminasi Gauss

3.1.1 Source Code

Nomor 1

```
#metode newton rapshon
from sympy import *
import numpy as np
from scipy.integrate import quad
import matplotlib.pylab as plt
import sys

x, y = symbols('x y', real = true)
fx = 0.5*(x**2) + y**2 + x*y - x - y
#fx = (x-0.5)**2 *(x + 1)**2 + (y + 1)**2 *(y - 1)**2
func = lambdify([x,y], fx, 'sympy')
difx = diff(fx, x)
funcx = lambdify([x,y], difx, 'sympy')
dify = diff(fx, y)
funcy = lambdify([x,y], dify, 'sympy')
hasil = [0, 0] #hasil/tebakan awal
#hasil = [0, 0.5] #hasil/tebakan awal
iterasi = 0
g = [funcx(hasil[0], hasil[1]), funcy(hasil[0], hasil[1])]

print("fx = ", fx)
print("tebakan awal :" '{0:.17f}'.format(hasil[0]),
'{0:.17f}'.format(hasil[1]))
while (g!=[0, 0] and iterasi<1000):
    J1 = lambdify([x,y], diff(diff(fx, x), x), 'sympy')
    J2 = lambdify([x,y], diff(diff(fx, x), y), 'sympy')
    J3 = lambdify([x,y], diff(diff(fx, y), x), 'sympy')
    J4 = lambdify([x,y], diff(diff(fx, y), y), 'sympy')

    J = [
        [J1(hasil[0],hasil[1]), J2(hasil[0],hasil[1])],
        [J3(hasil[0],hasil[1]), J4(hasil[0],hasil[1])]
    ]
    Jinv = np.dot((1/((J[0][0]*J[1][1])-(J[0][1]*J[1][0]))),
[[J[1][1], -J[0][1]],[-J[1][0], J[0][0]]])
    g = [funcx(hasil[0], hasil[1]), funcy(hasil[0], hasil[1])]
    hasil = hasil-np.dot(Jinv, g)
    print("iterasi ke",iterasi,":" '{0:.17f}'.format(hasil[0]),
'{0:.17f}'.format(hasil[1]))
```

```
    iterasi+=1
```

3.1.2 Hasil

```
fx =  0.5*x**2 + x*y - x + y**2 - y
tebakan awal :0.00000000000000000 0.00000000000000000
iterasi ke 0 :1.00000000000000000 0.00000000000000000
iterasi ke 1 :1.00000000000000000 0.00000000000000000
```

3.1.4 Penjelasan

Program di atas adalah program untuk mencari nilai minimum dari sebuah fungsi. Hal pertama yang dilakukan adalah menginput library autograd dan kemudian membuat program untuk metode newton nya. Setelah itu membuat program main nya. Disitu di input fungsi yang ingin dicari, dan titik awalnya, dan kemudian hasilnya akan ditampilkan

BAB IV
PENUTUPAN

4.1 Kesimpulan

Dengan menggunakan program di atas, kita dapat mencari nilai minimum dari suatu fungsi. Untuk soal pertama, nilai yang dihasilkan adalah [1 0] sedangkan untuk soal kedua nilai yang dihasilkan adalah [-0.25000003 -1].