WROCLAW UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRONICS

FIELD: Electronics
SPECIALITY: Advanced Applied Electronics

Optimization Methods: Nonlinear Equations

AUTHOR:
Jaroslaw M. Szumega

SUPERVISOR:
Rafal Zdunek, D.Sc, K-4/W4

GRADE:

WROCŁAW 08.06.2017

Contents

Chapter 1

Solution to the given problems

Problem 1: Show that the following system of nonlinear equations:

$$F_1(\mathbf{x}) = x_1^3 + 3x_1^2 + 3x_1 - x_2 = 0,$$
$$F_2(\mathbf{x}) = x_1^2 + 2x_1 - x_2 + 1 = 0,$$

has the global minimum at $x^* = \begin{bmatrix}0.46557 & 2.1479\end{bmatrix}^T$, the local minimum at $x_1 = \begin{bmatrix}-1 & -0.5\end{bmatrix}^T$, and the saddle point at $x_2 = \begin{bmatrix}-\frac{1}{3} & -\frac{7}{54}\end{bmatrix}^T$.

In general, the unconstrained nonlinear least squares problem is equivalent to:

$$\min_x \sum_{i=1}^{N} F_i^2(x)$$

with the objective function:

$$f(x) = \frac{1}{2}\sum_{i=1}^{N} F_i^2(x)$$

In the case of the following task, objective function will be

$$\begin{aligned}
f(x) &= \frac{1}{2}(F_1^2 + F_2^2) \\
&= \frac{1}{2}[(x_1^6 + 6x_1^5 + 15x_1^4 - 2x_1^3x_2 + 18x_1^3 - 6x_1^2x_2 + 9x_1^2 - 6x_1x_2 + x_2^2) \\
+ &\quad (x_1^4 + 4x_1^3 - 2x_1^2x_2 + 6x_1^2 - 4x_1x_2 + 4x_1 + x_2^2 - 2x_2 + 1)] \\
&= \frac{1}{2}(x_1^6 + 6x_1^5 + 16x_1^4 - 2x_1^3x_2 + 22x_1^3 - 8x_1^2x_2 + 15x_1^2 - 10x_1x_2 + 4x_1 + 2x_2^2 - 2x_2 + 1)
\end{aligned}$$

And the points to check are:

$$\begin{aligned}
x^* &= [0.46557\ 2.1479]^T \\
x_1 &= [-1\ \ -0.5]^T \\
x_2 &= [-\frac{1}{3}\ \ -\frac{7}{54}]^T
\end{aligned}$$

Firstly, we can calculate the cost function gradient to verify above–mentioned points:

$$\nabla f(x^*) = 0$$

$$\frac{\delta f(x)}{\delta x_1} = 3x_1^5 + 15x_1^4 + 32x_1^3 - 3x_1^2 x_2 + 33x_1^2 - 8x_1 x_2 + 15x_1 - 5x_2 + 2$$
$$\frac{\delta f(x)}{\delta x_2} = -x_1^3 - 4x_1^2 - 5x_1 + 2x_2 - 1$$

And in fact, the mentioned in task description points are recognized as a solutions of the cost function gradient.

To check their nature, we need to calculate the Hessian and evaluate it according to the stationary points.

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\delta^2 f(x)}{\delta x_1^2} & \frac{\delta^2 f(x)}{\delta x_1 x_2} \\ \frac{\delta^2 f(x)}{\delta x_1 x_2} & \frac{\delta^2 f(x)}{\delta x_2^2} \end{bmatrix}$$

$$\nabla^2 f(x) = \begin{bmatrix} 15x_1^4 + 60x_1^3 + 96x_1^2 - 6x_1 x_2 + 66x_1 - 8x_2 + 15 & -3x_1^2 + 8x_1 + \\ -3x_1^2 + 8x_1 + 5 & 2 \end{bmatrix}$$
$$for\ point\ [0.46557\ 2.1479] = \begin{bmatrix} 50.11 & -9.37 \\ -9.37 & 2 \end{bmatrix}$$
$$det > 0, M1 > 0, f(x) = 5.01e-11,\ there\ is\ a\ minimum\ (global)$$

$$for\ point\ [-1\ \ -0.5] = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$
$$det > 0, M1 > 0, f(x) = 0.25,\ there\ is\ a\ minimum$$

$$for\ point\ [-\frac{1}{3}\ \ -\frac{7}{54}] = \begin{bmatrix} 2.41 & -2.66 \\ -2.66 & 2 \end{bmatrix}$$
$$det < 0, M1 > 0, f(x) = 0.33,\ there\ is\ a\ saddle\ point$$

To make all the calculations above, the following script in Python was written to perform symbolic computations:

```
#!/usr/bin/python

from sympy import diff, Symbol, latex, Matrix


def optimization(function, x1, x2):

    print latex(function)

    #calculating the gradient's elements
    firstx1 = diff(function, x1)
    firstx2 = diff(function, x2)

    print firstx1.expand()
    print firstx2.expand()


    #calculations for Hessian elements
    secondx1x1 = diff(function, x1, x1)
    secondx1x2 = diff(function, x1, x2)
    secondx2x1 = diff(function, x2, x1)
    secondx2x2 = diff(function, x2, x2)

    print "\n Hessian elements:\n\n"
    print latex(secondx1x1) +"\t" + latex(secondx1x2)
    print latex(secondx2x1) + "\t"+latex(secondx2x2)


    point1 = [0.46557, 2.1479]
    point2 = [-1.0, -0.5]
    point3 = [-1.0/3, -7.0/54]

    point = point3
    print secondx1x1.subs(x1, point[0]).subs(x2, point[1]);
    print secondx1x2.subs(x1, point[0]).subs(x2, point[1]);
    print secondx2x1.subs(x1, point[0]).subs(x2, point[1]);
    print secondx2x2.subs(x1, point[0]).subs(x2, point[1]);
    print "\n\n"
    print function.subs(x1, point[0]).subs(x2, point[1]);


def main():
    x1 = Symbol('x1')
    x2 = Symbol('x2')

    f1 = x1**3 + 3*x1**2 + 3*x1 - x2
    f2 = x1**2+2*x1 -x2 + 1
    f = (f1**2 + f2**2)/2

    function = f.expand()
    optimization(function, x1,x2)

main()
```

Problem 2: Solve the following system of nonlinear equations with the selected nonlinear least squares methods.

$$F_1(\mathbf{x}) = x_1^3 - 3x_1^2 + 3x_1 - x_2 - 3 = 0,$$
$$F_2(\mathbf{x}) = x_1^2 - 2x_1 - x_2 = 0.$$

Draw the contour lines of the cost function in the nonlinear least square problem. Check the optimality points.

Chapter 2

Listings of algorithms

2.1 Coded selected algorithms

Algorithm 1 - Simplex algorithm

Algorithm 2 - Revised Simplex algorithm

Bibliography

[1] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, 1999,

[2] Zdunek R., Optimization Methods - lecture slides.

[3] Mathworks webpage, "Unconstrained Optimization Algorithms", https://www.mathworks.com/help/optim/ug/unconstrained-nonlinear-optimization-algorithms.html

[4] Nonlinear Programming course webpage, North Carolina State University, http://www4.ncsu.edu/ kksivara/ma706/