Optimización Tarea 8

Francisco Javier Peralta Ramírez

23 de abril de 2018

Resumen

En está tarea se implementan dos métodos para resolver sistemas de ecuaciones no lineales. El primer método es el de Newton y el segundo el de Broyden. Se probaron ambos métodos para la misma función, para la cual se proporcina el *Jacobiano* y se muestran sus resultados.

1. Introducción

Representamos nuestro sistema de ecuaciones no lineales como una función $F : \mathbb{R}^n \to \mathbb{R}^n$ y su función Jacobiana $J : \mathbb{R}^n \to \mathbb{R}^{n\times n}$.

La función F está dada por:

$$
\begin{aligned}
F(\boldsymbol{x}) &= (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}))^T \\
f_1(\boldsymbol{x}) &= 3x_1 - \cos(x_2x_3) - 0.5 \\
f_2(\boldsymbol{x}) &= x_1^2 - 81(x_2+0.1)^2 + \sin(x_3) + 1.06 \\
f_3(\boldsymbol{x}) &= \exp(-x_1x_2) + 20x_3 + \frac{10\pi - 3}{3}
\end{aligned}
$$

Y la función Jacobiano por

$$
\begin{aligned}
J(\boldsymbol{x}) &= (\nabla f_1(\boldsymbol{x}), \nabla f_2(\boldsymbol{x}), \nabla f_3(\boldsymbol{x}))^T \\
\nabla f_1(\boldsymbol{x}) &= \begin{pmatrix} 3 \\ x_3\sin(x_2x_3) \\ x_2\sin(x_2x_3) \end{pmatrix} \\
\nabla f_2(\boldsymbol{x}) &= \begin{pmatrix} 2x_1 \\ -162(x_2+0.1) \\ \cos(x_3) \end{pmatrix} \\
\nabla f_3(\boldsymbol{x}) &= \begin{pmatrix} -x_2\exp(-x_1x_2) \\ -x_1\exp(-x_1x_2) \\ 20 \end{pmatrix}
\end{aligned}
$$

Se probará con cuatro diferentes puntos iniciales:

$$
\begin{aligned}
&(0,0,0)^T \\
&(1.1, 1.1, -1.1)^T \\
&(-10, -10, 10)^T \\
&(3, -3, 3)^T
\end{aligned}
$$

2. Método de Newton

El método de consiste en resolver el sistema de equaciones dado por $J(x_k)p_k = -F(x_k)$ donde $J(\cdot)$ es el Jacobiano de la función y $F(\cdot)$ es la evaluación de la función. De esta forma obtenemos un *paso* p_k con el que actualizamos nuestra última solución, de modo que $x_{k+1} = x_k + p_k$. Esté método se puede resumir en

Algorithm 1 Newton no Lienal

```
function NEWTON_NOLINEAL(x, F, J, t)
    while ||F(x)|| > t do
        J(x)p := -F(x)
        x := x + p
    end while
    return x
end function
```

2.1. Resultados

A continuación se muestran los resultados de las corridas con los cuatro puntos iniciales.

```
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 0 1000 0
0          ||F|| = 10.58181        K(J) = 6.66667  x = (0 0 0)
1          ||F|| = 0.2506385       K(J) = 6.6668   x = (0.5 -0.0168888 -0.523599)
2          ||F|| = 0.0280495       K(J) = 6.66632  x = (0.500016 0.00172004 -0.523554)
3          ||F|| = 0.0002355973    K(J) = 6.66636  x = (0.5 1.45705e-05 -0.523598)
4          ||F|| = 1.719378e-08    K(J) = 6.66636  x = (0.5 1.06343e-09 -0.523599)
5          ||F|| = 1.803899e-15    K(J) = 6.66636  x = (0.5 -4.1405e-18 -0.523599)
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 0 1000 1
0          ||F|| = 115.934 K(J) = 64.557   x = (1.1 1.1 -1.1)
1          ||F|| = 28.48806        K(J) = 32.5908  x = (0.69298 0.503875 -0.504961)
2          ||F|| = 6.963824        K(J) = 16.7284  x = (0.501111 0.209889 -0.519456)
3          ||F|| = 1.559717        K(J) = 9.23885  x = (0.500618 0.0711234 -0.521741)
4          ||F|| = 0.2569837       K(J) = 6.66604  x = (0.500133 0.0147972 -0.523212)
5          ||F|| = 0.01551951      K(J) = 6.66634  x = (0.500009 0.000955301 -0.523574)
6          ||F|| = 7.322135e-05    K(J) = 6.66636  x = (0.5 4.5286e-06 -0.523599)
7          ||F|| = 1.661092e-09    K(J) = 6.66636  x = (0.5 1.02738e-10 -0.523599)
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 0 1000 2
0          ||F|| = 7841.155        K(J) = 523.555  x = (-10 -10 10)
1          ||F|| = 1565.071        K(J) = 283.739  x = (-25.1453 -5.30702 -0.473599)
2          ||F|| = 274.2438        K(J) = 93.0471  x = (0.209144 -1.94219 -0.473599)
3          ||F|| = 68.33979        K(J) = 49.06    x = (0.435809 -1.0233 -0.567271)
4          ||F|| = 16.86696        K(J) = 25.1944  x = (0.497621 -0.566924 -0.541105)
5          ||F|| = 4.024227        K(J) = 13.1653  x = (0.497069 -0.344029 -0.532521)
6          ||F|| = 0.8381121       K(J) = 7.67539  x = (0.497765 -0.242309 -0.529938)
7          ||F|| = 0.1067614       K(J) = 6.65862  x = (0.498087 -0.206004 -0.528993)
8          ||F|| = 0.003119166     K(J) = 6.65845  x = (0.498143 -0.199799 -0.528831)
9          ||F|| = 3.003173e-06    K(J) = 6.65844  x = (0.498145 -0.199606 -0.528826)
10         ||F|| = 2.794663e-12    K(J) = 6.65844  x = (0.498145 -0.199606 -0.528826)
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 0 1000 3
0          ||F|| = 8200.062        K(J) = 4.10353  x = (3 -3 3)
1          ||F|| = 597.598 K(J) = 444.901  x = (4.06716 -1.6039 -5.80533)
2          ||F|| = 1171.81 K(J) = 70.1609  x = (3.62285 -1.13277 54.9718)
3          ||F|| = 1085.668        K(J) = 5.73846  x = (-9.75682 -0.115804 53.5587)
4          ||F|| = 182.7962        K(J) = 81.9448  x = (-1.8808 1.41689 -0.74605)
5          ||F|| = 106.1273        K(J) = 38.0729  x = (2.66529 0.620511 4.51087)
6          ||F|| = 7.933578        K(J) = 17.6542  x = (0.400049 0.226956 -0.506644)
7          ||F|| = 1.771894        K(J) = 9.64418  x = (0.500549 0.0786332 -0.520927)
8          ||F|| = 0.304446        K(J) = 6.66599  x = (0.500154 0.0173257 -0.523146)
9          ||F|| = 0.02085108      K(J) = 6.66633  x = (0.500012 0.0012814 -0.523565)
10         ||F|| = 0.0001313203    K(J) = 6.66636  x = (0.5 8.12177e-06 -0.523599)
11         ||F|| = 5.342578e-09    K(J) = 6.66636  x = (0.5 3.30436e-10 -0.523599)
```

Como podemos ver en todos los casos converge de manera rápida.

En la siguiente tabla vemos los resultados de la última iteración con la que se cumple el criterio de convergencia.

Punto Inicial	K	$\boldsymbol{x_k}$	$\lvert\nabla f(\boldsymbol{x_k})\rvert$	$K(J(\boldsymbol{x_k}))$
$(0,0,0)^T$	5	$(0.5, 0, -0.523)^T$	1.8E-15	6.66
$(1.1, 1.1, -1.1)^T$	7	$(0.5, 0, -0.523)^T$	1.6E-9	6.66
$(-10, -10, 10)^T$	10	$(0.5, 0, -0.529)^T$	2.7E-12	6.66
$(3, -3, 3)^T$	11	$(0.5, 0, -0.524)^T$	5.3E-9	6.66

3. Método de Broyden

El método de Broyden es muy similar al de Newton, pero este no necesita calcular la matriz Jacobiana en cada iteración. Se parte de una matriz Jacobiana calculada para el primer prunto x_0 y luego se va actualizando usando el método de la secante, donde $\nabla f(x_k) \approx [f(x_k) - f(x_{k-1})]/(x_k - x_{k-1})$

De esta forma obtenemos el algirtmo

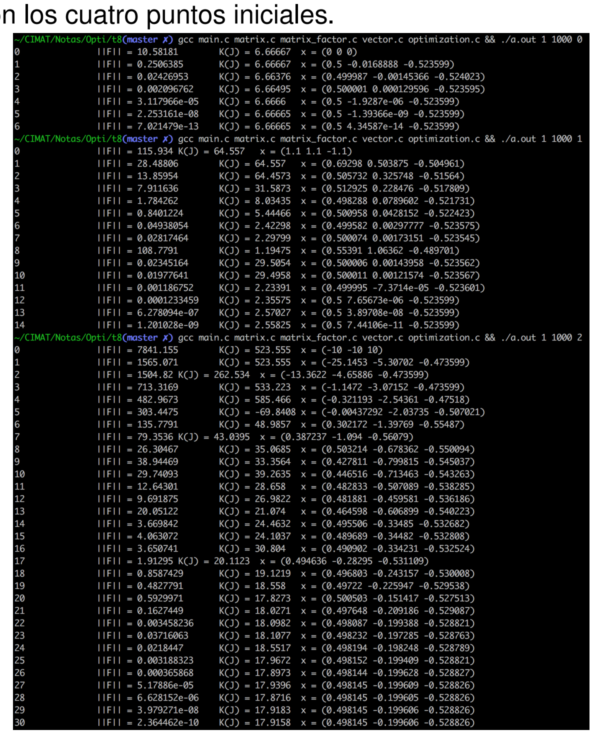
Algorithm 2 Boryden

```
function BROYDEL(x_0, F, A, t)
    k = 0
    while ||F(x)|| > t do
        A_k p_k := -F(x_k)
        x_{k+1} := x_k + p_k
        y_k := F(x_{k+1}) - F(x_k)
        A_k := A_k + ((y_k - A_k p_k) / (p_k^T p_k)) p_k^T
        k := k + 1
    end while
    return x
end function
```

Al utilizar una aproximación en lugar de la Jacobiana tenemos mayor libertad al no depender de una función para calcular continuamente esta matriz, pero a cambio tomamos un camino menos directo a la solución ya que la aproximación pude ser mala.

3.1. Resultados

A continuación se muestran los resultados de las corridas con los cuatro puntos iniciales.



```
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 1 1000 0
0           ||F|| = 10.58181        K(J) = 6.66667  x = (0 0 0)
1           ||F|| = 0.2506385       K(J) = 6.66667  x = (0.5 -0.0168888 -0.523599)
2           ||F|| = 0.02426953      K(J) = 6.66376  x = (0.499987 -0.00145366 -0.524023)
3           ||F|| = 0.002096762     K(J) = 6.66495  x = (0.500001 0.000129596 -0.523595)
4           ||F|| = 3.117966e-05    K(J) = 6.6666   x = (0.5 -1.9287e-06 -0.523599)
5           ||F|| = 2.253161e-08    K(J) = 6.66665  x = (0.5 -1.39366e-09 -0.523599)
6           ||F|| = 7.021479e-13    K(J) = 6.66665  x = (0.5 4.34587e-14 -0.523599)
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 1 1000 1
0           ||F|| = 115.934 K(J) = 64.557   x = (1.1 1.1 -1.1)
1           ||F|| = 28.48806        K(J) = 64.557   x = (0.69298 0.503875 -0.504961)
2           ||F|| = 13.85954        K(J) = 64.4573  x = (0.505732 0.325748 -0.51564)
3           ||F|| = 7.911636        K(J) = 31.5873  x = (0.512925 0.228476 -0.517809)
4           ||F|| = 1.784262        K(J) = 8.03435  x = (0.498288 0.0789602 -0.521731)
5           ||F|| = 0.8401224       K(J) = 5.44466  x = (0.500958 0.0428152 -0.522423)
6           ||F|| = 0.04938054      K(J) = 2.42298  x = (0.499582 0.00297777 -0.523575)
7           ||F|| = 0.02817464      K(J) = 2.29799  x = (0.500074 0.00173151 -0.523545)
8           ||F|| = 108.7791        K(J) = 1.19475  x = (0.55391 1.06362 -0.489701)
9           ||F|| = 0.02345164      K(J) = 29.5054  x = (0.500006 0.00143958 -0.523562)
10          ||F|| = 0.01977641      K(J) = 29.4958  x = (0.500011 0.00121574 -0.523567)
11          ||F|| = 0.001186752     K(J) = 2.23391  x = (0.499995 -7.3714e-05 -0.523601)
12          ||F|| = 0.0001233459    K(J) = 2.35575  x = (0.5 7.65673e-06 -0.523599)
13          ||F|| = 6.278094e-07    K(J) = 2.57027  x = (0.5 3.89708e-08 -0.523599)
14          ||F|| = 1.201028e-09    K(J) = 2.55825  x = (0.5 7.44106e-11 -0.523599)
~/CIMAT/Notas/Opti/t8(master ✗) gcc main.c matrix.c matrix_factor.c vector.c optimization.c && ./a.out 1 1000 2
0           ||F|| = 7841.155        K(J) = 523.555  x = (-10 -10 10)
1           ||F|| = 1565.071        K(J) = 523.555  x = (-25.1453 -5.30702 -0.473599)
2           ||F|| = 1504.82 K(J) = 262.534  x = (-13.3622 -4.65886 -0.473599)
3           ||F|| = 713.3169        K(J) = 533.223  x = (-1.1472 -3.07152 -0.473599)
4           ||F|| = 482.9673        K(J) = 585.466  x = (-0.321193 -2.54361 -0.47518)
5           ||F|| = 303.4475        K(J) = -69.8408 x = (-0.00437292 -2.03735 -0.507021)
6           ||F|| = 135.7791        K(J) = 48.9857  x = (0.302172 -1.39769 -0.55487)
7           ||F|| = 79.3536 K(J) = 43.0395  x = (0.387237 -1.094 -0.56079)
8           ||F|| = 26.30467        K(J) = 35.0685  x = (0.503214 -0.678362 -0.550094)
9           ||F|| = 38.94469        K(J) = 33.3564  x = (0.427811 -0.799815 -0.545037)
10          ||F|| = 29.74093        K(J) = 39.2635  x = (0.446516 -0.713463 -0.543263)
11          ||F|| = 12.64301        K(J) = 28.658   x = (0.482833 -0.507089 -0.538285)
12          ||F|| = 9.691875        K(J) = 26.9822  x = (0.481881 -0.459581 -0.536186)
13          ||F|| = 20.05122        K(J) = 21.074   x = (0.464598 -0.606899 -0.540223)
14          ||F|| = 3.669842        K(J) = 24.4632  x = (0.495506 -0.33485 -0.532682)
15          ||F|| = 4.063072        K(J) = 24.1037  x = (0.489689 -0.34482 -0.532808)
16          ||F|| = 3.650741        K(J) = 30.804   x = (0.490902 -0.334231 -0.532524)
17          ||F|| = 1.91295 K(J) = 20.1123  x = (0.494636 -0.28295 -0.531109)
18          ||F|| = 0.8587429       K(J) = 19.1219  x = (0.496803 -0.243157 -0.530008)
19          ||F|| = 0.4827791       K(J) = 18.558   x = (0.49722 -0.225947 -0.529538)
20          ||F|| = 0.5929971       K(J) = 17.8273  x = (0.500503 -0.151417 -0.527513)
21          ||F|| = 0.1627449       K(J) = 18.0271  x = (0.497648 -0.209186 -0.529087)
22          ||F|| = 0.003458236     K(J) = 18.0982  x = (0.498087 -0.199388 -0.528821)
23          ||F|| = 0.03716063      K(J) = 18.1077  x = (0.498232 -0.197285 -0.528763)
24          ||F|| = 0.0218447       K(J) = 18.5517  x = (0.498194 -0.198248 -0.528789)
25          ||F|| = 0.003188323     K(J) = 17.9672  x = (0.498152 -0.199409 -0.528821)
26          ||F|| = 0.000365868     K(J) = 17.8973  x = (0.498144 -0.199628 -0.528827)
27          ||F|| = 5.17886e-05     K(J) = 17.9396  x = (0.498145 -0.199609 -0.528826)
28          ||F|| = 6.628152e-06    K(J) = 17.8716  x = (0.498145 -0.199605 -0.528826)
29          ||F|| = 3.979271e-08    K(J) = 17.9183  x = (0.498145 -0.199606 -0.528826)
30          ||F|| = 2.364462e-10    K(J) = 17.9158  x = (0.498145 -0.199606 -0.528826)
```

En todos los casos el algoritmo tarda más en converger que Newton, y en el último caso este ni siquiera logra converger, esto se debe a que el condicionamiento de la matriz crece mucho.

Punto Inicial	K	$\boldsymbol{x_k}$	$\|\nabla f(\boldsymbol{x_k})\|$	$K(J(\boldsymbol{x_k}))$
$(0,0,0)^T$	6	$(0.5, 0, -0.524)^T$	7.0E-13	6.66
$(1.1, 1.1, -1.1)^T$	14	$(0.5, 0, -0.524)^T$	1.2E-9	2.55
$(-10, -10, 10)^T$	30	$(0.5, 0, -0.529)^T$	2.4E-10	17.91
$(3, -3, 3)^T$	231	$(0.5, 0, -0.529)^T$	1608	5E96

4. Conclución

Como era de esperarse, el método de Newton es mejor en todos los casos, esto se debe a que cuenta con más información en cada iteración, pero está información tiene un costo y puede haber casos en los que no se cuenta con una función para calcular el Jacobiano. Queda claro entonces que el método de Broyden es util cuando no contamos con el Jacobiano, pero de alguna manera tenemos el Jacobiano para el punto inicial.

Es importante notar que Borydel falla cuando la aproximación a la Jacobiana se mal condiciona, en el caso que no convergió el condicionamiento se dispara y nuestro algoritmo deja ya no logra bajar el número de condicionamiento de la matriz. Si pudieramos evitar el mal condicionamiento de la matriz de aproximación a la Jacobiana podríamos resolver el sistema aún en casos como el del punto $(3, -3, 3)^T$.

5. Ejercicio 5

Dada la función $f : \mathbb{R}^n \to \mathbb{R}$ y el cambio de variable $\hat{\boldsymbol{x}} = \boldsymbol{T}\boldsymbol{x}$, definimos la función

$$\hat{f}(\hat{\boldsymbol{x}}) = f(\boldsymbol{T}^{-1}\hat{\boldsymbol{x}})$$

Usando la regla de la cadena, pruebe que

$$\nabla^2 \hat{f}(\hat{\boldsymbol{x}}) = \boldsymbol{T}^{-T} \nabla^2 f(\boldsymbol{x}) \boldsymbol{T}^{-1}$$

Iniciamos calculando $\nabla \hat{f}(\hat{\boldsymbol{x}})$

$$\frac{\partial \hat{f}(\hat{\boldsymbol{x}})}{\partial \hat{x}_i} = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial \hat{x}_i} \tag{1}$$

Por definición tenemos que

$$x_i = \sum_{j=1}^{n} T^{-1}_{i,j} \hat{x}_j \tag{2}$$

$$\frac{\partial x_i}{\partial \hat{x}_j} = T^{-1}_{i,j} \tag{3}$$

Substituyedo (3) en (1), obtenemos

$$\frac{\partial \hat{f}(\hat{\boldsymbol{x}})}{\partial \hat{x}_i} = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} T^{-1}_{j,i} \tag{4}$$

O de manera equivalente

$$\frac{\partial \hat{f}(\hat{\boldsymbol{x}})}{\partial \hat{x}_i} = \sum_{j=1}^{n} T^{-T}_{i,j} \frac{\partial f}{\partial x_j} \tag{5}$$

$$\nabla \hat{f}(\hat{\boldsymbol{x}}) = T^{-T} \nabla f(\boldsymbol{x}) \tag{6}$$

Para $\nabla^2 \hat{f}(\hat{\boldsymbol{x}})$

$$
\begin{aligned}
\frac{\partial^2 \hat{f}(\hat{\boldsymbol{x}})}{\partial \hat{x}_j \partial \hat{x}_i} &= \frac{\partial}{\partial \hat{x}_j} \left[\sum_{k=1}^{n} \frac{\partial f}{\partial x_k} \frac{\partial x_k}{\partial \hat{x}_i} \right] \\
&= \sum_{k=1}^{n} \left[\sum_{l=1}^{n} \frac{\partial^2 f}{\partial x_l} \frac{\partial x_l}{\partial \hat{x}_j} \right] \frac{\partial x_k}{\partial \hat{x}_i} \\
&= \sum_{k=1}^{n} \left[\sum_{l=1}^{n} \frac{\partial^2 f}{\partial x_l} T^{-1}_{l,j} \right] T^{-1}_{k,i} \\
&= \sum_{k=1}^{n} T^{-T}_{i,j} \left[\sum_{l=1}^{n} \frac{\partial^2 f}{\partial x_l} T^{-1}_{l,j} \right] \\
&= \sum_{k=1}^{n} T^{-T}_{i,j} \nabla^2 f(\boldsymbol{x}) T^{-1} \\
&= T^{-T} \nabla^2 f(\boldsymbol{x}) T^{-1}
\end{aligned}
$$