# Proyecto II

Análisis Aplicado I

**Eric Bazaldua Miñana 155279**
**Hugo Delgado Curiel 155483**
**Adolfo Margain Orozco 157264**
**Francisco Gabriel Huerta Fernández 166040**

17 de mayo de 2020

# Índice

**1. Introducción** **2**

**2. Simulaciones con la función de Rosenbrock** **3**

**3. Simulación con la función DIXMAANG** **9**

**4. Conclusión** **9**

# 1. Introducción

Para este proyecto se programaron las siguientes funciones en MATLAB:

- punto de Cauchy (*pCauchy.m*),
- método de región de confianza con actualizaciones simétricas de rango 1 (*rcSR1.m*),
- algoritmo de búsqueda en línea (*lineSearch.m*),
- método BFGS con búsqueda en línea (*lsBFGS.m*),
- método BFGS con memoria limitada y búsqueda en línea (*lsBFGSLiMem.m*).

Nota: las funciones *lsBFGS.m* y *lsBFGSLiMem.m* fueron adaptadas de las programadas el día 28 de abril para encontrar $\alpha_k$ con la función *lineSearch.m*.

El objetivo es implementar el método de región de confianza con actualizaciones simétricas de rango 1, el método BFGS con búsqueda en línea y el método BFGS con memoria limitada y búsqueda en línea a la función de Rosenbrock extendida para diferentes valores de $n$; y también implementar el método BFGS con memoria limitada y búsqueda en línea a la función Dixmaan (con parámetros G en nuestro caso) para diferentes valores de $n$ y $m$.

La función de Rosenbrock extendida está dada por la siguiente expresión:

$$f(x) = \sum_{i=1}^{n/2} c(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \tag{1.1}$$

donde $c = 100$ y con punto inicial $x_0 = (-1.2, 1, \ldots, -1.2, 1)$.
Las funciones Dixmaan están dadas por la siguiente expresión:

$$f(x) = 1 + \sum_{i=1}^{n} \alpha x_i^2 \left(\frac{i}{n}\right)^{k1} + \sum_{i=1}^{n-1} \beta x_i^2 (x_{i+1} + x_{i+1}^2)^2 \left(\frac{i}{n}\right)^{k2} + \sum_{i=1}^{2m} \gamma x_i^2 x_{i+m}^4 \left(\frac{i}{n}\right)^{k3} + \sum_{i=1}^{m} \delta x_i x_{i+2m} \left(\frac{i}{n}\right)^{k4} \tag{1.2}$$

donde $m = n/3$ y con punto inicial $x_0 = (2, 2, \ldots, 2)$ En nuestro caso, usamos la función DixmaanG (es decir, los parámetros definidos por la letra G) donde $\alpha = 1$, $\beta = 0.125$, $\gamma = 0.125$, $\delta = 0.125$, $k1 = 1$, $k2 = 0$, $k3 = 0$, $k4 = 1$.

Para todos los códigos se utilizaron como parámetros generales:

$$c_1 = 10^{-4},\ c_2 = 0.99,\ tol = 10^{-5},\ \eta = 0.1,\ \Delta_{max} = 1.25. \tag{1.3}$$

## 2. Simulaciones con la función de Rosenbrock

Aplicamos los tres métodos a la función para $n \in \{2, 8, 32, 128\}$. Para cada método, se presenta una tabla que contiene las últimas cinco iteraciones (para cada $n$), los valores $||\nabla f(x_k)||_2$, $f(x_k)$, los errores $||x_k - x^*||_2$ y el tiempo. Las tablas presentadas a continuación (figuras 1 - 16) se encuentran en el script *Ej2_ 2.m*

Nota: puede tardar entre uno o dos minutos correr el script. Si se planea replicar los resultados presentados a continuación con los códigos adjuntos hay que tomar en cuenta que el tiempo para calcular cada procedimiento dependerá de las especificaciones y del uso actual de la computadora, por lo cual pueden ser no replicables exactamente, pero en general se obtendrán tiempos similares.

**a) ¿En términos de tiempo, cuál método conviene más?**
A simple vista podemos conjeturar que, en general, todos los métodos para una $n$ pequeña, en nuestro caso para $n = 2$ y $n = 8$, no cambiará mucho en cuestión de cuál escoger. Sin embargo, para una $n$ un poco más grande, en nuestro caso para $n = 32$ y $n = 128$, sí hay un cambio y muestra un incremento en el tiempo para el método BFGS con búsqueda en línea. Por lo cual para ese caso podemos asegurar que el método BFGS con búsqueda en línea no es bueno.

Ahora obtenemos el promedio del tiempo para cada procedimiento utilizando las tablas obtenidas con las últimas 5 iteraciones:

| Valor de n | Método | Tiempo Promedio | Iteraciones |
|---|---|---|---|
| 2 | rcSR1 | $5.020e-5$ | 19 |
| 2 | Line Search Memoria Limitada | $7.7020e-5$ | 34 |
| 2 | Line Search | $6.5560e-5$ | 33 |
| 8 | rcSR1 | $1.41e-2$ | 69 |
| 8 | Line Search Memoria Limitada | $1.2580e-2$ | 48 |
| 8 | Line Search | $2.1360e-2$ | 85 |
| 32 | rcSR1 | $7.926e-2$ | 175 |
| 32 | Line Search Memoria Limitada | $5.0180e-2$ | 86 |
| 32 | Line Search | $1.0380e-1$ | 198 |
| 128 | rcSR1 | $4.5640e-1$ | 240 |
| 128 | Line Search Memoria Limitada | $4.6080e-1$ | 84 |
| 128 | Line Search | 1.57 | 499 |

Tabla 1: Promedio de los últimos cinco tiempos de cada método para cada valor de $n$ y sus respectivas iteraciones.

Con esta tabla podemos concluir que:

- para $n = 2$ el mejor método sería el rcSR1,
- para $n = 8$ el mejor método sería el Line Search con Memoria Limitada,
- para $n = 32$ el mejor método sería el Line Search con Memoria Limitada,

- para $n = 128$ el mejor método sería el rcSR1.

Sin embargo, hay que notar que las diferencias no son muy grandes, por lo cual el resultado no es absoluto pues utilizar una máquina con mayor capacidad puede voltear las cosas, pero esta prueba nos muestra de manera muy clara que el Line Search es el peor de los métodos en cuestión de tiempo.

**b) ¿En términos de iteraciones, cuál método conviene más?**
Utilizando la información de la tabla 1 podemos ver cuál conviene más en cuestión de iteraciones:

- para $n = 2$ el mejor método sería el rcSR1,
- para $n = 8$ el mejor método sería el Line Search con Memoria Limitada,
- para $n = 32$ el mejor método sería el Line Search con Memoria Limitada,
- para $n = 128$ el mejor método sería el Line Search con Memoria Limitada.

En este caso es un poco más claro que esos métodos son los mejores y además podemos observar que también en cuestión de iteraciones el Line Search será el peor de los tres métodos.

**c) ¿Puede relacionar el crecimiento del número de iteraciones (o del tiempo) con el de $n$ ?**
Con los resultados obtenidos es bastante complejo observar algún patrón hacia el número de iteraciones y aún más sobre el tiempo. Sin embargo, podemos observar que tanto Line Search como rcSR1 tienden a incrementar el número de iteraciones conforme $n$ aumenta, pero no se ve algún posible patrón de cómo aumenta a partir del $n$. Ahora bien, por otro lado el Line Search con memoria limitada es más interesante, porque si bien en los primeros tres valores de $n$ sí van aumentando, pero al momento de llegar a $n = 128$ disminuye el número de iteraciones, por lo que no hay forma correcta de conjeturar alguna posible relación.

En términos del tiempo, ocurre lo mismo dicho anteriormente, pero como se menciono en la nota al inicio de esta sección esos tiempos puede que para algunas computadoras se logren ver algún patrón, pero sería muy complicado decir que en general sucede algo.

| | | Función Rosenbrock |
|---|---|---|
| Método | Iteraciones | Tamaño de n |
| rcSR1 | 19 | 2 |
| Line Search LM | 34 | 2 |
| Line Search | 33 | 2 |

Figura 1: Resumen de iteraciones por método.

Tabla para rcSR1 para n=2

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 15 | 6.331192e-03 | 2.279264e-05 | 1.066099e-02 | 8.5252e-03 |
| 16 | 1.385092e-02 | 9.773785e-08 | 1.001288e-04 | 3.9711e-03 |
| 17 | 2.655668e-03 | 5.482237e-09 | 9.915747e-05 | 7.5639e-03 |
| 18 | 3.878283e-05 | 1.885527e-09 | 9.717200e-05 | 3.2536e-03 |
| 19 | 3.478916e-06 | 6.999692e-12 | 5.919332e-06 | 2.0625e-03 |

Figura 2: Tabla para rcSR1 para $n = 2$.

Tabla para Line Search Memoria Limitada para n=2

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 30 | 5.587682e-02 | 3.339242e-05 | 1.259786e-02 | 7.3697e-03 |
| 31 | 4.437723e-02 | 1.337054e-06 | 1.330459e-03 | 8.5916e-03 |
| 32 | 1.946428e-02 | 1.966543e-07 | 1.947513e-04 | 9.5600e-03 |
| 33 | 6.284128e-05 | 1.333972e-11 | 7.548277e-06 | 6.7590e-03 |
| 34 | 1.477557e-07 | 2.543210e-15 | 1.124161e-07 | 6.2519e-03 |

Figura 3: Tabla para Line Search BFGS con Memoria Limitada para $n = 2$.

Tabla para Line Search para n=2

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 29 | 1.470641e-01 | 2.003460e-04 | 3.062765e-02 | 3.7094e-03 |
| 30 | 1.804506e-01 | 1.650286e-05 | 1.109202e-03 | 5.0766e-03 |
| 31 | 1.689127e-02 | 7.720899e-07 | 1.775392e-03 | 8.1063e-03 |
| 32 | 1.850000e-04 | 4.766979e-09 | 1.542586e-04 | 8.9115e-03 |
| 33 | 7.735030e-06 | 3.404000e-13 | 1.248112e-06 | 7.0021e-03 |

Figura 4: Tabla para Line Search BFGS para $n = 2$.

| | | Función Rosenbrock |
|---|---|---|
| Método | Iteraciones | Tamaño de n |
| rcSR1 | 69 | 8 |
| Line Search LM | 48 | 8 |
| Line Search | 85 | 8 |

Figura 5: Resumen de iteraciones por método.

Tabla para rcSR1 para n=8

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 65 | 6.945594e-04 | 7.142150e-08 | 5.972106e-04 | 1.3037e-02 |
| 66 | 2.657351e-04 | 7.114902e-08 | 5.969293e-04 | 1.3863e-02 |
| 67 | 5.406964e-04 | 7.096891e-08 | 5.957078e-04 | 1.4028e-02 |
| 68 | 3.311509e-05 | 5.725174e-13 | 3.699975e-07 | 1.5810e-02 |
| 69 | 5.642949e-06 | 2.312110e-14 | 1.939356e-07 | 1.3940e-02 |

Figura 6: Tabla para rcSR1 para $n = 8$.

Tabla para Line Search Memoria Limitada para n=8

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 44 | 5.483091e-05 | 1.571849e-11 | 8.438955e-06 | 1.1576e-02 |
| 45 | 4.865327e-05 | 6.235604e-12 | 5.029767e-06 | 1.3030e-02 |
| 46 | 4.981574e-05 | 1.251675e-12 | 2.423581e-07 | 1.2560e-02 |
| 47 | 1.310240e-05 | 2.206429e-13 | 8.190803e-07 | 1.2772e-02 |
| 48 | 1.954174e-06 | 4.706269e-15 | 1.218786e-07 | 1.3245e-02 |

Figura 7: Tabla para Line Search BFGS con Memoria Limitada para $n = 8$.

Tabla para Line Search para n=8

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 81 | 1.256078e-02 | 2.250895e-07 | 8.564344e-04 | 2.0428e-02 |
| 82 | 6.261258e-04 | 6.576021e-09 | 1.787934e-04 | 2.1469e-02 |
| 83 | 1.408302e-04 | 6.777834e-11 | 1.702688e-05 | 1.8004e-02 |
| 84 | 1.696270e-05 | 3.578286e-13 | 1.032444e-06 | 2.5714e-02 |
| 85 | 3.857959e-06 | 2.514703e-14 | 2.974192e-07 | 2.1311e-02 |

Figura 8: Tabla para Line Search BFGS para $n = 8$.

| | | Función Rosenbrock |
|---|---|---|
| Método | Iteraciones | Tamaño de n |
| rcSR1 | 175 | 32 |
| Line Search LM | 86 | 32 |
| Line Search | 198 | 32 |

Figura 9: Resumen de iteraciones por método.

Tabla para rcSR1 para n=32

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 171 | 6.522517e-03 | 2.981426e-08 | 2.074037e-04 | 9.4104e-02 |
| 172 | 8.579576e-05 | 2.925999e-09 | 1.209993e-04 | 8.2482e-02 |
| 173 | 6.303767e-05 | 2.920637e-09 | 1.209234e-04 | 7.9363e-02 |
| 174 | 6.023408e-05 | 2.916075e-09 | 1.208323e-04 | 7.3626e-02 |
| 175 | 5.128286e-07 | 1.391793e-14 | 2.632299e-07 | 6.6409e-02 |

Figura 10: Tabla para rcSR1 para $n = 32$.

Tabla para Line Search Memoria Limitada para n=32

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 82 | 9.762160e-05 | 4.782900e-11 | 1.469266e-05 | 4.3975e-02 |
| 83 | 1.072744e-04 | 3.507057e-11 | 1.212484e-05 | 4.7335e-02 |
| 84 | 8.628430e-05 | 3.217797e-11 | 1.194458e-05 | 4.3250e-02 |
| 85 | 2.085272e-05 | 2.600377e-11 | 1.136693e-05 | 4.6976e-02 |
| 86 | 9.447230e-06 | 2.572144e-11 | 1.134164e-05 | 6.9613e-02 |

Figura 11: Tabla para Line Search BFGS con Memoria Limitada para $n = 32$.

Tabla para Line Search para n=32

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 194 | 8.175871e-05 | 8.315651e-10 | 6.441607e-05 | 1.2061e-01 |
| 195 | 9.235555e-05 | 5.338488e-10 | 5.150983e-05 | 1.2263e-01 |
| 196 | 7.426674e-05 | 2.281573e-10 | 3.360483e-05 | 9.6305e-02 |
| 197 | 2.694693e-05 | 8.833925e-11 | 2.099445e-05 | 9.7663e-02 |
| 198 | 9.124669e-06 | 6.633709e-11 | 1.822473e-05 | 1.0155e-01 |

Figura 12: Tabla para Line Search BFGS para $n = 32$.

| | | Función Rosenbrock |
|---|---|---|
| Método | Iteraciones | Tamaño de n |
| rcSR1 | 240 | 128 |
| Line Search LM | 84 | 128 |
| Line Search | 499 | 128 |

Figura 13: Resumen de iteraciones por método.

Tabla para rcSR1 para n=128

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 236 | 2.158187e-04 | 1.496409e-10 | 2.516722e-05 | 4.3877e-01 |
| 237 | 5.844726e-05 | 1.158877e-10 | 2.391772e-05 | 5.3371e-01 |
| 238 | 5.721343e-05 | 1.158145e-10 | 2.391751e-05 | 4.4757e-01 |
| 239 | 1.112548e-04 | 1.016059e-10 | 2.186611e-05 | 4.5173e-01 |
| 240 | 9.011683e-06 | 9.539533e-11 | 2.185695e-05 | 4.3545e-01 |

Figura 14: Tabla para rcSR1 para $n = 128$.

Tabla para Line Search Memoria Limitada para n=128

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 80 | 9.146942e-05 | 3.399866e-11 | 1.222846e-05 | 1.5305e-01 |
| 81 | 7.884288e-05 | 2.427813e-11 | 1.030716e-05 | 1.5723e-01 |
| 82 | 6.539923e-05 | 2.323877e-11 | 1.028177e-05 | 1.5759e-01 |
| 83 | 1.086727e-05 | 2.118707e-11 | 1.028908e-05 | 1.5956e-01 |
| 84 | 4.672748e-06 | 2.109262e-11 | 1.027737e-05 | 1.5901e-01 |

Figura 15: Tabla para Line Search BFGS con Memoria Limitada para $n = 128$.

Tabla para Line Search para n=128

| Iteración | NormGrad | Evaluada | Error | Tiempo |
|---|---|---|---|---|
| 495 | 1.766879e-05 | 8.541804e-11 | 2.066790e-05 | 1.3337e+00 |
| 496 | 1.602958e-05 | 6.457857e-11 | 1.796928e-05 | 1.6801e+00 |
| 497 | 1.398916e-05 | 4.245457e-11 | 1.456736e-05 | 1.6377e+00 |
| 498 | 1.120470e-05 | 3.176358e-11 | 1.260244e-05 | 1.5343e+00 |
| 499 | 9.976341e-06 | 2.979854e-11 | 1.220826e-05 | 1.5442e+00 |

Figura 16: Tabla para Line Search BFGS para $n = 128$.

# 3. Simulación con la función DIXMAANG

Aplicamos el método BFGS con memoria limitada y búsqueda en línea para $n \in \{240, 960\}$ y para $m \in \{1, 3, 5, 17, 29\}$. A continuación se presenta una tabla que contiene el número de iteraciones, los últimos valores $||\nabla f(x_k)||_2$, $f(x_k)$, y el tiempo para cada combinación de $n$ y $m$. La tabla se generó con el script *Ej2_ 3.m*.

Nota: puede tardar entre tres y cuatro minutos correr el script. Este es el tiempo que le tomaba a nuestras computadoras pero depende de las especificaciones de las mismas.

Función DixmaanG

Tabla para Line Search BFGS con Memoria Limitada para n=240 y n=960

| Valor de n | Valor de m | Iteraciones | NormGrad | Evaluada | Tiempo |
|---|---|---|---|---|---|
| 240 | 1 | 84 | 8.449054e-06 | 1.000000e+00 | 2.1585e+00 |
| 240 | 3 | 81 | 5.868495e-06 | 1.000000e+00 | 2.0215e+00 |
| 240 | 5 | 82 | 7.538434e-06 | 1.000000e+00 | 1.8419e+00 |
| 240 | 17 | 69 | 5.782494e-06 | 1.000000e+00 | 1.5680e+00 |
| 240 | 29 | 67 | 7.280853e-06 | 1.000000e+00 | 1.5795e+00 |
| 960 | 1 | 162 | 8.162790e-06 | 1.000000e+00 | 5.2549e+01 |
| 960 | 3 | 135 | 8.390484e-06 | 1.000000e+00 | 4.3171e+01 |
| 960 | 5 | 134 | 9.633339e-06 | 1.000000e+00 | 4.1924e+01 |
| 960 | 17 | 111 | 9.468450e-06 | 1.000000e+00 | 3.5872e+01 |
| 960 | 29 | 111 | 8.701983e-06 | 1.000000e+00 | 3.5871e+01 |

Figura 17: Tabla para Line Search BFGS con memoria limitada.

De lo resultados de la figura 17 nos dimos cuenta que para cada valor de $n$, al ir aumentando el valor de $m$ el tiempo que tarda en encontrar la solución el algoritmo y el número de iteraciones que realiza disminuye.

# 4. Conclusión

En este proyecto pudimos ver la eficiencia de los métodos Line Search, Line Search con memoria Limitada y rcSR1.

Con la función de Rosenbrock se logró corroborar que el Line Search con memoria limitada es el método más eficiente, ya que utiliza mucha menos memoria y converge mucho más rápido que los otros. Sin embargo, en algunos casos el método rcSR1 dio mejores resultados.

Con la función DixmaanG se logro ver cómo puede ser más eficiente el Line Search con memoria limitada y claramente hay una reducción en el número de iteraciones conforme fuimos incrementando el valor de $m$ (el número de variables guardadas), ya que al momento de tener más información guardada, el código logra una mejor aproximación rápidamente. Aunque exagerar el valor de $m$ puede traer un incremento en tiempo para la computadora, pues el uso de memoria es mucho mayor.

Ambas cosas coinciden con la teoría, concluyendo que en general Line Search con memoria limitada es el método más eficiente para este tipo de trabajos.