Project 2

Pablo López Landeros
178863
Fernando Stein Vallarta
165455
Diego González
167429

Due Date: May 17,2020

Dr. Andreas Wachtel

Introduction

The purpose of this project is to implement the following algorithms and test their reliability when solving unconstrained optimization problems.

- Trust Region Method with symmetric rank 1 updates.
- Line Search method
- BFGS method
- Limited Memory BFGS method

Once implemented, our algorithms will try to solve two aritificial unconstrained optimization test problems: the Dixmaana function and the Extended Rosenbrock function. The Dixmaana function has eight different free parameters. Our team was assigned the **L** set of parameters by the professor. Each method has a different implementation that can be summarized as follows:

- Trust Region Method with symmetric rank 1 updates: The rank-1 update maintains symmetry of the matrix and satisfies the secant equation. The SR1 does not guarantee the positive definiteness.The ability to generate indefinite Hessian approximations is one advantage of the method.
- Line Search Method: In a broad manner, the line search strategy is an algorithm that chooses a direction p_k and searches along this direction from the current iteration x_k for a new iteration with a lower function value.
- BFGS method: The optimization problem is to minimize f, where x is a vector in $\mathbb{R}^n$, and f is a differentiable scalar function. There are no constraints on the values that x can take. The algorithm begins at x_0 and proceeds iteratively to try to reach a better estimate at each stage.The search direction p_k at stage k is given by the solution of the analogue of the Newton equation.
- Limited Memory BFGS method: The algorithm starts with an initial value, x_0, and proceeds iteratively to find a better estimate through a sequence of better estimates $x_1, x_2 \ldots$ The derivatives of the function $g_k := \nabla f(x_k)$ are used as key to identify the direction of steepest descent, and to form an estimate of the Hessian matrix of $f(x)$.

Test Functions

Dixmaana-Dixmaanl functions

As our team was assigned the **L** set of parameters, the Dixmaana function is given by:

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

where:

$$
\begin{aligned}
m &= n/3 \\
\alpha &= 1 \\
\beta &= 0.26 \\
\gamma &= 0.26 \\
\sigma &= 0.26 \\
k_1 &= 2 \\
k_2 &= 0 \\
k_3 &= 0 \\
k_4 &= 2
\end{aligned}
$$

And our starting point is given by:

$$x_0 = [2, 2, ..., 2]$$

Because m is defined as $\frac{n}{3}$ and then utilized as an index for our x vector in the function, we must note that n (the dimension of our x vector) must be a multiple of three. Otherwise, the indexes for the terms x_{i+2m} and x_{i+m}^4, as well as the upper limit on two of the sums, would be a non-integer number.

Extended Rosenbrock function

The Extended Rosenbrock function is given by the following equation:

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

where:

$$c = 100$$

And our staring point is given by:

$$x_0 = [-1.2, 1, -1.2, ..., 1]$$

Simulations

In order to run the simulations we require the following scripts:

1. *pcauchy.m*
2. *rcSR1.m*
3. *lineSearch.m*
4. *lsBFGS.m*
5. *lineLM_BFGS.m*
6. *apGrad.m*

Simulation with the Extended Rosenbrock Function

We must test out our three main codes (lineLM_BFGS, lsBFGS, rcSR1) with the Extended Rosenbrock function as our objective function. We will do so for $n \in \{2, 8, 32, 128\}$. For each n we will measure $||\nabla f(x_k)||_2$, $f(x_k)$, the error $||x_k - x^*||$ and the execution time. The results can be replicated by running the **Ejercicio2_2** script.

BFGS method
rcSR1 method
Limited Memory BFGS method

- In terms of execution time. Which method performs best?
- Which is the best method if we want the least number of iterations possible?
- Is there a direct relationship between n, the number of iterations and the execution times of the algorithms?

Trying to answer the questions we wrote a Matlab script named **Ejercicio2_2** which presents the following results.

```
------------Resultados del Problema Rosenbrock------------
------------Resultados con SCR y n=2 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 2   49     1.073882e-02     5.133837e-06   5.038410e-03    5.934000e-03

 2   50     4.690883e-02     1.565613e-06   1.533072e-03    4.802600e-03

 2   51     4.880273e-03     1.482286e-08   1.213086e-04    4.719000e-03

 2   52     1.671177e-04     3.414982e-11   1.006118e-05    5.078400e-03

 2   53     5.605332e-07     1.554585e-16   1.164885e-08    5.421400e-03
------------Resultados con lsBFGS y n=2 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 2   29     1.470635e-01     2.003461e-04   3.062767e-02    5.109300e-03

 2   30     1.804512e-01     1.650296e-05   1.109200e-03    3.197400e-03

 2   31     1.689129e-02     7.720936e-07   1.775396e-03    2.941300e-03

 2   32     1.850017e-04     4.767048e-09   1.542597e-04    4.008100e-03

 2   33     7.735128e-06     3.404058e-13   1.248122e-06    3.629300e-03
------------Resultados con lsLM_BFGS y n=2 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 2   30     5.587682e-02     3.339242e-05   1.259786e-02    4.842100e-03

 2   31     4.437723e-02     1.337054e-06   1.330459e-03    5.986700e-03

 2   32     1.946428e-02     1.966543e-07   1.947513e-04    5.074900e-03

 2   33     6.284128e-05     1.333972e-11   7.548277e-06    6.312200e-03

 2   34     1.477557e-07     2.543210e-15   1.124161e-07    8.157000e-03
------------Resultados con SCR y n=8 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 8   119    2.136366e-03     2.157770e-06   3.285032e-03    2.189560e-02

 8   120    8.274661e-03     3.717385e-07   1.300324e-03    2.304860e-02

 8   121    1.340287e-03     1.082690e-09   3.055484e-05    2.275750e-02

 8   122    3.586172e-05     7.444295e-13   7.253680e-07    2.311620e-02

 8   123    9.669313e-07     4.807388e-15   1.462496e-07    2.307430e-02
------------Resultados con lsBFGS y n=8 ------------

```

```
n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

8   77    2.436566e-02     9.556968e-07   1.817906e-03    1.750550e-02

8   78    1.783614e-03     3.592762e-08   4.148025e-04    1.341240e-02

8   79    2.105841e-04     5.001890e-10   4.893871e-05    1.294540e-02

8   80    4.874504e-05     2.808151e-12   2.848699e-06    1.308370e-02

8   81    4.064079e-06     3.580501e-14   3.717778e-07    1.354330e-02
------------Resultados con lsLM_BFGS y n=8 ------------

n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

8   55    2.901700e-05     2.697914e-12   3.380164e-06    1.384900e-02

8   56    2.144549e-05     1.605581e-12   2.627442e-06    1.142140e-02

8   57    3.883064e-05     8.194562e-13   5.843109e-07    1.107100e-02

8   58    2.376171e-05     4.079397e-13   7.915684e-07    1.045160e-02

8   59    3.424112e-07     2.998102e-15   1.205919e-07    1.125090e-02
------------Resultados con SCR y n=32 ------------

n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

32    90    4.660258e-04     9.878012e-09   2.212361e-04    4.950720e-02

32    91    3.952877e-04     9.846370e-09   2.212220e-04    5.093330e-02

32    92    9.056803e-05     9.764289e-09   2.211289e-04    5.397050e-02

32    93    2.692810e-05     1.800197e-11   9.400799e-06    5.716120e-02

32    94    2.956211e-06     2.267319e-14   3.031862e-07    5.512920e-02
------------Resultados con lsBFGS y n=32 ------------

n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

32    181   6.315315e-05     3.546718e-09   1.332624e-04    1.209894e-01

32    182   5.528508e-05     2.240160e-09   1.059031e-04    1.025323e-01

32    183   3.704025e-05     8.512657e-10   6.527946e-05    9.343610e-02

32    184   1.601396e-05     1.850496e-10   3.043777e-05    9.467410e-02

32    185   8.796154e-06     7.431937e-11   1.929109e-05    1.003144e-01
------------Resultados con lsLM_BFGS y n=32 ------------

n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

32    82    1.188136e-05     3.213366e-12   3.966640e-06    5.292140e-02
```

```

 32    83    1.409718e-05     2.432883e-12   3.417549e-06    4.594300e-02

 32    84    6.350781e-05     2.113737e-12   7.483853e-07    4.778080e-02

 32    85    1.193512e-05     1.617535e-13   6.814473e-07    4.619730e-02

 32    86    2.170545e-06     2.038054e-14   3.015623e-07    4.812110e-02
------------Resultados con SCR y n=128 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 128   213   2.451734e-04     3.230149e-11   3.413554e-06    8.776225e-01

 128   214   1.167828e-04     9.131948e-12   3.418664e-06    6.783124e-01

 128   215   3.090244e-05     1.648406e-12   2.422717e-06    6.695967e-01

 128   216   1.400741e-05     1.259176e-12   2.411593e-06    7.869590e-01

 128   217   8.560446e-06     4.981042e-13   1.518918e-06    6.557690e-01
------------Resultados con lsBFGS y n=128 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 128   521   1.639818e-05     4.671679e-11   1.527676e-05    1.814087e+00

 128   522   2.043138e-05     3.840090e-11   1.383277e-05    1.357265e+00

 128   523   2.017345e-05     2.764737e-11   1.172589e-05    1.370283e+00

 128   524   1.268579e-05     2.048246e-11   1.011014e-05    1.201405e+00

 128   525   7.216804e-06     1.863035e-11   9.654409e-06    1.244498e+00
------------Resultados con lsLM_BFGS y n=128 ------------

 n   Iteracion   ||delta(f(xk))||     f(xk)    Error       Tiempo

 128   148   6.983084e-05     6.393092e-11   1.755608e-05    2.847144e-01

 128   149   1.040635e-05     6.112395e-11   1.749134e-05    2.891835e-01

 128   150   1.212340e-05     6.097550e-11   1.746738e-05    3.020333e-01

 128   151   1.544758e-05     6.092971e-11   1.745485e-05    2.900884e-01

 128   152   8.395051e-06     6.078876e-11   1.744625e-05    2.958195e-01
```

NOTE: A computer with 8GB of RAM memory was used to obtain the aforementioned results. The CPU of the Computer was : Intel(R) Core(TM) i7-9300H CPU @ 2.40GHz, 2400 Mhz, with 4 principal and 8 logical components, while de CPU Cache is as follows: L1 = 256 Kb, L2 = 1.0 MB, L3 = 8.0 Mb.

With the previous tables we can then answer the questions that were posed. Firstly, for smaller sizes of n we can see that all methods converge in similar iterations and time spans. However as the size of n gets larger we can state that the BFGS method with Limited Memory converges faster and with less iterations. This result seems to be consistent with the theory since the Limited Memory method uses less memory to find the results. The previous statements allow us to answer both the first and the second question. As far as an answer for the third question, we could say that as n gets larger the number of iterations increases independently of the method used. However, there seems to be no exact relation between the size of n and the time elapsed.

Simulation with the Dixmaana function

For the second simulation, we just need to implement the Limited Memory BFGS method for $n \in \{240, 960\}$ and for each n we use $m \in \{1, 3, 5, 17, 29\}$. For each n dimension we will measure $||\nabla f(x_k)||_2$, $f(x_k)$, the error $||x_k - x^*||$ and the execution time. To run this simulation we used the **EjercicioDixmanna.m** script which outputs the following tables:

```
------------Resultados con n=240 ------------

 m   Iteraciones     ||delta(f(xk))||      f(xk)      Tiempo

 1   488      8.046335e-06     1.000000e+00    1.593685e+02

 3   512      9.266914e-06     1.000000e+00    1.124351e+02

 5   334      9.346795e-06     1.000000e+00    2.162487e+02

 17       326      8.771203e-06     1.000000e+00    1.314825e+02

 29       320      9.003821e-06     1.000000e+00    1.287973e+02
------------Resultados con n=960 ------------

 m   Iteraciones     ||delta(f(xk))||      f(xk)      Tiempo

 1   257      7.030674e-06     1.000000e+00    2.161423e+01

 3   182      9.675509e-06     1.000000e+00    2.253186e+01

 5   347      9.975379e-06     1.000000e+00    1.516246e+01

 17       229      9.720331e-06     1.000000e+00    1.450204e+01

 29       248      9.131316e-06     1.000000e+00    1.413731e+01
```

We must note that the time elapsed in order to present the results with $n = 960$ was approximately 15 minutes. Whereas the with $n = 240$ the elapsed time was around 4 minutes.