SR-1 and Conjugate Gradient Method in MATLAB

HOMEWORK II

Sinan Çavuşoğlu

December 31, 2022

1 Symmetric Rank-1(SR-1) Inverse Update with Armijo Condition

Quasi-Newton methods are methods used to either find zeroes or local maxima and minima of functions, as an alternative to Newton's method. They can be used if the Jacobian or Hessian is unavailable or is too expensive to compute at every iteration.

In the quasi-Newton method, a symmetric rank 1 update is a method for approximating the Hessian matrix, which is the matrix of second-order partial derivatives of a function. The Hessian matrix is used to calculate the search direction for an optimization problem. The update is based on the difference between the current gradient and the gradient at the previous iteration (y_k), and the change in the variables between these two iterations (s_k). In the homework, we used inverse symmetric rank-1 method which has the form:

$$H_k = H_{k-1} + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k} \tag{1}$$

and also used Armijo condition to find step length α:

$$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k \tag{2}$$

We used SR-1 method on Rosenbrock banana function to find local minima which is easily see $x^* = (1,1)^T$ since banana function range is $\mathbb{R}^+ \cup \{0\}$

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{3}$$

We used two different initial guess x_0 and three different tolerances.

```
For the initial value (-0.5,1)
  Tol | iteration   |      x value      |  Norm_Gradient
----- |------------|------------------|---------------

 1e-3 |      5      |    NaN   NaN      |      NaN
 1e-6 |      5      |    NaN   NaN      |      NaN
 1e-9 |      5      |    NaN   NaN      |      NaN
```

We know choice of x_0 is important in Quasi-Newton method so this example show us SR-1 method not always converge to minimizer. To see this, we can look iterates of the minimizer at tolerance 10^{-3}.

```
X1 =

   -0.7871   -0.9182   -0.8194   -0.8194       NaN
    0.7070    0.8259    0.7089    0.7089       NaN
```

Every columns show us how initial value changes and we can see it is not closed to $(1, 1)^T$ so not converge. For another initial value;

```
For the initial value (1.1,1.1)
  Tol | iteration  |                 x value                  | Norm_Gradient
----- |------------|----------------------------------------|-----------------
 1e-3 |      9     |    0.999999191805    0.999998507599     |  0.000056899591
 1e-6 |     11     |    1.000000000011    1.000000000022     |  0.000000000219
 1e-9 |     11     |    1.000000000011    1.000000000022     |  0.000000000219
```

The whole different tolerances, initial values goes to local minimizer of function f. We see this on the norm gradient. For tolerance 10^{-6},

```
Grad5 =

    6.2657    3.2752    0.6666    0.0911    0.3208    0.1769    0.0261    0.0014    0.0001    0.0000    0.0000
```

Last columns show us norm of gradient Rosenbrock banana function goes to zero. This means, we found local minimizer of function.

2 Conjugate Gradient Method

The conjugate gradient method is an iterative method for solving systems of linear equations, particularly those that are symmetric and positive definite.

$$Ax = b, \tag{4}$$

where A is an n $\times$ n matrix that is symmetric and positive definite. The problem can be stated equivalently as the following minimization problem:

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x, \tag{5}$$

Both equation (4) and (5) have the same unique solution. Also, define gradient of equals the residual of the linear system,

$$\bigtriangledown\phi(x) = Ax - b = r(x) \tag{6}$$

In the homework, A_{nxn} is Hilbert matrix s.t. $A_{i,j} = \frac{1}{i+j-1}$, $b = (1, 1, ..., 1)^T$, and initial guess $x_0 = 0$
For n = 5;

```
 n | ite |                  x value                 |  Norm_of_residual
---|-----|----------------------------------------|------------------
 5 |   6 |5.000 -120.000 630.000 -1120.000 630.000| 0.000000074570
```

Norm of residual is closed to 0.

x value is satisfy the equation s.t.

```
>> (A*X5(:,end))'

ans =

    1.0000    1.0000    1.0000    1.0000    1.0000
```

For n = 8;

```
 n | ite |                              x value                               |  Norm_of_residual
---|-----|--------------------------------------------------------------------|------------------
 8 |  19 |-8.00 504.00 -7560.01 46200.01 -138600.00 216215.99 -168168.01 51480.01| 0.000000028362
```

Also, X value satisfy the equation with 8x8 Hilbert matrix

```
>> (hilb(8)*X8(:,end))'

ans =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

For n = 12, with iteration 37

```
>> X12(:,end)'

ans =

   1.0e+06 *

   -0.0000    0.0008   -0.0165    0.1355   -0.5365    1.0254   -0.6426   -0.6576    0.8042    0.6631   -1.2413    0.4655
```

For n = 20, with iteration 74;

```
>> (X20(:,end))'

ans =

   1.0e+06 *

  Columns 1 through 13

   -0.0000    0.0011   -0.0240    0.2204   -0.9653    1.9901   -1.2527   -1.3435    0.8832    1.6880    0.3882   -1.3055   -1.7106

  Columns 14 through 20

   -0.5283    1.2087    2.0029    0.9446   -1.4340   -2.6509    1.8878
```