AMS230 – Homework #2

Zayd Hammoudeh

April 29, 2018

Name: Zayd Hammoudeh
Course Name: AMS230
Assignment Name: Homework #2
Due Date: May 4, 2018
Student Discussions: I discussed the problems with the following students. All write-ups were prepared separately and independently.

- Ben Sherman

Exercise #1
Exercise 5.2 in Nocedal and Wright.

Show that if the nonzero vectors $p_0, p_1, \ldots, p_l$ satisfy

$$p_i^{\mathrm{T}} A p_j = 0, \text{ for all } i \neq j,$$

where A is symmetric and positive definite, then these vectors are linearly independent. (This result implies that A has at most n conjugate directions.)

Exercise #2
Exercise 5.6 in Nocedal and Wright.

Show that

$$\beta_{k+1} = \frac{r_{k+1}^{\mathrm{T}} r_{k+1}}{r_k^{\mathrm{T}} r_k}$$

is equivalent to

$$\beta_{k+1} = \frac{r_{k+1}^{\mathrm{T}} A p_k}{p_k^{\mathrm{T}} A p_k}.$$

Exercise #3

In this problem we will test the performance of linear conjugate gradient method for minimizing quadratic function $f(x) = \frac{1}{2}x^{\mathrm{T}}Ax - b^{\mathrm{T}}x$, where A is symmetric and positive definite.

- **Program CG Algorithm 5.2 in the textbook.**
- **Apply your program on a symmetric and positive definite matrix A of dimension $10^3 \times 10^3$ with eigenvalues uniformly distributed between 10 and 10^3. Test the convergence of the algorithm and compare your numerical findings with the theoretical result shown in formula (5.36) in the textbook.**
- **Change the distribution of eigenvalues of A so that some eigenvalues are distributed between 9 and 11, the rest of eigenvalues are distributed between 999 and 1001, i.e., two clusters around 10 and 1000 with radius 1. Test the convergence performance on such matrix. Extra points if you can explain your numerical findings using the theoretical convergence results discussed in the lecture.**

The Python source code for the implementation of the linear conjugate gradient method as detailed in Algorithm 5.2 is included at the end of this submission. The specific parameters used are enumerated in Table **??**. Distribution $\mathcal{U}(a, b)$ indicates a uniform random variable drawn from the range $[a, b)$. Note that for all experiments in this program problem, the same random x_0 vector was used to ensure consistency.

Table 1: Parameters for the linear conjugate gradient experiments of problem #3

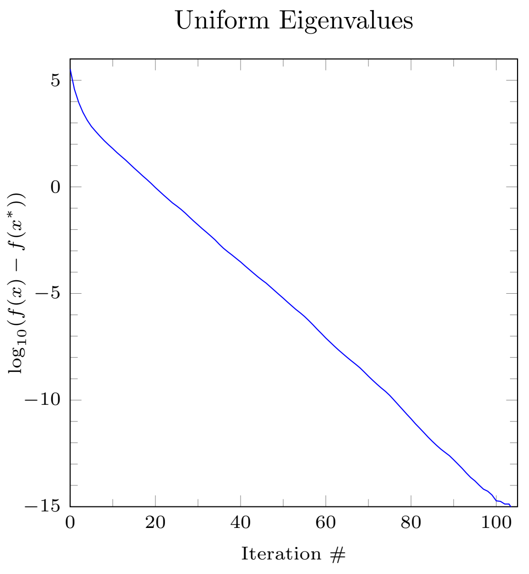
Parameter	**Value**
n	1,000
x_0	$2 \cdot \text{rand}(n, 1)$
Uniform	$\mathcal{U}(10, 1000)$
Bimodal	10% from $\mathcal{U}(9, 10)$ and 90% from $\mathcal{U}(999, 1001)$

Figures 1a and 1b show the performance of the linear conjugate gradient with n uniform and n bimodal eigenvalues respectively. Similar to the results shown in Figure 5.4 of Nocedal and Wright, matrix A with uniform eigenvalues takes longer to converge. For matrix A with bimodal eigenvalues, it took about 15 steps to fully converge. Considering that there were approximately 10 eigenvalues from the distribution $\mathcal{U}(9, 10)$, these findings align with the theoretical explanation in Nocedal and Wright.

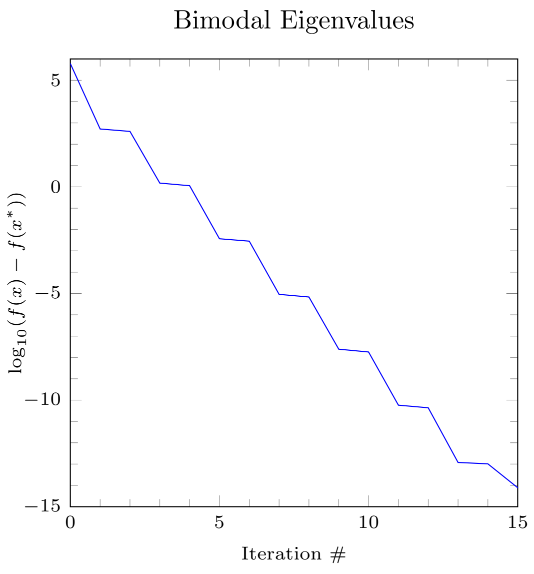
Figures 2a and 2b show the corresponding weighted error versus the calculated upper bound for the uniform and bimodal eigenvalues respectively. These results are based on Eq. (5.36) which states that:

$$\|x_k - x^*\|_A \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k \|x_0 - x^*\|_A$$

where $\kappa(A) = \lambda_n / \lambda_1$. Observe that the upper bound tracks well for the uniform eigenvalues. In contrast, the upper bound is quite loose when the eigenvalues are clustered bimodally.

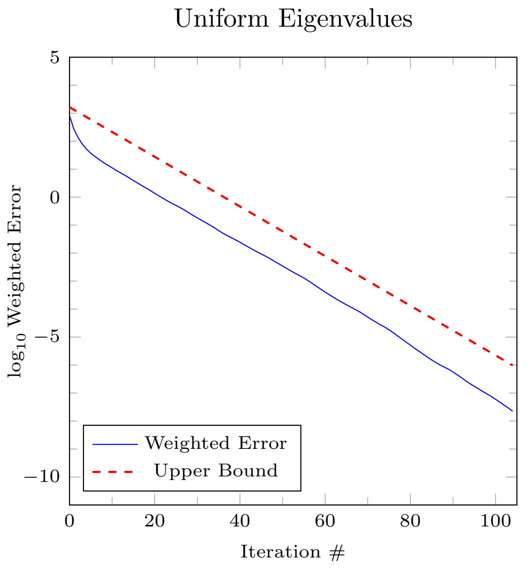


(a) 1,000 λ from the distribution $\mathcal{U}(10, 10^3)$

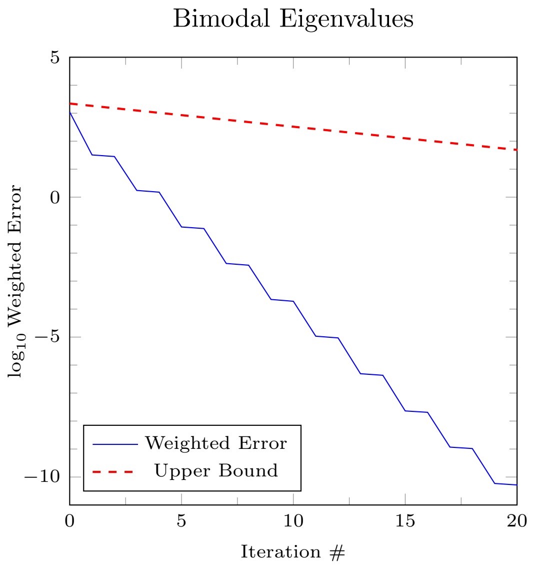


(b) 1,000 λ from distributions $\mathcal{U}(9, 11)$ and $\mathcal{U}(999, 1001)$

Figure 1: Cost function f for problem #3



(a) Weighted error for 1,000 uniform eigenvalues



(b) Weighted error for 1,000 bimodal eigenvalues

Figure 2: Weighted error results for problem #3

Exercise #4
Consider the problem of minimizing

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2].$$

The global minimum is at $x = [1, 1, \cdots, 1]$. Numerically solve this problem using nonlinear conjugate gradient algorithms:

1. **FR (Algorithm 5.4)**
2. **FR with restart based on (5.52)**
3. **PR based on (5.44)**

and compare their performance. (In the numerical experiments you can set $n = 100$ or any number that is not too small. The initial condition can be chosen as a random vector, for example, $2\text{rand}(n, 1)$.)

As their names indicate, the three non-linear conjugate gradient methods rely on the calculation of the gradient. The gradient of f is:

$$\nabla f = \begin{cases} 400x_i(x_i^2 - x_{i+1}) + 2(x_i - 1), & i = 1 \\ 400x_i(x_i^2 - x_{i+1}) + 2(x_i - 1) - 200(x_{i-1}^2 - x_i), & 1 < i < n \\ -200(x_{i-1}^2 - x_i), & i = n \end{cases}.$$

For the inexact line search, define $\psi_i = x_i + \alpha p_i$. Therefore, the derivative of ϕ is

$$\phi' = \sum_{i=1}^{n-1} 200(2p_i\psi_i - p_{i+1})(\psi_i^2 - \psi_{i+1}) + 2p_i(\psi_i - 1).$$

The parameters used for this experiment are specified in Table 2.

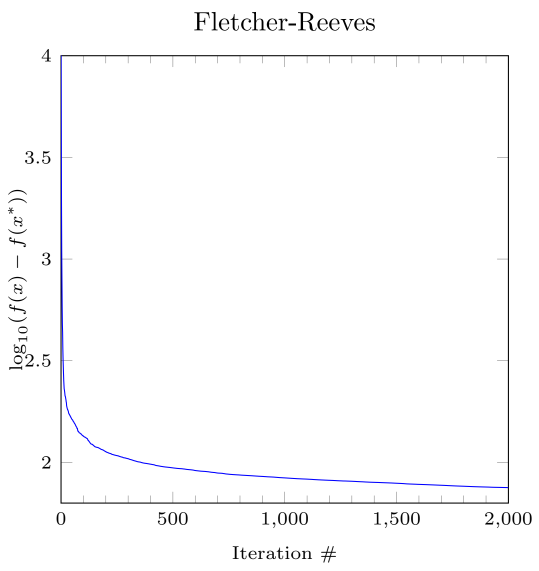
Table 2: Experiment parameters for problem #4

Parameter	Value
n	100
x_0	$2 \cdot \text{rand}(n, 1)$

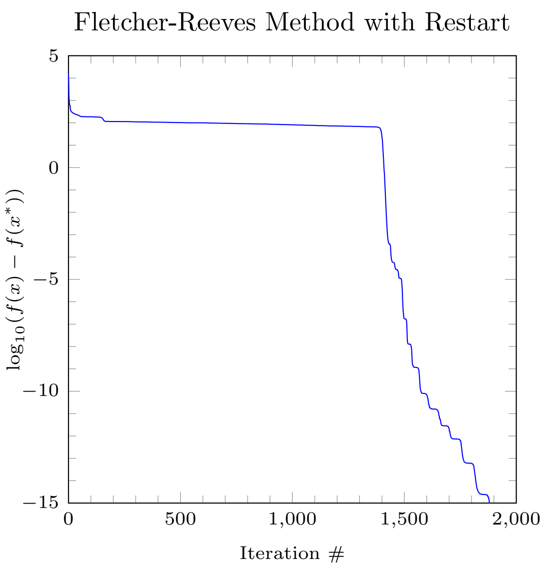
Figures 3a, 3b, and 4 show the performance of Fletcher-Reeves, Fletcher-Reeves with Restart, and Polak-Ribière respectively. In all experiments, the same random x_0 was used.

As expected, standard Fletcher-Reeves had the worst performance. Between 500 and 2,000 iterations, the algorithm made barely any progress and never came close to converging to the optimal solution. In Fletcher-Reeves with restart, the algorithm behaves similar to without restart for many iterations. Similar to what is explained in Nocedal, the algorithm then begins to enter a region (e.g., iteration 1,400) where the error decreases rapidly. Across multiple trials with different random x_0, this avalanche point shifted between around 50 to over 2,000 iterations.

Generally, across different random x_0, the Polak-Ribière method converged the fastest of the three methods. It was however only marginal better than Fletcher-Reeves with restart and still exhibited the same avalanche behavior.



(a)



(b)

Figure 3: Problem #4 performance for Fletcher-Reeves without and with restart

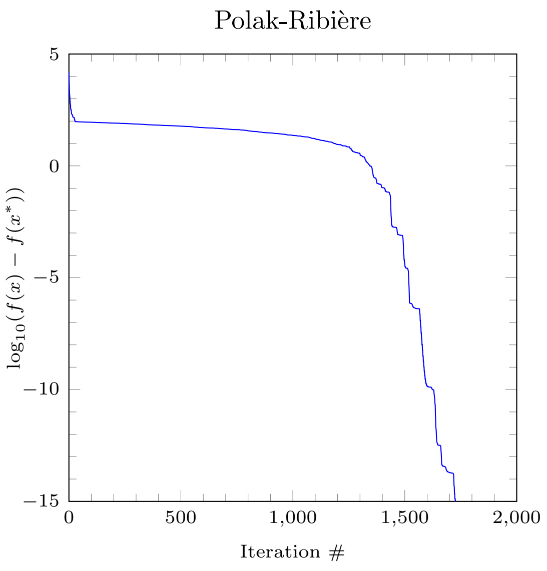


Figure 4: Problem #4 performance for the Polak-Ribière Method