

AMS230 – Homework #2

Zayd Hammoudeh

May 3, 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
- Bernardo Torres

Exercise #1

Exercise 5.2 in Nocedal and Wright.

Show that if the nonzero vectors p_0, p_1, \dots, p_l satisfy Eq. (5.5) from Nocedal and Wright that

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

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

Proof. By contradiction.

Assume vectors $\{p_0, p_1, \dots, p_l\}$ are A -orthogonal but are not linearly independent. That means there is a set constants $\{\alpha_0, \alpha_1, \dots, \alpha_l\}$ not all equal to zero such that:

$$\alpha_0 p_0 + \alpha_1 p_1 + \dots + \alpha_l p_l = \vec{0}.$$

For some $0 \leq i \leq l$ where $\alpha_i \neq 0$, multiply both sides by $A p_i$ yielding:

$$\alpha_0 p_0^T A p_i + \alpha_1 p_1^T A p_i + \dots + \alpha_l p_l^T A p_i = \vec{0}^T A p_i.$$

Since p_i and p_j are A -orthogonal for $i \neq j$, this simplifies to:

$$\alpha_i p_i^T A p_i = 0.$$

This is a contradiction since $\alpha_i \neq 0$ and A is symmetric, positive definite, i.e., $x^T A x > 0$ for all non-zero x . \square

Exercise #2

Exercise 5.6 in Nocedal and Wright.

Show that Eq. (5.24d)

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

is equivalent to Eq. (5.14d)

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

Below is the extension of Eq. (5.24d) to Eq. (5.14d) using the equations and theorems in Nocedal and Wright.

Proof.

$$\begin{aligned} \beta_{k+1} &= \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} && \text{Nocedal \& Wright (5.24d)} \\ &= \frac{r_{k+1}^T r_{k+1}}{\alpha_k p_k^T A p_k} && \text{By Nocedal \& Wright (5.24a)} \\ &= \frac{r_{k+1}^T (r_k + \alpha_k A p_k)}{\alpha_k p_k^T A p_k} && \text{By Nocedal \& Wright (5.24c)} \\ &= \frac{r_{k+1}^T \alpha_k A p_k}{\alpha_k p_k^T A p_k} && \text{By Theorem 5.3 Eq. (5.16)} \\ &= \boxed{\frac{r_{k+1}^T A p_k}{p_k^T A p_k}} && \text{Cancelling } \alpha_k \end{aligned}$$

□

Exercise #3

In this problem we will test the performance of linear conjugate gradient method for minimizing quadratic function $f(x) = \frac{1}{2}x^T Ax - b^T x$, 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 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 1. $\mathcal{U}(a, b)$ represents a uniform random variable drawn from the range $[a, b]$. Note that for all experiments in this problem, the same random x_0 vector was used to ensure consistency of results.

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

Parameter	Value
n	1,000
b	$\text{rand}(n, 1)$
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 method with n uniform and n bimodal eigenvalues respectively. For matrix A with bimodal eigenvalues, it took about 15 steps to fully converge. When the eigenvalues were uniform, CG took longer to converge. These results align with the examples presented in class. In the bimodal cases, there are essentially two “clusters” of eigenvalues as opposed to the uniform case where the eigenvalue spectrum is significantly more spread.

Figures 2a and 2b show the corresponding A -weighted error versus the calculated upper bound for the uniform and bimodal eigenvalues respectively. These results use Nocedal and Wright 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 $\kappa(A) = \lambda_n/\lambda_1$. For both sets of eigenvalues (i.e., uniform and bimodal), the condition number $\kappa(A) \approx \frac{1000}{10} = 100$. This means that the upper bound estimate will be approximately the same for both

experiments. 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. This is expected given an approximately equivalent upper bound and the large difference in Figures 1a and 1b.

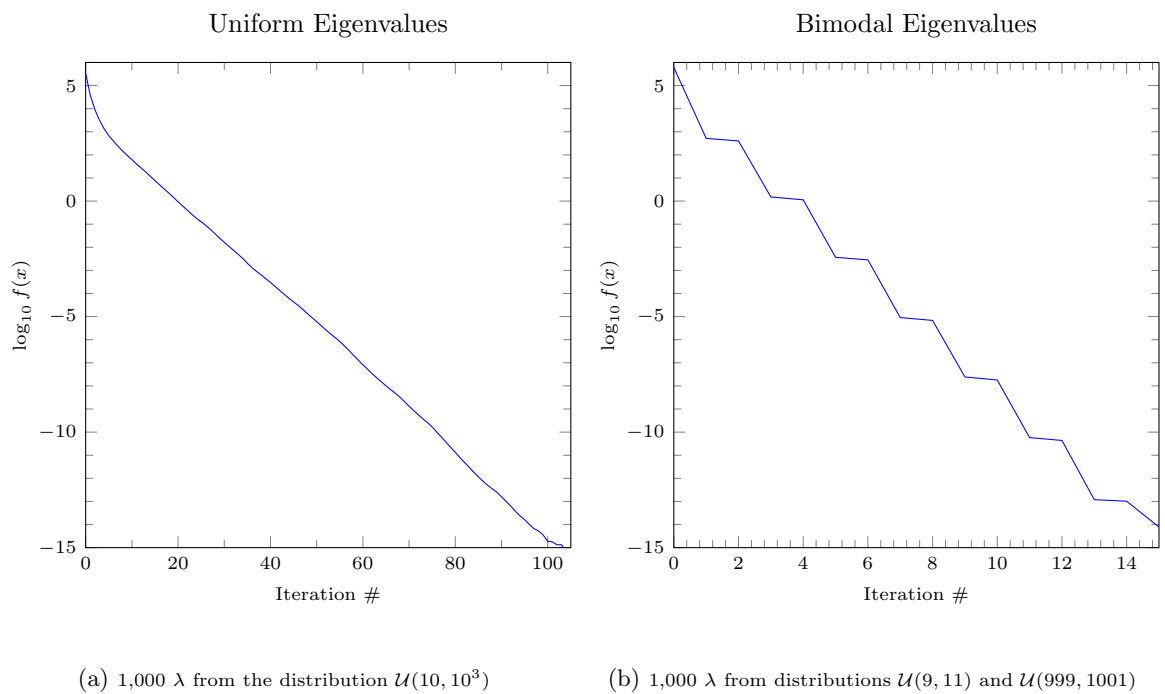


Figure 1: Log of function f error for problem #3

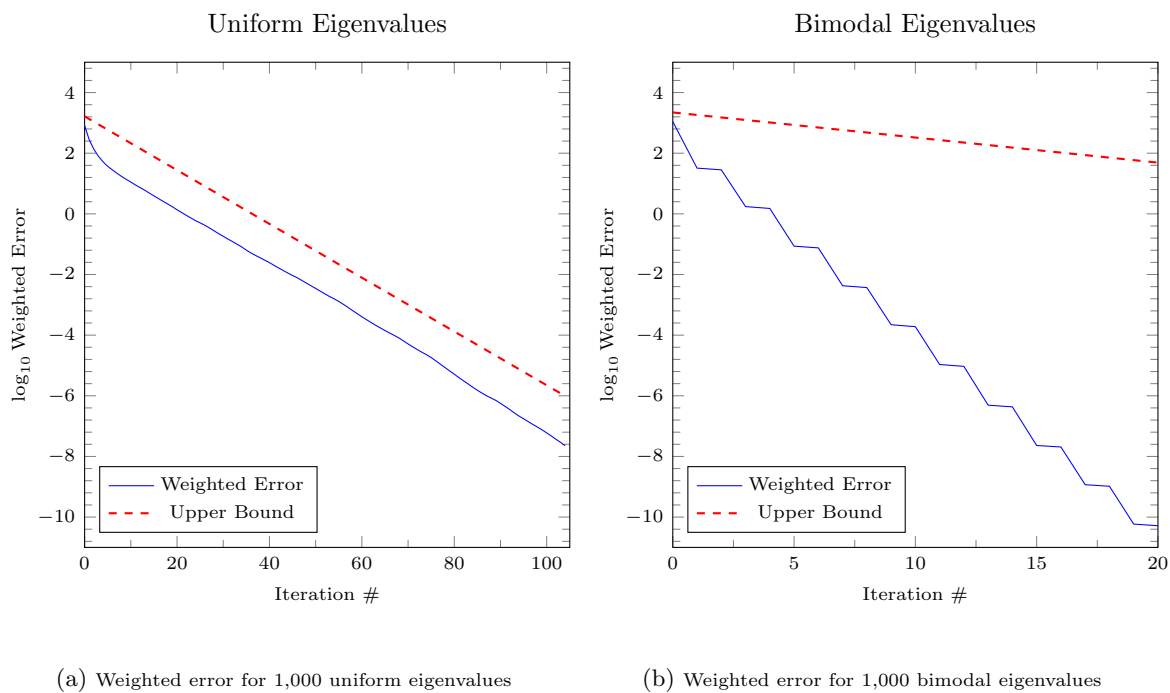


Figure 2: A -weighted error results for problem #3

Exercise #4

Consider the problem of minimizing

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

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

1. FR (Algorithm 5.4)
2. FR with restart based on Eq. (5.52) in Nocedal and Wright
3. PR based on Eq. (5.44) in Nocedal and Wright

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, $2 \cdot \text{rand}(n, 1)$.)

The gradient of f is:

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

Define $\psi_i = x_i + \alpha p_i$. Therefore, the derivative of ϕ is

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

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

Table 2: Experiment parameters for problem #4

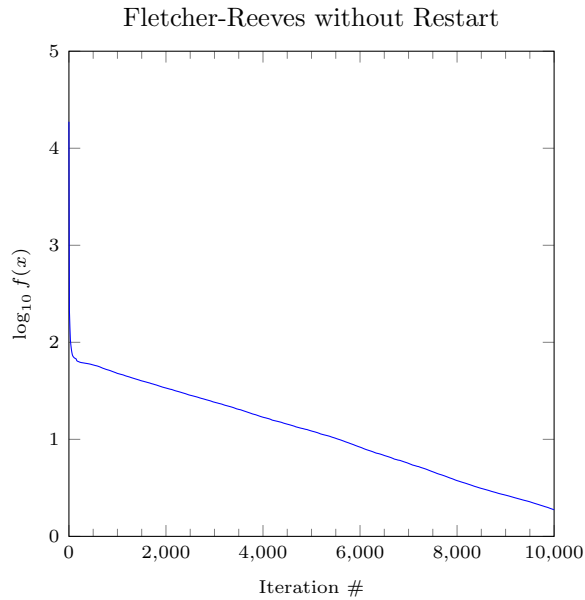
Parameter	Value
n	100
c_1	0.1
c_2	0.45
v	0.1
x_0	$2 \cdot \text{rand}(n, 1)$

To calculate α , I reused most of my implementation of inexact line search from homework #1.

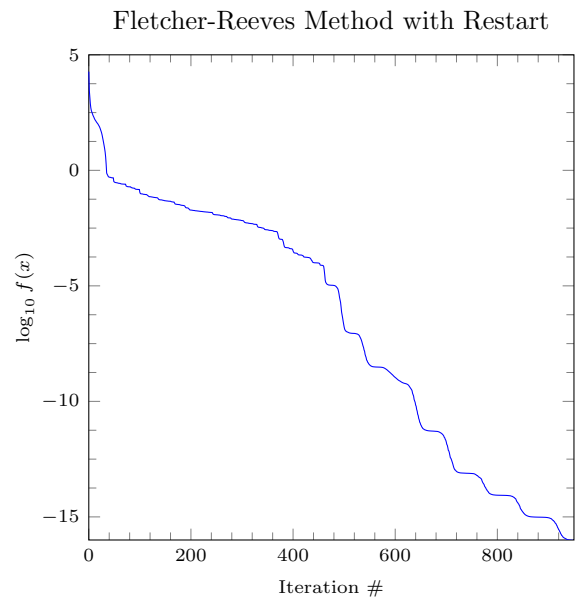
Figures 3a, 3b, and 4 show the performance of Fletcher-Reeves, Fletcher-Reeves with Restart, and Polak-Ribière respectively. In all experiments, the same random x_0 was used to ensure consistency. As expected,

standard Fletcher-Reeves had the worst performance. While it slowly makes progress, even after 10,000 iterations standard Fletcher-Reeves did not come close to converging to the optimal solution. Adding restart to Fletcher-Reeves improves the performance significantly, and the algorithm converged in about 900 iterations. Polak-Ribière converged the fastest in approximately 375 iterations.

It is important to note that different random x_0 could drastically effect the number of iterations it took all the algorithms to converged. In some rare cases, a different random start position caused Fletcher-Reeves with restart to converge the fastest.



(a)



(b)

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

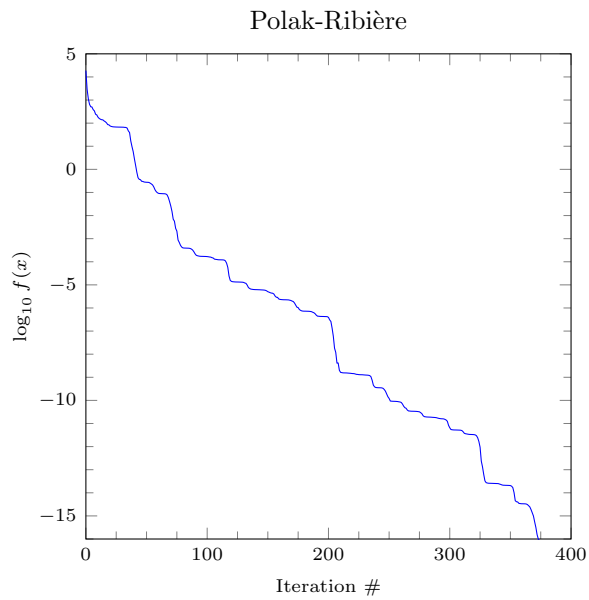


Figure 4: Problem #4 performance for the Polak-Ribière Method

Problem #3

Source Code

Problem #4

Source Code