ECE 506 Optimization Theory
Exam 1 with Suggested Solutions

Problem 1 ___ /10

Problem 2 ___ /25

Problem 3 ___ /10

Problem 4 ___ /20

Problem 5 ___ /10

Problem 6 ___ /25

Problem 1 (10 points)
1(a)(**8 points**) For two-dimensional problems, we can generate all possible sign combinations of the Hessian eigenvalues by considering:

$$f_1(x_1,\, x_2) = x_1^2 + x_2^2 \tag{1}$$
$$f_2(x_1,\, x_2) = x_1^2 - x_2^2 \tag{2}$$
$$f_3(x_1,\, x_2) = -x_1^2 + x_2^2 \tag{3}$$
$$f_4(x_1,\, x_2) = -x_1^2 - x_2^2 \tag{4}$$

For each function:

- Classify the stationary point for which $\nabla f_i = 0$ as a minimum point, a maximum point, or a saddle point.
- Sketch the contours around each stationary point.

1(b)(**2 points**) Based on your answer in 1(a), can you identify which two functions exhibit the same type of stationary point?

Answer:
1(a) We have:

$$\nabla f_1\,(x_1,\, x_2) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}, \qquad \nabla f_2\,(x_1,\, x_2) = \begin{bmatrix} 2x_1 \\ -2x_2 \end{bmatrix}$$
$$\nabla f_3\,(x_1,\, x_2) = \begin{bmatrix} -2x_1 \\ 2x_2 \end{bmatrix}, \qquad \nabla f_4\,(x_1,\, x_2) = \begin{bmatrix} -2x_1 \\ -2x_2 \end{bmatrix},$$

all of which have a single, globally stationary point for:

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Using the Hessian at $\mathbf{x}_0$, we get:

$$\nabla^2 f_1\,(x_1,\, x_2) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{has a minimum,} \qquad \nabla^2 f_2\,(x_1,\, x_2) = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix} \quad \text{has a saddle point,}$$
$$\nabla^2 f_3\,(x_1,\, x_2) = \begin{bmatrix} -2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{has a saddle point,} \quad \nabla^2 f_4\,(x_1,\, x_2) = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix} \quad \text{has a maximum.}$$

1(b) For both f_2, f_3, we have saddle points.

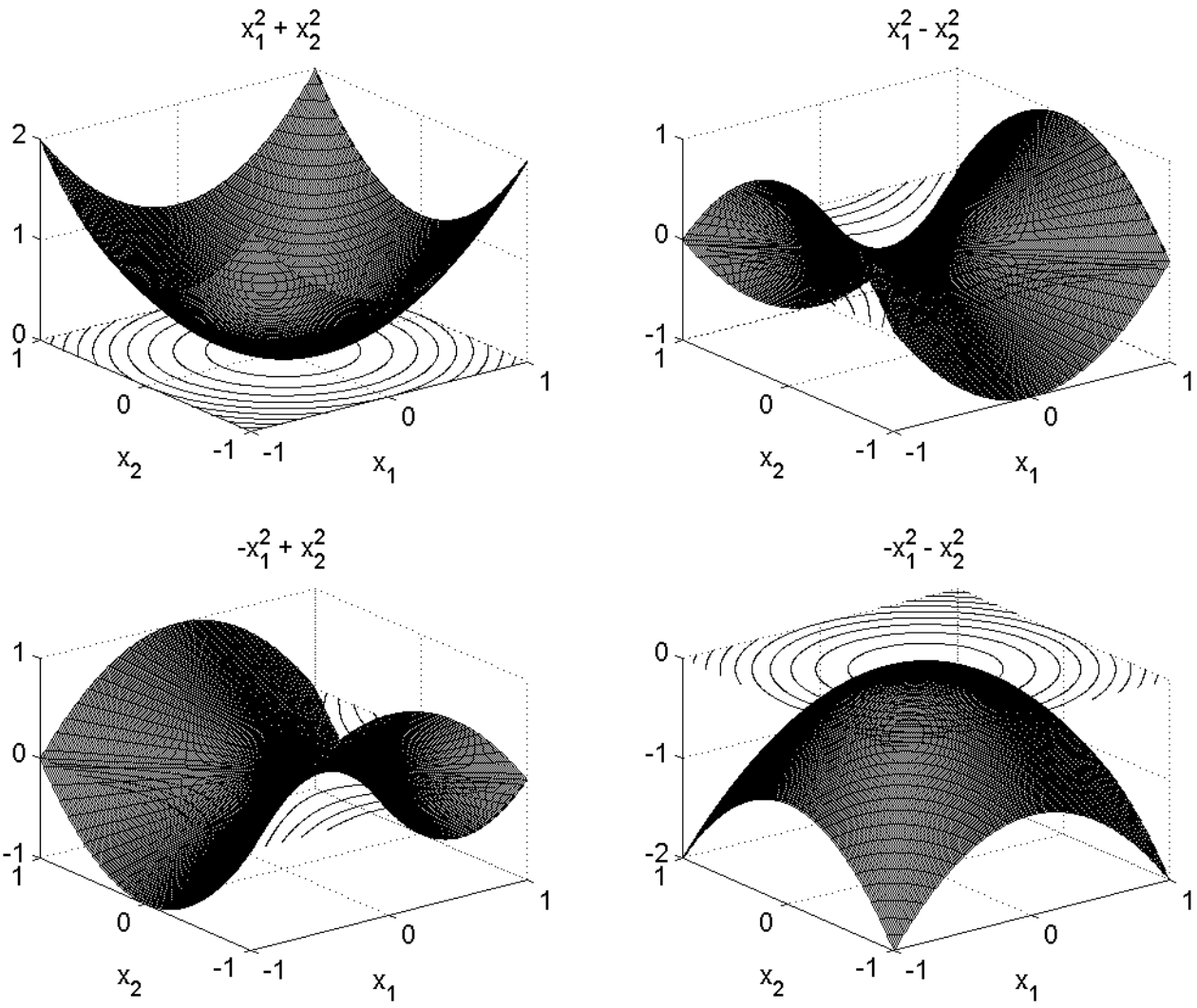

Figure 1: Contour Plots.

Matlab code for the plots:

```
close all
clear all

x_range  = -1:0.01:+1;
[x1, x2] = meshgrid(x_range, x_range);

f1 =  x1.^2 + x2.^2;
f2 =  x1.^2 - x2.^2;
f3 = -x1.^2 + x2.^2;
f4 = -x1.^2 - x2.^2;

figure(1);
subplot(2,2,1);
plot_fun_contours(x_range, x_range, f1, 'x_1^2 + x_2^2');

subplot(2,2,2);
plot_fun_contours(x_range, x_range, f2, 'x_1^2 - x_2^2');

subplot(2,2,3);
plot_fun_contours(x_range, x_range, f3, '-x_1^2 + x_2^2');

subplot(2,2,4);
plot_fun_contours(x_range, x_range, f4, '-x_1^2 - x_2^2');

print -deps all_funs.eps


function plot_fun_contours (x1_range, x2_range, f, f_title)
%
% plot_fun_contours (x1_range, x2_range, f, f_title)
%
% Plots the contours of f, where f is two-dimensional.
%
% Example:
%  x_range  = -1:0.01:1;
%  [x1, x2] = meshgrid(x_range, x_range);
%  f = x1.^2 + x2.^2;
%  plot_fun_contours(x_range, x_range, f, 'x_1^2 + x_2^2');
%

mesh(x1_range, x2_range, f), hold on;
contour(x1_range, x2_range, f);
title(f_title);
xlabel('x_1');
ylabel('x_2');
```

Problem 2 (25 points)
We begin by restating theorem 3.2 from your text.

Consider any iteration of the form:

$$x_{k+1} = x_k + \alpha_k p_k, \tag{5}$$

where a_k satisfies the Wolfe conditions:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{6}$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, \qquad 0 < c_1 < c_2 < 1. \tag{7}$$

Suppose that f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set N containing:

$$L = \{x : \, f(x) \leq f(x_0)\}, \tag{8}$$

where x_0 is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on N. Then

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty \tag{9}$$

2(a)(**5 points**) Use theorem 3.2 to show that the steepest descent algorithm will always converge.
2(b)(**8 points**) Provide a numerical example where the Wolfe conditions are satisfied. For full credit, you must provide the formula for the function and evaluate the Wolfe conditions at an admissible value of α_k, and you must also specify c_1, c_2.
2(c)(**6 points**) Suppose that you have developed a new algorithm for computing candidate directions p_k. Indicate how you would provide a new line-search algorithm that converges **at-least as many iterations** as steepest-descent or the Newton's method.

Answer:
2(a) Here, you were not expected to justify that steepest descent satisfies the Wolfe conditions. This is actually addressed in a separate question in the exam! In any case, it is easy to see that steepest descent satisfies the descent condition, while for the curvature condition we would have to use the mean value theorem.

To use theorem 3.2, note that $\cos^2 \theta_k = 1$, and thus:

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f_k\|^2 = \sum_{k \geq 0} \|\nabla f_k\|^2 < \infty \tag{10}$$

From this, it must be that $\|\nabla f_k\|^2 \to 0$ as $k \to 0$. To show this formally, note that if $\|\nabla f_k\|^2 \nrightarrow 0$, then it must be that $\|\nabla f_k\|^2 > a > 0$. We then have that:

$$\sum_{k \geq 0} \|\nabla f_k\|^2 > \sum_{k \geq 0} a \to \infty,$$

which would contradict (10). By definition, from $\|\nabla f_k\|^2 \nrightarrow 0$, we have convergence to a stationary point.

2(b) Consider $f(x_1, x_2) = x_1^2 + x_2^2$. Let $\mathbf{x}_k = \left[\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right]$. We have that $\nabla f(-1, 0) = \left[\begin{smallmatrix} -2 \\ 0 \end{smallmatrix}\right]$. We consider the steepest descent direction $p_k = -\nabla f(-1, 0)$.

We require:

$$f(-1 + 2\alpha_k, 0) \leq f(-1, 0) - 4c_1\alpha_k,$$
$$-\nabla f(-1 + 2\alpha_k, 0)^T \begin{bmatrix} -2 \\ 0 \end{bmatrix} \geq -4c_2, \qquad 0 < c_1 < c_2 < 1.$$

which reduces to:

$$(-1 + 2\alpha_k)^2 \leq 1 - 4c_1\alpha_k,$$
$$-4 + 8\alpha_k \geq -4c_2, \qquad 0 < c_1 < c_2 < 1.$$

For $\alpha_k = 1/2$, $0 < c_1 = 0.1 < c_2 = 0.5 < 1$:

$$0 \leq 0.8,$$
$$0 \geq -2.$$

2(c) The key to the answer was that we only require the same number of iterations. We do not require that we have fewer function evaluations. We thus modify the standard line-search framework to:

for $k = 0, 1, \ldots,$ *MaxIterations* **do**
  % Compute the directions for each method:
  $p_k^{steepest\_descent} = -\nabla f(x_k)$
  $p_k^{newton} = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$
  Compute $p_k^{new\_method}$ for the new method.
  % Compute the steps for each method:
  Compute $\alpha_k^{steepest\_descent}$, α_k^{newton}, $\alpha_k^{new\_method}$ so that the Wolfe, strong Wolfe, or Goldstein conditions are satisfied.
  % For α_k^{newton}, we could simply set it to $\alpha_k^{newton} = 1$.
  % Compute the next points for each method:
  $x_{k+1}^{steepest\_descent} = x_k + \alpha_k^{steepest\_descent} p_k^{steepest\_descent}$
  $x_{k+1}^{newton} = x_k + \alpha_k^{newton} p_k^{newton}$
  $x_{k+1}^{new\_method} = x_k + \alpha_k^{new\_method} p_k^{new\_method}$
  if $f(x_{k+1}^{new\_method}) \leq f(x_{k+1}^{steepest\_descent})$ **and** $f(x_{k+1}^{new\_method}) \leq f(x_{k+1}^{newton})$ **then**
    $x_{k+1} = x_{k+1}^{new\_method}$
  else if $f(x_{k+1}^{newton}) \leq f(x_{k+1}^{steepest\_descent})$ **and** $f(x_{k+1}^{newton}) \leq f(x_{k+1}^{new\_method})$ **then**
    $x_{k+1} = x_{k+1}^{newton}$
  else
    $x_{k+1} = x_{k+1}^{steepest\_descent}$
  end
end

2(d)(**6 points**) Indicate how you would establish that your new algorithm proposed in 2(c) is **at-least as fast** as the steepest-descent or the Newton's method. In other words you must specifically show how you would assess your algorithm's

- accuracy,
- robustness, and
- efficiency.

Answer:
For **accuracy**, we will need to run our algorithm for a number of problems for which the optimal point x_* is known. We will then compare the rate of convergence of x_k to the solution x_* by plotting $\|x_k - x_*\|$ for our algorithm, Newton's, and steepest descent, versus the number of function evaluations.

The problem with this approach is that this solution only works for well-known problems. How can we prove that our method works for generic problems?

In this case, we simply plot $f(x_k)$ versus number of function evaluations. The method that reduces $f(.)$ the fastest will also be the most accurate.

For **robustness**, we simply run the code with random initial points x_0 and check for accuracy as outlined above.

For **efficiency**, we can count the number of function evaluations that it takes for each method. A second measure would be to measure the execution time and plot it against the accuracy achieved, but this approach assumes that we have production-level quality optimization code. A better, third alternative is to perform asymptotic analysis on the execution time in terms of the characteristics of the optimization problem (problem size).

Problem 3 (10 points)
Given a Hessian matrix H, outline an algorithm that can be used to:

- establish when H is positive definite, and
- modify H so that it becomes positive definite.

Answer:
3(a) Here is the pseudocode that will detect if H is positive definite:

```
% Assume eigenvalues satisfy: lambda_1 < ... < lambda_n
[lamaba_1, ..., lambda_n] = eigenvalues(H);
if (lambda_1>0) then
  H is positive definite
else
  H is not positive definite
end
```

3(b) For the modification of the code, simply append the following lines to the code given in 3(a):

```
if (H is not positive definite)
  H = H - (lambda_1 - epsilon) I  % Use epsilon to control smallest eigenvalue
end
```

Problem 4 (20 points).
Suppose that f is twice differentiable and that the Hessian $\nabla^2 f(x)$ is Lipschitz continuous in a neighborhood of the solution x^*, at which the second order sufficient conditions for a strict minimum are satisfied. Show that the Newton algorithm:

$$x_{k+1} = x_k + p_k \tag{11}$$
$$p_k^N = -\nabla^2 f_k^{-1} \nabla f_k \tag{12}$$

converges quadratically, assuming that the starting point is sufficiently close to x^*.

Answer:
The lecture notes provide a very thorough discussion of this problem. Please consult the lecture notes.

Problem 5 (10 points)

for $k = 0, 1, \ldots,$ *MaxIterations* **do**
  Compute p_k
  Compute α_k so that the Wolfe, strong Wolfe, or Goldstein conditions are satisfied.
  $x_{k+1} = x_k + \alpha_k p_k$
end

Algorithm 1: General Framework for Line-Search Algorithms

5(a)(6 points) Give the formulas for computing p_k using

- the steepest descent method, and
- the exact Newton's method.

5(b)(4 points) Explain why we can always expect to find α_k that satisfies the Wolfe conditions.

Answer: 5(a)

$$p_k^{steepest\_descent} = -\nabla f(x_k)$$
$$p_k^{newton} = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

5(b) We provide a sketch of the proof given in class and in your textbook. The Wolfe condition for function reduction is satisfied if p_k is chosen to be a descent direction since it is clear that f will be reduced along the chosen direction. For the curvature condition, note that since the function is bounded from below, there will eventually be an intersection point where the slope line from the first condition meets the function. It then follows by the mean value theorem that there will be a slope of the function that is more positive (towards zero) from the starting slope.

Problem 6 (25 points)

1.1 Given $\bar{\Delta} > 0$, $\Delta_0 \in (0, \tilde{\Delta})$ and $\eta \in [0, 1/4]$.
1.2 **for** $k = 0, 1, 2, \ldots,$ *MaxIterations* **do**
1.3 Obtain p_k by approximately solving:
1.4 $\min_{p \in \mathbb{R}^n} m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$ such that: $\|p\| \leq \Delta_k$
1.5 Evaluate the reduction ratio:
1.6 $\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$
1.7 **if** $\rho_k < \frac{1}{4}$ **then**
1.8 $\Delta_{k+1} = \frac{1}{4}\Delta_k$;
1.9 **else**
1.10 **if** $\rho_k > \frac{3}{4}$ *and* $\|p_k\| = \Delta_k$ **then**
1.11 $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$
1.12 **else**
1.13 $\Delta_{k+1} = \Delta_k$
1.14 **end**
1.15 **end**
1.16 **if** $\rho_k > \eta$ **then**
1.17 $x_{k+1} = x_k + p_k$
1.18 **else**
1.19 $x_{k+1} = x_k$
1.20 **end**
1.21 **end**

Algorithm 2: General Framework for Trust-Region Algorithms

6(a)(6 points) Based on the trust-region algorithm, please explain:

- How is the trust region size increased?
- How is the trust region size decreased?
- When is a new p_k direction taken?

Answer:
6(a) The trust region size is reduced in 1.8. This happens if the reduction ratio is significantly less than what was expected.

The trust region size is increased in 1.11. This happens if the reduction ratio is very close or significantly larger than what was expected.

If the reduction ratio is greater a threshold value η, then the direction is taken in 1.16.

6(b)(13 points) Suppose that an efficient algorithm is available to you that can estimate any one eigenvalue or eigenvector of a given matrix. Indicate how you to use this algorithm to improve upon the Cauchy point given as:

$$p_k^c = \tau_k p_k^s \tag{13}$$

where:

$$p_k^s = \frac{-\Delta_k}{\|\nabla f_k\|} \nabla f_k \tag{14}$$

$$\tau_k = \begin{cases} 1, & \text{if} \quad \nabla f_k^T B_k \nabla f_k \leq 0 \\ \min\left(\frac{\|\nabla f_k\|^3}{\Delta_k \nabla f_k^T B_k \nabla f_k}, 1\right), & \text{otherwise.} \end{cases} \tag{15}$$

6(c)(6 points) Explain how your algorithm will work on

$$g(x_1, x_2) = x_1^2 + a x_2^2, \tag{16}$$

for the cases when (i) $a \gg 1$, (ii) $a \ll 1$, and $a = 1$.

Answer:

6(b) There are several possible answers. The following minimizes the function $f(.)$ along the dominant eigenvector direction, and accepts the new direction if it improves on the Cauchy point:

```
% Estimate the largest eigenvalue/eigenvector pair of the Hessian H
[lambda_1, e_1] = largest_eigenvalue_vector (H);

p_sign = sign(e_1 dot_product grad(f)) % Make p be a descent direction
p_dir  = Delta_k / ||e_1|| e_1         % Move along e_1 to boundary of trust region
if (p_sign > 0)
  p_new = p_dir
else
  p_new = -p_dir
end

if f(x_k+p_new) < f(x_k+p_cauchy)
  x_k_plus_1 = x_k + p_new
else
  x_k_plus_1 = x_k + p_cauchy
end
```

A second alternative is to replace all the p_k directions in the Conjugate gradient method using:

```
[lambda_i, e_i] = eigenvalue_eigenvector (H);
```

starting from the eigenvector and then using the smallest eigenvector in the matrix.

A third alternative is to perform *coordinate descent* on each eigenvector direction, or on just a few of the largest eigenvector directions.

6(c) In the example, we have that the Hessian is simply

$$\begin{bmatrix} 2 & 0 \\ 0 & 2a \end{bmatrix} \quad \text{with eigenvalues} \quad \lambda_1 = 2,\ \lambda_2 = 2a.$$

We provide answers for the first proposed algorithm. Note that the coordinate descent algorithm will reach the optimum point for this separable function. The coordinate descent algorithm will actually reach the optimum point for any quadratic function, because it will be perfectly separable along the eigenvector directions (stated here without proof).

Now, if $a \gg 1$, the second eigenvector $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ is the dominant one and the function is minimized along this direction instead of the Cauchy direction. Furthermore, since the reduction in the second coordinate is significantly more than the first, our method will lead to a reduction that will be more than the Cauchy point (which takes steps in both directions).

For $a \ll 1$, the first eigenvector $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ is the dominant one, and the situation is the same as for the first case, except that the first eigenvector replaces the role of the second eigenvector.

For $a = 1$, there is no preferred eigenvector direction. Thus, our algorithm may select the first or the second eigenvector direction. Here, we note that the coordinate descent direction might lead to significantly better reduction in the function (easy to draw an example that shows this). Thus as many students noted, the new algorithm is great for detecting scaling issues, detected when the ratio of the largest to the smallest eigenvalue is large. However, you still had to suggest a new direction in this case.