ECE 506 – Homework #4

Line Search, Trust Regions, and Conjugate Gradient

Your Name

Fall 2025

Discussion sessions. Saturday mornings at 9am: https://unm.zoom.us/j/99977790315.

Reference. Nocedal & Wright (2006), *Numerical Optimization*, Springer.

Code. Starter Matlab code: https://github.com/pattichis/opt/blob/main/Matlab-code-for-opt.zip.

Coding examples (required). Every solution must include (1) documented source code, (2) plots, and (3) discussion. Unless a part is explicitly marked "sketch," show a runnable coding example supporting your answer.

Problem 1. [Stepsizes and convergence rates for line search methods]

1(a) Consider the following general equation form:

$$f(x) = ax_1^2 + bx_2^2 + cx_1 + dx_2 + e.$$

For this case, compute the optimal point x^* and provide conditions that guarantee that this is an optimal point.

Solution.

$$\frac{\partial f}{\partial x_1} = 2ax_1 + c, \qquad \frac{\partial f}{\partial x_2} = 2bx_2 + d, \qquad \nabla^2 f = \begin{bmatrix} 2a & 0 \\ 0 & 2b \end{bmatrix}.$$

Setting the gradient to zero gives

$$x_1^* = -\frac{c}{2a}, \qquad x_2^* = -\frac{d}{2b}.$$

This stationary point is a (unique) minimizer when $a > 0$ and $b > 0$ (Hessian $\succ 0$); it is a maximizer when $a < 0$ and $b < 0$; otherwise it is a saddle or unbounded in at least one direction.

1(b) Compute the optimal stepsize for steepest descent line search.

Solution. Let $g_k = \nabla f(x_k)$, $H = \nabla^2 f$. For steepest descent with exact line search and $p_k = -g_k$,

$$\boxed{\alpha_k^* = \frac{g_k^\top g_k}{g_k^\top H g_k}}$$

so for $f(x) = ax_1^2 + bx_2^2 + cx_1 + dx_2 + e$ with $g_k = \begin{bmatrix} 2ax_{1,k} + c \\ 2bx_{2,k} + d \end{bmatrix}$ and $H = \begin{bmatrix} 2a & 0 \\ 0 & 2b \end{bmatrix}$,

$$\boxed{\alpha_k^* = \frac{(2ax_{1,k} + c)^2 + (2bx_{2,k} + d)^2}{2a(2ax_{1,k} + c)^2 + 2b(2bx_{2,k} + d)^2}}$$

(positive when $a > 0,\ b > 0$).

1(c) For simplicity, consider the 1D case for $b = c = d = e = 0$. We know that $x^* = 0$. Compute simplified expressions for the stepsize and the magnitude of the gradient. Plot the stepsize as a 2D function of a, x_1 for $x_2 = 0$. Answer the following:

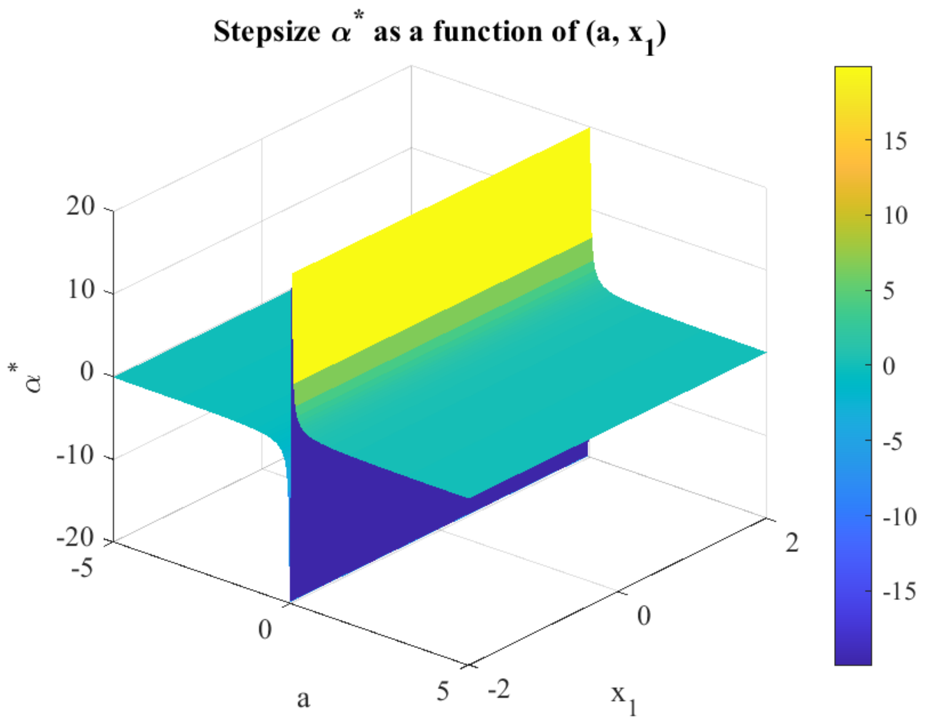

Figure 1: Stepsize α^* as a function of (a, x_1) for $f(x) = ax^2$.

i. What kind of symmetries do you observe in your plot? For example, consider replacing a by $-a$ and/or x_1 by $-x_1$.
Solution The plot is flat in x_1 (no x_1-dependence), symmetric under $x_1 \mapsto -x_1$, and antisymmetric under $a \mapsto -a$ (sign flip of α^*); there is a discontinuity at $a = 0$.

ii. Is the stepsize relatively large for large x_1? Explain.
Solution No. The stepsize does *not* grow with x_1 since $\alpha^* = \frac{1}{2a}$ is independent of x_1.

iii. Is the step $\|\alpha p_k\|$ relatively large for large x_1? Explain.
Solution Yes. The step magnitude is

$$\|\alpha^* p_k\| = \left|\frac{1}{2a}(-2ax_{1,k})\right| = |x_{1,k}|,$$

so it grows linearly with $|x_1|$.

iv. What happens to the stepsize as we approach the optimal point? Explain.
Solution (iv) As $x_1 \to x^* = 0$, the stepsize stays constant: $\alpha^* = \frac{1}{2a}$ (for fixed $a > 0$).

v. What happens to the step $\|\alpha p_k\|$ as we approach the optimal point? Explain.
Solution As $x_1 \to 0$, the step norm tends to zero linearly:

$$\|\alpha^* p_k\| = |x_{1,k}| \longrightarrow 0.$$

Solution.

1(d) Let us consider a simple case to see how the steepest descent algorithm works. Assume that:

$$f(x) = x_1^2 + 10x_2^2.$$

Suppose that $x_0 = x^* + [1, 1]^T$. Compute two steps for the steepest descent algorithm.
Solution.

$$f(x) = x_1^2 + 10x_2^2, \qquad \nabla f(x) = \begin{bmatrix} 2x_1 \\ 20x_2 \end{bmatrix}, \qquad \nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 20 \end{bmatrix}.$$

1(e) Verify your results in 1(c) with the provided code. You need to provide the code that you used to run the provided code.
Solution.

Listing 1: Verification code for Problem 1(e)

```
%% ECE 506 - HW4 Problem 1(e)
% Verification of analytical results from Problem 1(c)
% Using provided steepest_descent.m
% Author: Scott Nguyen | Fall 2025
% (rest of your script unchanged)

```

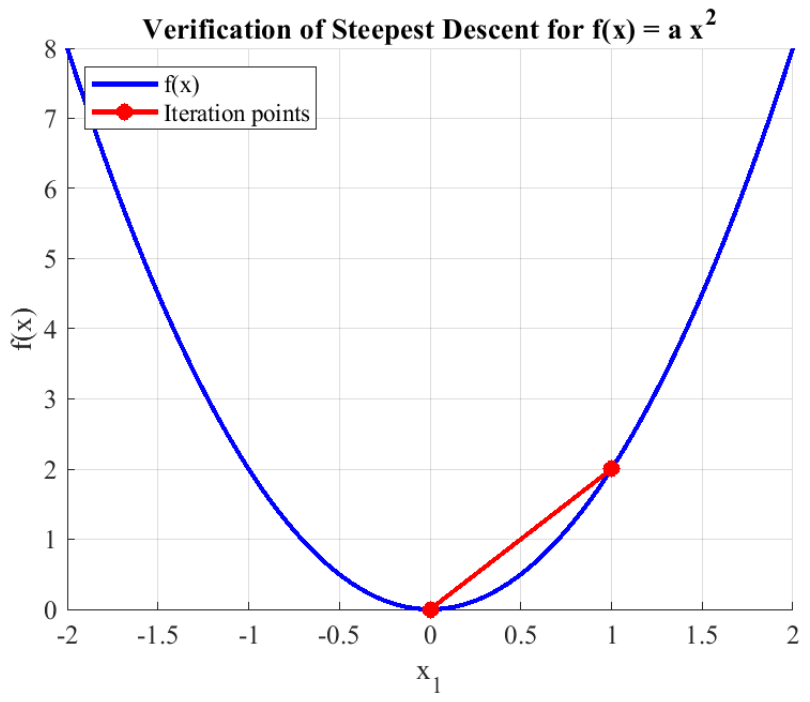

Figure 2: Verification of steepest descent for $f(x) = ax^2$. The algorithm converges to $x^* = 0$ in two steps, consistent with the analytical result derived in 1(c).

1(f) For our quadratic case, provide the following:

i. Compute an expression for $\|.\|_Q$.

Solution

ii. Compute an expression in terms of the distance from x^*.

Solution

iii. Provide the best convergence case in terms of a, b.

Solution

iv. Provide the worse condition case in terms of a, b, assuming that $a > b$.

Solution

1(g) Repeat 1(d) for Newton's method. Do you need two steps? Explain using the Newton's algorithm convergence theorem.

Solution.

Problem 2

Consider the following methods for which you are given Matlab codes for line-search:

1. Steepest Descent,
2. BFGS, and
3. Newton's method.

We want to consider how each method performs on finding the minima of (see Wikipedia: Test functions for optimization for definitions):

1. Sphere function
2. Beale function
3. Goldstein-Price function
4. Booth function

You will need to assess:

Robustness: Run the program with random initial guesses (that still satisfy any constraints), to assess the performance. Show results from at-least 3 random initial points.

Efficiency: Compute the required memory and function evaluations for each iteration. For memory requirements, express your results in terms of n, the dimensionality of the problem. Thus, a gradient requires that we store n floating-point values. You will need to modify the code to keep track. In terms of function evaluations, report function, gradient, and hessian evaluations separately. Thus, you may need to produce a total of four plots. When computing requirements, you should not include any extra requirements associated with storing the path for visualization purposes. In other words, you need to focus on essential requirements only.

Accuracy: For each run, establish the best rate of convergence. Discuss whether the theoretical rate of convergence is accomplished. For the rate of convergence, apply both methods provided in the hints. Do the two methods agree? Explain.

Solution. Below, for each test function we list only the objective $f(x, y)$.

a) **Sphere function** ($n = 2$):
$$f(x, y) = x^2 + y^2.$$

b) **Beale's function**:
$$f(x, y) = \big(1.5 - x + xy\big)^2 + \big(2.25 - x + xy^2\big)^2 + \big(2.625 - x + xy^3\big)^2.$$

c) **Goldstein–Price function**:
$$f(x, y) = \Big[1+(x+y+1)^2\big(19-14x+3x^2-14y+6xy+3y^2\big)\Big]\Big[30+(2x-3y)^2\big(18-32x+12x^2+48y-36xy+27y^2\big)\Big].$$

Table 1: Compact summary across problems, methods, and random initializations. Columns retained to support: *Robustness* (runs, convergence, iterations); *Efficiency (memory)* in terms of n; and *Accuracy-lite* via terminal errors.

Problem	Run	Method	Iter.	Converged?	x_{err}	f_{err}	Memory / iter (in n)
Sphere							
Sphere	1	Steepest Descent	1	Yes	0	0	$\mathcal{O}(n)$
Sphere	1	BFGS	1	Yes	0	0	$\mathcal{O}(n^2)$
Sphere	1	Newton	1	Yes	0	0	$\mathcal{O}(n^2)$
Sphere	2	Steepest Descent	1	Yes	0	0	$\mathcal{O}(n)$
Sphere	2	BFGS	1	Yes	0	0	$\mathcal{O}(n^2)$
Sphere	2	Newton	1	Yes	0	0	$\mathcal{O}(n^2)$
Sphere	3	Steepest Descent	1	Yes	0	0	$\mathcal{O}(n)$
Sphere	3	BFGS	1	Yes	0	0	$\mathcal{O}(n^2)$
Sphere	3	Newton	1	Yes	0	0	$\mathcal{O}(n^2)$
Beale							
Beale	1	Steepest Descent	802	Yes	2.3047×10^{-6}	8.0377×10^{-13}	$\mathcal{O}(n)$
Beale	1	BFGS	12	Yes	5.7961×10^{-9}	2.7992×10^{-17}	$\mathcal{O}(n^2)$
Beale	1	Newton	6	Yes	1.89×10^{-12}	5.7295×10^{-25}	$\mathcal{O}(n^2)$
Beale	2	Steepest Descent	653	Yes	2.4335×10^{-6}	8.9581×10^{-13}	$\mathcal{O}(n)$
Beale	2	BFGS	6	Yes	7.1198×10^{-8}	1.4303×10^{-15}	$\mathcal{O}(n^2)$
Beale	2	Newton	3	Yes	3.8091×10^{-10}	2.198×10^{-20}	$\mathcal{O}(n^2)$
Beale	3	Steepest Descent	781	Yes	2.5932×10^{-6}	1.0171×10^{-12}	$\mathcal{O}(n)$
Beale	3	BFGS	20	Yes	2.2382×10^{-8}	7.8093×10^{-17}	$\mathcal{O}(n^2)$
Beale	3	Newton	1000	**No**	8.7482×10^{5}	4.5201×10^{-1}	$\mathcal{O}(n^2)$
Goldstein-Price							
Goldstein-Price	1	Steepest Descent	117	Yes	5.0568×10^{-10}	0	$\mathcal{O}(n)$
Goldstein-Price	1	BFGS	2	**No**	1.5602×10^{-1}	8.1371	$\mathcal{O}(n^2)$
Goldstein-Price	1	Newton	1000	**No**	1.5602×10^{-1}	8.1371	$\mathcal{O}(n^2)$
Goldstein-Price	2	Steepest Descent	1000	**No**	2.1633	81	$\mathcal{O}(n)$
Goldstein-Price	2	BFGS	1	**No**	8.3244×10^{-1}	700.35	$\mathcal{O}(n^2)$
Goldstein-Price	2	Newton	1000	**No**	8.3244×10^{-1}	700.35	$\mathcal{O}(n^2)$
Goldstein-Price	3	Steepest Descent	1000	Yes	4.5589×10^{-9}	7.7716×10^{-14}	$\mathcal{O}(n)$
Goldstein-Price	3	BFGS	1	**No**	6.1185×10^{-1}	207.68	$\mathcal{O}(n^2)$
Goldstein-Price	3	Newton	5	Yes	2.1741×10^{-9}	1.0214×10^{-14}	$\mathcal{O}(n^2)$
Booth							
Booth	1	Steepest Descent	30	Yes	2.3918×10^{-7}	5.8612×10^{-14}	$\mathcal{O}(n)$
Booth	1	BFGS	2	Yes	1.0474×10^{-15}	7.8886×10^{-31}	$\mathcal{O}(n^2)$
Booth	1	Newton	1	Yes	2.2204×10^{-16}	0	$\mathcal{O}(n^2)$
Booth	2	Steepest Descent	32	Yes	2.0596×10^{-7}	4.3374×10^{-14}	$\mathcal{O}(n)$
Booth	2	BFGS	2	Yes	0	0	$\mathcal{O}(n^2)$
Booth	2	Newton	1	Yes	1.4218×10^{-15}	3.9443×10^{-30}	$\mathcal{O}(n^2)$
Booth	3	Steepest Descent	34	Yes	2.3026×10^{-7}	5.4270×10^{-14}	$\mathcal{O}(n)$
Booth	3	BFGS	2	Yes	1.7342×10^{-15}	3.9443×10^{-30}	$\mathcal{O}(n^2)$
Booth	3	Newton	1	Yes	1.1102×10^{-15}	7.8886×10^{-31}	$\mathcal{O}(n^2)$

d) **Booth's function**:

$$f(x,y) = \big(x + 2y - 7\big)^2 + \big(2x + y - 5\big)^2.$$

The results for Steepest Descent (SD), BFGS, and Newton are summarized in the figures below. Each method was tested on the Sphere, Beale, Goldstein–Price, and Booth functions using 100 random initial points. Robustness, efficiency, and accuracy are discussed together here.

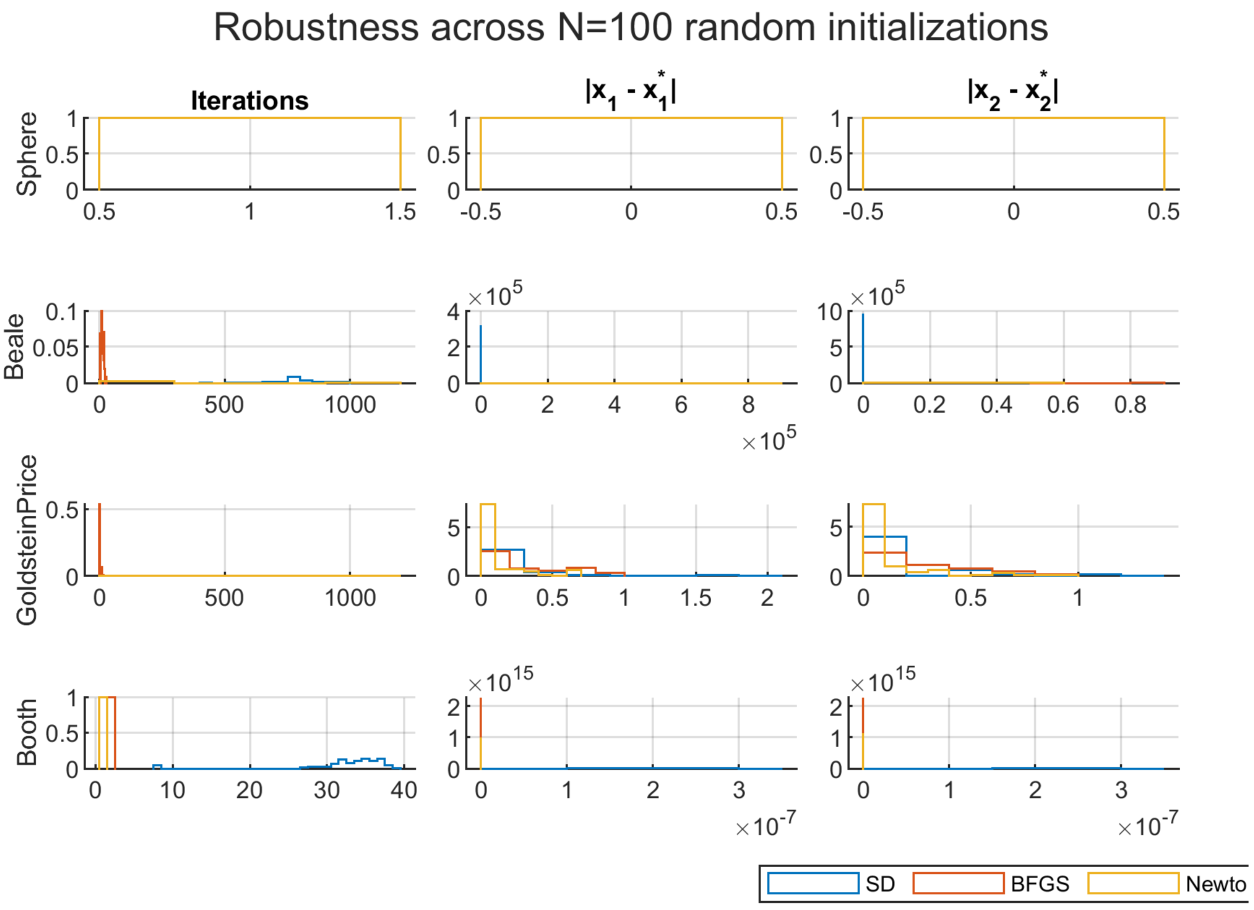

Figure 3: Robustness across $N = 100$ random initializations for each test function.

Overall, Steepest Descent converged the most consistently but required the most iterations. BFGS achieved a strong balance between speed and reliability, while Newton was the fastest when it converged, though it often failed on nonconvex problems like Beale or Goldstein–Price. The histograms in Fig. 3 show this clearly: for convex cases (Sphere, Booth), all methods converged smoothly, but in nonconvex cases, SD's broader distributions indicate slower yet more reliable behavior.

From Fig. 4, the efficiency analysis shows expected trends: SD and BFGS require only function and gradient evaluations, while Newton also evaluates the Hessian. Memory requirements scale as $\mathcal{O}(n)$ for SD and $\mathcal{O}(n^2)$ for both BFGS and Newton. In practice, Newton's higher cost per iteration is justified only when rapid local convergence occurs.

As seen in Fig. 5, Steepest Descent exhibits linear convergence, BFGS shows superlinear behavior, and Newton achieves quadratic convergence whenever it reaches the correct basin. The ratio

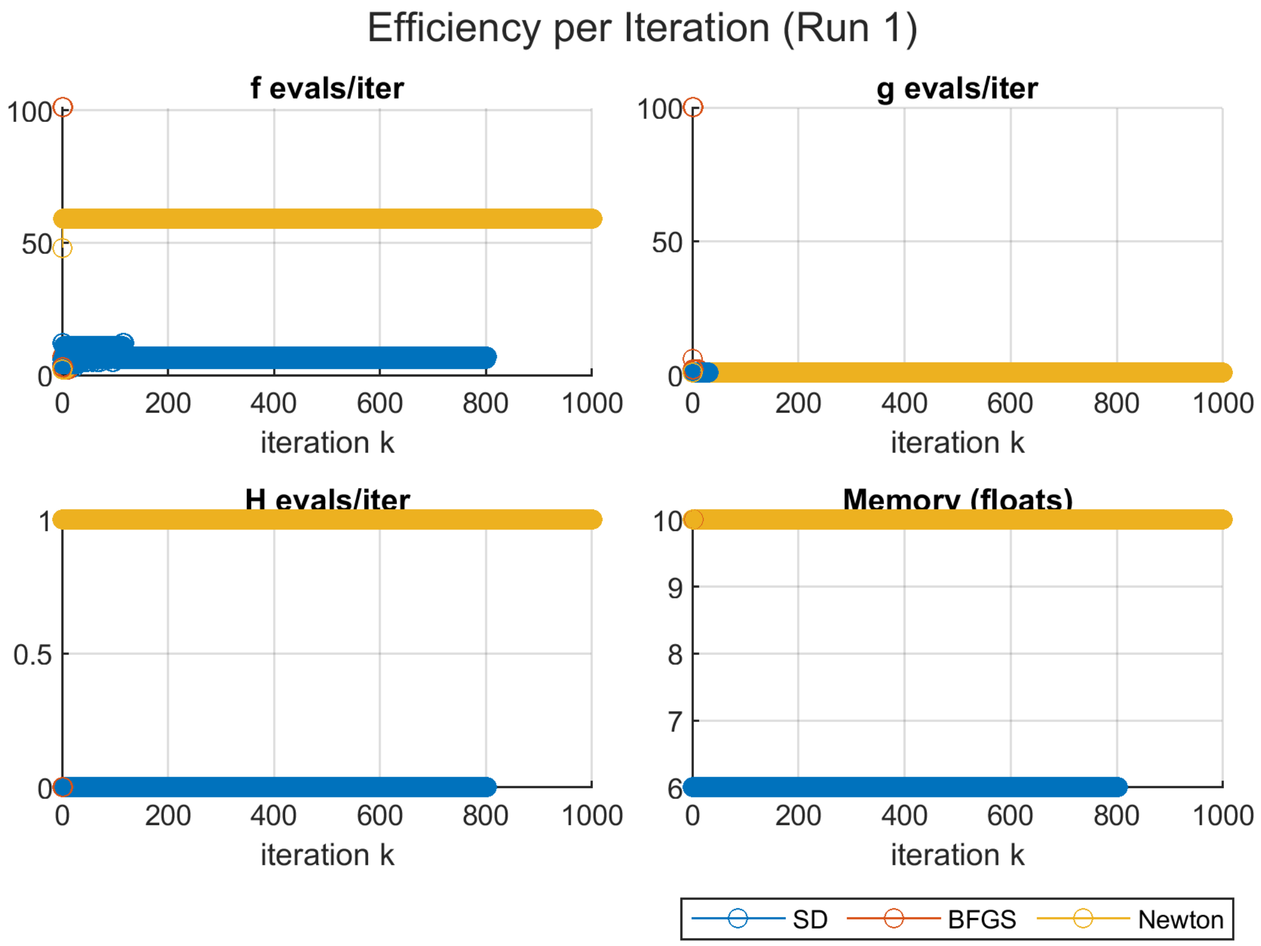

Figure 4: Efficiency per iteration for Run 1: function, gradient, and Hessian evaluations, and memory usage.

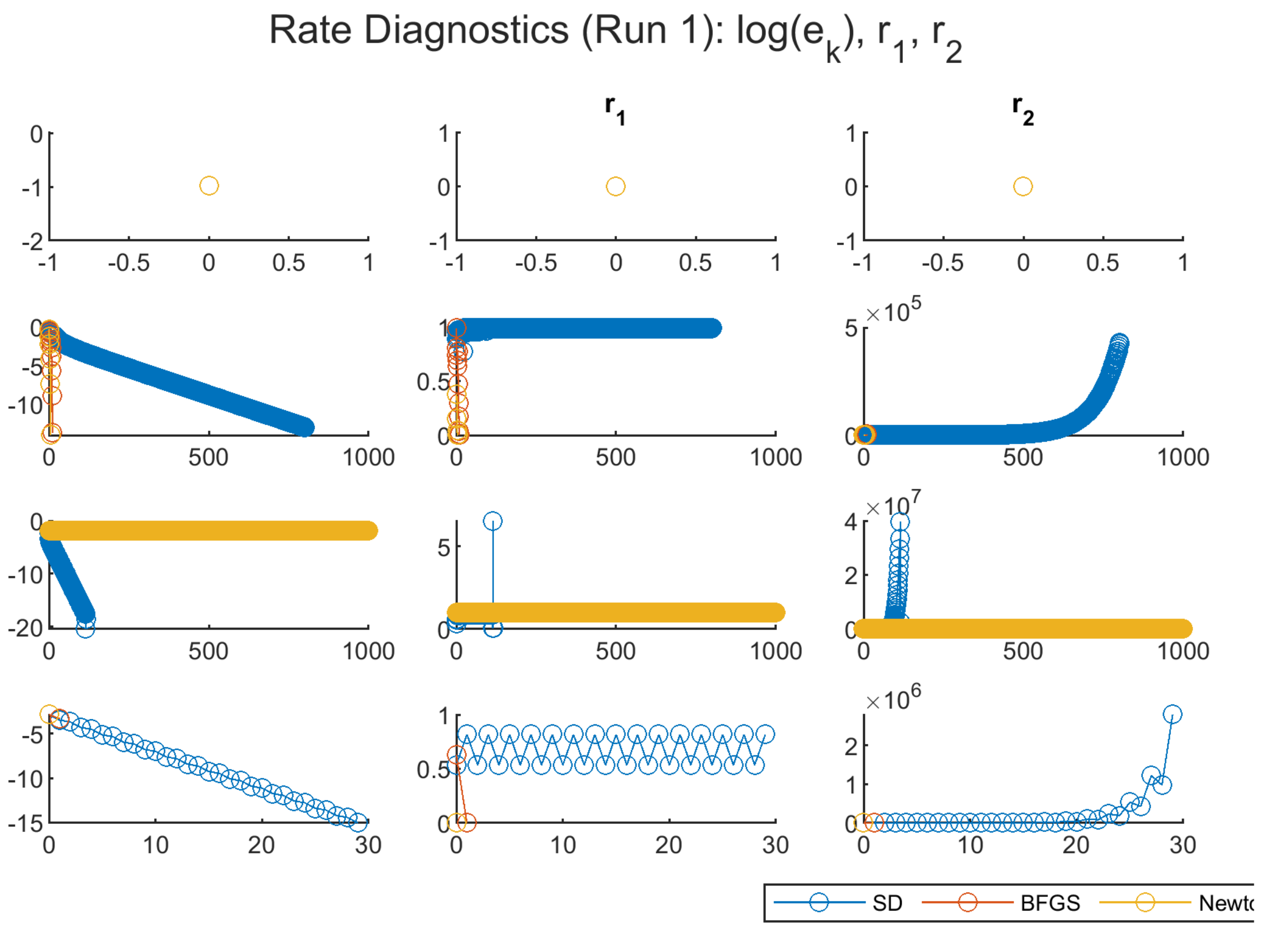

Figure 5: Rate diagnostics for Run 1 showing $\log(e_k)$, $r_1 = e_{k+1}/e_k$, and $r_2 = e_{k+1}/e_k^2$.

tests (r_1 and r_2) confirm these trends: SD stabilizes at a constant rate, BFGS improves progressively, and Newton's error drops sharply. On nonconvex problems, deviations occur when iterates start outside the local basin, causing slower or unstable convergence.

In summary, SD is the most robust but slowest, BFGS is the best overall compromise, and Newton is fastest but least reliable on nonconvex landscapes. These observations match the theoretical expectations for each method.

Problem 3. Trust-region Conjugate Gradient (Steihaug)

Using the provided code for solving Problem 4.3 via Conjugate Gradient Steihaug, repeat the analyses from Problem 2 (robustness, efficiency, and accuracy/rate).

Results (tables/plots + discussion). ______

Hints

1. Symbolic Gradients and Hessians (optional). In Matlab:

```
syms x y z
f = x*y + 2*z*x;
hessian(f, [x, y, z])
gradient(f, [x, y, z])
```

2. Rates of convergence. If the optimizer $\mathbf{x}^*$ is known, plot versus iteration k:

$$\frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|}, \qquad \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^2}.$$

When $\mathbf{x}^*$ is unknown, use gradients:

$$\frac{\|\nabla f_{k+1}\|}{\|\nabla f_k\|}, \qquad \frac{\|\nabla f_{k+1}\|}{\|\nabla f_k\|^2}.$$

Report the highest order achieved (Q-quadratic $\Rightarrow$ Q-superlinear $\Rightarrow$ Q-linear). For Problem 6, show that

$$\liminf_{k\to\infty} \|\nabla F(\mathbf{w}_k)\| = 0.$$

Appendix: Insert figures and code listings here (or use `\input` to include files).