Numerical Optimisation and Large–Scale Systems Assessed Coursework 2022–23

MA40050

Harry Lyness

April 24, 2023

Summary
Q1: 3/3
Q2: 2/2
Q3: 1.5/3
Q4: 5/5
Q5: 4/5
Q6: 7/7
tot: 22.5/25 (90/100)

Note: Guide on how to navigate and run MATLAB scripts in final section of document

1)

Statement: For $A \in \mathbb{R}^{N\times N}$ invertible and $U, V \in \mathbb{R}^{N\times M}$, $A+UV^\top$ is invertible $\iff$ $I+V^\top A^{-1}U$ is invertible

Proof: If V or U are the zero $N \times M$ matrix, then $A + UV^\top$ is certainly invertible since A is invertible. Proceed to consider the case when V and U are non-zero $N \times M$ matrices. Assume for contradiction that $A + UV^\top$ is not invertible and $I + V^\top A^{-1}U$ is invertible. Then, $\exists x \neq 0$ such that $(A + UV^\top)x = 0$.

$$\begin{aligned}(A+UV^\top)x = 0 \iff Ax = -UV^\top x \iff x = -A^{-1}UV^\top \iff V^\top x = -V^\top A^{-1}UV^\top x \\ \iff V^\top x + V^\top A^{-1}UV^\top x = 0 \iff (I + V^\top A^{-1}U)(V^\top x) = 0\end{aligned}$$

Since $V \neq 0$ and $x \neq 0$, $V^\top x \neq 0$. So,

$$(A + UV^\top)x = 0 \iff (I + V^\top A^{-1}U) = 0$$

Which is a contradiction since $(I + V^\top A^{-1}U)$ is invertible. Therefore the **statement** is proven

Sherman-Morrison-Woodbury formula proof:

$$\begin{aligned}(A+UV^\top)^{-1}(A+UV^\top) &= (A^{-1} - AU(I+V^\top A^{-1}U)^{-1}V^\top A^{-1})(A+UV^\top) \\ &= I + A^{-1}UV^\top - AU(I+V^\top A^{-1}U)^{-1}V^\top - AU(I+V^\top A^{-1}U)^{-1}V^\top A^{-1}U \\ &= I + A^{-1}UV^\top - AU((I+V^\top A^{-1}U)^{-1} + (I+V^\top A^{-1}U)^{-1}V^\top A^{-1}U)V^\top \\ &= I + A^{-1}UV^\top - AU((I+V^\top A^{-1}U)^{-1}(I+V^\top A^{-1}U))V^\top \\ &= I + A^{-1}UV^\top - A^{-1}UV^\top \\ &= I\end{aligned}$$

Since $(A + UV^\top)$ is square (right and left inverse condition satisfied), the Sherman-Morrison-Woodbury formula is proven

[Note: Another idea for the first half of this proof, which I thought was cool! If $(A + UV^\top)$ is invertible, then its inverse exists and is $(A^{-1} - AU(I+V^\top A^{-1}U)^{-1}V^\top A^{-1})$. Hence, by construction,

$(I+V^\top A^{-1}U)^{-1}$ must exist too, i.e $(I+V^\top A^{-1}U)$ is invertible. So, the explicit construction of the inverse (SMW) implies the existence of the inverse i.e. $(A+UV^\top)$ is invertible $\iff$ $(I+V^\top A^{-1}U)$ is invertible]

2)

$\hat{s}$ is the direction of steepest decent of f at x with respect to the A-norm. Therefore,

$$\nabla f(x) \cdot \hat{s} \leq \nabla f(x) \cdot s \quad \forall s \in \mathbb{R}^N \quad \text{with} \quad |s|_A = 1 \tag{1}$$

Proceed to prove (1) by letting $|s|_A = 1$. Initially, since A SPD $\iff$ A^{-1} SPD, notice that $(A^{-1}\nabla f(x))^\top = \nabla f(x)^\top (A^{-1})^\top = f(x)^\top A^{-1}$ (A^{-1} symmetric), and,

$$|A^{-1}\nabla f(x)|_A = ((A^{-1}\nabla f(x))^\top A(A^{-1}\nabla f(x))^{\frac{1}{2}} = (\nabla f(x)^\top A^{-1}\nabla f(x))^{\frac{1}{2}}$$

Therefore,

$$\nabla f(x)^\top \hat{s} = -\frac{\nabla f(x)^\top A^{-1}\nabla f(x)}{|A^{-1}\nabla f(x)|_A} = -(\nabla f(x)^\top A^{-1}\nabla f(x))^{\frac{1}{2}} = -|A^{-1}\nabla f(x)|_A \tag{2}$$

Apply the Cauchy-Schwarz inequality for SPD matrices, $x^\top Ay \leq |x|_A|y|_A$. Notice that multiplying the right hand side of the equality by -1 flips the inequality sign; therefore, $x^\top Ay \geq -|x|_A|y|_A$. Then, since $|s|_A = 1$, and using (2),

$$-|A^{-1}\nabla f(x)|_A = -|A^{-1}\nabla f(x)|_A|s|_A \leq (A^{-1}\nabla f(x))^\top As = \nabla f(x)^\top s = \nabla f(x) \cdot s$$

Therefore, $\hat{s}$ is indeed the direction of steepest decent of f at $x \in \mathbb{R}^N$, with respect to the A-norm.

3a)

The proof of this will be done in three parts

1. For any vector $c \in \mathbb{R}^N$, $(cc^\top)$ is a symmetric $N \times N$ matrix
2. For all $n \in \mathbb{N}_0$, $B_n \in \mathbb{R}^{N\times N}$ is symmetric.
3. If B_n is positive definite then B_{n+1} is positive definite (statement of question 3a)

Part 1: $(cc^\top)^\top = (c^\top)^\top(c^\top) = (cc^\top)$

Part 2: Proceed by induction. For $n = 0$, B_0 is SPD, so B_0 symmetric. Assume the statement of part 2 is true for some $n \geq 0$, then show it is true for B_{n+1}.

$$B_{n+1} = B_n - \frac{(B_n d_n)(B_n d_n)^\top}{d_n^\top B_n d_n} + \frac{y_n y_n^\top}{y_n^\top d_n}$$

Similarly, by part 1, both $(B_n d_n)(B_n d_n)^\top$ and $y_n y_n^\top$ are symmetric, $d_n^\top B_n d_n \in \mathbb{R}$, and $y_n^\top d_n \in \mathbb{R}$. By the induction hypothesis, B_n is symmetric. This implies B_{n+1} is symmetric. Therefore, part 2 is proved true by mathematical induction.

Part 3: Given B_n positive definite (B_n is also symmetric so B_n is SPD), to show B_{n+1} is positive definite it is required to show,

$$\forall x \neq 0 \quad x^\top B_{n+1} x > 0$$

An alternative expression for $x^\top B_{n+1} x$ is

$$x^\top B_{n+1} x = x^\top B_n x - \frac{x^\top (B_n d_n)(B_n d_n)^\top x}{d_n^\top B_n d_n} + \frac{x^\top y_n y_n^\top x}{y_n^\top d_n}. \tag{3}$$

Notice that $x^\top y_n = y_n^\top x$ and $x^\top (B_n d_n) = (B_n d_n)^\top x$

$$x^\top B_n x - \frac{(x^\top B_n d_n)^2}{d_n^\top B_n d_n} + \frac{(x^\top y_n)^2}{y_n^\top d_n}$$

$y_n^\top d_n > 0 \Rightarrow y_n \neq 0$ and $d_n \neq 0$. Assume $\lambda d_n \neq x$, $\lambda \in \mathbb{R} \backslash \{0\}$, then

$$\frac{(x^\top y_n)^2}{y_n^\top d_n} \geq 0 \quad \text{and} \quad (x^\top B_n x)(d_n^\top B_n d_n) > (x^\top B_n d_n)^2 \tag{4}$$

Where (4) is from the Cauchy-Schwarz inequality for SPD matrices (B_n is SPD), $(x^\top A y)^2 \leq (x^\top A x)(y^\top A y)$, but in this case we have a strict inequality since $\lambda d_n \neq x \neq 0$. Hence,

$$x^\top B_n x - \frac{(x^\top B_n d_n)^2}{d_n^\top B_n d_n} > 0 \tag{5}$$

Hence, using both (4) and (5), the correct inequality on (3) is produced for B_{n+1} to be positive definite (i.e. $x^\top B_{n+1} x > 0 \quad \forall x \neq 0$). Assume $\lambda d_n = x$, then

$$(x^\top B_n x)(d_n^\top B_n d_n) = (x^\top B_n d_n)^2 \iff \lambda^2 (d_n^\top B_n d_n)(d_n^\top B_n d_n) = \lambda^2 (d_n^\top B_n d_n)^2$$

Therefore, substituting $x = \lambda d_n$, obtain that

$$\frac{\lambda^2 (d_n^\top y_n)^2}{y_n^\top d_n} > 0 \quad \text{and} \quad \lambda^2 d_n^\top B_n d_n - \frac{\lambda^2 (d_n^\top B_n d_n)^2}{d_n^\top B_n d_n} = 0 \tag{6}$$

Hence, using (6) in (3), the inequality produced is $x^\top B_{n+1} x > 0 \quad \forall x \neq 0$. Part 3 is proven. Therefore, assertion of the question is proven.

3b)

In Q3a part 2, it was proved that B_{n+1} is symmetric. It remains to show that for all $n \geq 0$, B_n is positive definite. Proceed by induction. Notice that B_0 is SPD. Then assume true for some $n \geq 0$, then from question 3a part 3 it was shown that B_{n+1} is positive definite. Therefore B_{n+1} is SPD by mathematical induction. So $B_{n+1} \in \mathbb{R}^{N \times N}$ must be invertible. Notice that the BFGS update for B_n can be written as follows

$$B_{n+1} = \underbrace{B_n}_{A \in \mathbb{R}^{N \times N}} + \underbrace{\begin{pmatrix} \frac{B_n d_n}{d_n^\top B_n d_n} & \frac{y_n}{y_n^\top d_n} \end{pmatrix}}_{U \in \mathbb{R}^{N \times M}} \underbrace{\begin{pmatrix} B_n d_n & y_n \end{pmatrix}^\top}_{(V)^\top \in \mathbb{R}^{M \times N}} \tag{7}$$

and B_{n+1} invertible $\iff$ (7) invertible $\iff$ $I + V^\top A U$ invertible. Hence, the conditions for the SMW formula are satisfied.

4)

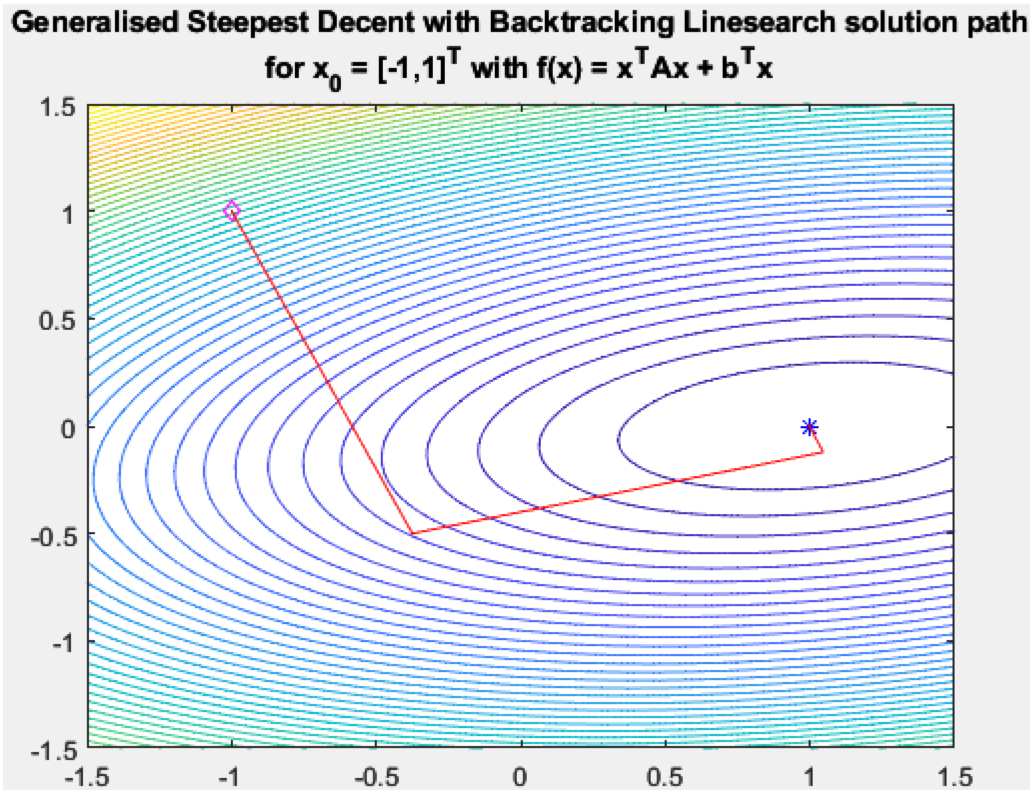

Figure 1: Plot produced by **visual.m** for solution trajectory $x_0 = [-1, 1]^\top$

Figure 2: A Table to show $||e_{k+1}||_2/||e_k||_2^2$ for each method with starting value $x_0 = [-1, 1]^\top$

Iteration	Generalised SD Backtracking LS	Newton	Steepest decent Backtracking LS
0	0.29262	0	0.29262
1	0.060426	Not computed	0.42436
2	0.32096	Not computed	0.72029
3	0.10367	Not computed	1.0446
4	Not computed	Not computed	1.773
$\vdots$	$\vdots$	$\vdots$	$\vdots$
30	Not computed	Not computed	2.1597e+05

Figure 3: A Table to show $||e_{k+1}||_2/||e_k||_2$ for each method with starting value $x_0 = [-1, 1]^\top$

Iteration	Generalised SD Backtracking LS	Newton	Steepest decent Backtracking LS
0	0.65431	0	0.65431
1	0.088409	Not computed	0.62088
2	0.041516	Not computed	0.65431
3	0.00055672	Not computed	0.62088
4	Not computed	Not computed	0.65431
$\vdots$	$\vdots$	$\vdots$	$\vdots$
30	Not computed	Not computed	0.65431

Note: Full tables can be produced by running 'question4.m'

Generalised steepest descent method (GSDM) with backtracking line search: The results from (3) suggest that

$$\lim_{k\to\infty} \frac{||e_{k+1}||_2}{||e_k||_2} = 0$$

This implies super-linear convergence. Note that for significantly worse starting values, for example $x_0 = [-50, 1000]^\top$, very few iterations were required to reach a sufficiently close solution x_* for the user ($x_0 = [-50, 1000]^\top$ took 6 iterations). This is not surprising since the method has global convergence, and super-linear convergence implies that very few iterations are required to reach an accurate solution.

Newtons method: For the function $f(x) = x^\top Ax + b^\top x$ and initial starting value x_0, one step convergence is proven.

$$\begin{aligned} x_1 &= x_0 - DF(x_0)^{-1}F(x_0) = x_0 - A^{-1}(Ax_0 + b) = x_0 - x_0 - A^{-1}b = -A^{-1}b \\ &\Rightarrow F(x_1) = F(-A^{-1}b) = A(-A^{-1}b) + b = 0 \end{aligned}$$

This one step convergence was observed when testing a variety of good and bad starting values x_0 ((2), (3) and **question4.m**) as expected. Hence, Newtons method, in this case, behaves much better than GSDM with backtracking line search.

Steepest decent with backtracking line search: The results from (3) suggest that

$$\lim_{k\to\infty} \frac{||e_{k+1}||_2}{||e_k||_2} \in (0, 1)$$

This suggests linear convergence. The problem has significantly less iterations than the Rosenbrock function from Q5. This is because the function $f(x) = x^\top Ax + b^\top x$ is significantly better conditioned than the Rosenbrock function. Since the convergence is linear, it was expected that more iterations would be required to reach an accurate solution than GSDM with backtracking line search. For a poor starting value $x_0 = [-50, 1000]^\top$, the method took 40 iterations to find a suitable approximation to x_*, compared to 31, for a better starting value $x_0 = [-1, 1]^\top$ (3) (2). The difference in iterations (9 iterations) compared to GSDM with backtracking line search (2 iterations) illustrates the impact of choosing an optimal decent direction s_k. The choice of s_k for GSDM is significantly better given a good starting choice B_0; the B_k^{-1} updates will iteratively become better approximations of $DF(x_k)^{-1} = A^{-1}$. This will then impact the convergence speeds (linear and super-linear respectively) and help to find an accurate solution to x_* in less iterations.

5)

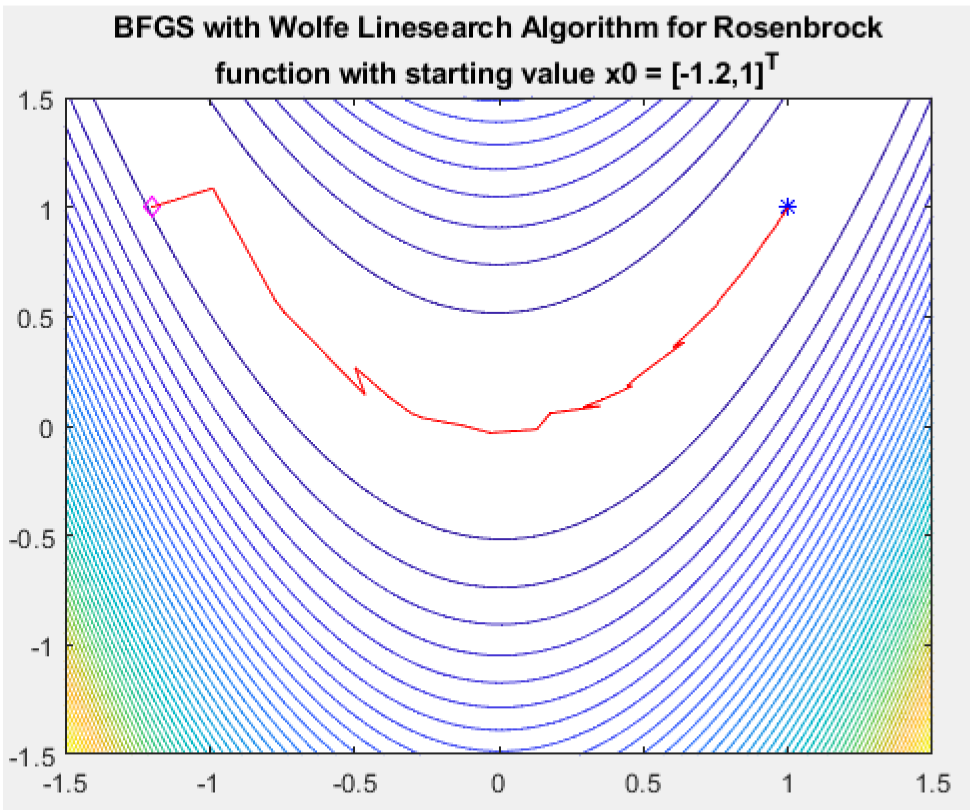

Figure 4: Plot produced by **visual.m** for solution trajectory $x_0 = [-1.2, 1]^\top$

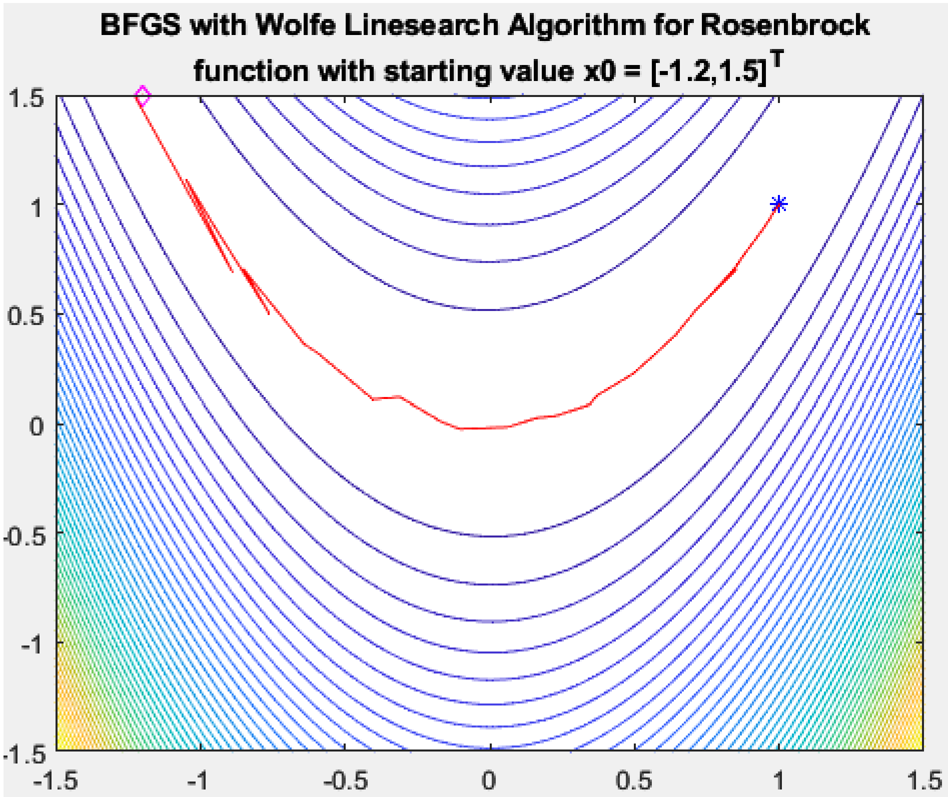

Figure 5: Plot produced by **visual.m** for solution trajectory $x_0 = [-1.2, 1.5]^\top$

```
Number of iterations: 34
Table for generalised steepest descent method with wlinesearch for x0 =[-1.2;1]^T
Table designed to show potential candidate convergence rates
```

iteration	||e(k+1)||_2/||e(k)||_2^2	||e(k+1)||_2/||e(k)||_2
0	0.41143	0.90514
1	0.45989	0.91579
2	0.54326	0.99069
3	0.51971	0.93893
4	0.5794	0.98285
5	0.58674	0.97823
6	0.60493	0.98661
7	0.61321	0.98671
8	0.59626	0.94669
9	0.64547	0.97018
10	0.62882	0.91697
11	0.69917	0.93492
12	0.71468	0.89346
13	0.92433	1.0324
14	0.8201	0.94573
15	0.82278	0.89733
16	1.034	1.0119
17	0.94614	0.93694
18	0.83467	0.77443
19	1.456	1.0462
20	1.1812	0.88794
21	1.1512	0.76841
22	1.8621	0.95509
23	1.4216	0.69639
24	2.1188	0.72279
25	2.7516	0.67847
26	2.9405	0.49193
27	11.387	0.93711
28	6.0374	0.46561
29	3.9606	0.14222
30	59.673	0.30474
31	10.185	0.01585
32	1305.8	0.032208
33	17214	0.013676

```
Final error: 1.0865e-08
```

Figure 6: Table for some potential convergence rates for the Generalised Steepest Decent Method with Wolfe Linesearch Algorithm, applied to Rosenbrock function with $x_0 = [-1.2, 1]^\top$

```
Number of iterations: 38
Table for generalised steepest descent method with wlinesearch for x0 =[-1.2;1.5]^T
Table designed to show potential candidate convergence rates
```

iteration	||e(k+1)||_2/||e(k)||_2^2	||e(k+1)||_2/||e(k)||_2
0	0.44731	1.0092
1	0.36912	0.84041
2	0.56053	1.0725
3	0.46413	0.9525
4	0.47919	0.93671
5	0.55926	1.024
6	0.52479	0.984
7	0.51749	0.95478
8	0.55855	0.98394
9	0.55365	0.95965
10	0.5707	0.94929
11	0.61472	0.97065
12	0.64178	0.98365
13	0.64907	0.97856
14	0.63498	0.93678
15	0.67368	0.93103
16	0.74598	0.95985
17	0.7382	0.9117
18	0.84984	0.95691
19	0.79796	0.85977
20	0.95803	0.88748
21	1.0297	0.84656
22	1.1574	0.80557
23	1.0573	0.59279
24	4.2697	1.4191
25	1.8274	0.86185
26	1.8302	0.74396
27	3.3778	1.0215
28	2.6726	0.82556
29	2.1142	0.53916
30	4.4507	0.61196
31	6.7324	0.56648
32	1.9596	0.093405
33	438.97	1.9544
34	39.857	0.3468
35	7.2784	0.021963
36	79.752	0.0052854
37	5.6764e+05	0.19883

```
Final error: 6.9645e-08
```

Figure 7: Table for some potential convergence rates for the Generalised Steepest Decent Method with Wolfe Linesearch Algorithm, applied to Rosenbrock function with $x_0 = [-1.2, 1.5]^\top$

Note: All figures for this question can be re-produced by running 'question5.m'

Generalised steepest descent method (GSDM) with Wolfe line search: Observe from both (4) and (5) that the trajectory direction does not always seem to make progress towards the minimum value. This may be due to the Rosenbrock function being a non-convex function. This is important because it implies that the hessian may not be SPD at every x_n, and the hessian being SPD is an assumption when computing the direction of steepest decent. Therefore the computed s_n at each iteration may not always be a direction of steepest decent. This behaviour is indicated in (6) and (7), with values of $||e_{k+1}||_2/||e_k||_2 > 1$ (making negative progress towards the minimum x_* at iteration n); perhaps more so for the starting value $x_0 = [-1.2, 1.5]^\top$, which is further away from the minimum.

From (6) and (7), observe that the sequence of convergence rates ($||e_{k+1}||_2/||e_k||_2$) seem to be pattern-less up to a large enough iteration n, where from there on, super linear convergence is observed. This behaviour is expected since from Theorem 6.1, the method is locally super-linearly convergent for large enough n.

Newtons method: In general, newtons method converges quadratically to a solution x_*. However, notice that both starting values, $x_0 = [-1.2, 1]^\top$ and $x_0 = [-1.2, 1.5]^\top$ are outside the basin of attraction of the Rosenbrock function. Therefore, Newtons method behaves badly (if starting values where chosen inside the basin of attraction, since the Rosenbrock function is quadratic, it would behave well). For the respective starting values, Newtons method (5 and 11 iterations) behaved better than GSDM with Wolfe Linesearch (34 and 38 iterations). Note that it is 'lucky' that Newtons method converges for these starting values since they are outside the basin of attraction; the more reliable choice for convergence is certainly GDSM with Wolfe Linesearch (and the convergence is still fast!).

Steepest decent with backtracking line search: Given the Rosenbrock function, for both starting values, the rate of convergence is linear.

- For $x_0 = [-1.2, 1]^\top$, the method took 8156 iterations to find an accurate solution (final error $= 1.8601e-05$), with the convergence factor $\sigma \approx 0.99922$.
- For $x_0 = [-1.2, 1.5]^\top$, the method took > 10000 iterations to converge, with a convergence factor $\sigma \approx 0.99922$.

These results can be seen in more detail by running **question5.m**. The convergence factor being close to 1 implies that there are very small changes in the updated solution x_{k+1} at each iteration. Despite having global convergence, steepest decent behaves poorly for functions, like the Rosenbrock function observed here, that are not well conditioned. The Rosenbrock function has long and thin contour lines,

$$\frac{d}{d\alpha}(f(x_{k+1}))\big|_{\alpha=\alpha_k} = \frac{d}{d\alpha}(f(x_k - \alpha\nabla f(x_k)))\big|_{\alpha=\alpha_k} = \nabla f(x_{k+1}) \cdot \nabla f(x_k) \approx 0,$$

and the gradients of f at x_{k+1} and x_k are certainly at least approximately orthogonal (in practice) at every iteration. This helps justify the high iteration zig-zag behaviour observed.

It was expected for GSDM with Wolfe Linesearch to perform better than Steepest decent with backtracking linesearch. One of the reasons for this is that the Wolfe linesearch algorithm guarantees that the computed B_{n+1}^{-1} (equation 3 in assignment 1) will be SPD (it is required for α to satisfy an extra constraint, the curvature condition). The ability to choose a better α certainly contributes to the increase in speed of convergence (helps reduce the orthogonal zig-zag behaviour observed before). The number of iterations required for GSDM with Wolfe Linesearch to converge to a solution of sufficient accuracy, for the respective starting values, was 34 and 38, compared to 8156 and > 10000.

6a)

Statement: For all $0 \leq k \leq n$,

$$B_{k+1}d_j = y_j\,, \quad \text{for all } 0 \leq j \leq k, \quad \text{and} \quad d_k^\top A d_j = 0, \quad \text{for all } 0 \leq j < k. \tag{8}$$

Initially, notice that since $B_0 = I$ and $y_k^\top d_k > 0 \quad \forall k \leq n$, B_k is SPD by Q3. Proceed to compute some results that will be useful for proving the **statement**. Notice that $f(x) = \frac{1}{2}x^\top Ax + b^\top x + c$ is quadratic, and the BFGS method is a Quasi-Newton method. This means that the secant condition, described in more detail at the beginning of chapter 6 in the lecture notes, is satisfied.

$$B_{j+1}(x_{j+1} - x_j) = \nabla f_{j+1} - \nabla f_j \iff B_{j+1}d_j = y_j \tag{9}$$

Using the same idea from PS2 Q3a,

$$\begin{aligned} f(x+h) &= \frac{1}{2}(x+h)^\top A(x+h) + b^\top(x+h) + c \\ &= \frac{1}{2}(x^\top Ax + 2x^\top Ah + h^\top Ah) + b^\top x + b^\top h + c \\ &= f(x) + (Ax+b)^\top h + \frac{1}{2}h^\top Ah + 0 \end{aligned}$$

Therefore $\nabla f(x) = Ax + b$ and $\nabla^2 f(x) = A$. Using $y_j = \nabla f(x_{j+1}) - \nabla f(x_j)$, and the secant condition (9)

$$B_{j+1}d_j = y_j = (Ax_{j+1} + b) - (Ax_j + b) = A(x_{j+1} - x_j) = Ad_j \tag{10}$$

Notice that (10) makes intuitive sense considering the mean value theorem. Also,

$$d_j = x_{j+1} - x_j = x_j + \alpha_j s_j - x_j = \alpha_j s_j \tag{11}$$

From PS3 Q3a, for any choice of decent direction s_j, the $(j+1)$th iterate obtained with exact line search satisfies

$$\nabla f(x_{j+1})^\top s_j = 0 \tag{12}$$

Proceed to prove the **statement** using induction on k. For $k = 0$, by the secant condition (9), $B_1 d_0 = y_0$. Notice that the second part of (8) is not defined for $k = 0$, so use $k = 1$,

$$\begin{aligned} d_1^\top A d_0 \quad &= \quad d_1^\top y_0 && \text{Applying } 10 \\ &= \quad d_1^\top B_1 d_0 && \text{Applying } 9 \\ &= \quad (\alpha_1 s_1)^\top B_1 d_0 && \text{Applying } 11 \\ &= \quad \alpha_1 (B_1 s_1)^\top d_0 && \text{Since } \alpha_1 \in \mathbb{R}, \text{ and } B_1 \text{ SPD so } s_1^\top B_1 = (B_1 s_1)^\top \\ &= \quad -\alpha_1 \nabla f(x_1)^\top (s_0 \alpha_0) && \text{Since } s_1 = -B_1^{-1}\nabla f(x_1) \text{ and } 11 \\ &= \quad -\alpha_1 \alpha_0 \nabla f(x_1)^\top s_0 = 0 && \text{Applying } 12 \end{aligned} \tag{13}$$

Therefore the **statement** is proved true for the base case. Assume the **statement** is true for some $0 < k \leq n$, then show it is true for $k+1$. Using the BFGS update formula, and multiplying through by d_j, obtain

$$B_{k+1}d_j = B_k d_j - \frac{(B_k d_k)(B_k d_k)^\top d_j}{d_k^\top B_k d_k} + \frac{y_k y_k^\top d_j}{y_k^\top d_n} \tag{14}$$

Using the induction hypothesis, $B_k d_j = y_j$, and (10) $Ad_k = y_k$, (14) becomes

$$B_{k+1}d_j = B_k d_j - \frac{(B_k d_k) d_k^\top y_j}{d_k^\top B_k d_k} + \frac{y_k (Ad_k)^\top d_j}{y_k^\top d_n}$$

Then, by the induction hypothesis ($d_k^\top Ad_j = 0$), (10) $y_j = Ad_j$, and A SPD so $(Ad_k)^\top = d_k^\top A$

$$B_{k+1}d_j = B_k d_j - \frac{(B_k d_k) d_k^\top Ad_j}{d_k^\top B_k d_k} + \frac{y_k d_k^\top Ad_j}{y_k^\top d_n} = B_k d_j = y_j \tag{15}$$

Then, in a similar fashion as before,

$$\begin{aligned}
d_{k+1}^\top Ad_j &= d_{k+1}^\top y_j && \text{Applying 10} \\
&= d_{k+1}^\top B_{k+1} d_j && \text{Applying the induction hypothesis} \\
&= (\alpha_{k+1} s_{k+1})^\top B_{k+1} d_j && \text{Applying 11} \\
&= \alpha_{k+1}(B_{k+1}s_{k+1})^\top d_0 && \text{Since } \alpha_{k+1} \in \mathbb{R}\text{, and } B_{k+1} \text{ SPD so } s_{k+1}^\top B_{k+1} = (B_{k+1}s_{k+1})^\top \\
&= -\alpha_{k+1}\nabla f(x_{k+1})^\top (s_j \alpha_j) && \text{Since } s_{k+1} = -B_{k+1}^{-1}\nabla f(x_{k+1}) \text{ and 11} \\
&= -\alpha_{k+1}\alpha_j \nabla f(x_{k+1})^\top s_j
\end{aligned} \tag{16}$$

Proceed to prove that $\nabla f(x_{k+1})^\top s_j = 0$ for all $0 \leq j < k+1$. If $j = k$, then by (12), $\nabla f(x_{k+1})^\top s_j = 0$. For $j < k$, using $x_{k+1} = x_k + \alpha_k s_k$, and that A is SPD,

$$\nabla f(x_{k+1})^\top s_j = (Ax_{k+1} + b)^\top s_j = (A(x_k + \alpha_k s_k) + b)^\top s_j = \nabla f(x_k)^\top s_j + \alpha_k s_k^\top A s_j \tag{17}$$

Then, using the induction hypothesis, $d_k^\top Ad_j = 0$, and rearranging (11),

$$\alpha_k s_k^\top A s_j = \alpha_k \left(\frac{d_k^\top}{\alpha_k}\right) A \left(\frac{d_j}{\alpha_j}\right) = \frac{1}{\alpha_j} d_k^\top A d_j = 0 \tag{18}$$

Also,

$$\begin{aligned}
\nabla f(x_k)^\top s_j &= -(B_k s_k)^\top s_j && \text{Since } \nabla f(x_k) = -B_k s_k \\
&= -s_k^\top B_k s_j && \text{Since } B_k \text{ SPD and } s_k^\top B_k = (B_k s_k)^\top \\
&= -\frac{1}{\alpha_j} s_k^\top y_j && \text{Applying (11) and the induction hypothesis } B_k d_j = y_j \\
&= -\frac{1}{\alpha_j} s_k^\top A d_j && \text{Applying (10)} \\
&= -\frac{1}{\alpha_j \alpha_k} d_k^\top A d_j = 0 && \text{Applying (11) and the induction hypothesis } d_k^\top Ad_j = 0
\end{aligned} \tag{19}$$

Therefore combining (19) $\nabla f(x_k)^\top s_j = 0$ with (18) implies that (17) $\nabla f(x_{k+1})^\top s_j = 0$ for $j < k$. Hence, since also true for $j = k$, by (16)

$$d_{k+1}^\top Ad_j = -\alpha_{k+1}\alpha_j \nabla f(x_{k+1})^\top s_j = 0 \quad \text{for all } 0 \leq j < k+1. \tag{20}$$

Combining both (15) and (20), the **statement** is proven by mathematical induction

6b)

Of course, if $\nabla f(x_k) = 0$ for any $k < N$, then the method converges in less than N iterations (i.e. at most N iterations). To prove the final case, the idea is to prove that

$$\nabla f(x_N)^\top s_j = 0 \quad \text{for all} \quad 0 \leq j < N \tag{21}$$

This can be done recursively using ideas from Q6a. The idea is as follows, for $N-1 = j$, by (12), (21) is satisfied. For $N-1 > j$, from (17), (18), and finally (13),

$$\nabla f(x_N)^\top s_j = \nabla f(x_{N-1})^\top s_j + \frac{1}{\alpha_j} d_{N-1}^\top A d_j = \nabla f(x_{N-1})^\top s_j = \ldots = \nabla f(x_1)^\top s_j = \nabla f(x_1)^\top s_0 = 0$$

More formally, this idea can be constructed using the hint given in the question,

$$x_N = x_0 + \alpha_0 s_0 + \alpha_1 s_1 + \ldots + \alpha_{N-1} s_{N-1} = x_1 + \alpha_1 s_1 + \ldots + \alpha_{N-1} s_{N-1}$$

Then, compute

$$\begin{aligned}
\nabla f(x_N)^\top s_j &= (A(x_1 + \alpha_1 s_1 + \ldots + \alpha_{N-1} s_{N-1}) + b)^\top s_j \\
&= \nabla f(x_1)^\top s_j + \alpha_1 s_1^\top A s_j + \ldots + \alpha_{N-1} s_{N-1}^\top A s_j \\
&= \nabla f(x_1)^\top s_0 + \sum_{i=1}^{N-1} \alpha_i s_i^\top A s_j \qquad (22) \\
&= \nabla f(x_1)^\top s_0 + \sum_{i=1}^{N-1} \frac{\alpha_i}{\alpha_j \alpha_i} d_i^\top A d_j = 0 \qquad (23)
\end{aligned}$$

Where (22) is true since for the statement $\nabla f(x_1)^\top s_j$, j must satisfy $0 \leq j < 1$, so $s_j = s_0$. (23) is true since at each step, $i > j$ so $d_i^\top A d_j = 0$ by Q6a, and by (13), $\nabla f(x_1)^\top s_0 = 0$. Therefore (21) holds. So, it has been proven that $\nabla f(x_N)$ is orthogonal to N many linearly independent vectors $s_j \neq 0$ that span $\mathbb{R}^N$. This implies that $\nabla f(x_N) = 0$. Hence, the generalised steepest descent method with BFGS updates and exact line search converges in at most N iterations.

Using (8), (10) and (11),

$$A d_j = y_j = B_{k+1} d_j \Rightarrow \alpha_j A s_j = \alpha_j B_{k+1} s_j$$

For $k = N-1$, since the decent directions are linearly independent and $\alpha_j \neq 0$ ($\nabla f(x_k) \neq 0$ for $k < N$)

$$A s_j = B_N s_j \Rightarrow A[s_0, \ldots, s_{N-1}] = B_N[s_0, \ldots, s_{N-1}] \Rightarrow A = B_N$$

Since $[s_0, \ldots, s_{N-1}]$ is a $N \times N$ invertible matrix (linearly independent columns).

6c)

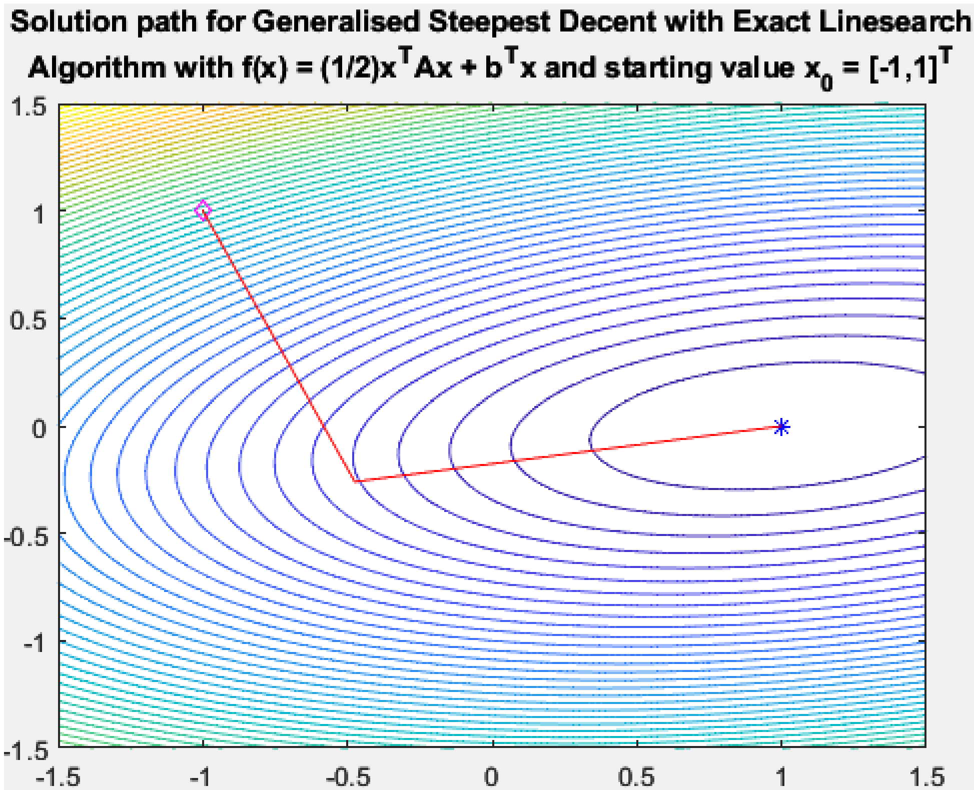

Figure 8: Plot produced by **visual.m** for solution trajectory of GSDM with Exact Linesearch for starting value $x_0 = [-1, 1]^\top$ and function $f(x)$ defined in question 4 of assignment 1.

GSDM with Exact Linesearch for 10 random starting values in range -1.1 to 1.1

Starting value x_0		Number of iterations to reach solution
0.69239	0.89274	2
-0.82063	0.90943	2
0.29119	-0.88541	2
-0.4873	0.10314	2
1.0065	1.0228	2
-0.75325	1.0353	2
1.0058	-0.032174	2
0.66062	-0.78785	2
-0.17213	0.91462	2
0.64286	1.0109	2

Figure 9: GSDM with Exact Linesearch for 10 uniformly chosen random starting values x_0 with components in range -1.1 to 1.1

GSDM with Exact Linesearch for 10 random starting values in range -2 to 2

Starting value x_0		Number of iterations to reach solution
0.62296	-1.8572	2
1.3965	1.736	2
0.71494	1.031	2
0.97253	-0.43109	2
0.62191	-1.3153	2
0.82418	-1.8727	2
-0.89231	-1.8153	2
-1.6115	1.2938	2
0.77931	-0.7316	2
1.8009	-1.8622	2

Figure 10: GSDM with Exact Linesearch for 10 uniformly chosen random starting values x_0 with components in range -2 to 2

GSDM with Exact Linesearch for 10 random starting values in range -10 to 10

Starting value x_0		Number of iterations to reach solution
-1.2251	-2.3688	2
5.3103	5.904	2
-6.2625	-0.20471	2
-1.0883	2.9263	2
4.1873	5.0937	2
-4.4795	3.5941	2
3.102	-6.7478	2
-7.62	-0.032719	2
9.1949	-3.1923	2
1.7054	-5.5238	2

Figure 11: GSDM with Exact Linesearch for 10 uniformly chosen random starting values x_0 with components in range -10 to 10

Plot and similar tables can be reproduced in file 'question6.m', with additional information

Figures (9), (10), and (11) document the number of iterations taken for GSDM with Exact Linesearch to converge given a range of good to increasingly bad random chosen starting values x_0. Notice that for all tested starting values, the number of iterations taken is indeed at most $N = 2$.

How to run/navigate the question#.m MATLAB scripts

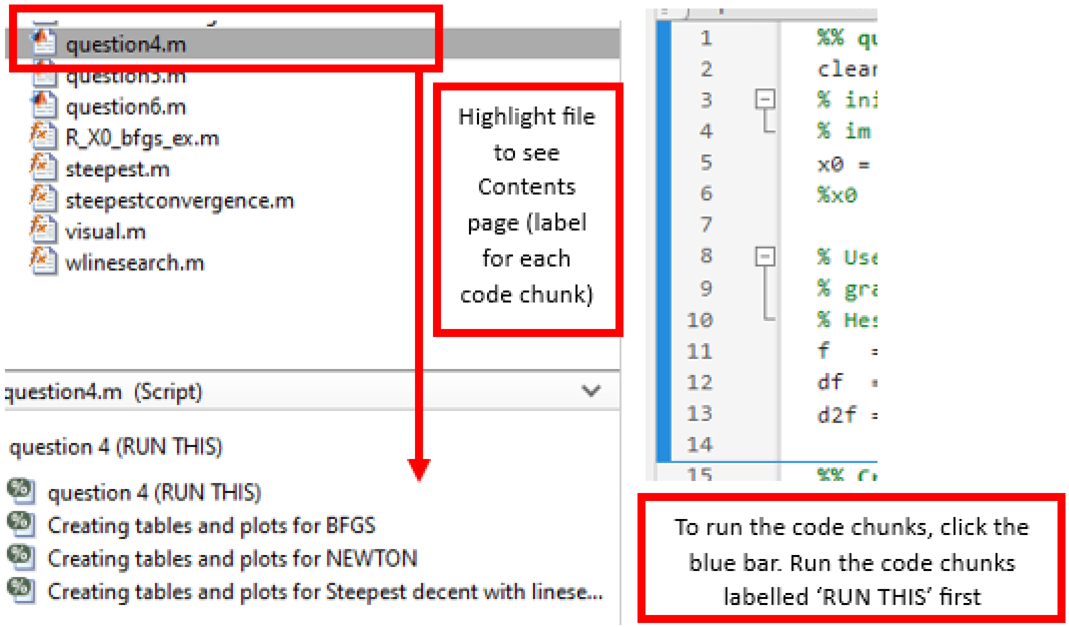

Figure 12: MATLAB screenshots with captions designed to aid reader when navigating/running **question#.m**

Functions used in main question#.m scripts

Algorithms

1. **bfgs.m:** Generalized Steepest descent method (Alg. 4.3) with backtracking line search (Alg. 4.1)

2. **bfgs_ex.m:** Generalized Steepest descent method (Alg. 4.3) with exact line search

3. **bfgsw.m:** Generalized Steepest descent method (Alg. 4.3) with wlinesearch (Alg. 6.1)

4. **steepest.m:** Steepest descent method (Alg. 4.2) with backtracking line search (Alg. 4.1)

5. **newton.m:** Newton method for finding gradient df = 0

Linesearch Algorithms

1. **linesearch.m:** Backtracking line search algorithm

2. **wlinesearch.m:** implementation of wlinesearch (alg. 6.1)

3. **exact_linesearch.m:** function to compute alpha using exact linesearch

Convergence, Tables and Post Processing Visualisation

1. **makeconvergencetable.m:** Function to make the testing convergence tables for all the algorithms tested
2. **newtonconvergence.m:** Function to compute rates of convergence, number of iterations, and final error for newtons method
3. **steepestconvergence.m:** Function to compute rates of convergence, number of iterations, and final error for steepest decent method with linesearch
4. **visual.m:** Plot contour lines for the objective function f and the solution path of a particular sequence of iterates
5. **bfgs_ex_convergence.m:** function to compute Generalized Steepest descent method (Alg. 4.3) with exact line search convergence information: number of iterations, convergence table and final error
6. **R_X0_bfgs_ex.m:** function to compute Generalized Steepest descent method (Alg. 4.3) with exact line search for random starting values x0 in range of [min,max], with option to generate
    - convergence information: number of iterations, convergence table and final error for each randomly chosen x0