MA40050 Numerical Optimisation and Large–Scale Systems
Assessed Coursework 2022–23

General instructions

Set: Wednesday 22 nd March 2023

Due: Wednesday 26 th April 2023 at 19:00 on Moodle

Estimated time required: 10-15 hours, on the assumption that you have studied and understood your lecture notes, and have done the problem sheets for this unit! If you have not done this then the assignment may take you longer.

Submission: Upload your coursework through the Moodle page of the course by the deadline. The submission should be a single zip file following the naming convention

```
FirstName_LastName.zip
```

This zip file should contain all MATLAB files required for running your code, including any relevant MATLAB files you downloaded from the course Moodle page. The zip file should also include a **single PDF file** with the naming convention

```
written_FirstName_LastName.pdf
```

which contains your answers to the **written** part and the plots from Questions 4, 5 and 6(c). The zip file should have **no folders** inside it.

Conditions: You should not discuss the details of your work with anyone else. The work which you hand in must be your own. You should be prepared to explain anything which you write to an examiner if asked to do so. In particular, if it is discovered that all or part of your code or your written work has been copied, both parties involved risk a severe penalty and might lose all their marks on the assignment.
You may use any results which have been proved in lectures or on problem sheets, provided you quote them clearly. Cite any sources that you use to complete this assignment.

Value: This assignment is worth 25% of the total assessment for MA40050. In the questions below, the notation **[marks]** indicates the marks available for each part out of a total of 25.

Length: There is no minimum or maximum length for this assignment; in marking emphasis will be placed upon clear argument and precision.
Answer all the questions and show your working clearly. In particular, explain your derivations explicitly and state clearly which results you use from the lectures and problems sheets. Make sure your work is **well-presented** and **readable**, in particular, use **full sentences**. The file name and function specifications in the programming

questions need to be followed **very precisely**. I will run multiple tests, and if your code fails due to **incorrect filenames, incorrect order of parameters** etc., **you will lose marks** for the failed tests. **I will reduce your overall mark by up to five marks if anything is unclear, unreadable, incoherent, not commented, not labelled, not concise or not well written.**

Support and advice: Contact the unit convenor via email (`tab73@bath.ac.uk`), or during office hours (Wednesdays at 11:15-12:05).

Feedback: You will receive feedback within a maximum of three semester weeks following the submission deadline. The feedback will consist of your marked work and an overall feedback document commenting on the assessment.

Late submission of coursework: If there are valid circumstances preventing you from meeting the deadline, your Director of Studies may grant you an extension to the specified submission date, if it is requested before the deadline. Forms to request an extension are available on SAMIS.

- If you submit a piece of work after the submission date, and no extension has been granted, the maximum mark possible will be the pass mark.
- If you submit work more than five working days after the submission date, you will normally receive a mark of 0 (zero), unless you have been granted an extension.

Academic integrity statement: Academic misconduct is defined by the University as "the use of unfair means in any examination or assessment procedure". This includes (but is not limited to) cheating, collusion, plagiarism, fabrication, or falsification. The University's Quality Assurance Code of Practice, QA53 Examination and Assessment Offences, sets out the consequences of committing an offence and the penalties that might be applied.

Let $f \in C^2(\mathbb{R}^N)$. Consider applying the generalised steepest descent method with starting point $x_0 \in \mathbb{R}^N$ and search direction

$$s_n := -B_n^{-1} \nabla f(x_n) \tag{1}$$

to find a local minimizer of f in $\mathbb{R}^N$.

To define B_n, for all $n \geq 0$, let $B_0 \in \mathbb{R}^{N \times N}$ be a given symmetric positive definite (SPD) matrix and let $d_n := x_{n+1} - x_n$ and $y_n := \nabla f(x_{n+1}) - \nabla f(x_n)$, where $(x_n)_{n \geq 0}$ is the sequence of iterates produced by the algorithm. We define B_{n+1} to be the BFGS update of B_n, i.e.

$$B_{n+1} = B_n - \frac{(B_n d_n)(B_n d_n)^T}{d_n^T B_n d_n} + \frac{y_n y_n^T}{y_n^T d_n}. \tag{2}$$

In the assignment, we will use different line search algorithms in conjunction with this generalised steepest descent method (1) & (2). If the Wolfe line search from lectures (Algorithm 6.1) is used, then this is the BFGS method from lectures (Algorithm 6.2).

1. Let $A \in \mathbb{R}^{N \times N}$ be invertible and let $U, V \in \mathbb{R}^{N \times M}$ be such that $I + V^T A^{-1} U$ is invertible, with $M, N \in \mathbb{N}$. Prove that then $A + UV^T$ is invertible and

   $$\left(A + UV^T\right)^{-1} = A^{-1} - A^{-1} U (I + V^T A^{-1} U)^{-1} V^T A^{-1}.$$

   This is the Sherman-Morrison-Woodbury (SMW) formula given in lectures. **[3]**

2. Let $A \in \mathbb{R}^{N \times N}$ be SPD. Recall that for $x \in \mathbb{R}^N$ we defined the A-norm: $|x|_A = (x^T A x)^{1/2}$. Consider

   $$\widehat{s} = -\frac{A^{-1} \nabla f(x)}{|A^{-1} \nabla f(x)|_A}.$$

   Prove that $\widehat{s}$ is the direction of steepest descent of f at $x \in \mathbb{R}^N$, with respect to the A-norm.

   [*Hint:* Use the following Cauchy-Schwarz inequality for SPD matrices A:

   $$x^T A y \leq |x|_A |y|_A. \qquad]$$

   [2]

3. Suppose that $B_0 \in \mathbb{R}^{N \times N}$ is SPD and that B_{n+1}, for $n \geq 0$, is recursively defined by (2). Assume further that $y_n^T d_n > 0$, for all $n \geq 0$.

   (a) Prove that if B_n is positive definite then B_{n+1} is also positive definite.

   (b) Prove that, for all $n \geq 0$, B_{n+1} is invertible and we can apply the SMW formula to it.

   [*Hint:* Use a proof by induction to verify the assumptions in Question 1.]

   [3]

For the rest of the assignment, you may use without proof that

$$B_{n+1}^{-1} = (I - \rho_n d_n y_n^T) B_n^{-1} (I - \rho_n y_n d_n^T) + \rho_n d_n d_n^T, \quad \text{where} \quad \rho_n = \frac{1}{y_n^T d_n}. \tag{3}$$

[I recommend that you do **not** try to actually check that this is the inverse and that you do **not** try to actually apply the SMW formula. These are very tedious calculations.]

4. Implement in Matlab the generalised steepest descent method (Algorithm 4.3 of the Lecture Notes) with backtracking line search (Algorithm 4.1 of the Lecture Notes), with B_0^{-1} a given SPD matrix and with B_{n+1}^{-1}, $n \geq 0$, as defined in (3). You may use and modify the code that you produced for Problem Sheets 3 or my model solutions for those codes available on Moodle.

   Implement this in the function file `bfgs.m` beginning with the line:

   ```
   function [x,n] = bfgs(f,df,B,xn,theta,tol)
   ```

   where `f` $= f(x)$, `df` $= \nabla f(x)$ are the function handles to the objective function and its gradient, resp., `B` $= B_0 \in \mathbb{R}^{N \times N}$, `xn` $= x_0 \in \mathbb{R}^N$, `theta` $= \theta_{sd} \in \mathbb{R}$ and `tol` $= \varepsilon > 0$ is a user-defined tolerance. The iterations should stop when $|\nabla f(x_n)| \leq \varepsilon$. You should return an array `x` containing all the iterates $x_0, x_1, \ldots, x_n$ and the number of iterations n, required to converge, as `n`. Comment your code and make sure it works correctly.

   Apply your code to the quadratic function $f(x) = \frac{1}{2} x^T A x + b^T x$ on $\mathbb{R}^2$ with

   $$A = \begin{pmatrix} 2 & -1 \\ -1 & 10 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} -2 \\ 1 \end{pmatrix}.$$

   Choose $x_0 = (-1, 1)^T$, $B_0 = I$, $\theta_{sd} = 0.1$ and $\varepsilon = 10^{-5}$. Visualise the solution path for $x_0 = (-1, 1)^T$ using the postprocessing function `visual.m` provided in the model codes for the problem sheets. The plot should be included in your written answers in the PDF file.

   Compare the speed of convergence to that of Newton's method and steepest descent method, applied to the same objective function and starting from the same initial guess. Experiment also with other choices of initial guesses x_0. Comment on the results. This commentary should be included in your written answers in the PDF file.

Add the plot of the solution path for $x_0 = (-1, 1)^T$ produced by your code and visualised by `visual.m`.	**[5]**

5. Implement in Matlab the Wolfe line search algorithm in Algorithm 6.1 of the Lecture Notes. Implement this in the function file `wlinesearch.m` beginning with the line:

   ```
   function alpha = wlinesearch(f,df,sn,xn,theta_sd,theta_c)
   ```

where `f` $= f(x)$, `df` $= \nabla f(x)$ are the function handles to the objective function and its gradient, resp., `sn` is the current search direction, `xn` is the current iterate, `theta_sd` $= \theta_{sd} \in \mathbb{R}$ and `theta_c` $= \theta_c \in \mathbb{R}$ are two user-defined parameters such that $0 < \theta_{sd} < \theta_c < 1$. The output of the function should be a steplength `alpha` $= \alpha > 0$ that satisfies the Wolfe conditions

$$f(x_n + \alpha s_n) \leq f(x_n) + \theta_{sd}\, \alpha \nabla f(x_n)^T s_n \quad \text{and} \quad \nabla f(x_n + \alpha s_n) \cdot s_n \geq \theta_c \nabla f(x_n)^T s_n \,.$$

Comment your code and make sure it works correctly.

Use `wlinesearch.m` instead of backtracking line search in the `bfgs.m` code you wrote in Question 4 and apply this to find the minimum of the Rosenbrock function

$$f(x) = (1 - x_1)^2 + 10(x_2 - x_1^2)^2$$

using starting guess $x_0 = (-1.2, 1)^T$, $B_0 = I$, $\theta_{sd} = 0.1$, $\theta_c = 0.9$ and $\varepsilon = 10^{-5}$. Visualise the solution path for $x_0 = (-1.2, 1)^T$ and $x_0 = (-1.2, 1.5)^T$ using the postprocessing function `visual.m` provided in the model codes for the problem sheets. Both plots should be included in your written answers in the PDF file.

Compare the speed of convergence to that of Newton's method and steepest descent method, applied to the same objective function and starting from the same initial guess. Repeat the experiment for $x_0 = (-1.2, 1.5)^T$. Comment on the results. This commentary should be included in your written answers in the PDF file.

Add the plot of the solution path for $x_0 = (-1.2, 1)^T$ and $x_0 = (-1.2, 1.5)^T$ produced by your code and visualised by `visual.m`.

[5]

6. Let $f(x) = \frac{1}{2} x^T A x + b^T x + c$, with symmetric positive definite $A \in \mathbb{R}^{N \times N}$, $b \in \mathbb{R}^N$ and $c \in \mathbb{R}$. Furthermore, let $B_0 = I$ and apply the BFGS method with exact line search. You may use without proof any results from the problem sheets.

   (a) Let $n < N$, and suppose that $\nabla f(x_k) \neq 0$ and $y_k^T d_k > 0$, for all $k \leq n$. Assume also that the descent directions $\{s_k\}_{k \leq n}$ are linearly independent. Prove that, for all $0 \leq k \leq n$,

   $$B_{k+1} d_j = y_j \,, \quad \text{for all } 0 \leq j \leq k, \quad \text{and} \quad d_k^T A d_j = 0, \quad \text{for all } 0 \leq j < k. \tag{4}$$

   [*Hint:* Use induction on k.]

   (b) Deduce that the generalised steepest descent method with BFGS updates and exact line search converges in at most N iterations. If $\nabla f(x_k) \neq 0$, for all $k < N$, then $B_N = A$.

   [*Hint:* Notice that $x_N = x_0 + \alpha_0 s_0 + \alpha_1 s_1 + \ldots + \alpha_{N-1} s_{N-1}$ and, from part (a), use the fact that the descent directions $\{s_k\}_{k \leq n}$ are linearly independent.]

   (c) Now, starting from `bfgs.m`, implement in Matlab a new function file `bfgs_ex.m` which uses exact line search instead of backtracking line search at each step. It should begin with the line:

```
function [x,n] = bfgs_ex(f,df,df2,B,xn,tol)
```

where `f` $= f(x)$, `df` $= \nabla f(x)$ and `df2` $= \nabla^2 f(x)$ are the function handles to the objective function, its gradient and its Hessian, resp., `B` $= B_0 \in \mathbb{R}^{N\times N}$, `xn` $= x_0 \in \mathbb{R}^N$ and `tol` $= \varepsilon > 0$ is a user-defined tolerance. The iterations should stop when $|\nabla f(x_n)| \leq \varepsilon$. You should return an array `x` containing all the iterates $x_0, x_1, \ldots, x_n$ and the number of iterations n, required to converge, as `n`. Apply the new code `bfgs_ex.m` to minimise the quadratic function in Question 4.

Verify numerically that, for a number of different starting guesses x_0, the BFGS method with exact line search always converges in at most 2 iterations. Document your experiments carefully.

Add the plot of the solution path for $x_0 = (-1, 1)^T$ produced by your code and visualised by `visual.m`.

[7]