Damped Newton's Method, Fixed Step Size, and Armijo Backtracking Search

Matthew Sun

Abstract

The first purpose of this assignment was to find and classify the stationary points of three objective functions with vector inputs and scalar outputs. The Jacobian matrices (which happen to be just a single gradient 1-form for these functions) were found for these functions and used to solve for the stationary points. The Hessian matrix for each stationary point was used to classify them as a strict local minimizer, a strict local maximize, a saddle point, or inconclusive. The second purpose of this assignment was to implement line search with a fixed step size, using Armijo backtracking, and using damped Newton's method to find a local minimizer. Results were evaluated based on whether the algorithm converged and if so, how many iterations it took. It was found that fixed step size and backtracking search were able to converge for all functions but took significantly longer for the third whereas damped Newton's could not converge for the first two functions but outperformed the two other algorithms in estimating the third function.

1 Introduction

This assignment had two objectives, both of which concern three functions with vector inputs and scalar outputs: $f_1(\vec{w}) = ((w_1 + 1.21) - 2(w_2 - 1))^4 + 64(w_1 + 1.21)(w_2 - 1)$, $f_2 = 2w_2^3 - 6w_2^2 + 3w_1^2 w_2$, and $f_3 = 100(w_2 - w_1^2)^2 + (1 - w_1)^2$. The first objective was to find the stationary points of each function using the Jacobian matrix, the Hessian matrix at the stationary point, the eigenvalues of the Hessian, and classify each stationary point. The second objective was to estimate a local minimizer for each function using a fixed step size line search, Armijo backtracking line search, and line search with damped Newton's method all using a starting estimate of $\vec{w}_0 = \begin{bmatrix} -1.2 \\ 1.0 \end{bmatrix}$.

1.1 Jacobian Matrix and Hessian Matrix

The scientific question to be explored is: What are the stationary points for each of the given functions, what are the types of the stationary points? The Jacobian matrices for the functions will be used to solve for the stationary points. The eigenvalues of the Hessian matrices for the functions will be used to classify each stationary point as a strict local minimizer, strict local maximizer, saddle point, or inconclusive.

The Jacobian matrix for a function with a vector input is a matrix comprising all of its first partial derivatives. Suppose a function $f : \mathbb{R}^n \to \mathbb{R}^m$ which has inputs $\vec{x} \in \mathbb{R}^n$ and outputs $f(\vec{x}) \in \mathbb{R}^m$. The Jacobian matrix J of this function would be a $m \times n$ matrix where each entry $J_{ij} := \frac{\partial f_i}{\partial x_j}$. This can also be thought of as stacking the row matrix of the partial derivatives of each $f(\vec{x})$. These row objects will be called the gradient 1-form or just the gradient. In matrix notation the Jacobian is

$$J_f := \begin{bmatrix} \underline{\nabla} f_1 \\ \underline{\nabla} f_2 \\ \vdots \\ \underline{\nabla} f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{1}$$

where $\underline{\nabla} f_i$ is the gradient. In the case of this assignment, the functions have scalar outputs so the Jacobian will only be a $1 \times n$ matrix. The Jacobian matrix at any given vector $\vec{v} \in \mathbb{R}^n$ can be calculated by taking the product between each gradient $\underline{\nabla} f_i$ and $\vec{v}$. Similarly to scalar output functions the stationary points for $f(\vec{x})$ can be found by solving for $\vec{x}$ where $J_f(\vec{x}) = 0_{m,n}$.

The Hessian matrix for a function with a vector input and scalar output is a square symmetric matrix that contains all partial second derivatives. Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ which has inputs $\vec{x} \in \mathbb{R}^n$ and outputs $f(\vec{x}) \in \mathbb{R}$. The Hessian matrix H will be a $n \times n$ matrix with entries $h_{ij} := \frac{\partial^2 f}{\partial x_i \partial x_j}$. Note that the order of differentiation does not matter meaning $h_{ij} = h_{ji}$ so consequently H is symmetric. H at a certain $\vec{v}$ can be written as $\underline{\nabla}^2 f(\vec{v})$. In matrix form, this is

$$\underline{\nabla}^2 f(\vec{v}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\vec{v}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\vec{v}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\vec{v}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\vec{v}) & \frac{\partial^2 f}{\partial x_2^2}(\vec{v}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\vec{v}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\vec{v}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\vec{v}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\vec{v}) \end{bmatrix} \tag{2}$$

The eigenvalues of the Hessian can be used to classify stationary points of the function by providing information about the behaviour of the function in the local region of the stationary point. This is an extension of the second derivative test for scalar output functions. Since H is symmetric the eigenvalues are guaranteed to be real numbers. There are five possible configurations that need to be considered regarding the values of all eigenvalues of H. H is positive definite, written as $H \succ 0$ if all eigenvalues $\lambda_i > 0$. H is positive semidefinite, written as $H \succcurlyeq 0$ if all eigenvalues $\lambda_i \geqslant 0$. H is indefinite if some eigenvalues $\lambda_i > 0$ and some eigenvalues $\lambda_j > 0$. H is negative definite, written as $H \prec 0$ if all eigenvalues $\lambda_i < 0$. H is negative semidefinite, written as $H \preccurlyeq 0$ if all eigenvalues $\lambda_i \leqslant 0$. Given some stationary point $\vec{w}^*$, it can classified based as such [Bec17]: if $H \succ 0$ then $\vec{w}^*$ is a strict local minimizer, if $H \succ 0$ then $\vec{w}^*$ is a strict local maximizer, if H is indefinite then $\vec{w}^*$ is a saddle point, if $H \succcurlyeq 0$ or $H \preccurlyeq 0$ then $\vec{w}^*$ can be any of the previous three types, or a local minimizer, or a local maximizer.

1.2 Line Search and Armijo Backtracking

The scientific question to be explored is: Will the fixed step size line search and Armijo backtracking search converge and if they do how close are each of the estimations to the actual local minimizer, how many iterations did it take to reach the output, and how do the algorithms compare to each other and damped Newton's method? The results were evaluated based on whether each algorithm converged, if it did whether it reached a local minimizer or not and the number of iterations it took to get there.

Line search is a process of successive iterations for estimating the local minimizer for a function. A starting estimate and step size are required for the fixed step size algorithm. Given a function $f(\vec{t})$, each iteration of a fixed step size search, the new estimate $\vec{t}_{k+1}$ for iteration k is

$$\vec{t}_{k+1} = \vec{t}_k + s_0 \vec{d}_k \tag{3}$$

where s_0 is the inputted step size and $\vec{d}_k$ is the directional derivative for that iteration and is defined as $\vec{d}_k := -[J_f(\vec{t}_k)]^T$. This term has to be transposed since the Iterations would be repeated until the estimate converges or reaches a maximum number of allowed iterations in practice.

The motivation for backtracking is that choosing an appropriate step size is a major concern for the fixed step size search as too large a step size can cause the estimates to oscillate between values and miss the local minimizer entirely. Conversely, too small a step size makes the algorithm inefficient by increasing the number of iterations required for convergence. This problem can be mitigated by estimating an appropriate step size for each iteration separately, such a process is called backtracking. One method is exponential back-off where, starting at iteration $\gamma = 0$, increment γ for the line search iteration k until $f(\vec{t}_k + \beta^\gamma s_\gamma \vec{d}_k) < f(\vec{t}_k)$ then stop where β is the backtrack rate. An assumption that is made about backtracking is that the user inputted step size s_0 is too large and backtracking estimates a better step size by decreasing the step size by a factor of β until the newly estimated step size results in an improvement (decrease in the objective function value). Exponential back-off can be improved upon using Armijo backtracking [Arm66] by adding an extra condition to the right-hand side of the previously described condition. So similarly, γ is incremented until $f(\vec{t}_k + \beta^\gamma s_\gamma \vec{d}_k) < f(\vec{t}_k) + \vec{\alpha}_k \beta^\gamma s_\gamma \vec{d}_k$ where $\vec{\alpha}_k := J_f(\vec{t}_k)/2$. The added component, known as the Armijo

condition, works by exponentially decaying the magnitude of the slope of the line at the current $f(\vec{t_k})$. One can imagine that as γ increases, this line becomes increasingly flat and eventually approaches that of the exponential back-off.

1.3 Damped Newton's Method

The scientific question to be explored is: Will the fixed step size line search and Armijo backtracking search converge and if they do how close are each of the estimations to the actual local minimizer, how many iterations did it take to reach the output, and how will this method compare to the two line search algorithms? The results were evaluated based on whether each algorithm converged, if it did whether it reached a local minimizer or not, and the number of iterations it took to get there.

Newton's method scales the search direction, which for the line search is the negative transpose of the gradient, using the inverse of the Hessian matrix. This is in fact a local quadratic estimation using the truncated Taylor series up to the second-order term. For functions with vector arguments, this truncated Taylor series around a point $\vec{w}_0$ for $f : \mathbb{R}^n \to \mathbb{R}$ is $T_2(\vec{w}) = f(\vec{w}_0) + \underline{\nabla}(\vec{w}_0)[\vec{w} - \vec{w}_0] + \frac{1}{2}[\vec{w} - \vec{w}_0]^T \underline{\nabla}^2 f(\vec{w}_0)[\vec{w} - \vec{w}_0]$. As per the rules mentioned in Section 1.1, $T_2(\vec{w}_0)$ will have a single stationary point that is a strict minimizer if $\underline{\nabla}^2 f(\vec{w}_0) > 0$. The derivative of the truncated Taylor series with respect to $\vec{w}$ is $\frac{\partial}{\partial \vec{w}} = \underline{\nabla}(\vec{w}_0) + [\vec{w} - \vec{w}_0]^T \underline{\nabla}^2 f(\vec{w}_0)$. This can be used to solve for the desired step direction $[\vec{w} - \vec{w}_0]$ by setting the transpose of the derivative to equal the zero vector:

$$[\underline{\nabla}(\vec{w}_0) + [\vec{w} - \vec{w}_0]^T \underline{\nabla}^2 f(\vec{w}_0)]^T = \vec{0} \tag{4}$$

$$[\underline{\nabla}(\vec{w}_0)]^T + \underline{\nabla}^2 f(\vec{w}_0)[\vec{w} - \vec{w}_0] = \vec{0} \tag{5}$$

$$[\vec{w} - \vec{w}_0] = -[\underline{\nabla}^2 f(\vec{w}_0)]^{-1} [\underline{\nabla}(\vec{w}_0)]^T \tag{6}$$

This result will be denoted as the search direction $\vec{d}$ for Newton's method search. Each iteration of this algorithm can be defined as $\vec{w}_{k+1} = \vec{w}_k + \vec{d}$ where $\vec{d} = -[\underline{\nabla}^2 f(\vec{w}_0)]^{-1} [\underline{\nabla}(\vec{w}_0)]^T$. As with a fixed step size search, Newton's method can overshoot the local minimizer so this method can be combined with Armijo backtracking to dampen the magnitude of each step. Thus each iteration for the damped Newton's method is $\vec{w}_{k+1} = \vec{w}_k + s_k \vec{d}$ where s_k is the step size found by Armijo backtracking for iteration k.

2 Methods

The three objective functions were $f_1(\vec{w}) = ((w_1 + 1.21) - 2(w_2 - 1))^4 + 64(w_1 + 1.21)(w_2 - 1)$, $f_2 = 2w_2^3 - 6w_2^2 + 3w_1^2 w_2$, and $f_3 = 100(w_2 - w_1^2)^2 + (1 - w_1)^2$. The gradient and Hessian matrix of each objective function were using the symbolic math toolbox in MATLAB and then converted to anonymous functions. This was done since doing calculations using symbolic expressions made the iterative algorithms extremely slow. For all functions, the starting estimate w_0 was selected to be $\begin{bmatrix} -1.2 \\ 1.0 \end{bmatrix}$. The step size for the fixed step size line search was selected to be $s = 0.001$. The initial step size for the Armijo backtracking and damped Newton's method search was selected to be $s_0 = 0.1$. The dampening factor β for Armijo backtracking and damped Newton's was selected to be 0.5.

2.1 Finding and Classifying Stationary Points

For each objective function, find the Jacobian of the function which in this case is just a 1×2 gradient 1-form of the partial derivatives, this was done by using the "gradient()" function from the MATLAB symbolic math package and then transposing the output. The built-in function MATLAB function solve "solve()" was used to solve for when the gradient was equal to $\vec{0}$ which will then output the stationary point(s) for each function. For each function, the Hessian matrix was found using the built-in MATLAB function "hessian()" and for each stationary point of each function, the Hessian was evaluated at the stationary point and its eigenvalues were found using the MATLAB function "eig()".

Each of the stationary points was classified based on the eigenvalues of its corresponding Hessian matrix, per the conditions mentioned in Section 1.1. Using conditional statements, if all of the eigenvalues are greater than zero then the stationary point is a strict local minimizer, if all of the eigenvalues are less than zero then the stationary point is a strict local maximizer, if some eigenvalues are greater than zero and some are less than zero then the stationary point is a saddle point, if otherwise then the classification is inconclusive.

2.2 Fixed Step Size Line Search

For each objective function, the algorithm receives four parameters from the user. The starting estimate is initialized as the previously mentioned $\vec{w}_0$ value and the step size s which was chosen to be 0.001. There are two other parameters involved in the convergence criteria. Firstly is that the maximum number of iterations has been limited to 50000 and secondly the Euclidean norm of $\underline{\nabla}(\vec{w}_k)$ must be equal to or less than 10^{-6}. So starting at $k = 0$, the algorithm will run while $k < 50000$ and $||\underline{\nabla} f(\vec{w}_k)|| > 10^{-6}$. To start the algorithm, initialize the current minimizer estimate as $\vec{w}_k = \vec{w}_0$, and initialize the values $f(\vec{w}_k)$ and $\underline{\nabla} f(\vec{w}_k)$. In each iteration, the new estimate $\vec{w}_{k+1}$ is found by carrying out the operation $\vec{w}_{k+1} = \vec{w}_k + s\vec{d}_k$ where the search direction $\vec{d}_k = -[\underline{\nabla} f(\vec{w}_k)]^T$. Afterwards, the current function value is updated to $f(\vec{w}_k) \leftarrow f(\vec{w}_{k+1})$ and the current gradient is updated to $\underline{\nabla} f(\vec{w}_k) \leftarrow \underline{\nabla} f(\vec{w}_{k+1})$. Iterations are repeated until the convergence criteria are met or the maximum number of iterations has been reached. The effectiveness of the estimation on each objective function is evaluated on whether the estimate converges or not and if it does, whether it reaches a local minimizer or not, and the number of iterations.

2.3 Line Search with Armijo Backtracking

Line search with Armijo backtracking has the same parameters mentioned previously with the change to a different s_0 value of 0.1 and the addition of the back-off rate $\beta = 0.5$. The process for a line search with Armijo backtracking is identical except that within iteration before $\vec{w}_{k+1}$ is calculated an additional sub-loop is added to compute a backtracked step. Assuming that the inputted s_0 is too large a step to be used in a fixed step size search, before the this sub-loop executes, the backtracked step is initialized as $s_\gamma \leftarrow s_0$, and the iteration counter starts at $\gamma = 0$. An estimate of the step is also initialized as $f(\vec{w}_k + \beta^\gamma s_0 \vec{d}_k)$. The backtracking loop condition that includes the Armijo condition is $f(\vec{w}_k + \beta^\gamma s\vec{d}_k) \geqslant f(\vec{w}_k) + \vec{\alpha}\beta^\gamma s_\gamma \vec{d}_k$ where $\vec{\alpha} = \frac{1}{2}\underline{\nabla} f(\vec{w}_k)$. If this condition is true $s_{\gamma+1} \leftarrow s_\gamma * \beta$, $f(\vec{w}_k + \beta^\gamma s_\gamma \vec{d}_k) \leftarrow f(\vec{w}_k + \beta^\gamma s_{\gamma+1}\vec{d}_k)$, and $\alpha\beta^\gamma s_\gamma \vec{d}_k \leftarrow \alpha\beta^\gamma s_{\gamma+1}\vec{d}_k$. Each iteration of this sub-loop will reduce s by a factor of β which will affect the values in the next loop. After γ iterations, this s_γ value will be used as the new step size for the update of the current estimate w_{k+1}. The rest of the algorithm carries identically to the fixed step size search. The effectiveness of the estimation on each objective function is evaluated on whether the estimate converges or not and if it does, whether it reaches a local minimizer or not, and the number of iterations.

2.4 Damped Newton's Method

The damped Newton's method search will have the same parameters as the Armijjo backtracking search except the maximum number of iterations allowed was reduced to 200. To start the algorithm, initialize the starting estimate $\vec{w} = \vec{w}_0$,cthe current function value as $f(\vec{w}_k)$, the current gradient as $\underline{\nabla}(\vec{w}_k)$, and the current Hessian matrix $H_f(\vec{w}_k)$. Before the backtracking sub-loop begins set $s = s_0$. For each iteration, perform Armijo backtracking to find the s_γ value. The estimate is updated to $\vec{w}_{k+1} = \vec{w}_k + s_\gamma * -[H_f(\vec{w}_k)]^{-1}[\underline{\nabla}(\vec{w}_k)]^T$. The current estimate, function value, gradient, and Hessian are updated and the iteration counter is incremented to $k+1$. The algorithm is executed until the convergence criteria are met or the maximum number of iterations is reached. The effectiveness of the estimation on each objective function is evaluated on whether the estimate converges or not and if it does, whether it reaches a local minimizer or not, and the number of iterations.

3 Results

Table 1: Numerical values of stationary points, numerical values of Hessian matrices, signs of the eigenvalues, and the kinds of stationary points. Each stationary point is characterized as a local minimizer (min), a local maximizer (max), a saddle point (sdl), or as inconclusive using derivatives (inc). Values are empty or marked "N/A" for entries that do not have a distinct stationary point. Values were rounded to two decimal places and are reported up to two decimal places if necessary.

Stationary Info.	**Function**		
	f_1	f_2	f_3
$\vec{w}_1$	$\begin{bmatrix} -0.21 \\ 0.50 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$\underline{\nabla}^2 f(\vec{w}_1)$	$\begin{bmatrix} 48 & -32 \\ -32 & 192 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & -12 \end{bmatrix}$	$\begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$
At $\vec{w}_1$, λ_1&λ_2:	$\lambda_1 > 0, \lambda_2 > 0$	$\lambda_1 < 0, \lambda_2 = 0$	$\lambda_1 > 0, \lambda_2 > 0$
At $\vec{w}_1$, kind is:	min	inc	min
$\vec{w}_2$	$\begin{bmatrix} -1.21 \\ 1.00 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 2 \end{bmatrix}$	N/A
$\underline{\nabla}^2 f(\vec{w}_2)$	$\begin{bmatrix} 0 & 64 \\ 64 & 0 \end{bmatrix}$	$\begin{bmatrix} 12 & 0 \\ 0 & 12 \end{bmatrix}$	
At $\vec{w}_2$, λ_1&λ_2:	$\lambda_1 < 0, \lambda_2 > 0$	$\lambda_1 > 0, \lambda_2 > 0$	
At $\vec{w}_2$, kind is:	sdl	min	
$\vec{w}_3$	$\begin{bmatrix} -2.21 \\ 1.50 \end{bmatrix}$	N/A	N/A
$\underline{\nabla}^2 f(\vec{w}_3)$	$\begin{bmatrix} 48 & -32 \\ -32 & 192 \end{bmatrix}$		
At $\vec{w}_3$, λ_1&λ_2:	$\lambda_1 > 0, \lambda_2 > 0$		
At $\vec{w}_3$, kind is:	min		

Table 2: Number of iterations and numerical results of estimated minimizers from unconstrained optimization for the three objective functions. Fixed step size search with 0.001 step size. Armijjo backtracking search with an initial step size of 0.1. Damped Newton's method search with an initial step size of 0.1. Values were rounded to two decimal places and are reported up to two decimal places if necessary.

Methods	**f_1**		**f_2**		**f_3**	
	Iters.	$\vec{w}^*$	Iters.	$\vec{w}^*$	Iters.	$\vec{w}^*$
Fixed	482	$\begin{bmatrix}-0.21\\0.50\end{bmatrix}$	1486	$\begin{bmatrix}0.00\\2.00\end{bmatrix}$	32076	$\begin{bmatrix}1.00\\1.00\end{bmatrix}$
Backtracking	37	$\begin{bmatrix}-0.21\\0.50\end{bmatrix}$	15	$\begin{bmatrix}0.00\\2.00\end{bmatrix}$	10704	$\begin{bmatrix}1.00\\1.00\end{bmatrix}$
Damped Newton's	200	$\begin{bmatrix}-1.20\\1.00\end{bmatrix}$	200	$\begin{bmatrix}-1.20\\1.00\end{bmatrix}$	200	$\begin{bmatrix}1.00\\1.00\end{bmatrix}$

4 Discussion

For objective function $f_1(\vec{w})$, three stationary points, which will be denoted as tuples of arguments (w_1, w_2), were found. They were (-0.21, 0.50) which was classified as a strict local minimizer, (-1.21, 1.00) which was classified as a saddle point, and (-2.21, 1.50) which was classified as a strict local minimizer. Two stationary points were found for $f_2(\vec{w})$. They were (0, 0) which had an inconclusive classification and (0, 2) which was classified as a strict local minimizer. For f_3, one stationary point was found. It was (1, 1) which was classified as a strict local minimizer. For the approximation methods, fixed step size search converged for all functions. For fixed step size line search, with a step size of 0.001, arrived at (-0.21, 0.50) for f_1 after 482 iterations, (0, 2) for f_2 after 1486 iterations, and (1, 1) for f_3 after 32076 iterations. Armijo backtracking search converged for all functions. For Armijo backtracking line search, with an initial step size of 0.1, arrived at (-0.21, 0.50) for f_1 after 37 iterations, (0, 2) for f_2 after 15 iterations, and (1, 1) for f_3 after 10704 iterations. The damped Newton's method search, with an initial step size of 0.1, arrived at (-1.20, 1.00) for f_1 after 200 iterations, (-1.20, 1.00) for f_2 after 200 iterations, (1, 1) for f_3 after 200 iterations. Strictly speaking, damped Newton's method did not converge for any of the functions. However, when rounded to two decimal places it does arrive at the correct stationary point for f_3.

Fixed step size line search managed to converge for all functions and all at local minimizes. It was observed that it took the fixed step size search significantly more iterations to reach convergence for f_3 than for f_1 and f_2. It is likely that the region near the starting estimate f_3 contains a portion that resembles a valley with a relatively flat "creek" at the bottom and this results in the fixed step size search requiring far more iterations since the magnitude of the gradient values in this creek are so low, each new estimation step makes very little progress. This is further supported by how Armijo backtracking search took far more iterations for f_3 than for f_1 and f_2. Peeking into the gradient at the estimation of each iteration for f_3 provides some insight into the behaviour of the search path. All following values will be rounded to two decimal places. At the starting estimate, $\underline{\nabla} f_2(\vec{w}_0) = [-215.60 \ -88.00]$. For the following three iterations, $\underline{\nabla} f_2(\vec{w}_1) = [-98.22 \ -42.12]$, $\underline{\nabla} f_2(\vec{w}_2) = [-9.20 \ -2.47]$, $\underline{\nabla} f_2(\vec{w}_3) = [-2.21 \ 0.90]$. From these three iterations, it can be observed that Armijo backtracking search takes the current estimate to a point with much lower gradient values than where the search started. From the rest of the gradient values, it was observed that there is little change from one iteration to the next meaning the search path is travelling through the bottom of a "valley" with low gradient values. This means that regardless of the step size provided by the backtracking process, the magnitude of the gradient values is too small which decreases the distance travelled by each iteration to the extent that it takes significantly more iterations to converge. The same idea applies to the fixed step size search except that method does not get the benefit of being able to take

larger steps when available. The "valley" region of f_3 also provides a reason why Armijo backtracking was more effective for f_1 and f_2 since in the local regions near the starting estimate and search path, f_1 and f_2 lack these regions with very low gradient values and because of that, are able to traverse efficiently since the magnitudes of the search directions are not affected negatively.

Damped Newton's method also managed to reach the actual local minimizer for f_3 but required significantly fewer iterations than the two other methods. The reason for this improved performance is that the direction vector depends on the inverse of the Hessian matrix. The second-order derivatives in a Hessian matrix describe the instantaneous rate of change of the gradient at a certain point. This means that for a point with large values in its Hessian matrix, the inverse of said Hessian would decrease the magnitude of the step direction. The reverse is true, if the Hessian contains smaller values then the inverse would increase the magnitude of the step direction or not decrease it as much. As mentioned previously, f_3 contains a "valley" where the rate of change of the slopes is small. In such a case, damped Newton's method is able to take larger steps than the fixed step size search and backtracking search in regions that are relatively "flat" where the points in said region have similar gradient values since the inverse of the Hessian would contain relatively large values. Although damped Newton's method has a performance advantage for f_3 it did not arrive at a local minimizer for f_1 and f_2, unlike the two other algorithms. The starting estimate used in this assignment happened to be very close to the saddle point of f_1 so intuitively it should not take many iterations for Newton's method to converge on the argument (-1.21, 1.00). It may have taken many iterations despite being so close since the search direction of the starting estimate was small (*dvec* at the starting estimate was $\begin{bmatrix}-0.01\\0.00\end{bmatrix}$) to begin with. So combined with an initial step size of only 0.1, the magnitude of each step was too small to make a noticeable difference, even after 200 iterations. This is supported by the fact that upon changing the initial step size for damped Newton's method to 1, the algorithm converged at point (-1.21, 1.00) after just 2 iterations. This specific implementation shows Newton's method's tendency to converge at stationary points that are not local minimizers since unlike the two other algorithms, whose search directions are defined to head for points with lower function values, the additional multiplication of the inverse of the Hessian removes that guarantee.

Similarly to f_1, f_2 for damped Newton's method is also stuck at the starting estimate. The inverse of the Hessian at the starting estimate rounded to three decimal places for f_2 was $\begin{bmatrix}0.000 & 0.016\\0.016 & 0.000\end{bmatrix}$ which would result in a very small *dvec*. Like with f_1, the combination of a small initial step size and a small *dvec* makes the iteration steps for f_2 so small that the magnitude of each step is negligible and damped Newton's gets stuck at the initial estimate. This highlights another weakness of damped Newton's method. The algorithm can get stuck in regions where there is little change in the gradient even if the magnitude of the gradient values is large since the inverse of the Hessian would scale down the search direction significantly. Increasing the initial step size for f_2 also seems to improve the performance of damped Newton's method. By increasing the initial step size from 0.1 to 3, the algorithm converged at argument (0, 2) after just 15 iterations. Even for f_3, which damped Newton's method has already correctly converged to a local minimizer, raising the initial step size to 1 still arrives at the actually local minimizer while reducing the number of iterations to only 21. It is clear that for these functions, the initial step size has a high impact on the performance of damped Newton's method. An improvement to this implementation can be made by tuning the initial step size as a hyper-parameter. Another option is to adopt adaptive backtracking which allows the initial step size to be increased if necessary and as shown for the implementation of damped Newton's method to these three objective functions, this can have a large impact on performance. A 2019 paper [FR19], proposes two adaptive backtracking methods that "can increase or decrease the step size as much as necessary in each iteration, with no additional gradient evaluations." Such methods, were it implemented here could potentially resolve the issues seen for damped Newton's method when applied to f_1 and f_2 and improve the performance for f_3.

For the given objective functions, it was found that fixed step size line search and Armijo backtracking search converged for all functions. Armijo backtracking had a significantly lower number of iterations until convergence compared to the fixed step size search. However, due to the behaviour of f_3 in the observed

region both of these algorithms performed poorly compared to their f_1 and f_2 results. Damped Newton's method performed poorly for f_1 and f_2 as it was stuck on the initial estimate partly due to the behaviour of the function at that point. It performed well for f_3 due to the incorporation of the inverse of the Hessian in its search direction, which allowed it to navigate the "valley" segment of f_3 efficiently. It was observed that the low initial step size for damped Newton's method was a major drawback for this implementation and increasing the initial step size significantly improved the results of this algorithm for these functions.

References

[Arm66] Larry Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives". In: *Pacific Journal of Mathematics* 16.1 (Jan. 1, 1966), pp. 1–3. DOI: 10.2140/pjm.1966.16.1. URL: https://doi.org/10.2140/pjm.1966.16.1.

[Bec17] Amir Beck. *Introduction To Nonlinear Optimization: Theory, Algorithms, And Applications With Matlab*. Jan. 2017. URL: http://ci.nii.ac.jp/ncid/BB18270797.

[FR19] Sara Fridovich-Keil and Benjamin Recht. *Choosing the Step Size: Intuitive Line Search Algorithms with Efficient Convergence*. 2019. URL: https://opt-ml.org/papers/2019/paper_17.pdf.