Numerical Optimization
Assignment 1

Christian Dybdahl Troesen (tfp233)

February 11, 2021

1 Introduction

In this report we detail the implementation of some case study functions and their first and second derivatives. In Section 2 we cover some theoretical contributions relevant for the implementation. Section 3 details the actual implementation. We start with an overview of the case study functions and their derivatives. This is followed by a implementation-specific discussion of data parallelism, handling overflow as well as testing procedures. We end the section by covering how to plot the case study functions in 3D and include these 3D plots. In Section 4 we draw some general conclusions.

2 Theory

2.1 Contribution 1

We wish to prove that the Hessian of $f(\boldsymbol{x}) = \sum_{i=1}^{N} g_i(\boldsymbol{x}_i), \boldsymbol{x} \in \mathbb{R}^N$ is a diagonal matrix with $(Hf(\boldsymbol{x}))_{ii} = g_i''(\boldsymbol{x}_i)$. First note that $\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}_i} = g_i'(\boldsymbol{x}_i)$. By extension, we must have $\frac{\partial^2 f(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_i} = g_i''(\boldsymbol{x}_i)$. On the other hand, assuming $i \neq j$, then $\frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j} = 0$ as $g_i'(\boldsymbol{x}_i)$ is a constant when differentiating with respect to $\boldsymbol{x}_j$. Given that the Hessian $Hf(\boldsymbol{x})$ is defined such that $(Hf(\boldsymbol{x}))_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}$, it follows that $(Hf(\boldsymbol{x}))_{ij} = 0$ for any $i \neq j$ and $(Hf(\boldsymbol{x}))_{ii} = g_i''(\boldsymbol{x}_i)$. In other words, the Hessian must be a diagonal matrix with $(Hf(\boldsymbol{x}))_{ii} = g_i''(\boldsymbol{x}_i)$.

2.2 Contribution 3

The Rosenbrock function is defined as $f_2(\boldsymbol{x}) = (1 - \boldsymbol{x}_1)^2 + 100 \cdot (\boldsymbol{x}_2 - \boldsymbol{x}_1^2)^2$. We wish to derive its gradient and Hessian as well as prove that the point $(1, 1)$ is a minimizer. Its gradient is defined by its first order partial derivatives. These can be determined using the chain rule:

$$\begin{aligned} \frac{\partial f_2(\boldsymbol{x})}{\partial \boldsymbol{x}_1} &= -2(1 - \boldsymbol{x}_1) + 100 \cdot 2 \cdot \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) \cdot (-2\boldsymbol{x}_1) \\ &= -2(1 - \boldsymbol{x}_1) - 400\boldsymbol{x}_1 \cdot \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) \end{aligned} \tag{1}$$

$$\frac{\partial f_2(\boldsymbol{x})}{\partial \boldsymbol{x}_2} = 100 \cdot 2 \cdot \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) \cdot 1 = 200 \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) \tag{2}$$

Its Hessian is defined by its 4 second-order partial derivatives. We determine these using the first order partial derivatives just computed:

$$\begin{aligned}
\frac{\partial^2 f_2(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_1} &= 2 - 400 \cdot \frac{\partial}{\partial \boldsymbol{x}_1}\left(\boldsymbol{x}_1 \cdot \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right)\right) \\
&= 2 - 400 \cdot \left(1 \cdot \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) + \boldsymbol{x}_1 \cdot (-2\boldsymbol{x}_1)\right) \qquad \text{(product rule)} \\
&= 2 - 400 \cdot \left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2 - 2\boldsymbol{x}_1^2\right) \\
&= 2 - 400\boldsymbol{x}_2 + 1200\boldsymbol{x}_1^2 \\
\frac{\partial^2 f_2(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_2} &= 200 \cdot \frac{\partial}{\partial \boldsymbol{x}_2}\left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) = 200 \\
\frac{\partial^2 f_2(\boldsymbol{x})}{\partial \boldsymbol{x}_1 \boldsymbol{x}_2} &= \frac{\partial^2 f_2(\boldsymbol{x})}{\partial \boldsymbol{x}_2 \boldsymbol{x}_1} \qquad \text{(symmetry of second derivatives)} \\
&= 200 \cdot \frac{\partial}{\partial \boldsymbol{x}_1}\left(\boldsymbol{x}_2 - \boldsymbol{x}_1^2\right) = -400\boldsymbol{x}_1
\end{aligned}$$

Each of the second order partial derivatives above are continuous functions, which means that the Hessian $Hf_2(\boldsymbol{x})$ must be continuous. Hence, if we can prove that $Hf_2(1,1)$ is positive definite and that the gradient $\nabla f(1,1) = \left(\frac{\partial f_2(1,1)}{\partial \boldsymbol{x}_1}, \frac{\partial f_2(1,1)}{\partial \boldsymbol{x}_2}\right)^{\mathrm{T}} = \mathbf{0}$ then $(1,1)$ must be a minimizer of f_2 by Theorem 2.4 [NW06, p. 16]. Proving $\nabla f(1,1) = \mathbf{0}$ can be done by simply substitution into Equation (1) and Equation (2) . We have:

$$\nabla f(1,1) = \begin{pmatrix} -2(1-1) - 400 \cdot 1 \cdot (1 - 1^2) \\ 200\,(1 - 1^2) \end{pmatrix} = \mathbf{0}$$

Now let us prove that $Hf_2(1,1)$ is positive definite. First note that

$$Hf_2(1,1) = \begin{pmatrix} 2 - 400 \cdot 1 + 1200 \cdot 1^2 & -400 \cdot 1 \\ -400 \cdot 1 & 200 \end{pmatrix} = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}$$

Because $Hf_2(1,1)$ is a 2×2 matrix we know that it must be positive definite if $\mathrm{Det}\left(Hf_2(1,1)\right) > 0$ and $\mathrm{trace}\left(Hf_2(1,1)\right) > 0$. The trace is obviously positive as the diagonal of $Hf_2(1,1)$ is positive. The determinant is positive because $\mathrm{Det}\left(Hf_2(1,1)\right) = 802 \cdot 200 - (-400 \cdot -400) = 400$. Hence, $(1,1)$ must be a minimizer for $f_2(\boldsymbol{x})$.

2.3 Contribution 4

The Ellipsoid function is defined as $f_1(\boldsymbol{x}) = \sum_{i=1}^{d} \alpha^{\frac{i-1}{d-1}} \boldsymbol{x}_i^2, \alpha = 1000$. We wish to derive its gradient and Hessian as well as prove that $(0,0)$ is a minimizer when $d = 2$. The function's ith first order partial derivative must be $\frac{\partial f_1(\boldsymbol{x})}{\partial \boldsymbol{x}_i} = 2 \cdot 1000^{\frac{i-1}{d-1}} \boldsymbol{x}_i$. This, by extension, gives us the gradient $\nabla f_1(\boldsymbol{x}) = \left(2 \cdot 1000^{\frac{1-1}{d-1}} \boldsymbol{x}_1, \ldots, 2 \cdot 1000^{\frac{d-1}{d-1}} \boldsymbol{x}_d\right)^{\mathrm{T}} = (2\boldsymbol{x}_1, \ldots, 2000\boldsymbol{x}_d)^{\mathrm{T}}$. Because the Ellipsoid function is a sum of functions of one variable we can use the theoretical contribution from Section 2.1 to determine its Hessian. To be specific, the Hessian $Hf_1(\boldsymbol{x})$ must be a diagonal matrix with $\left(Hf_1(\boldsymbol{x})\right)_{ii} = \frac{\partial^2 f_1(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_i} = \frac{\partial}{\partial \boldsymbol{x}_i}\left(2 \cdot 1000^{\frac{i-1}{d-1}} \boldsymbol{x}_i\right) = 2 \cdot 1000^{\frac{i-1}{d-1}}$. Each of these second order partial derivatives are continuous. As the non-diagonal elements of the Hessian are constant functions equal to 0 it follows that the Hessian itself must be continuous. To prove that $(0,0)$ is a minimizer of $f_1(\boldsymbol{x})$ it therefore suffices to prove that $\nabla f_1(0,0) = \mathbf{0}$ and that $Hf_1(0,0)$ is positive definite. First note that $\nabla f_1(0,0) = (2 \cdot 0, 2000 \cdot 0)^{\mathrm{T}} = \mathbf{0}$. To prove that $Hf_1(0,0)$ is positive definite we use the same approach as in Section 2.2. We note that $\mathrm{trace}\left(Hf_1(0,0)\right) = 2 + 2000 > 0$ and $\mathrm{Det}\left(Hf_1(0,0)\right) = 2 \cdot 2000 > 0$, meaning $Hf_2(0,0)$ is positive definite. Hence, $(0,0)$ must be a minimizer for $f_1(\boldsymbol{x})$.

3 Implementation

3.1 First and second derivatives of case study functions

We have already covered the Rosenbrock and Ellipsoid function and their derivatives in Section 2. The remaining case study functions are the Log-Ellipsoid function and the two attractive sector functions. The log-ellipsoid function is defined as $f_3(\boldsymbol{x}) = \log\left(\epsilon + f_1(\boldsymbol{x})\right), \epsilon = 10^{-16}$. The partial derivatives defining its gradient are

$$\frac{\partial f_3(\boldsymbol{x})}{\partial \boldsymbol{x}_i} = \frac{2}{\epsilon + f_1(\boldsymbol{x})} \cdot 1000^{\frac{i-1}{d-1}} \boldsymbol{x}_i, \text{ for } i = 1 \text{ to } d$$

Its Hessian is, contrary f_1, not a diagonal matrix. This is because it, contrary to f_1, is not a sum of univariate functions. The second order partial derivatives defining respectively its non-diagonal and diagonal elements are

$$\frac{\partial^2 f_3(\boldsymbol{x})}{\partial \boldsymbol{x}_i \boldsymbol{x}_j} = -\frac{4}{\left(\epsilon + f_1(\boldsymbol{x})\right)^2} \cdot 1000^{\frac{i+j-2}{d-1}} \boldsymbol{x}_i \boldsymbol{x}_j, \text{ for } i \neq j$$
$$\frac{\partial^2 f_3(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_i} = \frac{2}{\epsilon + f_1(\boldsymbol{x})} \cdot 1000^{\frac{i-1}{d-1}} - \frac{4}{\left(\epsilon + f_1(\boldsymbol{x})\right)^2} \cdot 1000^{\frac{2i-2}{d-1}} \boldsymbol{x}_i^2$$

The first attractive sector function is defined as $f_4(\boldsymbol{x}) = \sum_{i=1}^{d} h(\boldsymbol{x}_i) + 100 \cdot h(-\boldsymbol{x}_i)$ where $h(x) = \frac{\log(1+\exp(q \cdot x))}{q}, q = 10^8$. The partial derivatives defining its gradient are

$$\frac{\partial f_4(\boldsymbol{x})}{\partial \boldsymbol{x}_i} = \frac{1}{\exp\left(-10^8 \cdot \boldsymbol{x}_i\right) + 1} - \frac{100}{\exp\left(10^8 \cdot \boldsymbol{x}_i\right) + 1}, \text{ for } i = 1 \text{ to } d$$

Like f_1 this function is a sum of univariate functions so its Hessian is a diagonal matrix. The diagonal elements are determined by the unmixed second order partial derivatives:

$$\frac{\partial^2}{\partial^2 \boldsymbol{x}_i} f_4(\boldsymbol{x}) = s(\boldsymbol{x}_i) + 100 s(\boldsymbol{x}_i), \quad s(\boldsymbol{x}_i) = \frac{10^8}{\exp\left(-10^8 \cdot \boldsymbol{x}_i\right) + 1} \cdot \frac{1}{\exp\left(10^8 \cdot \boldsymbol{x}_i\right) + 1}$$

The second attractive sector function is defined as $f_5(\boldsymbol{x}) = \sum_{i=1}^{d} h(\boldsymbol{x}_i)^2 + 100 \cdot h(-\boldsymbol{x}_i)^2$ with $h(x)$ defined as before. The first order partial derivatives defining its gradient are

$$\frac{\partial f_5(\boldsymbol{x})}{\partial \boldsymbol{x}_i} = \frac{2}{10^8} \cdot \frac{\log\left(1 + \exp\left(10^8 \cdot \boldsymbol{x}_i\right)\right)}{\exp\left(-10^8 \cdot \boldsymbol{x}_i\right) + 1} - \frac{200}{10^8} \cdot \frac{\log\left(1 + \exp\left(10^8 \cdot -\boldsymbol{x}_i\right)\right)}{\exp\left(10^8 \cdot \boldsymbol{x}_i\right) + 1}, \text{ for } i = 1 \text{ to } d$$

The Hessian is a diagonal matrix with diagonal defined by the second order partial derivatives

$$\begin{aligned}\frac{\partial^2 f_5(\boldsymbol{x})}{\partial^2 \boldsymbol{x}_i} = 2 &\left(\frac{1}{\exp\left(-10^8 \cdot \boldsymbol{x}_i\right) + 1}^2 + \frac{\log\left(1 + \exp\left(10^8 \cdot \boldsymbol{x}_i\right)\right)}{\exp\left(-10^8 \cdot \boldsymbol{x}_i\right) + 1} \cdot \frac{1}{\exp\left(10^8 \cdot \boldsymbol{x}_i\right) + 1} \right) \\ &+ 200 \left(\frac{1}{\exp\left(10^8 \cdot \boldsymbol{x}_i\right) + 1}^2 + \frac{\log\left(1 + \exp\left(10^8 \cdot -\boldsymbol{x}_i\right)\right)}{\exp\left(-10^8 \cdot \boldsymbol{x}_i\right) + 1} \cdot \frac{1}{\exp\left(10^8 \cdot \boldsymbol{x}_i\right) + 1} \right)\end{aligned}$$

3.2 Data parallelism

All case study functions except f_2 are defined for an arbitrary number of variables. Hence, it is ideal to compute these and their derivatives in a data parallel manner. This can be done using Numpy. As an example, to compute $\nabla f_1(\boldsymbol{x})$ given input vector $\boldsymbol{x}$, one can

construct $\boldsymbol{y} = 1000^{[0,\ldots,d-1]/(d-1)}$ and $\boldsymbol{x}^2$ as vectorized operations and then perform vectorized (elementwise) multiplication of the two. The Hessian $H\left(f_1(\boldsymbol{x})\right)$ can be computed in a data parallel manner as well because it is a diagonal matrix. The vector constituting its diagonal can be computed using an approach similar to the one outlined above. By subsequently applying `numpy.diag` to this vector one finds the desired matrix. The attractive sector functions f_4 and f_5 and their derivatives can also be determined in a data parallel manner, namely because they too are sums of univariate functions. However, this requires some tweaks to deal with overflow (as detailed in Section 3.3). In fact the only quantity that is non-trivial to implement in a data parallel manner is the Hessian $Hf_3(\boldsymbol{x})$, namely because it is not a diagonal matrix. For this reason we choose to implement it in a sequential manner, determining each of its elements in turn using a double-nested for loop (with an appropriate conditional statement to distinguish diagonal and non-diagonal elements).

3.3 Handling overflow

When implementing f_4 and f_5 we have to contend with the fact that $\exp\left(10^8 x\right)$ is too large a number to be represented by most floating point data types, thereby causing overflow. We could use arbitrary precision floating point data types, however this is inefficient and does not work well with data parallelism. If we do want a data parallel implementation then we can make use of the shape of the expressions in which $\exp\left(10^8 x\right)$ occurs. For example, for both f_4 and f_5 we have that $\exp\left(10^8 x\right)$ is enclosed within a logarithm. Hence, if we determine the full expression without computing $\exp\left(10^8 x\right)$ directly we should be able to avoid overflow. We use `np.logaddexp()` for this purpose with arguments $c = \exp(\mathbf{1}) = \mathbf{0}$ and $d = 10^8 \cdot \boldsymbol{x}$. $\exp\left(10^8 x\right)$ also occurs in the derivatives of f_4 and f_5 but this time as part of 'sigmoid' expressions. These expressions can be computed in a data parallel manner without overflow using `scipy.special.expit`

3.4 Testing

Testing our implementation of the case study functions is trivial as we can simply evaluate them on representative test sets and then compare with the corresponding values that the actual function definitions produce. To test our implementation of their first and second derivatives we use finite difference approximations. For gradients we compare each of their constituent first order partial derivatives with the corresponding central limit approximations implemented as $\frac{\partial f}{\partial \boldsymbol{x}_i} \approx \frac{f(\boldsymbol{x}+\epsilon e_i)-f(\boldsymbol{x}-\epsilon e_i)}{2\epsilon}$, where e_i is the vector with entry $i = 1$ and all others zero and ϵ is a very small number, ideally $\mathbf{u}^{1/3}$, assuming $\mathbf{u}$ denotes the unit roundoff [NW06, p.196-197]. For those Hessians that are sparse (i.e. diagonal) we can test their implementation using the finite difference approximation $\frac{\partial^2}{\partial^2 \boldsymbol{x}_i} \approx \frac{\nabla f(\boldsymbol{x}+\epsilon \mathbf{p})_i - \nabla f(\boldsymbol{x})_i}{\epsilon}$, where $f(\boldsymbol{x})_i$ is the ith component of the gradient $\nabla f(\boldsymbol{x})$ and $\mathbf{p}$ is some perturbation vector [NW06, p.205]. Testing the one non-sparse Hessian (i.e $Hf_3(\boldsymbol{x})$) requires a more involved finite difference approximation which we omit for brevity.

3.5 Plotting functions

We can plot the case study function in 3D (for those with arbitrary number of variables we restrict d to 2) using the matplotlib framework in conjunction with `mpl_toolkits.mplot3d`. To create the data needed for plotting we use `np.meshgrid` with appropriate arguments $\mathbf{X}$ and $\mathbf{Y}$. This creates a mesh-grid of data points in 3D. We then apply each given function implementation to this grid to derive a matrix $\mathbf{Z}$ of function values and plot this matrix as

a function of the grid values induced by $\mathbf{X}$ and $\mathbf{Y}$. The resulting plots are shown below.

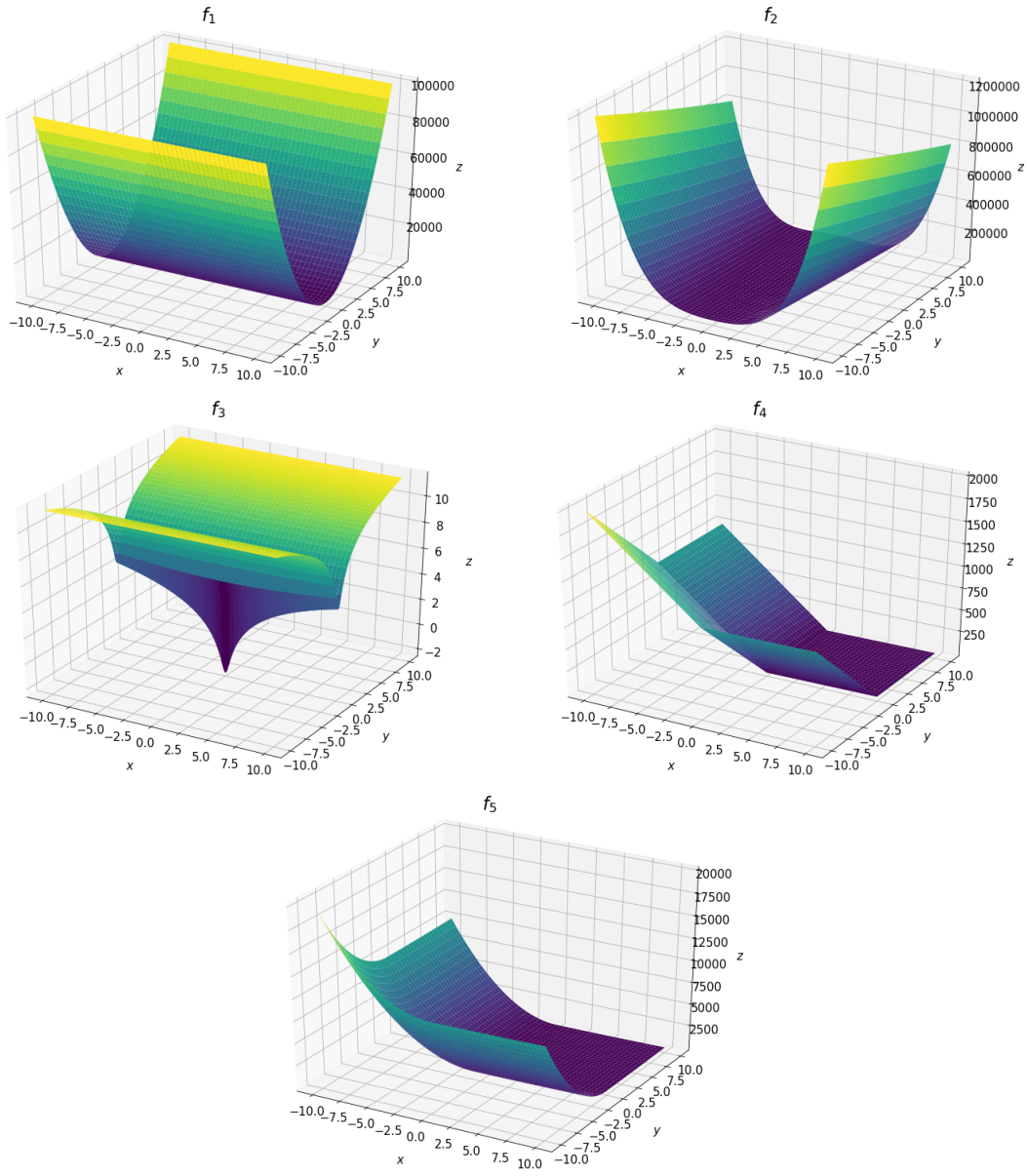

Figure 1: Left to right: Each case study function plotted in 3D (i.e. with 2 input variables). All functions are plotted on meshgrids of 1000×1000 points with coordinates ranging from -10 to 10.

4 Conclusion

Based on the findings of this report we can conclude that the case study function's derivatives are non-trivial to implement in general. This is especially the case for the higher order derivatives (hessians) whose analytic expressions are fairly complicated, particularly for the attractive sector functions and the Log-Ellipsoid function. Moreover, even for simple expressions, problems with overflow may complicate their implementation. In light of this, one might consider using the finite difference approximation method not simply for testing but as a substitute for actual derivative computations. This of course will incur some loss of accuracy, however such a loss may be worth trading for a simpler interface for derivative computations.

References

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006.