Chapter 8

GRADIENT-ONLY SOLUTION STRATEGIES

8.1 Introduction

As outlined in Section 1.1, mathematical optimization is the systematic process of finding a best solution, generally subject to some constraints, to a problem based on a mathematical model. Here the model is constructed in such a way that the solution we seek often corresponds to a quantity that minimizes some multi-dimensional scalar function which is the outcome of the model. Specifically in this chapter we restrict ourselves to unconstrained optimization problems. Thus formally the process now becomes (i) the formulation of the model $f(\mathbf{x})$ and (ii) the minimization of $f(\mathbf{x})$:

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n,$$

where $f(\mathbf{x})$ is a scalar function of the real *column vector* $\mathbf{x}$. Care is usually taken during the mathematical modelling and numerical computation of the scalar function $f(\mathbf{x})$ to ensure that it is smooth and twice continuously differentiable. As highlighted in Section 6.5, the presence of numerical noise in the objective function is sometimes an unintended consequence of the complicated numerical nature frequently associated with the computation of the output function of a multi-disciplinary design optimization model. Numerical noise can also be the conse-

J.A. Snyman and D.N. Wilke, *Practical Mathematical Optimization,*
Springer Optimization and Its Applications 133,
https://doi.org/10.1007/978-3-319-77586-9_8

quence of a deliberate computational savings strategy employed by a design engineer.

To elaborate, consider for example the computational cost associated with integrating a system of partial differential equations that is required to construct the objective function for some design optimization problem. For each chosen design vector $\mathbf{x}$, the numerical integration may require the solution of a finite element model. The associated computational cost to solve the finite element model is directly related to the number of elements used to discretize the spatial domain over which the integration needs to be performed. Not unexpectedly, more elements places higher demands on computing resources at the added benefit of reducing the discretization error which results in more accurate solutions. This clear trade-off between time to solution and solution accuracy may be exploited with care in the design optimization process. A concrete demonstration of this is given by the adaptive remeshing strategy proposed by Wilke et al. (2013a). In this strategy the accuracy of each analysis is increased as the optimizer converges towards an optimum when conducting structural shape optimization. A complication of this strategy is however that because of the initial rough meshing the computed piece-wise smooth objective function is discontinuous. This complication requires a significant adaptation in both (i) the formulation and (ii) the solution strategy to solve the successive approximate discontinuous mathematical optimization problem to the real continuous underlying problem.

This chapter is dedicated to explore alternative formulations and solution strategies when specifically dealing with piece-wise smooth discontinuous objective functions (Wilke et al. (2013b)). In essence, this chapter elaborates and formalizes the concepts and ideas introduced and hinted to in Section 6.5, that includes the handling of noisy objective functions and the use of gradient-only optimization strategies.

8.2 Piece-wise smooth step discontinuous functions

Nowadays, the computation of the objective function routinely requires the integration of a system of differential or partial differential equations. The inherent numerical nature of modern mathematical optimization often allows for the same mathematical model to be solved using different numerical methods (Strang (2007)). Applying care usually renders the same optimum when optimizing the computed objective function with different numerical methods. This observation is valid when the resulting discretization error (Strang (2007)) of the various numerical methods are (i) comparable, (ii) small, and (iii) varies smoothly and continuously between designs.

Failure to meet the first two criteria may still result in numerically computed continuous and smooth objective functions but may render inaccurate approximations of the optimum of the mathematical optimization problem. Consequently, the computed optima for the different methods may differ significantly as a direct result of large or non-comparable discretization errors for the various numerical methods used in computing the objective functions. Lastly, failing to conform to the third criterion results in piece-wise smooth discontinuous objective functions, where the size of the discontinuity decreases as the discretization error reduces. Hence, by ensuring that the discretization error is negligible both (i) smoothness of the objective function and (ii) accuracy in the determination of the optimum to the underlying mathematical model is ensured. This may however place an unattainable computational burden on available computing resources.

To illustrate these concepts consider the Lotka-Volterra system of first order, non-linear, differential equations:

$$\frac{dz(t)}{dt} = (1-\lambda)z(t) - \beta z(t)y(t) \tag{8.1}$$

$$\frac{dy(t)}{dt} = \delta z(t)y(t) - \gamma y(t), \tag{8.2}$$

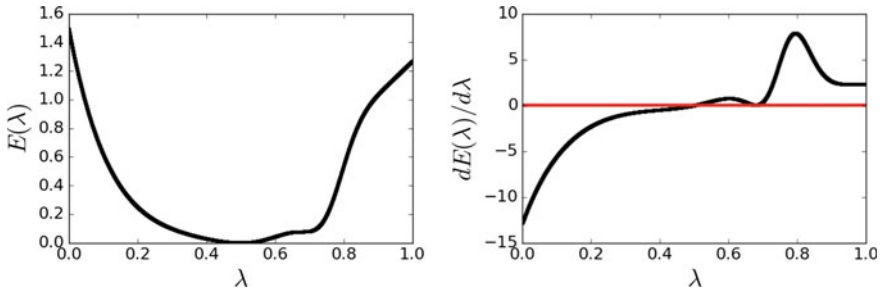
which was first proposed to model auto-catalytic chemical reactions (Yorke and W.N. Anderson (1973)). Two initial conditions $z(0) = z_0$ and $y(0) = y_0$ completes the formulation. Given that $\beta = 0.3$, $\delta = \gamma = 1$

and $z_0 = y_0 = 0.9$, the entire response of the system depends solely on λ. In addition, given that at $t = 8$ the actual values of the two functions $z(8) = \tilde{z}$ and $y(8) = \tilde{y}$ are known, we can construct an inverse problem in which we aim to find λ such that $z(8) \approx \tilde{z}$ and $y(8) \approx \tilde{y}$. The sum of the errors squared w.r.t. λ (Strang (2007)) is a convenient unconstrained optimization problem that defines this inverse problem, with the sum of the errors squared objective function given by

$$E(\lambda) = (z(t = 8, \lambda) - \tilde{z})^2 + (y(t = 8, \lambda) - \tilde{y})^2. \tag{8.3}$$

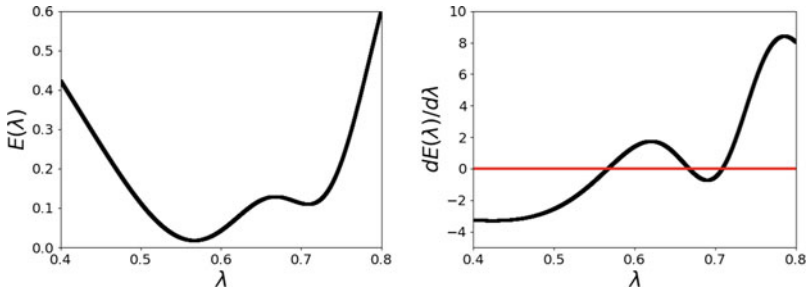
In particular, accept the accurate global optimum to be $\lambda^* = 0.5$ computed using a forward Euler integration scheme (Strang (2007)) for 50 000 equal time steps between 0 and 8 seconds to define $\tilde{z}$ and $\tilde{y}$. Computing the objective function and respective derivative using now only 10 000 equal time steps, values are obtained as depicted in Figures 8.1 (a)–(b) for λ between 0 and 1. Clearly the figures indicate that the accurate optimum λ^* and computed global optimum λ^o in the figures closely coincide with $\lambda^o \approx \lambda^* = 0.5$. There is no apparent difference since the discretization error is negligible whether computing the objective function using 50 000 or 10 000 equally spaced time steps. However when computing the objective function using only 30 equally spaced time steps gives an apparent global optimum $\lambda^o \approx 0.56$ that is significantly different from the accepted global optimum $\lambda^* = 0.5$. This is clear from Figures 8.2 (a)–(b) for which the objective function and its derivative plotted for λ ranging between 0.4 to 0.8. Also of interest is that the computed objective function and its derivative still appears to be continuous and smooth.

Instead of considering only fixed time steps, an integration scheme where the number of time steps varies for different values of λ could be considered. This is reminiscent of adaptive time stepping strategies often employed to solve systems of differential equations (Strang (2007)). As illustration, consider a time stepping strategy that selects a random number of time steps for each λ from a defined distribution around a mean number of time steps, μ. In particular, for $\mu = 30$ consider the number of time steps varying between 27 and 33 with an equal probability. When computing the objective function using between 27 and 33 randomly varying time steps gives an apparent global optimum $\lambda^o \approx 0.56$ that is significantly different from the accepted global optimum $\lambda^* = 0.5$. In addition, the inherent step discontinuous nature of



(a) (b)

Figure 8.1: (a) Error function $E(\lambda)$ and corresponding (b) derivative function $\frac{dE(\lambda)}{d\lambda}$ plotted against values of λ ranging from 0 to 1. In the numerical integration done in computing the function values for each plotted value of λ, 10 000 equally spaced time steps, over the time interval 0 to 8 seconds, were used. The figures show that the apparent global optimum $\lambda^o \approx 0.5$ closely coincides with the accepted accurate global optimum $\lambda^* = 0.5$



(a) (b)

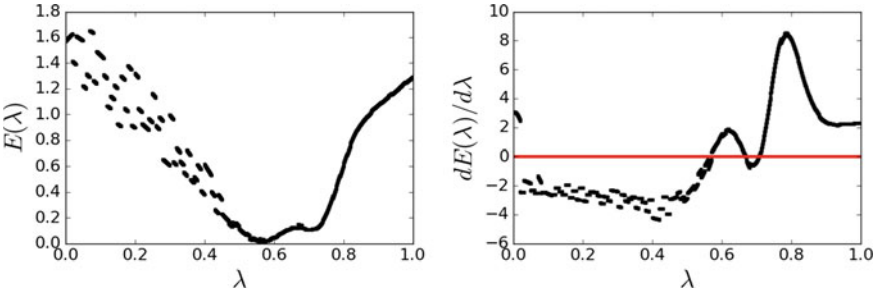
Figure 8.2: (a) Error function $E(\lambda)$ and corresponding (b) derivative function $\frac{dE(\lambda)}{d\lambda}$ plotted against values of λ ranging from 0.4 to 0.8. In the numerical integration done in computing the function values for each plotted value of λ, only 30 equally spaced time steps, over the time interval 0 to 8 seconds, were used. The figures show that the apparent global optimum $\lambda^o \approx 0.56$ significantly differs from the accepted accurate global optimum $\lambda^* = 0.5$

both the objective function and respective derivative function is evident from Figures 8.3 (a) and (b). The additional complications that the step discontinuities introduce to the minimization of $E(\lambda)$, is at first glance disconcerting.

The usual approach to address numerical step discontinuities is to increase the computational cost associated with each analysis with the aim of reducing the magnitudes of the step discontinuities. This is illustrated in Figures 8.4 (a)–(d), in which (a) and (b) indicate the resulting computed objective function and corresponding derivative function when the number of time steps are randomly selected with an equal probability between 54 and 66 for the computation for each λ, while (c) and (d) depicts the same but with the number of time steps now randomly selected with equal probability between 108 and 132. Clearly the magnitudes of these numerical step discontinuities decrease as the number of randomly selected time steps increases. Thus in practice the number of time steps may be increased until the step discontinuity magnitudes are small enough so as to not cause complications when minimization strategies are applied. In addition, the progression of the apparent global optimum λ^o to the accepted global optimum λ^* is also evident as the number of random selected time steps, for the computation of the objective function for each λ, is increased.

Another source of discontinuities in design optimization problems are discontinuities which are as a result of a sudden change in the physical response of a model as the design vector changes, hereafter referred to as physical discontinuities. Examples of physical discontinuities include the onset or breakdown of shock waves between designs in fluid dynamic applications (Homescu and Navon (2003)), or the inherent chaotic nature of dynamic strain ageing during plastic deformation (Sarkar et al. (2007)) that may abruptly change as the design vectors vary. Physical discontinuities are distinct from the numerically induced step discontinuities discussed above. Numerically induced step discontinuities are due to abrupt changes in the discretization error between design vectors. It is important to distinguish between physical and numerical step discontinuities as the complications that arise from their presence may need to be addressed differently.

Physical discontinuities are consequences of the underlying physics being modelled and need to be considered as they relate to actual information



(a) (b)

Figure 8.3: (a) Error function $E(\lambda)$ and corresponding (b) derivative function $\frac{dE(\lambda)}{d\lambda}$ plotted against values of λ ranging from 0 to 1. In the numerical integration over the time interval 0 to 8 seconds done for each plotted λ, the number of equally spaced time steps were randomly selected between 27 and 33. were used. The figures again show that the apparent global optimum $\lambda^o \approx 0.56$ differs significantly from the accepted accurate global optimum $\lambda^* = 0.5$. The piece-wise smooth step discontinuous nature of both the objective and derivative functions is evident

about the nature of the problem under consideration. Hence, physical discontinuities need to be resolved and, indeed, a significant effort has been made to explore their presence and impact in terms of a design optimization problem (Smith and Gilbert (2007)). On the other hand, as demonstrated for the example error function problem, numerical step discontinuities are a consequence of the numerical solution strategy employed to solve an underlying mathematical model, and if significantly present in the computed objective function they may hide the underlying physical nature of the problem under consideration. Thus, ideally we would wish for sufficient computational accuracy so that numerical step discontinuities are effectively eliminated and may be ignored, allowing for the underlying physical trends present in the problem to drive an optimizer towards the solution of an optimization problem. The remainder of this chapter is dedicated to addressing the complications arising from numerical step discontinuities in such a way that piece-wise smooth discontinuous objective functions can still be optimized, efficiently and robustly. The main emphasis will therefore be

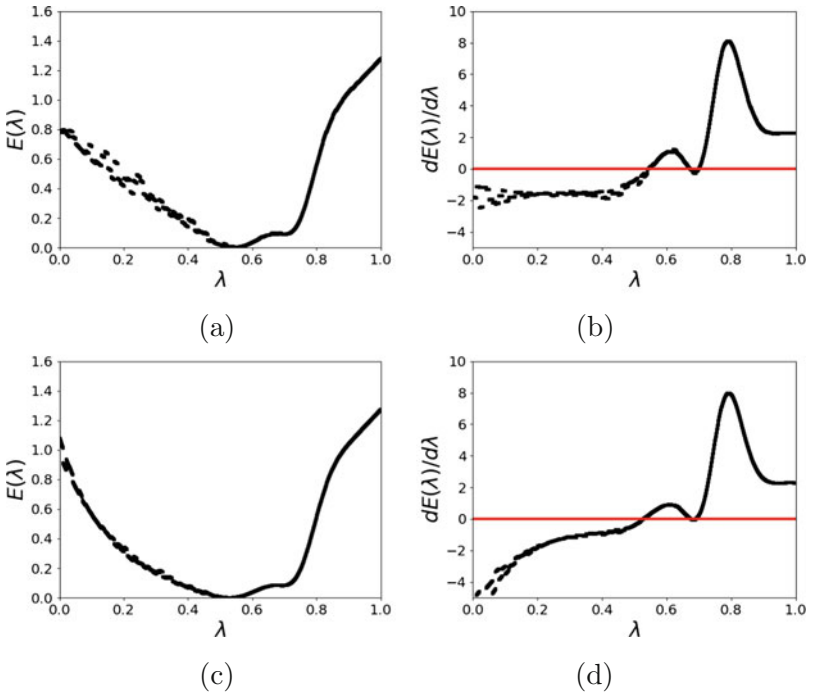


Figure 8.4: (a),(c) Error function $E(\lambda)$ and (b),(d) corresponding derivative function $\frac{dE(\lambda)}{d\lambda}$ plotted against values of λ ranging from 0 to 1 for respectively 54–66 and 108–132 randomly selected time steps computed for each λ. The numerical integration is conducted over the time interval 0 to 8 seconds. The figures show that the magnitude of the step discontinuities decreases as the number of randomly selected time steps increases. In addition, the apparent global optimum λ^o draws closer to the accepted accurate global optimum $\lambda^* = 0.5$ as the number of randomly selected time steps for each λ increases

on computational minimization procedures and strategies that allow for overcoming the presence of step discontinuities in piece-wise smooth step discontinuous objective functions. In addition, a gradient-only problem formulation is included that defines the underlying problem that is consistent with what the optimization procedures and strategies considered in this chapter actually solves.

8.3 Computational minimization procedures

Consideration is now given to different formulations and approaches to compute the unconstrained minimum of the model function $f(\mathbf{x})$, which represents the first step in the overall mathematical optimization process. Although the minimization is generally viewed as the systematic searching of the design space to find at least a local minimum, alternative approaches and solution strategies are at our disposal that may simplify the complexities arising from the presence of numerical step discontinuities in $f(\mathbf{x})$.

To aid the discussion consider Figures 8.5 (a)–(d) that depict the error function $E(\lambda)$ and corresponding derivative function $\frac{dE(\lambda)}{d\lambda}$ plotted against values of λ ranging from 0.5 to 0.6. The error function and corresponding derivative function in Figures 8.5 (a) and (b) respectively are computed using a fixed number of time steps, while they are computed using a randomly selected number of time steps in Figures 8.5 (c) and (d). In particular the numerical integration done, over the time interval 0 to 8 seconds, for each plotted value of λ uses either 30 equally spaced time steps or a randomly selected number of time steps between 27 and 33. Instead of now focusing on the difference between the apparent global minimum $\lambda^o \approx 0.56$ and accepted global minimum $\lambda^* = 0.5$, attention is specifically given to determine the apparent global minimum.

Consider Figure 8.5 (a) that depicts a smooth objective function. Conventional zero- or first order gradient based search minimization methods covered in the previous chapters can be used to accurately solve this optimization problem. An alternative second order approach may be to apply the so-called optimality criteria, i.e. the necessary and sufficient conditions to find all points with zero first derivative and corresponding positive second derivative to determine candidate points for the global minimum. The indicated horizontal line, in Figure 8.5 (b), determines the level-set of a derivative of magnitude zero. As the gradient is increasing with an increase in λ the second derivative is positive. It is clear that in this case (a) the minimization formulation and application of the optimality criteria would recover the same apparent minimum over the indicated λ domain.

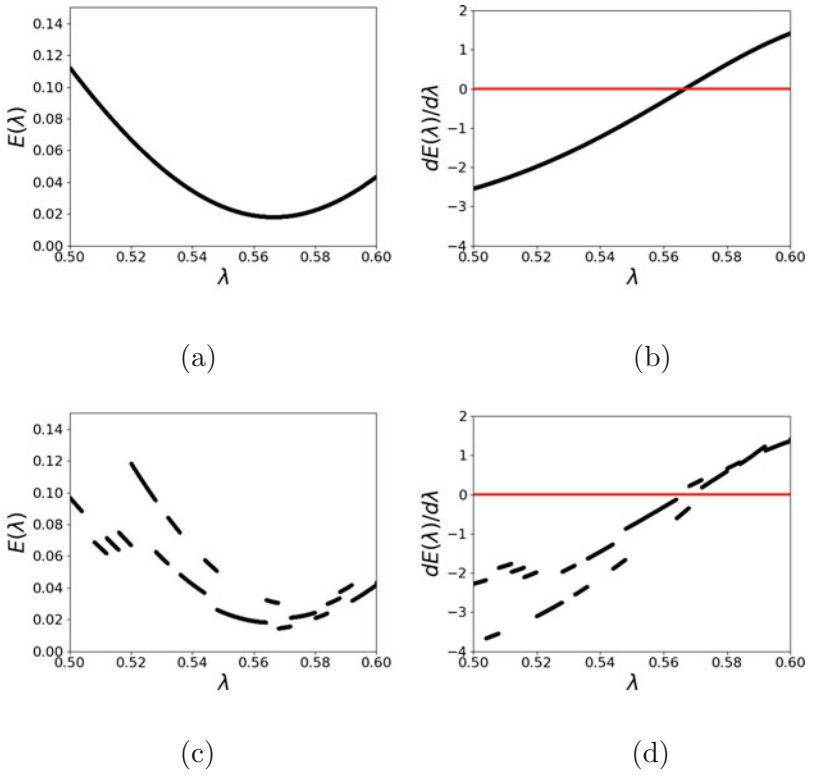


Figure 8.5: (a),(c) Error function $E(\lambda)$ and (b),(d) corresponding derivative function $\frac{dE(\lambda)}{d\lambda}$ plotted against values of λ ranging from 0.5 to 0.6 using respectively a fixed time step (a)–(b) and randomly selected time step (c)–(d) strategy. In particular the numerical integration done for each plotted value of λ uses respectively either 30 equally spaced time steps or 27–33 randomly selected time steps, over the time interval 0 to 8 seconds

The difference between the two approaches is more evident when multiple local minima occur. In this case the application of a zero to first order minimization approach will clearly terminate at a local minimum (not necessary the global minimum), whereas the application of optimality criteria may reveal all local minima including the global minimum, irrespective of their associated function values, as candidate solutions to the problem. Thus the implication for the two solution approaches changes significantly when applied to the piece-wise smooth step discontinuous function and related derivative function, depicted in Figures 8.5 (c) and

(d). It is evident that the application of either minimization strategy now poses additional challenges. The economically computed objective function, as result of the introduction of step discontinuities, has become highly multi-modal with the presence of numerous local minima. In the real world the difficulties in the application of the above two conventional approaches to step discontinuous functions become so daunting that it often leads to the justification and employment of computationally demanding evolutionary approaches (Arnold (2002)), over more efficient gradient based strategies. By inspection of Figure 8.5 (d) it is clear that the second order optimality criteria approach would not suffice as there is no λ for which the derivative is close to 0. It is cautionary noted that the two points with the smallest derivative magnitude is around $\lambda \approx 0.56$ and $\lambda \approx 0.58$, while the apparent global minimum is around $\lambda^o \approx 0.57$. This may have significant implications for solution strategies that aim to find points where the derivative is close to zero.

A third approach is now proposed that consistently interprets the derivative information presented in Figures 8.5 (b) and (d). This third approach, that may be called gradient-only minimization, follows from defining descent for a function along a search direction by directional derivatives that are negative, while ascent is associated with directional derivatives that are positive, with the directional derivative computed as the projection of the gradient vector onto the search direction. Instead of associating a candidate local minimum along a search direction as a point at which the directional derivative is zero, we associate a local minimum with a point where the directional derivative changes in sign from negative to positive. It is important to note that this defines a local minimum and not merely a candidate local minimum, since the second order information is incorporated by requiring the directional derivative to change from negative to positive. By inspection of Figure 8.5 (d) it is clear that by interpreting descent of the function indirectly by requiring the directional derivative to be negative and the minimum by a sign change in the directional derivative, from negative to positive, results in a robust minimization strategy for step discontinuous functions.

8.4 Overview of discontinuities

Before we proceed with an in depth investigation into gradient-only minimization strategies for piece-wise smooth step discontinuous functions (also known as semi-continuous functions), some background on discontinuities in general is required. To assist the discussion consider Figure 8.6 (a)–(d) that depicts four types of discontinuities for univariate functions $f(x)$ at $x = s$. They are respectively an infinite (or asymptotic) discontinuity, a removable discontinuity, an endpoint discontinuity and lastly a step (or jump) discontinuity.

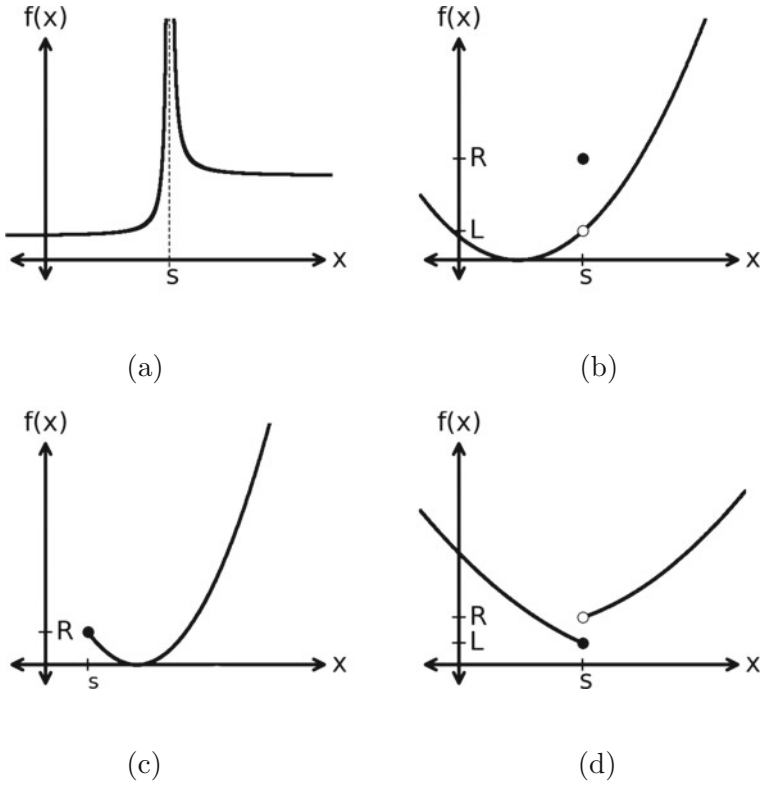


Figure 8.6: Four discontinuity types namely, (a) an infinite (or asymptotic) discontinuity, (b) a removable discontinuity, (c) an endpoint discontinuity and (d) a step (or jump) discontinuity

Figure 8.6 (a) indicates that an infinite discontinuity, at s, is usually the result of vertical asymptotes present at s, with the function

unbounded as x approaches s. Therefore, both the left-hand limit, $\lim_{x\to s^-} f(x) = +\infty$, and right-hand limit, $\lim_{x\to s^+} f(x) = +\infty$, indicate an unbounded response of the function. This type of discontinuity is usually the result of the behaviour of an underlying mathematical model and therefore inherent to the characteristics of a problem to be solved i.e. they can be seen as physical discontinuities. Examples include the Stokes phenomenon present in the solution of differential equations (Meyer (1989)) and the asymptotic Ruderman-Kittel-Kasuya-Yosida (RKKY) interaction between magnetic impurities in graphene (Klier et al. (2014)).

In some instances both one-sided limits exist and are equal, given by $\lim_{x\to s^-} f(x) = \lim_{x\to s^+} f(x) = L$, but the function at $x = s$ does not correspond to the value of the limits, that is $f(s) = R \neq L$, as illustrated in Figure 8.6 (b). This is referred to as a removable discontinuity, since the discontinuity can simply be removed by redefining the function $f(s) = \lim_{x\to s^-} f(x) = L$ at $x = s$.

An end-point discontinuity exists in cases where one of the one-sided limits does not exist, as depicted in Figure 8.6 (c). In this case only the right-hand limit exists, that is $\lim_{x\to s^+} f(x) = R$, while the left-hand limit is not defined. This type of discontinuity is often associated with mathematical models that cannot be evaluated over non-physical domains, e.g. non-positive mass. This type of discontinuity is again associated with physical discontinuities.

Consider the step or jump discontinuity, at $x = s$, in Figure 8.6 (d). Here, both one-sided limits exist and are finite. However, the one-sided limits, at $x = s$, have different values as the left-hand limit, $\lim_{x\to s^-} f(x) = L$, differs from the right-hand limit, $\lim_{x\to s^+} f(x) = R \neq L$. As demonstrated in previous sections, numerical inconsistencies as design vectors vary are a significant source of this type of discontinuity. However, not all step discontinuities are necessarily problematic using classical minimization strategies. We distinguish between two types of step discontinuities. Namely those that are *consistent* with the function *trend*, and those that are *inconsistent* with the function *trend*, as shown respectively in Figures 8.7 (a) and (b).

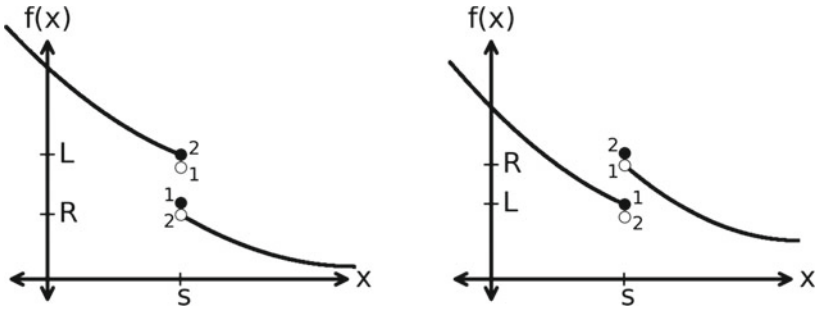
Consistent discontinuities do not hamper descent while inconsistent discontinuities result in a local minimum. However, the related derivative

function in both cases indicates only descent as the sign of the derivative along x remains negative over the depicted domain. This is an important distinction as a gradient-only minimization strategy would ignore a local minimum due to an inconsistent discontinuity, thereby improving the robustness of a minimization strategy, as opposed to a classical minimization strategy that may aim to resolve such a local minimum.

Note that other combinations of function trends and step discontinuities result in either local minima or local maxima that are consistently indicated when considering either only the function or only the corresponding derivative of the function. In addition to the discontinuity type, the semi-continuity of f is indicated using a double empty/filled circle convention in Figure 8.7 (a)–(b). First, a filled circle indicates that $f(s)$ is defined as indicated, while an empty circle indicates no function value is defined as indicated. Lower semi-continuity is represented by the filled/empty circle pairs annotated 1, i.e. the filled circle is always associated with the lower function value, in turn, upper semi-continuity is represented by the empty/filled circle pairs annotated 2, i.e. the filled circle is always associated with the higher function value at the discontinuity. This distinction allows for an explicit treatment of the defined function at a discontinuity, which will become evident as we treat derivatives and gradients associated with semi-continuous functions in more detail.

8.5 Derivatives and gradients of step discontinuous functions

Semi-continuous functions are not everywhere differentiable, as the derivative at step discontinuities is not defined. However, computationally the derivatives and gradients are everywhere computable since the analysis is *per se* restricted to the part of the objective function to the left, or right of a step discontinuity along a search direction. Reference to the derivative of a semi-continuous function therefore requires a rigorous treatment of limits and derivatives to allow for a new definition of what is implied with a derivative at a step discontinuity.



(a) (b)

Figure 8.7: Lower and upper semi-continuous univariate functions for (a) a consistent step discontinuity, and (b) an inconsistent step discontinuity, where the semi-continuity is indicated using a double empty/filled circle convention. The empty/filled circle pairs annotated, 1, are lower semi-continuous, whereas, empty/filled circle pairs annotated, 2, are upper semi-continuous function representations

For a univariate function $f(x)$, the limit at $x = a$ exists if (i) both the left-hand limit, $\lim_{x \to a^-} f(x) = L$, and right-hand limit, $\lim_{x \to a^+} f(x) = \lim_{x \to s^+} f(x) = R$, exist, and (ii) the left-hand and right-hand limits are equal, that is $L = R$. It follows from the limit definition of the derivative

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}, \tag{8.4}$$

that the derivative at $x = a$ is only defined if the limit at $x = a$ is defined. However, as treated in Section 8.4, both the left-hand and right-hand limits exist at a step discontinuity but the limits are not equal, implying that the derivative function is not everywhere defined.

We therefore supplement the definition of a derivative to define the *associated derivative*, $f'^A(x)$, that is given by either the left derivative or right derivative whenever the limit and therefore derivative does not exist, otherwise it is defined by the derivative when the limit does exist. Let $f : X \subset \mathbb{R} \to \mathbb{R}$ be a piece-wise smooth real univariate step-discontinuous function that is everywhere defined. The *associated derivative* $f'^A(x)$ for $f(x)$ at a point x is given by the derivative of $f(x)$ at x when $f(x)$ is differentiable at x. The *associated derivative*, f'^A, for

$f(x)$ non-differentiable at x, is given by the left derivative of $f(x)$:

$$f'^{-}(x) = \frac{df^{-}(x)}{dx} = \lim_{\Delta x^{-} \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}, \tag{8.5}$$

when x is associated with the piece-wise continuous section of the function to the left of the discontinuity, otherwise it is given by the right derivative of $f(x)$:

$$f'^{+}(x) = \frac{df^{+}(x)}{dx} = \lim_{\Delta x^{+} \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}. \tag{8.6}$$

The associated derivative is therefore everywhere defined when piece-wise smooth step discontinuous functions are considered. This implies that $f : (a, b) \subset \mathbb{R} \to \mathbb{R}$ for which $f(x)$ and $f'^{A}(x)$ are uniquely defined for every $x \in (a, b)$ is said to have a strictly negative *associated derivative* on (a, b) if $f'^{A}(x) < 0, \ \forall \ x \in (a, b)$, e.g. see Figure 8.7 (a) and (b). Conversely, $f(x)$ is said to have a strictly *positive associated derivative* on (a, b) if $f'^{A}(x) > 0, \ \forall \ x \in (a, b)$.

Similarly for multi-variate functions we define the *associated gradient* $\nabla_A f(\mathbf{x})$, by letting $f : X \subset \mathbb{R}^n \to \mathbb{R}$ be a piece-wise continuous function that is everywhere defined. The *associated gradient* $\nabla_A f(\mathbf{x})$ for $f(\mathbf{x})$ at a point $\mathbf{x}$ is given by the gradient of $f(\mathbf{x})$ at $\mathbf{x}$ when $f(\mathbf{x})$ is differentiable at $\mathbf{x}$. The *associated gradient* $\nabla_A f(\mathbf{x})$ for $f(\mathbf{x})$ non-differentiable at $\mathbf{x}$ is defined as the vector of partial derivatives with each partial derivative defined by its corresponding *associated derivative*. It follows that the *associated gradient* reduces to the gradient of a function when it is everywhere differentiable.

Similarly, to recognizing the lower and upper semi-continuity of a univariate function as highlighted in Figures 8.7 (a) and (b), we now consider the semi-continuous nature of the associated derivative for such functions. The associated derivative can be related to a univariate function or the directional derivative of a multivariate function. For example the *associated directional derivative* along a normalized direction $\mathbf{u} \in \mathbb{R}^n$ is lower semi-continuous at $\mathbf{x} \in X$, if

$$F'^{A}(\lambda) = {\nabla_A}^{\mathrm{T}} f(\mathbf{x})\mathbf{u} \leq \liminf_{\lambda \to 0^{\pm}} {\nabla_A}^{\mathrm{T}} f(\mathbf{x} + \lambda \mathbf{u})\mathbf{u}, \ \lambda \in \mathbb{R}, \tag{8.7}$$

as depicted in Figure 8.8 (b), with the related function, depicted in Figure 8.8 (a), which is also lower semi-continuous. An upper semi-continuous *associated directional derivative* at $\mathbf{x} \in X$ along a normalized

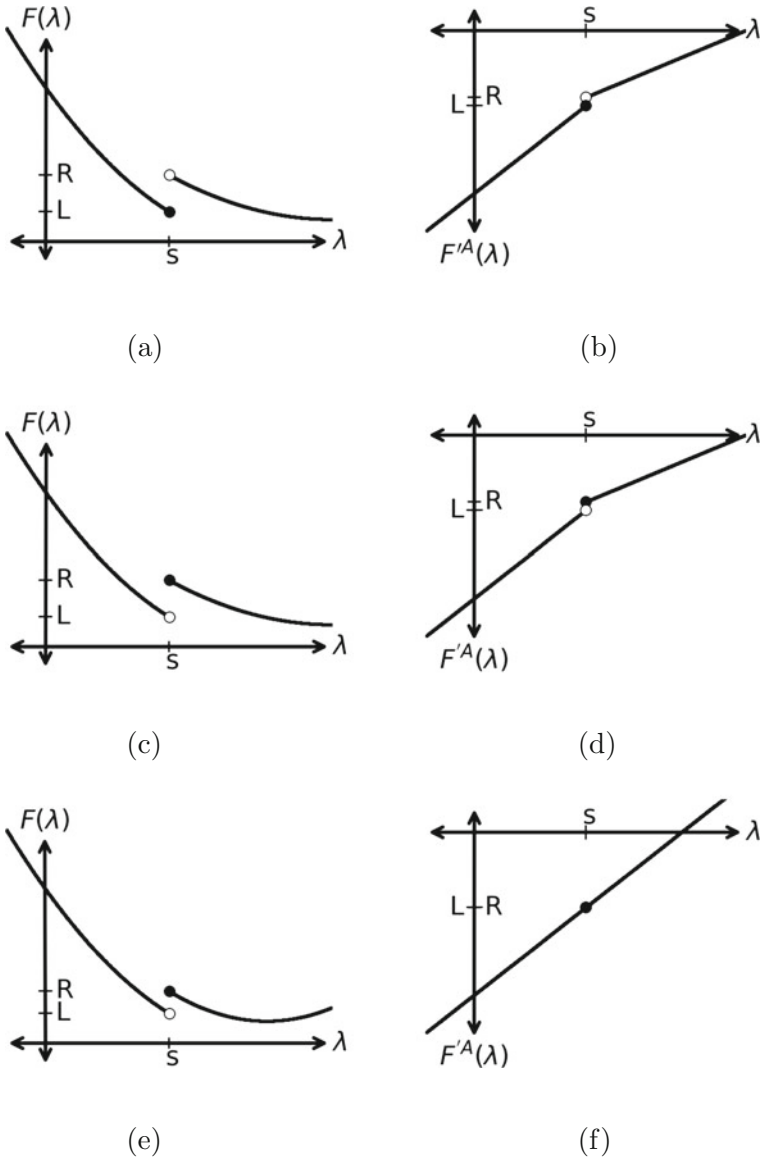


Figure 8.8: (a) Lower semi-continuous function and (b) associated derivative, (c) upper semi-continuous function and (d) associated derivative, and (e) upper semi-continuous function with (f) pseudo-continuous associated derivative

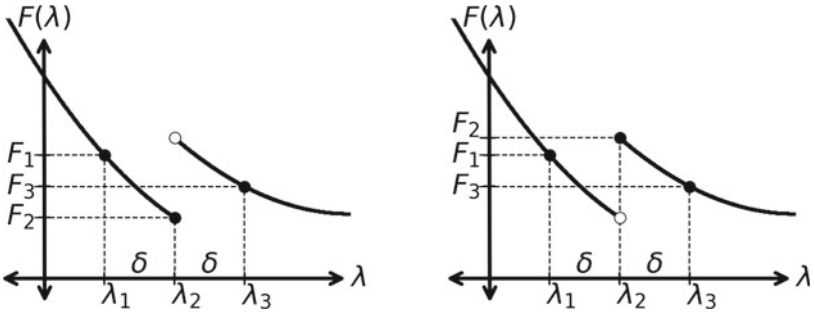
direction $\mathbf{u} \in \mathbb{R}^n$ is defined by

$$F'^A(\lambda) = \nabla_A{}^{\mathrm{T}} f(\mathbf{x})\mathbf{u} \geq \limsup_{\lambda \to 0^{\pm}} \nabla_A{}^{\mathrm{T}} f(\mathbf{x} + \lambda\mathbf{u})\mathbf{u}, \ \lambda \in \mathbb{R}, \tag{8.8}$$

depicted in Figure 8.8 (d) with the related function depicted in Figure 8.8 (c). Lastly, the *associated directional derivative* along a normalized direction $\mathbf{u} \in \mathbb{R}^n$ is pseudo-continuous at a step discontinuity, $\mathbf{x} \in \mathbb{R}^n$, if it is both upper and lower semi-continuous as demonstrated in Figures 8.8 (e) and (f).

8.5.1 Associated gradients by finite differences

The associated gradient can be computed analytically by direct differentiation of the equations that numerically evaluate the objective function (Strang (2007)). Recall that step discontinuities are due to changes in the discretization of the numerical scheme used to evaluate the objective function as the design vector changes, while the computed analytical sensitivity is associated with a given discretization for a specific design.



(a) (b)

Figure 8.9: Finite difference step, δ, over (a) a lower semi-continuous and (b) an upper semi-continuous function with inconsistent step discontinuities

Without loss of generality, we first limit the discussion to a single directional derivative $F'(\lambda)$ as the gradient vector, $\nabla f(\mathbf{x})$, is comprised of directional derivatives aligned with the Cartesian directions. First, consider the finite difference strategies outlined in Section 2.3.1.6, applied

here to an upper or lower step discontinuous function with an inconsistent step discontinuity. For the lower semi-continuous function, $F(\lambda)$, depicted in Figure 8.9 (a) we estimate only the sign of the derivative at λ_2 using the forward (FD), backward (BD) and central difference (CD) schemes, which gives

$$\begin{array}{lcll} \left(\frac{dF(\lambda_2)}{d\lambda}\right)_{FD} & \approx & \frac{F(\lambda_2+\delta)-F(\lambda_2)}{\delta} & = \frac{F_3-F_2}{\lambda_3-\lambda_2} > 0, \\ \left(\frac{dF(\lambda_2)}{d\lambda}\right)_{BD} & \approx & \frac{f(\lambda_2)-f(\lambda_2-\delta)}{\delta} & = \frac{F_2-F_1}{\lambda_2-\lambda_1} < 0, \\ \left(\frac{dF(\lambda_2)}{d\lambda}\right)_{CD} & \approx & \frac{F(\lambda_2+\delta)-F(\lambda_2-\delta)}{2\delta} & = \frac{F_3-F_1}{\lambda_3-\lambda_1} < 0, \end{array} \tag{8.9}$$

whereas for the upper semi-continuous function in Figure 8.9 (b) the sign of the derivative at λ_2 is estimated as follows:

$$\begin{array}{lcll} \left(\frac{dF(\lambda_2)}{d\lambda}\right)_{FD} & \approx & \frac{F(\lambda_2+\delta)-F(\lambda_2)}{\delta} & = \frac{F_3-F_2}{\lambda_3-\lambda_2} < 0, \\ \left(\frac{dF(\lambda_2)}{d\lambda}\right)_{BD} & \approx & \frac{F(\lambda_2)-F(\lambda_2-\delta)}{\delta} & = \frac{F_2-F_1}{\lambda_2-\lambda_1} > 0, \\ \left(\frac{dF(\lambda_2)}{d\lambda}\right)_{CD} & \approx & \frac{F(\lambda_2+\delta)-F(\lambda_2-\delta)}{2\delta} & = \frac{F_3-F_1}{\lambda_3-\lambda_1} < 0. \end{array} \tag{8.10}$$

It is evident that finite differences over inconsistent step discontinuities are problematic resulting in inconsistencies not only in the magnitude of the derivative but also the signs of the computed derivatives. Here, the problem is that an actual finite difference step, δ, is taken over the step discontinuity. This then results in inconsistent estimates of the derivative. Although not always practical or possible, the step discontinuity can be removed by forcing the numerical computation scheme to only have smooth variations in the discretization error whilst computing the derivatives.

The complex-step method circumvents the above issues related to conventional finite difference strategies by only taking finite difference steps of δi in the imaginary plane. The implication is that no step over a discontinuity is ever taken to compute the derivative even at a discontinuity (Wilke and Kok (2014)). As discussed in Section 2.3.1.6, the complex-step method has the additional advantages of allowing for much smaller finite difference steps to be taken as it is not susceptible to a subtraction error.

To demonstrate our arguments, consider the following simple piece-wise linear step discontinuous function:

$$f(x) = \begin{cases} x < 1: & -2x - 0.5 \\ x \geq 1: & -2x \end{cases} . \tag{8.11}$$

with analytical associated derivative of $f'(x) = -2$. The choice for a piece-wise linear function implies the Taylor series approximation is exact for all schemes. Hence, the truncation error is exactly zero for all finite difference schemes on each section of the piece-wise linear function. Any error that varies as a function of the step size is due to the numerical errors introduced by the subtraction of two numbers or the influence of the discontinuity errors.

Mathematically, the derivative is not defined at $x = 1$. However, the *associated* derivative of this function is continuous and -2 everywhere, including at $x = 1$. Computing the derivative with the complex-step method yields exactly -2 everywhere, including $x = 1$, allowing a full field computation of the derivative of a discontinuous function. The computed sensitivity difference for the forward difference scheme varies between $\approx 10^{-16}$ and $\approx 10^{0}$ as the step size is decreased from 10^{0} to 10^{-20}. In turn, the computed sensitivity difference for both the backward and central difference schemes vary between $\approx 10^{-1}$ and $\approx 10^{15}$ as the step size is decreased from 10^{0} to 10^{-20}. Hence, the difference in magnitude increases as the step size decreases, in addition to the derivative having the wrong sign. Therefore extending the numerical computation of the components of the associated gradient vector $\nabla_A f(\mathbf{x})$, namely $\frac{\partial_A f(\mathbf{x})}{\partial_A x_j}$, $j = 1, \ldots, n$, follows by complex-step differences:

$$\frac{\partial_A f(\mathbf{x})}{\partial_A x_j} \cong \frac{Im[f(\mathbf{x} + i\boldsymbol{\delta}_j)]}{\delta_j}, \tag{8.12}$$

where $\boldsymbol{\delta}_j = [0, 0, \ldots \delta_j, 0, \ldots, 0]^T$, $\delta_j > 0$ in the j-th position.

8.6 Gradient-only line search descent methods

Gradient-only line search descent methods closely follow the structure of line search descent methods treated in Chapter 2. Following the general six-step structure, outlined in Chapter 2, for line search descent methods, we outline gradient-only line search descent methods as follows:

Algorithm 8.1 Gradient-only line search descent framework.

Initialization: Select tolerance $\varepsilon_1 > 0$. Select the maximum number of iterations i_{max} and perform the following steps:

1. Given $\mathbf{x}^0$, set $i := 1$.
2. Select a descent direction $\mathbf{u}^i$ (see descent condition (2.1)).
3. Perform a *one-dimensional gradient-only line search* in direction $\mathbf{u}^i$: i.e. find λ_i that designates a sign change from negative to positive in the directional derivative:

$$F'(\lambda) = \nabla^{\mathrm{T}} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i)\mathbf{u}^i,$$

   to indirectly compute the minimizer, λ_i.
4. Set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$.
5. **Convergence test:** *if* $\|\mathbf{x}^i - \mathbf{x}^{i-1}\| < \varepsilon_1$ or $i > i_{max}$, then stop and $\mathbf{x}^* \cong \mathbf{x}^i$, *else* go to Step 6.
6. Set $i = i + 1$ and go to Step 2.

We turn our attention to the third step, which requires the minimum along the search direction to be indirectly resolved by finding a sign change in the derivative from negative to positive, as opposed to, by direct minimization of the function.

8.6.1 One-dimensional gradient-only line search

Clearly, in implementing the gradient-only descent algorithms as outlined above, requires the univariate problem along direction $\mathbf{u}^i$ to be solved:

Find λ_i that designates a sign change from negative to positive in the directional derivative:

$$F'(\lambda) = \nabla^{\mathrm{T}} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i), \mathbf{u}^i,$$

to indirectly compute the minimizer, λ_i.

The problem of finding a sign change in the directional derivative, $F'(\lambda)$, is closely related to a one-dimensional root finding problem, which is usually conducted in two phases (Wilke et al. (2013b)). First by bracketing a sign change and then secondly by reducing the bracket size to refine the location of the sign change.

8.6.1.1 Gradient-only exact line searches

The bracketing phase only requires two points, as two points uniquely define a sign change from a negative directional derivative to a positive directional derivative. Given some user specified parameter, h, this bracket can be achieved by evaluating the directional derivative at

$$\lambda_i = ih + h, \quad i = 0, 1, 2, \ldots,$$

until a sign change from negative to positive is located. Alternatively, instead of using fixed interval sizes between consecutive points, the interval sizes between consecutive points can be increased:

$$\lambda_i = \sum_{k=0}^{i} ha^k, \quad i = 0, 1, 2, \ldots,$$

where h is an initial interval step size and a the interval growth parameter. Choosing $a \approx 1.618$ recovers the bracketing strategy often used for the popular golden section method used in line search descent (Arora (2004)).

Once an interval has been bracketed, with lower bound λ_L^0 and upper bound λ_U^0, that isolates a sign change in the directional derivative from negative to positive, that is $F'^A(\lambda_L^0) < 0$ and $F'^A(\lambda_U^0) > 0$, the interval is reduced to isolate the sign change within a specified tolerance ϵ.

The interval reduction can be done using a standard bisection approach by evaluating the directional derivative in the middle of successively bracketed intervals. Thus starting with $k = 1$ set middle value for $\lambda_M^{k-1} = \frac{\lambda_U^{k-1} + \lambda_L^{k-1}}{2}$ to give $F'^A(\lambda_M^{k-1})$.

If $F'^A(\lambda_M^{k-1}) < 0$ then

1. set $\lambda_L^k = \lambda_M^{k-1}$, and

2. set $\lambda_U^k = \lambda_U^{k-1}$,

while if, $F'^A(\lambda_M^{k-1}) > 0$, then

1. set $\lambda_L^k = \lambda_L^{k-1}$, and
2. set $\lambda_U^k = \lambda_M^{k-1}$.

This process is repeated, for $k = 1, 2, \ldots$, until $\frac{\lambda_U^k - \lambda_L^k}{2} < \epsilon$.

Note that the gradient-only bi-section interval is reduced by 50% every iteration, while the most efficient line search descent interval strategy, namely the golden section method, only reduces the interval by 38.2% at every iteration. Since the bracketed interval is efficiently reduced using interval bi-section, it is preferable to opt for a bracketing strategy for which the interval between successive points increases.

8.6.2 Conditions for sufficient improvement

Similar to the conditions for sufficient improvement discussed in Section 2.3.1.5, there are gradient-only conditions that can be utilized to indicate sufficient improvement and that can be used as a termination criteria for a line search strategy. Consider the following conditions that may be imposed on the step $\lambda_i \mathbf{u}^i$ in the direction $\mathbf{u}^i$ during the line search:

1. Predefined:
$$\lambda_i = d_i, \text{where } d_i \text{ is prescribed at step } i$$
2. Descent:
$${\mathbf{u}^i}^{\mathrm{T}} \nabla f(\mathbf{x}^i + \lambda_i \mathbf{u}^i) \leq 0,$$
3. Curvature:
$$c_1 {\mathbf{u}^i}^{\mathrm{T}} \nabla f(\mathbf{x}^i) \leq {\mathbf{u}^i}^{\mathrm{T}} \nabla f(\mathbf{x}^i + \lambda_i \mathbf{u}^i),$$
4. Strong curvature:
$$|{\mathbf{u}^i}^{\mathrm{T}} \nabla f(\mathbf{x}^i + \lambda_i \mathbf{u}^i)| \leq c_2 |{\mathbf{u}^i}^{\mathrm{T}} \nabla f(\mathbf{x}^i)|,$$

5. Upper curvature:

$$\mathbf{u}^{i^{\mathrm{T}}} \nabla f(\mathbf{x}^i + \lambda_i \mathbf{u}^i) \leq c_3 |\mathbf{u}^{i^{\mathrm{T}}} \nabla f(\mathbf{x}^i)|,$$

with c_1, c_2 and c_3 required to be selected as non-negative parameters. These parameters control the degree to which the conditions are enforced. The simplest condition is a predefined strategy in which the step length evolution d_i is chosen *a priori* before the start of the optimization run and only depends on the iteration number i as detailed by Bertsekas (2015). A constant step length is popular amongst subgradient methods originally introduced by Shor et al. (1985). The other three strategies aims to assimilate information about the problem to inform step lengths. That is the step length depends on the gradient at the current point and the current search direction.

The descent condition ensures that the search direction remains a descent direction at the update. The disadvantage of such a condition is that the sign change is only approached from the left side. The curvature condition attempts to rectify this but may result in updates that are too large as any positive directional derivative satisfies this condition. The strong curvature condition in turn limits the largest update step size but do require the magnitude of the directional derivative to diminish as c_2 is reduced. This is sufficient for problems that are smooth in the vicinity of the optimum, whereas, it may be problematic at discontinuous solutions, i.e. it may be possible that no point along a search direction satisfies this condition. The upper curvature condition ensures that an update always exists, however, small step sizes also satisfy this condition. This can be circumvented by combining this condition with another condition that limits the minimum step size, e.g. using an *a priori* step length strategy.

8.7 Gradient-only sequential approximate optimization

In sequential approximate optimization (SAO) methods, the approximation functions used can easily be formulated using truncated second order Taylor expansions following Snyman and Hay (2001) and Groenwold et al. (2007). For the purposes of gradient-only optimization we

aim here to approximate the gradient of a function $\nabla f(\mathbf{x})$ around some current iterate $\mathbf{x}^i$ to be given by

$$\nabla \tilde{f}^i(\mathbf{x}) = \nabla f(\mathbf{x}^i) + \mathbf{H}_i(\mathbf{x} - \mathbf{x}^i), \tag{8.13}$$

where, according to Wilke et al. (2010), some approximation of the curvature $\mathbf{H}_i$ using only gradient information at iteration i is required. Approximations for $\mathbf{H}_i$ are usually obtained by requiring the gradient to be recovered at the previous iteration, $i-1$, where the gradient had been computed. Given the well-known secant equation,

$$\begin{aligned} \nabla f^i(\mathbf{x}^{i-1}) = \nabla f(\mathbf{x}^i) + \mathbf{H}_i(\mathbf{x}^{i-1} - \mathbf{x}^i), \\ \mathbf{H}_i(\mathbf{x}^{i-1} - \mathbf{x}^i) = \nabla f^i(\mathbf{x}^{i-1}) - \nabla f(\mathbf{x}^i), \\ \mathbf{H}_i \Delta \mathbf{x}^{i-1} = \Delta \nabla f^{i-1}, \end{aligned} \tag{8.14}$$

where $\mathbf{H}_i$ in general requires n^2 components to be solved or $\frac{n^2-n}{2} + n$ components for a symmetric $\mathbf{H}_i$, where symmetry is guaranteed for twice continuously differentiable functions. Hence, to uniquely solve for $\mathbf{H}_i$ requires n^2 linear equations, but each gradient vector only contributes n equations towards the system of linear equations. Generalizing the secant equation results in the following system of equations from which to solve for $\mathbf{H}_i$,

$$\mathbf{H}_i \left[\Delta \mathbf{x}^{i-1}, \ldots, \Delta \mathbf{x}^{i-k} \right] = \left[\Delta \nabla f^{i-1}, \ldots, \Delta \nabla f^{i-k} \right]. \tag{8.15}$$

Consequently, by requiring $k = n$ unique gradient vectors to be recovered at the n previous iterates results in a linear system of equations that can be uniquely solved to yield $\mathbf{H}_i$. However, choosing $k < n$ results in an underdetermined system of equations to be solved, which can be regularized by requiring a minimum norm solution to $\mathbf{H}_i$. Alternatively, instead of solving for the full $\mathbf{H}_i$, a form for $\mathbf{H}_i$ can be assumed that requires less components to be solved for. Here, different assumptions regarding the form of $\mathbf{H}_i$ results in different assumptions on the curvature of the problem, and ultimately different approximation strategies. Typical forms include a constant diagonal Hessian matrix that implies constant curvature (also known as spherical approximations), general

diagonal Hessian matrix implies that changes in curvature are aligned with the Cartesian coordinate axes (subset of separable problems), full non-symmetric Hessian matrix and a full symmetric Hessian matrix that assumes second order continuity.

Once $\mathbf{H}_i$ is approximated the current subproblem i is constructed and solved analytically since the subproblem is continuous by construction; the minimizer of subproblem i follows from setting the gradient of (8.13) equal to $\mathbf{0}$ to give the update

$$\mathbf{x}^{i*} = \mathbf{x}^i - (\mathbf{H}_i)^{-1}\nabla f(\mathbf{x}^i), \tag{8.16}$$

which can be solved from $\mathbf{H}_i(\mathbf{x}^{i*} - \mathbf{x}^i) = -\nabla f(\mathbf{x}^i)$. Solving a linear system may be computationally demanding when large systems are to be considered. Extending on the discussion in Section 2.3.2, this computational burden can be avoided when the inverse Hessian $\mathbf{G}_i = (\mathbf{H}_i)^{-1}$ is directly approximated. This then merely requires a matrix vector product

$$\mathbf{x}^{i*} = \mathbf{x}^i - \mathbf{G}_i\nabla f(\mathbf{x}^i) \tag{8.17}$$

to compute the update.

A general framework of gradient-only sequential approximation algorithms is listed in Algorithm 8.2.

8.7.1 Constant diagonal Hessian matrix approximations

Assuming the curvature can be described by a constant diagonal Hessian matrix results in a spherical approximation of the Hessian, which is approximated by a single scalar. As highlighted by Gould et al. (2005) this allows for a sparse description well suited for high-dimensional optimization problems. Hence, the approximate Hessian or curvature is of the form $\mathbf{H}_i = c_i\mathbf{I}$, with c_i a scalar, and $\mathbf{I}$ the identity matrix. This gives

$$\nabla\tilde{f}^i(\mathbf{x}) = \nabla f(\mathbf{x}^i) + c_i(\mathbf{x} - \mathbf{x}^i), \tag{8.18}$$

with the scalar curvature c_i unknown.

At $\mathbf{x} = \mathbf{x}^i$, the gradient of the function ∇f and the gradient of the approximation function $\nabla\tilde{f}$ match exactly. The approximate Hessian

Algorithm 8.2 Gradient-only sequential approximation algorithm.

Initialization: Given $\mathbf{x}^0$, select the real constant $\epsilon > 0$ and initial curvature $c_0 > 0$. Select the maximum number of iterations i_{max}. Set $\mathbf{H}_0 = \mathbf{I}$ or $\mathbf{G}_0 = \mathbf{I}$. Set $i := 0$ and perform the following steps:

1. **Gradient evaluation:** Compute $\nabla f(\mathbf{x}^i)$.
2. **Approximate optimization:** Construct a local approximate subproblem (8.13) at $\mathbf{x}^i$ using an appropriate approximation to $\mathbf{H}_i$ (or $\mathbf{G}_i$) obtained from only gradient information. Solve this subproblem analytically via (8.16) (or (8.17)) to arrive at a new candidate solution $\mathbf{x}^{i*}$.
3. **Move to the new iterate:** Set $\mathbf{x}^{i+1} := \mathbf{x}^{i*}$.
4. **Convergence test:** if $\|\mathbf{x}^{i+1} - \mathbf{x}^i\| \leq \epsilon$, OR $i = i_{max}$, stop.
5. **Initiate an additional outer loop:** Set $i := i + 1$ and go to Step 1.

$\mathbf{H}_i$ of the approximation $\tilde{f}$ is chosen to match additional information. c_i is obtained by matching the gradient vector at $\mathbf{x}^{i-1}$. Since only a single free parameter c_i is available, the n components of the respective gradient vectors are matched in a least square sense. The least squares error is given by

$$E^i = (\nabla \tilde{f}^i(\mathbf{x}^{i-1}) - \nabla f(\mathbf{x}^{i-1}))^{\mathrm{T}} (\nabla \tilde{f}^i(\mathbf{x}^{i-1}) - \nabla f(\mathbf{x}^{i-1})), \tag{8.19}$$

which, after substitution of (8.18) into (8.19), gives

$$\begin{aligned} E^i = (\nabla f(\mathbf{x}^i) + c_i(\mathbf{x}^{i-1} - \mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))^{\mathrm{T}} \\ (\nabla f(\mathbf{x}^i) + c_i(\mathbf{x}^{i-1} - \mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1})). \end{aligned} \tag{8.20}$$

Minimization of the least squares error E^i w.r.t. c_i then gives

$$\begin{aligned} \frac{dE^i}{dc_i} = (\nabla f(\mathbf{x}^i) + c_i(\mathbf{x}^{i-1} - \mathbf{x}^i) \\ - \nabla f(\mathbf{x}^{i-1}))^{\mathrm{T}}(\mathbf{x}^{i-1} - \mathbf{x}^i) \\ + (\mathbf{x}^{i-1} - \mathbf{x}^i)^{\mathrm{T}}(\nabla f(\mathbf{x}^i) \\ + c_i(\mathbf{x}^{i-1} - \mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1})) = 0, \end{aligned} \tag{8.21}$$

hence

$$c_i = \frac{(\mathbf{x}^{i-1} - \mathbf{x}^i)^{\mathrm{T}}(\nabla f(\mathbf{x}^{i-1}) - \nabla f(\mathbf{x}^i))}{(\mathbf{x}^{i-1} - \mathbf{x}^i)^{\mathrm{T}}(\mathbf{x}^{i-1} - \mathbf{x}^i)}. \tag{8.22}$$

The approximation (8.18) can be enforced to be strictly convex by enforcing $c_i = \max(\beta, c_i)$, with $\beta > 0$ small and prescribed.

8.7.2 Diagonal Hessian matrix approximations

A more general separable approximation is obtained by allowing the Hessian matrix to develop into a diagonal matrix allowing for n coefficients to be solved for. Hence, the approximate Hessian or curvature is of the form $\mathbf{D}_i$, with $\mathbf{D}$ signifying a diagonal matrix. This gives

$$\nabla \tilde{f}^i(\mathbf{x}) = \nabla f(\mathbf{x}^i) + \mathbf{D}_i(\mathbf{x} - \mathbf{x}^i), \tag{8.23}$$

with $D_{i_{jk}}$, $j = k$ unknown for all $j = 1, \ldots, n$ and $k = 1, \ldots, n$, while $D_{i_{jk}} = 0$ for $j \neq k$.

At $\mathbf{x} = \mathbf{x}^i$, the gradients of the function f and the gradient of the approximation function $\nabla \tilde{f}$ match exactly. Again, the approximate Hessian $\mathbf{D}_i$ of the approximation $\tilde{f}$ is chosen to match additional information. $\mathbf{D}_i$ is obtained by matching the gradient vector at $\mathbf{x}^{i-1}$. Since $\mathbf{D}_i$ has n unknowns that are separable, the n components of the gradient vector are matched exactly,

$$\nabla f(\mathbf{x}^{i-1}) = \nabla f(\mathbf{x}^i) + \mathbf{D}_i(\mathbf{x}^{i-1} - \mathbf{x}^i), \tag{8.24}$$

with each component solved for independently

$$D_{i_{jj}} = \frac{\nabla f_j(\mathbf{x}^{i-1}) - \nabla f_j(\mathbf{x}^i)}{(x_j^{i-1} - x_j^i)}, \; j = 1, \ldots, n. \tag{8.25}$$

8.7.3 Symmetric Hessian matrix approximations

Instead of approximating the entire $\mathbf{H}_i$ at every iteration, every iteration can add information to a previous approximation $\mathbf{H}_{i-1}$. Following conventional Quasi-Newton derivations consider the following Hessian update scheme

$$\mathbf{H}_i = \mathbf{H}_{i-1} + \Delta\mathbf{H}_{i-1}. \tag{8.26}$$

The rank of the incremental update $\Delta\mathbf{H}_{i-1}$ depends on the number of difference gradient vectors enforced per iteration. Assuming, $\Delta\mathbf{H}_{i-1} = \mathbf{a}^{i-1}\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}$, which is a rank-1 update, then by substituting $\Delta\mathbf{H}_{i-1} = \mathbf{a}^{i-1}\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}$ into (8.14) we obtain

$$\mathbf{H}_i\Delta\mathbf{x}^{i-1} = \left(\mathbf{H}_{i-1} + \mathbf{a}^{i-1}\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}\right)\Delta\mathbf{x}^{i-1} = \Delta\nabla f^{i-1}, \tag{8.27}$$

from which $\mathbf{a}^{i-1}$ can be solved

$$\mathbf{a}^{i-1} = \frac{\Delta\nabla f^{i-1}}{\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}\Delta\mathbf{x}^{i-1}} - \frac{\mathbf{H}_{i-1}\Delta\mathbf{x}^{i-1}}{\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}\Delta\mathbf{x}^{i-1}}. \tag{8.28}$$

Reconstructing $\mathbf{H}_i$ from (8.26) and (8.28) we obtain

$$\mathbf{H}_i = \mathbf{H}_{i-1} + \left(\frac{\Delta\nabla f^{i-1}}{\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}\Delta\mathbf{x}^{i-1}} - \frac{\mathbf{H}_i\Delta\mathbf{x}^{i-1}}{\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}\Delta\mathbf{x}^{i-1}}\right)\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}, \tag{8.29}$$

with $\mathbf{b}^{i-1}$ free to be chosen. By choosing $\mathbf{b}^{i-1} = (\Delta\nabla f^{i-1} - \mathbf{H}_{i-1}\Delta\mathbf{x}^{i-1})$ symmetry is enforced

$$\mathbf{H}_i = \mathbf{H}_{i-1} + \left(\frac{(\Delta\nabla f^{i-1} - \mathbf{H}_{i-1}\Delta\mathbf{x}^{i-1})(\Delta\nabla f^{i-1} - \mathbf{H}_{i-1}\Delta\mathbf{x}^{i-1})^{\mathrm{T}}}{(\Delta\nabla f^{i-1} - \mathbf{H}_{i-1}\Delta\mathbf{x}^{i-1})^{\mathrm{T}}\Delta x^{i-1})}\right), \tag{8.30}$$

which yields the symmetric rank-1 update investigated by Conn et al. (1991), subject to starting with an initial symmetric matrix $\mathbf{H}_0$.

8.7.4 Symmetric inverse Hessian matrix approximations

Approximating the inverse Hessian $\mathbf{G}_i$ allows for the search direction to be computed using a matrix-vector multiplication as opposed to solving a linear system of equations. $\mathbf{G}_i$ can be approximated incrementally by adding information per iteration to a previous approximation $\mathbf{G}_{i-1}$. Consider the following inverse Hessian update scheme

$$\mathbf{G}_i = \mathbf{G}_{i-1} + \Delta\mathbf{G}_{i-1}, \tag{8.31}$$

substituted into (8.14) and restructured to reflect the inverse Hessian, we obtain

$$\Delta\mathbf{x}^{i-1} = \left(\mathbf{G}_{i-1} + \Delta\mathbf{G}_{i-1}\right)\Delta\nabla f^{i-1}. \tag{8.32}$$

Assume $\Delta \mathbf{G}_{i-1} = \mathbf{a}^{i-1}\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}$, which when substituted into (8.32) gives

$$\Delta \mathbf{x}^{i-1} = \mathbf{G}_{i-1}\Delta\nabla f^{i-1} + \mathbf{a}^{i-1}\left(\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}\Delta\nabla f^{i-1}\right), \tag{8.33}$$

from which $\mathbf{a}^{i-1}$ is isolated to obtain

$$\mathbf{a}^{i-1} = \frac{\Delta \mathbf{x}^{i-1}}{(\mathbf{b}^{i-1})^{\mathrm{T}}\Delta\nabla f^{i-1}} - \frac{\mathbf{G}_{i-1}\Delta\nabla f^{i-1}}{(\mathbf{b}^{i-1})^{\mathrm{T}}\Delta\nabla f^{i-1}}. \tag{8.34}$$

Rewriting (8.31) in terms of only $\mathbf{b}^{i-1}$ gives

$$\mathbf{G}_i = \mathbf{G}_{i-1} + \left(\frac{\Delta x^{i-1}}{(\mathbf{b}^{i-1})^{\mathrm{T}}\Delta\nabla f^{i-1}} - \frac{\mathbf{G}_{i-1}\Delta\nabla f^{i-1}}{(\mathbf{b}^{i-1})^{\mathrm{T}}\Delta\nabla f^{i-1}}\right)\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}}. \tag{8.35}$$

By choosing $\left(\mathbf{b}^{i-1}\right)^{\mathrm{T}} = (\Delta \mathbf{x}^{i-1} - \mathbf{G}_{i-1}\Delta\nabla f^{i-1})$ and substituting the result into (8.35) we obtain

$$\mathbf{G}_{i+1} = \mathbf{G}_i + \left(\frac{(\Delta \mathbf{x}^i - \mathbf{G}_i\Delta\nabla f^i)(\Delta \mathbf{x}^i - \mathbf{G}_i\Delta\nabla f^i)^{\mathrm{T}}}{(\Delta x^i - \mathbf{G}_i\Delta\nabla f^i)^{\mathrm{T}}\Delta\nabla f^i)}\right), \tag{8.36}$$

yielding a symmetric update. This specific update was developed by Fletcher and Powell (1963) and is an adaptation of an original procedure first proposed by Davidon (1959).

8.7.5 Non-symmetric inverse Hessian matrix approximations

More generally the Hessian matrix can be approximated as a non-symmetric Hessian matrix approximation. In this section we demonstrate that conventional conjugate gradient directions imply a non-symmetric Hessian matrix. Using the conventional starting point for conjugate gradient directions as outlined in Section 2.3.2, we express a new search direction $\mathbf{u}^i$ as a linear combination of the gradient descent vector computed at the current minimum point $\mathbf{x}^i$, and the previous search direction $\mathbf{u}^{i-1}$. This then gives

$$\mathbf{u}^i = -\nabla f(\mathbf{x}^i) + \beta_i \mathbf{u}^{i-1}, \tag{8.37}$$

for which we now only have to solve for the scalar β_i such that $\mathbf{u}^i$ is indeed mutually conjugate to the other search directions w.r.t. an assumed matrix $\mathbf{H}_i$. The proposed update formula by Fletcher and Reeves (1964) is given by

$$\beta_i = \frac{\nabla^{\mathrm{T}} f(\mathbf{x}^i) \nabla f(\mathbf{x}^i)}{\nabla^{\mathrm{T}} f(\mathbf{x}^{i-1}) \nabla f(\mathbf{x}^{i-1})}, \tag{8.38}$$

with Polak and Ribiere (1969) proposing an alternative conjugate gradient update

$$\beta_i = \frac{\nabla^{\mathrm{T}} f(\mathbf{x}^i) \left(\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}) \right)}{\nabla^{\mathrm{T}} f(\mathbf{x}^{i-1}) \nabla f(\mathbf{x}^{i-1})}. \tag{8.39}$$

Substituting (8.38) into (8.37) we obtain

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda_i \left(-\nabla f(\mathbf{x}^i) + \frac{\nabla^{\mathrm{T}} f(\mathbf{x}^i) \nabla f(\mathbf{x}^i)}{\nabla^{\mathrm{T}} f(\mathbf{x}^{i-1}) \nabla f(\mathbf{x}^{i-1})} \mathbf{u}^{i-1} \right). \tag{8.40}$$

By factoring $\nabla f(\mathbf{x}^i)$ out of the (8.40) we obtain

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda_i \left(-\mathbf{I} + \frac{\nabla \mathbf{u}^{i-1} \nabla^{\mathrm{T}} f(\mathbf{x}^i)}{\nabla^{\mathrm{T}} f(\mathbf{x}^{i-1}) \nabla f(\mathbf{x}^{i-1})} \right) \nabla f(\mathbf{x}^i). \tag{8.41}$$

By comparing (8.16) with (8.41), we see that (8.41) approximates the inverse of the Hessian matrix

$$(\mathbf{H}_i)^{-1} = \lambda_i \left(\mathbf{I} - \frac{\nabla \mathbf{u}^{i-1} \nabla^{\mathrm{T}} f(\mathbf{x}^i)}{\nabla^{\mathrm{T}} f(\mathbf{x}^{i-1}) \nabla f(\mathbf{x}^{i-1})} \right), \tag{8.42}$$

which by inspection reveals that $(\mathbf{H}_i)^{-1}$ is not symmetric, since $\nabla \mathbf{u}^{i-1} \neq \nabla f(\mathbf{x}^i)$, which implies that $\mathbf{H}_i$ is also not symmetric.

8.7.6 Trust Region Methods and Conservatism

Strategies are required to ensure that sequential approximate optimization methods will terminate and converge. A priority therefore is to ensure that the constructed approximation yields a sufficiently accurate solution. Towards this aim Goldfeld et al. (1966) proposed restricting the step size based on the validity of the approximation over a domain, which was later coined by Sorensen (1982) as the well-known trust region

methods. As a modern alternative to trust region methods, conservatism was proposed by Svanberg (2002). Conservatism requires each proposed update to be feasible as well as an improvement to the previous iterate. The benefit of both approaches is that strong convergence characteristics of the sequential approximation approaches can be proved albeit for often highly restricted classes of functions.

In an effort to enforce conservatism within the context of gradient-only approaches, Wilke et al. (2010) suggested that the directional derivative of the actual problem at the proposed approximate solution along the update step direction should be negative. At iterate i, the proposed solution $\mathbf{x}^{i*}$ is obtained by taking the update step $\mathbf{x}^{i*} - \mathbf{x}^i$ from the previous solution $\mathbf{x}^i$. This update represents descent of $f(\mathbf{x})$ along the direction $\mathbf{x}^{i*} - \mathbf{x}^i$ if

$$\nabla^{\mathrm{T}} f(\mathbf{x}^{i*})(\mathbf{x}^{i*} - \mathbf{x}^i) \leq \nabla^{\mathrm{T}} \tilde{f}(\mathbf{x}^{i*})(\mathbf{x}^{i*} - \mathbf{x}^i) = 0. \tag{8.43}$$

Accordingly, any gradient-only approximation may be defined as conservative if (8.43) holds.

This gradient-only definition of conservatism is similar in intent to that of Svanberg's function value based definition that requires that the function value $f(\mathbf{x}^{i*})$ improve on that of the previous iterate. In the gradient-only approach only updates $\mathbf{x}^{i*}$ for which the genuine quality measure,

$$\nabla^{\mathrm{T}} f(\mathbf{x}^{i*})(\mathbf{x}^{i*} - \mathbf{x}^i),$$

is less than or equal to the approximated quality measure

$$\nabla^{\mathrm{T}} \tilde{f}(\mathbf{x}^{i*})(\mathbf{x}^{i*} - \mathbf{x}^i),$$

are accepted. Although no formal proofs are presented here, Wilke et al. (2013b) showed that this definition of conservatism guarantees convergence for certain classes of functions, e.g. smooth convex functions. It is also important to note that for non-smooth and discontinuous functions in general this definition falls short, and is not sufficient to guarantee convergence. It is important to note that, although strong theoretical evidence is lacking, this gradient-only definition of conservatism suffices in general to achieve convergence for practical engineering problems. In sequential approximate optimization, termination and convergence may be affected through this notion of conservatism. Therefore the minimizer

Algorithm 8.3 Affecting conservatism in gradient-only sequential approximate optimization using constant diagonal (8.18) Hessian matrix approximations.

Initialization: Given $\mathbf{x}^0$, select the real constant $\epsilon > 0$, initial curvature $c_0 > 0$ and conservatism parameter $\gamma > 1$. Select the maximum number of iterations i_{max}. Set $i := 0$, $l := 0$ and perform the following steps:

1. **Gradient evaluation:** Compute $\nabla f(\mathbf{x}^i)$.
2. **Approximate optimization:** Construct local approximate subproblem (8.18) at $\mathbf{x}^i$. Solve this subproblem analytically, to arrive at $\mathbf{x}^{i*}$.
3. **Evaluation:** Compute $\nabla f(\mathbf{x}^{i*})$.
4. **Test if $\mathbf{x}^{i*}$ is acceptable:** if (8.43) is satisfied, go to Step 6.
5. **Initiate an inner loop to effect conservatism:**
   (a) Set $l := l + 1$.
   (b) Set $c_i := \gamma c_i$.
   (c) Goto Step 2.
6. **Move to the new iterate:** Set $\mathbf{x}^{i+1} := \mathbf{x}^{i*}$.
7. **Convergence test:** if $\|\mathbf{x}^{i+1} - \mathbf{x}^i\| \leq \epsilon$, OR $i = i_{max}$, stop.
8. **Initiate an additional outer loop:** Set $i := i + 1$ and go to Step 1.

of the subproblem $\mathbf{x}^{i*}$ is accepted i.e. $\mathbf{x}^{i+1} := \mathbf{x}^{i*}$ only if $\mathbf{x}^{i*}$ is found to be a gradient-only conservative point. This modification to the gradient-only sequential approximate optimization is listed Algorithm 8.3.

8.8 Gradient-only optimization problem

An important consideration for a holistic understanding of gradient-only approaches, is to understand the characteristics of the designs to which gradient-only strategies converge. This would allow us to better differentiate gradient-only strategies from conventional minimization strategies when step discontinuous functions are considered. We therefore formally define the underlying optimization problem that is consistent with solution strategies that only consider gradient-only information.

Reconsider the derivatives presented in Figures 8.5 (b) and (d), that depict the smooth and piece-wise smooth step discontinuous derivative responses when the same problem is numerically integrated using different numerical strategies. Although conventional interpretations of the smooth derivative function highlights the design with zero slope and the lack thereof for the step discontinuous derivative function, there is a consistent interpretation between the smooth and step discontinuous derivative functions depicted in Figures 8.5 (b) and (d). This interpretation acknowledges that the smooth and step discontinuous derivative functions both change sign from negative to positive only once as λ increases. Therefore, if we define this point as the solution to the optimization problem then we have (i) a unique solution that is defined for both derivative functions based solely on first order information and (ii) the solution defines a minimum when estimated from only first order information as second order (curvature) information is implied by requiring the sign to change from negative to positive with increasing λ.

As illustration of (ii), consider Figure 8.5 (d). The sign change from negative to positive as λ increases is at $\lambda_g^* \approx 0.57$. Consider any point, λ_v to the left of λ_g^* i.e. $\lambda_v < \lambda_g^*$, then the direction is given by $d_v = \lambda_v - \lambda_g^* < 0$. The directional derivative is given by the projection of the derivative $\frac{dE(\lambda_v)}{d\lambda}$ computed at λ_v onto d_v, i.e. by $d_v \frac{dE(\lambda_v)}{d\lambda}$. Since $\frac{dE(\lambda_v)}{d\lambda} < 0$ for $\lambda_v < \lambda_g^*$, the directional derivative is positive. Similarly, the directional derivative for λ_v to the right of λ_g^* i.e. $\lambda_v > \lambda_g^*$ is also only positive. This implies that first order information estimates the function value to only increase irrespective of the direction of departure from λ_g^*. In contrast, when considering Figures 8.5 (a) and (c) it is evident that the function decreases but only as a result of step discontinuities and not because of the trends of the piece-wise smooth sections indicating

descent. We define λ_g^* as a strict *non-negative associated gradient (or derivative) projection point* (Wilke et al. (2013b)). This requires the directional derivative at any point λ_v to be positive, where λ_v is in the vicinity of λ_g^*, and the direction defined by $\lambda_v - \lambda_g^*$.

In general, given a real-valued function $f : X \subset \mathbb{R}^n \rightarrow \mathbb{R}$, the general unconstrained gradient-only optimization problem is to find a *non-negative associated gradient projection point* $\mathbf{x}_g^* \in X$ such that for every $\mathbf{u} \in \{\mathbf{y} \in \mathbb{R}^n \mid \|\mathbf{y}\| = 1\}$ there exists a real number $r_u > 0$ for which the following holds:

$$\nabla_A{}^{\mathrm{T}} f(\mathbf{x}_g^* + \lambda \mathbf{u})\mathbf{u} \geq 0 \; \forall \; \lambda \in (0, r_u].$$

This allows us to determine distinct candidate solutions to an unconstrained gradient-only optimization problem. It is important to note that when multiple candidate solutions exist additional information may be required, after obtaining these solutions using only first order information, in order to uniquely obtain the best solution.

8.9 Exercises

The reader is encouraged to employ a convenient computing environment to complete the exercises. With Python being freely available it is recommended to be used as outlined in Chapter 9.

8.9.1 Consider the Lotka-Volterra system with unknown parameter λ:

$$\begin{aligned} \frac{dz(z,y,t)}{dt} &= (1-\lambda)z(t) - 0.3z(t)y(t) \\ \frac{dy(z,y,t)}{dt} &= z(t)y(t) - y(t), \end{aligned}$$

with the two initial conditions $z(0) = 0.9$ and $y(0) = 0.9$ integrated using the forward Euler scheme, over an eight second interval, using 50 000 equally spaced time steps. Given $z(8) = 0.722962$ and $y(8) = 1.110567$ plot the sum of the errors squared objective function given in (8.3) for λ between 0 and 1 using 101 equally spaced points.

8.9.2 Consider the Lotka-Volterra system with unknown parameter λ:

$$\begin{aligned} \frac{dz(z,y,t)}{dt} &= (1-\lambda)z(t) - 0.3z(t)y(t) \\ \frac{dy(z,y,t)}{dt} &= z(t)y(t) - y(t), \end{aligned}$$

with the two initial conditions $z(0) = 0.9$ and $y(0) = 0.9$ integrated using the forward Euler scheme, over an eight second interval, that starts with 50 equally spaced time steps and adding 10 time steps every time the computation is done for a new λ. Given $z(8) = 0.722962$ and $y(8) = 1.110567$ plot the sum of the errors squared objective function given in (8.3) for λ between 0 and 1 using 101 equally spaced points.

8.9.3 Optimize the problem outlined in Exercise 8.9.1 using the Golden section strategy and the gradient-only bisection approach using 100 random starts each. Use the same random starting points for the two line search strategies. Compare the obtained results in terms of the apparent optimal λ, required number of function and derivative evaluations to solve the problem 100 times.

8.9.4 Optimize the problem outlined in Exercise 8.9.2 using the Golden section strategy and the gradient-only bisection approach using 100 random starts each. Use the same random starting points for the two line search strategies. Compare the obtained results in terms of the apparent optimal λ, required number of function and derivative evaluations to solve the problem 100 times.

8.9.5 Critically compare the results obtained in Exercises 8.9.3 and 8.9.4.

8.9.6 Consider the Lotka-Volterra system with unknown parameters λ and δ:

$$\begin{aligned} \frac{dz(z,y,t)}{dt} &= (1-\lambda)z(t) - 0.3z(t)y(t) \\ \frac{dy(z,y,t)}{dt} &= \delta z(t)y(t) - y(t), \end{aligned}$$

with the two initial conditions $z(0) = 0.9$ and $y(0) = 0.9$ integrated using the forward Euler scheme, over an eight second interval, using 50 000 equally spaced time steps. Given $z(8) = 0.722962$ and $y(8) = 1.110567$ plot the sum of the errors squared

objective function given in (8.3) for λ between 0 and 1 and δ between 0.5 and 1.5 using 101 equally spaced points.

8.9.7 Consider the Lotka-Volterra system with unknown parameters λ and δ:

$$\begin{aligned} \frac{dz(z,y,t)}{dt} &= (1-\lambda)z(t) - 0.3z(t)y(t) \\ \frac{dy(z,y,t)}{dt} &= \delta z(t)y(t) - y(t), \end{aligned}$$

with the two initial conditions $z(0) = 0.9$ and $y(0) = 0.9$ integrated using the forward Euler scheme, over an eight second interval, that starts with 50 equally spaced time steps and adding 10 time steps every time the computation is done for a new (λ, δ) pair. Given $z(8) = 0.722962$ and $y(8) = 1.110567$ plot the sum of the errors squared objective function given in (8.3) for λ between 0 and 1 and δ between 0.5 and 1.5 using 101 equally spaced points.

8.9.8 Optimize the problem outlined in Exercise 8.9.6 using the Golden section strategy and the gradient-only bisection approach using 100 random starts each. Use the same random starting points for the two line search strategies. Compare the obtained results in terms of the apparent optimal λ, required number of function and derivative evaluations to solve the problem 100 times.

8.9.9 Optimize the problem outlined in Exercise 8.9.7 using the Golden section strategy and the gradient-only bisection approach using 100 random starts each. Use the same random starting points for the two line search strategies. Compare the obtained results in terms of the apparent optimal λ, required number of function and derivative evaluations to solve the problem 100 times.

8.9.10 Consider the Lotka-Volterra system with four unknown parameters λ, δ, β and γ :

$$\begin{aligned} \frac{dz(z,y,t)}{dt} &= (1-\lambda)z(t) - \beta z(t)y(t) \\ \frac{dy(z,y,t)}{dt} &= \delta z(t)y(t) - \gamma y(t), \end{aligned}$$

with the two initial conditions $z(0) = 0.9$ and $y(0) = 0.9$ integrated using the forward Euler scheme, over an eight second interval, that starts with 50 equally spaced time steps. For every 10 evaluations of a $(\lambda, \delta, \beta, \gamma)$ increase the number of time steps by 10. Given $z(8) = 0.722962$ and $y(8) = 1.110567$ solve the problem to find the optimal (λ, β, δ and γ) using random starting points between 0 and 1 for, λ and β, and between 0.5 and 1.5 for δ and γ.

8.9.11 Construct two third order polynomial approximations of the objective in Exercise 8.9.1. For the first approximation use only zero order information, while for the second approximation use only first order information. Compare the two approximations with each other.

8.9.12 Construct two third order polynomial approximations of the objective in Exercise 8.9.2. For the first approximation use only zero order information, while for the second approximation use only first order information. Compare the two approximations with each other.

8.9.13 Optimize the piece-wise smooth step discontinuous quadratic function given by (7.14) for $n = 2$ using a gradient-only symmetric Hessian matrix approximation. Compare the obtained optimum against the graphical solution of the problem.

8.9.14 Optimize the piece-wise smooth step discontinuous quadratic function given by (7.14) for $n = 4$ using a gradient-only symmetric Hessian matrix approximation.

8.9.15 Compare the symmetric Hessian matrix approximation in Exercises 8.9.13 and 8.9.14 against the actual Hessian of the piece-wise smooth step discontinuous quadratic function.