Numerical Differentiation

using Finite Difference and Complex-Step Approximations

Tamas Kis | tamas.a.kis@outlook.com

Tamas Kis
https://tamaskis.github.io

Contents

List of Algorithms

Forward Difference Approximations

Central Difference Approximations

Complex-Step Approximations

Complexified Functions

PART I

Definitions

1 Differentiation

1.1 Derivative of a Univariate, Scalar-Valued Function

Consider a univariate, scalar-valued function $f:\mathbb{R}\to\mathbb{R}$. Its derivative with respect to $x\in\mathbb{R}$ is defined as

$$\frac{df}{dx}=\lim_{h\to 0}\left[\frac{f(x+h)-f(x)}{h}\right]$$

The derivative of f with respect to x, *evaluated at* the point $x=x_0$, is then

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0}=\lim_{h\to 0}\left[\frac{f(x_0+h)-f(x_0)}{h}\right]} \tag{1.1}$$

1.2 Derivative of a Univariate, Vector-Valued Function

Consider a univariate, vector-valued function $\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$. Its derivative with respect to $x\in\mathbb{R}$ is defined as [28, pp. 895-896]

$$\frac{d\mathbf{f}}{dx}=\lim_{h\to 0}\left[\frac{\mathbf{f}(x+h)-\mathbf{f}(x)}{h}\right]$$

The derivative of $\mathbf{f}$ with respect to x, *evaluated at* the point $x=x_0$, is then

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}=\lim_{h\to 0}\left[\frac{\mathbf{f}(x_0+h)-\mathbf{f}(x_0)}{h}\right]} \tag{1.2}$$

Alternatively, we can note that the function, $\mathbf{f}$, can be written as

$$\mathbf{f}(x)=\begin{bmatrix} f_1(x)\\ \vdots \\ f_m(x)\end{bmatrix}$$

The derivative of $\mathbf{f}$ with respect to $x \in \mathbb{R}$, evaluated at the point $x = x_0$, is then

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} = \begin{bmatrix} \left.\dfrac{df_1}{dx}\right|_{x=x_0} \\ \vdots \\ \left.\dfrac{df_m}{dx}\right|_{x=x_0} \end{bmatrix}} \tag{1.3}$$

The individual derivatives in Eq. (1.3) are defined using Eq. (1.1). In a more compact form, this can be expressed as the summation [32]

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} = \sum_{j=1}^{m} \mathbf{e}_j \left.\frac{df_j}{dx}\right|_{x=x_0}} \tag{1.4}$$

1.3 Partial Derivative of a Multivariate, Scalar-Valued Function

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. The partial derivative of f with respect to $x_k \in \mathbb{R}$ is defined as [23]

$$\frac{\partial f}{\partial x_k} = \lim_{h \to 0} \left[\frac{f(\mathbf{x} + h\mathbf{e}_k) - f(\mathbf{x})}{h}\right]$$

where $\mathbf{e}_k$ is the kth standard basis vector.

$$\boxed{\mathbf{e}_k = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow k\text{th element} \\ \\ \\ \end{matrix}} \tag{1.5}$$

Note that $\mathbf{e}_k$ is also the kth column of the $n \times n$ identity matrix, $\mathbf{I}_{n \times n}$.

The partial derivative of f with respect to x_k, *evaluated at* the point $\mathbf{x} = \mathbf{x}_0$, is then

$$\boxed{\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} = \lim_{h \to 0} \left[\frac{f(\mathbf{x}_0 + h\mathbf{e}_k) - f(\mathbf{x}_0)}{h}\right]} \tag{1.6}$$

1.4 Partial Derivative of a Multivariate, Vector-Valued Function

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Its partial derivative with respect to $x_k \in \mathbb{R}$ is defined as [24]

$$\frac{\partial \mathbf{f}}{\partial x_k} = \lim_{h \to 0} \left[\frac{\mathbf{f}(\mathbf{x} + h\mathbf{e}_k) - \mathbf{f}(\mathbf{x})}{h}\right]$$

The partial derivative of f with respect to x_k, *evaluated at* the point $\mathbf{x} = \mathbf{x}_0$, is then

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} = \lim_{h \to 0} \left[\frac{\mathbf{f}(\mathbf{x}_0 + h\mathbf{e}_k) - \mathbf{f}(\mathbf{x}_0)}{h}\right]} \tag{1.7}$$

Alternatively, we can note that the function, $\mathbf{f}$, can be written as

$$\mathbf{f}(\mathbf{x})=\begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

The partial derivative of $\mathbf{f}$ with respect to $x_k \in \mathbb{R}$, evaluated at the point $\mathbf{x}=\mathbf{x}_0$, is then

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}=\begin{bmatrix} \left.\frac{\partial f_1}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots \\ \left.\frac{\partial f_m}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}} \tag{1.8}$$

The individual partial derivatives in Eq. (1.8) are defined using Eq. (1.6). In a more compact form, this can be expressed as the summation [32]

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}=\sum_{j=1}^{m}\mathbf{e}_j\left.\frac{\partial f_j}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}} \tag{1.9}$$

1.5 Gradient of a Multivariate, Scalar-Valued Function

Consider a multivariate, scalar-valued function $f:\mathbb{R}^n\to\mathbb{R}$. The gradient of f with respect to $\mathbf{x}\in\mathbb{R}^n$, evaluated at $\mathbf{x}=\mathbf{x}_0$, is defined as [4]

$$\boxed{\mathbf{g}(\mathbf{x}_0)=\nabla f(\mathbf{x}_0)=\begin{bmatrix} \left.\frac{\partial f}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots \\ \left.\frac{\partial f}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}} \tag{1.10}$$

The individual partial derivatives in Eq. (1.10) are defined using Eq. (1.6).

1.6 Directional Derivative of a Multivariate, Scalar-Valued Function

Consider a multivariate, scalar-valued function $f:\mathbb{R}^n\to\mathbb{R}$. The directional derivative of f with respect to $\mathbf{x}\in\mathbb{R}^n$ in the direction of $\mathbf{v}\in\mathbb{R}^n$ is defined as

$$\nabla_{\mathbf{v}} f(\mathbf{x})=\lim_{h\to 0}\left[\frac{f(\mathbf{x}+h\mathbf{v})-f(\mathbf{x})}{h}\right]$$

The directional derivative of f with respect to $\mathbf{x}$ in the direction of $\mathbf{v}$, *evaluated at* the point $\mathbf{x}=\mathbf{x}_0$, is then

$$\boxed{\nabla_{\mathbf{v}} f(\mathbf{x}_0)=\lim_{h\to 0}\left[\frac{f(\mathbf{x}_0+h\mathbf{v})-f(\mathbf{x}_0)}{h}\right]} \tag{1.11}$$

Alternatively, the directional derivative can be computed as the inner product between the gradient of f with respect to $\mathbf{x}$ and the direction, $\mathbf{v}$ [15, p. 22].

$$\nabla_{\mathbf{v}} f(\mathbf{x})=\nabla f(\mathbf{x})^T\mathbf{v}$$

The directional derivative of f with respect to $\mathbf{x}$ in the direction of $\mathbf{v} \in \mathbb{R}^n$, *evaluated at* the point $\mathbf{x} = \mathbf{x}_0$, is then

$$\boxed{\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \nabla f(\mathbf{x}_0)^T \mathbf{v}} \tag{1.12}$$

Note that gradients are discussed in Section 1.5.

The most convenient definition of the directional derivative for computational implementation can be formed by first defining the univariate, scalar-valued auxiliary function $g : \mathbb{R} \to \mathbb{R}$.

$$\boxed{g(\alpha) = f(\mathbf{x}_0 + \alpha \mathbf{v})} \tag{1.13}$$

The derivative of g with respect to α is defined as[1]

$$\frac{dg}{d\alpha} = \lim_{h \to 0} \left[\frac{g(\alpha + h) - g(\alpha)}{h} \right]$$

The derivative of g with respect to α, *evaluated at* the point $\alpha = 0$, is then

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} = \lim_{h \to 0} \left[\frac{g(h) - g(0)}{h} \right] = \lim_{h \to 0} \left[\frac{g(h) - g(0)}{h} \right]$$

Using the fact that $g(\alpha) = f(\mathbf{x}_0 + \alpha\mathbf{v})$ (so $g(h) = f(\mathbf{x}_0 + h\mathbf{v})$ and $g(0) = f(\mathbf{x}_0)$),

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} = \lim_{h \to 0} \left[\frac{f(\mathbf{x}_0 + h\mathbf{v}) - f(\mathbf{x}_0)}{h} \right]$$

Comparing this to Eq. (1.11), we can clearly see that

$$\boxed{\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \left.\frac{dg}{d\alpha}\right|_{\alpha=0}} \tag{1.14}$$

1.7 Jacobian of a Multivariate, Vector-Valued Function

The Jacobian of a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $\mathbf{x} \in \mathbb{R}^n$ is defined as

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

The Jacobian can also be written in terms of its column vectors as [12]

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix}$$

The Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$, *evaluated at* the point $\mathbf{x} = \mathbf{x}_0$, is then

$$\boxed{\mathbf{J}(\mathbf{x}_0) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial f_1}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial f_m}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial f_m}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix} = \begin{bmatrix} \left.\frac{\partial \mathbf{f}}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial \mathbf{f}}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}} \tag{1.15}$$

The partial derivatives forming each column of the Jacobian are discussed in Section 1.4.

[1] See Section 1.1

1.8 Hessian of a Multivariate, Scalar-Valued Function

The Hessian of a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

In a more compact form, we can write the (j,k)th element of the Hessian as

$$[\mathbf{H}]_{j,k} = \frac{\partial^2 f}{\partial x_j \partial x_k}$$

The Hessian of f with respect to $\mathbf{x}$, *evaluated at* the point $\mathbf{x} = \mathbf{x}_0$, is then

$$\boxed{\mathbf{H}(\mathbf{x}_0) = \begin{bmatrix} \left.\frac{\partial^2 f}{\partial x_1^2}\right|_{\mathbf{x}=\mathbf{x}_0} & \left.\frac{\partial^2 f}{\partial x_1 \partial x_2}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial^2 f}{\partial x_1 \partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \left.\frac{\partial^2 f}{\partial x_2 \partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} & \left.\frac{\partial^2 f}{\partial x_2^2}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial^2 f}{\partial x_2 \partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots & \vdots & \ddots & \vdots \\ \left.\frac{\partial^2 f}{\partial x_n \partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} & \left.\frac{\partial^2 f}{\partial x_n \partial x_2}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\frac{\partial^2 f}{\partial x_n^2}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}} \tag{1.16}$$

where the (j,k)th element is given by

$$\boxed{[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial^2 f}{\partial x_j \partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}} \tag{1.17}$$

Note that from Schwarz's theorem, we know

$$\therefore \frac{\partial^2 f}{\partial x_j \partial x_k} = \frac{\partial^2 f}{\partial x_k \partial x_j}$$

which implies that the Hessian is symmetric and satisfies the property [6]

$$\boxed{[\mathbf{H}]_{j,k} = [\mathbf{H}]_{k,j}} \tag{1.18}$$

Since the Hessian is symmetric, we only have to evaluate the derivatives in the upper triangle of the matrix (shown in red below) to form the full matrix:

$$\begin{bmatrix} \color{red}{H_{1,1}} & \color{red}{H_{1,2}} & \color{red}{\cdots} & \color{red}{H_{1,n}} \\ H_{2,1} & \color{red}{H_{2,2}} & \color{red}{\cdots} & \color{red}{H_{2,n}} \\ \vdots & \vdots & \color{red}{\ddots} & \color{red}{\vdots} \\ H_{n,1} & H_{n,2} & \cdots & \color{red}{H_{n,n}} \end{bmatrix}$$

1.9 Vector Hessian of a Multivariate, Vector-Valued Function

Consider a multivariate, vector-valued function $\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$.

$$\mathbf{f}(\mathbf{x})=\begin{bmatrix}f_1(\mathbf{x})\\ \vdots\\ f_m(\mathbf{x})\end{bmatrix}$$

Its **vector Hessian**[2] with respect to $\mathbf{x}\in\mathbb{R}^n$ is defined as [6, 7]

$$\mathbf{H}(\mathbf{f})=\left(\mathbf{H}(f_1),\dots,\mathbf{H}(f_m)\right)$$

where $\mathbf{H}(f_k)$ represents the Hessian of the kth component of $\mathbf{H}$. The vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x}$, *evaluated at* the point $\mathbf{x}=\mathbf{x}_0$, is then

$$\boxed{\mathbf{H}(\mathbf{f}(\mathbf{x}_0))=\left(\mathbf{H}(f_1(\mathbf{x}_0)),\dots,\mathbf{H}(f_m(\mathbf{x}_0))\right)}\tag{1.19}$$

Each “page” of this 3D array (i.e. the kth page would be $\mathbf{H}_k(\mathbf{f}(\mathbf{x}_0))$) is a Hessian matrix defined by Eq. (1.16).

The vector Hessian of $\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$ is not a $n\times n$ matrix as was the case for $f:\mathbb{R}^n\to\mathbb{R}$, but rather it is a *collection*, *array*, or *vector* of $n\times n$ matrices [6]. Generally speaking, $\mathbf{H}(\mathbf{f})\in\mathbb{R}^{n\times n\times m}$ can be treated as a 3rd-order tensor. For example, the Taylor series expansion of a vector-valued function $\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$ about the point $\mathbf{a}\in\mathbb{R}^n$ is defined[3] as

$$\mathbf{f}(\mathbf{x})\approx\mathbf{f}(\mathbf{a})+\mathbf{J}(\mathbf{a})(\mathbf{x}-\mathbf{a})+\frac{1}{2}\sum_{k=1}^{m}\mathbf{e}_k(\mathbf{x}-\mathbf{a})^T\mathbf{H}(f_k(\mathbf{a}))(\mathbf{x}-\mathbf{a})^T\tag{1.20}$$

where $\mathbf{e}_k$ is the kth standard basis vector [26, p. 413], i.e.

$$\mathbf{e}_k=\begin{bmatrix}0\\ \vdots\\ 1\\ \vdots\\ 0\end{bmatrix}\begin{matrix}\\ \\ \leftarrow k\text{th element}\\ \\ \\ \end{matrix}$$

[2] In many sources, this quantity is nameless; for example, in [6], $\mathbf{H}(\mathbf{f})$ is referred to as “a collection of second order partial derivatives” and as “an array of m Hessian matrices”. Typically, is not referred to as a Hessian, since a Hessian is a matrix, whereas $\mathbf{H}(\mathbf{f})$ can be considered a third-order tensor. [7] suggests that $\mathbf{H}(\mathbf{f})$ is commonly referred to as the vector Hessian.

[3] It is important to note that taking the Taylor series of a vector-valued function is rather uncommon in engineering applications. In higher-level mathematics, this is known as a jet [13, 29, 30]. Eq. (1.20) was used in the context of deriving/defining a second-order extended Kalman filter for state estimation applications.

2

Finite Difference Approximations

2.1 Forward Difference Approximation

Consider a univariate scalar-valued function $f : \mathbb{R} \to \mathbb{R}$. Recall that its Taylor series expansion about the point $x = a$ is given by

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{6}f'''(a)(x-a)^3 + \cdots$$

Let's replace x with $x + h$ and a with x. Then

$$\begin{aligned} f(x+h) &= f(x) + f'(x)[(x+h) - x] + \frac{1}{2}f''(x)[(x+h) - x]^2 + \cdots \\ &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots \end{aligned}$$

Solving for $f'(x)$,

$$\begin{aligned} hf'(x) &= f(x+h) - \left[f(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots\right] \\ f'(x) &= \frac{f(x+h)}{h} - \frac{1}{h}\left[f(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots\right] \\ &= \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) - \cdots \\ &= \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \end{aligned}$$

From this, we can directly extract the approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

The derivative of f with respect to x, *evaluated at* the point $x = x_0$, is then

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{f(x_0+h) - f(x_0)}{h}} \tag{2.1}$$

This approximation is known as the **forward difference approximation** since the additional point we are using to construct the approximation is *forward* of (i.e. greater than) the evaluation point, x_0. Note that we can also obtain this directly from the definition of the derivative. If h is small, then

$$\frac{df}{dx} = \lim_{h \to 0} \left[\frac{f(x+h) - f(x)}{h} \right] \approx \frac{f(x+h) - f(x)}{h}$$

Since the error term associated with the forward difference approximation is $\mathcal{O}(h)$, the error in the approximation decreases linearly as h approaches 0. Therefore, the forward difference approximation is a first-order approximation.

The forward difference approximation can be visualized using the stencil shown in Fig. 2.1.

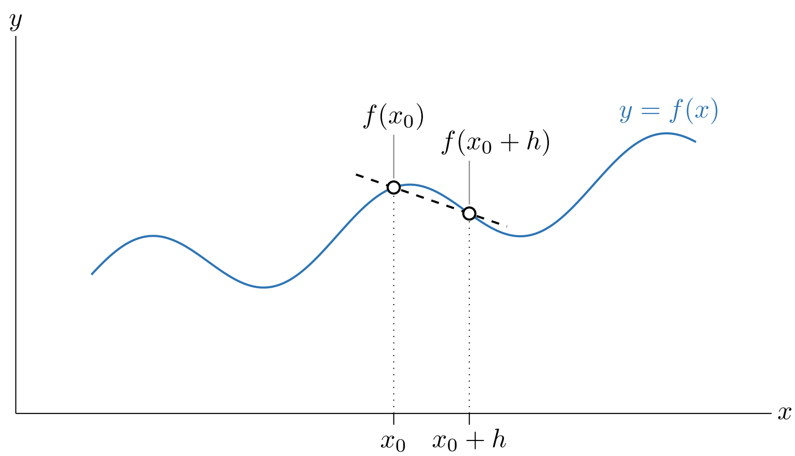

Figure 2.1: Forward difference approximation.

2.2 Backward Difference Approximation

Consider a univariate scalar-valued function $f : \mathbb{R} \to \mathbb{R}$. Recall that its Taylor series expansion about the point $x = a$ is given by

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{6}f'''(a)(x-a)^3 + \cdots$$

Let's replace x with $x - h$ and a with x. Then

$$\begin{aligned} f(x-h) &= f(x) + f'(x)[(x-h) - x] + \frac{1}{2}f''(x)[(x-h) - x]^2 + \cdots \\ &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) - \cdots \end{aligned}$$

Solving for $f'(x)$,

$$hf'(x) = -f(x-h) + \left[f(x) - \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots \right]$$

$$
\begin{aligned}
f'(x) &= -\frac{f(x-h)}{h} + \frac{1}{h}\left[f(x) - \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots\right] \\
&= \frac{f(x) - f(x-h)}{h} - \frac{h}{2}f''(x) + \frac{h^2}{6}f'''(x) + \cdots \\
&= \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h)
\end{aligned}
$$

From this, we can directly extract the approximation

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

The derivative of f with respect to x, *evaluated at* the point $x = x_0$, is then

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{f(x_0) - f(x_0 - h)}{h}} \tag{2.2}$$

This approximation is known as the **backward difference approximation** since the additional point we are using to construct the approximation is *backward* of (i.e. less than) the evaluation point, x_0.

Since the error term associated with the backward difference approximation is $\mathcal{O}(h)$, the error in the approximation decreases linearly as h approaches 0. Therefore, the backward difference approximation is a first-order approximation.

The backward difference approximation can be visualized using the stencil shown in Fig. 2.2.

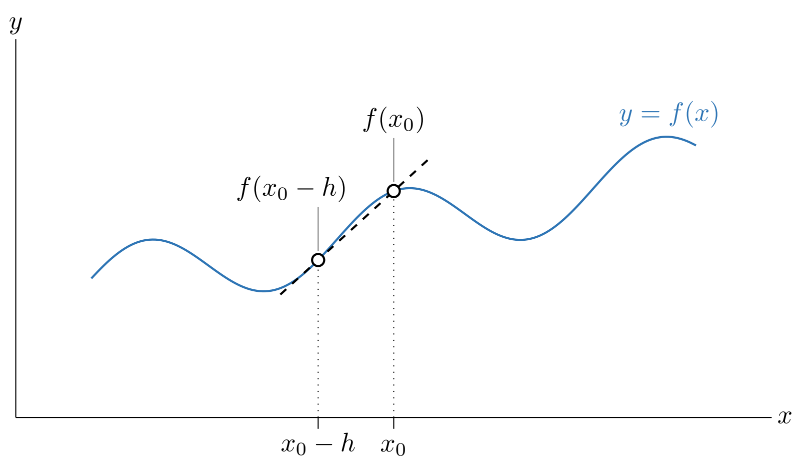

Figure 2.2: Backward difference approximation.

2.3 Central Difference Approximation

Consider a univariate scalar-valued function $f : \mathbb{R} \to \mathbb{R}$. From the Taylor series expansions developed in Sections 2.1 and 2.2, we can write

$$
\begin{aligned}
f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots \\
f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) - \cdots
\end{aligned}
$$

It follows that

$$\begin{aligned} f(x+h) - f(x-h) &= \left[f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots\right] \\ &\quad - \left[f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(x) - \cdots\right] \\ &= 2hf'(x) + \frac{h^3}{3}f'''(x) + \cdots \end{aligned}$$

Solving for $f'(x)$,

$$\begin{aligned} f'(x) &= \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f'''(x) + \cdots \\ &= \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h) \end{aligned}$$

From this, we can directly extract the approximation

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

The derivative of f with respect to x, *evaluated at* the point $x = x_0$, is then

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{f(x_0+h) - f(x_0-h)}{2h}} \tag{2.3}$$

This approximation is known as the **central difference approximation** since the point at which we are evaluating the derivative, x_0, is *centered* between the two points used to construct the approximation.

Since the error term associated with the forward difference approximation is $\mathcal{O}(h^2)$, the error in the approximation decreases quadratically as h approaches 0. Therefore, the central difference approximation is a second-order approximation.

The central difference approximation can be visualized using the stencil shown in Fig. 2.3.

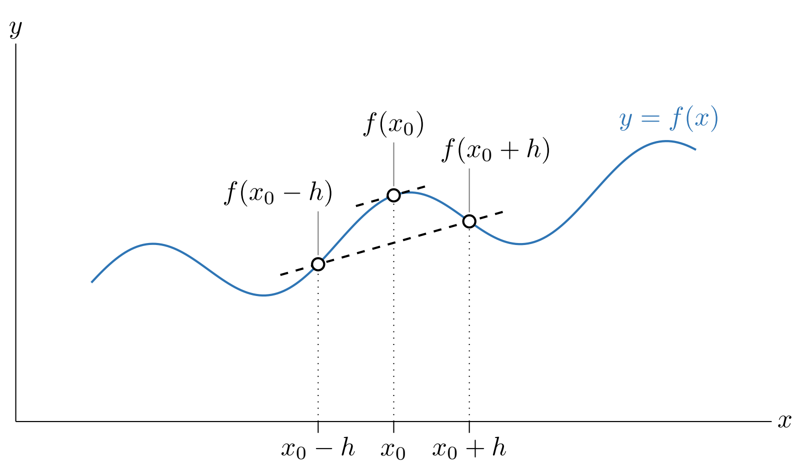

Figure 2.3: Central difference approximation.

2.4 Choosing a Step Size

To improve our derivative estimate, we want to reduce the step size, h, as much as possible. However, as h is reduced, the finite difference approximations become dominated by subtractive cancellation errors [18, pp. 229–230]. Once we decrease h beyond some lower bound, the finite difference approximations will actually start to get worse.

The machine zero, ε, is defined as the smallest positive number ε such that $1+\varepsilon>1$ when calculated using a computer). For double precision, $\varepsilon = 2^{-52} \approx 2.2\times10^{-16}$ [18, p. 55]. The optimal step size for the forward and backward difference approximations is approximately $\sqrt{\varepsilon}$, while the optimal step size for the central difference approximation is approximately $\varepsilon^{1/3}$ [18, p. 230]. These are the values used by default for h in the *Numerical Differentiation Toolbox*.

However, it is also helpful to scale the step size based on the value of x_0 (i.e. the point at which we are differentiating). [18, p. 231] defines h as the **relative step size** and suggests using the finite difference approximations

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}} \tag{2.4}$$

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{f(x_0)-f(x_0-\Delta x)}{\Delta x}} \tag{2.5}$$

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{f(x_0+\Delta x)-f(x_0-\Delta x)}{2\Delta x}} \tag{2.6}$$

where the **absolute step size**, Δx, is defined as

$$\boxed{\Delta x = h(1+|x_0|)} \tag{2.7}$$

3

The Complex-Step Approximation

3.1 Definition

Recall from Section 2.1 that the Taylor series expansion for a univariate, scalar-valued function $f:\mathbb{R}\to\mathbb{R}$ is given by

$$f(x+h)=f(x)+hf'(x)+\frac{h^2}{2}f''(x)+\frac{h^3}{6}f'''(x)+\cdots$$

However, if we instead take a step, ih, in the imaginary direction (where $i=\sqrt{-1}$), then the Taylor series expansion is

$$\begin{aligned}f(x+ih)&=f(x)+ihf'(x)-\frac{h^2}{2}f''(x)-\frac{ih^3}{6}f'''(x)+\cdots\\&=\left[f(x)-\frac{h^2}{2}f''(x)+\cdots\right]+i\left[hf'(x)-\frac{h^3}{6}f'''(x)+\cdots\right]\end{aligned}$$

Taking the imaginary component of each side,

$$\operatorname{Im}\left[f(x+ih)\right]=hf'(x)-\frac{h^3}{6}f'''(x)+\cdots$$

Solving for $f'(x)$,

$$hf'(x)=\operatorname{Im}\left[f(x+ih)\right]+\frac{h^3}{6}f'''(x)+\cdots$$

$$\begin{aligned}f'(x)&=\frac{\operatorname{Im}\left[f(x+ih)\right]}{h}+\frac{h^2}{6}f'''(x)+\cdots\\&=\frac{\operatorname{Im}\left[f(x+ih)\right]}{h}+\mathcal{O}(h^2)\end{aligned}$$

From this, we can directly extract the approximation

$$f'(x)\approx\frac{\operatorname{Im}\left[f(x+ih)\right]}{h}$$

The derivative of f with respect to x, *evaluated at* the point $x=x_0$, is then [15, p. 25][19, 27]

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0}\approx\frac{\operatorname{Im}\left[f(x+ih)\right]}{h}}\tag{3.1}$$

This approximation is known as the **complex-step approximation** since the additional point we are using to construct the approximation is *forward* of (i.e. greater than) the evaluation point, x_0. Note that we can also obtain this directly from the definition of the derivative. If h is small, then

$$\frac{df}{dx} = \lim_{h \to 0} \left[\frac{f(x+h) - f(x)}{h}\right] \approx \frac{f(x+h) - f(x)}{h}$$

Since the error term associated with the complex-step approximation is $\mathcal{O}(h^2)$, the error in the approximation decreases quadratically as h approaches 0. Therefore, the complex-step approximation is a second-order approximation.

3.1.1 Limitation: Higher-Order Derivatives

It is relatively straightforward to develop higher-order finite difference approximation; we could even use nested calls on a finite difference differentiation algorithm such as Algorithm 13, with the caveat that subtractive cancellation errors (discussed in Section 2.4). However, we cannot use nested calls on a complex-step differentiation algorithm to obtain higher-order derivatives. Consider trying to approximate a second derivative by nesting one complex-step approximation within another:

$$\left.\frac{df}{dx}\right|_{x=x_0} \approx \left.\frac{d}{dx}\right|_{x=x_0} \left[\frac{\operatorname{Im}\left[f(x_0+ih)\right]}{h}\right] \approx \frac{\operatorname{Im}\overbrace{\left[\frac{\operatorname{Im}\left[f(x_0+2ih)\right]}{h}\right]}^{\text{this term has no imaginary part}}}{h}$$

Since the term in the bracket has no imaginary part, we would simply get

$$\left.\frac{df}{dx}\right|_{x=x_0} = 0$$

which is incorrect.

One proposed extension of the complex-step approximation to second derivatives is

$$\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{2\left(f(x_0) - \operatorname{Re}\left[f(x_0+ih)\right]\right)}{h^2}$$

However, this second derivative approximation can also produce subtractive cancellation errors if the step size is made too small [16], which largely defeats the entire purpose of the complex-step approximation in the first place. Another approach is to use the **multicomplex-step method** which makes use of **multicomplex numbers**. However, the algebra of multicomplex numbers is *not* built into common programming languages [17], making the implementation of the multicomplex-step substantially more difficult. While the complex-step method requires some additional work (such as ensuring all functions are complexified), it is fairly easy to fix existing code to make it compatible with the complex-step approximation.

When forming the Hessian matrix in Section 7.6, we will evaluate second derivatives using an entirely different approach that utilizes a hybrid of complex-step and central difference approximations.

3.2 Choosing a Step Size

As noted in Cleve Moler's blog post [22] on the topic, the complex-step approximation of a derivative generally converges to within machine zero (ε, see Section 2.4) of the true derivative at a step size of about $h \approx \sqrt{\varepsilon}$ (due to the $\mathcal{O}(h^2)$ convergence). However, [18, p. 234] notes that a step size of 10^{-200} works well for double-precision functions. Therefore, for all complex-step approximations, the *Numerical Differentiation Toolbox* uses a step size of $h = 10^{-200}$.

3.3 Transposes

It is also vital to note how matrix transposes should be taken when using the complex-step approximation. The transpose operation in MATLAB is typically performed using an apostrophe (`'`), but this is problematic because it actually performs the conjugate transpose (i.e. it also takes the complex conjugate of each element). Therefore, we must use a dot before the apostrophe (`.'`) to perform the non-conjugate transpose [20].

> Vector and matrix transposes must be performed using the dot-apostrophe syntax (`.'`).

This also leads to issues with differentiating the standard `norm` and `dot` functions provided by MATLAB. Complexified version of the `norm` and `dot` functions are not included in [3], but we define them in Sections 3.4.5 and 3.4.11.

3.4 Complexification

The complex-step approximation can only be used to approximate the derivatives of real-valued functions with real-valued inputs. However, to use the complex-step approximation, we still need to be able to evaluate these functions using complex inputs. Additionally, these functions must be defined "correctly" for complex-valued inputs.

In popular programming languages such as MATLAB and Python, most common functions have already been written to be compatible with complex inputs. However, there are many functions that we must **complexify** to make them compatible with the complex-step approximation; these can even include functions that already accept complex-valued inputs. In the subsequent subsection, we include common complexified functions (in alphabetical order). Note that this is not an exhaustive list of complexified functions; see [3] for more complexified definitions.

3.4.1 `iabs`

The absolute value function is implemented as `abs` in most programming languages. The complexified version of this function, `iabs`, is defined by Algorithm 1 below [25].

Algorithm 1: `iabs`
Absolute value (complexified version of `abs`).

Inputs:

- $x \in \mathbb{C}$ - input argument

Procedure:

if $\mathrm{Re}\left[x\right] < 0$
  $y = -x$
else
  $y = x$
end
return y

Outputs:

- $y \in \mathbb{C}$ - absolute value of x

3.4.2 `iatan2`

The four-quadrant inverse tangent (in radians) is implemented as `atan2` in most programming languages. The complexified version of this function, `iatan2`, is defined by Algorithm 2 below [25].

Algorithm 2: `iatan2`
Four-quadrant inverse tangent in radians (complexified version of `atan2`).

Inputs:
- $y \in \mathbb{C}$ - input argument
- $x \in \mathbb{C}$ - input argument

Procedure:

$$a = \text{Re}\left[y\right]$$
$$b = \text{Im}\left[y\right]$$
$$c = \text{Re}\left[x\right]$$
$$d = \text{Im}\left[x\right]$$
$$z = \texttt{atan2}(a, c) + i\left(\frac{cb - ad}{a^2 + c^2}\right)$$

return z

Outputs:
- $z \in \mathbb{C}$ - four quadrant inverse tangent of (x, y) $[\text{rad}]$

3.4.3 `iatan2d`

The four-quadrant inverse tangent (in degrees) is implemented as `atan2d` in most programming languages. The complexified version of this function, `iatan2d`, is defined by Algorithm 3 below [25].

Algorithm 3: `iatan2d`
Four-quadrant inverse tangent in degrees (complexified version of `atan2d`).

Inputs:
- $y \in \mathbb{C}$ - input argument
- $x \in \mathbb{C}$ - input argument

Procedure:

$$a = \mathrm{Re}\,[y]$$
$$b = \mathrm{Im}\,[y]$$
$$c = \mathrm{Re}\,[x]$$
$$d = \mathrm{Im}\,[x]$$
$$z = \frac{180}{\pi}\left[\texttt{atan2}(a,c) + i\left(\frac{cb-ad}{a^2+c^2}\right)\right]$$

return z

Outputs:

- $z \in \mathbb{C}$ - four quadrant inverse tangent of (x, y) $[°]$

3.4.4 `iceil`

The ceiling function (round up to next integer towards positive infinity) is implemented as `ceil` in most programming languages. If the `ceil` function does not accept complex arguments, then its complexified version, `iceil`, should be used instead. `iceil` is defined by Algorithm 4 below.

Algorithm 4: `iceil`
Round towards positive infinity (complexified version of `ceil`).

Inputs:

- $x \in \mathbb{C}$ - input argument

Procedure:

$$y = \texttt{ceil}\,(\mathrm{Re}\,[x]) + (i)\texttt{ceil}\,(\mathrm{Im}\,[x])$$

return y

Outputs:

- $y \in \mathbb{C}$ - x rounded up to next integer towards positive infinity

3.4.5 `idot`

The dot product of $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$ is defined as

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T\mathbf{b}$$

When using the complex-step approximation, the vectors will be complex-valued (i.e. $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$). Nonetheless, we still want the dot product to be computed in the same way as for real-valued vectors.

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T\mathbf{y} \tag{3.2}$$

MATLAB's `dot` function already accepts complex-valued inputs, but instead of using the standard (non-conjugate) transpose, it uses the Hermitian (conjugate) transpose:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^H\mathbf{y} \tag{3.3}$$

For real-valued $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{x}^H\mathbf{y} = \mathbf{x}^T\mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

However, for complex-valued $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{x}^H\mathbf{y} \neq \mathbf{x}^T\mathbf{y}, \quad \mathbf{x},\mathbf{y} \in \mathbb{C}^n$$

Thus, we need to redefine the dot product function to use Eq. (3.2) instead of Eq. (3.3). The complexified version of the dot product function, `idot`, is defined by Algorithm 5 below.

Algorithm 5: `idot`
Vector dot product (complexified version of `dot`).

Inputs:
- $\mathbf{x} \in \mathbb{C}^n$ - input argument
- $\mathbf{y} \in \mathbb{C}^n$ - input argument

Procedure:

$z = \mathbf{x}^T\mathbf{y}$
return z

Outputs:
- $z \in \mathbb{C}$ - dot product of $\mathbf{x}$ and $\mathbf{y}$

3.4.6 `ifix`

The fix function (round to next integer towards zero) is implemented as `fix` in most programming languages. If the `fix` function does not accept complex arguments, then its complexified version, `ifix`, should be used instead. `ifix` is defined by Algorithm 6 below.

Algorithm 6: `ifix`
Round towards zero (complexified version of `fix`).

Inputs:
- $x \in \mathbb{C}$ - input argument

Procedure:

$y = \texttt{fix}\left(\text{Re}\left[x\right]\right) + (i)\texttt{fix}\left(\text{Im}\left[x\right]\right)$
return y

Outputs:
- $y \in \mathbb{C}$ - x rounded to next integer towards zero

3.4.7 `ifloor`

The floor function (round down to next integer towards negative infinity) is implemented as `floor` in most programming languages. If the `floor` function does not accept complex arguments, then its complexified version, `ifloor`, should be used instead. `ifloor` is defined by Algorithm 7 below.

Algorithm 7: `ifloor`
Round towards negative infinity (complexified version of `floor`).

Inputs:
- $x \in \mathbb{C}$ - input argument

Procedure:

$y = \texttt{floor}\,(\mathrm{Re}\,[x]) + (i)\texttt{floor}\,(\mathrm{Im}\,[x])$
return y

Outputs:
- $y \in \mathbb{C}$ - x rounded down to next integer towards negative infinity

3.4.8 `imax`

The function that returns the maximum of two numbers is implemented as `max` in most programming languages. The complexified version of this function, `imax`, is defined by Algorithm 8 below [3].

Algorithm 8: `imax`
Maximum of two numbers (complexified version of `max`).

Inputs:
- $x \in \mathbb{C}$ - input argument
- $y \in \mathbb{C}$ - input argument

Procedure:

```
if Re[x] < Re[y]
  |   m = y
else
  |   m = x
end
```

Outputs:
- $m \in \mathbb{C}$ - maximum of x and y

3.4.9 `imin`

The function that returns the minimum of two numbers is implemented as `min` in most programming languages. The complexified version of this function, `imin`, is defined by Algorithm 9 below [3].

Algorithm 9: `imin`
Minimum of two numbers (complexified version of `min`).

Inputs:
- $x \in \mathbb{C}$ - input argument
- $y \in \mathbb{C}$ - input argument

Procedure:

if $\mathrm{Re}[x] > \mathrm{Re}[y]$
  $m = y$
else
  $m = x$
end

Outputs:
- $m \in \mathbb{C}$ - minimum of x and y

3.4.10 `imod`

The modulo operation, $\mathrm{mod}\,(a, n)$, returns the remainder, r, of the division a/n, where r has the same sign as the divisor, n. While there are multiple definitions of this operation, many programming languages (such as MATLAB, Python, and Julia) use the "floored" definition

$$\mathrm{mod}\,(a, n) = a - \left\lfloor \frac{a}{n} \right\rfloor n$$

where $\lfloor \cdot \rfloor$ represents the floor function [21]. Since the `mod` functions in MATLAB and Python don't accept a complex divisor, we introduce Algorithm 10 to define the complexified version of this function, `imod`.

Algorithm 10: `imod`
Remainder after division with divisor's sign (complexified version of `mod`).

Inputs:
- $a \in \mathbb{C}$ - dividend
- $n \in \mathbb{R}$ - divisor

Procedure:

$$r = a - \texttt{floor}\left(\frac{a}{n}\right) n$$

Outputs:
- $r \in \mathbb{C}$ - remainder of a/n (with divisor's sign)

Note:
- The `floor` function in some programming languages may not accept a complex input. If this is the case, use Algorithm 7 (`ifloor`) instead.

3.4.11 `inorm`

For $\mathbf{x} \in \mathbb{R}^n$, 2-norm is defined as

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}} \tag{3.4}$$

However, MATLAB's `norm` uses

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^H \mathbf{x}} \tag{3.5}$$

For more detail on why this causes issues, see Section 3.4.5. Essentially, we need to redefine the 2-norm function to use Eq. (3.4) instead of Eq. (3.5). The complexified version of the 2-norm function, `inorm`, is defined in Algorithm 11 below.

Algorithm 11: `inorm`
2-norm of a vector (complexified version of `norm`).

Inputs:
- $\mathbf{x} \in \mathbb{C}^n$ - input argument

Procedure:

$y = \sqrt{\mathbf{x}^T \mathbf{x}}$
return y

Outputs:
- $y \in \mathbb{C}$ - 2-norm of $\mathbf{x}$

3.4.12 `irem`

The remainder function, $\mathrm{rem}(a, n)$, returns the remainder, r, of the division a/n, where r has the same sign as the dividend, a. While there are multiple definitions of this operation, many programming languages (such as MATLAB, Python, and Julia) use the "truncated" definition

$$\mathrm{mod}\,(a, n) = a - \mathrm{fix}\left(\frac{a}{n}\right) n$$

where $\mathrm{fix}(\cdot)$ represents the fix function which rounds a number towards 0 [21]. Since the `rem` functions in most programming languages don't accept a complex divisor, we introduce Algorithm 12 to define the complexified version of this function, `irem`.

Algorithm 12: `irem`
Remainder after division with dividend's sign (complexified version of `rem`).

Inputs:
- $a \in \mathbb{C}$ - dividend
- $n \in \mathbb{R}$ - divisor

Procedure:

$r = a - \texttt{fix}\left(\frac{a}{n}\right) n$

Outputs:

- $r \in \mathbb{C}$ - remainder of a/n (with dividend's sign)

Note:

- The `fix` function in some programming languages may not accept a complex input. If this is the case, use Algorithm 6 (`ifix`) instead.

4

Generalizing the Approximations to Higher Dimensions

4.1 Derivatives of Univariate, Vector-Valued Functions

Recall from Section 2.1 that the Taylor series expansion for a univariate, scalar-valued function $f:\mathbb{R}\to\mathbb{R}$ is given by

$$f(x+h)=f(x)+hf'(x)+\frac{h^2}{2}f''(x)+\frac{h^3}{6}f'''(x)+\cdots$$

Now, consider the case where we have a univariate, vector-valued function $\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$. The Taylor series expansion of $\mathbf{f}$ is

$$\mathbf{f}(x+h)=\mathbf{f}(x)+h\mathbf{f}'(x)+\frac{h^2}{2}\mathbf{f}''(x)+\frac{h^3}{6}\mathbf{f}'''(x)+\cdots$$

Thus, the forward, backward, and central difference approximations (Sections 2.1–2.3), as well as the complex-step approximation (Section 3.1) retain the exact same form, except we replace the scalar-valued f with the vector-valued $\mathbf{f}$.

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\underbrace{\frac{\mathbf{f}(x_0+h)-\mathbf{f}(x_0)}{h}}_{\text{forward difference}}}\tag{4.1}$$

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\underbrace{\frac{\mathbf{f}(x_0)-\mathbf{f}(x_0-h)}{h}}_{\text{backward difference}}}\tag{4.2}$$

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\underbrace{\frac{\mathbf{f}(x_0+h)-\mathbf{f}(x_0-h)}{2h}}_{\text{central difference}}}\tag{4.3}$$

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\underbrace{\frac{\operatorname{Im}\left[\mathbf{f}(x_0+ih)\right]}{h}}_{\text{complex-step}}}\tag{4.4}$$

4.2 Partial Derivatives of Multivariate, Scalar-Valued Functions

Recall from Section 2.1 that the Taylor series expansion for a univariate, scalar-valued function $f : \mathbb{R} \to \mathbb{R}$ is given by

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots$$

Now, consider the case where we have a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. *If we only allow the kth component of its independent variable to vary (i.e. x_k)*, then its Taylor series expansion in the kth direction is

$$f(\mathbf{x} + h\mathbf{e}_k) = f(\mathbf{x}) + h\frac{\partial f}{\partial x_k} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x_k^2} + \frac{h^3}{6}\frac{\partial^3 f}{\partial x_k^3} + \cdots$$

Using an identical procedure to the one in Section 2.1, we can find the forward difference approximation of the *partial* derivative to be

$$\boxed{\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{f(\mathbf{x}_0 + h\mathbf{e}_k) - f(\mathbf{x}_0)}{h}}_{\text{forward difference}}} \tag{4.5}$$

Similarly, the backward difference, central difference, and complex-step approximations are [18, pp. 227–228, 233]

$$\boxed{\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{f(\mathbf{x}_0) - f(\mathbf{x}_0 - h\mathbf{e}_k)}{h}}_{\text{backward difference}}} \tag{4.6}$$

$$\boxed{\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{f(\mathbf{x}_0 + h\mathbf{e}_k) - f(\mathbf{x}_0 - h\mathbf{e}_k)}{2h}}_{\text{central difference}}} \tag{4.7}$$

$$\boxed{\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{\operatorname{Im}\left[f(\mathbf{x}_0 + ih\mathbf{e}_k)\right]}{h}}_{\text{complex-step}}} \tag{4.8}$$

4.3 Partial Derivatives of Multivariate, Vector-Valued Functions

In the most general case, we can consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. By combining the analyses from Sections 4.1 and 4.2, we can generalize the approximations to find the partial derivatives of vector-valued functions.

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{\mathbf{f}(\mathbf{x}_0 + h\mathbf{e}_k) - \mathbf{f}(\mathbf{x}_0)}{h}}_{\text{forward difference}}} \tag{4.9}$$

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{\mathbf{f}(\mathbf{x}_0) - \mathbf{f}(\mathbf{x}_0 - h\mathbf{e}_k)}{h}}_{\text{backward difference}}} \tag{4.10}$$

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{\mathbf{f}(\mathbf{x}_0 + h\mathbf{e}_k) - \mathbf{f}(\mathbf{x}_0 - h\mathbf{e}_k)}{2h}}_{\text{central difference}}} \tag{4.11}$$

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \underbrace{\frac{\operatorname{Im}\left[\mathbf{f}(\mathbf{x}_0 + ih\mathbf{e}_k)\right]}{h}}_{\text{complex-step}}} \tag{4.12}$$

4.4 Summary

In this section, we summarize all the various approximations. Note that for the finite difference approximations, we replace the relative step size, h, with the absolute step size, Δx, which we can recall[1] is defined as

$$\boxed{\Delta x = h(1 + |x_0|)} \tag{4.13}$$

when evaluating a derivative about the point $x_0 \in \mathbb{R}$. In the multivariate case, when evaluating a derivative about the point $\mathbf{x}_0 = (x_{0,1}, ..., x_{0,n}) \in \mathbb{R}^n$, we use

$$\boxed{\Delta x_k = h(1 + |x_{0,k}|)} \tag{4.14}$$

Derivatives of univariate, scalar-valued functions.

Consider a univariate, scalar-valued function $f : \mathbb{R} \to \mathbb{R}$. Approximations for the derivative of f with respect to $x \in \mathbb{R}$, evaluated at the point $x = x_0$, are given by Eq. (4.15) below.

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_0} \approx \begin{cases} \dfrac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}, & \text{forward difference} \\[2ex] \dfrac{f(x_0) - f(x_0 - \Delta x)}{\Delta x}, & \text{backward difference} \\[2ex] \dfrac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x}, & \text{central difference} \\[2ex] \dfrac{\operatorname{Im}\left[f(x_0 + ih)\right]}{h}, & \text{complex-step} \end{cases}} \tag{4.15}$$

Derivatives of univariate, vector-valued functions.

Consider a univariate, vector-valued function $\mathbf{f} : \mathbb{R} \to \mathbb{R}^m$. Approximations for the derivative of $\mathbf{f}$ with respect to

[1] See Section 2.4.

$x \in \mathbb{R}$, evaluated at the point $x = x_0$, are given by Eq. (4.16) below.

$$\boxed{\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} \approx \begin{cases} \dfrac{\mathbf{f}(x_0+\Delta x)-\mathbf{f}(x_0)}{\Delta x}, & \text{forward difference} \\[2ex] \dfrac{\mathbf{f}(x_0)-\mathbf{f}(x_0-\Delta x)}{\Delta x}, & \text{backward difference} \\[2ex] \dfrac{\mathbf{f}(x_0+\Delta x)-\mathbf{f}(x_0-\Delta x)}{2\Delta x}, & \text{central difference} \\[2ex] \dfrac{\operatorname{Im}\left[\mathbf{f}(x_0+ih)\right]}{h}, & \text{complex-step} \end{cases}} \tag{4.16}$$

Partial derivatives of multivariate, scalar-valued functions.

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Approximations for the partial derivative of f with respect to $x_k \in \mathbb{R}$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, are given by Eq. (4.17) below.

$$\boxed{\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \begin{cases} \dfrac{f(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-f(\mathbf{x}_0)}{\Delta x_k}, & \text{forward difference} \\[2ex] \dfrac{f(\mathbf{x}_0)-f(\mathbf{x}_0-\mathbf{e}_k\Delta x_k)}{\Delta x_k}, & \text{backward difference} \\[2ex] \dfrac{f(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-f(\mathbf{x}_0-\mathbf{e}_k\Delta x_k)}{2\Delta x_k}, & \text{central difference} \\[2ex] \dfrac{\operatorname{Im}\left[f(\mathbf{x}_0+ih\mathbf{e}_k)\right]}{h}, & \text{complex-step} \end{cases}} \tag{4.17}$$

Partial derivatives of multivariate, vector-valued functions.

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Approximations for the partial derivative of $\mathbf{f}$ with respect to $x_k \in \mathbb{R}$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, are given by Eq. (4.18) below.

$$\boxed{\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \begin{cases} \dfrac{\mathbf{f}(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-\mathbf{f}(\mathbf{x}_0)}{\Delta x_k}, & \text{forward difference} \\[2ex] \dfrac{\mathbf{f}(\mathbf{x}_0)-\mathbf{f}(\mathbf{x}_0-\mathbf{e}_k\Delta x_k)}{\Delta x_k}, & \text{backward difference} \\[2ex] \dfrac{\mathbf{f}(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-\mathbf{f}(\mathbf{x}_0-\mathbf{e}_k\Delta x_k)}{2\Delta x_k}, & \text{central difference} \\[2ex] \dfrac{\operatorname{Im}\left[\mathbf{f}(\mathbf{x}_0+ih\mathbf{e}_k)\right]}{h}, & \text{complex-step} \end{cases}} \tag{4.18}$$

PART II

Implementation

5

Numerical Differentiation Using the Forward Difference Approximation

5.1 Derivatives

Consider a univariate, vector-valued function $\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$. Recall[1] that the forward difference approximation of the derivative of $\mathbf{f}$ with respect to $x\in\mathbb{R}$, evaluated at the point $x=x_0$, is defined as

$$\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\frac{\mathbf{f}(x_0+\Delta x)-\mathbf{f}(x_0)}{\Delta x}$$

where the absolute step size is defined as

$$\Delta x=h(1+|x_0|)$$

Algorithm 13: `fderivative`

Derivative of a univariate, vector-valued function using the forward difference approximation.

Inputs:

- $\mathbf{f}(x)$ - univariate, vector-valued function ($\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$)
- $x_0\in\mathbb{R}$ - evaluation point
- $h\in\mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\sqrt{\varepsilon}$)

Procedure:

1. Default the relative step size to $h=\sqrt{\varepsilon}$ if not input.
2. Absolute step size.

$$\Delta x=h(1+|x_0|)$$

[1] See Sections 2.1 and 4.4.

3. Evaluate the derivative.

$$\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} = \frac{\mathbf{f}(x_0+\Delta x)-\mathbf{f}(x_0)}{\Delta x}$$

4. Return result.

$$\textbf{return } \left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}$$

Outputs:

- $\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} \in \mathbb{R}^m$ - derivative of $\mathbf{f}$ with respect to x, evaluated at $x = x_0$

Note:

- This algorithm requires 2 evaluations of $\mathbf{f}(x)$.

5.2 Partial Derivatives

Consider a multivariate, vector-valued function $\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^m$. Recall[2] that the forward difference approximation of the partial derivative of $\mathbf{f}$ with respect to $x_k \in \mathbb{R}$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{\mathbf{f}(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-\mathbf{f}(\mathbf{x}_0)}{\Delta x_k}$$

where the absolute step size is defined as

$$\Delta x_k = h(1+|x_{0,k}|)$$

Algorithm 14: `fpartial`

Partial derivative of a multivariate, vector-valued function using the forward difference approximation.

Inputs:

- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $k \in \mathbb{Z}$ - element of $\mathbf{x}$ to differentiate with respect to
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\sqrt{\varepsilon}$)

Procedure:

1. Default the relative step size to $h=\sqrt{\varepsilon}$ if not input.
2. Evaluate and store the value of $\mathbf{f}(\mathbf{x}_0)$.

$$\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0)$$

3. Absolute step size.

$$\Delta x_k = h(1+|x_{0,k}|)$$

[2] See Sections 2.1 and 4.4.

4. Step in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

5. Evaluate the partial derivative of $\mathbf{f}(\mathbf{x})$ with respect to x_k.

$$\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} = \frac{\mathbf{f}(\mathbf{x}_0) - \mathbf{f}_0}{\Delta x_k}$$

6. Return result.

$$\textbf{return } \left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}$$

Outputs:

- $\left.\dfrac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \in \mathbb{R}^m$ - partial derivative of $\mathbf{f}$ with respect to x_k, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires 2 evaluations of $\mathbf{f}(\mathbf{x})$.

5.3 Gradients

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[3] that the gradient of f with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{g}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) = \begin{bmatrix} \left.\dfrac{\partial f}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots \\ \left.\dfrac{\partial f}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}$$

The procedure for evaluating the individual partial derivatives in the equation above is detailed in Section 5.2. Algorithm 15 below is adapted from [18, p. 232].

Algorithm 15: `fgradient`

Gradient of a multivariate, scalar-valued function using the forward difference approximation.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f : \mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\sqrt{\varepsilon}$)

Procedure:

[3] See Section 1.5.

1. Default the relative step size to $h = \sqrt{\varepsilon}$ if not input.
2. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
3. Preallocate the vector $\mathbf{g} \in \mathbb{R}^n$ to store the gradient.
4. Evaluate and store the value of $f(\mathbf{x}_0)$.

$$f_0 = f(\mathbf{x}_0)$$

5. Evaluate the gradient.

for $k = 1$ **to** n

(a) Absolute step size.

$$\Delta x_k = h(1 + |x_{0,k}|)$$

(b) Step in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

(c) Partial derivative of f with respect to x_k.

$$g_k = \frac{f(\mathbf{x}_0) - f_0}{\Delta x_k}$$

(d) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} - \Delta x_k$$

end

6. Return result.

return g

Outputs:

- $\mathbf{g} = \mathbf{g}(\mathbf{x}_0) \in \mathbb{R}^n$ - gradient of f with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $n + 1$ evaluations of $f(\mathbf{x})$.

5.4 Directional Derivatives

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[4] that the directional derivative of f with respect to $\mathbf{x} \in \mathbb{R}^n$ in the direction of $\mathbf{v} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, can be defined as

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \left.\frac{dg}{d\alpha}\right|_{\alpha=0} \tag{5.1}$$

where

$$g(\alpha) = f(\mathbf{x}_0 + \alpha\mathbf{v}) \tag{5.2}$$

[4] See Section 1.6.

From the definition[5] of the forward difference approximation, we can write

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} \approx \frac{g(\Delta\alpha)-g(0)}{\Delta\alpha}$$

where the absolute step size[6] is

$$\Delta\alpha = h(1+|0|) = h$$

Thus, we have

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} \approx \frac{g(h)-g(0)}{h} \tag{5.3}$$

Substituting Eq. (5.3) into Eq. (5.1),

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) \approx \frac{g(h)-g(0)}{h} \tag{5.4}$$

From Eq. (5.2), we can write

$$g(h) = f(\mathbf{x}_0 + h\mathbf{v})$$
$$g(0) = f(\mathbf{x}_0)$$

Substituting these into Eq. (5.4),

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) \approx \frac{f(\mathbf{x}_0 + h\mathbf{v}) - f(\mathbf{x}_0)}{h}$$

Algorithm 16: `fdirectional`

Directional derivative of a multivariate, scalar-valued function using the forward difference approximation.

Inputs:
- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f : \mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $\mathbf{v} \in \mathbb{R}^n$ - vector defining direction of differentiation
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\sqrt{\varepsilon}$)

Procedure:
1. Default the relative step size to $h = \sqrt{\varepsilon}$ if not input.
2. Evaluate the directional derivative.

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \frac{f(\mathbf{x}_0 + h\mathbf{v}) - f(\mathbf{x}_0)}{h}$$

3. Return result.

return $\nabla_{\mathbf{v}} f(\mathbf{x}_0)$

Outputs:
- $\nabla_{\mathbf{v}} f(\mathbf{x}_0) \in \mathbb{R}$ - directional derivative of f with respect to $\mathbf{x}$ in the direction of $\mathbf{v}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:
- This algorithm requires 2 evaluations of $f(\mathbf{x})$.
- This implementation does *not* assume that $\mathbf{v}$ is a unit vector.

[5] See Eq. (4.15) in Section 4.4.
[6] See Section 2.4.

5.5 Jacobians

Consider a multivariate, vector-valued function $\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$. Recall[7] that the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}\in\mathbb{R}^n$, evaluated at the point $\mathbf{x}=\mathbf{x}_0$, is defined as

$$\mathbf{J}(\mathbf{x}_0)=\left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}=\left[\left.\frac{\partial\mathbf{f}}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0}\quad\cdots\quad\left.\frac{\partial\mathbf{f}}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0}\right]$$

The procedure for evaluating the individual partial derivatives in the equation above is detailed in Section 5.2.

Algorithm 17: `fjacobian`

Jacobian of a multivariate, vector-valued function using the forward difference approximation.

Inputs:

- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$)
- $\mathbf{x}_0\in\mathbb{R}^n$ - evaluation point
- $h\in\mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\sqrt{\varepsilon}$)

Procedure:

1. Default the relative step size to $h=\sqrt{\varepsilon}$ if not input.
2. Evaluate and store the value of $\mathbf{f}(\mathbf{x}_0)$.

   $$\mathbf{f}_0=\mathbf{f}(\mathbf{x}_0)$$

3. Determine m given that $\mathbf{f}_0\in\mathbb{R}^m$.
4. Determine n given that $\mathbf{x}_0\in\mathbb{R}^n$.
5. Preallocate the matrix $\mathbf{J}\in\mathbb{R}^{m\times n}$ to store the Jacobian.
6. Evaluate the Jacobian.

   for $k=1$ **to** n

   (a) Absolute step size.

   $$\Delta x_k=h(1+|x_{0,k}|)$$

   (b) Step in the kth direction.

   $$x_{0,k}=x_{0,k}+\Delta x_k$$

   (c) Partial derivative of $\mathbf{f}$ with respect to x_k.

   $$\mathbf{J}_{:,k}=\frac{\mathbf{f}(\mathbf{x}_0)-\mathbf{f}_0}{\Delta x_k}$$

   (d) Reset $\mathbf{x}_0$.

   $$x_{0,k}=x_{0,k}-\Delta x_k$$

   end

7. Return result.

   return J

[7] See Section 1.7.

Outputs:

- $\mathbf{J} = \mathbf{J}(\mathbf{x}_0) \in \mathbb{R}^{m\times n}$ - Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $n+1$ evaluations of $\mathbf{f}(\mathbf{x})$.

5.6 Hessians

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[8] that the (j, k)th element of the Hessian of f with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial^2 f}{\partial x_j \partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}$$

Let's rewrite this equation in a slightly different form.

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_0} \left(\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}\right)$$

Recall[9] that the forward difference approximation for the partial derivative of f is

$$\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k) - f(\mathbf{x}_0)}{\Delta x_k}$$

Replacing the derivative in the parentheses with its forward difference approximation, we can write

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} \approx \left.\frac{\partial}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_0} \left[\frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k) - f(\mathbf{x}_0)}{\Delta x_k}\right]$$

Applying the forward difference approximation once more,

$$\begin{aligned}
[\mathbf{H}(\mathbf{x}_0)]_{j,k} &\approx \frac{1}{\Delta x_j}\left[\frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k + \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 + \mathbf{e}_j \Delta x_j)}{\Delta x_k}\right] - \frac{1}{\Delta x_j}\left[\frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k) - f(\mathbf{x}_0)}{\Delta x_k}\right] \\
&\approx \frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k + \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 + \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k) + f(\mathbf{x}_0)}{\Delta x_j \Delta x_k}
\end{aligned}$$

With these expression for the (j, k)th element of the Hessian, we can develop Algorithm 18 below. Recall that since the Hessian matrix is symmetric, we only have to evaluate the derivatives in the upper triangle of the matrix (see Section 1.8). Through experimentation, we find that a relative step size of $h = \varepsilon^{1/3}$ is suitable.

[8] See Section 1.8

[9] See Sections 2.1 and 4.4.

Algorithm 18: `fhessian`

Hessian of a multivariate, scalar-valued function using the forward difference approximation.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f:\mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
3. Preallocate the matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ to store the Hessian.
4. Evaluate and store the value of $f(\mathbf{x}_0)$.

$$f_0 = f(\mathbf{x}_0)$$

5. Preallocate the vector $\mathbf{a} \in \mathbb{R}^n$ to store the absolute step size for each direction k.
6. Preallocate the vector $\mathbf{b} \in \mathbb{R}^n$ to store the evaluations of f with steps in each direction k.
7. Populate $\mathbf{a}$ and $\mathbf{b}$.

for $k = 1$ **to** n

(a) Absolute step size.

$$\Delta x_k = h(1 + |x_{0,k}|)$$

(b) Step in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

(c) Function evaluation.

$$b_k = f(\mathbf{x}_0)$$

(d) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} - \Delta x_k$$

(e) Store Δx_k in $\mathbf{a}$.

$$a_k = \Delta x_k$$

end

8. Evaluate the Hessian, looping through the upper triangular elements.

for $k = 1$ **to** n

for $j = k$ **to** n

(a) Step in the jth and kth directions.

$$x_{0,j} = x_{0,j} + a_j$$
$$x_{0,k} = x_{0,k} + a_k$$

(b) Evaluate the (j,k)th element of the Hessian.

$$H_{j,k} = \frac{f(\mathbf{x}_0) - b_j - b_k + f_0}{a_j a_k}$$

(c) Evaluate the $(k,j)^{\text{th}}$ element of the Hessian using symmetry.

$$H_{k,j} = H_{j,k}$$

(d) Reset $\mathbf{x}_0$.

$$x_{0,j} = x_{0,j} - a_j$$
$$x_{0,k} = x_{0,k} - a_k$$

end

end

9. Return result.

return H

Outputs:

- $\mathbf{H} = \mathbf{H}(\mathbf{x}_0) \in \mathbb{R}^{n \times n}$ - Hessian of f with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $\frac{n(n+1)}{2} + 1$ evaluations of $f(\mathbf{x})$ (the upper triangular matrix entries (including the diagonal) consist of $n(n+1)/2$ entries [11], each entry requires 1 evaluation of $f(\mathbf{x})$, and $f(\mathbf{x})$ is evaluated a single time at the beginning of the algorithm).

5.7 Vector Hessians

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Recall[10] that the vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{H}(\mathbf{f}(\mathbf{x}_0)) = (\mathbf{H}(f_1(\mathbf{x}_0)), \dots, \mathbf{H}(f_m(\mathbf{x}_0)))$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

[10] See Section 1.9

Algorithm 19: `fvechessian`

Vector Hessian of a multivariate, vector-valued function using the forward difference approximation.

Inputs:

- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Determine m given that $\mathbf{f}(\mathbf{x}_0) \in \mathbb{R}^m$.
3. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
4. Preallocate the array $\mathbf{H} \in \mathbb{R}^{n\times n\times m}$ to store the Hessian.
5. Evaluate the vector Hessian.

for $k = 1$ **to** m

(a) Define a function for the kth component of $\mathbf{f}(\mathbf{x})$.

$$f_k(\mathbf{x}) = \texttt{helper}(\mathbf{f}, \mathbf{x}, k)$$

(b) Evaluate the kth Hessian (Algorithm 18).

$$\mathbf{H}_{:,:,k} = \texttt{fhessian}(f_k, \mathbf{x}_0, h)$$

end

6. Return result.

return H

Outputs:

- $\mathbf{H} = \mathbf{H}(\mathbf{f}(\mathbf{x}_0)) \in \mathbb{R}^{n\times n\times m}$ - vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $m\left[\frac{n(n+1)}{2} + 1\right] + 1$ evaluations of $f(\mathbf{x})$.

6

Numerical Differentiation Using the Central Difference Approximation

6.1 Derivatives

Consider a univariate, vector-valued function $\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$. Recall[1] that the central difference approximation of the derivative of $\mathbf{f}$ with respect to $x\in\mathbb{R}$, evaluated at the point $x=x_0$, is defined as

$$\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\frac{\mathbf{f}(x_0+\Delta x)-\mathbf{f}(x_0-\Delta x)}{2\Delta x}$$

where the absolute step size is defined as

$$\Delta x=h(1+|x_0|)$$

Algorithm 20: `cderivative`

Derivative of a univariate, vector-valued function using the central difference approximation.

Inputs:

- $\mathbf{f}(x)$ - univariate, vector-valued function ($\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$)
- $x_0\in\mathbb{R}$ - evaluation point
- $h\in\mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the relative step size to $h=\varepsilon^{1/3}$ if not input.
2. Absolute step size.

$$\Delta x=h(1+|x_0|)$$

[1] See Sections 2.3 and 4.4.

3. Evaluate the derivative.

$$\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} = \frac{\mathbf{f}(x_0+\Delta x)-\mathbf{f}(x_0-\Delta x)}{2\Delta x}$$

4. Return result.

$$\textbf{return } \left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}$$

Outputs:

- $\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} \in \mathbb{R}^m$ - derivative of $\mathbf{f}$ with respect to x, evaluated at $x = x_0$

Note:

- This algorithm requires 2 evaluations of $\mathbf{f}(x)$.

6.2 Partial Derivatives

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Recall[2] that the central difference approximation of the partial derivative of $\mathbf{f}$ with respect to $x_k \in \mathbb{R}$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\left.\frac{\partial\mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{\mathbf{f}(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-\mathbf{f}(\mathbf{x}_0-\mathbf{e}_k\Delta x_k)}{2\Delta x_k}$$

where the absolute step size is defined as

$$\Delta x_k = h(1+|x_{0,k}|)$$

Algorithm 21: `cpartial`

Partial derivative of a multivariate, vector-valued function using the central difference approximation.

Inputs:

- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $k \in \mathbb{Z}$ - element of $\mathbf{x}$ to differentiate with respect to
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Absolute step size.

$$\Delta x_k = h(1+|x_{0,k}|)$$

3. Step forward in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$
$$\mathbf{f}_1 = \mathbf{f}(\mathbf{x}_0)$$

[2] See Sections 2.3 and 4.4.

4. Step backward in the kth direction.

$$x_{0,k} = x_{0,k} - 2\Delta x_k$$
$$\mathbf{f}_2 = \mathbf{f}(\mathbf{x}_0)$$

5. Evaluate the partial derivative of $\mathbf{f}(\mathbf{x})$ with respect to x_k.

$$\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} = \frac{\mathbf{f}_1 - \mathbf{f}_2}{2\Delta x_k}$$

6. Return result.

$$\textbf{return } \left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}$$

Outputs:

- $\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \in \mathbb{R}^m$ - partial derivative of $\mathbf{f}$ with respect to x_k, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires 2 evaluations of $\mathbf{f}(\mathbf{x})$.

6.3 Gradients

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[3] that the gradient of f with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{g}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) = \begin{bmatrix} \left.\frac{\partial f}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots \\ \left.\frac{\partial f}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}$$

The procedure for evaluating the individual partial derivatives in the equation above is detailed in Section 6.2.

Algorithm 22: `cgradient`

Gradient of a multivariate, scalar-valued function using the central difference approximation.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f : \mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:

[3] See Section 1.5.

1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
3. Preallocate the vector $\mathbf{g} \in \mathbb{R}^n$ to store the gradient.
4. Evaluate the gradient.

for $k = 1$ **to** n

(a) Absolute step size.

$$\Delta x_k = h(1 + |x_{0,k}|)$$

(b) Step forward in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$
$$f_1 = f(\mathbf{x}_0)$$

(c) Step backward in the kth direction.

$$x_{0,k} = x_{0,k} - 2\Delta x_k$$
$$f_2 = f(\mathbf{x}_0)$$

(d) Partial derivative of f with respect to x_k.

$$g_k = \frac{f_1 - f_2}{2\Delta x_k}$$

(e) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

end

5. Return result.

return g

Outputs:

- $\mathbf{g} = \mathbf{g}(\mathbf{x}_0) \in \mathbb{R}^n$ - gradient of f with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $2n$ evaluations of $f(\mathbf{x})$.

6.4 Directional Derivatives

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[4] that the directional derivative of f with respect to $\mathbf{x} \in \mathbb{R}^n$ in the direction of $\mathbf{v} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, can be defined as

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \left.\frac{dg}{d\alpha}\right|_{\alpha=0} \tag{6.1}$$

[4] See Section 1.6.

where

$$g(\alpha) = f(\mathbf{x}_0 + \alpha\mathbf{v}) \tag{6.2}$$

From the definition[5] of the central difference approximation, we can write

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} \approx \frac{g(\Delta\alpha) - g(-\Delta\alpha)}{2\Delta\alpha}$$

where the absolute step size[6] is

$$\Delta\alpha = h(1 + |0|) = h$$

Thus, we have

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} \approx \frac{g(h) - g(-h)}{2h} \tag{6.3}$$

Substituting Eq. (6.3) into Eq. (6.1),

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) \approx \frac{g(h) - g(-h)}{2h} \tag{6.4}$$

From Eq. (6.2), we can write

$$g(h) = f(\mathbf{x}_0 + h\mathbf{v})$$
$$g(-h) = f(\mathbf{x}_0 - h\mathbf{v})$$

Substituting these into Eq. (6.4),

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) \approx \frac{f(\mathbf{x}_0 + h\mathbf{v}) - f(\mathbf{x}_0 - h\mathbf{v})}{2h}$$

Algorithm 23: `cdirectional`

Directional derivative of a multivariate, scalar-valued function using the central difference approximation.

Inputs:
- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f : \mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $\mathbf{v} \in \mathbb{R}^n$ - vector defining direction of differentiation
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:
1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Evaluate the directional derivative.

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \frac{f(\mathbf{x}_0 + h\mathbf{v}) - f(\mathbf{x}_0 - h\mathbf{v})}{2h}$$

3. Return result.

return $\nabla_{\mathbf{v}} f(\mathbf{x}_0)$

Outputs:
- $\nabla_{\mathbf{v}} f(\mathbf{x}_0) \in \mathbb{R}$ - directional derivative of f with respect to $\mathbf{x}$ in the direction of $\mathbf{v}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:
- This algorithm requires 2 evaluations of $f(\mathbf{x})$.
- This implementation does *not* assume that $\mathbf{v}$ is a unit vector.

[5] See Eq. (4.15) in Section 4.4.
[6] See Section 2.4.

6.5 Jacobians

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Recall[7] that the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{J}(\mathbf{x}_0) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \left[\left.\frac{\partial \mathbf{f}}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} \quad \cdots \quad \left.\frac{\partial \mathbf{f}}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0}\right]$$

The procedure for evaluating the individual partial derivatives in the equation above is detailed in Section 6.2.

Algorithm 24: `cjacobian`

Jacobian of a multivariate, vector-valued function using the central difference approximation.

Inputs:
- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:
1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Determine m given that $\mathbf{f}(\mathbf{x}_0) \in \mathbb{R}^m$.
3. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
4. Preallocate the matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$ to store the Jacobian.
5. Evaluate the Jacobian.

for $k = 1$ **to** n

(a) Absolute step size.

$$\Delta x_k = h(1 + |x_{0,k}|)$$

(b) Step forward in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$
$$\mathbf{f}_1 = \mathbf{f}(\mathbf{x}_0)$$

(c) Step backward in the kth direction.

$$x_{0,k} = x_{0,k} - 2\Delta x_k$$
$$\mathbf{f}_2 = \mathbf{f}(\mathbf{x}_0)$$

(d) Partial derivative of $\mathbf{f}$ with respect to x_k.

$$\mathbf{J}_{:,k} = \frac{\mathbf{f}_1 - \mathbf{f}_2}{2\Delta x_k}$$

(e) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

end

[7] See Section 1.7.

6. Return result.

return J

Outputs:

- $\mathbf{J} = \mathbf{J}(\mathbf{x}_0) \in \mathbb{R}^{m \times n}$ - Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $2n + 1$ evaluations of $\mathbf{f}(\mathbf{x})$.

6.6 Hessians

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[8] that the (j, k)th element of the Hessian of f with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial^2 f}{\partial x_j \partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}$$

Let's rewrite this equation in a slightly different form.

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_0} \left(\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \right)$$

Recall[9] that the central difference approximation for the partial derivative of f is

$$\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k) - f(\mathbf{x}_0 - \mathbf{e}_k \Delta x_k)}{2\Delta x_k}$$

Replacing the derivative in the parentheses with its central difference approximation, we can write

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} \approx \left.\frac{\partial}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_0} \left[\frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k) - f(\mathbf{x}_0 - \mathbf{e}_k \Delta x_k)}{2\Delta x_k} \right]$$

Applying the central difference approximation once more,

$$\begin{aligned}[\mathbf{H}(\mathbf{x}_0)]_{j,k} \approx \frac{1}{2\Delta x_j} &\left[\frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k + \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 - \mathbf{e}_k \Delta x_k + \mathbf{e}_j \Delta x_j)}{2\Delta x_k} \right] \\ &- \frac{1}{2\Delta x_j} \left[\frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k - \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 - \mathbf{e}_k \Delta x_k - \mathbf{e}_j \Delta x_j)}{2\Delta x_k} \right]\end{aligned}$$

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} \approx \frac{f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k + \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 - \mathbf{e}_k \Delta x_k + \mathbf{e}_j \Delta x_j) - f(\mathbf{x}_0 + \mathbf{e}_k \Delta x_k - \mathbf{e}_j \Delta x_j) + f(\mathbf{x}_0 - \mathbf{e}_k \Delta x_k - \mathbf{e}_j \Delta x_j)}{4\Delta x_j \Delta x_k}$$

With these expression for the (j, k)th element of the Hessian, we can develop Algorithm 25 below[10]. Recall that since the Hessian matrix is symmetric, we only have to evaluate the derivatives in the upper triangle of the matrix (see Section 1.8). Through experimentation, we find that a relative step size of $h = \varepsilon^{1/3}$ is suitable.

[8] See Section 1.8.
[9] See Sections 2.3 and 4.4
[10] This algorithm was inspired by [2] and [5].

Algorithm 25: `chessian`

Hessian of a multivariate, scalar-valued function using the central difference approximation.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f:\mathbb{R}^n\to\mathbb{R}$)
- $\mathbf{x}_0\in\mathbb{R}^n$ - evaluation point
- $h\in\mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the relative step size to $h=\varepsilon^{1/3}$ if not input.
2. Determine n given that $\mathbf{x}_0\in\mathbb{R}^n$.
3. Preallocate the matrix $\mathbf{H}\in\mathbb{R}^{n\times n}$ to store the Hessian.
4. Preallocate the vector $\mathbf{a}\in\mathbb{R}^n$ to store the absolute step size for each direction k.
5. Populate $\mathbf{a}$.

   for $k=1$ **to** n

   $\quad a_k=h(1+|x_{0,k}|)$

   end

6. Evaluate the Hessian, looping through the upper triangular elements.

   for $k=1$ **to** n

for $j = k$ **to** n

(a) Step forward in the jth and kth directions.

$$x_{0,j} = x_{0,j} + a_j$$
$$x_{0,k} = x_{0,k} + a_k$$
$$b = f(\mathbf{x}_0)$$
$$x_{0,j} = x_{0,j} - a_j$$
$$x_{0,k} = x_{0,k} - a_k$$

(b) Step forward in the jth direction and backward in the kth direction.

$$x_{0,j} = x_{0,j} + a_j$$
$$x_{0,k} = x_{0,k} - a_k$$
$$c = f(\mathbf{x}_0)$$
$$x_{0,j} = x_{0,j} - a_j$$
$$x_{0,k} = x_{0,k} + a_k$$

(c) Step backward in the jth direction and forward in the kth direction.

$$x_{0,j} = x_{0,j} - a_j$$
$$x_{0,k} = x_{0,k} + a_k$$
$$d = f(\mathbf{x}_0)$$
$$x_{0,j} = x_{0,j} + a_j$$
$$x_{0,k} = x_{0,k} - a_k$$

(d) Step backward in the jth and kth directions.

$$x_{0,j} = x_{0,j} - a_j$$
$$x_{0,k} = x_{0,k} - a_k$$
$$e = f(\mathbf{x}_0)$$
$$x_{0,j} = x_{0,j} + a_j$$
$$x_{0,k} = x_{0,k} + a_k$$

(e) Evaluate the (j, k)th element of the Hessian.

$$H_{j,k} = \frac{b - c - d + e}{4a_j a_k}$$

(f) Evaluate the $(k, j)^{\text{th}}$ element of the Hessian using symmetry.

$$H_{k,j} = H_{j,k}$$

end

end

7. Return result.

return H

Outputs:
- $\mathbf{H} = \mathbf{H}(\mathbf{x}_0) \in \mathbb{R}^{n \times n}$ - Hessian of f with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:
- This algorithm requires $2n(n+1)$ evaluations of $f(\mathbf{x})$ (the upper triangular matrix entries (including the diagonal) consist of $n(n+1)/2$ entries [11], and each entry requires 4 evaluations of $f(\mathbf{x})$).

6.7 Vector Hessians

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Recall[11] that the vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{H}(\mathbf{f}(\mathbf{x}_0)) = (\mathbf{H}(f_1(\mathbf{x}_0)), \dots, \mathbf{H}(f_m(\mathbf{x}_0)))$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

Algorithm 26: `cvechessian`

Vector Hessian of a multivariate, vector-valued function using the central difference approximation.

Inputs:
- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to $\varepsilon^{1/3}$)

Procedure:
1. Default the relative step size to $h = \varepsilon^{1/3}$ if not input.
2. Determine m given that $\mathbf{f}(\mathbf{x}_0) \in \mathbb{R}^m$.
3. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
4. Preallocate the array $\mathbf{H} \in \mathbb{R}^{n \times n \times m}$ to store the Hessian.
5. Evaluate the vector Hessian.

for $k = 1$ **to** m

(a) Define a function for the kth component of $\mathbf{f}(\mathbf{x})$.

$$f_k(\mathbf{x}) = \texttt{helper}(\mathbf{f}, \mathbf{x}, k)$$

(b) Evaluate the kth Hessian (Algorithm 25).

$$\mathbf{H}_{:,:,k} = \texttt{chessian}(f_k, \mathbf{x}_0, h)$$

end

[11] See Section 1.9

6. Return result.

 return H

Outputs:

- $\mathbf{H} = \mathbf{H}(\mathbf{f}(\mathbf{x}_0)) \in \mathbb{R}^{n \times n \times m}$ - vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $2mn(n+1)+1$ evaluations of $\mathbf{f}(\mathbf{x})$.

7

Numerical Differentiation Using the Complex-Step Approximation

7.1 Derivatives

Consider a univariate, vector-valued function $\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$. Recall[1] that the complex-step approximation of the derivative of $\mathbf{f}$ with respect to $x\in\mathbb{R}$, evaluated at the point $x=x_0$, is defined as

$$\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}\approx\frac{\mathrm{Im}\left[\mathbf{f}(x_0+ih)\right]}{h}$$

Algorithm 27: `iderivative`

Derivative of a univariate, vector-valued function using the complex-step approximation.

Inputs:

- $\mathbf{f}(x)$ - univariate, vector-valued function ($\mathbf{f}:\mathbb{R}\to\mathbb{R}^m$)
- $x_0\in\mathbb{R}$ - evaluation point
- $h\in\mathbb{R}$ - *(OPTIONAL)* step size (defaults to 10^{-200})

Procedure:

1. Default the step size to $h=10^{-200}$ if not input.
2. Evaluate the derivative.

$$\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}=\frac{\mathrm{Im}\left[\mathbf{f}(x_0+ih)\right]}{h}$$

3. Return result.

$$\textbf{return}\ \left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0}$$

[1] See Sections 3.1 and 4.4.

Outputs:

- $\left.\frac{d\mathbf{f}}{dx}\right|_{x=x_0} \in \mathbb{R}^m$ - derivative of $\mathbf{f}$ with respect to x, evaluated at $x = x_0$

Note:

- This algorithm requires 1 evaluation of $\mathbf{f}(x)$.

7.2 Partial Derivatives

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Recall[2] that the complex-step approximation of the partial derivative of $\mathbf{f}$ with respect to $x_k \in \mathbb{R}$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{\operatorname{Im}\left[\mathbf{f}(\mathbf{x}_0 + ih\mathbf{e}_k)\right]}{h}$$

Algorithm 28: `ipartial`

Partial derivative of a multivariate, vector-valued function using the complex-step approximation.

Inputs:

- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $k \in \mathbb{Z}$ - element of $\mathbf{x}$ to differentiate with respect to
- $h \in \mathbb{R}$ - *(OPTIONAL)* step size (defaults to 10^{-200})

Procedure:

1. Default the step size to $h = 10^{-200}$ if not input.
2. Step in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

3. Evaluate the partial derivative of $\mathbf{f}(\mathbf{x})$ with respect to x_k.

$$\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} = \frac{\operatorname{Im}\left[\mathbf{f}(\mathbf{x}_0)\right]}{h}$$

4. Return result.

$$\textbf{return } \left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}$$

Outputs:

- $\left.\frac{\partial \mathbf{f}}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \in \mathbb{R}^m$ - partial derivative of $\mathbf{f}$ with respect to x_k, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires 1 evaluation of $\mathbf{f}(\mathbf{x})$.

[2] See Sections 3.1 and 4.4.

7.3 Gradients

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[3] that the gradient of f with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{g}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) = \begin{bmatrix} \left.\dfrac{\partial f}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} \\ \vdots \\ \left.\dfrac{\partial f}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}$$

The procedure for evaluating the individual partial derivatives in the equation above is detailed in Section 7.2. Algorithm 29 below is adapted from [18, p. 235].

Algorithm 29: `igradient`

Gradient of a multivariate, scalar-valued function using the complex-step approximation.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f : \mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* step size (defaults to 10^{-200})

Procedure:

1. Default the step size to $h = 10^{-200}$ if not input.
2. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
3. Preallocate the vector $\mathbf{g} \in \mathbb{R}^n$ to store the gradient.
4. Evaluate the gradient.

for $k = 1$ **to** n

(a) Step in the kth direction.

$$x_{0,k} = x_{0,k} + \Delta x_k$$

(b) Partial derivative of f with respect to x_k.

$$g_k = \frac{\operatorname{Im}\left[f(\mathbf{x}_0)\right]}{h}$$

(c) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} - \Delta x_k$$

end

5. Return result.

return g

Outputs:

[3] See Section 1.5.

- $\mathbf{g} = \mathbf{g}(\mathbf{x}_0) \in \mathbb{R}^n$ - gradient of f with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires n evaluations of $f(\mathbf{x})$.

7.4 Directional Derivatives

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[4] that the directional derivative of f with respect to $\mathbf{x} \in \mathbb{R}^n$ in the direction of $\mathbf{v} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, can be defined as

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \left.\frac{dg}{d\alpha}\right|_{\alpha=0} \tag{7.1}$$

where

$$g(\alpha) = f(\mathbf{x}_0 + \alpha\mathbf{v}) \tag{7.2}$$

From the definition[5] of the complex-step approximation, we can write

$$\left.\frac{dg}{d\alpha}\right|_{\alpha=0} \approx \frac{\operatorname{Im}[g(ih)]}{h} \tag{7.3}$$

Substituting Eq. (7.3) into Eq. (7.1),

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) \approx \frac{\operatorname{Im}[g(ih)]}{h} \tag{7.4}$$

From Eq. (7.2), we can write

$$g(ih) = f(\mathbf{x}_0 + ih\mathbf{v})$$

Substituting this into Eq. (7.4),

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) \approx \frac{\operatorname{Im}[f(\mathbf{x}_0 + ih\mathbf{v})]}{h}$$

Algorithm 30: `idirectional`

Directional derivative of a multivariate, scalar-valued function using the complex-step approximation.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f : \mathbb{R}^n \to \mathbb{R}$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $\mathbf{v} \in \mathbb{R}^n$ - vector defining direction of differentiation
- $h \in \mathbb{R}$ - *(OPTIONAL)* relative step size (defaults to 10^{-200})

Procedure:

1. Default the relative step size to $h = 10^{-200}$ if not input.
2. Evaluate the directional derivative.

$$\nabla_{\mathbf{v}} f(\mathbf{x}_0) = \frac{\operatorname{Im}[f(\mathbf{x}_0 + ih\mathbf{v})]}{h}$$

[4] See Section 1.6.
[5] See Eq. (4.15) in Section 4.4.

3. Return result.

return $\nabla_{\mathbf{v}} f(\mathbf{x}_0)$

Outputs:
- $\nabla_{\mathbf{v}} f(\mathbf{x}_0) \in \mathbb{R}$ - directional derivative of f with respect to $\mathbf{x}$ in the direction of $\mathbf{v}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:
- This algorithm requires 1 evaluation of $f(\mathbf{x})$.
- This implementation does *not* assume that $\mathbf{v}$ is a unit vector.

7.5 Jacobians

Consider a multivariate, vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Recall[6] that the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$\mathbf{J}(\mathbf{x}_0) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} \left.\dfrac{\partial \mathbf{f}}{\partial x_1}\right|_{\mathbf{x}=\mathbf{x}_0} & \cdots & \left.\dfrac{\partial \mathbf{f}}{\partial x_n}\right|_{\mathbf{x}=\mathbf{x}_0} \end{bmatrix}$$

The procedure for evaluating the individual partial derivatives in the equation above is detailed in Section 7.2.

Algorithm 31: `ijacobian`
Jacobian of a multivariate, vector-valued function using the complex-step approximation.

Inputs:
- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$)
- $\mathbf{x}_0 \in \mathbb{R}^n$ - evaluation point
- $h \in \mathbb{R}$ - *(OPTIONAL)* step size (defaults to 10^{-200})

Procedure:
1. Default the step size to $h = 10^{-200}$ if not input.
2. Determine m given that $\mathbf{f}(\mathbf{x}_0) \in \mathbb{R}^m$.
3. Determine n given that $\mathbf{x}_0 \in \mathbb{R}^n$.
4. Preallocate the matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$ to store the Jacobian.
5. Evaluate the Jacobian.

for $k = 1$ **to** n

6 See Section 1.7.

(a) Step in the kth direction.

$$x_{0,k} = x_{0,k} + h$$

(b) Partial derivative of $\mathbf{f}$ with respect to x_k.

$$\mathbf{J}_{:,k} = \frac{\text{Im}\left[\mathbf{f}(\mathbf{x}_0)\right]}{h}$$

(c) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} - h$$

end

6. Return result.

return J

Outputs:

- $\mathbf{J} = \mathbf{J}(\mathbf{x}_0) \in \mathbb{R}^{m \times n}$ - Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $n+1$ evaluations of $\mathbf{f}(\mathbf{x})$.

7.6 Hessians

Consider a multivariate, scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$. Recall[7] that the (j,k)th element of the Hessian of f with respect to $\mathbf{x} \in \mathbb{R}^n$, evaluated at the point $\mathbf{x} = \mathbf{x}_0$, is defined as

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial^2 f}{\partial x_j \partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0}$$

Let's rewrite this equation in a slightly different form.

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} = \left.\frac{\partial}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_0} \left(\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \right) \tag{7.5}$$

As mentioned in Section 3.1.1, we will not be using a true complex-step approximation for the second derivative in this section. Instead, we will evaluate the first derivative using a complex-step approximation, and the second derivative using a central difference approximation. Recall[8] that the complex-step approximation for the partial derivative of f is

$$\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{\text{Im}\left[f(\mathbf{x}_0 + ih\mathbf{e}_k)\right]}{h}$$

Replacing the derivative in the parentheses in Eq. (7.5) with its complex-step approximation, we can write

$$[\mathbf{H}(\mathbf{x}_0)]_{j,k} \approx \left.\frac{\partial}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}_0} \left(\frac{\text{Im}\left[f(\mathbf{x}_0 + ih\mathbf{e}_k)\right]}{h} \right) \tag{7.6}$$

[7] See Section 1.8.
[8] See Sections 3.1 and 4.4

Next, recall[9] that the central difference approximation for the partial derivative of f is

$$\left.\frac{\partial f}{\partial x_k}\right|_{\mathbf{x}=\mathbf{x}_0} \approx \frac{f(\mathbf{x}_0+\mathbf{e}_k\Delta x_k)-f(\mathbf{x}_0-\mathbf{e}_k\Delta x_k)}{2\Delta x_k}$$

where

$$\Delta x_k = h_c(1+|x_{0,k}|)$$

is the absolute step size, and where h_c is the relative step size *for the central difference approximation*. Applying this approximation to Eq. (7.6),

$$\begin{aligned}[\mathbf{H}(\mathbf{x}_0)]_{j,k} &\approx \frac{1}{2\Delta x_j}\left[\left(\frac{\mathrm{Im}\left[f(\mathbf{x}_0+ih_i\mathbf{e}_k+\mathbf{e}_j\Delta x_j)\right]}{h_i}\right)-\left(\frac{\mathrm{Im}\left[f(\mathbf{x}_0+ih_i\mathbf{e}_k-\mathbf{e}_j\Delta x_j)\right]}{h_i}\right)\right]\\ &\approx \frac{\mathrm{Im}\left[f(\mathbf{x}_0+ih_i\mathbf{e}_k+\mathbf{e}_j\Delta x_j)\right]-\mathrm{Im}\left[f(\mathbf{x}_0+ih_i\mathbf{e}_k-\mathbf{e}_j\Delta x_j)\right]}{2h_i\Delta x_j}\end{aligned}$$

Algorithm 32: `ihessian`

Hessian of a multivariate, scalar-valued function using the complex-step and central difference approximations.

Inputs:

- $f(\mathbf{x})$ - multivariate, scalar-valued function ($f:\mathbb{R}^n\to\mathbb{R}$)
- $\mathbf{x}_0\in\mathbb{R}^n$ - evaluation point
- $h_i\in\mathbb{R}$ - *(OPTIONAL)* step size for the complex-step approximation (defaults to 10^{-200})
- $h_c\in\mathbb{R}$ - *(OPTIONAL)* relative step size for the central difference approximation (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the step size for the complex-step approximation to $h_i = 10^{-200}$ if not input.
2. Default the relative step size for the central difference approximation to $h_c=\varepsilon^{1/3}$ if not input.
3. Determine n given that $\mathbf{x}_0\in\mathbb{R}^n$.
4. Preallocate the matrix $\mathbf{H}\in\mathbb{R}^{n\times n}$ to store the Hessian.
5. Preallocate the vector $\mathbf{a}\in\mathbb{R}^n$ to store the absolute step size for each direction k.
6. Populate $\mathbf{a}$.

    for $k=1$ **to** n

    $\quad a_k = h_c(1+|x_{0,k}|)$

    end

7. Loop through columns.

    for $k=1$ **to** n

[9] See Sections 2.3 and 4.4

(a) Imaginary step forward in kth direction.

$$x_{0,k} = x_{0,k} + ih_i$$

(b) Loop through rows.

for $j = k$ **to** n

i. Real step forward in jth direction.

$$x_{0,j} = x_{0,j} + a_j$$
$$b = f(\mathbf{x}_0)$$

ii. Real step backward in jth direction.

$$x_{0,j} = x_{0,j} - 2a_j$$
$$c = f(\mathbf{x}_0)$$

iii. Reset $\mathbf{x}_0$.

$$x_{0,j} = x_{0,j} + a_j$$

iv. Evaluate the (j,k)th element of the Hessian.

$$H_{j,k} \approx \frac{\operatorname{Im}(b-c)}{2h_i a_j}$$

v. Evaluate the (k,j)th element of the Hessian using symmetry.

$$H_{k,j} = H_{j,k}$$

end

(c) Reset $\mathbf{x}_0$.

$$x_{0,k} = x_{0,k} - \Delta x_k$$

end

8. Return result.

return H

Outputs:

- $\mathbf{H} = \mathbf{H}(\mathbf{x}_0) \in \mathbb{R}^{n \times n}$ - Hessian of f with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $n(n+1)$ evaluations of $f(\mathbf{x})$ (the upper triangular matrix entries (including the diagonal) consist of $n(n+1)/2$ entries [11], and each entry requires 2 evaluations of $f(\mathbf{x})$).

7.7 Vector Hessians

Consider a multivariate, vector-valued function $\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$. Recall[10] that the vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x}\in\mathbb{R}^n$, evaluated at the point $\mathbf{x}=\mathbf{x}_0$, is defined as

$$\mathbf{H}(\mathbf{f}(\mathbf{x}_0))=(\mathbf{H}(f_1(\mathbf{x}_0)),\dots,\mathbf{H}(f_m(\mathbf{x}_0)))$$

where

$$\mathbf{f}(\mathbf{x})=\begin{bmatrix}f_1(\mathbf{x})\\ \vdots\\ f_m(\mathbf{x})\end{bmatrix}$$

Algorithm 33: `ivechessian`

Vector Hessian of a multivariate, vector-valued function using the complex-step and central difference approximations.

Inputs:

- $\mathbf{f}(\mathbf{x})$ - multivariate, vector-valued function ($\mathbf{f}:\mathbb{R}^n\to\mathbb{R}^m$)
- $\mathbf{x}_0\in\mathbb{R}^n$ - evaluation point
- $h_i\in\mathbb{R}$ - *(OPTIONAL)* step size for the complex-step approximation (defaults to 10^{-200})
- $h_c\in\mathbb{R}$ - *(OPTIONAL)* relative step size for the central difference approximation (defaults to $\varepsilon^{1/3}$)

Procedure:

1. Default the step size for the complex-step approximation to $h_i=10^{-200}$ if not input.
2. Default the relative step size for the central difference approximation to $h_c=\varepsilon^{1/3}$ if not input.
3. Determine m given that $\mathbf{f}(\mathbf{x}_0)\in\mathbb{R}^m$.
4. Determine n given that $\mathbf{x}_0\in\mathbb{R}^n$.
5. Preallocate the array $\mathbf{H}\in\mathbb{R}^{n\times n\times m}$ to store the Hessian.
6. Evaluate the vector Hessian.

for $k=1$ **to** m

(a) Define a function for the kth component of $\mathbf{f}(\mathbf{x})$.

$$f_k(\mathbf{x})=\texttt{helper}(\mathbf{f},\mathbf{x},k)$$

(b) Evaluate the kth Hessian (Algorithm 32).

$$\mathbf{H}_{:,:,k}=\texttt{ihessian}(f_k,\mathbf{x}_0,h_i,h_c)$$

end

7. Return result.

return H

Outputs:

[10] See Section 1.9

- $\mathbf{H} = \mathbf{H}(\mathbf{f}(\mathbf{x}_0)) \in \mathbb{R}^{n \times n \times m}$ - vector Hessian of $\mathbf{f}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_0$

Note:

- This algorithm requires $mn(n+1)+1$ evaluations of $\mathbf{f}(\mathbf{x})$.

PART III

Appendices

A

Differentiator *Objects*

Many different applications require the numerical differentiation of functions. However, with various numerical differentiation methods to choose from (central difference, forward difference, and complex-step), and ways to custom-tailor those methods (specifying relative or absolute step sizes), it would be convenient to have a way to store differentiation "settings". Essentially, we want all functions to know which differentiation method to use and with which step size, without having to pass the method and step size to the function and then go through logical checks within the function to determine which differentiation functions to use. An example to further motivate this idea is included below.

Consider a MATLAB function that takes two functions, $f(x)$ and $g(x)$, and returns the sum of their derivatives evaluated at x_0. Since we want to be able to specify the differentiation method as well as the step size, these need to be input as well.

```
function S = derivsum(f,g,x0,h,method)
    if strcmpi(method,'central difference')
        S = cderivative(f,x0,h)+cderivative(g,x0,h)
    elseif strcmpi(method,'forward difference')
        S = fderivative(f,x0,h)+fderivative(g,x0,h)
    elseif strcmpi(method,'complex-step')
        S = iderivative(f,x0,h)+iderivative(g,x0,h)
    end
end
```

Already, this is quite a cumbersome function to write, but it doesn't even take into account optional inputs (for example, we may want the function to default to a specific differentiation method and step size if we choose not to specify it).

Here is where `Differentiator` objects come in. First, in a script, we can define a default[1] `Differentiator` object.

```
d = Differentiator
```

The `derivsum` function from before can be rewritten to use a `Differentiator` object.

```
function S = derivsum(f,g,x0,d)
    S = d.derivative(f,x0)+d.derivative(g,x0)
end
```

[1] When the differentiation method isn't specified, the resulting `Differentiator` object will use the central difference approximation with a relative step size of $h = \varepsilon^{1/3}$ by default.

B

Test Cases

The analytical derivatives used for these test cases are compiled from [1, 8–10, 31].

> Some of the derivative approximations may be accurate to more than 16 decimal places; however, we only test up to 16 decimal places.

B.1 Derivative Test Cases

B.1.1 Polynomial and Square Root Functions

				Decimal Places of Precision		
$f(x)$	$f'(x)$	n	x_0	Forward Difference	Central Difference	Complex-Step
x^n	nx^{n-1}	0	2	16	16	16
x^n	nx^{n-1}	1	2	16	11	16
x^n	nx^{n-1}	2	2	7	11	16
x^n	nx^{n-1}	3	2	6	9	16
x^n	nx^{n-1}	7	2	4	6	13
x^n	nx^{n-1}	-1	2	8	10	16
x^n	nx^{n-1}	-2	2	7	10	16
x^n	nx^{n-1}	-3	2	7	9	16
x^n	nx^{n-1}	-7	2	8	10	16
x^n	nx^{n-1}	$1/3$	2	8	9	16
x^n	nx^{n-1}	$7/3$	2	7	10	16
x^n	nx^{n-1}	$-1/3$	2	8	11	16
x^n	nx^{n-1}	$-7/3$	2	7	9	16
$\sqrt{x}$	$\frac{1}{2\sqrt{x}}$	-	0.5	7	10	16

		1.0	16	10	16
		1.5	8	10	16

B.1.2 Exponential and Power Functions

			Decimal Places of Precision		
$f(x)$	$f'(x)$	x_0	Forward Difference	Central Difference	Complex-Step
e^x	e^x	-1	7	10	16
		0	16	10	16
		1	6	10	16
b^x	$b^x \ln b$	-1	7	10	16
		0	7	9	16
		1	6	8	16

B.1.3 Logarithmic Functions

			Decimal Places of Precision		
$f(x)$	$f'(x)$	x_0	Forward Difference	Central Difference	Complex-Step
$\ln x$	$\frac{1}{x}$	0.5	7	9	16
		1	7	10	16
		1.5	7	10	15
$\log_{10} x$	$\frac{1}{x \ln 10}$	0.5	6	9	16
		1	8	10	16
		1.5	8	10	15

B.1.4 Trigonometric Functions

			Decimal Places of Precision		
$f(x)$	$f'(x)$	x_0	Forward Difference	Central Difference	Complex-Step
$\sin x$	$\cos x$	0	16	10	16
		$\pi/4$	7	10	15
		$\pi/2$	7	15	16
		$3\pi/4$	7	9	16
		π	9	9	16
		$5\pi/4$	7	9	16
		$3\pi/2$	7	15	16
		$7\pi/4$	7	9	16
		2π	8	9	16
$\cos x$	$-\sin x$	0	7	16	16
		$\pi/4$	7	10	16
		$\pi/2$	9	10	16
		$3\pi/4$	7	9	15
		π	7	15	16
		$5\pi/4$	7	9	16

		$3\pi/2$	8	9	16
		$7\pi/4$	6	9	16
		2π	6	15	16
$\tan x$	$\sec^2 x; \quad x \neq \frac{\pi}{2} + n\pi$	0	16	10	16
		$\pi/4$	6	9	16
		$3\pi/4$	6	8	16
		π	9	9	16
		$5\pi/4$	6	8	16
		$7\pi/4$	6	8	16
		2π	8	8	16
$\csc x$	$-\csc x \tan x; \quad x \neq n\pi$	$\pi/4$	6	8	16
		$\pi/2$	7	15	14
		$3\pi/4$	6	8	16
		$5\pi/4$	5	8	15
		$3\pi/2$	7	15	16
		$7\pi/4$	6	7	16
$\sec x$	$\sec x \tan x; \quad x \neq \frac{\pi}{2} + n\pi$	0	7	16	16
		$\pi/4$	7	8	16
		$3\pi/4$	5	8	16
		π	7	15	16
		$5\pi/4$	6	8	16
		$7\pi/4$	5	7	16
		2π	6	15	16
$\cot x$	$-\csc^2 x; \quad x \neq n\pi$	$\pi/4$	6	9	16
		$\pi/2$	9	9	16
		$3\pi/4$	6	8	15
		$5\pi/4$	6	8	16
		$3\pi/2$	8	9	15
		$7\pi/4$	6	8	16

B.1.5 Inverse Trigonometric Functions

			Decimal Places of Precision		
$f(x)$	$f'(x)$	x_0	Forward Difference	Central Difference	Complex-Step
$\arcsin x$	$\frac{1}{\sqrt{1-x^2}}; \quad x \in (0,1)$	-0.5	7	10	16
		0	16	10	16
		0.5	7	10	16
$\arccos x$	$-\frac{1}{\sqrt{1-x^2}}; \quad x \in (0,1)$	-0.5	9	10	16
		0	16	10	16
		0.5	7	10	16
$\arctan x$	$\frac{1}{1+x^2}$	-1.5	8	10	16
		-1	7	10	16
		-0.5	7	11	15
		0	16	10	16
		0.5	7	11	15
		1	7	10	16
		1.5	7	10	16
$\operatorname{arccsc} x$	$-\frac{1}{\lvert x\rvert\sqrt{x^2-1}}; \quad \lvert x\rvert > 1$	-1.5	7	9	16

		1.5	7	9	16
$\operatorname{arcsec} x$	$\dfrac{1}{\lvert x\rvert\sqrt{x^2-1}}; \quad \lvert x\rvert > 1$	-1.5	7	9	16
		1.5	7	9	16
$\operatorname{arccot} x$	$-\dfrac{1}{1+x^2}$	-1.5	8	10	16
		-1	7	10	16
		-0.5	7	11	15
		0	16	N/A[1]	16
		0.5	7	11	15
		1	7	10	16
		1.5	7	10	16

B.1.6 Hyperbolic Functions

$f(x)$	$f'(x)$	x_0	Decimal Places of Precision Forward Difference	Central Difference	Complex-Step
$\sinh x$	$\cosh x$	-1	7	10	16
		0	16	10	16
		1	6	10	16
$\cosh x$	$\sinh x$	-1	7	9	16
		0	16	16	16
		1	7	9	16
$\tanh x$	$\operatorname{sech}^2 x$	-1	7	10	16
		0	16	10	16
		1	7	10	16
$\operatorname{csch} x$	$-\operatorname{csch} x \coth x, \quad x \neq 0$	-1	6	8	16
		1	7	8	16
$\operatorname{sech} x$	$-\operatorname{sech} x \tanh x$	-1	8	10	16
		0	16	16	16
		1	8	10	16
$\coth x$	$-\operatorname{csch}^2 x$	-1	7	10	16
		0	16	10	16
		1	7	10	16

B.1.7 Inverse Hyperbolic Functions

$f(x)$	$f'(x)$	x_0	Decimal Places of Precision Forward Difference	Central Difference	Complex-Step
$\operatorname{arsinh} x$	$\dfrac{1}{\sqrt{1+x^2}}$	-1.5	7	11	16
		-1	7	11	16
		-0.5	7	11	16
		0	16	10	16
		0.5	8	11	16

[1] Numerically unstable about $x = 0$.

		1	8	11	16
		1.5	7	11	16
$\operatorname{arcosh} x$	$-\frac{1}{\sqrt{x^2-1}}; \quad x > 1$	1.5	7	9	16
$\operatorname{artanh} x$	$\frac{1}{1-x^2}; \quad \lvert x \rvert < 1$	-0.5	7	9	15
		0	16	10	16
		0.5	6	9	15
$\operatorname{arcsch} x$	$-\frac{1}{\lvert x \rvert \sqrt{x^2+1}}; \quad x \neq 0$	-1.5	7	10	16
		-1	7	10	16
		-0.5	7	9	16
		0.5	6	9	16
		1	7	10	16
		1.5	7	10	16
$\operatorname{arsech} x$	$-\frac{1}{x\sqrt{1-x^2}}; \quad x \in (0,1)$	0.5	6	9	16
$\operatorname{arcoth} x$	$\frac{1}{1-x^2}; 7 \quad \lvert x \rvert > 1$	-1.5	7	9	15
		1.5		9	15

B.2 Partial Derivative Test Cases

$\mathbf{f}(\mathbf{x})$	k	$\left.\frac{\partial \mathbf{f}}{\partial x_k}\right\rvert_{\mathbf{x}=\mathbf{x}_0}$	$\mathbf{x}_0$	**Decimal Places of Precision** Forward Difference	Central Difference	Complex-Step
x^2	1	$2x$	2	7	11	16
$\begin{bmatrix} x^4 \\ x^3 \end{bmatrix}$	1	$\begin{bmatrix} 4x^3 \\ 3x^2 \end{bmatrix}$	2	5	8	16
$x_1^3 x_2^3$	2	$3x_1^3 x_2^2$	$\begin{bmatrix} 3 \\ 2 \end{bmatrix}$	4	7	16
$\begin{bmatrix} x_1^4 \\ x_2^3 \end{bmatrix}$	2	$\begin{bmatrix} 0 \\ 3x_2^2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	6	9	16
$\begin{bmatrix} x_1 \\ 5x_3 \\ 4x_2^2 - 2x_3 \\ x_3 \sin x_1 \end{bmatrix}$	3	$\begin{bmatrix} 0 \\ 5 \\ -2 \\ \sin x_1 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$	8	10	16

B.3 Gradient Test Cases

$f(\mathbf{x})$	$\nabla f(\mathbf{x})$	$\mathbf{x}_0$	**Decimal Places of Precision** Forward Difference	Central Difference	Complex-Step
x^2	$2x$	2	7	11	16
$x_1^2 + x_2^3$	$\begin{bmatrix} 2x_1 \\ 3x_2^2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	6	9	16

$x_1^5 + \sin^3 x_2$	$\begin{bmatrix} 5x_1^4 \\ 3\sin^2 x_2 \cos x_2 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 8 \end{bmatrix}$	3	6	11

B.4 Directional Derivative Test Cases

$f(\mathbf{x})$	$\nabla_{\mathbf{v}} f(\mathbf{x})$	$\mathbf{x}_0$	$\mathbf{v}$	Decimal Places of Precision: Forward Difference	Central Difference	Complex-Step
x^2	$2xv$	2	0.6	7	9	16
$x_1^2 + x_2^3$	$\begin{bmatrix} 2x_1 \\ 3x_2^2 \end{bmatrix}^T \mathbf{v}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$	5	8	16
$x_1^5 + \sin^3 x_2$	$\begin{bmatrix} 5x_1^4 \\ 3\sin^2 x_2 \cos x_2 \end{bmatrix}^T \mathbf{v}$	$\begin{bmatrix} 5 \\ 8 \end{bmatrix}$	$\begin{bmatrix} 10 \\ 20 \end{bmatrix}$	1	4	14

B.5 Jacobian Test Cases

$\mathbf{f}(\mathbf{x})$	$\mathbf{J}(\mathbf{x})$	$\mathbf{x}_0$	Decimal Places of Precision: Forward Difference	Central Difference	Complex-Step
x^2	$2x$	2	7	11	16
$\begin{bmatrix} x^2 \\ x^3 \end{bmatrix}$	$\begin{bmatrix} 2x \\ 3x^2 \end{bmatrix}$	2	6	9	16
$x_1^2 + x_2^3$	$\begin{bmatrix} 2x_1 & 3x_2^2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	6	9	16
$\begin{bmatrix} x_1^2 \\ x_2^3 \end{bmatrix}$	$\begin{bmatrix} 2x_1 & 0 \\ 0 & 3x_2^2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	6	9	16
$\begin{bmatrix} x_1 \\ 5x_3 \\ 4x_2^2 - 2x_3 \\ x_3 \sin x_1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \\ 0 & 8x_2 & -2 \\ x_3 \cos x_1 & 0 & \sin x_1 \end{bmatrix}$	$\begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$	5	9	16

B.6 Hessian Test Cases

$f(\mathbf{x})$	$\mathbf{H}(\mathbf{x})$	$\mathbf{x}_0$	Decimal Places of Precision: Forward Difference	Central Difference	Complex-Step
x^3	$6x$	2	3	6	10
$x_1^2 + x_2^3$	$\begin{bmatrix} 2 & 0 \\ 0 & 6x_2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	3	5	10
$x_1^5 x_2 + x_1 \sin^3 x_2$	$\mathbf{H}_1(\mathbf{x})$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	2	4	7

$$\mathbf{H}_1(\mathbf{x}) = \begin{bmatrix} 20x_1^3 x_2 & 5x_1^4 + 3\sin^2 x_2 \cos x_2 \\ 5x_1^4 + 3\sin^2 x_2 \cos x_2 & 6x_1 \sin x_2 \cos^2 x_2 - 3x_1 \sin^3 x_2 \end{bmatrix}$$

B.7 Vector Hessian Test Cases

$\mathbf{f}(\mathbf{x})$	$\mathbf{H}(\mathbf{x})$	$\mathbf{x}_0$	**Decimal Places of Precision** Forward Difference	Central Difference	Complex-Step
x^3	$6x$	2	3	6	10
$x_1^2 + x_2^3$	$\begin{bmatrix} 2 & 0 \\ 0 & 6x_2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	3	5	10
$x_1^5 x_2 + x_1 \sin^3 x_2$	$\mathbf{H}_1(\mathbf{x})$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	2	4	7
$\begin{bmatrix} x_1^5 x_2 + x_1 \sin^3 x_2 \\ x_1^3 + x_2^4 - 3x_1^2 x_2^2 \end{bmatrix}$	$(\mathbf{H}_1(\mathbf{x}), \mathbf{H}_2(\mathbf{x}))$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	2	4	7

$$\mathbf{H}_1(\mathbf{x}) = \begin{bmatrix} 20x_1^3 x_2 & 5x_1^4 + 3\sin^2 x_2 \cos x_2 \\ 5x_1^4 + 3\sin^2 x_2 \cos x_2 & 6x_1 \sin x_2 \cos^2 x_2 - 3x_1 \sin^3 x_2 \end{bmatrix}$$

$$\mathbf{H}_2(\mathbf{x}) = \begin{bmatrix} 6x_1^2 - 6x_2^2 & -12x_1 x_2 \\ -12x_1 x_2 & 12x_2^2 - 6x_1^2 \end{bmatrix}$$

B.8 Complexified Functions Test Cases

The MATLAB implementation of these differentiation routines [14] also includes test cases demonstrating situations when the complex-step approximation fails to differentiate the "standard" version of a function.

B.8.1 Derivatives of Univariate, Scalar-Valued Complexified Functions

$f(x)$	$f'(x)$	x_0	Decimal Places of Precision
`iabs`(x)	$\dfrac{x}{\lvert x\rvert}, \quad x \neq 0$	-1 1	16 16
`imax`(x, x^3)	*derivative of* $\max(f(x), g(x))$ *defined below*	-1.5 -0.5 0.5 1.5	16 16 16 16
`imin`(x, x^3)	*derivative of* $\min(f(x), g(x))$ *defined below*	-1.5 -0.5 0.5 1.5	16 16 16 16
`idot`$(\mathbf{f}(x), \mathbf{g}(x))$	$\left[\dfrac{d\mathbf{f}(x)}{dx}\right]^T \mathbf{g}(x) + [\mathbf{f}(x)]^T \dfrac{d\mathbf{g}(x)}{dx}$	2	16

$$\frac{d}{dx} \max(f(x), g(x)) = \begin{cases} f'(x), & f(x) > g(x) \\ g'(x), & f(x) < g(x) \end{cases}$$

$$\frac{d}{dx}\min\left(f(x),g(x)\right)=\begin{cases}f'(x), & f(x)<g(x)\\ g'(x), & f(x)>g(x)\end{cases}$$

$$\mathbf{f}(x)=\begin{bmatrix}x\\x^2\\x^3\end{bmatrix},\qquad \frac{d\mathbf{f}(x)}{dx}=\begin{bmatrix}1\\2x\\3x^2\end{bmatrix}$$

$$\mathbf{g}(x)=\begin{bmatrix}\sin x\\\cos x\\\tan x\end{bmatrix},\qquad \frac{d\mathbf{g}(x)}{dx}=\begin{bmatrix}\cos x\\-\sin x\\\sec^2 x\end{bmatrix}$$

B.8.2 Gradients of Multivariate, Scalar-Valued Complexified Functions

$f(\mathbf{x})$	$\nabla f(\mathbf{x})$	$\mathbf{x}_0$	Decimal Places of Precision
`iatan2`(x_2,x_1)	$\begin{bmatrix}\frac{-x_2}{\sqrt{x_1^2+x_2^2}}\\\frac{x_1}{\sqrt{x_1^2+x_2^2}}\end{bmatrix}$	$\left(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right)^T$	15
		$\left(\frac{-\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right)^T$	15
		$\left(\frac{-\sqrt{2}}{2},-\frac{\sqrt{2}}{2}\right)^T$	16
		$\left(\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2}\right)^T$	16
`iatan2d`(x_2,x_1)	$\frac{180}{\pi}\begin{bmatrix}\frac{-x_2}{\sqrt{x_1^2+x_2^2}}\\\frac{x_1}{\sqrt{x_1^2+x_2^2}}\end{bmatrix}$	$\left(\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right)^T$	13
		$\left(\frac{-\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right)^T$	13
		$\left(\frac{-\sqrt{2}}{2},-\frac{\sqrt{2}}{2}\right)^T$	13
		$\left(\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2}\right)^T$	13
`inorm`$(\mathbf{x})$	$\frac{\mathbf{x}}{\lVert\mathbf{x}\rVert}$	$(1,2,3)^T$	16

B.8.3 Partial Derivatives of Multivariate, Scalar-Valued Complexified Functions

$f(\mathbf{x})$	$\frac{\partial f(\mathbf{x})}{\partial x_k}$	$\mathbf{x}_0$	k	Decimal Places of Precision
`inorm`$(\mathbf{x})$	$\frac{x_k}{\lVert\mathbf{x}\rVert}$	$(1,2,3)^T$	2	16

B.8.4 Functionality of Complexified Rounding Functions

Function	Input(s)	Expected Output
`iceil`(x)	$x=1.1+1.1i$	$2+2i$
	$x=-1.1-1.1i$	$-1-i$
`ifloor`(x)	$x=1.1+1.1i$	$1+i$

	$x = -1.1 - 1.1i$	$-2 - 2i$
`ifix`(x)	$x = 1.1 + 1.1i$	$1 + i$
	$x = -1.1 - 1.1i$	$-1 - i$
`imod`(a, n)	$a = 10 + 10i, n = 3 + 3i$	$1 + i$
	$a = 10 + 10i, n = -3 - 3i$	$-2 - 2i$
	$a = 10 + 10i, n = 5 + 5i$	0
	$a = 10 + 10i, n = -5 - 5i$	0
`irem`(a, n)	$a = 10 + 10i, n = 3 + 3i$	1
	$a = 10 + 10i, n = -3 - 3i$	1
	$a = 10 + 10i, n = 5 + 5i$	0
	$a = 10 + 10i, n = -5 - 5i$	0

References

[1] *Absolute value*. Wikipedia. Accessed: November 11, 2022. URL: https://en.wikipedia.org/wiki/Absolute_value.

[2] Yi Cao. *Complex step Hessian*. MATLAB Central File Exchange. Accessed: August 3, 2021. URL: https://www.mathworks.com/matlabcentral/fileexchange/18177-complex-step-hessian?s_tid=srchtitle.

[3] *complexify.f90*. MDO Lab. Accessed: August 3, 2021. URL: https://mdolab.engin.umich.edu/misc/files/complexify.f90.

[4] *Gradient*. Wikipedia. Accessed: August 3, 2021. URL: https://en.wikipedia.org/wiki/Gradient.

[5] Daniel R. Herberg. `DTQP_hessian_complex_step` *from DT QP Project*. MATLAB Central File Exchange. Accessed: August 3, 2021. URL: https://www.mathworks.com/matlabcentral/fileexchange/65434-dt-qp-project?s_tid=srchtitle.

[6] *Hessian matrix*. Wikipedia. Accessed: August 3, 2021. URL: https://en.wikipedia.org/wiki/Hessian_matrix.

[7] *Hessian of a vector function?* GitHub. Accessed: August 30, 2022. URL: https://github.com/JuliaDiff/ForwardDiff.jl/issues/61.

[8] *Hyperbolic functions*. Wikipedia. Accessed: October 31, 2022. URL: https://en.wikipedia.org/wiki/Hyperbolic_functions.

[9] *Inverse hyperbolic functions*. Wikipedia. Accessed: October 31, 2022. URL: https://en.wikipedia.org/wiki/Inverse_hyperbolic_functions.

[10] *Inverse trigonometric functions*. Wikipedia. Accessed: October 31, 2022. URL: https://en.wikipedia.org/wiki/Inverse_trigonometric_functions.

[11] *Is there an analytic expression for a number of elements inside a triangular matrix (with and without items on diagonal)*. Stack Exchange. Accessed: December 26, 2021. URL: https://math.stackexchange.com/questions/2388887/is-there-an-analytic-expression-for-a-number-of-elements-inside-a-triangular-mat/2388889.

[12] *Jacobian matrix and determinant*. Wikipedia. Accessed: August 3, 2021. URL: https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant.

[13] *Jet (mathematics)*. Wikipedia. Accessed: August 30, 2022. URL: https://en.wikipedia.org/wiki/Jet_(mathematics).

[14] Tamas Kis. *Numerical Differentiation Toolbox*. 2021. URL: https://github.com/tamaskis/Numerical_Differentiation_Toolbox-MATLAB.

[15] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. Cambridge, MA: The MIT Press, 2019. URL: https://algorithmsbook.com/optimization/.

[16] Kok-Lam Lai and John L. Crassidis. “Generalizations of the Complex-Step Derivative Approximation”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit* (2006). DOI: 10.2514/6.2006-6348.

[17] Gregory Lantoine, Ryan P. Russell, and Thierry Dargent. "Using Multicomplex Variables for Automatic Computation of High-Order Derivatives". In: *ACM Transactions on Mathematical Software* 38.3 (Apr. 2012). DOI: 10.1145/2168773.2168774.

[18] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 2021. URL: https://mdobook.github.io/.

[19] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. "The Complex-Step Derivative Approximation". In: *ACM Transactions on Mathematical Software* 29.3 (Sept. 2003), pp. 245–262. DOI: 10.1145/838250.838251.

[20] *Matlab Implementation*. MDO Lab. Accessed: August 3, 2021. URL: https://mdolab.engin.umich.edu/misc/complex-step-guide-matlab.

[21] *Modulo operation*. Wikipedia. Accessed: November 6, 2022. URL: https://en.wikipedia.org/wiki/Modulo_operation.

[22] Cleve Moler. *Complex Step Differentiation*. MathWorks. Accessed: August 3, 2021. URL: https://blogs.mathworks.com/cleve/2013/10/14/complex-step-differentiation/.

[23] *Partial derivative*. Wikipedia. Accessed: April 13, 2022. URL: https://en.wikipedia.org/wiki/Partial_derivative.

[24] *Partial Derivatives of Vector Valued Functions*. Stack Exchange. Accessed: April 13, 2022. URL: https://math.stackexchange.com/questions/2823300/partial-derivatives-of-vector-valued-functions.

[25] *Python Implementation*. MDO Lab. Accessed: November 6, 2021. URL: https://mdolab.engin.umich.edu/misc/complex-step-guide-python.

[26] Dan Simon. *Optimal State Estimation*. Hoboken, NJ: John Wiley & Sons, 2006.

[27] William Squire and George Trapp. "Using Complex Variables to Estimate Derivatives of Real Functions". In: *SIAM Review* 40.1 (Mar. 1998), pp. 110–112. DOI: 10.1137/S003614459631241X.

[28] James Stewart. *Calculus*. 8th. Boston, MA: Cengage Learning, 2015.

[29] *Taylor expansion for vector-valued function?* Stack Exchange. Accessed: August 30, 2022. URL: https://math.stackexchange.com/questions/2648512/taylor-expansion-for-vector-valued-function.

[30] *Taylor's formula with remainder for vector-valued functions*. Stack Exchange. Accessed: August 30, 2022. URL: https://math.stackexchange.com/questions/2860582/taylors-formula-with-remainder-for-vector-valued-functions/2975683#2975683.

[31] *Trigonometric functions*. Wikipedia. Accessed: October 31, 2022. URL: https://en.wikipedia.org/wiki/Trigonometric_functions.

[32] *Vector-valued function*. Wikipedia. Accessed: April 13, 2022. URL: https://en.wikipedia.org/wiki/Vector-valued_function.