Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 19, No. 3, pp. 391–410, 2023

NUMERICAL STUDY OF APPLYING COMPLEX-STEP GRADIENT AND HESSIAN APPROXIMATIONS IN DERIVATIVE-FREE OPTIMIZATION

Warren Hare and Kashvi Srivastava

Abstract: We consider the problem of minimizing an objective function that is provided by an oracle. We assume that while the optimization problem seeks a real-valued solution, the oracle is capable of accepting complex-valued input and returning complex-valued output. We explore using complex-variables in order to approximate gradients and Hessians within a derivative-free optimization method. We provide several complex-variable based methods to construct approximate gradients and Hessians, then provide numerical error bounds for these methods. We apply the approximations in a Newton algorithm to numerically explore the pros and cons of each approximation technique.

Results find that complex-variable based methods improve the chances finding higher accuracy solutions and allow for smaller step sizes to be applied; but, require increased cpu time that outstrips any reduction in function calls used or iterations applied.

Key words: *derivative-free optimization, complex-variables, gradient approximation, Hessian approximation, Newton algorithm*

Mathematics Subject Classification: *90C56, 30E10; 65K10, 90C53*

1 Introduction

Consider the problem

$$\min\{f(x) : x \in \mathbb{R}^n\},$$

where $f : \mathbb{C}^n \to \mathbb{C}$ is provided by an oracle such that if the input is a real-valued, then the output will also be a real number. That is, given an input $z \in \mathbb{C}^n$, f returns a complex number $f(z)$, but the process of how it obtains that number is (possibly) unknown. And, given an input $x \in \mathbb{R}^n$, f returns a real number $f(x)$, but the process of how it obtains that number is (possibly) unknown.

Such processes are not uncommon in applied optimization. For example, any simulation that relies on matrix multiplication can fit into this framework provided the appropriate complex variable libraries are included [3, 8]. The popular mathematics programming software Matlab includes these libraries by default.

One approach to optimizing such a function is to use approximations of gradients and Hessians in a classical smooth optimization method [4, 9]. As a result, some researchers have concerned themselves with studying methods for constructing approximate gradients and Hessians. Classical methods, such as linear and quadratic interpolation, date back to at

least the 1970s [33], but were not well-established until the 1990s [5, 26]. More recent ideas include the development of *minimal Frobenius norm* models for under-determined quadratic models [6, 27, 28], non-polynomial models [18, 25, 30, 32], models for nonsmooth functions [9, 12, 14, 17], and calculus-based approaches to gradient approximation [10, 11, 29].

Our research explores the idea of complex-step modelling. We introduce this through a simple example first developed by Lyness and Moler [19].

For a complex-variable $z = z_{re} + z_{im}i$, we shall use $\Re(z) = z_{re}$ and $\Im(z) = z_{im}$ to denote the real and imaginary parts of z. Recall that a function is *analytic* at a fixed point, $\bar{z} \in \mathbb{C}^n$, if locally it agrees with its (infinite) Taylor expansion.

Consider an analytic single-variable function $f : \mathbb{C} \to \mathbb{C}$ at a real point $\bar{x} \in \mathbb{R}$. By Taylor's theorem, the classical divided difference formula can be derived as follows,

$$\begin{array}{rcl} f(\bar{x}+h) & = & f(\bar{x}) + hf'(\bar{x}) + O(h^2), \\ f(\bar{x}+h) - f(\bar{x}) & = & hf'(\bar{x}) + O(h^2), \\ \dfrac{f(\bar{x}+h) - f(\bar{x})}{h} & = & f'(\bar{x}) + O(h). \end{array}$$

This demonstrates that the divided difference formula, $(f(\bar{x}+h) - f(\bar{x}))/h$, gives an approximation of the true derivative with an error of order $O(h)$.

In [19], Lyness and Moler cleverly noted that using Taylor's expansion in the direction of the imaginary axis can provide an alternate formula. In particular, a Taylor expansion centered at $\bar{x}$ using a step of ih yields

$$\begin{array}{rcl} f(\bar{x}+ih) & = & f(\bar{x}) + ihf'(\bar{x}) + \frac{1}{2}(ih)^2 f''(\bar{x}) + O((ih)^3), \\ f(\bar{x}+ih) & = & f(\bar{x}) + ihf'(\bar{x}) - \frac{1}{2}(h^2) f''(\bar{x}) + iO(h^3), \\ \Im(f(\bar{x}+ih)) & = & hf'(\bar{x}) + O(h^3). \end{array}$$

This yields the complex-step derivative approximation

$$\frac{\Im(f(\bar{x}+ih))}{h} = f'(\bar{x}) + O(h^2).$$

Notice that, despite using only 1 function evaluation (instead of the 2 required in the divided difference formula), the error term is now of order $O(h^2)$.

In [15], it was noted that this idea could easily be adapted to approximate the gradient of an analytic multivariate complex-valued function. That is, if $f : \mathbb{C}^n \to \mathbb{C}$, then

$$\begin{bmatrix} \frac{\Im(f(x+ihe_1))}{h} \\ \frac{\Im(f(x+ihe_2))}{h} \\ \vdots \\ \frac{\Im(f(x+ihe_n))}{h} \end{bmatrix} = \nabla f(x) + O(h^2),$$

where e_i is the i^{th} coordinate vector.

Other approximations and approximating the Hessian for such a function is also possible; we discuss this in Section 2.

Once approximations of gradients and Hessians are established, these approximations can be applied in any classical smooth optimization method. The idea of using complex-step approximate gradients in such a framework has arisen in a number of applications, including, for example, Finance [2], Geophysics [1], Image Reconstruction [13], Sensitivity Analysis [20], and Uncertainty Analysis [31]. In these applications, the researchers argue that the complex-step approximation

(i) provides higher accuracy, so improves the chances of solving a given problem;

(ii) avoids *catastrophic cancellation* and allows smaller step sizes to be applied safely;

(iii) is reduced work as it uses fewer function evaluations to solve the problem.

The first of these arguments has been examined through mathematical theory [15], but not numerical testing with regards to optimization. The second of these arguments has been established through both theoretical methods and numerical testing [21, 15]. However, the value of the reduction in numerical error in optimization has not been fully studied. For example, none of the above citations examine the value of using complex-step Hessian approximations. The third of these arguments is clearly focused on gradient approximations, as we shall see in Table 1. The argument has not been explored with regards to using complex-step Hessian approximations. Moreover, the argument ignores the fact that evaluating functions at complex points often requires additional computation, so can be more time consuming.

In this research, our goal is to numerically explore the value of using complex-step approximate gradients and Hessians in optimization. We use four variations of complex-step approximations within an approximate Newton method. We compare their effectiveness against a classical approximation method and applying exact calculations within the same Newton method. Analysis of speed (time and function calls) and final solution accuracy is provided.

The remainder of this paper is organized as follows. In Section 2, we provide details on the four complex-step methods examined. We include each method's derivation and error analysis. Table 1 provides a summary of methods' properties. In Section 3, we briefly overview the approximate Newton method that was adopted for the experiments. We also provide details on the test problems and experimental set-up. In Section 4, we detail the results of the experiment. Performance profiles and data profiles are provided therein. Full details of the collected data can be obtained by contacting the corresponding author. In Section 5, we provide some concluding thoughts.

Remark: This paper makes heavy use of both real and complex numbers. To aid reading, throughout this paper we shall use $h \in \mathbb{R}$ and $x \in \mathbb{R}^n \subseteq \mathbb{C}^n$ when variables/constants are real, and use $z \in \mathbb{C}^n$ when variables/constants may be complex.

2 Quadratic Models

Consider a quadratic function

$$Q(z) = \alpha_0 + g^\top z + \frac{1}{2} z^\top H z,$$

where $\alpha_0 \in \mathbb{C}, g \in \mathbb{C}^n, H \in \mathbb{C}^{n\times n}$. Suppose we desire this quadratic to agree with an objective function $f : \mathbb{C}^n \to \mathbb{C}$ at a collection of points $\{z^0, z^1, \ldots, z^m\}$.

We begin by assuming, without loss of generality, that $H = H^\top$. From here, we can easily derive that α_0 has 2 unknown values (1 real and 1 imaginary part), g has $2n$ unknown values, and H has $2\dfrac{n(n+1)}{2} = n(n+1)$ unknown values. Thus, we need a total of $(n+1)(n+2)$ pieces of information in order to construct Q. Note that each equation $Q(z^i) = f(z^i)$ gives us 2 pieces of information: $\Re(Q(z^i)) = \Re(f(z^i))$ and $\Im(Q(z^i)) = \Im(f(z^i))$. Thus, it would

Catastrophic cancellation is created by a near-zero subtraction combined with a near-zero division.

appear that a minimum of $(n+1)(n+2)/2$ function evaluations are required to construct a well-poised model.

Suppose however that we have the additional information that $f(x) \in \mathbb{R}$ whenever $x \in \mathbb{R}^n$ and therefore we desire $Q(x) \in \mathbb{R}$ whenever $x \in \mathbb{R}^n$. Considering $Q(0) = \alpha_0 = f(0)$, we know $\alpha_0 \in \mathbb{R}$. Note that we do not actually evaluate $f(0)$ to harvest this information, we have only applied our assumption that $f(x) \in \mathbb{R}$ whenever $x \in \mathbb{R}^n$. Next, for each $i = 1, 2, \ldots, n$, consider $\frac{1}{2}(Q(e_i) - Q(-e_i)) = g_i$. Thus, our assumption implies $g \in \mathbb{R}^n$. Similarly, our assumption implies $H \in \mathbb{R}^{n \times n}$. In summary, if we desire $Q(x) \in \mathbb{R}$ whenever $x \in \mathbb{R}^n$, then we must have $\alpha_0 \in \mathbb{R}, g \in \mathbb{R}^n, H \in \mathbb{R}^{n \times n}$. This effectively provides us with $(n+1)(n+2)/2$ pieces of information without evaluating f at a single point.

At this point, if we limit ourselves to evaluating f at points $x^i \in \mathbb{R}^n$, then this will require $(n+1)(n+2)/2$ function evaluations. While each evaluation $f(x^i)$ still provides us with $\Re(Q(x^i)) = \Re(f(x^i))$ and $\Im(Q(x^i)) = \Im(f(x^i))$, the second equation reduces to $0 = 0$ and therefore gives no new information. We will label this as the real-step quadratic method (RQM) for our calculations. More details on the method are presented in Section 3.

However, if we consider $z^i \in \mathbb{C}^n$, then both the real and imaginary information will be valuable. This will allow us to construct gradient and Hessian approximations using less than $(n+1)(n+2)/2$ function evaluations.

2.1 Multivariate Taylor's theorem

Before developing complex-step quadratic methods, we recall Taylor's theorem for multivariate functions. To express the multivariate version of Taylor's theorem it is helpful to first review multi-index notation. A multi-index, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, is an n-tuple of non-negative integers. For $z \in \mathbb{C}^n$, multi-index notation defines the following operations

$$\begin{aligned} |\alpha| &= \alpha_1 + \alpha_2 + \cdots + \alpha_n, \\ \alpha! &= \alpha_1! \alpha_2! \ldots \alpha_n!, \\ z^\alpha &= z_1^{\alpha_1} z_2^{\alpha_2} \ldots z_n^{\alpha_n}, \\ \partial^\alpha f &= \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_n^{\alpha_n}} f. \end{aligned}$$

Applying multi-index notation, the multivariate version of Taylor's theorem is as follows. We remark that f being an analytic function tells us that locally it agrees with its infinite Taylor expansion, Theorem 2.1 allows us to quantify the error of using a finite Taylor expansion.

Theorem 2.1 (Multivariate Taylor's theorem)**.** *Let $f : \mathbb{C}^n \to \mathbb{C}$ be analytic at $\bar{z} \in \mathbb{C}^n$. Let $n \geq 1$ and define*

$$P_n(z) = f(\bar{z}) + \sum_{|\alpha|=1}^{|\alpha|=n} \frac{1}{\alpha!} \left(\partial^\alpha f\right)(\bar{z})(z - \bar{z})^\alpha. \tag{2.1}$$

Then, there exists a radius $\delta > 0$ such that if $\|z - \bar{z}\| < \delta$, then

$$P_n(z) - f(z) = \sum_{|\alpha| \geq n+1} \frac{1}{\alpha!} \left(\partial^\alpha f\right)(\bar{z})(z - \bar{z})^\alpha. \tag{2.2}$$

Consequently,

$$\begin{aligned} |P_n(z) - f(z)| &= O((z - \bar{z})^{n+1}), \\ |\Re(P_n(z) - f(z))| &= O((z - \bar{z})^{n+1}), \text{ and} \\ |\Im(P_n(z) - f(z))| &= O((z - \bar{z})^{n+1}). \end{aligned} \tag{2.3}$$

This result will be most useful when $z \in \mathbb{C}^n$ is such that $(z - \bar{z}) = h\exp(i\theta)e_j$ or $(z - \bar{z}) = h\exp(i\theta)(e_j + e_k)$ with $\theta \in [0, 2\pi)$.

Corollary 2.2. *Let $f : \mathbb{C}^n \to \mathbb{C}$ be analytic near $\bar{z} \in \mathbb{C}^n$. Let $n \geq 1$ and define P_n as in equation* (2.1)*.*

i. If $z = \bar{z} + h\exp(i\theta)e_j$, then

$$P_n(z) = f(\bar{z}) + \sum_{k=1}^{n} \exp(ik\theta)\frac{h^k}{k!}\frac{\partial^k f}{\partial x_j^k}(\bar{z}).$$

ii. If $z = \bar{z} + h\exp(i\theta)(e_{j_1} + e_{j_2})$, then

$$P_n(z) = f(\bar{z}) + \sum_{k=1}^{n} \exp(ik\theta)h^k \sum_{\substack{k_1+k_2=k \\ k_i \geq 0}} \frac{1}{(k_1!)(k_2!)}\frac{\partial^k f}{\partial x_{j_1}^{k_1}\partial x_{j_2}^{k_2}}(\bar{z}).$$

In either case, equation (2.3) *reduces to*

$$\begin{array}{rl} |P_n(z) - f(z)| & = O(h^{n+1}), \\ |\Re(P_n(z) - f(z))| & = O(h^{n+1}), \text{ and} \\ |\Im(P_n(z) - f(z))| & = O(h^{n+1}). \end{array} \tag{2.4}$$

2.2 Basic complex-step quadratic method (BCQM)

Suppose we desire to approximate $\nabla f(\bar{x})$ and $\nabla^2 f(\bar{x})$ for $\bar{x} \in \mathbb{R}^n$ and analytic function $f : \mathbb{C}^n \to \mathbb{C}$, where f is such that $f(x) \in \mathbb{R}$ whenever $x \in \mathbb{R}^n$.

Select a step size $h > 0$. By Corollary 2.2(i) at $\theta = \pi/2$, the 3^{rd}-order complex-step Taylor expansion of f at $\bar{x}$ satisfies

$$\begin{array}{rcl} f(\bar{x} + (ih)e_j) & = & P_3(x) + O(h^4) \\ & = & f(\bar{x}) + \sum_{k=1}^{3} i^k \frac{h^k}{k!}\frac{\partial^k f}{\partial x_j^k}(\bar{x}) + O(h^4). \end{array}$$

This provides

$$\Re(f(\bar{x} + (ih)e_j)) = f(\bar{x}) - \frac{h^2}{2}e_j^\top \nabla^2 f(\bar{x})e_j + O(h^4).$$

Similarly, the 2^{nd}-order complex-step Taylor expansion of f at $\bar{x}$ provides

$$\Im(f(\bar{x} + (ih)e_j)) = h\nabla f(\bar{x})^\top e_j + O(h^3).$$

This immediately yields

$$\nabla f(\bar{x})^\top = \left[\frac{\Im(f(\bar{x}+ihe_1))}{h} \quad \frac{\Im(f(\bar{x}+ihe_2))}{h} \quad \cdots \quad \frac{\Im(f(\bar{x}+ihe_n))}{h}\right] + O(h^2). \tag{2.5}$$

Focusing on the real part of the Taylor expansion, we find that

$$\frac{\partial^2 f}{\partial x_j^2}(\bar{x}) = \frac{2(f(\bar{x}) - \Re(f(\bar{x} + (ih)e_j)))}{h^2} + O(h^2). \tag{2.6}$$

To find the remaining terms of the Hessian, consider Corollary 2.2(ii) at $\theta = \pi/2$ to see

$$\begin{aligned} & \Re(f(\bar{x} + ih(e_j + e_k))) \\ = \; & f(\bar{x}) - \tfrac{1}{2}h^2(e_j + e_k)^\top \nabla^2 f(\bar{x})(e_j + e_k) + O(h^4) \\ = \; & f(\bar{x}) - \tfrac{1}{2}h^2 \left(\frac{\partial^2 f}{\partial x_j^2}(\bar{x}) + 2\frac{\partial^2 f}{\partial x_j x_k}(\bar{x}) + \frac{\partial^2 f}{\partial x_k^2}(\bar{x}) \right) + O(h^4). \end{aligned}$$

Combining this with equation (2.6) shows that

$$\begin{aligned} \frac{\partial^2 f}{\partial x_j x_k}(\bar{x}) &= \tfrac{(f(\bar{x}) - \Re f(\bar{x} + ih(e_j + e_k)))}{h^2} - \tfrac{1}{2}\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) - \tfrac{1}{2}\tfrac{\partial^2 f}{\partial x_k^2}(\bar{x}) + O(h^2) \\ &= \tfrac{\Re(f(\bar{x} + (ih)e_j)) - \Re(f(\bar{x} + ih(e_j + e_k))) + \Re(f(\bar{x} + (ih)e_k)) - f(\bar{x})}{h^2} + O(h^2). \end{aligned} \tag{2.7}$$

Equations (2.5), (2.6), and (2.7) provide us with a method to approximate the gradient and Hessian of the function. We call this the *Basic Complex-step Quadratic Method (BCQM)* for constructing approximations of ∇f and $\nabla^2 f$.

The total number of distinct function evaluations used in the BCQM is $(n^2 + n + 2)/2$. This is only a minor improvement over the $(n^2 + 3n + 2)/2$ function evaluations required if we limit ourselves to only using real-valued points. However, we should also notice that quadratic interpolation using real-valued points results in a Hessian approximation with an error term that is $O(h)$, whereas the BCQM results in a Hessian approximation with an error term that is $O(h^2)$. So, our minor reduction in function evaluations has resulted in an increased curvature accuracy.

Remark 2.3. BCQM returns approximations for ∇f and $\nabla^2 f$, which we shall label G and H respectively. If one desires to create a quadratic model $Q(z) = \alpha_0 + g^\top z + \frac{1}{2}z^\top H z$, then it is necessary to remember that

$$\begin{aligned} H &= \nabla^2 Q(\bar{x}) && = H, \\ g &= \nabla Q(\bar{x}) - H\bar{x} && = G - H\bar{x}, \\ \alpha_0 &= Q(\bar{x}) - g^\top \bar{x} - \tfrac{1}{2}\bar{x}^\top H \bar{x} && = F - g^\top \bar{x} - \tfrac{1}{2}\bar{x}^\top H \bar{x}, \end{aligned}$$

where $F = f(\bar{x})$. (Note, we overload the variable H as it is consistent in both usages.)

2.3 General complex-step quadratic methods (GCQM)

The BCQM uses the imaginary-coordinate directions in order to generate gradient and Hessian approximations, which then can be used to generate a quadratic model. Like real-valued approaches, except for the simplicity in presentation, there is very little special about the coordinate directions. Indeed, in [16], Lai and Crassidis presented a variety of alternate methods. We present some of them here. The methods presented herein represent the methods with either the least required function evaluations or the highest quality error bound.

By Corollary 2.2(i) using $n \geq 2$

$$\begin{aligned} f(\bar{z} + h\exp(i\theta)e_j) &= f(\bar{z}) + \textstyle\sum_{k=1}^{n} \exp(ik\theta)\tfrac{h^k}{k!}\tfrac{\partial^k f}{\partial x_j^k}(\bar{z}) + O(h^{n+1}), \\ f(\bar{z} + h\exp(i(\theta + \pi))e_j) &= f(\bar{z}) + \textstyle\sum_{k=1}^{n} \exp(ik(\theta + \pi)))\tfrac{h^k}{k!}\tfrac{\partial^k f}{\partial x_j^k}(\bar{z}) + O(h^{n+1}). \end{aligned}$$

Subtracting and recalling that $\exp(i\theta) = -\exp(i(\theta + \pi))$ yields

$$\begin{aligned} & f(\bar{z} + h\exp(i\theta)e_j) - f(\bar{z} + h\exp(i(\theta + \pi))e_j) \\ & = 2\exp(i\theta)h\tfrac{\partial f}{\partial x_j}(\bar{z}) + \textstyle\sum_{k=2}^{n} \left(\exp(ik\theta) - \exp(ik(\theta + \pi))\right)\tfrac{h^k}{k!}\tfrac{\partial^k f}{\partial x_j^k}(\bar{z}) + O(h^{n+1}). \end{aligned}$$

Rearranging and applying $\exp(ik(\theta+\pi)) = \exp(ik\theta)\exp(ik\pi) = (-1)^k \exp(ik\theta)$ leads to the formula

$$\begin{aligned}\exp(i\theta)\tfrac{\partial f}{\partial x_j}(\bar{z}) = \quad & \tfrac{f(\bar{z}+h\exp(i\theta)e_j)-f(\bar{z}+h\exp(i(\theta+\pi))e_j)}{2h} \\ & - \textstyle\sum_{k=2}^{n} \exp(ik\theta)\tfrac{1-(-1)^k}{2}\tfrac{h^{k-1}}{k!}\tfrac{\partial^k f}{\partial x_j^k}(\bar{z}) + O(h^n).\end{aligned}$$

This reduces to the final formula

$$\begin{aligned}\exp(i\theta)\tfrac{\partial f}{\partial x_j}(\bar{z}) = \quad & \tfrac{f(\bar{z}+h\exp(i\theta)e_j)-f(\bar{z}+h\exp(i(\theta+\pi))e_j)}{2h} \\ & - \sum_{\substack{k=2 \\ k\ odd}}^{n} \exp(ik\theta)\tfrac{h^{k-1}}{k!}\tfrac{\partial^k f}{\partial x_j^k}(\bar{z}) + O(h^n).\end{aligned} \tag{2.8}$$

Notice, if $\theta = 0$ and $n = 2$, then this returns the classic centred-divided difference formula and error bound. But, if we select other values of θ, we get novel methods to approximate the gradient. Examples are given in the next subsections.

Returning to Corollary 2.2(i), this time with $n \geq 3$, following similar arguments it is straightforward to confirm the following equation that can be used to approximate the diagonal elements of the Hessian,

$$\begin{aligned}\exp(i2\theta)\tfrac{\partial^2 f}{\partial x_j^2}(\bar{z}) = \quad & \tfrac{1}{h^2}\left(f(\bar{z}+h\exp(i\theta)e_j) - 2f(\bar{z}) + f(\bar{z}+h\exp(i(\theta+\pi))e_j)\right) \\ & - \sum_{\substack{k=3 \\ k\ even}}^{n} 2\exp(ik\theta)\tfrac{h^{k-2}}{k!}\tfrac{\partial^k f}{\partial x_j^k}(\bar{z}) + O(h^{n-1}).\end{aligned} \tag{2.9}$$

Notice, if $\bar{z} = \bar{x} \in \mathbb{R}^n$, $\theta = 0$, and $n = 3$, then this recovers the classic second derivative midpoint formula. Examples that use other values for θ are given in the next subsections.

To approximate the off-diagonal elements of the Hessian we use Corollary 2.2(ii) with $n \geq 3$ to find

$$\begin{aligned}&\exp(2i\theta)\tfrac{\partial^2 f}{\partial x_j \partial x_k}(\bar{z}) = \\ &\tfrac{1}{2h^2}\left(f(\bar{z}+h\exp(i\theta)(e_j+e_k)) + f(\bar{z}+h\exp(i(\theta+\pi))(e_j+e_k)) - 2f(\bar{z})\right) \\ &-\tfrac{\exp(2i\theta)}{2}\left(\tfrac{\partial^2 f}{\partial x_j^2} + \tfrac{\partial^2 f}{\partial x_k^2}\right)(\bar{z}) - 2\sum_{\substack{|\alpha|=3 \\ |\alpha|\ even}}^{n} \exp(i|\alpha|\theta)h^{|\alpha|-2}\frac{1}{\alpha!}\partial^\alpha f(\bar{z}) + O(h^{n-1}).\end{aligned} \tag{2.10}$$

Using equations (2.8), (2.9), and (2.10), we can construct a wide variety of gradient and Hessian approximation techniques. For a given θ, we shall call this the *General Complex-step Quadratic Method-θ* (GCQM-θ). Two of the most interesting of these are when $\theta = \pi/4$ and $\theta = \pi/3$. The details are presented in the following subsections. Section 2.6 further presents the use of Richardson extrapolation on the first variation of GCQM.

Note that GCQM-θ constructs gradient and Hessian approximations. If a quadratic model is desired, then the process in Remark 2.3 should be applied.

2.4 GCQM-$\pi/4$

Substituting $\theta = \frac{\pi}{4}$ and $n = 2$ in (2.8) at the point $\bar{x} \in \mathbb{R}^n$, and combining it with $\exp(i\theta) = i^{2\theta/\pi}$, we obtain

$$\exp(i\pi/4)\frac{\partial f}{\partial x_j}(\bar{x}) = \frac{1}{2h}(f(\bar{x}+hi^{1/2}e_j) - f(\bar{x}+hi^{5/2}e_j)) + O(h^2).$$

Using $\exp(i\pi/4) = (i+1)/\sqrt{2}$ and taking imaginary parts on both sides yields

$$\frac{\partial f}{\partial x_j}(\bar{x}) = \frac{1}{\sqrt{2}h}\Im(f(\bar{x} + hi^{1/2}e_j) - f(\bar{x} + hi^{5/2}e_j)) + O(h^2). \tag{2.11}$$

The error bound for this gradient approximation is equivalent to that of BCQM.

Substituting $\theta = \frac{\pi}{4}$ and $n = 5$ in (2.9), again using $\exp(i\theta) = i^{2\theta/\pi}$, we obtain

$$\begin{aligned}
\exp(i\pi/2)\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) = \quad & \tfrac{1}{h^2}\left(f(\bar{x} + hi^{1/2}e_j) - 2f(\bar{x}) + f(\bar{x} + hi^{5/2}e_j)\right) \\
& -2\exp(i\pi)\tfrac{h^2}{4!}\tfrac{\partial^4 f}{\partial x_j^4}(\bar{x}) + O(h^4), \\
i\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) = \quad & \tfrac{1}{h^2}\left(f(\bar{x} + hi^{1/2}e_j) - 2f(\bar{x}) + f(\bar{x} + hi^{5/2}e_j)\right) \\
& +2\tfrac{h^2}{4!}\tfrac{\partial^4 f}{\partial x_j^4}(\bar{x}) + O(h^4).
\end{aligned}$$

Taking the imaginary parts on both sides, recalling that $x \in \mathbb{R}^n$ implies $f(x) \in \mathbb{R}$ by assumption, yields

$$\frac{\partial^2 f}{\partial x_j^2}(\bar{x}) = \frac{1}{h^2}\Im(f(\bar{x} + hi^{1/2}e_j) + f(\bar{x} + hi^{5/2}e_j)) + O(h^4).$$

Note that, here, the error bound is $O(h^4)$ as compared to the $O(h^2)$ obtained in BCQM calculations.

Finally, for off-diagonal elements of the Hessian, substituting $\theta = \frac{\pi}{4}$ and $n = 5$ in (2.10) yields

$$\begin{aligned}
\exp(i\pi/2)\tfrac{\partial^2 f}{\partial x_j \partial x_k}(\bar{x}) = \quad & \tfrac{1}{2h^2}\left(f(\bar{x} + hi^{1/2}(e_j + e_k)) - 2f(\bar{x}) + f(\bar{x} + hi^{5/2}(e_j + e_k))\right) \\
& -\tfrac{\exp(i\pi/2)}{2}\left(\tfrac{\partial^2 f}{\partial x_j^2} + \tfrac{\partial^2 f}{\partial x_k^2}\right)(\bar{x}) - 2\exp(i\pi)h^2 \sum_{|\alpha|=4} \frac{1}{\alpha!}\partial^\alpha f(\bar{x}) + O(h^4), \\
i\tfrac{\partial^2 f}{\partial x_j \partial x_k}(\bar{x}) = \quad & \tfrac{1}{2h^2}\left(f(\bar{x} + hi^{1/2}(e_j + e_k)) - 2f(\bar{x}) + f(\bar{x} + hi^{5/2}(e_j + e_k))\right) \\
& -\tfrac{i}{2}\left(\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) + \tfrac{\partial^2 f}{\partial x_k^2}(\bar{x})\right) + 2h^2 \sum_{|\alpha|=4} \frac{1}{\alpha!}\partial^\alpha f(\bar{x}) + O(h^4).
\end{aligned}$$

Taking the imaginary parts of both sides, recalling that $\bar{x} \in \mathbb{R}^n$ implies $f(\bar{x}) \in \mathbb{R}$ by assumption, yields

$$\begin{aligned}
\frac{\partial^2 f}{\partial x_j \partial x_k}(\bar{x}) = \frac{1}{2h^2}\Im(f(\bar{x} + hi^{1/2}(e_j + e_k)) + f(\bar{x} + hi^{5/2}(e_j + e_k))) \\
- \frac{1}{2}\left(\frac{\partial^2 f}{\partial x_j^2}(\bar{x}) + \frac{\partial^2 f}{\partial x_k^2}(\bar{x})\right) + O(h^4).
\end{aligned}$$

Note that, again the error bound is $O(h^4)$ as compared to the $O(h^2)$ obtained in BCQM calculations.

Examining the function evaluations used, notice that the gradient approximation and the diagonal elements of the Hessian approximation both use the same objective function values: $f(\bar{x} + hi^{1/2}e_j)$ and $f(\bar{x} + hi^{5/2}e_j)$. These sum up to $2n$ function evaluations. The off-diagonal elements of the Hessian approximation each use two more function evaluations: $f(\bar{x} + hi^{1/2}(e_j + e_k))$ and $f(\bar{x} + hi^{5/2}(e_j + e_k))$. Through symmetry, only the upper portion of the Hessian approximation needs to be constructed, this requires $2n(n-1)/2 = n^2 - n$ function evaluations. Therefore, the total number of function evaluations used in this complex-step method is $n^2 + n$.

2.5 GCQM-$\pi/3$

Another proposed variation of the general complex-step quadratic method is at $\theta = \frac{\pi}{3}$. An alternate derivation can be found in [16].

Similar to previous calculations, substituting $\theta = \frac{\pi}{3}$ and $n = 4$ in equation (2.8) at the point $x \in \mathbb{R}^n$, we obtain

$$\frac{\partial f}{\partial x_j}(\bar{x}) = \frac{1}{\sqrt{3}h}\Im(f(\bar{x} + hi^{2/3}e_j) - f(\bar{x} + hi^{8/3}e_j)) + O(h^4).$$

Notice the error term in this approximation is $O(h^4)$, in comparison with the $O(h^2)$ in the previous methods.

To calculate diagonal elements of Hessian, we substitute $\theta = \frac{\pi}{3}$ and $n = 3$ in equation (2.9) to obtain

$$\frac{\partial^2 f}{\partial x_j^2}(\bar{x}) = \frac{2}{\sqrt{3}h^2}\Im(f(\bar{x} + hi^{2/3}e_j) + f(\bar{x} + hi^{8/3}e_j)) + O(h^2).$$

The off-diagonal elements of the Hessian are generated by substituting $\theta = \frac{\pi}{3}$ and $n = 3$ in equation (2.10),

$$\begin{aligned}\frac{\partial^2 f}{\partial x_j \partial x_k}(\bar{x}) = \quad & \frac{1}{\sqrt{3}h^2}\Im(f(\bar{x} + hi^{2/3}(e_j + e_k)) + f(\bar{x} + hi^{8/3}(e_j + e_k))) \\ & -\frac{1}{2}\left(\frac{\partial^2 f}{\partial x_j^2} + \frac{\partial^2 f}{\partial x_k^2}\right)(\bar{x}) + O(h^2).\end{aligned}$$

The Hessian approximation for this method has error $O(h^2)$.

The number of function evaluations required by this method are identical to GCQM-$\pi/4$ from Subsection 2.4. The gradient vector and the diagonal elements of the Hessian are computed using the same objective function values $f(\bar{x} + hi^{2/3}e_j)$ and $f(\bar{x} + hi^{8/3}e_j)$. The off-diagonal elements of the Hessian use function values $f(\bar{x} + hi^{2/3}(e_j + e_k))$ and $f(\bar{x} + hi^{8/3}(e_j + e_k))$. Therefore, the total number of function evaluations used in the complex-step method with $\theta = \frac{\pi}{3}$ is $n^2 + n$.

2.6 GCQM-$\pi/4$ using Richardson Extrapolation

We next present a variation of the GCQM-$\pi/4$ that provides an error term of $O(h^4)$ for both gradient and Hessian approximations. This improvement in accuracy is achieved through Richardson extrapolation.

Applying $\theta = \pi/4$ and $n = 4$ in equation (2.8) with yields

$$\exp(i\pi/4)\frac{\partial f}{\partial x_j}(\bar{z}) = \frac{f(\bar{z}+hi^{1/2}e_j)-f(\bar{z}+hi^{5/2}e_j)}{2h} - i^{3/2}\frac{h^2}{6}\frac{\partial^3 f}{\partial x_j^3}(\bar{z}) + O(h^4) \tag{2.12}$$

and (by substituting h with $h/2$)

$$\exp(i\pi/4)\frac{\partial f}{\partial x_j}(\bar{z}) = \frac{f(\bar{z}+\frac{1}{2}hi^{1/2}e_j)-f(\bar{z}+\frac{1}{2}hi^{5/2}e_j)}{h} - i^{3/2}\frac{h^2}{24}\frac{\partial^3 f}{\partial x_j^3}(\bar{z}) + O(h^4). \tag{2.13}$$

Multiplying equation (2.13) by 4 and subtracting equation (2.12), after simplification, produces

$$\frac{\partial f}{\partial x_j} = \frac{\Im\left(8(f(x+\frac{h}{2}i^{1/2}e_j)-f(x+\frac{h}{2}i^{5/2}e_j))-(f(x+hi^{1/2}e_j)-f(x+hi^{5/2}e_j))\right)}{3\sqrt{2}h} + O(h^4).$$

This method uses $4n$ function evaluations to approximate the gradient, instead of the $2n$ used by GCQM-$\pi/4$, but provides a gradient error of $O(h^4)$. The Hessian calculations are unchanged from Subsection 2.4. Therefore, total number of function evaluations required are $n^2 + 3n$. Henceforth we shall refer to this method as GCQM-$\pi/4$-R.

3 Using Complex-step Methods in an Approximate Newton's Algorithm

In Section 2, we outlined 4 methods to approximate gradients and Hessians: BCQM, GCQM-$\pi/4$, GCQM-$\pi/3$, and GCQM-$\pi/4$-R. Our goal is to use these 4 methods in a Newton style algorithm to gauge their effectiveness to help solve optimization problems. In addition to the 4 complex-variable based methods, we shall consider 2 other methods.

The first will be to use the true gradient and true Hessian. In evaluating this approach with regards to function evaluations, we consider each required value to be the result of 1 function call. That is, the gradient vector at $\bar{x}$ costs n function calls (to n different functions). Since Hessian is symmetric, the diagonal and off-diagonal elements of Hessian are evaluated through $n(n+1)/2$ function calls. Thus, we consider the total number of function calls used in this method to be $n + n(n+1)/2 = (n^2+3n)/2$. Henceforth, we shall refer to this as the TVQM (True Valued Quadratic Method).

Remark 3.1. We note that $(n^2+3n)/2$ is an inaccurate surrogate for the amount of function call equivalents used in TVQM. One referee stated that "Depending on f and the implementation of the gradient and Hessian functions, the function calls required for TVQM could be as low as 1 or as high as $(n^2+3n)/2$." There is no correct answer to this, as any real-world comparison would be oracle dependent. As such, our analysis will primarily focus on the approximation based methods, while we maintain TVQM as a baseline. All results involving comparing function evaluations used by TVQM should be read with this in mind.

We also remark that, similar to BCQM, GCQM-$\pi/4$, GCQM-$\pi/3$, and GCQM-$\pi/4$-R, when considering TVQM, we only consider the number of function evaluations to construct the gradient and Hessian. As we shall be using Newton's method (see Subsection 3.1), the actual value of f is not used.

The second real-valued method we shall consider will be constructed via quadratic interpolation using only real-valued points. Consider a quadratic function defined via

$$Q(x) = \alpha_0 + g^\top x + \frac{1}{2} x^\top H x$$

where $\alpha_0 \in \mathbb{R}$, $g \in \mathbb{R}^n$, and $H \in \mathbb{R}^{n\times n}$. Section 2 outlines that a well-poised problem requires $(n+1)(n+2)/2 = (n^2+3n+2)/2$ function evaluations.

By Corollary 2.2(i) at $\theta = 0$, the 2^{nd}-order Taylor expansion of f at a fixed point $\bar{x} \in \mathbb{R}$ satisfies

$$f(\bar{x} + he_j) \quad = \quad f(\bar{x}) + h\tfrac{\partial f}{\partial x_j}(\bar{x}) + \tfrac{h^2}{2!}\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) + O(h^3).$$

Similarly, at $\theta = \pi$,

$$f(\bar{x} - he_j) \quad = \quad f(\bar{x}) - h\tfrac{\partial f}{\partial x_j}(\bar{x}) + \tfrac{h^2}{2!}\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) + O(h^3).$$

Combining these, the first-order partial derivatives are given by

$$\tfrac{\partial f}{\partial x_j}(\bar{x}) \quad = \quad \tfrac{1}{2h}(f(\bar{x} + he_j) + f(\bar{x} - he_j)) + O(h^2).$$

Using the above result, we also obtain the second-order partial derivatives,

$$\tfrac{\partial^2 f}{\partial x_j^2}(\bar{x}) \quad = \quad \tfrac{2}{h^2}(f(\bar{x} + he_j) - f(\bar{x})) - \tfrac{2}{h}(\tfrac{\partial f}{\partial x_j}(\bar{x})) + O(h).$$

Finally, using Corollary 2.2(ii) at $\theta = 0$ with the above results yields,

$$\begin{aligned}\frac{\partial^2 f}{\partial x_j \partial x_k}(\bar{x}) = \quad & \frac{1}{h^2}(f(\bar{x} + h(e_j + e_k)) - f(\bar{x})) - \frac{1}{h}(\frac{\partial f}{\partial x_j} + \frac{\partial f}{\partial x_k})(\bar{x}) \\ & -\frac{1}{2}(\frac{\partial^2 f}{\partial x_j^2} + \frac{\partial^2 f}{\partial x_k^2})(\bar{x}) + O(h).\end{aligned}$$

Henceforth, we shall refer to this as the RQM (Real-step Quadratic Method).

RQM uses a total number of $1 + 2n + n(n-1)/2 = (n^2 + 3n + 2)/2$ distinct function evaluations.

Table 1 summarizes the methods used, the order of the error bounds, and the number of function calls required by the method. This summary will provide us a better view for comparison of the performance of these methods in terms of complexity and accuracy.

Table 1: The error bounds and function calls used by proposed methods as used with Newton's method; h denotes the real step-size and n is the dimension of the optimization problem. See Remark 3.1 regarding function calls for TVQM.

Approximation Method	Gradient error	Hessian error	Function Calls
TVQM	$O(0)$	$O(0)$	$\frac{1}{2}(n^2 + 3n)$
RQM	$O(h^2)$	$O(h)$	$\frac{1}{2}(n^2 + 3n + 2)$
BCQM	$O(h^2)$	$O(h^2)$	$\frac{1}{2}(n^2 + n + 2)$
GCQM-$\pi/4$	$O(h^2)$	$O(h^4)$	$n^2 + n$
GCQM-$\pi/3$	$O(h^4)$	$O(h^2)$	$n^2 + n$
GCQM-$\pi/4$-R	$O(h^4)$	$O(h^4)$	$n^2 + 3n$

3.1 Newton's Optimization Algorithm

In order to gauge the value of different methods in an optimization algorithm, the six methods in Table 1 were used in a Newton style algorithm. Assuming a true Hessian and gradient, the classical Newton step for minimization is provided by

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

Our approximate Newton's methods simply replaces the true gradient/Hessian values with approximations.

We shall use a target-based approach to evaluate the performance. That is, how many resources are required to achieve a solution with a target accuracy. As we are using a target-based approach, our stopping criterion will be that a desired relative-reduction has been achieved. We can do this, as all of our test problems have known solutions. To avoid infinite looping, a maximum number of function calls is also enforced.

Details of our test algorithm are provided in Algorithm 1.

The approximation methods and optimization algorithm are implemented in MATLAB version 2021a.

3.2 Test Problems

The methods were tested on 40 optimization problems from [23]. It is well known that Newton's method is quadratically convergent provided that the function is well-behaved and

Algorithm 1 Newton's optimization algorithm

1: **Inputs:**
- x_0 : Initial point
- `Method` : Method for gradient/Hessian approximation
- h : Step size used in approximation methods
- `MaxFCall` : Maximum number of function calls allowed
- `StopTol` : Stopping tolerance for relative reduction
- f^* : True optimal value

2: **Initialize:**
- $k \leftarrow 0$: Iteration counter
- `FCall` $\leftarrow 0$: Function call counter

3: **Iterations:**
- $x_{k+1} \leftarrow x_k - (H)^{-1} g$
- $k \leftarrow k + 1$
- where,
  - g = Approximated gradient by `Method` at x_k
  - H = Approximated Hessian by `Method` at x_k
- Increment `FCall` by number of function calls used

4: **Stopping Conditions:**
- If $\dfrac{|f(x_k) - f^*|}{|f(x_0) - f^*|} <$ `StopTol`, then **STOP** (Success)
- If `FCall` $>$ `MaxFCall`, then **STOP** (Failure)

the initial point is sufficiently close to a minimizer [24]. On the other hand, if the function is misbehaved, or the initial point is too far from a local minimizer, then Newton's method can fail spectacularly [24]. Therefore, we begin by running TVQM on all test problems, using the recommended initial point, to see if Newton's method will converge under ideal conditions (i.e., when the gradient and Hessian are exact).

If Newton's method converges for the TVQM in less than $50n^2$ function evaluations, where n is the dimension of each problem, then we keep the problem as an *intriguing* test problem. Recalling Table 1, we can see that all approximation techniques use $O(n^2)$ evaluations, which explains the n^2 in this value. Recalling that Newton's method is quadratically convergent, as the dimension increases, the number of iterations is not expected in increase dramatically. The 50 is chosen as a constant that should be well above the convergence needs for Newton's method.

If Newton's method fails under these ideal conditions, then we have no expectation of convergence when the gradient and Hessian approximations contains errors. In this case, we remove the test problem. This left 26 test problems.

Table 2 lists the names and dimensions of the 26 test problems that were found intriguing and therefore used for the remaining tests.

4 Numerical Tests and Results

To gauge the impact of gradient and Hessian accuracy, various values of the step-size h and stopping tolerance `StopTol` were tested. In particular, $h \in \{1/2, 1/2^2, 1/2^4, 1/2^8, 1/2^{16}, 1/2^{32}\}$ and `StopTol` $\in \{10^{-3}, 10^{-6}, 10^{-9}\}$ were tested. We consider an algorithm successful if it identifies a minimum value with relative reduction less than `StopTol` (see Algorithm 1 step 4) in less than $5N_{\texttt{it}}$ iterations, where $N_{\texttt{it}}$ is the number of iterations required by TVQM to

Table 2: Test Problems

No.	Function Name	n	No.	Function Name	n
1	Rosenbrock	2	14	Extended Powell	12
2	Jennrich and Sampson	2	15	Penalty	4
3	Helical valley	3	16	Penalty	10
4	Bard	3	17	Penalty II	4
5	Box three-dimensional	3	18	Variably dim.	4
6	Powell singular	4	19	Variably dim.	8
7	Brown and Dennis	4	20	brown almost linear	4
8	Osborne 1	5	21	Discrete boundary value	4
9	Watson	6	22	Discrete integral eq.	4
10	Watson	9	23	Broyden Tridiagonal	4
11	Extended Rosenbrock	6	24	Broyden Tridiagonal	12
12	Extended Powell	4	25	Broyden banded	4
13	Extended Powell	8	26	Linear - full rank	4

solve the same problem with the same stopping tolerance.

Complete testing data and software is available upon request to the corresponding author. Herein we limit ourselves to analysis via data and performance profiles. We present data profiles [22] (based on function calls) and performance profiles [7] (based on cpu time) for interpretation of our results.

4.1 Data Profiles

Data profiles were developed especially for numerical comparison of algorithms in derivative-free optimization [22]. The x-axis counts the number of function evaluations divided by the dimension of the problem plus 1: *f-evals*$/(n+1)$. The division by $n+1$ represents the number of function evaluations required to construct a simplex gradient in $\mathbb{R}^n$. The y-axis gives the portion of problems solved for that ratio. Please see [22] for further details.

With 6 values of h and 3 stopping tolerances tested, there are 18 data profiles. All data profiles are available by request to the corresponding author. Herein we focus on two key trends that arise when examining the profiles: the impact of `StopTol` and the impact of h.

Impact of `StopTol`

The first trend is the unsurprising result that as `StopTol` is decreased from 10^{-3} to 10^{-9} the performance of every algorithm decreases. This is visualized in Figure 1, which compares the three data profiles using $h = 1/2^4$.

Examining Figure 1, notice that while every algorithm is impacted by changing `StopTol`, the amount of impact is not equal. The TVQM is barely impacted. In fact, on most problems TVQM only takes 1 or 2 more iterations to achieve `StopTol` $= 10^{-6}$ as it took to achieve `StopTol` $= 10^{-3}$. Similarly TVQM only takes 1 or 2 more iterations to achieve `StopTol` $= 10^{-9}$ as it took to achieve `StopTol` $= 10^{-6}$. This is expected in light of the quadratic convergence of Newton's method.

Interestingly, GCQM-$\pi/4$-R also shows a very robust convergence. Although the change in function calls required is more dramatic, the GCQM-$\pi/4$-R solves the same number of problems regardless of `StopTol`. The remaining methods (RQM, BCQM, GCQM-$\pi/4$, and

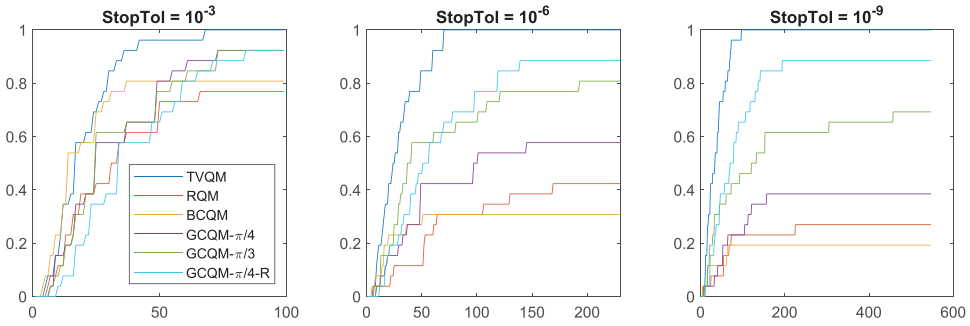

Figure 1: Data profiles resulting from $h = 1/2^4$.

GCQM-$\pi/3$) all see a decrease in the number of problems that they are able to solve as the value of `StopTol` decreases.

Notice that when `StopTol`=10^{-3} the algorithms are fairly clustered with no clear winner from among the approximation methods. But, when `StopTol`=10^{-9} there is a very clear ranking across the methods. From best to worst we have GCQM-$\pi/4$-R, GCQM-$\pi/3$, GCQM-$\pi/4$, RQM, and BCQM. Compare this to Table 1, it appears that the improved accuracy of the approximations used in GCQM-$\pi/4$-R, GCQM-$\pi/3$, GCQM-$\pi/4$ are allowing for more accurate final solutions. Surprisingly, the BCQM is ranking last, instead of second last. However, both RQM and BCQM are doing exceedingly poorly, solving less than 25% of problems, so this result is more likely due to random aspects of the problems than of attributes of the approximation methods. I.e., it is more likely that this result is due to RQM getting 'lucky' on some test problems, than due to RQM actually having some advantage over BCQM for $h = 1/2^4$. Indeed, when h becomes smaller, we shall see BCQM outperform RQM. We explore this next.

Overall, these results support claim (i) on page 393 that *higher accuracy approximations improve the chances of solving a given problem.*

Impact of h

Examining the impact of h, the trend becomes more complex. We select the three data profiles using $h \in \{1/2^8, 1/2^{16}, 1/2^{32}\}$ and `StopTol`$= 10^{-9}$ to illustrate this trend. These are presented in Figure 2.

Comparing $h = 1/2^8$ to $h = 1/2^4$ (from Figure 1), we see that both RQM and BCQM are now solving over 50% of problems and that BCQM is now outperforming RQM.

For all of the complex step methods (BCQM, GCQM-$\pi/4$, GCQM-$\pi/3$, and GCQM-$\pi/4$-R), we see a steady trend that as h decreases the number of test problems solved increases. RQM also follows this trend for $h \in \{1/2^8, 1/2^{16}\}$, but when $h = 1/2^{32}$ RQM solves no problems. In fact, RQM crashes from numerical errors on every test problem when $h = 1/2^{32}$. This supports the fact that complex step methods are less prone to numerical errors than real valued methods.

Perhaps surprising is that when $h = 1/2^{32}$, we still do not have any of the approximation methods solve all problems. Examining Table 1, notice that when $h = 1/2^{32}$, errors $O(h^4)$ result in theoretical errors of order $1/2^{128} \approx 10^{-39}$. As this value is well below machine precision, floating-point errors are inevitably invoked [16]. As such, while we might theoretical hope to achieve a final precision of 10^{-9}, it is not numerically ensured and should not be

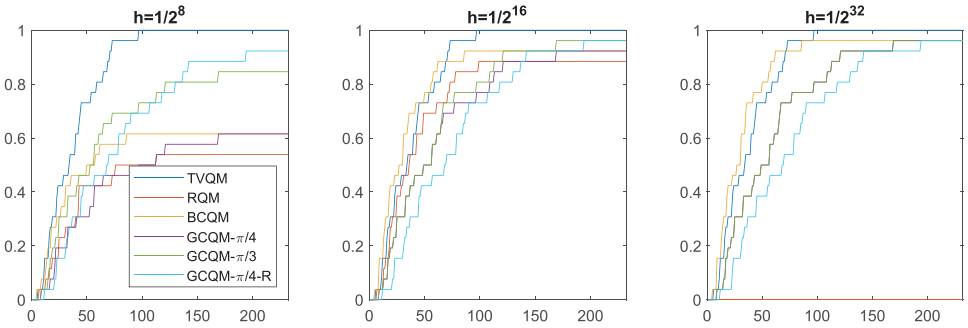

Figure 2: Select data profiles resulting from `StopTol`$= 10^{-9}$.

expected on a standard machine. Interestingly, BCQM fails to solve problem 8, while the GCQM-$*$ methods fail to solve problem 3. (TVQM solved all problems and RQM failed on all problems using $h = 1/2^{32}$, as mentioned above.)

Overall, these results support claim (ii) on page 393 that *avoiding catastrophic cancellation allows smaller step sizes to be applied safely.*

4.2 Performance Profiles

Examining the data profiles in Figures 1 and 2, notice that all profiles tend to climb at approximately the same slope. The implies that in terms of number of function calls, all methods converge similarly. I.e., the main difference lies in whether a method can solve a problem, not how fast (in terms of function calls) that solution is obtained. However, this conclusion ignores the fact that function evaluations at complex-valued points can be more time consuming than function evaluations at real-valued points. Therefore we perform further analysis examining performance in terms of time.

Before presenting the time-based analysis, let us demonstrate how function evaluations at complex-valued points can be more time consuming than function evaluations at real-valued points.

Example 4.1. Consider the following simple experiment in MATLAB. First, use a simple for-loop to sum the squares of the numbers $k \in \{1, 2, ..., MAX\}$. Next, use similar for-loop to sum the squares of the numbers $k \in \{1 + i, 2 + i, ..., MAX + i\}$. In Table 3 we present the time results for $MAX \in \{10^2, 10^4, 10^6, 10^8\}$.
Notice that the time to work with complex variables is consistently one order of magintude higher than the time to work with just real variables.

Returning to our numerical experiments, we now present performance profiles focused on cpu time.

Performance profiles provide a comprehensive visualization of the different optimization methods on the entire set of test problems [7]. The x-axis gives the ratio of cpu time used in comparison to the best solver for a given problem. The y-axis gives the portion of problems solved for that ratio. Please see [7] for further details.

In Figure 3 we present the performance profiles corresponding to $h = 1/2^4$ for the three values of `StopTol`. Note, the performance profiles use log-base-10 on the x-axis.

We note that for-loops in MATLAB can be highly inefficient, but the purpose of this is demonstration so the approach is justified.

Table 3: Time differences arising through the use of complex variables

MAX	Time (s) to find $\sum_{k=1}^{MAX}(k)^2$	Time (s) to find $\sum_{k=1}^{MAX}(k+i)^2$
10^2	6.2e-04	1.4e-03
10^4	1.8e-04	1.8e-03
10^6	7.9e-03	3.8e-02
10^8	1.1e-01	2.5e+00

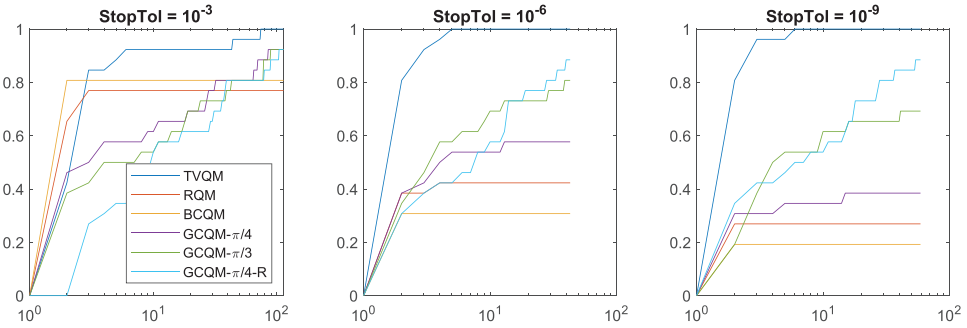

Figure 3: Performance profiles based on cpu time resulting from $h = 1/2^4$.

Comparing Figure 1 to Figure 3, notice the changes in slope in various methods.

For example, examining RQM with `StopTol`$=10^{-3}$, we see that the data profile suggests RQM has one of the slowest speeds of solving (in terms of function calls). However, the performance profile suggests that RQM has a faster speed of solving (in terms of cpu time) than TVQM.

Another clear example comes from examining GCQM-$\pi/4$-R. The data profiles all suggest that (in terms of function calls), GCQM-$\pi/4$-R takes about twice as long as TVQM to solve 80% of problems. However, the performance profiles make it clear that this solve time (in terms of cpu time) is closer to 80 times as long.

The performance profiles involving other values of h show a similar trend, although less dramatically.

Overall, these results contradict claim (iii) on page 393. In particular, it appears that *the use of complex variable requires an increased cpu time that outstrips any reduction in function calls or iterations.*

In Figure 4 we present the performance profiles corresponding to $h = 1/2^{16}$ for the three values of `StopTol`. Like Figure 1, the performance profiles use log-base-10 on the x-axis.

Examining Figure 4, we again see that RQM is faster in terms of cpu time than other methods. However, when `StopTol`$= 10^{-9}$, RQM struggles to solve as many problems as other methods.

4.3 Impact of dimension

It is natural to wonder if dimension impacts the solve time of each method equally. In Table 4 we present the average solve time for each method (in seconds) over varying dimensions

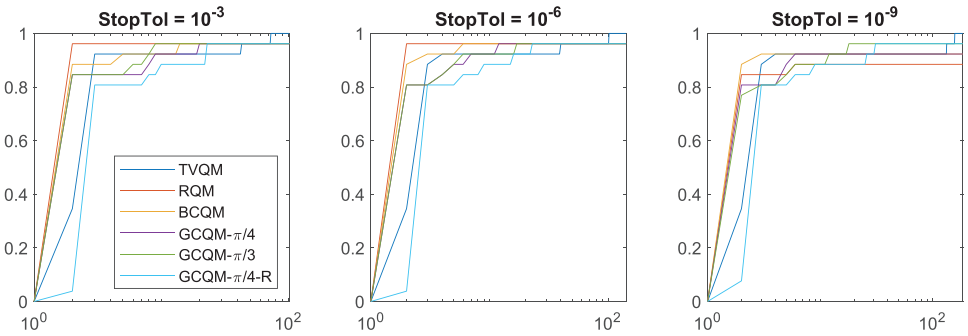

Figure 4: Performance profiles based on cpu time resulting from $h = 1/2^{16}$.

when `StopTol` $= 10^{-6}$ and $h = 1/2^{16}$. Only problems solved by all solvers are considered.

Table 4: Average time (seconds) to solve problems of varying dimension to an accuracy of 10^{-6} for each method. Only problems solved by all solvers are considered.

Dim.	TVQM	RQM	BCQM	GCQM-$\pi/4$	GCQM-$\pi/3$	GCQM-$\pi/4$-R
2-3	0.97	0.47	0.63	1.23	1.27	2.02
4-5	2.12	1.07	0.91	1.81	1.81	3.05
6-9	241.66	6.15	5.57	10.06	10.10	12.19
10-12	21.69	13.46	12.29	23.00	23.07	26.55

Examining Table 4, we remark that TVQM took an excessively long time to solve problems 9 and 10 (Watson 6-variable and Watson 9-variables). This skews the data for TVQM when considering problem of with dimensions from 6 to 9. Setting this value aside, we see that GCQM-$\pi/4$, GCQM-$\pi/3$, and GCQM-$\pi/4$-R, generally take about twice as long as RQM or BCQM regardless of dimension. So, this limited data set suggests that all of the methods are impacted equally by dimension.

5 Conclusions

In this research, we have explored the value of using complex-variables in a gradient and Hessian approximation for derivative-free optimization. Our analysis has presented 4 methods of approximating gradients and Hessians making use of complex-variables. Other methods can be found in [16]. Our numerical tests have used these methods, along with a classical real-variable method for gradient and Hessian approximation and a method using the true gradient and Hessians. These methods were embedded into a Newton style algorithm and tested on a suite of 26 test problems.

The results suggest that while complex-step approximations,

- provide higher accuracy, so improve the chances of solving a given problem; and
- avoid catastrophic cancellation, so allow smaller step sizes to be applied safely;

complex-step approximations

- require an increased cpu time that outstrips any reduction in function calls used or iterations applied.

As such, complex-step approximations should be reserved for environments where solution quality is of greater importance than solve time.

It is worth noting that, unless Hessians are easily provided, Newton's method is seldom applied. Instead, quasi-Newton methods, such as BFGS, are more likely to be applied [24, Chpt 6]. Indeed, instead of attempting to approximate an accurate Hessian every iteration, the BFGS method uses one additional gradient to slightly improve an approximate Hessian every iteration. Future research could explore whether any conclusions of this research change when an alternate method is considered.

Acknowledgements

Hare's research partially supported by NSERC of Canada Discovery Grant 2018-03865. Srivastava's research supported by Mitacs Globalink. The authors acknowledge the fruitful conversations with anonymous referees which lead to a better presentation of this work.

References

[1] A. Abokhodair, Complex differentiation tools for geophysical inversion, *Geophys.* 74 (2009) H1–11.

[2] D. Ackerer and D. Filipović, Option pricing with orthogonal polynomial expansions, *Math. Finance* 30 (2020), 47–84.

[3] A.H. Al-Mohy and N.J. Higham, The complex step approximation to the Fréchet derivative of a matrix function, *Numer. Algorithms* 53 (2010) 133–148.

[4] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*, Springer Series in Operations Research and Financial Engineering. Springer International Publishing AG, 2017.

[5] A.R. Conn and P.L. Toint, An algorithm using quadratic interpolation for unconstrained derivative free optimization, in: *Nonlinear Optim. Appl.* G. Di Pillo and F. Giannessi (eds.), Springer, 1996, pp. 27–47.

[6] A.L. Custódio, H. Rocha, and L.N. Vicente, Incorporating minimum Frobenius norm models in direct search, *Comput. Optim. Appl.* 46 (2009) 265–278.

[7] E.D. Dolan and J.J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.* 91 (2002) 201–213.

[8] M. Giles and M. Duta, Algorithm developments for discrete adjoint methods, *AIAA J.* 41 (2003) 198–205.

[9] W. Hare, A discussion on variational analysis in derivative-free optimization, *Set-Valued Var. Anal.* 28 (2020) 643–659.

[10] W. Hare and G. Jarry-Bolduc, Calculus identities for generalized simplex gradients: Rules and applications. *SIAM J. Optim.* 30 (2020) 853–884.

[11] W. Hare, G. Jarry–Bolduc and C. Planiden, Error bounds for overdetermined and underdetermined generalized centred simplex gradients, *IMA J. Numer. Anal.* 12 (2020) 744–770.

[12] W. Hare, C. Planiden and C. Sagastizábal, A derivative-free VU-algorithm for convex finite-max problems, *Optim. Methods Softw.* 35 (2020) 521–559.

[13] R.W. Ibrahim and H.A. Jalab, Image denoising based on approximate solution of fractional cauchy-euler equation by using complex-step method, *Iran. J. Sci. Technol.* 39 (2015) 243–251.

[14] K.A. Khan, J. Larson, and S.M. Wild, Manifold sampling for optimization of nonconvex functions that are piecewise linear compositions of smooth components, *SIAM J. Optim.* 28 (2018) 3001–3024.

[15] K.-L. Lai and J. Crassidis, Generalizations of the complex-step derivative approximation, in: *AIAA Guidance, Navigation, and Control Conference*, 2006, p. 6348.

[16] K.-L. Lai and J.L. Crassidis, Extensions of the first and second complex-step derivative approximations, *Journal of Computational and Applied Mathematics* 219 (2008) 276–293.

[17] J. Larson, M. Menickelly and S.M. Wild, Manifold sampling for ℓ_1 nonconvex optimization, *SIAM J. Optim.* 26 (2016) 2540–2563.

[18] D. Lera and Y.D. Sergeyev, GOSH: derivative-free global optimization using multi-dimensional space-filling curves, *J. Global Optim.* 71 (2018) 193–211.

[19] J.N. Lyness and C.B. Moler, Numerical differentiation of analytic functions, *SIAM J. Numer. Anal.* 4 (1967) 202–210.

[20] J. Martins and J. Hwang, Review and unification of methods for computing derivatives of multidisciplinary computational models, *AIAA J.* 51 (2013) 2582–2599.

[21] J. Martins, P. Sturdza and J.J. Alonso, The complex-step derivative approximation, *ACM Trans. Math. Softw.* 29 (2003) 245–262.

[22] J.J. Moré and S.M. Wild, Benchmarking derivative-free optimization algorithms, *SIAM J. Optim.* 20 (2009) 172–191.

[23] J.J. Moré, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Softw.* 7 (1981) 17–41.

[24] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer New York, 2006.

[25] R. Oeuvray and M. Bierlaire, Boosters: A derivative-free algorithm based on radial basis functions, *Int. J. Model. Sim.* 29 (2009) 26–36.

[26] M.J. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in: *Advances in Optimization and Numerical Analysis*, Springer, 1994, pp. 51–67.

[27] M.J.D. Powell, Least Frobenius norm updating of quadratic models that satisfy interpolation conditions, *Math. Program.* 100 (2004) 183–215.

[28] M.J.D. Powell, Developments of NEWUOA for minimization without derivatives, *IMA J. Numer. Anal.* 28 (2008):649–664.

[29] R. Regis, The calculus of simplex gradients, *Optim. Lett.* 9 (2015) 845–865.

[30] R. Regis and C. Shoemaker, Constrained global optimization of expensive black box functions using radial basis functions, *J. Global Optim.* 31 (2005) 153–171.

[31] A.S. Ribeiro, J. Alves e Sousa, M.G. Cox, A.B. Forbes, L.C. Matias and L.L. Martins, Uncertainty analysis of thermal comfort parameters, *International Journal of Thermophysics* 36 (2015) 2124–2149.

[32] S. Wild and C. Shoemaker, Global convergence of radial basis function trust region derivative-free algorithms, *SIAM J. Optim.* 21 (2011) 761–781.

[33] D.H. Winfield, *Function and functional optimization by interpolation in data tables*, PhD thesis, Harvard University, 1970.

Manuscript received 6 October 2021
revised 18 January 2022
accepted for publication 7 May 2022

Warren Hare
Department of Mathematics
University of British Columbia, Okanagan Campus
Kelowna, B.C. V1V 1V7, Canada
E-mail address: warren.hare@ubc.ca

Kashvi Srivastava
Department of Mathematics
University of Michigan, Ann Arbor
Michigan 48109, U.S.A.
E-mail address: kashvi@umich.edu