OAK RIDGE
NATIONAL LABORATORY
MANAGED BY UT-BATTELLE
FOR THE DEPARTMENT OF ENERGY

ORNL/TM-2007/152

On The Reversibility of Newton-Raphson Root-Finding Method

July 8, 2008

Prepared by
Kalyan S. Perumalla
Senior Research Staff Member

UT-BATTELLE
ORNL-27 (4-00)

ORNL/TM-2007/152

Computational Sciences and Engineering Division

ON THE REVERSIBILITY OF NEWTON-RAPHSON ROOT-FINDING METHOD

Kalyan S. Perumalla
John P. Wright
Phani T. Kuruganti

Date Published: July 8, 2008

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283
Managed by
UT-BATTELLE, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

ABSTRACT

Reversibility of a computational method is the ability to execute the method forward as well as backward. Reversible computational methods are generally useful in undoing incorrect computation in a speculative execution setting designed for efficient parallel processing. Here, reversibility is explored of a common component in scientific codes, namely, the Newton-Raphson root-finding method. A reverse method is proposed that is aimed at retracing the sequence of points that are visited by the forward method during forward iterations. When given the root, along with the number of iterations, of the forward method, this reverse method is aimed at backtracking along the reverse sequence of points to finally recover the original starting point of the forward method. The operation of this reverse method is illustrated on a few example functions, serving to highlight the method's strengths and shortcomings.

1. INTRODUCTION

In speculative approaches to parallel computing, the ability to execute code both forward and in reverse is useful in improving parallel execution efficiency. While most conventional computation is limited to uni-directional (forward) execution, the enhanced capability of bi-directional (combinations of forward and reverse) execution finds application in relaxing the synchronization operations in parallel computing. Reversibility can be used in rolling back incorrect execution that is permitted to occur during speculative execution. Unfortunately, not all code is readily capable of bi-directional execution. For certain codes, memory is accumulated to remember the path of the forward execution in order to trace it back correctly in backward execution. In other codes, it might be possible to greatly reduce or eliminate the memory required to trace back the execution. However, the possibility of reduction or elimination of the memory trace is not readily evident in many computational methods. By current state of the art, a careful study of the method is needed to discover reversibility.

Here a case study is presented on the reversibility of a widely used computational method, namely, the well-known Newton-Raphson method for computing the roots of a function. The goal of this study is to investigate ways to reduce the amount of memory needed to remember the forward execution of the method in order to be able to retrace its path backwards. The operation of this reversal method is illustrated with a few examples that show its correct operation on some functions and its shortcomings on others.

The Newton-Raphson root finding method has been chosen for reversibility as an initial case study, in order to gain insights into the nature of reversibility problems, on the way towards building a more extensive set of reversible computational methods. Also, the Newton-Raphson method is extensively used in scientific computing and hence finds relevance in a range of scientific codes that could benefit from speculative parallel computing. This study is also aimed at helping develop reversals of other basic computational methods, and development of a larger set of reverse computation tools.

The remainder of the document is organized as follows: The original (forward) Newton-Raphson method is reviewed in the next section, followed by a section in which our reverse method is presented. A few illustrative case studies of the reversibility are given in the section after that, followed by a generalization of the method from scalar to vector functions. The work is summarized and concluded in the final section.

2. FORWARD NEWTON-RAPHSON

2.1 FORWARD FORMULATION

The forward problem is defined as follows: Given a function $f(x)$, and a value x_0 denoting a starting "guess point," find the root of $f(x)$ by starting at x_0 (*i.e.*, find an x for which $f(x)=0$). The Newton-Raphson method (Kelley 2003) is a well-known method to solve this problem.

Starting with x_0, the Newton-Raphson iteration proceeds by generating a sequence of values x_k until such a time that $|f(x_n)| \leq \varepsilon$ for a small ε at some $k=n$, n being the number of iterations of the forward algorithm.

2.2 FORWARD ALGORITHM

The forward algorithm finds a root by repeatedly linearizing the target function and using the root of this linearization to update the guess value. Each forward iteration, from k to $k+1$, is given by the following equation:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

where, $f'(x_k)$ is the first derivative of $f(x)$ with respect to x at $x=x_k$.

The quotient term tells how us how far we move the x value to its next value in the iteration. This value is the distance from the current point to the root of the linearization of the target function at the current point. This iteration is visualized in Fig. 1.

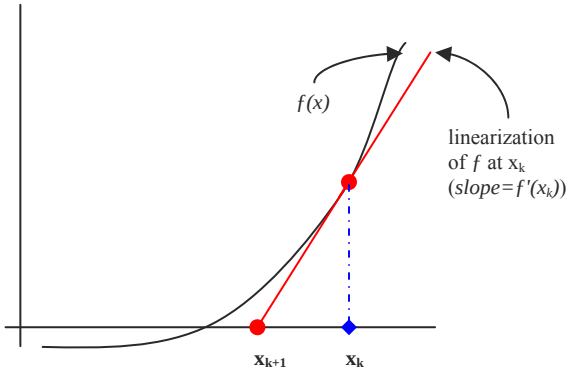

Fig. 1: Linearization in forward Newton-Raphson method

This process typically continues until the desired precision has been reached or until the algorithm exceeds a set number of iterations. If the number of iterations exceeds a small count, it is generally assumed that the algorithm has probably diverged. Exceeding a small number of iterations is understood as failure of the method or that the desired accuracy is not achievable with this method.

The importance of a good initial guess value and the shortcomings of the method in general have been studied extensively in mathematical and computational science literature and will not be discussed here.

3. REVERSE NEWTON-RAPHSON

3.1 REVERSE FORMULATION

The reverse problem is defined as follows: For a function *f(x)*, given that the forward Newton-Raphson method has generated the root x_n after n iterations, and given the number n, find x_0 which was used as the guess value in the forward method to prime the iteration. In other words, trace back the path of the forward iteration.

We will first focus on functions with scalar x, and then discuss generalization to vector x.

From this point on we will use the terms "current point" or "current value" to mean x_k corresponding to some forward iteration count k.

3.2 REVERSE ALGORITHM

The key to a successful reversal lies in understanding the mechanics of the forward algorithm. In order to reverse the forward algorithm, we want to find the point on target function whose linearization passes through the current point. We know (or have a formula for) two points that lie on this linearization. The first is the current point

$$(x_{k+1},0)$$

The second point (the previous point in the forward iteration) lies some, as yet unknown, distance from the current point:

$$(x_{k+1}-h,\, f(x_{k+1}-h))$$

where h is the unknown distance. We need to find the value for h that will bring us to the previous point in the iteration. Note that the previous point is one where a linearization of the function passes through $(x_{k+1}, 0)$. The relation of these points to the target function is shown in Fig. 2.

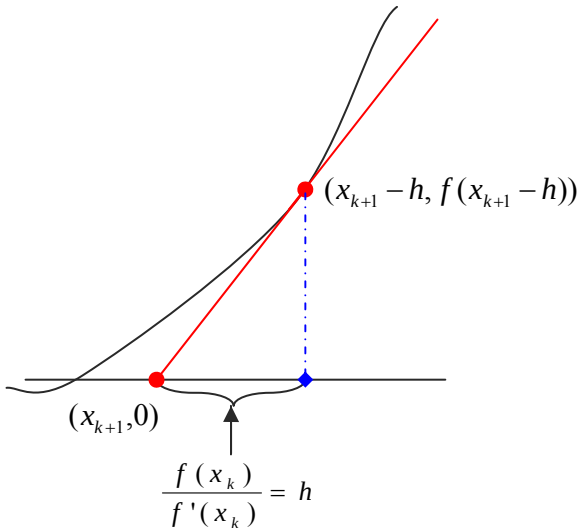

Fig. 2: Linearization for reversal of Newton-Raphson

We can use the definition of slope to find this value:

$$slope(f(x_k)) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$

We can fill in the values to find an equation which we can solve to find the value for *h*:

$$f'(x_{k+1} - h) = \frac{f(x_{k+1} - h) - 0}{(x_{k+1} - h) - x_{k+1}}$$

After simplification, the equation satisfied by the target function for the reverse method becomes

$$0 = \frac{f(x_{k+1} - h)}{-h} - f'(x_{k+1} - h)$$

The key to reversal lies in computing the correct value of h that satisfies the preceding equation. Interestingly, solving this equation represents finding the root of another function *g(h)*:

$$g(h) = \frac{f(x_{k+1} - h)}{-h} - f'(x_{k+1} - h) \qquad \text{(Equation 1)}$$

Note that the independent variable in this sub-problem for reversal is *h* (not *x*)! Since *g(h)* is evaluated at a given, known constant value of x_{k+1}, only *h* remains as the independent variable.

To find the value for *h* we are essentially presented with another root-finding problem, albeit with a different independent variable and a different function on that variable. To find the root of this new function, we can simply resort to the forward Newton-Raphson method, and use it to solve *g(h)=0* for *h*.

In this new root-finding sub-step, it is important to note that the derivative used in the Newton-Raphson method for solving *g(h)=0* is the derivative of *g(h)* with respect to *h*; all occurrences of *x* in *g()* are simply constants, taking the *x* value equal to the current point of reversal. Also, this sub-step is iteratively executed *n* number of times, equal to the number of iterations of the forward execution of the original problem. Each reverse iteration traces back the computation by one step, recovering the most recent *x* value (estimate of the root) in the ordered sequence of *x* values computed by the forward iterations.

3.2.1 Example

For example, with $f(x) = x^2$, we get

$$f'(x) = 2x,$$

$$g(h) = \frac{(x_{k+1} - h)^2}{-h} - 2(x_{k+1} - h),$$

and

$$g'(h) = \frac{2(x_{k+1} - h)}{h} + \frac{(x_{k+1} - h)^2}{h^2} + 2.$$

Thus, at any given point, x_{k+1}, *g(h)* and *g'(h)* can be defined, which can be used to invoke the forward Newton-Raphson method in order to find the root h_{root} of *g(h)* at that given x_{k+1}.

In the preceding example, it is clear that an initial guess value of *h=0* does not work, so a small non-zero value of *h* needs to be used to prime the reversal.

4. EVALUATION RESULTS AND ISSUES

Generally our reverse method works well on functions which also behave well during the forward method. Since the reverse method uses the forward method, the reversal suffers from (at least) the same inconsistencies found in the forward method.

Due to the form of target function for the reverse method, it will always have a discontinuity at $h = 0$. As a result the initial guess for the reverse method can never be 0.

First we show two successful cases of reversal. Fig. 3 shows a successful reversal for $f(x)=x^2$ and Fig. 4 shows a successful reversal for $f(x)=sin(x)$. Red crosses denote points visited on the forward method and green error bars show the difference between the forward and reverse methods (the bars are not visible denoting a successful reversal).

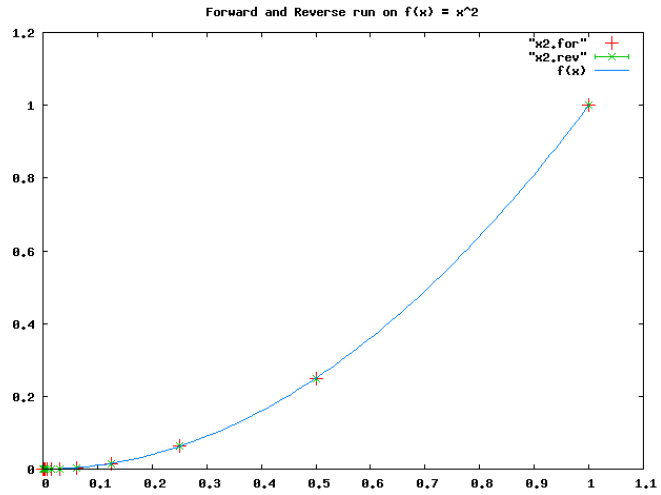

Fig. 3: Forward and reverse execution on x^2.

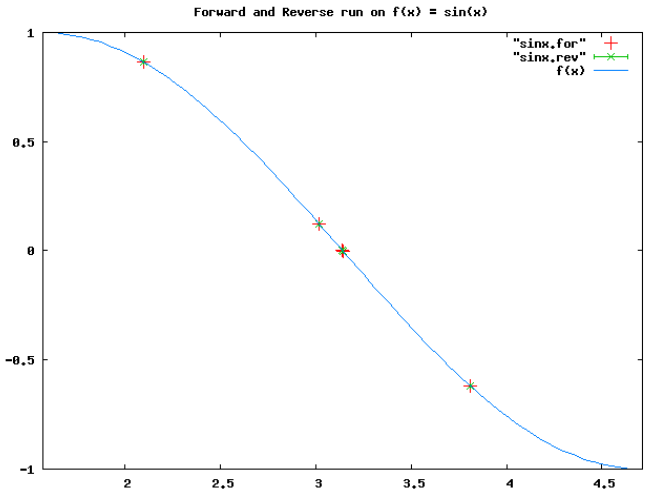

Fig. 4: Forward and reverse execution on *sin(x)*

The forward method typically has trouble with the sine function but with a good guess point the method will converge. The reverse method exhibits the same behavior.

The forward method works well on *f(x)=log(x)* and we would expect the reverse method to also work well. Fig. 5 shows how the reverse method may not find the starting value used in the forward method. Fig. 6 gives us an idea about why the reverse method failed.

The reverse method finds a point on the target function whose tangent passes through the current point. As a result, any such point may be found, not necessarily uniquely the point from which the forward execution may have arrived. In the case of *log(x)* at x ≈ 0.5 (Fig. 6), there are two possibilities. The point at which we arrive after the iteration depends on the guess provided to the root finding method.

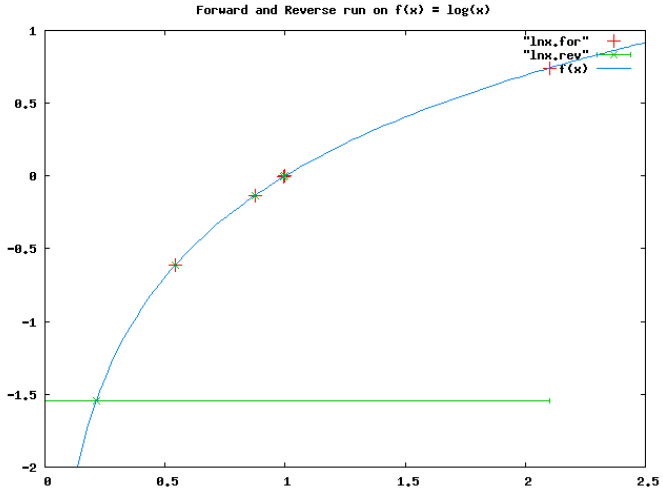

Fig. 5: Forward and reverse execution on *log(x)*

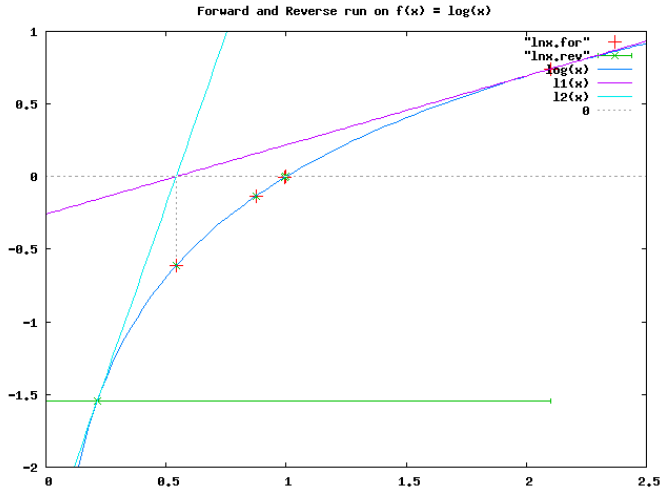

Fig. 6: Possible next steps in the reverse iteration for *log(x)*

Fig. 7 shows a more extreme case of how the reverse method can give initially unexpected results. On the domain $(-2\pi, 2\pi)$ there are three possible next steps in the reverse method for $x = 1$. Due to the oscillating nature of the sine function there are an unlimited number of valid next steps for the reverse method. Fig. 8 shows the target function used in the reverse iteration at $x = 1$. We see that the presence of multiple roots, as well as a discontinuity, presents problems with root-finding step in the reverse iteration.

Further work is needed to improve the initial value passed into each reverse-iteration to improve consistency. The target function for the reverse method may also need to be improved to minimize trouble with root-finding.

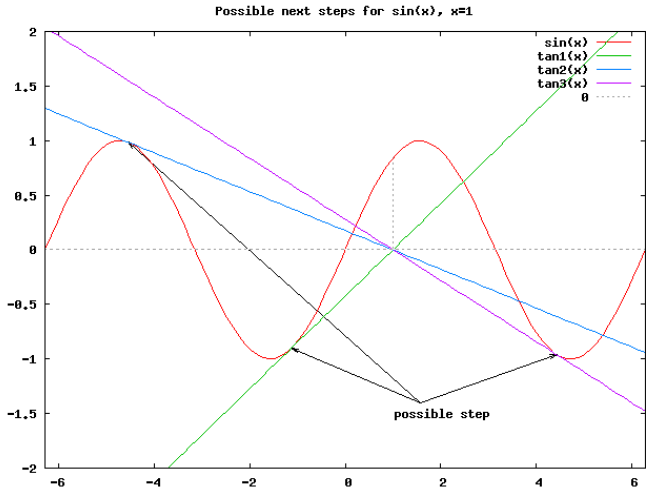

Fig. 7: Valid next steps for *sin(x)* at *x* = 1

Fig. 8 shows *g(h)* for *f(x)=sin(x)*, where

$$g(h) = \frac{sin(x_{k+1} - h)}{-h} - cos(x_{k+1} - h)\,.$$

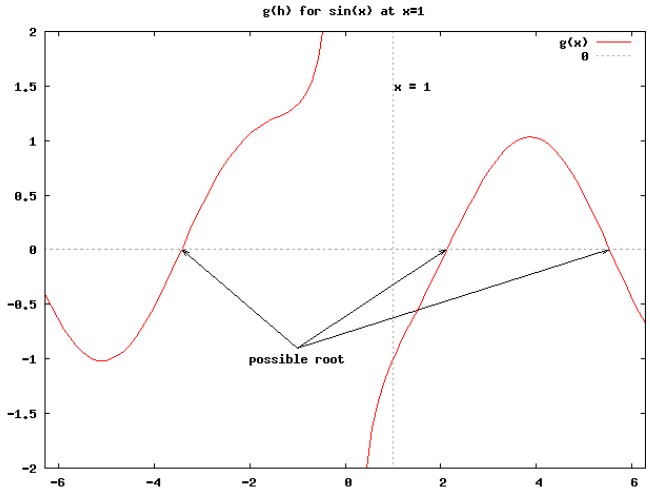

Fig. 8: Reverse target function for *sin(x)* at *x* = 1

5. VECTOR FORMULATION

We have shown how to reverse the Newton-Raphson root-finding method on *f(x)* when *x* is scalar. Let us now look at the case when *x* is a vector *X* of size *m*, and *f* is a vector function *F* (of size *m*) on *X*. This is the solution of a non-degenerate *m*×*m* system of *m* non-linear equations on *m* unknowns. We will refer to an individual i^{th} element of *X* by *X*[*i*], and the i^{th} scalar function of F by F[*i*]. The forward method can be modified to suit this vector formulation simply by replacing $f'(x_k)$ by $J_F(X_k)$, where $J_F(X_k)$ is the Jacobian matrix of *F(X)* evaluated at $X=X_k$. The Jacobian is a matrix of size *m*×*m*, where the element *i*,*j* of $J_F(X)$ is equal to $\frac{\partial F[i]}{\partial X[j]}$.

$$J_F(X_k)\cdot(X_{k+1}-X_k) = -F(X_k)$$
$$\Rightarrow J_F(X_{k+1}-H)\cdot H = -F(X_{k+1}-H)$$

5.1 EXAMPLE

Let *m*=2, and let the following be the function definitions of $f_1 \cdots f_m$:

$$f_1(X) = X[1]^2 + \log(X[2])$$
$$f_2(X) = X[1] + \sin(X[2])$$

Find *X* such that $|\ f_1(X),\ f_2(X)| = 0$, *i.e.*, a given norm (typically, 1-norm or 2-norm) of |*F(X)*| nearly vanishes.

$$J_F(X_k) = \begin{bmatrix} 2X_k[1] & \frac{1}{X_k[2]} \\ 1 & \cos(X_k[2]) \end{bmatrix}$$

$$J_F(X_k)\cdot H = -F(X_k)$$
$$\Rightarrow 2X_k[1].H[1] + \frac{H[2]}{X_k[2]} = -X_k[1]^2 - \log(X_2)$$
$$\text{and, } H[1] + H[2]\cdot\cos(X_k[2]) = -X_k[1] - \sin(X_k[2])$$

This can be written as $AH = b$, for some constant matrix *A* and vector *b*. This vector equation needs to be solved for the vector *H*. Once such a *H* is found, the *X* vector is updated for next iteration as $X_{k+1} = X_k + H$.

In order to reverse the preceding update to the vector *H*, *i.e.*, to go backward, we need to find the root H_{root} of the following vector function *G(H)*.

$$G(H) = F(X_{k+1}-H) + J_F(X_{k+1}-H)\cdot H\ .$$

For the preceding example, *G(H)* is given by:

$$G(H) = \begin{bmatrix} (X_{k+1}-H[1])^2 + \log(X_{k+1}[2]-H[2]) + 2(x_{k+1}[1]-H[1])\cdot H[1] + \frac{H[2]}{x_{k+1}[2]-H[2]} \\ x_{k+1}[1] + \sin(x_{k+1}[2]-H[2]) + H[1] + \cos(x_{k+1}[2]-H[2])\cdot H[2] \end{bmatrix},$$

using which we need to find H_{root} such that $|G(H_{root})|=0$

5.2 SCALAR FUNCTION AS SPECIAL CASE OF VECTOR FUNCTION

Based on the vector formulation, the problem of root-finding for the scalar function can be viewed as a special case of the vector function formulation†. The scalar case can be viewed as the solution of a 2×2 system, i.e., two equations with two unknowns, as follows:

$$F = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix},$$

where

$$\begin{aligned} f_1(x,y) &= f(x) \\ f_2(x,y) &= y \end{aligned},$$

giving

$$J_F = \begin{bmatrix} f'(x) & 0 \\ 0 & 1 \end{bmatrix}.$$

Setting the 2-norm of F to zero results in solution of this system to achieve the same result as the root-finding operation on the original scalar formulation of $f(x)=0$.

† Thanks to the reviewer (Dr. Protopopescu) for suggesting this clarification.

6. SUMMARY & CONCLUSIONS

A formulation of a reversible execution problem has been presented for the well-known (forward) Newton-Raphson root-finding method. An approach to the reversibility problem is presented in the form of a reverse algorithm for the forward Newton-Raphson. The operation of the algorithm has been illustrated in some cases in which Newton-Raphson may be perfectly reversed, and its shortcomings are identified in the more general case.

In the general case, the reversal may not lead uniquely to the starting point originally provided to the forward method. However, any subsequent forward execution that starts at the point given by the reversal will preserve perfect reversibility. Analysis of such self-adjusting perfect reversibility may be interpreted as another valid "plausible reversal."

The reversal method, articulated in the scalar function context, has also been generalized to vector functions.

6.1 ALTERNATIVE METHODS FOR USE AS SUB-STEP IN REVERSAL

Note that the forward Newton-Raphson method is invoked only as a sub-step in the reverse method. We do not discount the possibility that alternative reversal methods might in fact be useful. For considerations of simplicity and ease, we resort to the forward Newton-Raphson method. It is conceivable to use a different root-finding method here in the reversal, with better resilience to problems from which Newton-Raphson suffers; we leave exploration of this alternative as future work.

6.2 MEMORY SAVINGS POTENTIAL WITH PERFECT REVERSAL

A reason for finding a reversal is to reduce the amount of memory needed to remember changed state. Memory reduction is realized by remembering the number of iterations for reversal instead of the original guess value. Memory reduction will almost always occur due to the speed at which this algorithm converges. Only a few bits will be required to store the number of iterations (*e.g.*, 4 bits to represent the 10 or fewer iterations typically taken by Newton-Raphson) compared to sixty-four or more bits required to store the original guess value. Memory reductions are envisioned to be even greater on vector functions; reverse method can avoid saving the initial guess vector, resulting in savings of order N, where N is the vector length.

6.3 FUTURE WORK

Exact *vs*. plausible reversal can perhaps be traced back to the shortcomings present in the forward method itself, since the forward method is reused as a sub-component by the reverse method, albeit on a specially constructed sub-problem. Determination of a good guess value in that sub-problem remains as future work, as does a way to limit the reverse search domain to improve perfection of reversibility.

As pointed out by one of the reviewers (Dr. Protopopescu), another important aspect that needs to be explored is the complementary nature of the original forward Newton-Raphson method and our proposed reverse method. The question is on how the forward and reverse methods relate when applied alternately one after the other in a cycle of reversible execution. Specifically, given the $f(x)$ and $g(h)$ pair as specified earlier in Equation 1, one can view $g(h)=0$ as a new, forward root-finding problem and can ask if the reverse method as proposed here may be used to recover the root given by the forward execution of $f(x)$.

While, at the time of writing, obvious applications of this reversible method are not fully determined, it is envisioned that reversible execution (recovery of starting vector) in determining the

roots of a vector function appears to be useful in speculative parallel execution of non-linear models such as astrophysics simulations (Stone 2007).

ACKNOWLEDGEMENTS

The authors are grateful for a critical review by Drs. V. Protopopescu and J. Nutaro. This effort has been supported by research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES

Kelley, C. T. (2003). Solving Nonlinear Equations with Newton's Method, Society for Industrial and Applied Mathematics (SIAM).

Stone, J. M. (2007). "Computational Astrophysics." Scholarpedia Retrieved 2007/10/05, 2007, from www.scholarpedia.org/article/Computational_Astrophysics.