Automatic Differentiation for Iterative Process and Its Applications in Network Performance Analysis

by Mingyan Liu

CSHCN T.R. 97-35
(ISR T.R. 97-93)

Automatic Differentiation for Iterative Process and Its Applications in Network Performance Analysis

Mingyan Liu
enter for Satellite and Hybrid Communication Networks
University of Maryland
College Park, MD 20742 [*]

Abstract

In this paper we focus on the application of automatic differentiation (AD) technique on iterative processes. We review some of the results on the convergence of general iterative processes and the convergence of the derivative code of such iterative processes. We are especially interested in a class of fixed point iteration problems and we extended some those results to discuss this class of problems. Finally we apply an AD package ADIC to a network performance evaluation problem for numerical experiment to get sensitivities of network blocking probabilities w.r.t. network offered traffic load.

[*]This work was supported by the Center for Satellite and Hybrid Communication Networks, under NASA cooperative agreement NCC3-528

1 Introduction

The idea behind automatic differentiation (AD) is not a new one. The utility of computers for evaluating functions defined by formulas has long been recognized. And since differentiation of functions defined by formulas is a mechanical process done according to fixed rules, it is highly suitable for automation along the same lines as function evaluation. However, the technique of automatic differentiation did not become popular and really applicable until pretty recently, due to the remarkable work by Andreas Griewank and Christian Bischof, who is also the major contributer of the AD package ADIFOR for FORTRAN and the recently available ADIC (test version) for C.

However, just because the derivatives computed by automatic differentiation are those defined by the statements that were executed by a particular program run, which is what was actually computed and may differ significantly from the derivative of the function one intended to compute [1]. This is especially true in the iterative evaluation of a function defined implicitly or otherwise. Usually the iteration continues until the value of $f(x)$ meet certain criteria. However, this may not be true of the value of $f^{'}(x)$ or higher derivatives. On the other hand, many engineering problems are impossible or impractical to be expressed in an explicit or exact form, and we have to turn to approximations which often take an iterative form. The problems lie in the design of programs to which AD is to be applied and can be handled most effectively by the programmer, especially for pitfalls arising from branching or iteration.

This paper will be focusing on approaching iterative processes, especially fixed point problems, using automatic differentiation. It is organized as follows: Section 2 summarizes the findings and results of [2] on convergence of derivatives of functions defined implicitly or iteratively, and its implication on design of good AD packages. Section 3 discusses a proposed automatic differentiation procedure for fixed point problems by [3], which is also partly based on the result of [2]. Section 4 discusses cases of two or more fixed points. In Section 5, the AD package ADIC is applied to a fixed point algorithm, which is developed to approximate the blocking probabilities in a network, to get the sensitivities of blocking probabilities in respect to traffic load. The efficiency of ADIC is also evaluated in that sense. Section 6 is conclusion of the paper.

2 Derivative Convergence of Implicit Functions

Consider a system of nonlinear equations, where dependent variable $y \in \mathcal{R}^n$ and independent variable $x \in \mathcal{R}^m$ are implicitly defined by $R(y, x) = 0$, $R : \mathcal{R}^n \times \mathcal{R}^m \to \mathcal{R}$. Suppose given x_*, we want to calculate y_* such that $R(y_*, x_*) = 0$. The iterative process employed to solve this problem is as follows:

```
Given $y_0$,

Until $\|R(y_k, x_*)\| \leq Tol_R$ or $\|y_k - y_{k-1}\| \leq Tol_y$

    Compute some preconditioner $P_k$
    $y_{k+1} = y_k - P_k R(y_k, x_*)$            (1)

    $k = k + 1$
```

Where the choice of preconditioner P_k determines the type of method the iterative process uses, e.g., Newton's method with $P_k = R_y(y_k, x_*)^{-1}$.

Now suppose we want to compute $y^{'} = \frac{\partial y}{\partial x}$, then the application of automatic differentiation to the previous iteration results in the following code:

```
Given $y_0$,

Until $\|R(y_k, x_*)\| \leq Tol_R$ or $\|y_k - y_{k-1}\| \leq Tol_y$

    Compute $P_k$ and $P_k^{'} = \frac{\partial P_k}{\partial x}$

    $y_{k+1} = y_k - P_k R_k$

    $y_{k+1}^{'} = y_k^{'} - P_k R_k^{'} - P_k^{'} R_k$

    $k = k + 1$
```

Now we have the following questions:

1. Does $y_k^{'}$ converge assuming y_k does?

2. If $y_k^{'}$ does converge, at what rate? If it does not converge as fast as y_k, then it is clear we need a separate stopping criteria for $y_k^{'}$, which may make the whole process not automatic anymore.

Also, since we have for implicit function

$$R_x^{'}(y_*, x_*) = \frac{\partial R}{\partial y} \cdot y_x^{'} + \frac{\partial R}{\partial x}, \tag{2}$$

i.e., only the first-order derivatives are needed to determine $y_x^{'} = \frac{\partial y}{\partial x}|_{x=x_*}$. Since $R_x^{'}$ can be evaluated cheaply (at approximately the same cost of evaluating R itself), the computation of $P_k^{'}$ could be possibly computationally dominant.

Adopting two assumptions, namely, the *regularity* assumption and the *contractivity* assumption, [2] studied these questions and proved that given y_k converges to y_* in a reasonably rapid and stable fashion, $y_k' \to y_*'$ for a large class of iterations, with or without P_k', which is very important because we know that the evaluation of P_k' is computationally dominant. More specifically, it is shown that for Newton's method and for secant updating methods, the derivatives converge R-quadratically and R-linearly, respectively. Also, for a large class of memoryless contractions, where P_k does not depend on previous iterates, derivative convergence can be achieved with an R-linear or possibly R-superlinear rate.

Secondly, the convergence of y_k' is generally expected to lag behind the convergence of y_k so that an augmented stopping criteria for y_k' is necessary. [2] also provided a constructive stopping criterion for the derivative iteration by bounding the derivative errors (Lemma 1).

Numerical experiments were also given in [2] on small test examples, which confirmed the theoretical results.

3 Function Evaluation Involving a Fixed Point

[3] examined a class of functions which include in their computation a convergent iterative process of the form $y = \Phi(y, u)$, $\Phi : \mathcal{R}^n \times \mathcal{R}^m \to \mathcal{R}^n$, where $u : \mathcal{R}^k \to \mathcal{R}^m$ is some function of independent variable $x \in \mathcal{R}^k$. Fixed point y_* is the convergent result of Φ and the final dependent variable $z = f(x, y)$, $f : \mathcal{R}^k \times \mathcal{R}^n \to \mathcal{R}^h$.

It can be illustrated as:

```
Given $x_*$ and $u_* = u(x_*)$,
Until $\|y_k - y_{k-1}\| \leq Tol_y$
    $y_{k+1} = \Phi(y_k, u_*)$
    $k = k + 1$
$z_* = f(y_*, x_*)$
```

[3] approached this class of functions relatively independent from [2], however, we can actually establish a relationship between the two in the following way. Let $R(y, u_*) = \Phi(y, u_*) - y = 0$ be the implicit function, thus we have the update

$$y_{k+1} = \Phi(y_k, u_*) = y_k + R(y_k, u_*). \qquad (3)$$

Choosing $P_k = -I$, the above equation becomes

$$y_{k+1} = y_k - P_k R(y_k, u_*), \qquad (4)$$

which is just equation (1) in Section 2. We see that the fixed point problem is actually a subset of the problems discussed in Section 2.

For $y_k^{'}$ to converge, according to [2]'s contractivity assumption, the above iteration is a memoryless contraction if

$$D_k = [I - P_k R_y(y_k, u_*)] \quad \text{satisfy} \quad \delta_k \equiv \|D_k\| \leq \delta < 1 \tag{5}$$

with respect to some induced matrix norm so that in the limit

$$\delta_* \equiv \lim_k \delta_k \leq \delta.$$

From (5),

$$P_k = -I \Longrightarrow D_k = [I + R_y(y_k, u_*)] = \Phi_y(y_k, u_*),$$

Equivalent to the contractivity assumption, the convergent requirement that

$$\|\Phi_y(y_k, u_*)\| < \tau < 1 \tag{6}$$

is examined in [3], described as y_* being an *attractive* fixed point of Φ if the above holds.

Finally, $z_x^{'}$ is calculated by using chain rule:

$$z_x^{'} = \frac{\partial f}{\partial y} \cdot y_*^{'} + \frac{\partial f}{\partial u} \cdot u_*^{'}.$$

[3] derived convergence theorems from the definition of a *well behaved iterative constructor* Φ and also proposed an implementation strategy for this particular type of functions. The implementation used reverse accumulation of automatic differentiation, and the basic idea is to switch on and off the graph construction of u, Φ and f accordingly. Error estimates and stopping criterion for derivative were also given in [3].

Here we would like to summarize the stopping criteria proposed by both papers.

[2] has the following result: For $R(y, x) = 0$, denote

$$\rho_k \equiv \|y_k - y_*\| \quad \text{and} \quad \mu_k \equiv \|y_k^{'} - y_*^{'}\|$$

and set

$$\eta_k \equiv (Lc_1 + \|P_k^{'}\|)\rho_k \quad \text{with} \quad c_1 \equiv 2(c_0^2 + 1),$$

where L and c_0 are constants, following the regularity and contractivity assumption, it can be shown that

$$\mu_k \leq \frac{1}{(1-\delta)} \|P_k(R_y(y_k, x) y_k^{'} + R_x(y_k, x))\| + \frac{1}{2} L c_0 c_1 \rho_k, \tag{7}$$

$$\mu_{k+1} \leq \delta_k \mu_k + c_0 \eta_k, \quad \text{and} \quad \mathcal{O}(\|y_k - y_*\|) \leq c_1 c_0 L \rho_k, \tag{8}$$

for all $\rho_k < \rho$.

(7) provides us with a stopping criterion for the derivative iteration if we can make some reasonable estimates on L, c_0 and δ.

In [3], a stopping criterion is provided in terms of the desired accuracy of $\xi\|r\|$, where $\xi < 1$ and r is a fixed arbitrary row vector:

$$\|y_{k+1} - y_k\| < \frac{\xi(1-\tau)^3}{2C(1-\tau_k)}, \tag{9}$$

and

$$\|y_k^{'} + r - y_{k+1}^{'}\| < \frac{\xi(1-\tau)}{2k} \cdot \|r\|, \tag{10}$$

where τ and k are bounds for $\|\Phi_y\|$ and $\|\Phi_u\|$, respectively, in a neighborhood of $(y_*(u), u)$, and C is the Lipschitz constant for the map $\Phi^{'}$ in this neighborhood.

4 Multiple Fixed Points

In this section we consider the case where there are two fixed points (y_*, u_*) (both of them can be vectors) defined as follows:

$$y = f_1(x, u), \quad f_1 : \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{R}^l$$

$$u = f_2(x, y), \quad f_2 : \mathcal{R}^m \times \mathcal{R}^l \to \mathcal{R}^n$$

where x is the independent variable. This is of interested to us because in many engineering instances, the system operating parameters are unknown and can only be calculated through establishing iterative process among them. The example in next section is of this type.

Suppose given x_*, $y \to y_*$ and $u \to u_*$ in the following way:

```
Given $x_*$ and $u_0$,

Until $\|y_k - y_{k-1}\| \le Tol_y$ or $\|u_k - u_{k-1}\| \le Tol_u$

    $y_k = f_1(x_*, u_k)$

    $u_{k+1} = f_2(x_*, y_k)$

    $k = k + 1$

$z_* = f(y_*, u_*)$
```

Similarly to what we did in Section 3, let

$$R(x_*, u) = f_2(x_*, f_1(x_*, u))$$

be the implicit function, so the update becomes:

$$u_{k+1} = R(x_*, u_k) = u_k - P_k R(x_*, u_k),$$

where the $n \times n$ preconditioner $P_k = Q_k - I$, and $Q_k R(x_*, u_k) = u_k$, $Q_k R_u + Q_k^{'} R = I$.

Applying the contractivity assumption, the discrepancies

$$\begin{aligned} D_k &= [I - P_k R_u(x_*, u_k)] \\ &= I - (Q_k - I) R_u(x_*, u_k) \\ &= I - Q_k R_u(x_*, u_k) + R_u(x_*, u_k) \\ &= Q_k^{'} R + R_u \end{aligned}$$

It is not the intention of this paper to develop rigorous mathematical proof here. However, we can easily see that this, again, is a subset of the problems discussed in [2], and by using suitable argument and conditioning on $Q_k^{'} R + R_u$, it can be shown that $u_x^{'}$ converges. Similarly, $y_x^{'}$ converges. By using the chain rule, we have

$$z_x^{'} = f_u^{'} u_x^{'} + f_x^{'} y_x^{'}.$$

To summarize, ideally we would want the derivative evaluation code generated by an AD preprocessor to take the form of the following:

```
Given $x_*$, $u_0$ and $u_0^{'}$,
Until $\|y_k - y_{k-1}\| \le Tol_y$ or $\|u_k - u_{k-1}\| \le Tol_u$
Until $\|y_k^{'} - y_{k-1}^{'}\| \le Tol_{y'}$ or $\|u_k^{'} - u_{k-1}^{'}\| \le Tol_{u'}$
    $y_k = f_1(x_*, u_k)$
    $y_k^{'} = \frac{\partial f_1}{\partial x} + \frac{\partial f_1}{\partial u} \cdot u_k^{'}$
    $u_{k+1} = f_2(x_*, y_k)$
    $u_{k+1}^{'} = \frac{\partial f_2}{\partial x} + \frac{\partial f_2}{\partial y} \cdot y_k^{'}$
    $k = k + 1$
$z_* = f(y_*, u_*)$
```

$$z_x' = f_u' u_x' + f_x' y_x'$$

Similar argument can be applied to problems involving more than two fixed-points.

5 Application in Network Performance Evaluation

In this example, iteration takes the form of what is described in Section 4.

Consider a loss network, which is basically a circuit-switched network, where a call requires certain amount of bandwidth on every link on a path between the source and the destination. If the network has the required bandwidth on those links when it gets the request, the call is admitted and it will be using the requested capacities for some time; otherwise the call is rejected. The performance metric here of interest is the blocking probability which is the probability that a call finds the network unavailable when it arrives and is thus rejected.

Because an exact form of this blocking probability is generally unavailable due to the size of network, number of traffic types, traffic pattern, etc., various approximation schemes have been studied. One popular and quite efficient way is called the fixed point or reduced load approximation method. The idea is to consider the traffic load on each single link and the blocking on each single link as two set of unknowns (we use set here because load and blocking are both further classified according to traffic type and source-destination pair) of the network. If, under certain reasonable assumptions, we can express the traffic load on a single link in terms of the original load and the blocking on other links – the load is reduced by blocking on other links, and express the blocking on a single link in terms of traffic load, then by continuous substitution, hopefully the problem can converge and we can solve for both sets of unknown parameters, which we assume to be the equilibrium operating point of the network under stable condition. And the final blocking probability should be a function of blocking on individual links and the traffic load as well.

So the problem formulation falls under the category discussed in Section 4. Let a denote the vector of link admissibility probabilities, ν denote the vector of link traffic load, and x be the set of independent variables like the distribution of traffic load, network resource allocation (link capacities), or admission control parameters. We have two approximations:

$$\nu = f_1(a, x) \quad \text{and} \quad a = f_1(\nu, x)$$

the fixed point (a_*, ν_*) and the blocking probabilities $B = B(a_*, \nu_*)$. This is a very brief sketch of the algorithm, details can be found in [4].

Naturally we are interested in $\frac{\partial B}{\partial x}$, the sensitivity of blocking probability with respect to network design parameters, and eventually we would like to solve the following optimization problem:

$$min \quad f(B), \quad \text{such that} \quad g(x) \geq 0,$$

where $f(\cdot)$ is some costpenalty function and $g(\cdot)$ is the restriction on network designs.

In this experiment, the AD package ADIC [5] is chosen to generate derivative code to calculate $\frac{\partial B}{\partial x}$. There is no separate stopping criterion implemented for derivative iteration. The whole iteration is terminated when the function convergence criterion is satisfied as illustrated in Section 2. So stopping criterion for derivatives had to be manually added.

We used a five-node fully connected network, with three different classes of traffic for the experiment. Details on this network can be found in [4]. The computation results are quite satisfying, and correspond to previous discussions, $B_x^{'}$ did converge but was slower than B itself, 18 iterations vs. 11 iterations.

6 Conclusion

This paper focuses on application of automatic differentiation technique on iterative processes. Results from [2] and [3] were discussed and evaluated regarding convergence of general iterative processes and additional stopping criteria which should be implemented in existing AD packages. Some extension of these results were made to discuss a class of fixed point iteration commonly encountered in network performance evaluation and numerical experiment were made by using the preprocessor ADIC.

References

[1] Louis B. Rall and George F. Corliss. An introduction to automatic differentiation. pages 1–18, 1990.

[2] Andreas Griewank, Christian Bischof, George Corliss, Alan Carle, and Karen Williamson. Derivative convergence for iterative equation solvers. *Optimization Methods and Software*, 2:321–355, 1993.

[3] Bruce Dhristianson. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3:311–326, 1994.

[4] M. Liu, A. Misra, and J. S. Baras. Performance evaluation in multi-rate multi-hop communication network with adaptive routing. *Proc. Annual ATIRP Conferrence*, 2, 1998.

[5] Christian Bischof and Lucas Roh. *User's Guide to ADIC 1.0 Revision 1.0 Beta 1*, 1997.