Application of AD based Quasi-Newton-Methods on stiff ODEs

Sebastian Schlenkrich and Andrea Walther
Institute of Scientific Computing
Technische Universität Dresden
Germany
[schlenk,awalther]@math.tu-dresden.de

Andreas Griewank
Institute of Mathematics
Humboldt University Berlin
Germany
griewank@mathematik.hu-berlin.de

9. April 2004

1 Introduction

For the integration of stiff ordinary differential equations (ODEs), it is advantageous to employ implicit methods. Hence, one has to solve a system of mostly nonlinear equations in each time step. For that purpose, Newtons method can be applied if the inverse of the full Jacobian is available. Since computing the inverse of the Jacobian is usually quite costly, we want to apply a Quasi-Newton method that provides a factorized approximation of the Jacobian. The Quasi-Newton update that we propose here, makes explicit use of tangent and adjoint information which can be obtained using methods of algorithmic differentiation. Furthermore, we elaborate the conditions for invariance with respect to similarity transformations of the state space. The resulting update formulas are implemented using C as programming language and the AD tool ADOL-C for providing the required derivatives. Numerical results for Runge-Kutta methods and linear multistep methods will be discussed.

2 Implicit methods for ODEs

Suppose that we want to compute a solution of the initial value problem (IVP)

$$\dot{x}(t) = f(x(t)) \quad t \in (0, T) \qquad x(0) = \eta \in \mathbb{R}^n,$$

where $x(t)$ denotes the state variable. Equidistant discretization of the time interval $[0, T]$ using the step size h yields the discrete solution

$$x_k = x_{k-1} + h\Phi(x_k, x_{k-1}, \ldots, x_{k-l}, h) \qquad x_0 = \eta.$$

Here, $\Phi = \Phi(x_k, \ldots)$ stands for the step function of the integration method. If Φ depends also on x_k, which has to be calculated, the integration method is called implicit [3]. Therefore, the new state x_k is obtained by solving the n-dimensional system of equations

$$F_k(x_k) := x_k - x_{k-1} - h\Phi(x_k, x_{k-1}, \ldots, x_{k-l}, h) = 0 \in \mathbb{R}^n \tag{1}$$

Since f is usually nonlinear, the system to be solved is also nonlinear. Hence, one may apply iterative methods, for example Newtons method or a Quasi-Newton approach. Then, an iteration of the form

$$x^{(i+1)} = x^{(i)} - A_i^{-1} F_k(x^{(i)}) \tag{2}$$

is performed with $A_i \approx F_k'(x^{(i)})$. Naturally the sequence $\{x^{(i)}\}_{i \in \mathbb{N}}$ should converge to the solution x_k of (1).

3 Evaluating A_i

Applying Newtons method to the system (1), the full Jacobian of F is required for each time step. Additionally, one has to factorize the Jacobian to solve the linear system. For large dimensions n or if f respectively F is given implicitly the provision of the Jacobian is difficult since it is not available explicitly or computationally very expansive.

An alternative idea is to use information on F from previous iterations and update an approximation of the Jacobian. For this purpose, one may use rank-1-updates. Then the approximation A_{i+1} of the Jacobian at x_{i+1} is derived by

$$A_{i+1} = A_i + uv^T$$

With the capabilities of algorithmic differentiation the directions u and v can be chosen in a way such that the new approximation A_{i+1} satisfies a direct and an adjoint tangent condition, namely

$$A_{i+1} s_i = F_k'(x^{(i+1)}) s_i \quad \text{and} \quad z_i^T A_{i+1} = z_i^T F_k'(x^{(i+1)})$$

for certain directions s_i and z_i. The corresponding update is given by

$$A_{i+1} = A_i + \frac{(F_k'(x^{(i+1)}) s_i - A_i s_i)(z_i^T F_k'(x^{(i+1)}) - z_i^T A_i)}{(z_i^T F_k'(x^{(i+1)}) - z_i^T A_i) s_i}.$$

This formula is referred to as *Twosided Rank-1 (TR1)* update and already exploited in the context of nonlinear optimization [2].

To integrate stiff ODEs, special attention will be given to the choice of the directions s_i and z_i. In accordance to secant methods, s_i is the direction of the previous step $s_i = x^{(i+1)} - x^{(i)}$, whereas we will discuss two alternatives for choosing the adjoint direction z_i.

To motivate the choice of z_i, we employ the linear Model

$$M(x) := F_k(x^{(i+1)}) + A_{i+1}(x - x_{i+1})$$

of F in x^{i+1}. The first approach refers to a minimization problem corresponding to (1). With $J(x) := \|F_k(x)\|_2^2$ it is given by

$$J(x^*) \to min \qquad \Leftrightarrow \qquad F_k(x^*) = 0$$

We suppose that the gradient of $J(x^{(i+1)})$ provides a decent direction. Then the gradient of the corresponding minimization problem $\tilde{J}(x) := \|M(x)\|_2^2 \to min$ of the model M should be the same in $x^{(i+1)}$. This yields the condition

$$\nabla J(x^{(i+1)}) = \nabla \tilde{J}(x^{(i+1)}) \qquad \Leftrightarrow \qquad z_i = F_k(x^{(i+1)})$$

and we call the resulting formula for A_{i+1} *Least-Squares-Update*.

A favored property of the iterative method (2) would be the independence with respect to similarity transformations in the state space. Hence, the iteration (2) should yield $\tilde{x}_i = T x_i \quad \forall i$ when a linear transformation $\tilde{x} = Tx$ is applied to the state. The last equality is valid for the TR1 method if

$$\tilde{A}_i = T A_i T^{-1} \qquad \Leftrightarrow \qquad \tilde{s}_i = T s_i \quad \text{and} \quad \tilde{z}_i^T = z_i^T T^{-1} \tag{3}$$

For the direction $s_i = x^{(i+1)} - x^{(i)}$, condition (3) is naturally fulfilled. To scaling invariance also for the adjoint direction z_i, one can make use of a problem dependent functional $\phi(x)$. Then the adjoint direction $z_i = \lambda^{(i+1)} - \lambda^{(i)}$ can be computed by the Quasi-Newton iteration for solving the adjoint system

$$G_x(\lambda) := \lambda^T F_k'(x) - \nabla\phi(x)^T = 0.$$

This second approach is called *Adjoint Update*. Because the functional ϕ relates to the problem, it depends on the state, too. Therefore, a transformation of x forces a consistent transformation of ϕ which ensures the transformation invariance.

4 Implementation and numerical results

The TR1 update requires the evaluation of $F_k'(x^{(i+1)})s_i$ and $z_i^T F_k'(x^{(i+1)})$. These can be calculated with the scalar forward and reverse mode of AD. Since the implementation of the methods are done using C/C++ we used the tool ADOL-C [1]. This provides derivatives by an internal function representation and operator overloading.

To solve initial value problems, several methods were used: Implicit Euler method, the 3-stage Radau IIA method and BDF formulas [3]. For the first method, the system to be solved is rather straight forward $F_k(x) = x - x_{k-1} - hf(x) = 0$. The approximation to the solution is of order 1. Using the 3-stage Runge-Kutta method Radau IIA means solving a system of dimension $3n$. Hence, the complexity increases, but the method is of order 5. The BDF formulas correspond to linear multi step methods, where the system of equations is given by

$$F_k(x) = \alpha_0 x - \sum_{j=1}^{l} \alpha_j x_{k-j} - hf(x) = 0$$

with certain scalars α_j. This method is of order l.

For the numerical tests, we take two initial value problems from the *Testset for Initial Value Problem Solvers*, University of Bary, Italy [4], namely the *Pollution Problem* and the *Medical Akzo Nobel Problem.* The first one describes a chemical reaction model of an air pollution model. Its dimension is 21. The latter comes from a semidiscretized PDE concerning the penetration of radio-labeled antibodies into a tissue that has been infected by a tumor. The dimension of this model is 401.

For comparison the tests were also performed using the Newton method with AD based Jacobians and the First Broyden update which is a secant method using only information of F. A comprehensive discussion of the achieved iteration counts for the TR1 update and the two other solution approaches will be contained in the full paper.

5 Conclusions and Outlook

Integrating stiff ODEs using AD based Quasi-Newton methods promises a reduction of computational effort. The selective choice of tangents and adjoints facilitates invariance and therefore norm independence of the state space.

Further development has to be done in stabilizing the iterations to improve the convergence behavior. Moreover the influence of different functionals ϕ should be examined.

Literatur

[1] A. Griewank, D. Juedes, and J. Utke: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. TOMS **22**, 131–167, 1996.

[2] A. Griewank and A. Walther: On constrained optimization by adjoint based quasi-Newton methods. Opt. Meth. and Soft. 17, pp. 869–889 (2002).

[3] E. Hairer and G. Wanner: Solving Ordinary Differential Equations II. Springer-Verlag, 1996.

[4] http://www.dm.uniba.it/~testset.