1

Application of AD-based Quasi-Newton Methods to Stiff ODEs *

Sebastian Schlenkrich[1] and Andrea Walther[1] and Andreas Griewank[2]

[1] Technische Universität Dresden, Institute for Scientific Computing, Dresden, Germany
[2] Humboldt Universität Berlin, Department of Mathematics, Berlin, Germany

Summary. Systems of stiff ordinary differential equations (ODEs) can be integrated properly only by implicit methods. For that purpose, one usually has to solve a system of nonlinear equations at each time step. This system of equations may be solved by variants of Newton's method. Here, the main computing effort lies in forming and factoring the Jacobian or a suitable approximation to it.

In this paper, we examine a new approach of constructing an appropriate quasi-Newton approximation for solving stiff ODEs. The method makes for the first time explicit use of tangent and adjoint information that can be obtained using the forward and the reverse mode of algorithmic differentiation (AD). We elaborate the conditions for invariance with respect to linear transformations of the state space and thus similarity transformations of the Jacobian. One new updating variant that yields such an invariant method is presented. Numerical results for Runge-Kutta methods and linear multi-step methods are discussed.

Key words: quasi-Newton, stiff ODE, adjoint-based update, scaling invariance

1.1 Introduction

For many time-dependent simulations, the underlying system can be modelled as the solution of an initial value problem (IVP)

$$\dot{x}(t) = f(x(t)) \quad t \in (0,T) \qquad x(0) = \eta \in \mathbb{R}^n \tag{1.1}$$

on a given time interval $[0,T]$, where $x(t)$ denotes the state variable. To compute a numerical approximation of the solution x, we perform a discretization $\{t_0,\ldots,t_N\}$ of the time interval $[0,T]$ using the step size $h_k = t_k - t_{k-1}$. Applying a numerical integration method yields the discrete solution vectors

* Partially supported by the DFG Research Center MATHEON "Mathematics for Key Technologies" in Berlin

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

at the times t_k. Here, $\Phi = \Phi(x_k, \ldots)$ represents the step function of the integration method. If Φ does not depend on the so far unknown x_k, the integration method is called explicit and is well suited for a wide range of ordinary differential equations (ODEs).

However, as soon as the underlying problem is described by stiff ODEs, for example due to very different time scales (see [HW91]), it is advantageous to employ implicit methods in order to allow a reasonable size of the time steps. Then, the step function Φ also depends on the unknown value x_k. Therefore, the new state x_k can be obtained by solving the n-dimensional system of equations

$$F_k(x_k) := x_k - x_{k-1} - h_k \Phi(x_k, x_{k-1}, \ldots, x_{k-l}, h_k) = 0 \in \mathbb{R}^n. \tag{1.2}$$

Since f is usually nonlinear, the system (2) is often also nonlinear, although there are classes of linearly implicit ODE solvers. To compute the next state x_k, one may apply an iterative Krylov method. As an alternative, one may solve a system of mostly nonlinear equations in each integration step using an iteration of the form

$$x^{(i+1)} = x^{(i)} - A_i^{-1} F_k(x^{(i)}), \tag{1.3}$$

where the sequence $\{x^{(i)}\}_{i \in \mathbb{N}}$ should converge to the solution $x^* = x_k$ of (2) for given values of $x_{k-1}, \ldots, x_{k-l}$. For that purpose, Newton's method can be applied by setting $A_i = F'(x_k)$ if the complete Jacobian is available at a reasonable cost. Since factoring the Jacobian at each time step is usually quite expensive, we present here a new adjoint-based quasi-Newton method that provides a factorized approximation of the Jacobian. First studies in this direction were made by Brown et al. [BHW85] for Broyden's method. However, the usage of quasi-Newton methods is not widespread, which may be explained by the following two reasons. First, the quasi-Newton methods proposed up to now were not scaling invariant. Hence, a simple scaling of the variables may strongly effect the convergence behaviour. Second, so far it is not possible to perform adaptive time stepping with a cost that is quadratic in the dimension of the problem, because the factorization that is updated can not be adapted to the new time step in a cheap fashion. Hence, one has to perform a new QR-factorization as soon as the time step changes. Therefore, the most well-known packages for the integration of stiff ODEs, as for example DASPK [BHP94] and CVODE [CH96], only provide several variants of Newton's method as direct methods or Krylov methods as indirect variants but no low-rank updating approach.

The quasi-Newton updates that we propose in this paper make for the first time explicit use of tangent and adjoint information that can be obtained using methods of algorithmic differentiation (AD). One of them has the property of scaling invariance to overcome the first problem of quasi-Newton updates in

the context of stiff ODEs. Future work will be dedicated to a new factorization procedure that allows also a comparatively cheap change of the time step size. These two ingredients would form a powerful combination that should allow a more extensive use of quasi-Newton methods for the integration of stiff ODEs.

The paper has the following structure. The new quasi-Newton updates are presented in Section 2. Furthermore, we elaborate the conditions for invariance with respect to similarity transformations of the Jacobian. The resulting update formulas are implemented using C/C++ as the programming language and the AD-tool ADOL-C for providing the required derivatives. Implementation details will be described in Section 3. The numerical results obtained for two Runge-Kutta methods and two BDF methods will be discussed in Section 4. Finally, a summary as well as an outlook of future work are given in Section 5.

1.2 Quasi-Newton Approximations

Applying Newton's method to the system (2), the complete 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, the derivative information $F'(x_i)$ can be computed within machine accuracy using AD but the computational complexity may grow linearly in n, for example if the Jacobian is dense. Together with the cubic effort required for the factorization this cost is often not acceptable.

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

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

with two vectors u and $v \in \mathbb{R}^n$ to be determined in the remainder of the section. For almost all proposed quasi-Newton methods so far, the two vectors are chosen such that the direct tangent condition

$$A_{i+1}s_i = F_k'(x^{(i+1)})s_i \tag{1.4}$$

or the secant condition

$$A_{i+1}s_i = F_k(x^{(i+1)}) - F_k(x^{(i)}) \equiv y_i \tag{1.5}$$

is fulfilled with $s_i = x^{i+1} - x^i$. Since the forward mode of AD provides the information $F_k'(x^{(i+1)})s_i$ at a very moderate cost, we will use the exact direct tangent condition (4) throughout this paper. The secant condition (5) is used for example in Broyden's method given by

$$A_{i+1} = A_i + \frac{(y_i - A_i s_i) s_i^T}{s_i^T s_i}.$$

Applying the reverse mode of AD, one can evaluate the product $z_i^T F_k'(x^{(i+1)})$ for a given vector z_i also at a moderate cost. In the context of solving nonlinear equations using quasi-Newton methods, this property yields the adjoint tangent condition given

$$z_i^T A_{i+1} = z_i^T F_k'(x^{(i+1)}). \tag{1.6}$$

Provided $z_i^T A_i s_i \neq z_i^T F_k'(x^{(i+1)}) s_i$ the two tangent conditions (4) and (5) are consistent, and there is exactly one rank-1 update of A_i satisfying them, namely

$$A_{i+1} = A_i + \frac{(F_k'(x^{(i+1)}) s_i - A_i s_i)(z_i^T F_k'(x^{(i+1)}) - z_i^T A_i)}{(z_i^T F_k'(x^{(i+1)}) - z_i^T A_i) s_i}. \tag{1.7}$$

This formula is referred to as *Two-sided Rank-1 (TR1)* update and has already been exploited in the context of nonlinear optimization [GW02]. Whereas we choose naturally $s_i = x^{(i+1)} - x^{(i)}$ the question remains how we select the adjoint directions z_i. However, for integrating stiff ODEs, there is no obvious choice for the weight vector z_i appearing in the adjoint tangent condition. This situation differs significantly from the nonlinear optimization context where the adjoint weight vectors can be defined as corrections of the Lagrange multipliers in a natural way. Since this is not possible for the pure integration of stiff ODEs, we will discuss two alternatives for choosing the adjoint weight vector z_i. Nevertheless, this setting will be completely changed if the ODEs form equality constraints in an optimal control setting. Then, once more, the adjoint weight vector can be defined on the base of Lagrange multipliers.

Least-squares approach

To motivate the definition 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}. Then, the first approach refers to a minimization problem corresponding to (2). With $J(x) := \|F_k(x)\|_2^2$ it is given by

$$J(x^*) = \min_x J(x) \qquad \Longleftrightarrow \qquad F_k(x^*) = 0.$$

We suppose that the gradient of $J(x^{(i+1)})$ provides a decent direction. Then the gradient of the 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

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

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

Scaling invariance

A favored property of the iterative method (3) would be the independence with respect to linear transformations in the state space of the ODE. Suppose additionally to (1) one has a transformed IVP

$$\dot{\tilde{x}}(t) = \tilde{f}(\tilde{x}(t)) \quad t \in (0,T) \qquad \tilde{x}(0) = \tilde{\eta} \in \mathbb{R}^n. \tag{1.8}$$

This is related to the original problem by a linear transformation of the state space with a regular $T \in \mathbb{R}^{n\times n}$ such that,

$$\tilde{f}(\tilde{x}(t)) = Tf(T^{-1}\tilde{x}(t)) \quad t \in (0,T) \quad \text{and} \quad \tilde{\eta} = T\eta. \tag{1.9}$$

Then for the solutions of the original problem (1) and the transformed problem (8) $\tilde{x}(t) = Tx(t)$ is valid for $t \in (0,T)$.

For the implicit Euler's method, transformation (9) yields with $\tilde{x} = Tx$

$$\begin{aligned}\tilde{F}_k(\tilde{x}) &= \tilde{x} - \tilde{x}_{k-1} - h\tilde{f}(\tilde{x}) = Tx - Tx_{k-1} - hTf(x)\\ &= TF_k(T^{-1}\tilde{x}).\end{aligned}$$

This holds in a similar way for Runge-Kutta and BDF methods. According to such a transformation of F_k to $\tilde{F}_k$ the iteration (3) should yield $\tilde{x}^{(i)} = Tx^{(i)}$ for all iterations i when a linear transformation $\tilde{x}^{(0)} = Tx^{(0)}$ is applied to the state. Analysing the TR1 update, one obtains the following result with respect to scaling invariance:

Theorem 1 (Conditions for Scaling Invariance).
Suppose $T \in \mathbb{R}^{n\times n}$ is a regular matrix and $F : \mathbb{R}^n \to \mathbb{R}^n$ a given vector function. For $x \in \mathbb{R}^n$, define $\tilde{x} = Tx$ and $\tilde{F} : \mathbb{R}^n \to \mathbb{R}^n$ with $\tilde{F}(\tilde{x}) = TF(T^{-1}\tilde{x})$. Then one has for the rank-1 updates given by

$$A_{i+1} = A_i + \frac{(F'(x^{(i+1)}) - A_i)s_i z_i^T (F'(x^{(i+1)}) - A_i)}{z_i^T (F'(x^{(i+1)}) - A_i)s_i} \tag{1.10}$$

and

$$\tilde{A}_{i+1} = \tilde{A}_i + \frac{(\tilde{F}'(\tilde{x}^{(i+1)}) - \tilde{A}_i)\tilde{s}_i \tilde{z}_i^T (\tilde{F}'(\tilde{x}^{(i+1)}) - \tilde{A}_i)}{\tilde{z}_i^T (\tilde{F}'(\tilde{x}^{(i+1)}) - \tilde{A}_i)\tilde{s}_i} \tag{1.11}$$

that $\tilde{A}_i = TA_iT^{-1}$ holds for all i if

$$\tilde{x}^{(0)} = Tx^{(0)}, \qquad \tilde{A}_0 = TA_0T^{-1}, \qquad \tilde{s}_i = Ts_i, \quad \tilde{z}_i = T^{-T}z_i \tag{1.12}$$

is valid for all i.

Proof: We prove the assertion by induction. For $i = 0$, one has

$$\tilde{x}^{(1)} = \tilde{x}^{(0)} - \tilde{A}_0^{-1}\tilde{F}(\tilde{x}^{(0)}) = T\left(x^{(0)} - A_0^{-1}F(x^{(0)})\right) = Tx^{(1)}.$$

It follows immediately that $\tilde{F}'(\tilde{x}^{(1)}) = TF'(x^{(1)})T^{-1}$. Furthermore, one obtains

$$\begin{aligned}
\tilde{A}_1 &= TA_0T^{-1} + \\
&\quad \frac{(TF'(x^{(1)})T^{-1} - TA_0T^{-1})Ts_0z_0^TT^{-1}(TF'(x^{(1)})T^{-1} - TA_0T^{-1})}{z_0^TT^{-1}(TF'(x^{(1)})T^{-1} - TA_0T^{-1})Ts_0} \\
&= TA_0T^{-1} + T\frac{(F'(x^{(1)}) - A_0)s_0z_0^T(F'(x^{(1)}) - A_0)}{z_0^T(F'(x^{(1)}) - A_0)s_0}T^{-1} \\
&= T\left(A_0 + \frac{(F'(x^{(1)}) - A_0)s_0z_0^T(F'(x^{(1)}) - A_0)}{z_0^T(F'(x^{(1)}) - A_0)s_0}\right)T^{-1} \\
&= TA_1T^{-1},
\end{aligned}$$

and the assertion is shown for $i = 0$.

The induction step $i \mapsto i+1$ can be proven in the same way by substituting the subscripts 0 and 1 with the subscripts i and $i+1$, respectively, in the last two equations. ∎

For the direction $s_i = x^{(i+1)} - x^{(i)}$, condition $\tilde{s}_i = Ts_i$ is naturally fulfilled. Unfortunately, this is not true for the weight vector $z_i = F_k(x^{(i+1)})$ used in the Least-squares update. Hence, it yields only the same invariance properties with respect to scaling in the range as the so-called Bad Broyden update.

Another aspect, that has to be considered, is that the denominator in (7) might vanish before the iteration converged. In this situation several strategies are conceivable. An approach is to disturb the vectors s_i and z_i. In the implementation we choose to perform no update and reuse the current approximation. A further solution for this problem could be choosing the adjoint direction as $z_i = (F'(x^{(i+1)}) - A_i)s_i$. With this the denominator is greater zero as long as the iteration did not yet converge and the approximation is not exact. However this update is not invariant with respect to linear transformations in the state space.

Therefore, we also present an alternative definition of z_i to maintain full transformation invariance in the domain and range.

Adjoint approach

Quite often one has a problem-dependent functional $\phi(x)$ in addition to the initial value problem to be solved, e.g. the output of one product, the concentration of all ingredients for a chemical reaction or the total loss of energy. Then one can use an adjoint vector to quantify the influence of discretization errors or errors in the solution of (2) on the problem-dependent functional $\phi(x)$. For that purpose, we define the adjoint vector $\lambda \in \mathbb{R}^n$ as solution of the adjoint system

$$G_{x_k}(\lambda) := F_k'(x_k)^T\lambda - \nabla\phi(x_k) = 0. \tag{1.13}$$

Consequently λ can be interpreted as the sensitivity of $\phi(x_k)$ with respect to changes in the equation $F_k(x_k) = 0$. Solving (13) by the quasi-Newton iteration

$$\lambda^{(i+1)} = \lambda^{(i)} - A_i^{-T} \left(F'(x^{(i+1)})^T \lambda^{(i)} - \phi(x^{(i+1)}) \right) \tag{1.14}$$

yields the direct tangent condition $A_{i+1}^T \sigma_i = F'(x^{(i+1)})^T \sigma_i$ with $\sigma_i = \lambda^{(i+1)} - \lambda^{(i)}$ for the system $G_{x_k}(\lambda) = 0$. This is equivalent to an adjoint tangent condition with $z_i = \sigma_i$ for the system $F_k(x) = 0$. Hence, two quasi-Newton iterations are performed simultaneously: The first one to solve (2), and the second one to solve (13). However, due to the definition of both nonlinear systems of equations the system matrix is exactly the same and therefore can be reused. The 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. To prove this assertion, we first show the following theorem:

Theorem 2 (Scaling Invariance of the Adjoint Information).
Suppose $T \in \mathbb{R}^{n \times n}$ is a regular matrix and $F : \mathbb{R}^n \to \mathbb{R}^n$ a given vector function. For $x \in \mathbb{R}^n$, define $\tilde{x} = Tx$ and $\tilde{F} : \mathbb{R}^n \to \mathbb{R}^n$ with $F(\tilde{x}) = TF(T^{-1}\tilde{x})$. Furthermore, assume that $\phi, \tilde{\phi} : \mathbb{R}^n \to \mathbb{R}$ are given with $\tilde{\phi}(\tilde{x}) = \phi(x)$. Let $\lambda, \tilde{\lambda} : \mathbb{R}^n \to L(\mathbb{R}^n, \mathbb{R})$ be the solutions of

$$\lambda(x)^T F'(x) = \nabla\phi(x)^T \quad \textit{and} \quad \tilde{\lambda}(\tilde{x})^T \tilde{F}'(\tilde{x}) = \tilde{\nabla}\tilde{\phi}(\tilde{x})^T \ .$$

Then, the equality

$$\tilde{\lambda}(\tilde{x})^T = \lambda(x)^T T^{-1} \tag{1.15}$$

is valid

Proof: One has $\tilde{F}'(\tilde{x})^{-1} = TF'(x)^{-1}T^{-1}$. Furthermore, the equality $\tilde{\phi}(\tilde{x}) = \phi(T^{-1}\tilde{x})$ holds. It follows that

$$\tilde{\nabla}\tilde{\phi}(\tilde{x})^T = \frac{d}{d\tilde{x}}\tilde{\phi}(\tilde{x}) = \frac{d}{d\tilde{x}}\phi(\underbrace{T^{-1}\tilde{x}}_{x}) = \frac{d}{dx}\phi(x)T^{-1} = \nabla\phi(x)^T T^{-1}.$$

Therefore, one obtains

$$\tilde{\lambda}(\tilde{x})^T = \tilde{\nabla}\tilde{\phi}(\tilde{x})^T \tilde{F}'(\tilde{x})^{-1} = \nabla\phi(x)^T T^{-1} T F'(x)^{-1} T^{-1} = \lambda(x)^T T^{-1},$$

and the assertion is proven. ∎

The property (15) can be transfered directly to the quasi-Newton iteration to solve (13) if $\tilde{\lambda}^{(0)} = T^{-T}\lambda^{(0)}$. Hence, one obtains that

$$\tilde{z}_i = \tilde{\lambda}^{(i+1)} - \tilde{\lambda}^{(i)} = T^{-T}(\lambda^{(i+1)} - \lambda^{(i)}) = T^{-T} z_i$$

holds, and therefore the Adjoint update is scaling invariant. Although it did not occour in the numerical tests, with this update it is also possible that the denominator in (7) vanishes. Disturbing the directions s_i and z_i would destroy the transformational invariance. Therefore performing no update is appropriate to keep invariance of the iteration.

1.3 Implementation Details

The software to test and compare the proposed rank-1 updates is written in C/C++. It provides three different integration methods, namely the implicit Euler method, the 3-stage Radau IIA method and BDF formulas [HW91].

Applying the implicit Euler method, the system to be solved is given by $F_k(x) = x - x_{k-1} - hf(x) = 0$ and therefore has dimension n. The convergence order of this method is 1. Using the 3-stage Runge-Kutta method Radau IIA, one has to solve a nonlinear system of dimension $3n$. Hence, the complexity increases, but on the other hand the method has order 5. The BDF formulas correspond to linear multi-step methods, where the system of equations is given by

$$F_k(x) = \alpha_0 x - \sum_{j=1}^{l} \alpha_j x_{k-j} - h_k f(x) = 0 \in \mathbb{R}^n$$

with certain scalars α_j. These methods are of order l.

For the solution of the nonlinear systems, we implemented Newton's method in the following way. To compute the complete Jacobian of the right hand side function $f(x(t))$, we employ the AD-tool ADOL-C [GJU96] that provides exact first and higher order derivatives for C/C++ function evaluations using operator overloading. Subsequently, the Jacobian of the nonlinear system is easily computed from the Jacobian of the right-hand side using vector forward mode of AD. Finally, a QR-factorization is performed to compute the next Newton step.

Furthermore, we coded Broyden's method as well as the two new quasi-Newton approaches to approximate the Jacobian information during the solution of the nonlinear system. Once more, we maintain a QR-factorization of the corresponding updates in order to compute the next iteration step efficiently. As starting point we compute the exact Jacobian for the initial value x_0. The derivative information required by the TR1 update namely $F_k'(x^{(i+1)})s_i$ and $z_i^T F_k'(x^{(i+1)})$ are calculated using the scalar forward and reverse mode provided by ADOL-C.

For stabilizing the Newton as well as the quasi-Newton approach we perform a line search with quadratic and cubic interpolation, respectively, as described in [DS96]. Furthermore, we incorporated a simple adaptive time stepping according to the approach analyzed in [SW95].

1.4 Numerical Results

For the numerical tests, we take two initial value problems from the *Testset for Initial Value Problem Solvers*, University of Bary, Italy [Tes]. The first one is the *Pollution Problem*, a stiff system of 20 non-linear ODEs. It describes a chemical reaction as part of the air pollution model developed at The Dutch National Institute of Public Health and Environmental Protection (RIVM), and consists of 25 reaction and 20 reacting compounds. The second test is the *Medical Akzo Nobel Problem* consisting originally of two partial differential equations. Semi-discretization of this system yields 400 stiff ODEs. The Akzo Nobel research laboratories formulated this problem in their study of the penetration of radio-labeled antibodies into a tissue that has been infected by a tumor. Both problems have in common that the right-hand side of the ODE is nonlinear.

Pollution Problem

To suit our software, we reformulate the Pollution Problem as autonomous ODE system with 21 component functions. For comparison, the tests were performed using Newton's method with AD based Jacobians as described in Section 3, the Broyden update, which is a secant method using only information of F, and the two presented variants of the TR1 update. Since the Pollution Problem describes a chemical reaction, we chose the problem-dependent functional $\phi(x)$ to be the concentration of CO_2. For this problem we use adaptive time stepping where the step size criteria are the same for all numerical tests.

	$h_0 = 10^{-2}$				$h_0 = 10^{-3}$			
	Newt	LS	Adj	Broy	Newt	LS	Adj	Broy
time steps	412	412	412	—	4077	4077	4077	4077
iterations	713	716	918	—	4512	4504	4521	11319
CPU-time (s)	0.33	0.15	0.20	—	2.07	0.98	1.05	1.69

Table 1.1. Euler method with adaptive time stepping

The numerical results achieved with the Euler method are given in Table 1. The integration was performed for the time interval $[0, 60]$, i.e. $T = 60$. As can be seen, the integration fails using a larger time step h_0 as initialization if Broyden's method is applied. All other approaches yield the results reported as solutions at the test suite website [Tes]. The number of time steps is the same for all methods where the integration over the whole time interval was possible. However, the numbers of iterations for solving the nonlinear systems differ remarkably. These numbers are again almost the same for Newton's approach and the Least-squares update, i.e. the TR1 update with $z_i = F_k(x^{(i+1)})$, but due to the computation of the complete Jacobian and its factorization required

for Newton's method the corresponding run time is naturally significantly larger. The iteration count for the adjoint update, i.e. the TR1 update with z_i based on the problem dependent function, is higher which is reflected also in the run times. Since one has to perform two reverse mode differentiations the factor between the run times is larger than the factor between the iteration counts. The iteration count for Broyden's method is even higher but since no derivative calculations are performed, the run time is less than the run time for the Newton's method.

	$h_0 = 10^{-2}$				$h_0 = 10^{-3}$			
	Newt	LS	Adj	Broy	Newt	LS	Adj	Broy
time steps	412	412	412	412	565	565	565	—
iterations	708	723	1081	2661	1037	1041	1462	—
CPU-time (s)	5.66	0.85	1.30	2.42	8.34	1.24	1.80	—

Table 1.2. Radau IIA with adaptive time stepping

	$l = 3$				$l = 6$			
	Newt	LS	Adj	Broy	Newt	LS	Adj	Broy
time steps	412	412	412	412	412	412	412	—
iterations	693	702	929	2144	690	695	933	—
CPU-time (s)	0.37	0.21	0.27	0.40	0.37	0.19	0.28	—

Table 1.3. BDF-formula with adaptive time stepping

The numerical results achieved with Radau IIA and with two BDF formulas, i.e. $l = 3$ and $l = 6$, are given in Tables 2 and 3, respectively. Once more, the integration was performed for the time interval $[0, 60]$ to verify the results. Now, the integration fails using a smaller time step h_0 as initialization (Radau IIA) or a higher order method (BDF) if Broyden's method is applied. The numbers for the methods where the integration converges confirm the behaviour of the solution methods for the nonlinear system of equations already observed for the Euler method.

Medical Akzo Nobel Problem

We reformulate this problem as an autonomous ODE system yielding a system of 401 ODEs. For this example, it was not possible to get results using Broyden's method despite the fact that intensive testing with respect to step sizes was done. As functional ϕ in the Adjoint update we choose the product of the concentrations of the reacting components. Furthermore, we do not apply varying step sizes since our step size heuristic is not appropriate for this

problem. Furthermore, the sparsity of the Jacobian is not taken into account.

	$h = 10^{-1}$			$h = 10^{-2}$		
	Newt	LS	Adj	Newt	LS	Adj
time steps	200	200	200	2000	2000	2000
iterations	570	1290	2012	3912	4891	7121
CPU-time (s)	1209.62	70.54	124.01	8370.17	254.39	425.75

Table 1.4. Euler method with constant time steps

The numerical results achieved with the Euler method are given in Table 4. All approaches yield the results reported as solutions at the test suite website [Tes]. However, it was necessary to provide the exact Jacobian at $t = 5$ if $h = 10^{-1}$ applying the TR1 updates since the right hand side jumps exactly at that place and is therefore not continuous. As can be seen, the iteration increases when using the inexact derivative information provided by the Least-squares and Adjoint update. However, due to the lower cost to perform one iteration the factor of the run times equals 17 for the Least-squares and 10 for the Adjoint update with $h = 10^{-1}$ and 33 for the Least-squares and 20 for the Adjoint update with $h = 10^{-2}$.

This observation is also confirmed by examining the runtimes needed for the calculation of one Jacobian and its factorization compared to one rank-1 update of an approximation. For the computation of the Jacobian 0.051 seconds are needed while its factorization lasts 2.1 seconds. In contrast to this, computing the new factorized approximation in the Least-squares update only needs 0.054 seconds. This shows that the main computing computing effort lies in the factorization of the Jacobian to solve the linear system.

	$h = 10^{-1}$		
	Newt	LS	Adj
time steps	200	200	200
iterations	591	1705	2842
CPU-time (s)	32175.02	988.96	1756.97

Table 1.5. Radau IIA with constant time steps

The numerical results achieved with Radau IIA and with two BDF formulas, i.e. $l = 3$ and $l = 6$, are given in Tables 5 and 6, respectively. The behaviour already observed for the Euler method is confirmed: The iteration count increases due to the approximation of the Jacobian, but the overall run time is drastically reduced due to the much lower computation effort required by one quasi-Newton iteration in comparison to the calculation and factorization of the complete Jacobian.

	$l = 3$			$l = 6$		
	Newt	LS	Adj	Newt	LS	Adj
time steps	2000	2000	2000	2000	2000	2000
iterations	3880	4771	6684	3798	4667	6730
CPU-time (s)	8228.23	266.97	418.12	7932.48	266.80	425.66

Table 1.6. BDF-formula with constant time steps $h = 10^{-2}$

1.5 Conclusions and Outlook

So far the use of quasi-Newton methods for the solution of nonlinear systems that arise during the integration of stiff ODEs is not widespread. We present two new variants of the Two-sided Rank-1 update (TR1). These AD-based quasi-Newton methods fulfill the exact direct tangent condition as well as exact adjoint tangent condition. Here, the selective choice of tangents and adjoints facilitates invariance and therefore norm independence of the state space. The proposed update formulas were tested using a well-known IVP test suite. For the examples considered during this project the achieved numerical results are very promising. Usually the Least-squares and the Adjoint updates perform significantly better than Broyden's method.

However, there are several open questions. First, detailed convergence analysis of the TR1-update for the solution of nonlinear equations is needed. This theoretical examination may also motivate alternative choices of s_i and z_i. Additionally, the maintaining or adjustment of a suitable factorization in the case of varying time step sizes has to be studied for a successful integration of the quasi-Newton methods for the integration of stiff ODEs. Here the task is to find a factorization that allows a change in the step size without performing a complete factorization again.

References

[BHP94] P. Brown, A. C. Hindmarsh, and L. R. Petzold. Using krylov methods in the solution of large-scale differential-algebraic systems. *SIAM J. Sci. Comp.*, 15:1467–1488, 1994.

[BHW85] P. Brown, A. C. Hindmarsh, and H. Walker. Experiments with quasi-newton methods in solving stiff ODE systems. *SIAM J. Sci. Stat. Comput.*, 6:297–313, 1985.

[CH96] S. D. Cohen and A. C. Hindmarsh. Cvode, a stiff/nonstiff ode solver in c. *Computers in Physics*, 10:138–143, 1996.

[DS96] J. E. Jr. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics, SIAM, 1996.

[GJU96] A. Griewank, D. Juedes, and J. Utke. Griewank1996AAp: A package for automatic differentiation of algorithms written in C/C++. *TOMS*, 22:131–167, 1996.

[GW02] A. Griewank and A. Walther. On constrained optimization by adjoint based quasi-newton methods. *Opt. Meth. and Soft.*, 17:869–889, 2002.

[HW91] E. Hairer1991SOD and G. Wanner. *Solving Ordinary Differential Equations II.* Springer-Verlag, 1991.

[SW95] L. F. Shampine and A. Witt. A simple stepsize selection algorithm for ODE codes. *J. Comput. Appl. Math.*, 58:345–354, 1995.

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