Sensitivity Analysis of Initial Value Problems with Error Control

Stephan Franz and Martin Kiehl

Contents

Abstract

In this article we present two FORTRAN-codes DOPRI8S and GRK4AS based on FORTRAN-subroutines for the solution of initial value problems for ordinary differential equations which calculate the sensitivity of the solution with respect to the initial values and the parameters. These subroutines compute the sensitivity by internal difference approximations. Compared to the classical external difference approximations we obtain higher accuracies and a better reliability due to error control. Moreover we can save execution time especially for implicit integration subroutines. These advantages are demonstrated by some examples.

AMS Subject Classification: 65 D25, 65 L05, 65 L07

1 Introduction

Let us consider an initial value problem (IVP) for a semi-implicit differential algebraic equation (DAE) of index one with consistent initial values

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t), p) \\ 0 &= g(t, y(t), z(t), p) \\ y(t_0) &= y_0 \in \mathbb{R}^{n_y} \\ z(t_0) &= z_0 \in \mathbb{R}^{n_z} \end{aligned} \tag{1}$$

with a parameter vector $p \in \mathbb{R}^{n_p}$. For index one problems consistent means $g(t_0, y_0, z_0, p) = 0$.
The solution of the problem will be denoted by $y(t; t_0, y_0, z_0, p), z(t; t_0, y_0, z_0, p)$.
The sensitivity matrix of this problem is given by:

$$\begin{aligned} S(t; t_0, y_0, z_0, p) &:= \\ := \frac{\partial(y, z)}{\partial(y_0, z_0, p)}(t; t_0, y_0, z_0, p) &= \begin{pmatrix} \frac{\partial y}{\partial y_0} & \frac{\partial y}{\partial z_0} & \frac{\partial y}{\partial p} \\ \frac{\partial z}{\partial y_0} & \frac{\partial z}{\partial z_0} & \frac{\partial z}{\partial p} \end{pmatrix} (t; t_0, y_0, z_0, p) \end{aligned} \tag{2}$$

For a theoretical analysis it is useful to treat the parameters p as state variables by formulating the trivial ODE $\dot{p} = 0$ and substituting $w := (y, p, z)$. Then we get a DAE without parameters, where the parameters are initial values:

$$
\begin{array}{ccccc}
\left(\begin{array}{c} \dot{w}_1 \\ \vdots \\ \dot{w}_{n_y} \\ \hline \dot{w}_{n_y+1} \\ \vdots \\ \dot{w}_{n_y+n_p} \end{array}\right) & = & F(t,w) & := & \left(\begin{array}{c} f(t,y,z,p) \\ \hline 0 \\ \vdots \\ 0 \end{array}\right) \\
0 & = & G(t,w) & := & g(t,y,z,p) \\
w(t_0) & = & w_0 & := & (y_0,p,z_0)^T
\end{array}
\tag{3}
$$

The sensitivity matrix of this problem contains all the information of the sensitivity matrix of the problem (1):

$$
S(t;t_0,w_0) := \frac{\partial w}{\partial w_0}(t;t_0,w_0) = \begin{pmatrix} \frac{\partial y}{\partial y_0} & \frac{\partial y}{\partial p} & \frac{\partial y}{\partial z_0} \\ 0 & I & 0 \\ \frac{\partial z}{\partial y_0} & \frac{\partial z}{\partial p} & \frac{\partial z}{\partial z_0} \end{pmatrix} (t;t_0,y_0,z_0,p) \tag{4}
$$

If we differentiate the DAE (3) with respect to w_0 we obtain a DAE

$$
\begin{array}{rrcl}
 & \dot{S}(t;t_0,w_0) & = & F_w(t,w(t;t_0,y_0)) \cdot S(t;t_0,w_0) \\
 & 0 & = & G_w(t,w(t;t_0,y_0)) \cdot S(t;t_0,w_0) \\
\text{with} & F_w(t,w) & = & \begin{pmatrix} f_y & f_p & f_z \\ 0 & 0 & 0 \end{pmatrix} (t,y,z,p) \\
\text{and} & G_w(t,w) & = & (g_y, g_p, g_z)(t,y,z,p)
\end{array}
\tag{5}
$$

for the sensitivity matrix $S(t;t_0,w_0)$. Because of the algebraic or more exactly the linear equation of this DAE the columns of $S(t;t_0,w_0)$ have to be in the kernel of $G_w(t,w)$, i. e. to be consistent. Especially the initial value $S(t_0)$ has to be consistent.

Ordinary differential equations (ODE) are only a special case of DAE's with no algebraic equations. So everything also applies to ODE's, but for ODE's there are no consistency conditions for $S(t_0)$.

In the following sections we will explain how the sensitivity matrix can be computed efficiently using nearly arbitrary existing numerical integration routines for ODE's and DAE's.

2 Numerical Sensivity Analysis and Implementation

The sensitivity is a derivative of the solution of a DAE or ODE. One way to calculate derivatives numerically is to use difference approximations. We will describe this in the next section. Then we explain some details of the implementation.

2.1 Internal Difference Approximations

We consider the problem

$$\begin{aligned} \begin{pmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_m \end{pmatrix} &= F(t, y) \qquad (6) \\ 0 &= G(t, y) \\ y(t_0) &= y_0 \in \mathbb{R}^n \end{aligned}$$

For this problem we calculate a numerical solution $\eta(t; t_0, y_0, [h])$ where $[h]$ denotes the step size sequence $(h_i)_{i=1}^k$ used to calculate η. As an approximation of the sensitivity in direction r we use the following difference approximation:

$$\begin{aligned} S(t; t_0, y_0) \cdot r &\approx \frac{y(t; t_0, y_0 + \varepsilon r) - y(t; t_0, y_0)}{\varepsilon} \approx \\ &\approx \frac{\eta(t; t_0, y_0 + \varepsilon r, [h]_1) - \eta(t; t_0, y_0, [h]_2)}{\varepsilon} \qquad (7) \end{aligned}$$

We get the best results if we use the same step size sequences $[h]_1 = [h]_2$ in (7). For fixed step size sequences, the numerical approximation is a differentiable function of the initial values. Otherwise it has a stochastic component due to non-differentiable decisions in the step size control and the solutions have to be computed with high accuracy. This is very time consuming and can be avoided when using $[h]_1 = [h]_2$. The exact theoretical background for this better behaviour can be read in [2], [3], [4], [6] and [11].
The result of the error analysis (compare [6]) for this fixed step sizes is that the error of the difference approximation is of the following order:

$$\begin{aligned} &\frac{\eta(t; t_0, y_0 + \varepsilon r, [h]) - \eta(t; t_0, y_0, [h])}{\varepsilon} = \\ &= S(t; t_0, y_0) \cdot r + O(\bar{h}^p) + O(\varepsilon) + O\left(\frac{\varepsilon_{mach}}{\varepsilon}\right) \qquad (8) \end{aligned}$$

with $\bar{h} = \max_i |h_i|$, ε_{mach} the machine precision and p the order of the numerical integration scheme. This means that the error of the difference approximation of S is of the same order as the error of the numerical solution. Note, that the step size need not to be fixed a priori. It is sufficient to use the same sequences $[h]_1 = [h]_2$ where the elements of $[h]_1$ can still be chosen during runtime of the integration.

The fixing of the step sizes will be realized in the following way. If a numerical approximation η_i of the solution $y(t_i; t_0, y_0)$ and a numerical approximation $\sigma_{j,i}$ of the sensitivities in directions r_j, $j = 1, \ldots, n_s$ at point t_i are computed we calculate approximations at the next point $t_{i+1} = t_i + h$ by

$$\eta_{i+1} = \eta(t_{i+1}; t_i, \eta_i, h) \tag{9}$$

$$\hat{\eta}_{j,i+1} = \eta(t_{i+1}; t_i, \eta_i + \varepsilon\sigma_{j,i}, h), \quad j = 1, \ldots, n_s \tag{10}$$

The new numerical approximation of the sensitivity at t_{i+1} is then given by

$$\sigma_{j,i+1} = \frac{\hat{\eta}_{j,i+1} - \eta_{i+1}}{\varepsilon}, \quad j = 1, \ldots, n_s \tag{11}$$

Thereby we obtain the numerical solution and the numerical sensitivity at the new point.

2.2 Rescaling of the Perturbations

A decisive topic when using difference approximations is the proper choice of the increment. In case of the difference approximations for the sensitivities the increment is the perturbation of the initial values $\varepsilon\sigma_{j,i}$. In our implementation it is possible to change the scaling factor ε after every step and to adapt it to the new size of the solution and the sensitivity at the current time point. This is sensible, because too small perturbations can lead to cancellation and too large perturbations to a dominant error term of order $O(\varepsilon)$.

If we choose $\varepsilon = O(\sqrt{\varepsilon_{mach}})$ the two parts of the error of the sensitivity $O(\varepsilon) + O\left(\frac{\varepsilon_{mach}}{\varepsilon}\right)$ in equation (8) are of the same size. In this case we obtain very good results as shown in [11] and [6].

Let us now consider the situation at point t with a given numerical solution η and a sensitivity σ in direction r. We want to calculate an appropriate scaling factor ε. We compute the vector v

$$v_k = \frac{|\sigma_k|}{|\eta_k| + 1} \tag{12}$$

a mixture of relative and absolute perturbations. Then a proper scaling factor is

$$\varepsilon = \frac{1}{||v|| + \sqrt{\varepsilon_{mach}}} \sqrt{\varepsilon_{mach}} \tag{13}$$

If we have more information on the expected errors of the different components a more subtle scaling is possible (compare with [12]).

2.3 Error Control and Step Size Choice

A further advantage of the method is that the implemented error control of most integration methods can easily be extended to an error control of the computed sensitivity.

We take a look at the i-th step from t_i to $t_{i+1} = t_i + h$ of a given integration scheme. To control the error of the solutions η_{i+1} and $\hat{\eta}_{j,i+1}$ the considered integration schemes calculate further approximations $\bar{\eta}_{i+1}$ and $\bar{\hat{\eta}}_{j,i+1}$ which are less accurate than η_{i+1} and $\hat{\eta}_{j,i+1}$. Then the differences $d_{i+1} := \bar{\eta}_{i+1} - \eta_{i+1}$ and $\hat{d}_{j,i+1} := \bar{\hat{\eta}}_{j,i+1} - \hat{\eta}_{j,i+1}$ are taken as approximations for the error of $\bar{\eta}_{i+1}$ and $\bar{\hat{\eta}}_{j,i+1}$. An approximation for the error of the sensitivity is given by

$$\begin{aligned}
\Delta_{j,i+1} &:= \frac{\hat{d}_{j,i+1} - d_{i+1}}{\varepsilon} = \\
&= \frac{(\bar{\hat{\eta}}_{j,i+1} - \hat{\eta}_{j,i+1}) - (\bar{\eta}_{i+1} - \eta_{i+1})}{\varepsilon} = \\
&= \frac{\bar{\hat{\eta}}_{j,i+1} - \bar{\eta}_{i+1}}{\varepsilon} - \frac{\hat{\eta}_{j,i+1} - \eta_{i+1}}{\varepsilon} = \\
&= \bar{\sigma}_{j,i+1} - \sigma_{j,i+1} \\
&\overset{(8)}{=} \bar{C}h^{\bar{p}} - Ch^{p} + O(\varepsilon) + O\left(\frac{\varepsilon_{mach}}{\varepsilon}\right) \approx O(h^{\bar{p}})
\end{aligned} \tag{14}$$

Thus $\Delta_{j,i+1}$ is an approximation of the error of $\bar{\sigma}$ which usually is much bigger than the error of σ.

If all error approximations $||d_{i+1}||, ||\hat{d}_{j,i+1}||$ and $||\Delta_{j,i+1}||$ are smaller than a given tolerance we accept the step, otherwise it is repeated with a smaller step size. For a repeated as well as for a next step, this new step size is chosen by a heuristic approach, in order to fulfill the tolerance requirements in the next step.

The formula for the new step size takes into account the error of the last step, the required tolerance and the approximation order p. According to (14) the error of σ has the same order p as the error of η so we can use the same step size formula with $||\Delta_{j,i+1}||$ instead of $||d_{i+1}||$, the tolerance δ_S for the sensitivity and

the order p. Because of (8) it is not possible to obtain higher accuracies than of order $O(\sqrt{\varepsilon_{mach}})$. So the user can only require a tolerance $\delta_S > \sqrt{\varepsilon_{mach}}$ for the sensitivity. Otherwise instead of (7) higher order approximations like symmetric difference approximations have to be used.
In each step we obtain different step size guesses for the next step, based on the errors $||d_{i+1}||, ||\hat{d}_{j,i+1}||$ and $||\Delta_{j,i+1}||$. To make sure that all these errors will also fulfill our tolerance requirements in the next step, we take the minimum of these step size guesses. The error control of the sensitivity then leads to more reliable results and usually also to savings in computing time.

2.4 Calculation of the Solution with Higher Accuracy

In many applications the required accuracy δ of the solution and the accuracy δ_S of the sensitivity are different (compare [4], [11], [13]). One possibility would be to calculate the solution and the sensitivity independently with different accuracies. But the global error of the computed sensitivity can profit from the higher accuracy of the solution if the solution and the sensitivity are calculated simultaneously.
We consider the situation at point t_i as in section 2.1 where the numerical approximation η_i of $y(t_i; t_0, y_0)$ and sensitivities $\sigma_{j,i}$ in directions $r_j,\ j = 1, \ldots, n_s$, are all computed with low accuracy δ_S and additionally a numerical approximation $\tilde{\eta}_i$ of $y(t_i; t_0, y_0)$ is computed with high accuracy δ.
Now we calculate solutions at $t_{i+1} = t_i + h$ as in section 2.1 equations (9), (10)

$$\begin{aligned} \eta_{i+1} &= \eta(t_{i+1}; t_i, \tilde{\eta}_i, h) \\ \hat{\eta}_{j,i+1} &= \eta(t_{i+1}; t_i, \tilde{\eta}_i + \varepsilon\sigma_{j,i}, h), \quad j = 1, \ldots, n_s\ , \end{aligned}$$

but we use $\tilde{\eta}_i$ instead of η_i.
With these solutions we calculate also approximations $\sigma_{j,i+1}$ of the sensitivities (compare equation (11)). Let us analyse the error of $\sigma_{j,i+1}$ using equation (8).

$$\begin{aligned} \sigma_{j,i+1} &= \\ &= S(t_{i+1}; t_i, \tilde{\eta}_i) \cdot \sigma_{j,i} + O(\sqrt{\varepsilon_{mach}}) + O(h^p) \ = \\ &= S(t_{i+1}; t_i, y(t_i; t_0, y_0)) \cdot \underbrace{(S(t_i; t_0, y_0) \cdot r_j + O(\sqrt{\varepsilon_{mach}}) + O(h^p))}_{\sigma_{j,i}} + \\ &\quad + \ (S(t_{i+1}; t_i, \tilde{\eta}_i) - S(t_{i+1}; t_i, y(t_i; t_0, y_0))) \cdot \sigma_{j,i} + O(\sqrt{\varepsilon_{mach}}) + O(h^p) \ = \\ &= S(t_{i+1}; t_0, y_0) \cdot r_j + (S(t_{i+1}; t_i, \tilde{\eta}_i) - S(t_{i+1}; t_i, y(t_i; t_0, y_0))) \cdot \sigma_{j,i} + \end{aligned}$$

$$+ O(\sqrt{\varepsilon_{mach}}) + O(h^p)$$

The error of $\sigma_{j,i+1}$ gets the smaller the better $\tilde{\eta}_i$ approximates $y(t_i; t_0, y_0)$. So if we use $\tilde{\eta}_i$ instead of η_i this decreases the error of the sensitivity.

But for the next step we now need an approximation of $y(t_{i+1}; t_0, y_0)$ with accuracy δ, if we want to get the same situation as at t_i. η_{i+1} is only of accuracy δ_S. So we have to calculate a further solution $\tilde{\eta}_{i+1}$ with smaller step sizes starting with $\tilde{\eta}_i$ at time point t_i.

Because $\tilde{\eta}_{i+1}$ cannot be used in the nominator of $\sigma_{j,i+1}$ because of equation (8), we have to calculate two different approximations of $y(t_{i+1}; t_0, y_0)$. This disadvantage is neglectable for high dimensional problems.

Remark: for a method of order p the computation of $\tilde{\eta}$ requires k steps for each step of the computation of η with $k \approx \sqrt[p]{\frac{\delta_S}{\delta}}$.

2.5 Reuse of Jacobians and Matrix Decompositions

For stiff ODE's implicit integration routines are used. In the most widely used algorithms for stiff problems the most time consuming part is to solve linear equations with matrices of the form

$$I_n - h\gamma f_y(\tau, \eta) \tag{15}$$

(γ a coefficient of the integration method) or with similar matrices. Thereby sometimes f_y has to be the exact jacobian at (τ, η), sometimes an approximation is sufficient.

If we use similar methods for semi-implicit DAE's the matrices in the linear equations have the form

$$\begin{pmatrix} I_n - h\gamma f_y(\tau, \eta, \zeta) & h\gamma f_z(\tau, \eta, \zeta) \\ g_y(\tau, \eta, \zeta) & g_z(\tau, \eta, \zeta) \end{pmatrix} \tag{16}$$

By linearization and neglection of small nonlinear contributions we find that the solution to the perturbed initial values can be calculated using the same matrices as for the solution to the unperturbed initial values.

This leads to great savings in computing time as we do not need to compute jacobians and matrix decompositions for the sensitivity calculation. This leads to savings of about factor n in setting up f_y resp. f_y and g_y and decomposing matrices of the form (15) or (16).

2.6 Sensitivity of Parameters

Up to now we did not make differences between the variables y and z in the DAE (or ODE) and the parameters p. Normally parameters do not appear in the subroutines for the integration of DAE's or ODE's. They are constant and only appear in the right-hand side of the equation. If we want to compute the sensitivity with respect to parameters we have to add the parameters and their dimension to the calling sequence. Then we have to change the calling sequence of the function calls for the right-hand side of the equation, too.
Unlike the other variables the parameters remain unchanged during the integration. So we need not calculate the increments for the parameters. The parameters will not have or gain errors during the integration. Therefore we do not need to consider the parameters in the error control.
Furthermore we can calculate the vector of relative perturbations for the parameters once at the beginning of the integration. If a parameter η_k is zero we have no information about the size of this parameter and we choose

$$v_k = \begin{cases} \frac{\sigma_k}{\eta_k} & \text{if} \quad \eta_k \neq 0 \\ 1 & \text{if} \quad \eta_k = 0 \end{cases} \tag{17}$$

Thus even small (badly scaled) parameters are perturbed relatively and we obtain better results for the sensitivity with respect to parameters. Especially we do not get rounding errors in the constant parts of the sensitivity and the variables.

3 Examples of Implementation

Here we present two FORTRAN-77-subroutines which allow to compute the numerical solution of an ODE for nonstiff or stiff problems together with the sensitivity of the problem. This subroutines are extensions of well known subroutines which calculate only the solution of the ODE with one step methods. We added a sensitivity analysis option to this subroutines using the ideas and methods presented in this paper.

3.1 DOPRI8S

DOPRI8S is based on DOPRI8. It is an explicit Runge-Kutta method with 13 steps of order (7)8 using coefficients calculated by Dormand and Prince (for the coefficients compare [7]). It is used for nonstiff IVP's with ODE's $\dot{y} = f(t, y, p)$, $y(t_0) = y_0$ to calculate the solution $y(t_f)$ and the sensitivity $S(t_f) \cdot R$ in directions of the columns of R. The calling sequence of DOPRI8S for such an IVP is:

CALL DOPRI8S (N,NP,NS,FCN,T,TEND,Y,P,SENS,TOL,TOLS,HMAX,H).

The parameters are

N, NP, NS	dimension of y, p and number of sensitivity directions (= number of columns of S)
FCN	name of the subroutine for the calculation of $f(t, y, p)$
T	input: initial time t_0; output: current value of t, (after successful integration t_f)
TEND	end point t_f of integration
Y	vector of dimension N; input: initial value y_0; output: current value $y(t)$; (after successful integration $y(t_f)$)
P	vector of dimension NP; constant parameter vector p
SENS	matrix of dimension (N+NP)$\times$NS; input: initial sensitivity directions R; output: current value $S(t) \cdot R$
TOL, TOLS	tolerance for the solution and for the sensitivity
HMAX	maximal step size
H	input: initial step size guess; output: last step size before stop of integration

To compute only the solution of the ODE choose NS=0 and provide a dummy matrix SENS and a dummy variable EPSS. If the complete sensitivity matrix is required choose NS = N+NP and initialize SENS = I.

The user has to provide two subroutines FCN and OUTSENS which are called by DOPRI8S. The calling sequence of FCN is CALL FCN (N,NP,T,Y,P,F). The parameters of this routine have to remain unchanged except for F which is a vector of dimension N and has to contain the value $f(t, y, p)$ as output.

OUTSENS is called by CALL OUTSENS (N,NP,NS,T,Y,P,SENS). This subroutine can be used for output of the solution and the sensitivity at intermediate points. The user has to provide at least a dummy subroutine. The parameters have to remain unchanged during the call.

For statistical information about the integration the subroutine DOPRI8S uses the COMMON-block COMMON /STAT/ NFCN,NSTEP,NACCPT,NREJCT.

The variables are the number of calls of the subroutine FCN, of computed steps from t_0 to current t, of accepted steps and of rejected steps.

3.2 GRK4AS

GRK4AS is based on GRK4A. It is a Rosenbrock-Wanner-method with four steps of order (3)4 using coefficients calculated by Kaps and Rentrop (for the coefficients compare [8]). It is used for stiff IVP's with ODE's. Therefore it has almost the same calling sequence as DOPRI8S: CALL
GRK4AS (N,NP,NS,FCN,T,TEND,Y,P,JAC,IJAC,SENS,TOL,TOLS,HMAX,H).
It contains following additional parameters:

IJAC	switch:	IJAC=0	numerical approximation of the jacobian $f_y(t,y,p)$
		IJAC=1	analytical calculation of the jacobian $f_y(t,y,p)$ by call of subroutine JAC
JAC	name of the subroutine for the calculation of $f_y(t,y,p)$		

As DOPRI8S the subroutine also calls the subroutines FCN and OUTSENS with the same calling sequences. Additionally the user has to provide the subroutine JAC. The calling sequence is CALL JAC(N,NP,T,Y,P,DFY). If the user wants to use numerical approximations of the jacobian $f_y(t,y,p)$ it is sufficient to provide a dummy subroutine. Otherwise the user has to program the jacobian $f_y(t,y,p)$ which will be the output of JAC. The output is saved in the matrix DFY of dimension N$\times$N. The other parameters have to remain unchanged.
As in DOPRI8S the user can get some statistical information about the integration by the COMMON-block
COMMON /STAT/ NFCN,NSTEP,NACCPT,NREJCT,NDECOMP,NJAC.
The additional variables NDECOMP and NJAC are the number of decompositions of matrices and the number of calls of the subroutine JAC or of computations of numerical approximations of the jacobian $f_y(t,y,p)$.

4 Test Examples

In this section we present some examples we used to test our implementations. The results were compared to the results using the original integration subroutines with unperturbed and perturbed initial values to compute external difference approximations of the sensitivity by (7), which is the usual way to compute $S(t)\cdot R$. These integrations use different step size sequences for the different solutions.

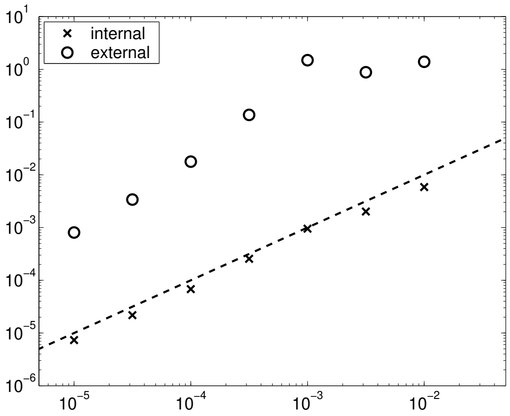
As tolerances we choose $\delta = \delta_S^2$. In case of external difference approximations the tolerance δ has to be used for all approximations $\eta(t; t_0, y_0 + \delta_S r_j, [h]_i)$ leading to increasing computing time but still less reliability.
For each computation we present the error of the sensitivity and the computing time for the computation of the solution and the sensitivity. More test examples can be found in [6].

4.1 Nonstiff ODE's

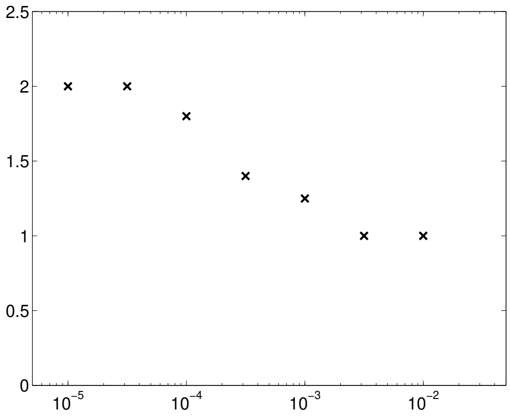
For nonstiff ODE's we used DOPRI8S and the examples from the testset DETEST [10].
The example C1 of this testset is linear of dimension 10:

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_{10}' \end{pmatrix} = \begin{pmatrix} -1 & & & & \\ 1 & -1 & & & \\ & 1 & \ddots & & \\ & & \ddots & -1 & \\ & & & 1 & -1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{10} \end{pmatrix}, \qquad y(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

In all plots the x-axis represents the tolerances. The first plot shows if the error is lower than the required tolerance. This demonstrates the reliability of the program. The markers 'x' are the results of DOPRI8S and the markers 'o' that of the external difference approximations with DOPRI8. The second plot compares the computing times of the two calculation possibilities at given tolerances; the ratio of DOPRI8 to DOPRI8S is plotted.



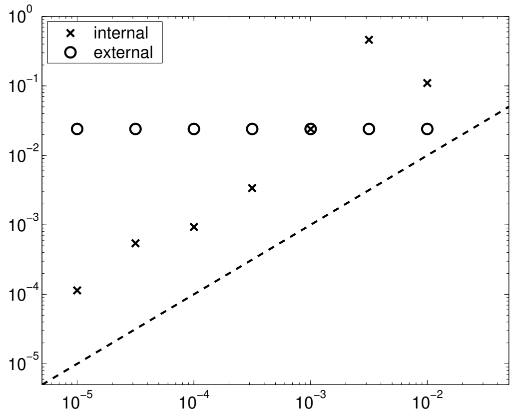
error via tolerance for example C1 with DOPRI8S and DOPRI8



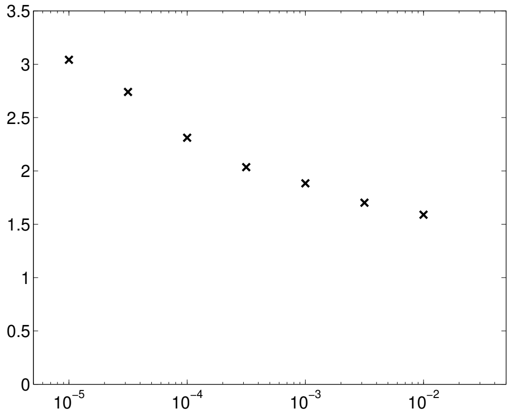
ratio of computing times between DOPRI8 and DOPRI8S via tolerance for example C1

The accuracy of DOPRI8S is much better and complies with the tolerance requirements very precise. At the same time for smaller tolerances the saving in computing time is about one half.

The second example is the five body motion (example C5 in DETEST). This is a highly nonlinear problem of dimension 30 if it is transformed into a system of first order.
We present the same plots as for the first example:



error via tolerance for example C5 with DOPRI8S and DOPRI8


ratio of computing times between DOPRI8 and DOPRI8S via tolerance for example C5

For this nonlinear example the errors are larger than the required tolerance both for DOPRI8S and DOPRI8. But we can see that by the local error control in the subroutine DOPRI8S we can control the global error of the sensitivity and get better results for small tolerances. The saving in computing time is up to a factor three.
The other examples of the testset show similar results. For almost all examples DOPRI8S is more reliable. Sometimes DOPRI8 is more accurate and often DOPRI8 is less accurate than DOPRI8S and the required tolerance. In the lower dimensional examples the overhead is not neglectable. So there are no large savings in computing time with DOPRI8S. But for the high dimensional examples as the two presented here we observe savings up to an factor three for small tolerances in addition to higher accuracy.

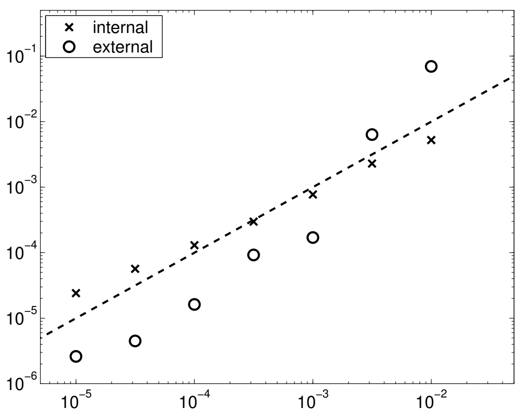
4.2 Stiff ODE's

For stiff ODE's we used GRK4AS and the testset STIFFDETEST [5].

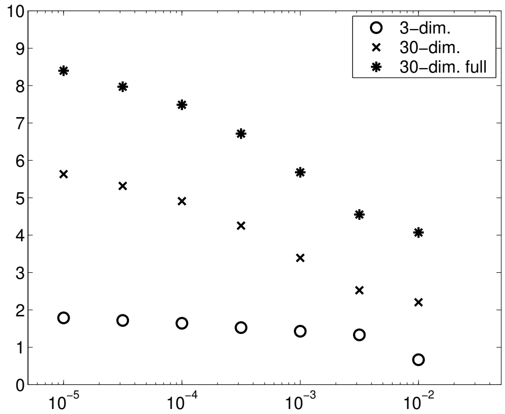
The example E3 of this testset is:

$$\begin{aligned} y_1' &= -(55+y_3)y_1+65y_2 & \qquad y_1(0) &= 1 \\ y_2' &= 0.0785(y_1-y_2) & y_2(0) &= 1 \\ y_3' &= 0.1y_1 & y_3(0) &= 0 \\ x_f &= 500 & h_0 &= 0.02 \end{aligned}$$

The main savings in computing time depend on the dimension of the linear equations in GRK4AS. So we do not obtain big savings for this low dimensional problem. Therefore we copied the example ten times obtaining dimension 30. But still the matrices are structured. If we do not use this structure we obtain larger savings in computing time. We cannot use any structure if the matrices are full.
The ratios of computing time are marked for the simple example with 'o', for the example copied 10 times with 'x' and for the same example not using the structure of the matrices with '*'.



error via tolerance for example E3 with GRK4AS and GRK4A



ratio of computing times between GRK4A and GRK4AS via tolerance for example E3

The error for all examples stay the same. In the second plot we see the dependence of the savings in computing time of the dimension of the linear equations and the structure of the matrices. The savings are the largest for high dimensional linear equations with full matrices.
For the other examples in STIFFDETEST we obtain similar or even better results. The reliability is better and we obtain savings in computing especially for high dimensional problems.
We have presented example E3 because it is one of the very few examples of the testset, where GRK4A can compute $S(t) \cdot R$ with reliable accuracy. In all the

other examples external difference approximations are not reliable enough or too accurate which makes it difficult to compare the computing times.

4.3 Conclusion

The method presented is more reliable with respect to the accuracy of the calculated sensitivity. In addition big savings in computing time are possible, especially for high dimensions if the matrix f_y is full.
A great advantage of this method is the possibility to apply this extension to many integration subroutines and also to methods for DAE's.

Availability of Software

All the presented subroutines are freely available. If interested in further details on the implementation or the source code please contact Stephan Franz at following adress or email:

Stephan Franz Technische Universität Darmstadt
Fachbereich Mathematik AG 8
Schloßgartenstraße 7
D-64289 Darmstadt
email: franz@mathematik.tu-darmstadt.de

References

[1] H. Bock: " Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen ", Bonner mathematische Schriften, Bonn, 1987

[2] H. Bock: " Numerical Treatment of Inverse Problems in Chemical Reaction Kinetics ", in K. Ebert, P. Deuflhard, W. Jäger (Eds.), " Modelling of Chemical Reaction Systems ", Vol. 18, Springer Series in Chemical Physics, Springer, Heidelberg, 1981

[3] H. Bock: " Derivative Free Methods for Parameter Identification in Nonlinear Differential Equations ", PhD thesis, Universität Bonn, 1981

[4] O. Buchauer, P. Hiltmann, M. Kiehl: " Sensitivity Analysis of Initial-Value-Problems with Application to Shooting Techniques ", Schwerpunktprogramm der DFG: anwendungsbezogene Optimierung und Steuerung, Report Nr. 403, 1992, (also Numerische Mathematik, Vol. 67, Nr. 2, 1994)

[5] W. H. Enright, T. E. Hull, B. Lindberg: " Comparing Numerical Methods for Stiff Systems of Ordinary Differential Equations ", BIT, Vol. 15, 1975, (10–48)

[6] S. Franz: " Sensitivitätsanalyse von Anfangswertproblemen bei gewöhnlichen Differentialgleichungen ", diploma thesis, Zentrum Mathematik, TU München, 1999

[7] E. Hairer, S. P. Nørsett, G. Wanner: " Solving Ordinary Differential Equations I ", 2. edition, Springer, 1992

[8] E. Hairer, G. Wanner: " Solving Ordinary Differential Equations II ", 2. edition, Springer, 1996

[9] A. Heim: " Parameteridentifikation in differentialalgebraischen Systemen ", diploma thesis, Mathematisches Institut der TU München, 1992

[10] T. E. Hull, W. H. Enright, B. M. Fellen, A. E. Sedgwick: " Comparing Numerical Methods for Ordinary Differential Equations ", SIAM Journal, Numerical Analysis, Vol. 9 No. 4, Dec. 1972, (617–621)

[11] M. Kiehl: " Sensitivity Analysis of Ordinary Differential Equations and Differential-Algebraic Systems – Theory and Implementation Guide ", TU München, Fakultät für Mathematik, Report TUM 5001, 1998

[12] T. Maly, L. Petzold: " Numerical Methods and Software for Sensitivity Analysis of Differential Algebraic Equations ", Applied Numerical Mathematics 20, 1996, (57–79)

[13] J. Stoer, R. Bulirsch: " Introduction to Numerical Analysis ", Springer, Berlin Heidelberg New York, 1980