Proceedings of the 9th International Symposium on
Dynamics and Control of Process Systems (DYCOPS 2010),
Leuven, Belgium, July 5-7, 2010
Mayuresh Kothare, Moses Tade, Alain Vande Wouwer, Ilse Smets (Eds.)

Adaptive Stepsize Control in Implicit Runge-Kutta Methods for Reservoir Simulation ⋆

Carsten Völcker [*] **John Bagterp Jørgensen** [*]
Per Grove Thomsen [*] **Erling Halfdan Stenby** [**]

[*] *Department of Informatics and Mathematical Modeling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail:* {cv,jbj,pgt}@imm.dtu.dk*)*
[**] *Department of Chemical and Biochemical Engineering, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail:* ehs@kt.dtu.dk*)*

Abstract: This paper concerns predictive stepsize control applied to high order methods for temporal discretization in reservoir simulation. The family of Runge-Kutta methods is presented and in particular the explicit singly diagonally implicit Runge-Kutta (ESDIRK) methods are described. A predictive stepsize adjustment rule based on error estimates and convergence control of the integrated iterative solver is presented. We try to improve the predictive stepsize control by smoothing the stepsize sequence through combining the control of error with the control of convergence.

Keywords: Reservoir simulation, Runge-Kutta methods, convergence control, stepsize selection.

1. INTRODUCTION

Reservoir simulators are computer programs that solve the equations for heat and mass flow in porous media. Numerical integration is one of the basic steps involved in the simulation process. The number and type of equations to be solved depend on the geological characteristics of the reservoir, the characteristics of the oil and the oil recovery process to be modeled. Choosing the appropriate method of integration involves deciding on factors such as the order of the integration scheme, stability properties and concern on computational efficiency. ESDIRK methods have been applied successfully for solution of convection-diffusion-reaction problems, see Kennedy and Carpenter (2003). This class of methods is computationally efficient, and both A- and L-stable and stiffly accurate ESDIRK methods of various order with an embedded method for error estimation have been derived by Kværnø (2004) and Jørgensen et al. (2008). In addition, a robust adaptive stepsize selection is essential to an efficient numerical integration. An adaptive stepsize selection aims to keep the error estimate bounded i.e. close to a user-specified tolerance by adjusting the timestep. Gustafsson (1992) suggested a strategy for stepsize selection based on the rigorous error estimates provided by embedded Runge-Kutta methods.
We have applied the controller by Gustafsson and Söderlind (1997) to three different ESDIRK methods used for solving a two-phase reservoir model. Although the control strategy has proven efficient we observed that certain steps were rejected due to irregularities in the stepsize selection. We found that a different interaction between the error and the convergence control in the stepsize selection process may solve this problem. The idea is to combine the control of error with control of convergence in the inner iterations in a simple logic that minimizes the number of rejected steps and thereby improves the efficiency.

2. DIFFERENTIAL EQUATION MODEL

In this section we briefly outline the two-phase flow problem and we present the typical formulation of a system of ordinary differential equations (ODE) based on conservation laws.

2.1 The two-phase flow problem

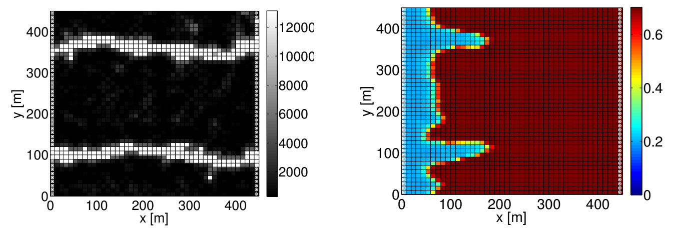
We consider immiscible two-phase flow of oil and water in porous media. Let $P_o = P_o(t, x)$ be the pressure of oil and $S_w = S_w(t, x)$ be the saturation of water, as function of time $t \geq 0$ and position $x \subset \mathbb{R}^2$, and let $C_w = C_w(P_o, S_w)$ and $C_o = C_o(P_o, S_w)$ be the mass concentrations of water and oil respectively. Then the mass balances for water and oil in the reservoir is expressed by the following system of partial differential equations

$$\frac{\partial}{\partial t}C_w = -\nabla \cdot F_w + Q_w \tag{1a}$$

$$\frac{\partial}{\partial t}C_o = -\nabla \cdot F_o + Q_o \tag{1b}$$

$F_w = F_w(P_o, S_w)$ and $F_o = F_o(P_o, S_w)$ are the fluxes of water and oil through the porous media. The source/sink

⋆ This research project is financially supported by the Danish Research Council for Technology and Production Sciences. FTP Grant no. 274-06-0284.



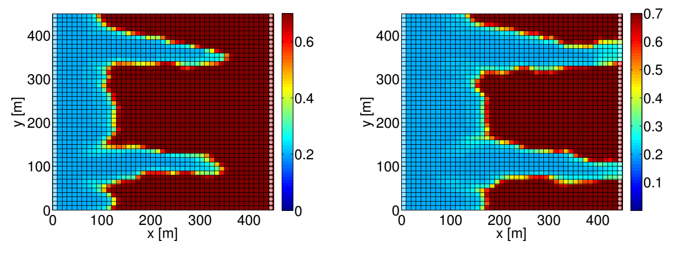


Fig. 1. Field development after 9 weeks of water injection.

terms of water and oil are denoted $Q_w = Q_w(P_o, S_w)$ and $Q_o = Q_o(P_o, S_w)$. They are used to describe the flow from injection wells and the flow to production wells. A more profound description can be found in Chen (2007) and Völcker et al. (2009). We use a standard 2-D problem defined by Brouwer and Jansen (2004), depicted in Figure 1.

2.2 An ODE system in general

Many process simulation problems in general are based on conservation of mass, energy and momentum. It is desirable to preserve such properties upon numerical integration in time. As proposed by Völcker et al. (2009) a general formulation of such an ODE system may be

$$\frac{d}{dt}g(x(t)) = f(t, x(t)) \quad x(t_0) = x_0 \tag{2}$$

where $x(t)$ denotes the system states, $g(x(t))$ are the properties conserved, while the right-hand side function $f(t, x(t))$ has the usual interpretation.

3. INTEGRATION METHODS

In this section different classes of Runge-Kutta methods are outlined. In particular ESDIRK methods are described.

3.1 Runge-Kutta Integration

An s-stage Runge-Kutta method for integration of (2) can be expressed as

$$T_i = t_n + h_n c_i \qquad i \in \mathbb{S}_1 \tag{3a}$$

$$g(X_i) = g(x_n) + h_n \sum_{j=1}^{s} a_{ij} f(T_j, X_j) \quad i \in \mathbb{S}_1 \tag{3b}$$

$$g(x_{n+1}) = g(x_n) + h_n \sum_{j=1}^{s} b_j f(T_j, X_j) \tag{3c}$$

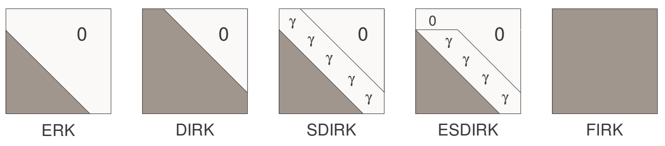


Fig. 2. The A-matrix of Runge-Kutta methods.

where T_i and X_i are the internal stage values being numerical approximations to $x(T_i)$. x_{n+1} is the step computed at $t_{n+1} = t_n + h_n$. The set $\mathbb{S}_i$ denotes the internal stages $i, i+1, \ldots, s$.
Different classes of Runge-Kutta methods can be obtained depending on the structure of the matrix $A = [a_{ij}]$. This is illustrated in Figure 2. Explicit Runge-Kutta (ERK) methods have a strictly lower triangular A-matrix which allows (3b) to be solved explicitly without iterations. Therefore, ERK methods are computationally fast but cannot be applied to stiff problems because of poor stability properties. All implicit methods are characterized by an A-matrix that is not strictly lower triangular and the state values X_i are computed iteratively by solution of (3b). Fully implicit Runge-Kutta (FIRK) methods, identified by a full A-matrix, have excellent stability properties making them usefull for solving stiff systems of ODE's. However, the excellent stability properties comes with high computational cost in solving (3b) simultaneously at each iteration step. To achieve some of the stability properties of the FIRK methods but at lower computational cost, various methods in between the ERK and the FIRK methods have been constructed.

3.2 ESDIRK Methods

ESDIRK methods have a lower triangular A-matrix. By construction they retain the stability properties of FIRK methods but at significant lower computational cost. Because $c_1 = 0$ and $a_{11} = 0$ the first stage in ESDIRK methods is explicit implying that the first state value equals the last step $(T_1, X_1) = (t_n, x_n)$. The subsequent stages are singly diagonally implicit such that the state values X_i at $T_i = t_n + h_n c_i$ for $i \in \mathbb{S}_2$ may be solved sequentially by solution of the residual

$$R(T_i, X_i) = g(X_i) - h_n \gamma f(T_i, X_i) - \psi_i = 0 \quad i \in \mathbb{S}_2 \tag{4}$$

with the term

$$\psi_i = g(x_n) + h_n \sum_{j=1}^{i-1} a_{ij} f(T_j, X_j) \quad i \in \mathbb{S}_2 \tag{5}$$

using Newton-Raphson's iterative method. The Jacobian $J_R(T_i, X_i) = \frac{\partial}{\partial X_i} R(T_i, X_i)$ of the residual is

$$J_R(T_i, X_i) = J_g(X_i) - h_n \gamma J_f(T_i, X_i) \quad i \in \mathbb{S}_2 \tag{6}$$

where $J_g(X_i) = \frac{d}{dX_i} g(X_i)$ and $J_f(T_i, X_i) = \frac{\partial}{\partial X_i} f(T_i, X_i)$ are the Jacobiants of the right- and left-hand sides of (2) respectively. We only consider methods assumed to be stiffly accurate by construction i.e. $c_s = 1$ and $a_{sj} = b_j$ for $j \in \mathbb{S}_1$. This implies that the quadrature function (3c) corresponds to the last internal stage in (3b). Consequently the next step equals the last state value $(t_{n+1}, x_{n+1}) = (T_s, X_s)$. The Butcher tableau for stiffly accurate ESDIRK methods is represented in (7).

$$\begin{array}{c|cccccc} 0 & 0 \\ c_2 & a_{21} & \gamma \\ c_3 & a_{31} & a_{32} & \gamma \\ \vdots & \vdots & & & \ddots \\ 1 & b_1 & b_2 & b_3 & \cdots & \gamma \\ \hline x_{n+1} & b_1 & b_2 & b_3 & \cdots & \gamma \\ \hat{x}_{n+1} & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \cdots & \hat{b}_s \\ \hline e_{n+1} & d_1 & d_2 & d_3 & \cdots & d_s \end{array} \tag{7}$$

4. ERROR AND CONVERGENCE CONTROL

In this section we describe how to estimate the integration error, how the error is related to the user specified tolerances and how to control the convergence of the iterative solver.

4.1 Integration error

The ESDIRK method stated in (7) is equipped with an embedded Runge-Kutta method

$$g(\hat{x}_{n+1}) = g(x_n) + h_n \sum_{j=1}^{s} \hat{b}_j f(T_j, X_j) \tag{8}$$

computing the embedded solution $\hat{x}_{n+1}$. The embedded method is of different order, which then provides an estimate of the local truncation error

$$e_{n+1} = g(x_{n+1}) - g(\hat{x}_{n+1}) = h_n \sum_{j=1}^{s} d_j f(T_j, X_j) \tag{9}$$

corresponding to the numerical solution x_{n+1}. The integration error (9) is controlled adjusting the timestep by monitoring the root mean square of the error-tolerance relation

$$r_{n+1} = \frac{1}{\sqrt{m}} \left\| \frac{e_{n+1}}{atol + |g(x_{n+1})| rtol} \right\|_2 \tag{10}$$

where $atol$ and $rtol$ are componentwise user specified absolute and relative error tolerances and m is the dimension of the solution vector. Only stepsizes for which $r_{n+1} \leq 1$ are accepted.

4.2 Convergence control

The solution of (4) is done iteratively by a modified Newton-Raphson's method i.e. the Jacobian of the residual is not evaluated/factorized at each timestep. There is always a trade-off between the rate of convergence of the equation solver and the frequency of Jacobian updates/factorizations. For reasons of robustness the convergence rate is measured by the residuals Houbak et al. (1985)

$$\alpha_i = \frac{(r_R)_i^{k-1}}{(r_R)_i^k} \quad i \in \mathbb{S}_2 \tag{11}$$

where the iteration error of the k^{th} iteration is computed as the root mean square of the residual-tolerance relation

$$(r_R)_i^k = \frac{1}{\sqrt{m}} \left\| \frac{(R(T_i, X_i))^k}{atol + |(g(X_i))^k| rtol} \right\|_2 \quad i \in \mathbb{S}_2 \tag{12}$$

using the same componentwise absolute and relative error tolerances as in (10). If for some k during the iterations $\alpha \geq 1$ the iteration sequence is terminated and the stepsize is restricted. In case of convergence the iterations are successfully stopped when $(r_R)_i^k \leq \tau$. As noticed in Hairer and Wanner (1996) the choice of τ affects the efficiency of the algorithm. A large value of τ may lead to one or more large components in the integration error (10) with too many rejected steps as a result. We have chosen $\tau = 0.1$ as a compromise between robustness and computational speed.

5. STEPSIZE SELECTION

This section is divided into a brief description of the stepsize selection rule adopted, a description of the modifications that we suggest in order to simplify and stabilize the control algorithm and finally an outline of the complete controller is presented.

5.1 Predictive control

The integration error is controlled using a predictive controller for stepsize selection as presented by Gustafsson (1992). The controller must keep the estimate (9) of the local truncation error bounded and minimize the computational work in the solution process by trying to keep $r_{n+1} = 1$ by maximizing the stepsize. Based on empirical evidence Gustafsson (1992) suggested a proportional integral (PI) stepsize adjustment rule on the form

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{r_{n-1}}{r_n} \right)^{k_1/\hat{k}} \left(\frac{\epsilon}{r_n} \right)^{k_2/\hat{k}} h_n \tag{13}$$

where k_1 and k_2 are the gain parameters of the proportional and the integral parts respectively and $\hat{k}$ is the order of the embedded Runge-Kutta method, while ϵ is the desired tolerance (including a safety factor). Gustafsson (1992) suggests $k_1 = k_2 = 1$ corresponding to deadbeat control and a safety factor of 0.8.

5.2 Modified controller

The core stepsize adjustment rule (13) must be implemented along with a number of extensions and various safety nets and the original framework from which we propose our modifications can be found in Gustafsson (1992). Additionally a modification suggested by Gustafsson and Söderlind (1997) is described and implemented. The modified PI controller that we suggest is presented in Algorithm 5.1.

Since we are only considering stiffly accurate methods the order reduction for stiff systems can be avoided, see Prothero and Robinson (1974). Consequently the strategy described by Gustafsson (1992) for estimating $\hat{k}$ after successive rejects can be omitted. This does not make any noticeable change in the controller performance but simplifies the algorithm a great deal.
Besides the frequency of Jacobian updates/factorizations the stepsize is the only available control variable affecting the convergence rate of the equation solver. In order to assure convergence in the equation solver the stepsize has to be restrained in some situations. If convergence is too slow i.e. if $\alpha > \alpha_{ref}$ Gustafsson (1992) suggest the stepsize to be chosen as

$$h_\alpha = \frac{\alpha_{ref}}{\alpha} h_n \tag{14}$$

to obtain $\alpha = \alpha_{ref}$ in the next step. The stepsize suggested by (14) must be coordinated with the requirements from the error control. If $\alpha > \alpha_{ref}$ the stepsize in Gustafsson (1992) is implemented as

$$h_{n+1} = \min(h_r, h_\alpha) \tag{15}$$

restraining the stepsize if $h_\alpha < h_r$. The strategy adopted by (14) and (15) may be too aggressive in the sense that the corresponding error estimate (10) becomes very low compared to ϵ. Hence the subsequent stepsize estimated by the asymptotic controller

$$h_r = \left(\frac{\epsilon}{r}\right)^{1/\hat{k}} h_n \tag{16}$$

will be too large making the error estimate and thereby the stepsize fluctuate wildly. We try to avoid this by modifying (14) to

$$h_\alpha = \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h_n \tag{17}$$

which means that the deviation of the convergence rate from α_{ref} is not necessarily corrected in one step. If a step has been rejected and restricted by slow convergence, then in combination (16) is filtered by the relation between the previous accepted step and the current accepted step

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{\epsilon}{r_n}\right)^{1/\hat{k}} h_n \tag{18}$$

which further reduces the stepsize following a convergence restricted step. If the current step is accepted we neglect the condition $\alpha > \alpha_{ref}$ on (15). In addition (13) is always used estimating the next stepsize, whenever a step is accepted. Consequently we allow the convergence of the equation solver to gain more influence on the stepsize selection. While Gustafsson (1992) suggests $0.2 \lesssim \alpha_{ref} \lesssim 0.5$ as set-points for the convergence rate we chose $\alpha_{ref} = 0.6$. This value favours robustness and a minimum amount of work needed to complete the integration fairly equal.
Slow convergence in the equation solver and in particular rejected steps because of convergence failure is very costly. This can to some extend be controlled by the stepsize but also by the frequency of Jacobian updates/factorizations. Considering (6) we see that stepsize changes invokes a refactorization of the Jacobian but not necessarily a Jacobian reevaluation - if on the other hand the Jacobian is updated a factorization is always called for. Good convergence can be obtained by both updating and factorizing the Jacobian at every stepsize change. For large systems though this may be the dominating part of the computations and large savings can be made by utilizing a strategy for reusing the same Jacobian for several timesteps. Gustafsson (1992) monitors the relative stepsize change since the last factorization was done and suggests

$$|h_{n+1} - h_{LU}|/h_{LU} > \alpha_{LU} \tag{19}$$

as a refactorization strategy. The strategy is preventive in the sense that it tries to avoid convergence failures by factorizing whenever planning to do a stepsize change that is likely to jeopardize convergence. Should poor convergence be experienced despite a factorization based on current data, say $\alpha > \alpha_{Jac}$, then a reevaluation of the Jacobian is called for. Gustafsson and Söderlind (1997) suggests the combination

$$\alpha - |h_{n+1} - h_{LU}|/h_{LU} > \alpha_{Jac} \tag{20}$$

as decision for when to compute a new Jacobian. Besides monitoring the convergence rate of the equation solver this strategy also trades Jacobian updates with factorizations and function evaluations. The value of α_{ref} sets an upper limit on α_{Jac} and α_{LU}, see Gustafsson (1992). In the two-phase flow problem the administration of the Jacobian is expensive compared to one iteration. This argues for large values of α_{Jac} and α_{LU}. To be more specific it is more costly to update the Jacobian than factorizing it, consequently we have chosen $\alpha_{Jac} = 0.5$ and $\alpha_{LU} = 0.3$. Because of the large value of α_{Jac} we allow a fairly large maximum number of iterations in the equation solver, setting $k_{\max} = 20$.

5.3 The complete controller

The complete modified PI controller for an implicit Runge-Kutta method is outlined in Algorithm 5.1. The controller includes three main parts:

- A stepsize selection rule based on both the error-tolerance relation and the convergence of the equation solver.
- An update/factorization strategy for the Jacobian that supervises the convergence and the iteration error of the equation solver.
- A strategy for handling convergence failures.

6. CHOICE OF METHODS

In this section the two-phase flow problem is used as a benchmark. We compare and discuss the performance of the controller by Gustafsson and Söderlind (1997) and the controller suggested in Algorithm 5.1 when applied to three different ESDIRK methods.
In this section ESDIRK$k\hat{k}$ refers to an ESDIRK method of

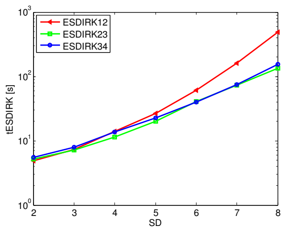
Algorithm 5.1: The complete modified PI controller for an implicit Runge-Kutta method.

```
if *iterations converged* then
    $h_r \leftarrow \left(\frac{\epsilon}{r}\right)^{1/\hat{k}} h$
    if *step accepted* then
        if *step restricted* then
            $h_r \leftarrow \frac{h}{h_{acc}} h_r$
        else
            $h_r \leftarrow \frac{h}{h_{acc}} \left(\frac{r_{acc}}{r}\right)^{1/\hat{k}} h_r$
        $r_{acc} \leftarrow r$
        $h_{acc} \leftarrow h$
    $h \leftarrow \min\left(h_r, \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h\right)$
    if $\alpha - |h - h_{LU}|/h_{LU} > \alpha_{Jac}$ then
        Form new Jacobian and factorize iteration matrix.
        $h_{LU} \leftarrow h$
    else if $|h - h_{LU}|/h_{LU} > \alpha_{LU}$ then
        Factorize iteration matrix.
        $h_{LU} \leftarrow h$
else
    if *new Jacobian* then
        if $\alpha > \alpha_{ref}$ then
            $h \leftarrow \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h$
        else
            $h \leftarrow h/2$
        Step restricted.
    else
        Form new Jacobian.
    Factorize iteration matrix.
    $h_{LU} \leftarrow h$
```

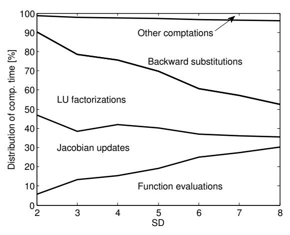
order k with an embedded method for error estimation of order $\hat{k}$. PI97 denotes the controller by Gustafsson and Söderlind (1997) and PI09 refers to the controller presented in Algorithm 5.1. For the work-precision diagrams we used a fixed absolute tolerance of 10^{-8} and the relative tolerances from 10^{-2} to 10^{-8}, denoted as significant digits (SD).

6.1 *Choice of ESDIRK method*

As can be seen from the work-precision diagram in Figure 3(a), the computational cost of ESDIRK12 increases dramatically with the requirement in SD's. This is due to the small stepsizes, which yields an increased workload of the equation solver trying to retain $(r_R)_i^k \leq \tau$ (12). This is seen in Figure 4(a), where the number of function evaluations reflects the number of iterations. ESDIRK23 and ESDIRK34 are better at maintaining an appropriate distribution of the workload as the requirements of the number in SD's increases. The distribution of workload of the two methods are almost identical and only the distribution of ESDIRK23 is depicted in Figure 3(b).
Except for SD = 2 we observe from the work-precision diagram of the three methods, that ESDIRK23 is the most computationally efficient method for temporal discretization of problems like the two-phase flow.

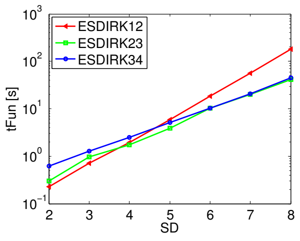


(a) Computational cost of the three ESDIRK methods.

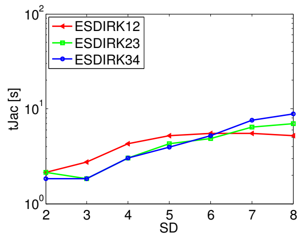


(b) Distribution of computational cost in ESDIRK23.

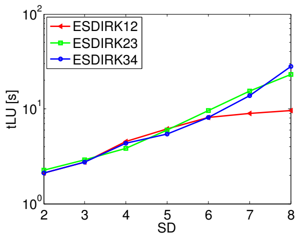
Fig. 3. Total computational cost of the three ESDIRK methods applied with the PI09 controller and the distribution of the computational cost in ESDIRK23.



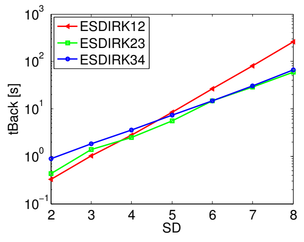
(a) Computational cost of function evaluations.



(b) Computational cost of Jacobian updates.



(c) Computational cost of LU factorizations.

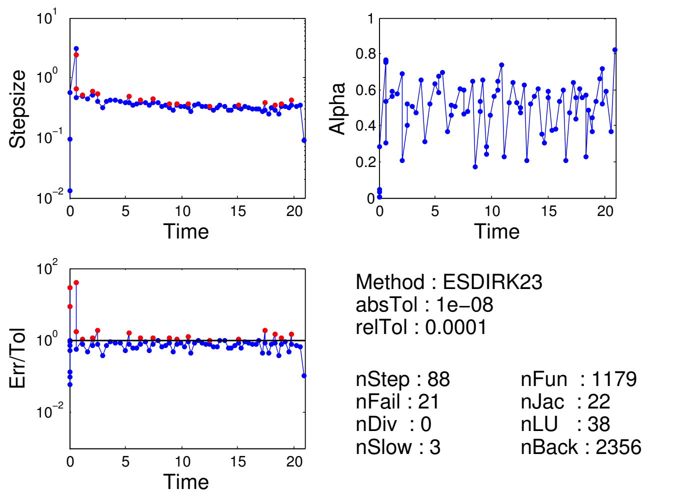


(d) Computational cost of back substitutions.

Fig. 4. Performance comparison of the three ESDIRK methods applied with the PI09 controller.

6.2 *Choice of controller*

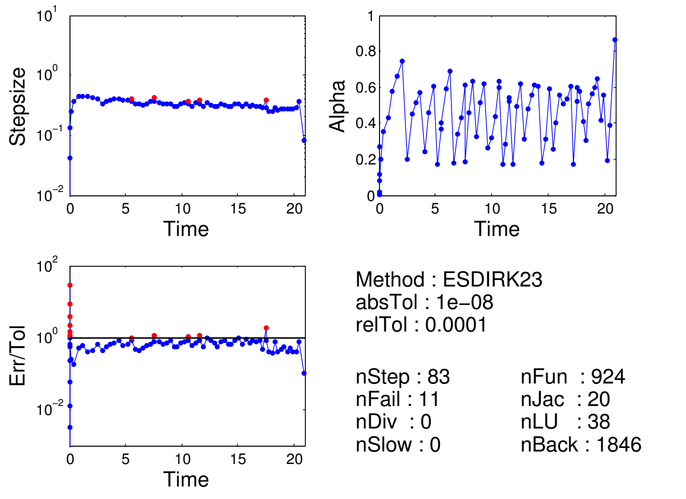
The stepsize sequences for the PI97 and the PI09 controllers are depicted in Figure 5 and 6 respectively. As expected, we observe a reduction in rejected steps (nFail and nSlow) and fewer iterations done by the equation solver (nFun). In the PI97 controller, convergence is only allowed to restrict the stepsize, if $\alpha > \alpha_{ref}$. The PI09 controller allows convergence to restrict the stepsize by combining (15) and (17), hence the relation between α and α_{ref} is taken into account in each stepsize selection. Due to this improved interaction between the error and the convergence control in the stepsize selection process, large fluctuations of the stepsize, when advancing in time, is avoided. Consequently a smoother stepsize sequence is obtained and the need for heuristics to restrain large stepsize changes no longer applies.
As seen in Figure 7, it is difficult to make a general conclusion of the difference in computational cost for the two controllers. Typically we require 3 to 4 SD's in reservoir simulation, as a consequence we suggest applying the PI09 controller, when solving problems like the two-phase flow.



```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 88        nFun  : 1179
nFail : 21        nJac  : 22
nDiv  : 0         nLU   : 38
nSlow : 3         nBack : 2356
```

Fig. 5. Performance of ESDIRK23 applied with the PI97 controller computing the solution in Figure 1(b).



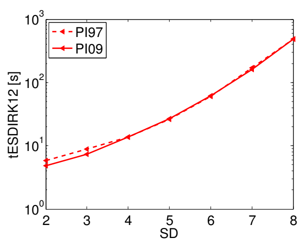
```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 83        nFun  : 924
nFail : 11        nJac  : 20
nDiv  : 0         nLU   : 38
nSlow : 0         nBack : 1846
```

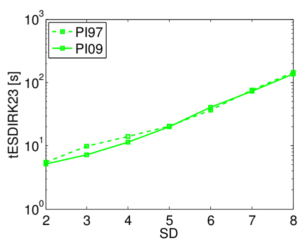
Fig. 6. Performance of ESDIRK23 applied with the PI09 controller computing the solution in Figure 1(b).

7. CONCLUSION

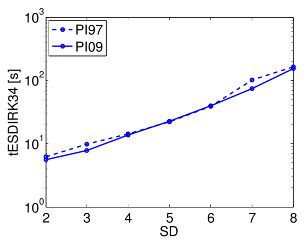
In this paper we combined the control of error with the convergence control of the equation solver in a simple logic that decreases the number of rejected steps and produces a smoother stepsize sequence. In some cases, better convergence of the equation solver is obtained i.e. fewer iterations is needed in order to meet the required tolerance. For large scale systems, which is typical in reservoir simulation, it may be necessary to solve the linearized equations iteratively. If this is the situation, the cost per iteration, both for the equation solver and the iterative solver of the linearized system, can be significant. Consequently, it is crucial for the solution of large scale systems to minimize the number of iterations per timestep, when performing implicit numerical integration.
In addition, the integration of the convergence control has the effect that extreme variations in stepsize are eliminated making the logics in the control algorithm free of heuristics.



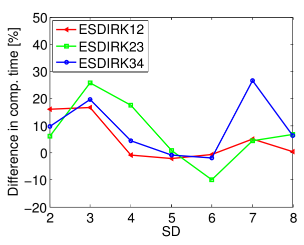
(a) Computational cost of ESDIRK12.



(b) Computational cost of ESDIRK23.



(c) Computational cost of ESDIRK34.



(d) Difference in computational cost.

Fig. 7. Comparison of the PI97 and the PI09 controller applied to the three ESDIRK methods.

REFERENCES

Brouwer, D.R. and Jansen, J.D. (2004). Dynamic optimization of waterflooding with smart wells using optimal control theory spe-78278-pa. *The 2002 SPE European Petroleum Conference.*

Chen, Z. (2007). *Reservoir Simulation : Mathematical Techniques in Oil Recovery.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

Gustafsson, K. (1992). *Control of Error and Convergence in ODE Solvers.* Ph.D. thesis, Department of Automatic Control, Lund University, Sweden.

Gustafsson, K. and Söderlind, G. (1997). Control strategies for the iterative solution of nonlinear equations in ode solvers. *SIAM J. Sci. Comput.*, 18(1), 23–40.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.* Springer, 2nd edition.

Houbak, N., Nörsett, S., and Thomsen, P. (1985). Displacement or residual test in the application of implicit methods for stiff problems. *IMA Journal of Numerical Analysis*, 5(3), 297–305.

Jørgensen, J.B., Kristensen, M.R., and Thomsen, P.G. (2008). A family of esdirk integration methods. *SIAM Journal on Scientific Computing.*

Kennedy, C.A. and Carpenter, M.H. (2003). Additive runge-kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1-2), 139 – 181.

Kværnø, A. (2004). Singly diagonally implicit runge-kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44, 489 – 502.

Prothero, A. and Robinson, A. (1974). On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 28(125), 145–162.

Völcker, C., Jørgensen, J.B., Thomsen, P.G., and Stenby, E.H. (2009). Simulation of subsurface two-phase flow in an oil reservoir. *The European Control Conference 2009.*