Online Checkpointing for Parallel Adjoint Computation in PDEs: Application to Goal-Oriented Adaptivity and Flow Control

Vincent Heuveline[1] and Andrea Walther[2]

[1] Universität Karlsruhe, Institute for Applied Mathematics, Karlsruhe, Germany `vincent.heuveline@math.uni-karlsruhe.de`

[2] Technische Universität Dresden, Institute of Scientific Computing, Dresden, Germany `Andrea.Walther@tu-dresden.de`

Abstract. The computation of derivatives for the optimization of time-dependent flow problems is based on the integration of the adjoint differential equation. For this purpose, the knowledge of the complete forward solution is required. Similar information is needed for a posteriori error estimation with respect to a given functional. In the area of flow control, especially for three dimensional problems, it is usually impossible to store the full forward solution due to the lack of memory capacities. Additionally, adaptive time-stepping procedures are needed for efficient integration schemes in time. Therefore, standard optimal offline checkpointing strategies are usually not well-suited in that framework.

We present a new online procedure for determining the checkpoint distribution on the fly. Complexity estimates and consequences for storing and retrieving the checkpoints using parallel I/O are discussed. The resulting checkpointing approach is integrated in HiFlow, a multipurpose parallel finite-element package with a strong emphasis in computational fluid dynamic, reactive flows and related subjects. Using an adjoint-based error control for prototypical three dimensional flow problems, numerical experiments demonstrate the effectiveness of the proposed approach.

1 Introduction

In time-dependent flow control as well as in the framework of goal-oriented a posteriori error control, the calculation of adjoint information forms a basic ingredient to generate the required derivatives of the cost functional (see e.g. [8]). However, the corresponding computations may become extremely tedious if possible at all because of the sheer size of the resulting discretized problem as well as its nonlinear character, which requires keeping track of the complete forward solution to be able to integrate the corresponding adjoint differential equation backwards. This fact still forms a main bottleneck in the overall optimization process despite the ever-growing size of memory devices. For that reason, several checkpointing techniques have been developed. Here, only a few intermediate states are stored as checkpoints. Subsequently, the required forward information is recomputed piece by piece from the checkpoints according to the adjoint calculation. Hence, checkpointing methods seek for an acceptable compromise between memory requirements and run time increase due to re-computations that cannot be avoided.

If the number of time steps for integrating the differential equation describing the state is known a priori, one very popular checkpointing strategy is to distribute the checkpoints equidistantly over the time interval. However, it was shown in [18] that this approach is not optimal. One can compute optimal checkpointing schedules in advance to achieve for a given number of checkpoints an optimal, i.e. minimal, run time increase [7]. This procedure is referred to as offline checkpointing and implemented in the package revolve [7]. However, in the context of flow control, the partial differential equations to be solved are usually stiff, and the solution process relies therefore on some adaptive time stepping procedure. Hence, the number of time steps performed is known only after the complete integration. This fact makes an offline checkpointing intractable. Instead, one may apply a straightforward checkpointing by placing a checkpoint each time a certain number of time steps has been executed. This transforms the uncertainty in the number of time steps to a uncertainty in the number of checkpoints needed. This approach is used by CVODES [17]. However, when the amount of memory per checkpoint is very high one certainly wants to determine the number of checkpoints required a priori. For that purpose, we propose a new procedure for online checkpointing that distributes a given number of checkpoints during the integration procedure. This new approach yields a time-optimal adjoint computation for a given number of checkpoints. The present paper focus on practical aspects, i.e. the specific online checkpointing algorithm and the consequences for the computation of adjoint information on parallel computers. Furthermore, it describes the coupling of the presented online checkpointing software with the package HiFlow (see www.hiflow.de) for the parallel computation of adjoints. A companion paper [10] concentrates on the theoretical aspects of the goal-oriented adaptivity and the online checkpointing algorithm.

The outline of this paper is as follows. Section 2 is dedicated to the derivation of adjoint-based a posteriori error control for flow problems and its link to flow control. The new online checkpointing strategy is presented in Section 3. Here also complexity estimates for the resulting checkpointing strategy and the usage of the algorithm on parallel computers are addressed. First numerical experiments are presented in Section 4. Finally, conclusions are drawn in Section 5.

2 Adjoint Based Techniques for Error Estimation

2.1 Problem Formulation

Let u denote the state variables, g the control variables, $J(u, g)$ the objective functional, and $G(u, g) = 0$ the constraints. A standard formulation for the related optimization problem reads

Problem 1: Find controls g and states u such that $J(u, g)$ is minimized subject to $G(u, g) = 0$.

Our goal in this paper is to address the case where the constraints are defined by means of time-dependent partial differential equations. Even though the derived method is very general, we concentrate on the case of instationary, incompressible, viscous flows modeled by means of the Navier-Stokes equations, i.e., ignoring the controls g we have

$$\frac{\partial u}{\partial t} - \nu \Delta u + u \cdot \nabla u + \nabla p = f \qquad \text{in } (0,T) \times \Omega, \tag{1}$$

$$\nabla \cdot u = 0 \quad \text{in } (0,T) \times \Omega, \qquad u|_{t=0} = u_0, \tag{2}$$

where $u \in \mathbb{R}^d$ describes the velocity field and $p \in \mathbb{R}$ the pressure. We assume that the velocity field is subject to adequate boundary conditions. A standard approach to solve such problems is based on the solution of an adjoint system backward in time to compute the gradient of the functional $J(u,g)$ (see e.g. [8]). The state variables appear in the coefficients and right-hand sides of the adjoint equations and must be available as the solver marches backward in time. Generally for flow control problems the storage of the state variables for every time step results in a huge amount of data. Therefore, we propose a checkpointing technique that relies on the storage of a few selected time steps. One then recomputes the information required by the adjoint calculation time step per time step.

Similarly to Problem 1, the proposed framework for checkpointing can be used in the context of goal-oriented a posteriori error estimation for time-dependent problems. Again one considers a state equation defined by partial differential equations $F(u)$. We suppose that these equations are discretized by means of a Galerkin method (e.g. finite-element method) and that the corresponding discrete solution is denoted by u_h. The goal is to determine the discretization error with respect to some functional $J(\cdot)$, i.e. $J(u) - J(u_h)$. This problem can be formulated as a control problem similar to Problem 1. In the remainder of this paper we will consider this setup for the derivation of the proposed checkpointing strategy.

2.2 A General Paradigm for Dual-Based a Posteriori Error Estimation

In this section, we outline the concepts related to dual-based error estimation following the general paradigm introduced in Eriksson et al. [3]. We refer to Machiels et al. [13], Oden and Prudhomme [14], and Giles [6] for related approaches to goal-oriented error estimation.

Let $A(\cdot;\cdot)$ be a differentiable semi-linear form and $F(\cdot)$ a linear functional defined on some function space V. For $u \in V$, $A'(u;v)(\cdot)$ denotes the directional derivative of $A(u;\cdot)$ in the v direction. The second derivative of $A(u;\cdot)$ is refereed to by $A''(\cdot;\cdot)(\cdot,\cdot)$. We seek a solution $u \in V$ to the variational equation

$$A(u;\varphi) = F(\varphi) \qquad \forall \varphi \in V. \tag{3}$$

This problem is approximated by a *Galerkin method* using a sequence of finite dimensional subspaces $V_h \subset V$ parameterized by a parameter h. The corresponding discrete problem seeks $u_h \in V_h$ satisfying

$$A(u_h;\varphi_h) = F(\varphi_h) \qquad \forall \varphi_h \in V_h. \tag{4}$$

We assume that equations (3) and (4) possess unique solutions. A key feature of the discrete problem (4) is the property of *Galerkin orthogonality,* which reads in the general nonlinear case

$$A(u;\varphi_h) - A(u_h;\varphi_h) = 0 \qquad \forall \varphi_h \in V_h. \tag{5}$$

Suppose that the quantity $J(u)$ has to be computed, where $J(\cdot)$ is a differentiable functional defined on V. To control the error with respect to the functional J we introduce the following dual problem

$$\begin{aligned} A'(\overline{uu_h};\varphi)(\hat{z}) &= J'(\overline{uu_h})(\varphi) \qquad \forall \varphi \in V, \qquad \text{where} \qquad (6)\\ A'(\overline{uu_h};\varphi)(\psi) &= \int_0^1 A'(su + (1-s)u_h;\varphi)(\psi)\,ds,\\ J'(\overline{uu_h})(\varphi) &= \int_0^1 J'(su + (1-s)u_h)(\varphi)\,ds. \end{aligned}$$

We assume that (6) possesses a solution. Based on the dual solution $\hat{z}$ and due to the Galerkin orthogonality (5), we obtain the following error representation

$$\begin{aligned} J(u) - J(u_h) &= A'(\overline{uu_h};e)(\hat{z}) = A(u;\hat{z}) - A(u_h,\hat{z})\\ &= A(u;\hat{z}-\hat{z}_h) - A(u_h;\hat{z}-\hat{z}_h)\\ &= F(\hat{z}-\hat{z}_h) - A(u_h;\hat{z}-\hat{z}_h) = \rho(u_h,\hat{z}-\hat{z}_h) \end{aligned}$$

for any $\hat{z}_h \in V_h$, where $\rho(u_h,\cdot) = F(\cdot) - A(u_h;\cdot)$ describes the *primal* residual, and $e := u - u_h$. In practice, the previously derived error representation cannot be used directly since the adjoint problem (6) involves the unknown solution u. One alternative is to replace the exact solution u by its approximation u_h in the adjoint problem (6). The resulting adjoint problem reads

$$A'(u_h;\varphi)(z) = J'(u_h;\varphi) \qquad \forall \varphi \in V. \tag{7}$$

One can show that the following modified error representation holds

$$J(u) - J(u_h) = \rho(u_h, z - z_h) + R, \tag{8}$$

for any $z_h \in V_h$, where the remainder term R depends on the second order derivatives of $A(\cdot;\cdot)$ and $J(\cdot)$. The remainder term vanishes if $A(\cdot;\cdot)$ and $J(\cdot)$ are linear.

From now on, we consider procedures based on the error representation (8) for the a posteriori error control with respect to the functional J. The remainder term is neglected since, in our context, it involves higher order terms with respect to the discretization parameter h which can be omitted for h small enough.

The solution of the dual problem (7) needed for the error representation related to (8) corresponds in our context of time-dependent problems to the adjoint problem which has to be solved backward in time.

2.3 Galerkin Discretization in Time and Space

We consider a discretization of the problem (1)-(2) using a Galerkin finite element discretization simultaneously in space and in time. This setup allows us to rely on the error representation (8) for the error control. Following the lines of Eriksson and Johnson [4,5] we consider the dG(r)-method for the time discretization, i.e. we allow discontinuous functions in time. This discontinuity can be used to decouple the considered system on each subinterval $I_n = (t_{n-1}, t_n]$ of the time interval $(0, T]$, where $0 = t_0 < \cdots < t_n < \cdots < t_N = T$, $k_n = t_n - t_{n-1}$. For simplicity, we consider for each time step t_n a unique regular spatial mesh. Then, we can write the solution process as a standard time-stepping scheme. For $r = 0$, the corresponding dG(0)-method is equivalent to the backward-Euler scheme.

The Galerkin space discretization using conforming mixed finite elements with continuous pressure is based on a variational formulation of the Navier-Stokes equations (1)-(2). For this purpose, we employ standard Hood-Taylor finite elements [12] for the trial and test spaces (for a detailed description see, e.g., [1]). This choice for the trial and test functions guarantees a stable approximation of the pressure since the Babuska-Brezzi inf-sup stability condition is satisfied uniformly in h (see [2] and references therein). The advantage, when compared to equal order function spaces for the pressure and the velocity, is that no additional stabilization terms are needed.

Based on this space discretization, the arising nonlinear algebraic systems are then solved implicitly in a fully coupled manner by means of a damped Newton method. The linear subproblems are solved by the Generalized Minimal Residual Method (GMRES) (see [15]) preconditioned by means of a geometric multigrid iteration (see [19]). Two specific features characterize the scheme we consider: varying orders of the FEM ansatz on the mesh hierarchy and a Vanka-type smoother. This somewhat technical part is described in full detail in [9].

3 Online Checkpointing Algorithms

Having a fixed number of checkpoints to store intermediate states but an unknown number of time steps for which the adjoint has to be computed on the base of the forward trajectory, one has to decide on the fly, i.e., during the forward integration, where to place the checkpoints. Hence, without knowing how many time steps are left to perform, one has to analyze the current distribution of the checkpoints. Depending on the time steps performed so far, one may then discard the contents of one checkpoint to store the current available state. Obviously, one may think that this procedure could not be optimal since it may happen that one reaches the final time just after replacing a checkpoint, in which case another checkpoint distribution may be advantageous. A surprising efficient heuristic strategy to rearrange the checkpoints is implemented by the online procedure arevolve [11]. Here, a checkpoint distribution is judged by computing an approximation of the overall re-computation cost caused by the current distribution. This number is compared with an approximation of the re-computation cost if one resets a checkpoint to the currently available state. Despite the fact that significant simplifications are made for approximating the required re-computations, the resulting checkpointing schemes are comparatively cheap. Naturally, the optimal cost can be computed only afterwards when the number of time steps is known.

3.1 Optimal Online checkpointing

However, a main drawback of arevolve is that it is not possible to prove an upper bound on the deviation from the optimal checkpointing schedule because a heuristic is used to judge the current checkpointing distributions. In this paper, we present online checkpointing strategies for an a priori unknown number l of time steps and a given number of checkpoints c under the assumption that

$$l \leq \binom{c+2}{c} = \frac{(c+2)(c+1)}{2} = \sum_{i=1}^{c+1} i \equiv b_c \,. \tag{9}$$

Hence, the upper bound b_c on the number of time steps is directly determined by the number of checkpoints c. Let $F_l(x)$ denote the execution of the lth time step corresponding to the discretized PDE. Using p as a pointer to the next state where a checkpoint is set and s as a flag if a checkpoint has to be set, the proposed online checkpointing procedure reads as follows:

Algorithm 1: Online Checkpointing Algorithm

```
**Start:** Set $i = 0$, $o = c$, $p = c$, $s = 1$
**for** $l = 0, 1, \dots$
    1. Evaluate $x_{l+1} = F_l(x_l)$
    2. **If** termination criterion fulfilled **then** start reversal
       **If** $s = 1$ **then**
           Store state $x_l$ in checkpoint $i$
           $i = i + 1$
           **If** $i > o$ **then** $i = 1$
    3. **If** $l + 1 = p$ **then** $s = 0$
    4. **If** $l = p$ **then**
           $p = p + o$, $o = o - 1$, $i = o$
           **If** $o > 0$ **then** $s = 1$ **else** $s = 0$
    5. **If** $l = p$ **and** $o = -1$ **then** error: $l > b_c$
```

For a given value of c, this algorithm stores the states $0, \dots, c-1$ in the checkpoints $0, \dots, c-1$. Subsequently, the state $c+1$ is copied to the checkpoint $c-1$. Then the states $c+2, \dots, 2c-1$ are stored in the checkpoints $1, \dots, c-2$ by overwriting the information already contained in these memory pads. This process continues until either the termination criterion is fulfilled or the number of time steps exceeds the upper bound b_c. If a reversal is started in step 2, the optimal offline checkpointing provided by revolve is applied. Analyzing the described online checkpointing in more detail, we can prove the following complexity result:

Table 1. Upper bound b_c

c	10	20	40	80	160	320
b_c	66	231	861	3321	13041	51681

Theorem 1 (Optimal Online Checkpointing). *Let the number of available checkpoints equal c. Then the online checkpointing procedure given by Algorithm 1 ensures a time-minimal adjoint computation storing no more than c checkpoints at any time for any number l of time steps if l satisfies the inequality $l \leq b_c$.*

Proof: See [10]. ■

Hence, provided that the number of time steps does not exceed the upper bound b_c one can compute the adjoint of a time step sequence with an a priori unknown length using up to c checkpoints at any time with the optimal, i.e. minimal, run time. This minimal run time is given by the number of time step evaluations in addition to the evaluations of adjoint time steps. Since each adjoint time step has to be executed exactly once, only the number of time steps performed can vary for different checkpointing approaches. In [7], checkpoint strategies were studied for an a priori known number l of time steps the adjoint of which has to be calculated. It was shown that the minimal number of time step executions is given by an explicit formula in the following way: Let $t(c, l)$ denote the minimal number of time steps evaluated to compute the adjoint of l time steps storing up to c checkpoints at any time. Then $t(c, l)$ has the explicit form

$$t(c, l) = rl - \beta(c + 1, r - 1) + 1, \tag{10}$$

where r is the unique integer satisfying $\beta(c, r - 1) < l \leq \beta(c, r) \equiv \binom{c+r}{c}$. Surprisingly, the checkpoint algorithm proposed in this paper reaches this minimal number of time steps even for an unknown number l of time steps as long as l does not exceed the upper bound b_c. The constant b_c grows quadratically in the number of checkpoints as illustrated by Table 1. Therefore, already a moderate number of checkpoints ensures an optimal run time for a reasonable number of time steps to be reversed. For example, usually no more than 200 checkpoints are required for the problems considered in this paper.

3.2 Online Checkpointing on Parallel Computers

The optimal online checkpointing of Algorithm 1 has been implemented as an extension of the optimal offline checkpointing software revolve [7]. It is planed for a future version of revolve to incorporate the heuristics of arevolve in the case of online checkpointing and $l > b_c$. Here, one would perform the optimal online checkpointing as long as $l \leq b_c$. If l exceeds b_c the heuristics of arevolve will be applied to avoid a break down of the overall adjoint computation.

Applying the checkpointing routine revolve on a parallel computer, one faces two very different situations: The first possibility is that all checkpoints can be kept in main memory. Then the access time to all checkpoints is negligible as assumed in the theoretical analysis contained in [7]. However, the maximal number of checkpoints may be considerably limited due to this approach. Taking advantage of new features of parallel IO filesystems such as Lustre allows to extend the number of checkpoints by storing checkpoints also on disc. Then the access cost of the checkpoints is no longer negligible for all checkpoints because of the parallel I/O. Hence, one has to take the memory access costs into account resulting in a so-called multi-stage checkpointing. For this purpose, we present the following result:

Theorem 2 (Number of Checkpoint Writes). *Let $l > c+2$ be the number of time steps the adjoint of which is computed using c checkpoints and the online checkpointing Algorithm 1. If w_i denotes the number of times data is written onto the checkpoint i during the first integration to state x_l, then one has for*

$$l = \sum_{i=1}^{j}(c+2-i) + q \in \left\{ \sum_{i=1}^{j}(c+2-i), \ldots, \sum_{i=1}^{j+1}(c+2-i) - 1 \right\}$$

$$\begin{array}{lll}
\textit{if } \ q \in \{0,1\}: & w_0 = 1, & w_i = j \quad 0 < i \le c-j, \quad w_i = c-i+1 \quad c-j < i < c \\
\textit{if } \ q = 2: & w_0 = 1, & w_i = j \quad 0 < i < c-j, \quad w_i = c-i+1 \quad c-j \le i < c \\
\textit{if } \ q > 2: & w_0 = 1, & w_i = j+1 \quad 0 < i < \min\{q-1, c-j\}, \\
& & w_i = j \quad q-2 < i < c-j, \\
& & w_i = c-i+1 \quad c-j \le i < c.
\end{array}$$

Proof: For $l \le c+2$, the checkpointing schedule is trivial. Therefore, we do not consider this case here. For $c+2 < l \le b_c$, one can divide the range $\{0, \ldots, b_c\}$ into the c ranges

$$\begin{aligned}
R_j &\equiv \{l_j, \ldots, u_j\} \equiv \left\{ \sum_{i=1}^{j}(c+2-i), \ldots, \sum_{i=1}^{j+1}(c+2-i) - 1 \right\} \quad 0 \le j < c-1 \\
R_{c-1} &\equiv \{l_{c-1}, \ldots, u_{c-1}\} \equiv \{b_c - 3, \ldots, b_c\}.
\end{aligned}$$

This separation is based on the definition of b_c. Applying Algorithm 1, checkpoint 0 stores the initial state x_0 and is not overwritten afterwards. Furthermore, the states $1, \ldots, c-1$ are stored in the checkpoints $1, \ldots, c-1$ since $l > c+2$. Then, for each range R_j with $j > 0$ and $\tilde{l} < l$ for all $\tilde{l} \in R_j$, the checkpoint $c-j$ stores the state l_j and is not overwritten afterwards. Furthermore, the states $l_j+1, \ldots, l_j+c-j-1 = l_j+1, \ldots, u_j-1$ are stored in the checkpoints $1, \ldots, c-j-1$. For j_u with $l \in R_{j_u}, l_{j_u} \le l \le u_{j_u}$, one has that $l = l_{j_u} + q$ with $q \le c-j+1$. Then, the checkpoint $c-j_u$ stores the state l_{j_u} if $q > 1$. If $q > 2$, additionally the states $l_{j_u}+1, \ldots, l_{j_u}+q-2$ are stored in the checkpoints $1, \ldots, q-2$. Summarizing these observations proves the assertion. ■

The checkpoint write counts proved in the last theorem form a first step to allow larger checkpoint numbers based for example on parallel IO filesystems such as Lustre. The remaining part is an analysis of the checkpoint write and read counts for the reversal process initiated by revolve. This topic is currently investigated to allow an overall minimization of the access time to the checkpoints. From the results obtained so far in this direction, a suitable strategy seems to be that one assigns the more expensive checkpoints, i.e., the checkpoints distributed on the file system to the checkpoints with higher numbers and to assign the less expensive checkpoints, i.e., the checkpoints in main memory to the checkpoints with lower numbers.

4 Numerical Experiments

The HiFlow package is a multipurpose parallel finite-element package with a strong emphasis in computational fluid dynamic, reactive flows and related subjects. It is developed in C++, and its design takes great advantage of the object-oriented concepts and of the generic programming capabilities offered by this language. The overall design of this project is highly modular and allows an interplay of its different submodules. The computations presented in this paper rely especially on two submodules: *HiFlowOpti* and *HiFlowNavierStokes*. The *HiFlowOpti* submodule contains generic solvers for optimal control and parameter identification as well as experimental design. This module has been extended by means of the checkpointing strategy described in the previous section. The *HiFlowNavierStokes* module contains the solvers related to the resolution of the instationary Navier-Stokes equations. In both modules all methods are available for both sequential and parallel platforms. The numerical experiments presented in this paper have been performed on the high performance computer HP XC 6000 at the Computing Center of the University Karlsruhe. This parallel computer is based Itanium2 processors with a frequency of 1.5 GHz. On each node 8 GB RAM are available.

4.1 Three Dimensional Benchmark Channel Flow

In order to validate the proposed checkpointing strategy we consider the three dimensional benchmark configuration proposed by Schäfer et al. [16]. The proposed setup consists of a flow channel around a cylinder with squared crossed section. The height and width of the channel are $H = 0.41m$, and the diameter of the cylinder is $D = 0.1m$. The goal of this benchmark is to compute accurately the drag and lift forces acting on the cylinder, where the cost functional is the averaged value of the drag over the interval $I = [50, 100]$.

We stress that our aim in this section is to illustrate the capabilities of the proposed checkpointing strategy. The exact analysis of the impact of such a technique in relation with adjoint-based a posteriori error estimation is beyond the scope of this paper and is described in more detail in [10].

Table 2. Results for the proposed checkpointing scheme for various computational setups with l time steps. The amount of available memory capacity is equal for all configurations and results in c checkpoints.

	# Unknowns (space)	l	c	N_{new}	N_{old}
global refinement	1.2 10^6	912	600	310	1222
global refinement	1.0 10^7	702	36	664	1366
local refinement	8.2 10^5	854	715	137	991

In Table 2, results of the proposed checkpointing scheme are presented. For a fixed amount of available memory, we consider three different levels of refinement in space. For these three configurations the discretization in space is so fine that the full storage of the forward solution in main memory would be impossible even on the considered parallel platform. For this application, the number of available checkpoints in main memory is a priori fixed due to the enormous amount of memory needed for each checkpoint but the number of time steps is a priori unknown. Therefore, one alternative checkpointing strategy would be to first perform a pure function evaluation without adjoint computations to determine the number of time steps to perform and then to apply revolve for the distribution of the checkpoints. The required number of additional time steps needed by this alternative is given by N_{old} in the last column of Table 2. Using the new optimal online checkpointing proposed in this paper, the number of additional forward steps can be reduced significantly, as shown by the column N_{new} in Table 2. As can be seen, the equation $N_{old} = N_{new} + l$ holds since the new checkpointing approach does not require an extra integration to determine the value of l. For the most memory consuming case of the 3D-channel with global refinement leading to 10^7 unknowns we impose the number of checkpoints to be equal to $c = 36$. The performances which are measured in Table 2 with respect to the number of extra forward steps clearly show the high efficiency of the proposed scheme.

5 Conclusion

We present a provable optimal, i.e., time-minimal, online checkpointing procedure. In the present paper, we focus on the practical aspects, that is the specific application of revolve and its coupling with the parallel finite-element package HiFlow for solving optimal control problems and goal-oriented error estimation on parallel machines. The main advantage of the presented online checkpointing is that it guarantees a time-minimal run time for an a priori unknown number of time steps as long as this number does not exceed a given upper bound. Due to the semi-implicit time stepping applied, this upper bound is only a very weak restriction. Additionally, we proved an explicit formula for the number of times data is written onto the checkpoints during the generation of the checkpoint distribution. This forms the first step to allow an improved checkpointing strategy if parallel IO filesystems such as Lustre are used.

References

1. S.C. Brenner and R.L. Scott. *The mathematical theory of finite element methods.* Springer, Berlin-Heidelberg-New-York, 1994.
2. F. Brezzi and R. Falk. Stability of higher-order Hood-Taylor methods. *SIAM J. Numer. Anal.*, 28(3):581–590, 1991.
3. K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, 4:105–158, 1995.
4. K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problems, I: A linear model problem. *SIAM J. Numer. Anal.*, 28:43–77, 1991.
5. K. Eriksson and C. Johnson. Adaptive finite element methods for parabolic problems, II, IV, V. *SIAM J. Numer. Anal.*, 32:706–740, 32:1729–1763, 1995.
6. M.B. Giles. On adjoint equations for error analysis and optimal grid adaptation. In D.A. Caughey and M.M. Hafez, editors, *In Frontiers of Computational Fluid Dynamics 1998*, pages 155–170. World Scientific, 1998.
7. A. Griewank and A. Walther. Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Software*, 26:19–45, 2000.
8. M.D. Gunzburger. *Perspectives in flow control and optimization.* Advances in Design and Control 5. Philadelphia, SIAM., 2003.
9. V. Heuveline. On higher-order mixed FEM for low Mach number flows: Application to a natural convection benchmark problem. *Int. J. Num. Meth. Fluids*, 41(12):1339–1356, 2003.
10. V. Heuveline and A. Walther. Towards the economical computation of adjoints in PDEs using optimal online checkpointing. In preparation, 2006.
11. M. Hinze and J. Sternberg. A-revolve: An adaptive memory- and run-time-reduced procedure for calculating adjoints; with an application to the instationary Navier-Stokes system. *Opti. Meth. Softw.*, 20:645–663, 2005.
12. P. Hood and C. Taylor. A numerical solution of the Navier-Stokes equations using the finite element techniques. *Comp. and Fluids*, 1:73–100, 1973.
13. L. Machiels, A.T. Patera, and J. Peraire. Output bound approximation for partial differential equations; application to the incompressible Navier-Stokes equations. In S. Biringen, editor, *Industrial and Environmental Applications of Direct and Large Eddy Numerical Simulation.* Springer, 1998.
14. J.T. Oden and S. Prudhomme. On goal-oriented error estimation for elliptic problems: Application to the control of pointwise errors. *Comput. Methods Appl. Mech. Eng.*, 176:313–331, 1999.
15. Y. Saad. *Iterative methods for sparse linear systems.* Computer Science/Numerical Methods. PWS Publishing Company, 1996.
16. M. Schäfer and S. Turek. Benchmark computations of laminar flow around cylinder. *Notes on numerical fluid mechanics*, 52:856–869, 1996.
17. R. Serban and A.C. Hindmarsh. CVODES: An ODE solver with sensitivity analysis capabilities. UCRL-JP-20039, LLNL, 2003.
18. A. Walther and A. Griewank. Advantages of binomial checkpointing for memory-reduced adjoint calculations. In M. Feistauer et al., editor, *Numerical mathematics and advanced applications*, pages 834–843. Springer, 2004.
19. P. Wesseling. *An introduction to multigrid methods.* Wiley, Chichester, 1992.