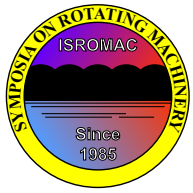
Coarse checkpointing for URANS- or DES-based unsteady adjoint shape optimisation

Jan C. Hückelheim, School for Engineering and Materials Science, Queen Mary University of London, United Kingdom

Jens-Dominik Müller, School for Engineering and Materials Science, Queen Mary University of London, United Kingdom

Shenren Xu, School of Engineering, University of Liverpool, United Kingdom



Long Abstract

Introduction

The adjoint method is an established way to compute gradients in applications as diverse as industrial fluid flow, structural mechanics, weather forecasting or finance. Gradients are a key ingredient in optimisation, uncertainty quantification or flow control, which are used in academia and industry.

The appeal of the adjoint method is that its computational expense is independent of the number of design parameters, and instead grows with the number of cost function values that define the quality of any given design. The number of design parameters, e.g. CAD parameters, is typically in the thousands or millions. In contrast, the quality of a design can often be expressed as a scalar, e.g. fuel consumption, making the adjoint method affordable in situations where competing methods are not[1].

Nevertheless, the adjoint method faces challenges, particularly in the presence of unsteady flow, one of them being the computational cost and memory requirements that arise from the need to store the full flow trajectory for the adjoint computation. The proposed solutions such as checkpointing[2] or checkpoint compression[3] trade some memory requirements for even higher computational cost.

Checkpointing with time gaps was presented to overcome some of the memory consumption issues and trade some of the memory requirements for solution accuracy[4]. This method was demonstrated to work well for highly unsteady URANS flow, where reductions of the memory footprint by more than 90% were achieved with only a slight reduction in the accuracy of adjoint sensitivities.

In this work, we propose to reduce memory requirements by storing selected temporal checkpoints, *gappy checkpointing*. The states between temporal checkpoints are then reconstructed in space and time from the reduced checkpointed data. Similarly, coarsening the computational grid of these checkpoints, *coarse checkpointing*, as typically arising in geometric multigrid methods is used to compress the spatial variation in the flowfield. The restriction and prolongation operators built into the multigrid CFD solver are then used to reconstruct an approximation of the flow field during the adjoint computation.

The method will be applied to flow through a U-Bend cooling channel simulated with DES or URANS. The effect of the coarsened checkpoints on the accuracy of adjoint sensitivities of pressure drop to surface deformation parameters will be assessed.

1. Unsteady adjoint

We use a CFD solver with DES turbulence model that is based on an unsteady viscous flow solver for unstructured grids, BDF2 dual time stepping and an implicit solver to converge the inner iterations which was presented in [5]. The adjoint solver is generated using the automatic differentiation tool Tapenade [6] with some hand-coded optimisations for improved speed [7]. To reduce the computation time, the solver uses a store-all approach and avoids recomputations of flow results.

The viscous unsteady flow equations, including the DES turbulence model, can be discretised and

evolved in time using a BDF2 formulation as

$$\left[\frac{\partial \hat{R}(U_{t-2}, U_{t-1}, U_k)}{\partial U_k}\right] \delta U_k = -\hat{R}(U_{t-2}, U_{t-1}, U_k)$$

Similarly, the adjoint system can be evolved as

$$-\frac{v^{t-2} - 4 \cdot v^{t-1} + 3 \cdot v^t}{2\Delta t} + R_v(v^t) := \hat{R}_v\left(U_{t-2}, U_{t-1}, U_t\right)$$

using the same BDF2 and multigrid method as the primal equation and an adjoint kernel generated by source-transformation AD. The solution of the adjoint equation requires the history of the flow solution U_t at each time step for the calculation of R_v and the preconditioning matrix P^T. This flow field can be stored during the flow solution and loaded during the adjoint solution, recomputed by running the flow solver again (e.g. following the REVOLVE algorithm), reconstructed using temporal interpolation[4], or reconstructed from a coarse grid solution following our new approach.

2. Coarse checkpointing

During the primal flow computation, every time that the inner BDF2 iteration is fully converged, the so-obtained flow field is restricted to a coarse grid and then stored in memory.

For each BDF2 outer iteration of the adjoint computation, the correct coarse time step will be retrieved from memory and extrapolated to the finest mesh resolution using the multigrid prolongation operator. This flow field will be used as the basis for the construction of all adjoint operators during the BDF2 inner loop that then follows.

3. Test case

The test case is a duct with rectangular profile that features a 180° turn in the middle, which is often referred to as a *U-bend* due to its shape. Ducts of this kind can be found in the internal cooling of turbine blades and have been studied extensively [8]. We will compute flow results at a Reynolds number of 40,000 to show primal flow and adjoint sensitivity results using a Spalart-Allmaras turbulence model or a DES model. The cost function for the adjoint computation will be pressure loss.

4. Results

We will present sensitivities of pressure drop with respect to shape variables. The effect of time-coarsening and space-coarsening of the primal checkpoints on the accuracy of these sensitivities will be investigated and compared to the gain in reduced check-point size.

References

[1] M. Giles, N. Pierce, M. Giles, and N. Pierce. *Adjoint equations in CFD - Duality, boundary conditions and solution behaviour*. American Institute of Aeronautics and Astronautics, 2015/05/03 1997.

[2] Andreas Griewank and Andrea Walther. Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Softw.*, 26(1):19–45, March 2000.

[3] Eric C. Cyr Tim Wildey and John Shadid. Adjoint based a posteriori error estimates using data compression. In C. Tiago J. P. Moitinho de Almeida, P. D ıez and N. Parés, editors, *VI International Conference on Adaptive Modeling and Simulation*, 2013.

[4] Jan Hueckelheim and Jens-Dominik Müller. Checkpointing with time gaps for unsteady adjoint cfd. In *EUROGEN-2015*, volume 11, September 2015.

[5] Shenren Xu, David Radford, Marcus Meyer, and Jens-Dominik Müller. Stabilisation of discrete steady adjoint solvers. *Journal of Computational Physics*, (accepted for publication):–, 2015.

[6] Laurent Hascoët and Valérie Pascual. The Tapenade Automatic Differentiation tool: principles, model, and specification. Research Report RR-7957, May 2012.

[7] Faidon Christakopoulos, Dominic Jones, and Jens-Dominik Müller. Pseudo-timestepping and verification for automatic differentiation derived {CFD} codes. *Computers Fluids*, 46(1):174 – 179, 2011. 10th {ICFD} Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[8] Sebastian Willeke and Tom Verstraete. Adjoint optimization of an internal cooling channel u-bend. In *ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*, pages V05AT11A029–V05AT11A029. American Society of Mechanical Engineers, 2015.