Checkpointing with time gaps for unsteady adjoint CFD

Jan Christian Hueckelheim and Jens-Dominik Mueller

Abstract Gradient-based optimisation using adjoints is an increasingly common approach for industrial flow applications. For cases where the flow is largely unsteady however, the adjoint method is still not widely used, in particular because of its prohibitive computational cost and memory footprint. Several methods have been proposed to reduce the peak memory usage, such as checkpointing schemes or checkpoint compression, at the price of increasing the computational cost even further. We investigate incomplete checkpointing as an alternative, which reduces memory usage at almost no extra computational cost, but instead offers a trade-off between memory footprint and the fidelity of the model. The method works by storing only selected physical time steps and using interpolation to reconstruct time steps that have not been stored. We show that this is enough to compute sufficiently accurate adjoint sensitivities for many relevant cases, and does not add significantly to the computational cost. The method works for general cases and does not require to identify periodic cycles in the flow.

1 Introduction

The adjoint method is commonly used in academia and industry to compute the derivative of a cost function with respect to its design variables. Its greatest appeal lies in the fact that the computational cost is constant in the number of design variables, in contrast to simpler approaches such as finite differences or tangent-linear derivatives. This makes the method feasible for industrial applications with rich design space [4].

Jan Christian Hueckelheim
Queen Mary University of London, London, UK e-mail: j.c.hueckelheim@qmul.ac.uk

Jens-Dominik Mueller
Queen Mary University of London, London, UK e-mail: j.mueller@qmul.ac.uk

Many real-world problems however still present a challenge for the adjoint method. A particular problem is severe unsteadiness, as can be found in turbines, including wind turbines, aircraft wings in high-lift configuration, car engines and many more [12]. The adjoint method has been formulated for this kind of problem in frequency [13] and temporal space [15], but requires the storage of the full flow history, resulting in prohibitive memory requirements in most cases.

A well-known way to mitigate this problem is the REVOLVE checkpointing algorithm [5]. It stores checkpoints only at carefully chosen time steps, and recomputes each time step when it is needed, starting from the time steps that have been stored. Another approach that has recently been proposed [16] is the compression of checkpoints. Both ideas have one thing in common: the memory requirements are relaxed at the cost of increased computational expense. Furthermore, lossless data compression does not offer large savings in storage space[14], and so the method becomes more useful if lossy compression is used, resulting in errors in the reconstruction of the primal flow field.

We investigate incomplete checkpointing as an alternative, which reduces memory usage at no extra computational cost, but instead offers a direct trade-off between memory footprint and the fidelity of the model. We use a dual timestepping scheme in which the inner iterations are fully converged, so that only physical time steps need to be stored and the adjoint field can be reconstructed based on the fully converged checkpoint, which preserves the accuracy of the result if the innter iteration was fully converged [3]. In addition, we store only selected physical time steps and use interpolation to reconstruct time steps that have not been stored.

The scheme comes at negligible cost for linear or other low order interpolation methods. In particular, the reconstruction from data available in memory is significantly faster than reading the checkpoint from disk, which would be another possible (but slow) way of addressing the memory limitations. Finally, our method does not require any assumptions about the flow such as periodicity.

Since the computational cost of interpolation is negligible in most cases, this work focuses on assessing the accuracy of the adjoint results obtained with this approach.

2 Background

We use 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 [19]. The adjoint solver is generated using the automatic differentiation tool Tapenade [7] with some hand-coded optimisations for improved speed [2].

2.1 Solving the flow and adjoint equations

The viscous unsteady flow equations can be written as

$$\frac{\partial U}{\partial t} + R(U) = 0$$

and can be discretised using a third-order accurate BDF2 time marching scheme as

$$\begin{aligned}\frac{\partial U}{\partial t} + R(U_t) &= \frac{U_{t-2} - 4 \cdot U_{t-1} + 3 \cdot U_t}{2\Delta t} + R(U_t) \\ &:= \hat{R}(U_{t-2}, U_{t-1}, U_t)\end{aligned}$$

The above system can be evolved in time by solving the linearised system for U_k and successively updating the converged flow solution U_t at time t

$$\begin{aligned}\left[\frac{\partial \hat{R}(U_{t-2}, U_{t-1}, U_k)}{\partial U_k}\right] \delta U_k &= -\hat{R}(U_{t-2}, U_{t-1}, U_k) \\ U_t &= U_{t-1} + \delta U_k\end{aligned}$$

The unsteady adjoint system can be written as

$$-\frac{\partial v}{\partial t} + \underbrace{\left(\frac{\partial R}{\partial U}\right)^T v - \left(\frac{\partial J}{\partial U}\right)^T}_{:=R_v} = 0$$

and can, like the primal equation, be discretised using BDF2 as

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

and solved using the same method as the primal equation.

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), restored approximately from a compressed data, or reconstructed using interpolation, following our new approach.

2.2 Physical checkpointing

We use an approach in which only the physical time steps are stored during the primal computation and restored during the adjoint computation, as presented

in [9] and Algorithm 1. The memory requirements are orders of magnitude smaller compared to the brute-force method of storing every iteration.

Algorithm 1: Dual timestep with physical checkpointing

```
n ← 0;
U_0, V_0 ← initial guess;
while t < t_final do // primal loop
    n ← 0;
    while R(U_{t,n}) ¿ cutoff do // primal loop
        U_{t,n+1} ← flow_pseudostep(U_{t,n});
        n ← n + 1
    end
    t ← t + Δt;
    U_{t+1,0} ← U_{t,n}; // init for next step
    store(U_t);
end
while t > t_init do // adjoint loop
    load(U_t);
    V_{t,0} ← V_{t+1,n}; // init for next step
    t ← t − Δt;
    while R(V_t) ¿ cutoff do // primal loop
        V_{t,n+1} ← adjoint_pseudostep(V_{t,n}, U_t);
        n ← n + 1
    end
end
```

Aside from the memory requirements that arise from storing the flow trajectory, one major challenge in this approach lies in the need to fully converge the inner loop so that an efficient adjoint method for fixed-point loops can be used[3]. To address this, we use an implicit solver with geometric multigrid and ILU preconditioning to converge in an acceptable time[19].

For our proposed method, the calls to `store()` and `load()` in this algorithm are replaced by calls to augmented routines `gappyStore()` and `gappyLoad()` as described below.

3 Checkpointing with gaps

The routine `gappyStore()` contains a logic that selects certain snapshots worth storing, which are denoted by the set of stored time steps T_s which are a subset of all time steps T.

In the simplest incarnation of this method, T_s would contain only every n-th time step, for some fixed n. In some special cases it might be beneficial to vary the checkpoint density over time, e.g. to capture a particular phenomenon with a higher accuracy. This was not investigated in this work.

If our method is regarded as a very simple form of data compression, then the data compression ratio is $\|T\|/\|T_s\|$. Obviously we get better compression ratios if we store fewer time steps. For evenly spaced snapshots as suggested above, we obtain a compression ratio $\|T\|/\|T_s\| = n$.

Checkpoints that have not been stored need to be reconstructed. If linear interpolation is used, we can formalise this method as follows. Let t denote the time for which a checkpoint needs to be reconstructed. Also, let t^+ and t^- denote the unique time steps for which all of the following conditions hold:

$$\begin{aligned} t^+, t^- &\in T_s \\ \nexists t^* \in T_s &: t < t^* < t^+ \\ \nexists t^* \in T_s &: t^- < t^* < t \end{aligned}$$

In other words, t^+ and t^- are the closest stored time steps just after and before t, respectively.

We can then implement the `gappyLoad()` routine that can perform linear interpolation or constant interpolation as follows:

Function gappyLoad(t)
if $t \in T_s$ **then** `// t was stored`
  return *load*(U_t);
else
  $U^- \leftarrow$ *load*(U_{t^-});
  $U^+ \leftarrow$ *load*(U_{t^+});
  return $U^- + \frac{t-t^-}{t^+-t^-} \cdot (U^+ - U^-)$;
end

If the gap $t^+ - t^-$ is larger than two, this method can be implemented much more efficiently by storing most of the intermediate results. The routine could be implemented for higher orders of interpolation, taking into account more of the surrounding stored time steps.

4 Test case

4.1 Primal solver setup

To test our method we use a RAE2822 aerofoil with a trailing edge that ia truncated at 10% cord length and a 30° angle of attack to provoke a high amount of shedding. The freestream velocity is 0.2 Ma, we use viscous flow. The mesh has around 25000 cells, the solver is node-centred and uses 4 levels of geometric multigrid for faster convergence. The setup and the primal and adjoint flow field are shown in fig. 1.

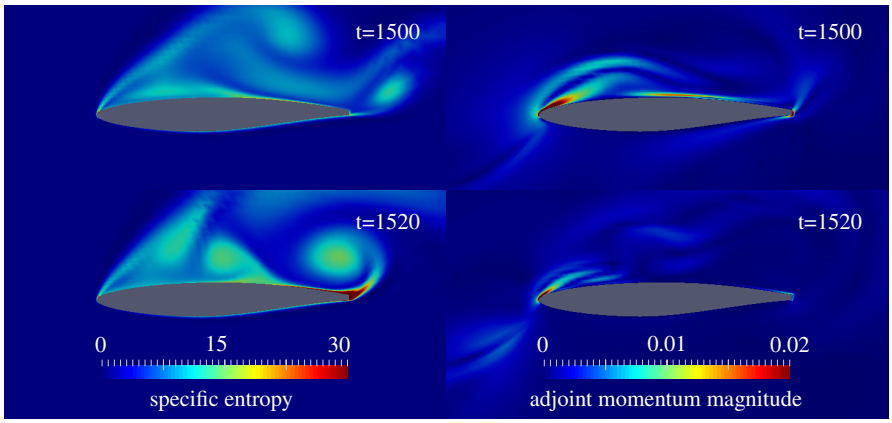


Fig. 1: Snapshots after 1.5 s (1500 time steps) and 1.52 s (1520 time steps). **Left:** Specific entropy, showing strong vortex shedding above the airfoil with a frequency of roughly 70 time steps per period (0.07s). **Right**: Adjoint momentum at the same time steps. A high sensitivity close to the top surface in step 1500 appears where a vortex is about to form, which can be seen in the primal flow at step 1520. This adjoint peak stems from the small adjoint momentum peaks that are above the airfoil in step 1520 and are propagated back to the airfoil surface during the reverse sweep.

For the reference setup, we use a time step size of $t_{ref} = 1$ ms which corresponds to ca. 70 checkpoints for each flow period. While this flow does exhibit periodic cycles, our solver does not exploit this periodicity. Our findings should therefore apply to non-periodic flow as well.

To validate that the chosen time step is fine enough to resolve the primal flow sufficiently we perform another simulation with a time step size of $0.5 \cdot t_{ref}$. Furthermore, to obtain benchmark results for the incomplete checkpointing method we run a series of 6 additional simulations with time step sizes 2, 4, 8, 16, 64 and 256 times t_{ref}. We will refer to these setups as $t_{0.5}, t_2 \ldots t_{256}$.

4.2 Incomplete checkpointing setup

We use the primal results generated with t_{ref} to initialise the adjoint solver. To investigate if this temporal resolution is also sufficient to get accurate adjoint results, we perform another adjoint simulation with time step size $0.5 \cdot t_{ref}$ for comparison.

Finally, we create incomplete checkpoint trajectories from the reference primal result t_{ref} as follows: For the setup that we will refer to as a_2, we discard all primal states at even time step numbers, and reconstruct them using linear interpolation from the nearest odd time steps. We replace all but every 4^{th} time step by linear interpolation for the a_4 setup and proceed likewise to obtain a_8, a_{16}, a_{64} and a_{256}.

4.3 Adjoint solver setup

The adjoint solver is set to the same time step size as the primal sovler for the $t_{0.5}, t_2 \dots t_{256}$ setups. For the $a_2 \dots a_{256}$ setups the numerical time step size is t_{ref}.

The cost function is total drag of the airfoil averaged over a time window with a weight function ω to progressively switch the averaging process on and off. Given the total drag at each time step $J(t)$, the average drag J_{avg} can be formulated as

$$J_{avg} = \frac{\sum_{t=1}^{t_\infty} J(t) \cdot \omega(t)}{\sum_{t=1}^{t_\infty} \omega(t)} \tag{1}$$

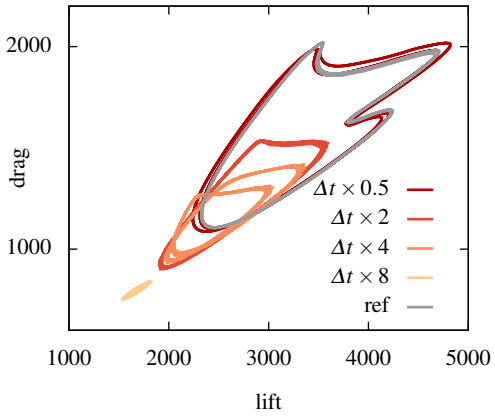
We use a weight function that ramps up linearly for 0.05 s until it reaches its peak at which it remains for 0.175 s, then ramps down to deactivate the averaging after another 0.05 s. We study three design parameters that are shown in previous works:

1. **Flow control:** (e.g. [6, 8, 1]) We consider a valve that can inject or remove tangential momentum on the airfoil top just behind the leading edge. This design parameter can vary in time and thus allows us to study transient behaviour of the adjoint field. We compute this sensor for each time step by integrating the adjoint momentum field over a small circular area around the valve location.
2. **Surface node displacement:** (e.g. [18, 10]) We consider the surface nodes' displacement in normal direction as the design vector, which is common for shape optimisation applications. The design parameter does not allow variations in time and is thus based on the time-averaged adjoint field. We use this to study the spatial behaviour of the adjoint field. The spring analogy model is used to project volume sensitivities onto the surface.
3. **Angle of attack:** (e.g. [11]) We consider the shape fixed and only allow an adjustment of the angle of attack. Since this will hide oscillatory errors in space and time, it can be used to study the overall trend of the adjoint field. This sensor is computed based on the cross product of surface sensitivity vectors and point vectors of surface nodes, integrated over the entire airfoil surface.

For the surface sensitivity and angle of attack sensors we require a time-averaged adjoint field. The average is taken over a time window given by the window function ω_a. The adjoint averaging window is twice as long as the cost averaging window.

5 Results

The reference and $t_{0.5}$ solution show very similar primal flow features and lift/drag values match well, suggesting that the reference step size is small enough to resolve the primal flow. Using coarser time steps $t_2 \dots t_8$, the primal flow is no longer correctly resolved, see fig. 2.



$t_{ref} \times$	lift	drag
0.5	3530 ± 1291	1552 ± 465
1	3517 ± 1192	1561 ± 458
2	2747 ± 846	1219 ± 318
4	2659 ± 700	1175 ± 242
8	1680 ± 134	797 ± 42
16	1446 ± 10^{-12}	702 ± 10^{-12}
64	1446 ± 10^{-2}	702 ± 10^{-2}
256	1445 ± 0.3	701 ± 0.15

Fig. 2: Lift/drag history of fully developed flow for various time step sizes. The reference solution and the twice refined solution agree qualitatively. With coarser time steps, we are unable to resolve the transient behaviour correctly. The table shows that results for t_{ref} and $t_{0.5}$ differ by 0.4%, which is acceptable for many applications. With larger time steps, the unsteadiness vanishes and mean values for lift and drag differ from the reference by more than 60%.

We observe strong unsteadiness for the adjoint field, see fig. 1, with peaks of sensitivity close to the leading and trailing edges, and a reverse wake propagated from the leading edge towards the incoming flow. Just like in the primal solution, the unsteadiness is strongest downstream of the airfoil.

5.1 Overall accuracy: angle of attack

We first consider $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$, which is the sensitivity of drag J with respect to changes in angle of attack α. This is the crudest sensor that we have, in the sense that it regards a time- and space-averaged result which allows temporal and spatial error modes to cancel out to some extent, see Table 1 for results.

Coarsening the primal temporal solution has a strong effect on the sensitivity results: relative errors rise above 60% for setup $t_8 \ldots t_{256}$. In contrast to this, we observe that the sensitivity results produced with incomplete checkpointing are acceptable. To give an example from the table: the a_{64} setup only keeps every 64^{th} time step and results in a sensitivity that differs from the reference by less than 0.2%. This is achieved with the same memory requirements as for the t_{64} setup, which has an error of over 60%. The a_4 result agrees with the reference result within 0.003%, but requires 75% less memory than the reference computation.

step/gap	$\frac{dJ}{d\alpha}$ for step	$\frac{dJ}{d\alpha}$ for gap
0.5	**3**844.451056	*N/A*
1.0	**3612.203084**	**3612.203084**
2.0	2541.615522	**3612.2**35254
4.0	2227.642980	**3612**.317687
8.0	1504.326992	**361**5.459025
16.0	1288.293729	**3612**.068629
64.0	1262.006153	**36**06.972954
256.0	1228.272627	**3**364.129272

Table 1: **Centre column:** $\frac{dJ}{d\alpha}$ for different time step sizes. **Right column:** $\frac{dJ}{d\alpha}$ for different gap sizes using incomplete checkpointing. The sensitivity is more dependent on the time step size than the primal result: The reference and $t_{0.5}$ sensitivities differ more than 6%, an order of magnitude more than the primal drag.

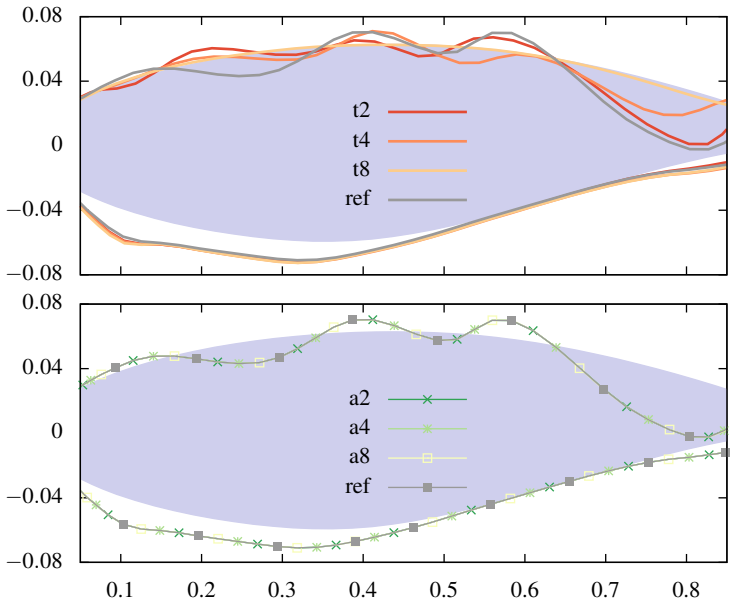


Fig. 3: Surface sensitivity scaled by a factor of $2e-4$ for plotting, superimposed on the airfoil surface. **Top:** Results for different primal time step sizes. The results agree qualitatively, although the sensitivity modes are shifted along the top surface. Reducing the temporal accuracy leads to a decrease in sensitivity. For t_8 and above, the surface sensitivity on the airfoil top is almost zero (i.e. is aligned with the current shape). The airfoil bottom shows a strong sensitivity which is resolved correctly regardless of time step size. **Bottom:** Different gap sizes for incomplete checkpointing. The sensitivities match that of the reference solution to plotting accuracy.

5.2 Spatial accuracy: Surface sensitivity

We investigate the spatial accuracy of the adjoint field by studying the sensitivity $\frac{\mathrm{d}J}{\mathrm{d}(\mathbf{x}\cdot\mathbf{n})}$ of the cost function with respect to normal displacements of airfoil surface nodes. This is based on the time-averaged adjoint field.

This sensor also shows that the temporal primal accuracy is crucial. There is a significant difference in surface sensitivities between the t_{ref} and $t_{0.5}$ setup, and an even larger one for coarser time steps. For setup t_8 and beyond, the sensitivity on the airfoil top vanishes completely, which is an indicator that the unsteadiness is no longer resolved. The surface sensitivity on the airfoil bottom is resolved correctly for large time step sizes and is the only contributor to the angle of attack sensitivity above $8 \cdot t_{ref}$.

Incomplete checkpointing results in relatively small errors for this case, see fig. 3. The error grows somewhat with the gap size. Surprisingly, the a_{16} setup is more accurate than the a_8 setup in this case, see fig. 4. Looking at the spatial distribution of errors, we find that errors are highest on the airfoil top and several orders of magnitude smaller at the airfoil bottom, see fig. 4. This is due to the more intense unsteadiness above the airfoil.

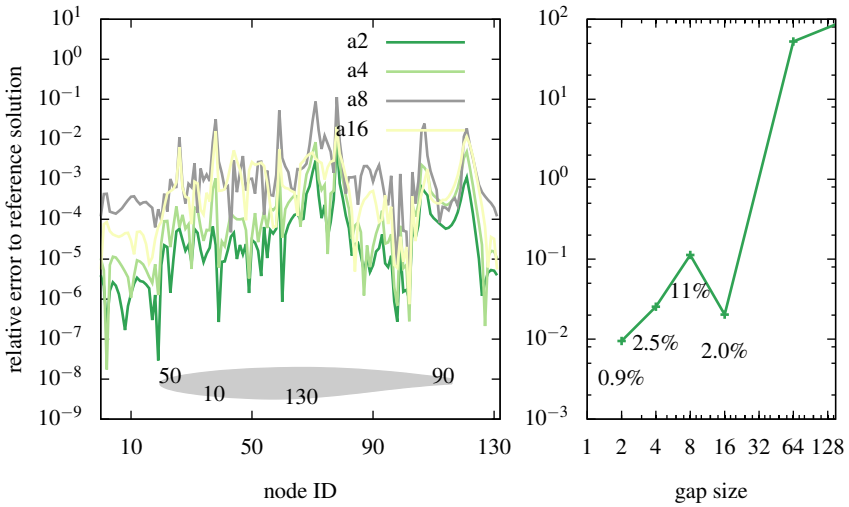


Fig. 4: **Left:** Surface sensitivity error along the airfoil surface. Node 1 is at the bottom centre, all other nodes are numbered continuously and clockwise as shown in the bottom of the left plot. Errors are larger on the airfoil top (nodes 50 to 95) than at the bottom (nodes 105 to 132 and 1 to 20). **Right:** Maximum relative error of surface sensitivity for different gap sizes. The relative error for $a_2 \ldots a_{16}$ is labeled and surprisingly smaller for a_{16} than for a_8.

5.3 Temporal accuracy: Flow control

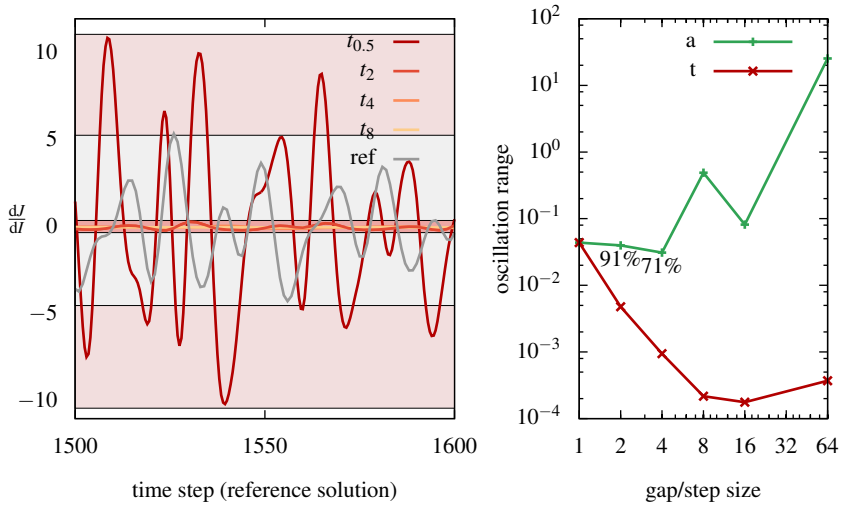


Fig. 5: **Left:** Time history of flow control sensitivities for the reference, $t_{0.5}$ and $t_2 \dots t_8$ setups. The oscillation range for the finest 3 setups are highlighted with background colour. Due to the nonlinear behaviour of the primal flow, sensitivity values at any particular time are highly dependent on the temporal resolution. **Right:** Range of oscillation (difference between minimum and maximum value over time) for a range of time step and gap sizes. Coarsening the time steps removes temporal oscillations in the adjoint field, while incomplete checkpointing with increasing gap sizes leads to stronger oscillations as the gap size becomes very large. The a_2 and a_4 setups reduce the oscillation range to 91% and 71% of the original range, respectively.

Finally, we study the sensitivity of average drag with respect to momentum injection close to the leading edge by means of a flow control valve, the location of which is shown in fig. 7. The time window in which the drag is averaged is illustrated in fig. 6. We observe that the adjoint momentum field for all time step and gap sizes is zero after the end of the cost function window, which is to be expected: An injection of momentum at any given time will not affect the drag in the past.

If the flow is chaotic [17] or almost chaotic [**?**] the sensitivities can start to diverge as the adjoint field is computed backwards in time. For some simpler test cases we can expect the adjoint field to converge to zero as we proceed backwards in time from the beginning of the cost averaging window. This behaviour can be observed for our test case and is shown in fig. 6. The sensitivity is largest for all points in

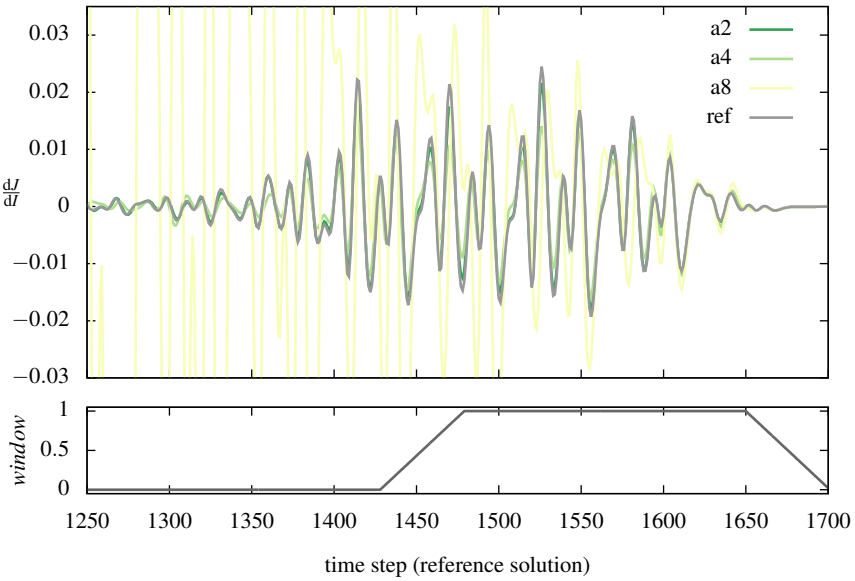


Fig. 6: **Bottom:** cost function averaging time window ω. **Top:** Sensitivity of drag with respect to momentum injection above the leading edge, plotted over time for a series of incomplete checkpointing gap sizes. We observe a zero adjoint field for time step 1700 and above, as any change happening after the averaging window can not affect the cost function. The sensitivity peaks for a time frame that is slightly longer than the cost averaging window and shifted towards lower time step numbers. The a_8 setup is oscillatory with a growing amplitude towards earlier time steps.

time from which an injection of momentum could be propagated to some point of the airfoil surface during the cost function averaging time.

For gap sizes 8 and above, we find that the sensitivity does not settle down as we proceed backwards in time, and instead starts to oscillate with an exponentially growing magnitude. Surprisingly, this growing error mode canceled out in the time-averaging process that we used for the angle of attack and surface sensitivity studies.

In contrast to this, a coarsening of primal temporal resolution leads to a decrease in transient oscillations. Due to the highly nonlinear nature of the primal flow, it is impractical to compare the results of different time step or gap sizes directly for a given point in time, since even a minor change in the frequency of oscillations introduced by a change in time step size can accumulate over time, see fig. 5. Hence, we compare the range in which the sensitivity oscillates over time.

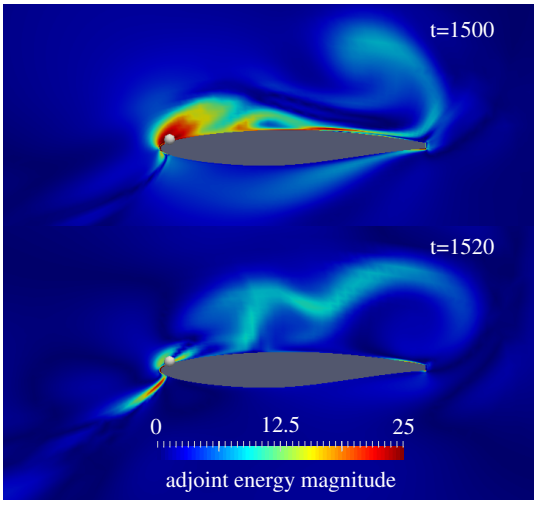


Fig. 7: Adjoint energy magnitude field after 1.5 s and 1.52 s. The flow control valve is marked with a sphere above the leading edge. The flow control sensitivity results are based on a design variable that is the momentum injection rate in airfoil surface tangential direction at this location.

6 Conclusion, possible extensions

We show that gaps in the stored time trajectory are an easy and effective way of reducing the memory footprint of unsteady adjoint calculations. The effect on the sensitivity accuracy is acceptable for many industrial cases even for relatively large gap sizes, making our approach worth considering as an alternative to lossy checkpoint compression, with a significantly smaller implementation effort and computational cost.

In particular, the error introduced by storing an incomplete trajectory is much smaller than the error introduced by under-resolving the physical time during the primal flow computation. It is therefore preferable to perform the primal simulation with a fine temporal resolution and to use incomplete checkpointing, compared to the alternative of reducing the number of primal time steps to a number that fits into memory.

An error estimation strategy for incomplete checkpointing would be useful to choose the gap size and interpolation order. This could be used to adapt the checkpoint storing interval during the primal simulation and the restoration order during the adjoint simulation dynamically, e.g. to capture some flow features with a higher accuracy.

Finally, a similar method could be implemented for spatial coarsening, using interpolation or coarse grid multigrid solutions if available.

7 Acknowledgement

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no [317006].

This research utilised Queen Mary's MidPlus computational facilities, supported by QMUL Research-IT and funded by EPSRC grant EP/K000128/1.

References

1. Berggren, M.: Numerical solution of a flow-control problem: Vorticity reduction by dynamic boundary action. SIAM Journal on Scientific Computing **19**(3), 829–860 (1998)
2. Christakopoulos, F., Jones, D., Müller, J.D.: Pseudo-timestepping and verification for automatic differentiation derived {CFD} codes. Computers Fluids **46**(1), 174 – 179 (2011). DOI http://dx.doi.org/10.1016/j.compfluid.2011.01.039. URL http://www.sciencedirect.com/science/article/pii/S0045793011000466. 10th {ICFD} Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010)
3. Christianson, B.: Reverse accumulation and implicit functions. Optimization Methods and Software **9**(4), 307–322 (1998). DOI 10.1080/10556789808805697. URL http://dx.doi.org/10.1080/10556789808805697
4. Giles, M., Pierce, N., Giles, M., Pierce, N.: Adjoint equations in CFD - Duality, boundary conditions and solution behaviour. American Institute of Aeronautics and Astronautics (1997). DOI doi:10.2514/6.1997-1850. URL http://dx.doi.org/10.2514/6.1997-1850
5. Griewank, A., Walther, A.: Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. ACM Trans. Math. Softw. **26**(1), 19–45 (2000). DOI 10.1145/347837.347846. URL http://doi.acm.org/10.1145/347837.347846
6. Gunzburger, M.: Adjoint equation-based methods for control problems in incompressible, viscous flows **65**(3-4), 249–272 (2000). DOI 10.1023/A:1011455900396. URL http://dx.doi.org/10.1023/A%3A1011455900396
7. Hascoët, L., Pascual, V.: The Tapenade Automatic Differentiation tool: principles, model, and specification. Research Report RR-7957 (2012). URL https://hal.inria.fr/hal-00695839
8. Heuveline, V., Walther, A.: Online checkpointing for parallel adjoint computation in pdes: Application to goal-oriented adaptivity and flow control. In: Euro-Par 2006 Parallel Processing, pp. 689–699. Springer (2006)
9. Hueckelheim, J., Xu, S., Gugala, M., Müller, J.D.: Time-averaged steady vs. unsteady adjoint: a comparison for cases with mild unsteadiness. American Institute of Aeronautics and Astronautics (2015). DOI doi:10.2514/6.2015-1953. URL http://dx.doi.org/10.2514/6.2015-1953
10. Jameson, A.: Aerodynamic shape optimization using the adjoint method
11. Krakos, J.A., Darmofal, D.L.: Effect of small-scale output unsteadiness on adjoint-based sensitivity. AIAA Journal **48**(11), 2611–2623 (2010). DOI 10.2514/1.J050412. URL http://arc.aiaa.org/doi/abs/10.2514/1.J050412
12. Lee, B.J., Liou, M.S.: Unsteady adjoint approach for design optimization of flapping airfoils. AIAA Journal **50**(11), 2460–2475 (2012). DOI 10.2514/1.J051663. URL http://dx.doi.org/10.2514/1.J051663
13. Nadarajah, S., Jameson, A.: Optimum shape design for unsteady three-dimensional viscous flows using a nonlinear frequency-domain method. Journal of Aircraft **44**(5), 1513–1527

(2007). DOI 10.2514/1.27601. URL http://arc.aiaa.org/doi/abs/10.2514/1.27601

14. Ratanaworabhan, P., Ke, J., Burtscher, M.: Fast lossless compression of scientific floating-point data. In: Proceedings of the Data Compression Conference, DCC '06, pp. 133–142. IEEE Computer Society, Washington, DC, USA (2006). DOI 10.1109/DCC.2006.35. URL http://dx.doi.org/10.1109/DCC.2006.35
15. Rumpfkeil, M., Zingg, D.: A General Framework for the Optimal Control of Unsteady Flows with Applications. American Institute of Aeronautics and Astronautics (2007). DOI doi: 10.2514/6.2007-1128. URL http://dx.doi.org/10.2514/6.2007-1128
16. Tim Wildey, E.C.C., Shadid, J.: Adjoint based a posteriori error estimates using data compression. In: C.T. J. P. Moitinho de Almeida P. D ez, N. Parés (eds.) VI International Conference on Adaptive Modeling and Simulation (2013)
17. Wang, Q., Hu, R., Blonigan, P.: Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. Journal of Computational Physics **267**(0), 210 – 224 (2014). DOI http://dx.doi.org/10.1016/j.jcp.2014.03.002. URL http://www.sciencedirect.com/science/article/pii/S0021999114001715
18. Xu, S., Jahn, W., Müller, J.D.: Cad-based shape optimisation with cfd using a discrete adjoint. International Journal for Numerical Methods in Fluids **74**(3), 153–168 (2014)
19. Xu, S., Radford, D., Meyer, M., Müller, J.D.: Stabilisation of discrete steady adjoint solvers. Journal of Computational Physics **(accepted for publication)**, – (2015). DOI http://dx.doi.org/10.1016/j.jcp.2015.06.036. URL http://www.sciencedirect.com/science/article/pii/S0021999115004349