



Adjoint-Based Methodology for Time-Dependent Aerodynamic Optimization

Nail K. Yamaleev

*Department of Mathematics
North Carolina A&T State University*

SEAMS Workshop, Oct. 31 – Nov. 2, 2008



Acknowledgements

This work has been funded by the NASA NRA grant NNL07AA23C.

Collaborators

B. Diskin, *National Institute of Aerospace*

E. J. Nielsen, *NASA Langley Research Center*

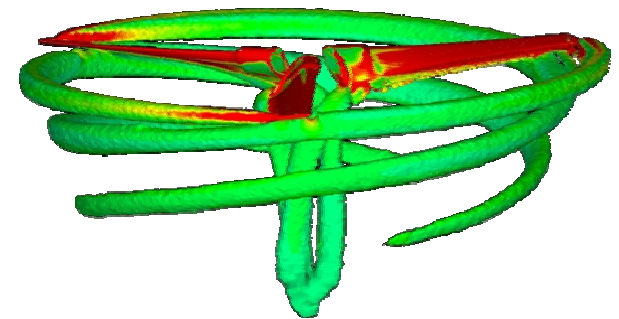
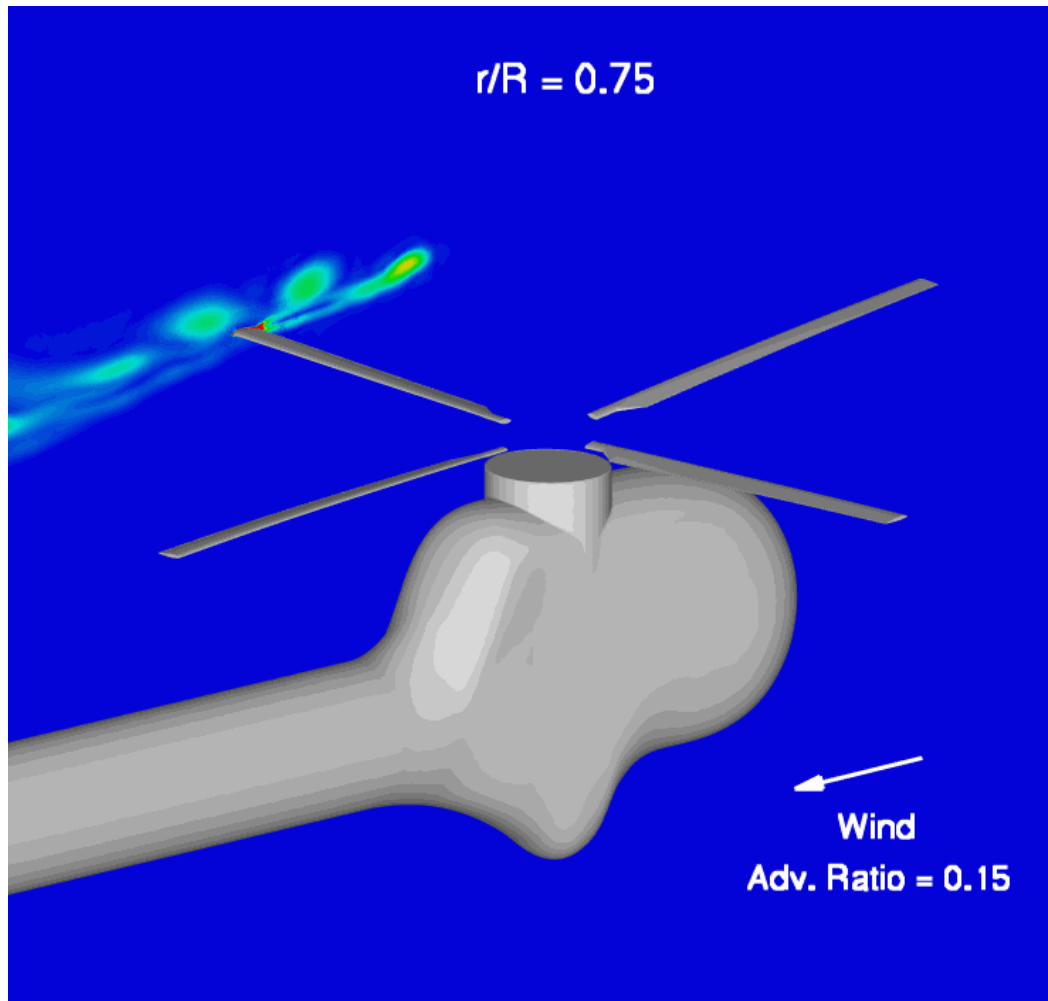


Outline

- Motivation
- Brief overview of aerodynamic optimization methods
- Time-dependent adjoint equations
- Sensitivity derivatives and optimization procedure
- Numerical results and validation
- Computational complexity issues
- Future directions

Motivation

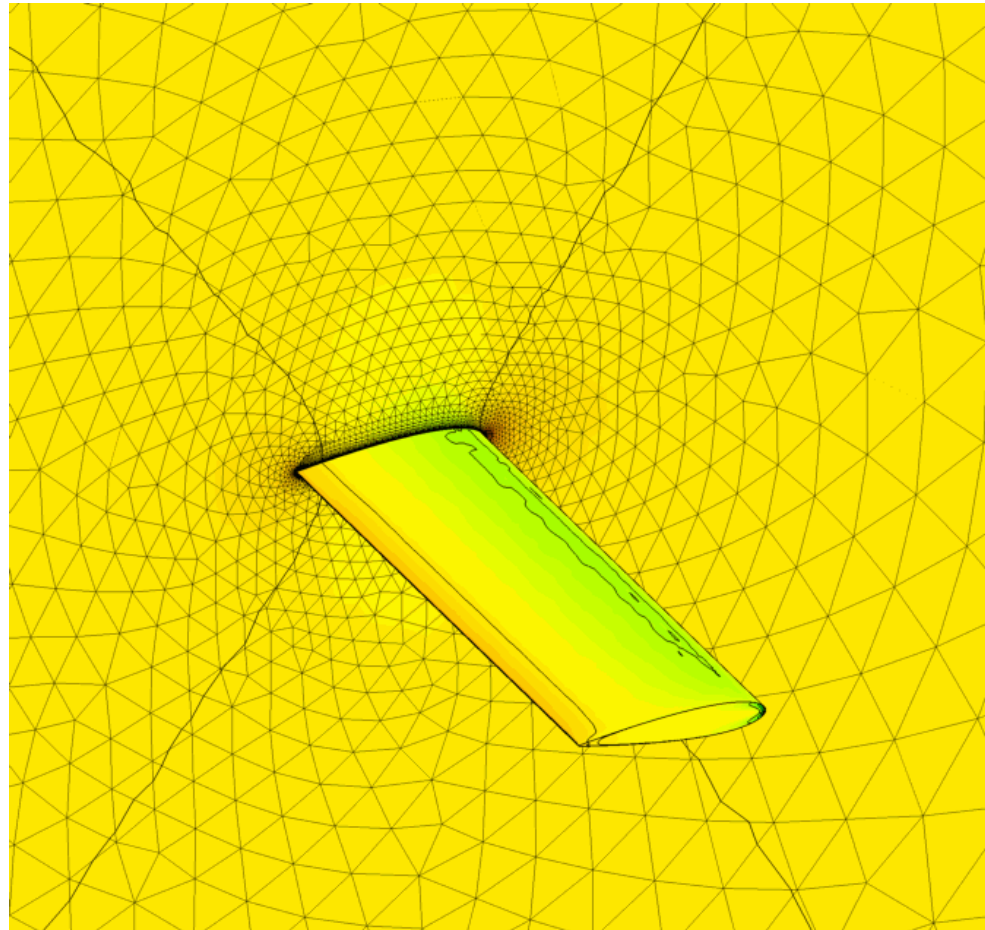
Rotorcraft design/optimization





Motivation (cont.)

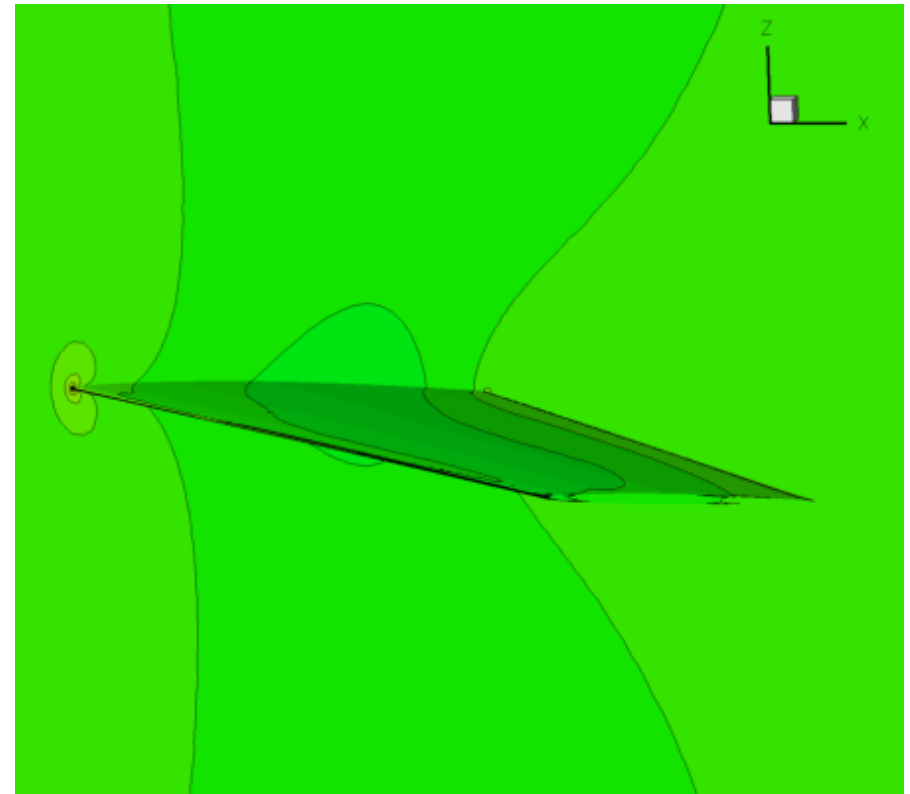
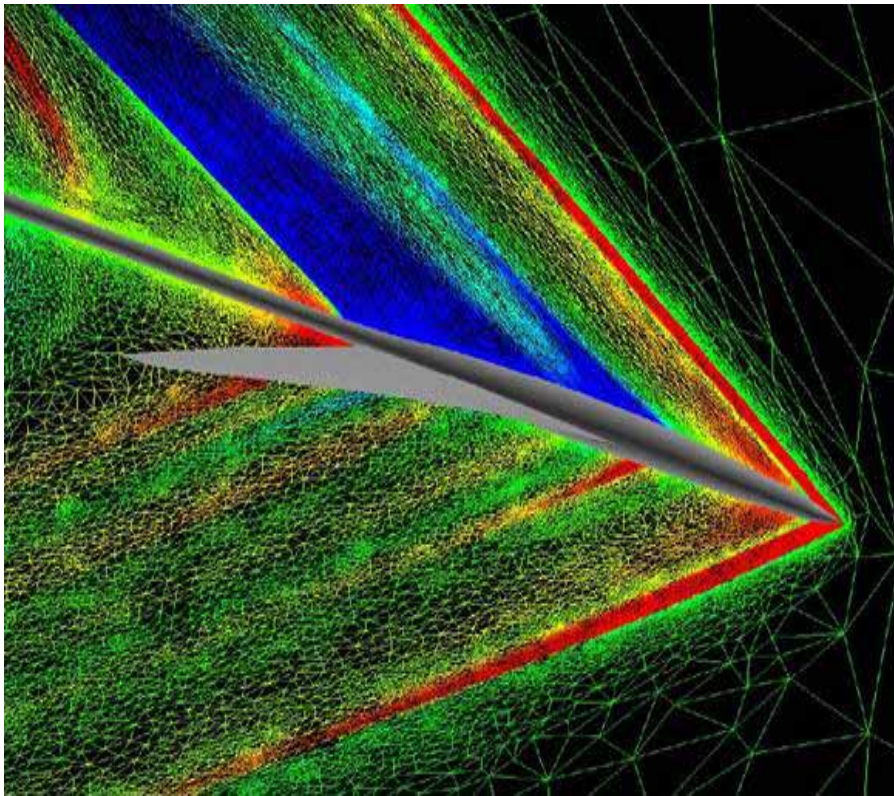
Optimization of aeroelastic configurations



Courtesy of the FUN3D team, NASA LaRC

Motivation (cont.)

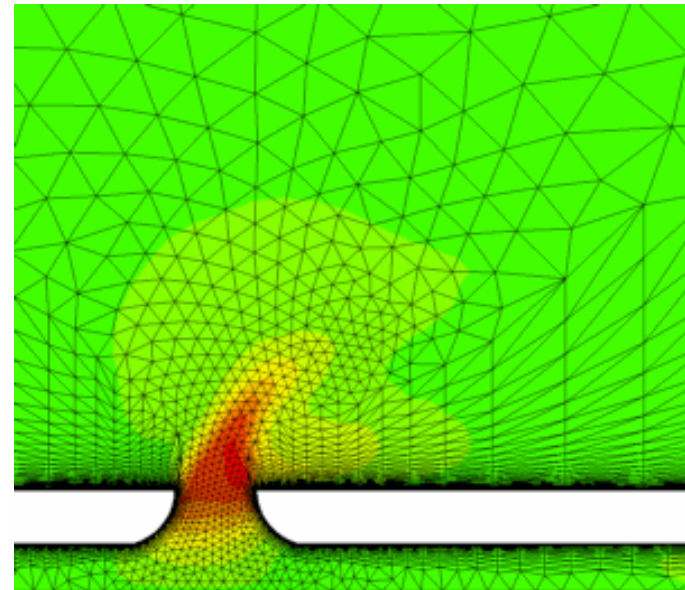
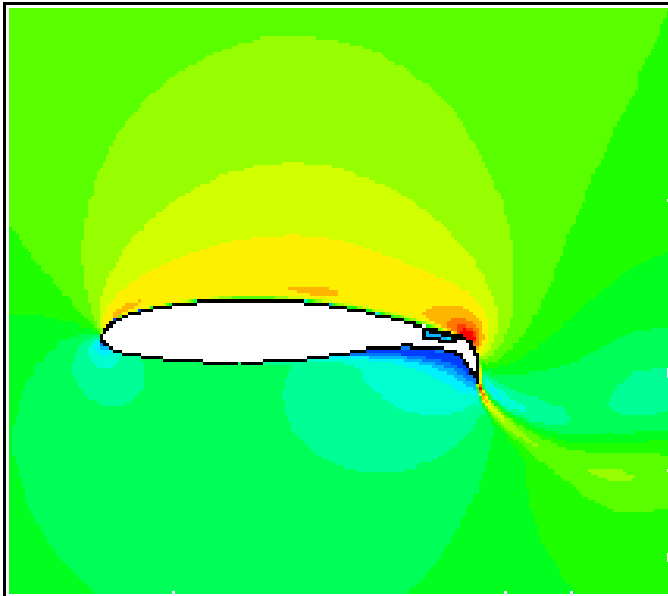
Supersonic flutter/ vibration control



Courtesy of the FUN3D team, NASA LaRC

Motivation (cont.)

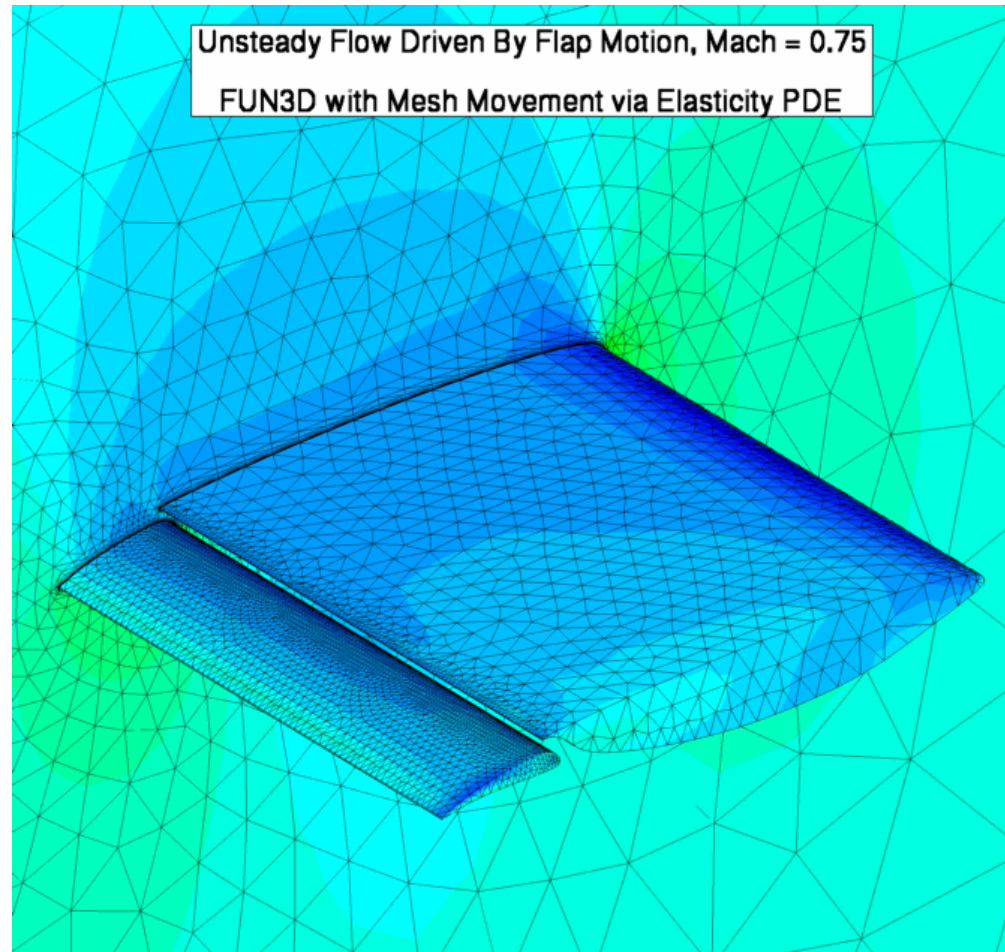
Active flow control



Courtesy of the FUN3D team, NASA LaRC

Motivation (cont.)

Maneuvering/morphing



Courtesy of the FUN3D team, NASA LaRC

Aerodynamic Optimization Problem



Find control/design variables \mathbf{D} and states \mathbf{Q} such that

$$\min_{\mathbf{D} \in \mathcal{D}_a} f(\mathbf{Q}, \mathbf{D}); \quad f = \int_0^{T_f} F_{\text{obj}}(\mathbf{Q}, \mathbf{D}) dt$$

s.t. the unsteady Euler/Navier-Stokes equations:

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \mathbf{F}(\mathbf{Q}, \mathbf{D}) = 0$$

inequality constraints: $\varphi(\mathbf{Q}, \mathbf{D}) \leq 0$

equality constraints: $\psi(\mathbf{Q}, \mathbf{D}) = 0$

side constraints: $\|\mathbf{D}\| \leq b$

- f , φ , ψ are not readily available!
- The number of design variables \mathbf{D} is very large.

The Unsteady Reynolds-Averaged Navier-Stokes Equations



$$V \frac{\partial Q}{\partial t} + \oint_{\Omega} (\vec{F}_i \cdot \hat{n}) d\Omega - \oint_{\Omega} (\vec{F}_v \cdot \hat{n}) d\Omega = 0 \quad Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} \quad \vec{F}_i = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (E+p)u \end{bmatrix} \hat{i} + \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ (E+p)v \end{bmatrix} \hat{j} + \begin{bmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ (E+p)w \end{bmatrix} \hat{k}$$

$$\vec{F}_v = f_v \hat{i} + g_v \hat{j} + h_v \hat{k} \quad f_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{bmatrix} \quad g_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{zy} - q_y \end{bmatrix} \quad h_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \end{bmatrix}$$

$$\tau_{xx} = (\mu + \mu_t) \frac{M_{\infty}^2}{Re} \frac{2}{3} [2u_x - (v_y + w_z)] \quad \tau_{yy} = (\mu + \mu_t) \frac{M_{\infty}^2}{Re} \frac{2}{3} [2v_y - (u_x + w_z)] \quad \tau_{zz} = (\mu + \mu_t) \frac{M_{\infty}^2}{Re} \frac{2}{3} [2w_z - (u_x + v_y)]$$

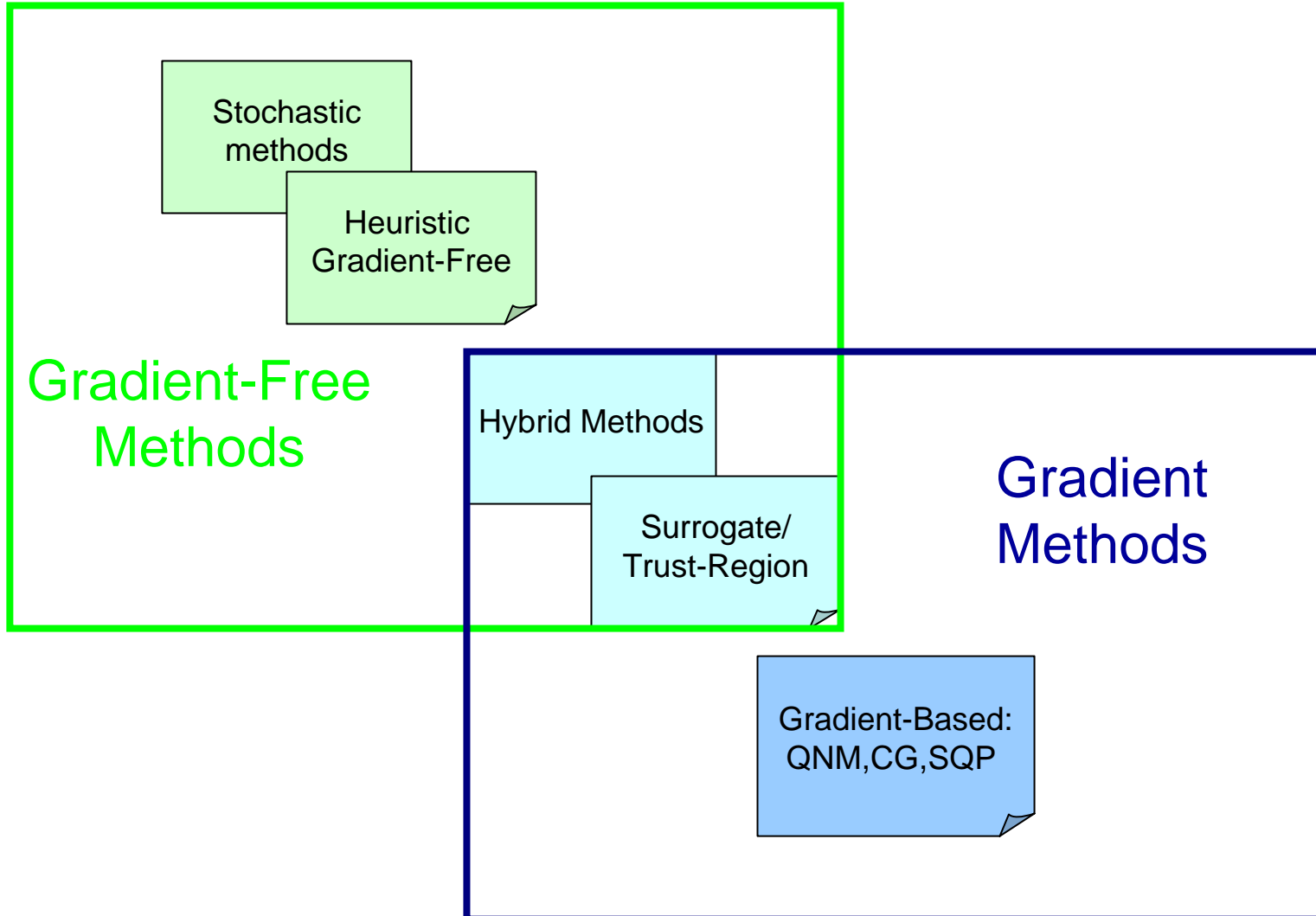
$$\tau_{xy} = \tau_{yx} = (\mu + \mu_t) \frac{M_{\infty}^2}{Re} (u_y + v_x) \quad \tau_{xz} = \tau_{zx} = (\mu + \mu_t) \frac{M_{\infty}^2}{Re} (u_z + w_x) \quad \tau_{yz} = \tau_{zy} = (\mu + \mu_t) \frac{M_{\infty}^2}{Re} (v_z + w_y)$$

$$q_x = \frac{-M_{\infty}}{Re(\gamma-1)} \left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t} \right) \frac{\partial a^2}{\partial x} \quad q_y = \frac{-M_{\infty}}{Re(\gamma-1)} \left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t} \right) \frac{\partial a^2}{\partial y} \quad q_z = \frac{-M_{\infty}}{Re(\gamma-1)} \left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t} \right) \frac{\partial a^2}{\partial z}$$



Optimization Methods

Search Methods





Gradient Methods

The necessary condition of extrema: $\nabla f(\mathbf{D}) = 0$

Steepest descent: $\mathbf{D}_{k+1} = \mathbf{D}_k - \delta_k \nabla f(\mathbf{D}_k)$

Conjugate gradients: $\mathbf{D}_{k+1} = \mathbf{D}_k - \delta_k \left[\nabla f(\mathbf{D}_k) + \frac{|\nabla f(\mathbf{D}_k)|^2}{|\nabla f(\mathbf{D}_{k-1})|^2} \mathbf{S}_{k-1} \right]$

Quasi-Newton methods: $\mathbf{D}_{k+1} = \mathbf{D}_k - \delta_k H_k \nabla f(\mathbf{D}_k)$



Gradient Methods (cont.)

Gradient Estimation

Finite Differences

$$\frac{df}{dD_m} = \frac{f(\mathbf{D} + h\mathbf{e}_m) - f(\mathbf{D} - h\mathbf{e}_m)}{2h} + O(h^2)$$

Complex Step Method

$$\frac{df}{dD_m} = \frac{\text{Im}[f(\mathbf{D} + i h \mathbf{e}_m)]}{h} + O(h^2)$$

Automatic Differentiation

$$\frac{df(\mathbf{Q}(\mathbf{D}), \mathbf{D})}{d\mathbf{D}} = \frac{\partial f}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial \mathbf{D}} + \frac{\partial f}{\partial \mathbf{D}}$$

Gradients through sensitivities

$$\frac{\partial}{\partial t} \left(\frac{\partial \mathbf{Q}}{\partial \mathbf{D}} \right) + \nabla \left(\frac{\partial F(\mathbf{Q}, \mathbf{D})}{\partial \mathbf{D}} \right) = 0$$

The computational cost **depends** on the number of design variables!

Adjoint:
Continuous /
Discrete

The computational cost is **independent** of the number of design variables!



Gradient Methods (cont.)

Continuous Adjoints vs. Discrete Adjoints

Continuous N - S equations

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \mathbf{F} = 0$$

Continuous adjoint equations

$$-\frac{\partial \Lambda}{\partial t} + \left(\frac{\partial \mathbf{F}}{\partial \mathbf{Q}} \right)^T \nabla \Lambda = -\frac{\partial f}{\partial \mathbf{Q}}$$

Pros:

- No mesh sensitivities are required

Cons:

- The gradient may be inconsistent for shocks or if the mesh is not sufficiently fine.
- BC have to be imposed and implemented again for different cost functionals.
- A new code is needed for the adjoints.

Discrete N - S equations

$$\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}(\mathbf{Q}^n, \mathbf{D}) = 0$$

Discrete adjoint equations

$$-\frac{\Lambda^n - \Lambda^{n+1}}{\Delta t} + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right)^T \Lambda = -\frac{\partial f}{\partial \mathbf{Q}^n}$$

Pros:

- Provides the exact discrete gradient on any mesh

Cons:

- Requires the mesh sensitivities.

Discrete Time-Dependent Optimization Problem



Discrete optimization problem:

$$\min_{\mathbf{D} \in D_a} f(\mathbf{Q}, \mathbf{X}, \mathbf{D}); \quad f = \sum_{n=1}^N f^n(\mathbf{Q}^n, \mathbf{X}^n, \mathbf{D}^n) \Delta t; \quad f^n = F_{\text{obj}}^n + F_{\text{pen}}^n$$

Objective functional: $F_{\text{obj}}^n = \beta_1 \left(C_L^n - C_{L_{\text{target}}}^n \right)^2 + \beta_2 \left(C_D^n - C_{D_{\text{target}}}^n \right)^2$

Penalty term:

$$F_{\text{pen}}^n = \frac{\alpha_1}{2} [\mathbf{D}^n]^T \mathbf{D}^n + \frac{\alpha_2}{2\Delta t^2} [\mathbf{D}^n - \mathbf{D}^{n-1}]^T (\mathbf{D}^n - \mathbf{D}^{n-1}) + \alpha_3 \left(\max[0, \varphi(\mathbf{Q}^n, \mathbf{D}^n)] \right)^2 + \alpha_4 [\psi(\mathbf{Q}^n, \mathbf{D}^n)]^2$$

s.t. the discretized Euler/Navier-Stokes and grid equations:

$$\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}^n(\mathbf{Q}^n, \mathbf{X}^n, \mathbf{D}^n) = 0$$

$$K\mathbf{X}^n - \mathbf{X}_{\text{surf}}^n = 0$$

Discrete Time-Dependent Adjoint Equations



Lagrangian functional:

$$L(\mathbf{Q}, \mathbf{X}, \mathbf{D}, \Lambda_f, \Lambda_g) = \sum_{n=1}^N f^n \Delta t + \sum_{n=1}^N [\Lambda_f^n]^T \left(\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}^n \right) \Delta t + \sum_{n=1}^N [\Lambda_g^n]^T (K\mathbf{X}^n - \mathbf{X}_{surf}^n) \Delta t$$

$$\mathbf{Q}^0 = \mathbf{Q}_{in} - \text{initial condition}$$

$$\Lambda_f^{N+1} = \mathbf{0}$$

Differentiating L with respect to \mathbf{D} yields:

$$\frac{dL}{d\mathbf{D}^n} = \left\{ \sum_{n=1}^N \frac{\partial f^n}{\partial \mathbf{D}^n} \Delta t + \sum_{n=2}^N \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{D}^n} \right]^T \Lambda_f^n \Delta t - \sum_{n=1}^N \left[\frac{\partial \mathbf{X}_{surf}^n}{\partial \mathbf{D}^n} \right]^T \Lambda_g^n \Delta t - \left[\frac{\partial \mathbf{Q}_{in}}{\partial \mathbf{D}^n} \right]^T \Lambda_f^1 \right\}$$

$$+ \sum_{n=1}^N \left[\frac{\partial \mathbf{Q}^n}{\partial \mathbf{D}^n} \right]^T \left(\frac{\Lambda_f^n - \Lambda_f^{n+1}}{\Delta t} + \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n} \right]^T \Lambda_f^n + \frac{\partial f^n}{\partial \mathbf{Q}^n} \right) \Delta t + \sum_{n=1}^N \left[\frac{\partial \mathbf{X}^n}{\partial \mathbf{D}^n} \right]^T \left(K\Lambda_g^n + \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{X}^n} \right]^T \Lambda_f^n + \frac{\partial f^n}{\partial \mathbf{X}^n} \right) \Delta t$$

0
0

Discrete Time-Dependent Adjoint Equations (cont.)



Unsteady flow adjoint equations are integrated **backward in time!**

$$\frac{\Lambda_f^n - \Lambda_f^{n+1}}{\Delta t} + \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n} \right]^T \Lambda_f^n = - \frac{\partial f^n}{\partial \mathbf{Q}^n}$$

Grid adjoint equations:

$$K \Lambda_g^n = - \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{X}^n} \right]^T \Lambda_f^n - \frac{\partial f^n}{\partial \mathbf{X}^n}$$

The same adjoint equations can be used regardless of the number of the control variables \mathbf{D} !

Sensitivity derivative:

$$\frac{dL}{d\mathbf{D}} = \sum_{n=1}^N \frac{\partial f^n}{\partial \mathbf{D}^n} \Delta t + \sum_{n=2}^N \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{D}^n} \right]^T \Lambda_f^n \Delta t - \sum_{n=1}^N \left[\frac{\partial \mathbf{X}_{surf}^n}{\partial \mathbf{D}^n} \right]^T \Lambda_g^n \Delta t - \left[\frac{\partial \mathbf{Q}_{in}}{\partial \mathbf{D}} \right]^T \Lambda_f^1$$

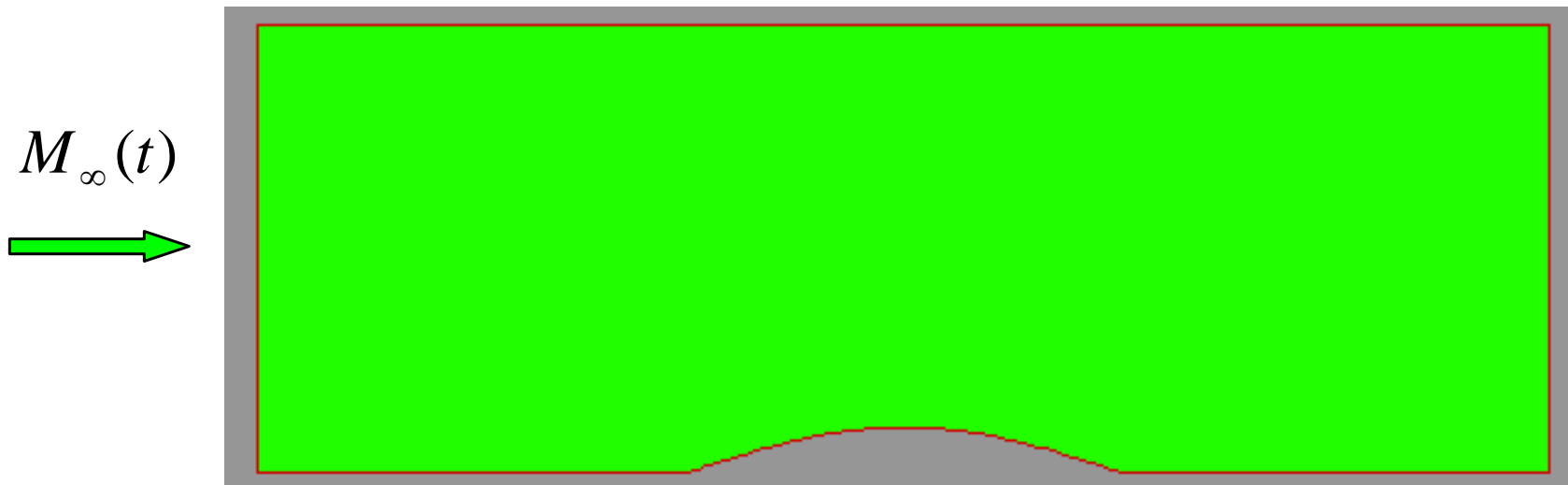
Search procedure: $\mathbf{D}_{k+1} = \mathbf{D}_k - \delta_k \frac{dL}{d\mathbf{D}}$



Numerical Results and Validation

2-D unsteady Euler equations

Supersonic bump flow: $M_\infty(t) = 2 + \Delta M \cos \omega t$

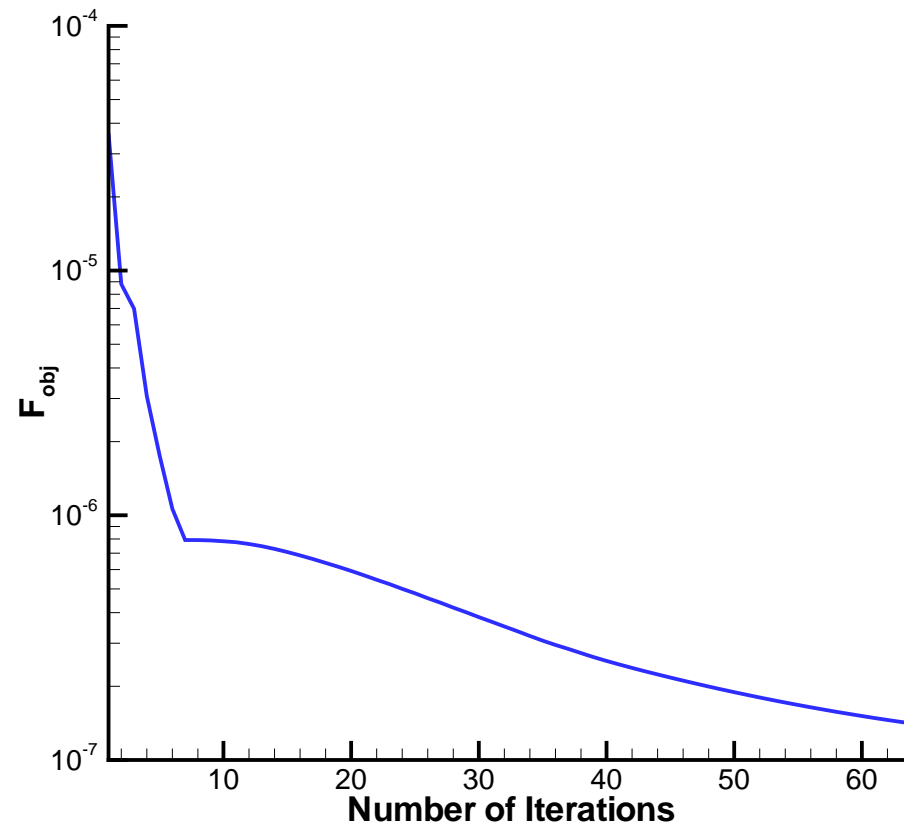




Optimal Control Problem

Objective functional: $F_{\text{obj}}^n = \sum_{j \in \Gamma_c} \left(P_j^n - P_{\text{target}}^n \right)^2$; $M_{\text{target}}(t) = 2 + 0.5 \cos(17\pi t / 9)$

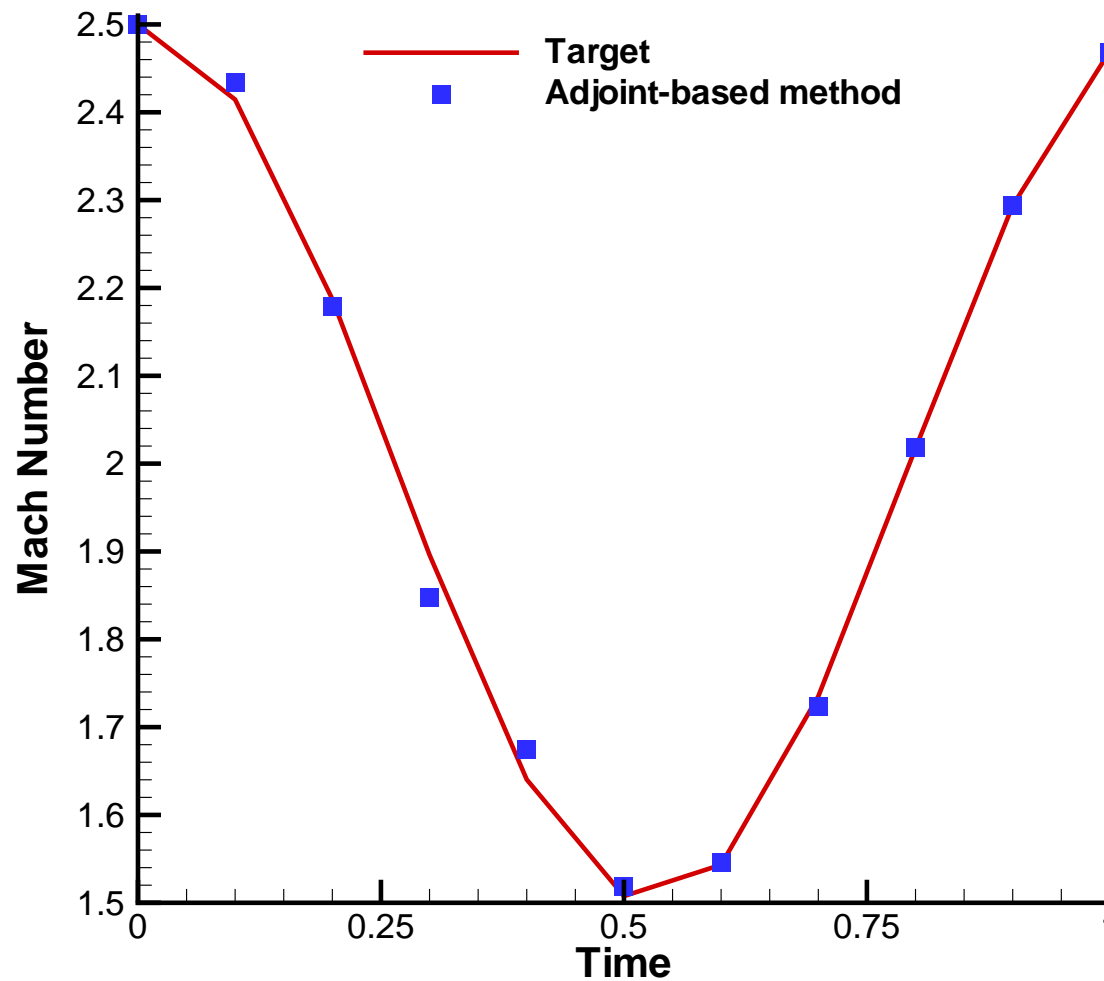
Control variables: $\mathbf{D} = (M_1, \dots, M_N)^T$





Optimal Control Problem (cont.)

Comparison of the target and optimal Mach numbers

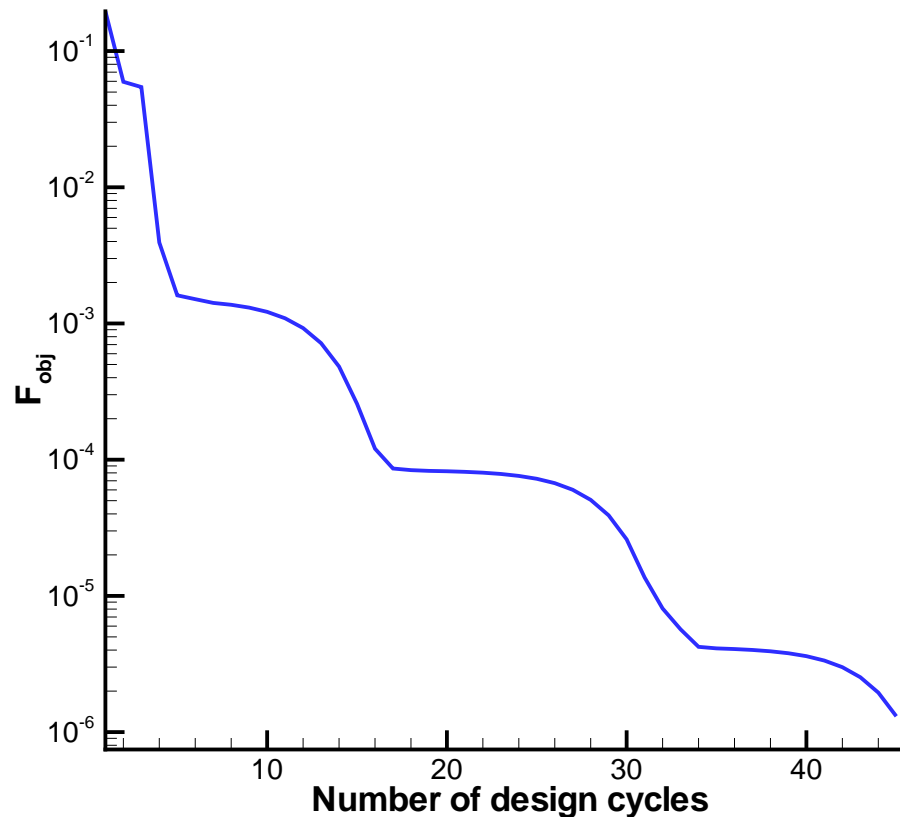




Design Optimization Problem

Objective functional: $F_{\text{obj}}^n = \sum_{j \in \Gamma_c} \left(P_j^n - P_{\text{target}}^n \right)^2$; $M_{\infty}(t) = 2 + 0.2 \cos(17\pi t / 9)$

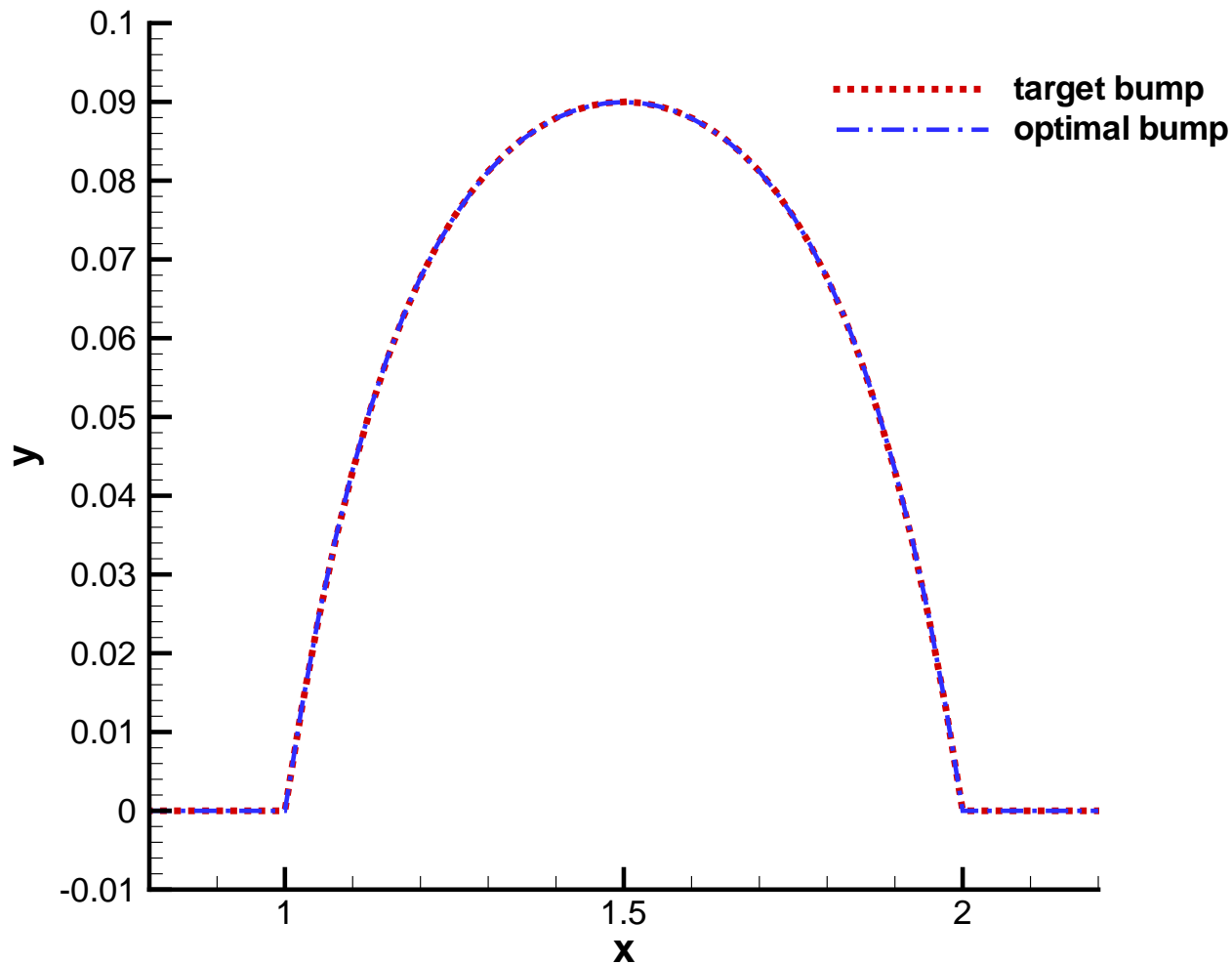
Control variables: $\mathbf{D} = (d_1, d_2, d_3)^T$, $y = d_1\psi_1(x) + d_2\psi_2(x) + d_3\psi_3(x)$



Design Optimization Problem (cont.)



Comparison of the target bump and the optimal bump

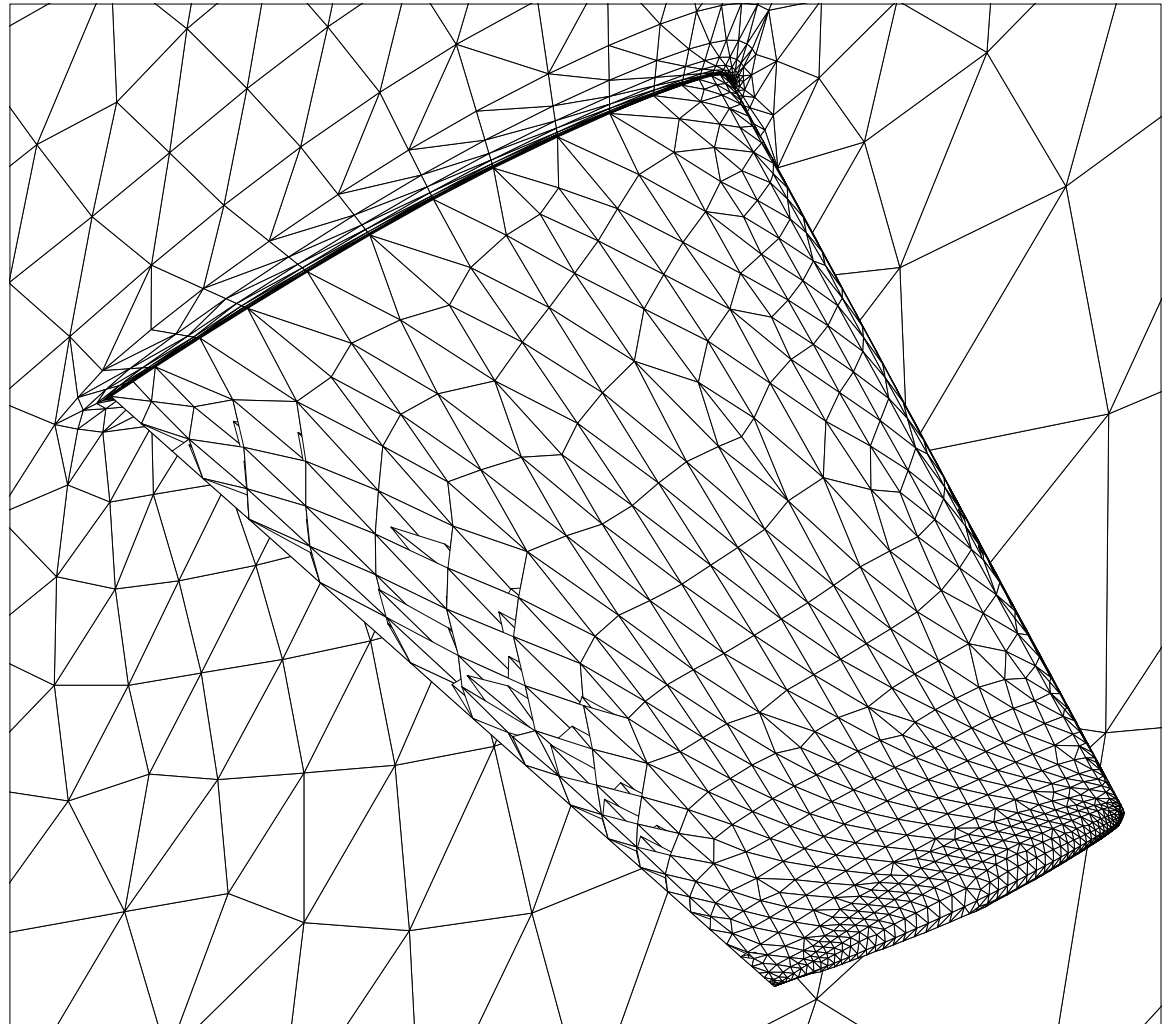


Pitching 3-D Wing

$M_\infty = 0.3$, $Re_\infty = 10^6$,
 $\alpha = \pm 3^\circ$, URANS + SA

$$F_{\text{obj}}^n = \left(\frac{C_L}{C_D} - 10 \right)^2$$

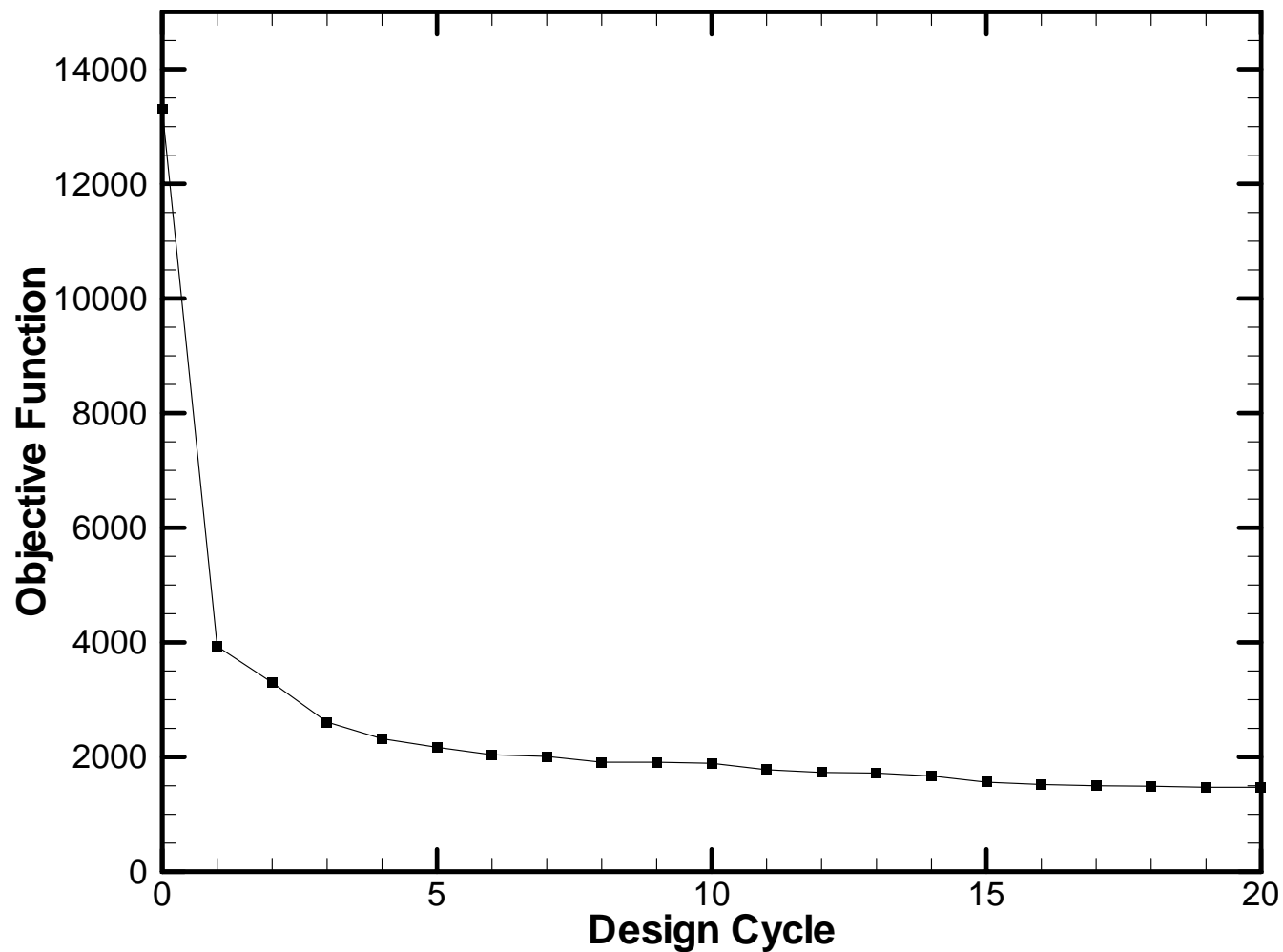
55 design variables



Pitching 3-D Wing (cont.)



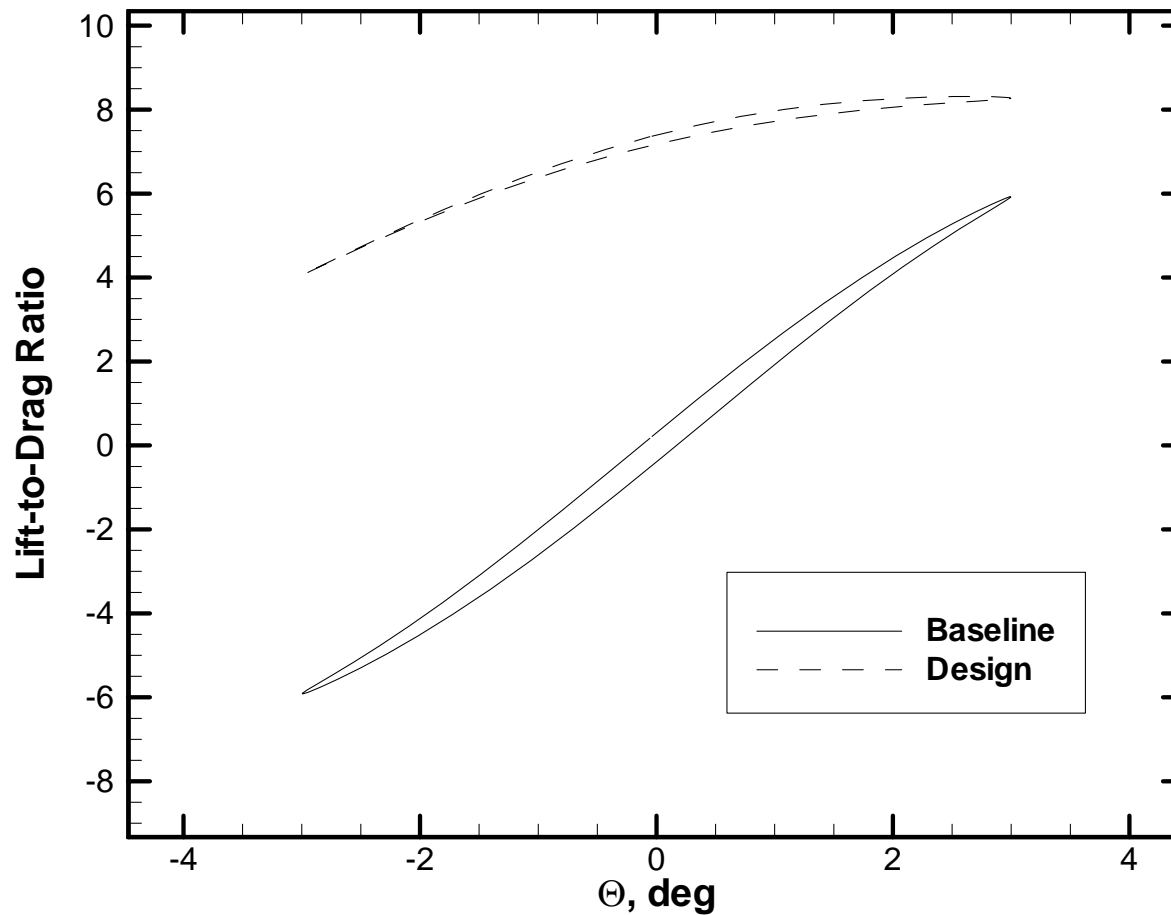
Convergence history



Pitching 3-D Wing (cont.)



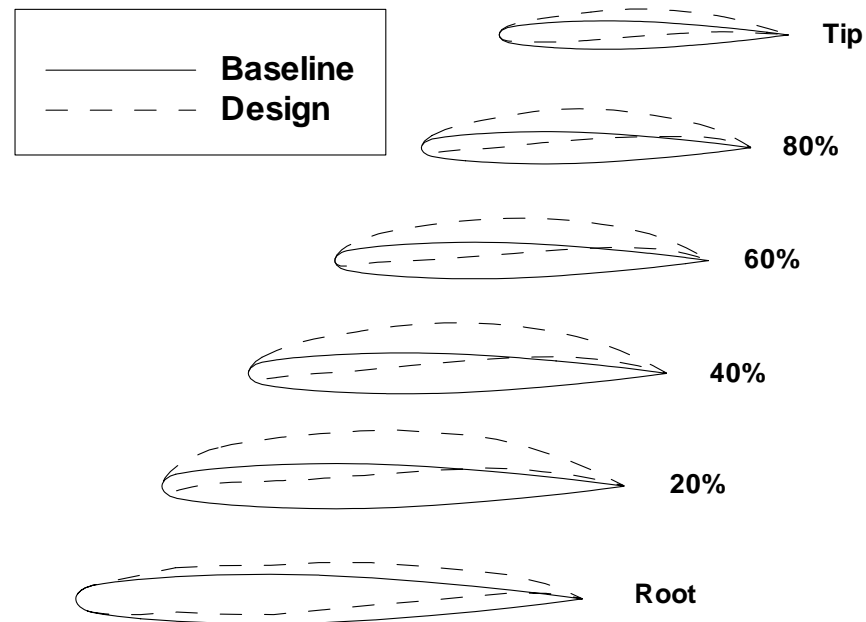
Lift-to-Drag ratio



Pitching 3-D Wing (cont.)



Optimal design



Computational Complexity



Memory Requirements

The flow solution must be known at all time levels!

For 100,000 grid points per processor, 1000 time steps, the memory cost of a typical N-S discretization is of the order of $O(10)$ - $O(100)$ Gb.

Factors affecting the CPU time:

- Multiple evaluations of the primal and adjoint variables
- Resolution of multiple spatial and temporal scales
- Linearization cost
- Mesh movement
- Cost associated with physical models (turbulence, aeroelasticity, etc.)



Memory Saving Techniques

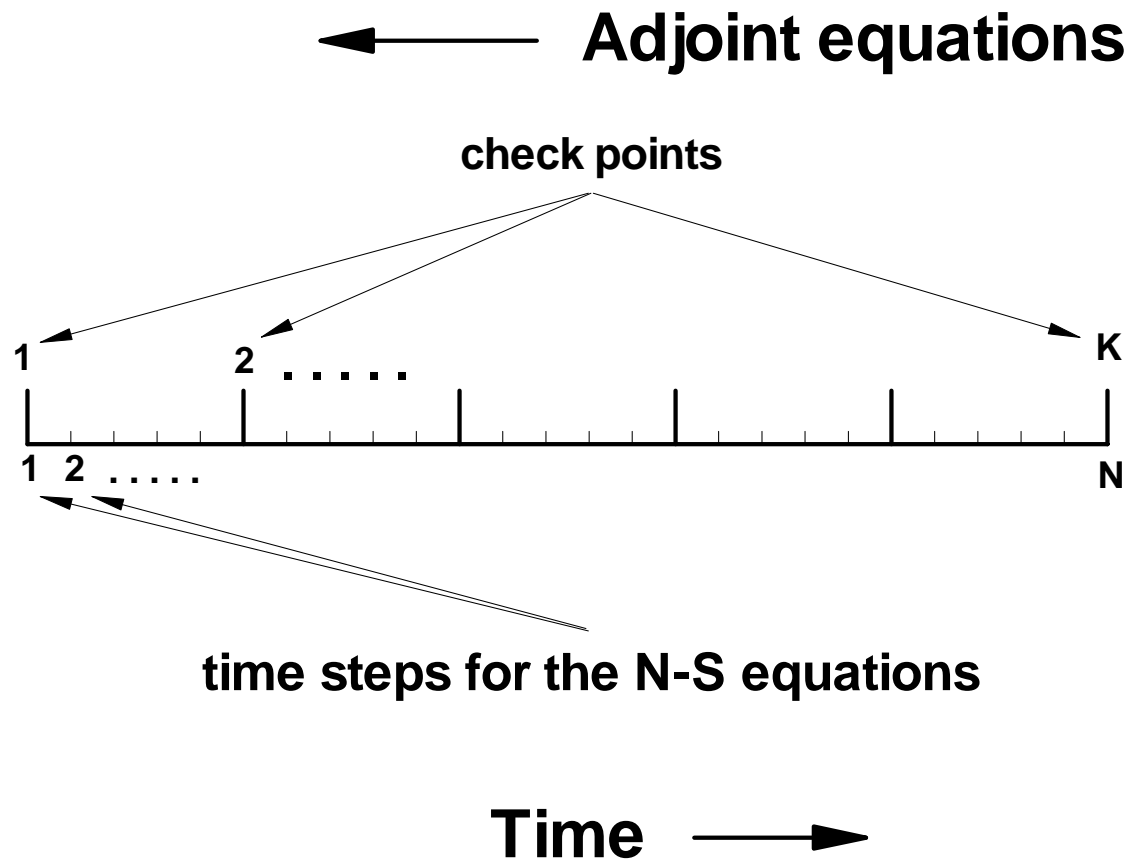
Checkpointing approach

Pros:

- Provides the same accuracy as the full method
- Reduces storage by a factor N/K .

Cons:

- Doubles the CPU time.



Memory Saving Techniques (cont.)



Time-spectral method

Approximate \mathbf{Q} by a discrete Fourier series

$$\mathbf{Q}_j = \sum_{m=-K/2}^{K/2-1} \hat{\mathbf{Q}}_m e^{imt_j} \quad (j = 0, \dots, K-1), \quad t_j = \frac{T}{K} j$$

The discretized Navier - Stokes equations are written the following form :

$$VD\mathbf{Q}_j + R(\mathbf{Q}_j) = 0, \quad (j = 0, \dots, K-1),$$

where
$$D_{lj} = \frac{1}{N} \sum_{m=-K/2}^{K/2-1} ime^{im(t_l-t_j)}$$

Pros:

- Efficient for periodic or quasi-periodic problems.
- Reduces storage by factor N/K , where K is the number of modes.

Cons:

- Is not applicable for non-periodic problems.
- Computational cost is large if the Fourier series converges slowly.



Memory Saving Techniques (cont.)

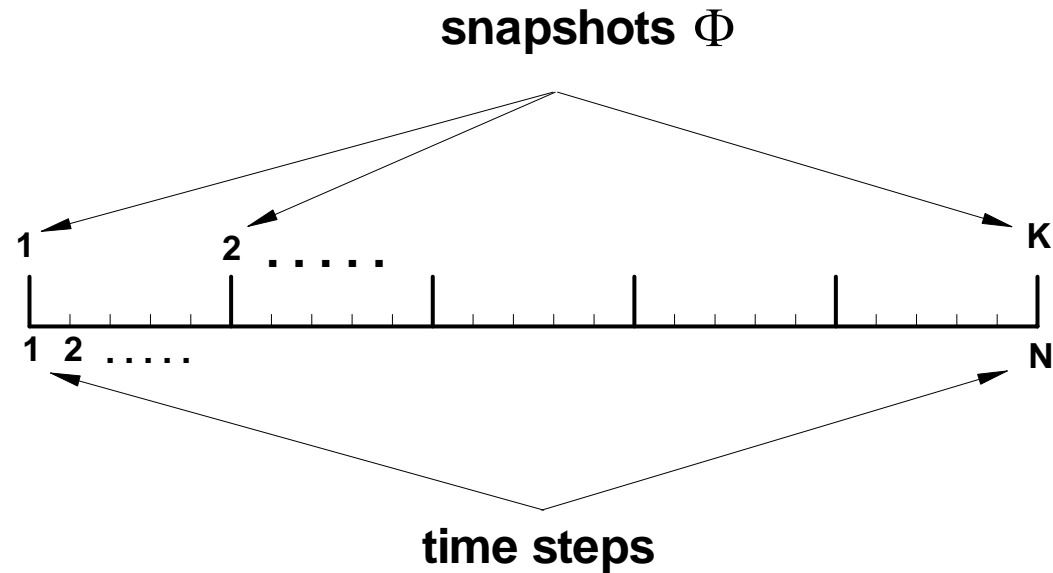
POD-based reduced-order model

Pros:

- Reduces storage by a factor N/K .
- The same set of POD basis functions can be used over several optimization iterations.

Cons:

- No good strategy for selecting snapshots.
- The POD basis has to be reconstructed if the flow differs from the state for which the basis functions were constructed.



$$K \ll N$$

Memory Saving Techniques (cont.)



Locally optimal methods

for $k = 0, 1, \dots, K - 1$ minimize the functional

$$\min_{t \in [T_k, T_{k+1}]} f_k(\mathbf{Q}, \mathbf{D}), \quad f_k = \sum_{m=N_k}^{N_{k+1}} F(\mathbf{Q}^m, \mathbf{D}) \Delta t$$

with the initial condition : $\mathbf{Q}^{in} = \mathbf{Q}^m$

Pros:

- The storage cost is comparable with that of the steady problem.
- Each local problem is much smaller than the global-in-time problem.

Cons:

- Solution of the optimization problem is suboptimal.

CPU Time Saving Techniques



- High-order spatial discretizations (DG, SUPG, FV, FD)
- High-order temporal discretization (RK, BDF)
- Efficient multigrid solvers
- POD-based ROMs
- Adjoint-based grid adaptation for unsteady flows



Future Directions

- Reduce storage and computational costs
 - Locally optimal schemes
 - POD-based reduced-order models
 - High-order methods
 - Efficient multigrid solvers
- Adjoint-based time-dependent grid adaptation
- Demonstrate the capabilities of the present time-dependent adjoint-based methodology on large-scale problems.



Questions ?