Strategies for computing second-order derivatives in CFD design problems

Massimiliano Martinelli   Alain Dervieux   Laurent Hascoët

INRIA Sophia-Antipolis
Project TROPICS

WEHSFF, Moscou, november 20, 2007

Nonlinear constrained functional

The problem

- Given a computer program computing a functional $j(c) = J(c, W(c))$ we want, applying a *source-to-source programme differentiation software*, viz. TAPENADE, to get a computer program computing the second derivatives $\dfrac{d^2 j}{dc^2} = (H_{ii})$
- $W(c)$ is solution of the state equation $\Psi(c, W) = 0$

$$\Longrightarrow \quad \frac{\partial \Psi}{\partial c} + \frac{\partial \Psi}{\partial W}\frac{dW}{dc} = 0$$

- W obtained by *explicit* or *implicit* pseudo-time advancing techniques
- $c \in \mathbb{R}^n$ and $W \in \mathbb{R}^N$ with $n \ll N$

Remark

We assume that the solution $W(c)$ *is not time-dependent* (steady-state solution)

Why we need Hessian?

Perturbative methods for uncertainty propagation (Taylor expansion-based)

- Method of Moments

$$\begin{cases} \mu_j \simeq j(\mu_c) + \frac{1}{2}\sum_i H_{ii}\sigma_i^2 \\ \sigma_j^2 \simeq \sum_i G_i^2\sigma_i^2 + \frac{1}{2}\sum_{i,k} H_{ik}^2\sigma_i^2\sigma_k^2 \end{cases}$$

- “Inexpensive Monte Carlo” methods of M.Giles.

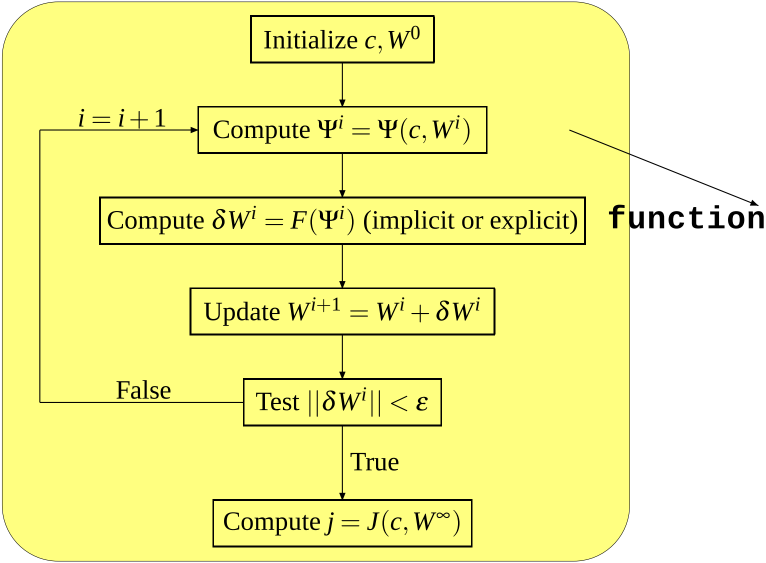
Robust optimization

- Gradient-based methods for $j^{\text{robust}}(c) = j(c) + \varepsilon||\frac{dj}{dc}||$
- Gradient-free methods for $j^{\text{robust}}(c) = j(c) + \frac{1}{2}\sum_i H_{ii}\sigma_i^2$
- Gradient-free methods for $j^{\text{robust}}(c) = j(c) + k\sigma_j^2$

Adjoint-corrected functionals

- Gradient-based methods for $j^{\text{corr}}(c) = j(c) - \left\langle \Psi_{\text{ex}}(c,W), \Pi_0 \right\rangle$

Flow solver: basic algorithm



Initialize c, W^0

$i = i + 1$

Compute $\Psi^i = \Psi(c, W^i)$

Compute $\delta W^i = F(\Psi^i)$ (implicit or explicit)

Update $W^{i+1} = W^i + \delta W^i$

False

Test $||\delta W^i|| < \varepsilon$

True

Compute $j = J(c, W^\infty)$

`functional(j,c)`

Differentiability/Differentiation modes

For a given $\varepsilon > 0$

Functional j is only piecewise differentiable. Values of state W depend on initial conditions of solution algorithm, in a similar manner to unsteady system.

Assuming $\varepsilon = 0$

Functional j is differentiable. Values of state W do not depend on initial conditions of solution algorithm (if convergent).

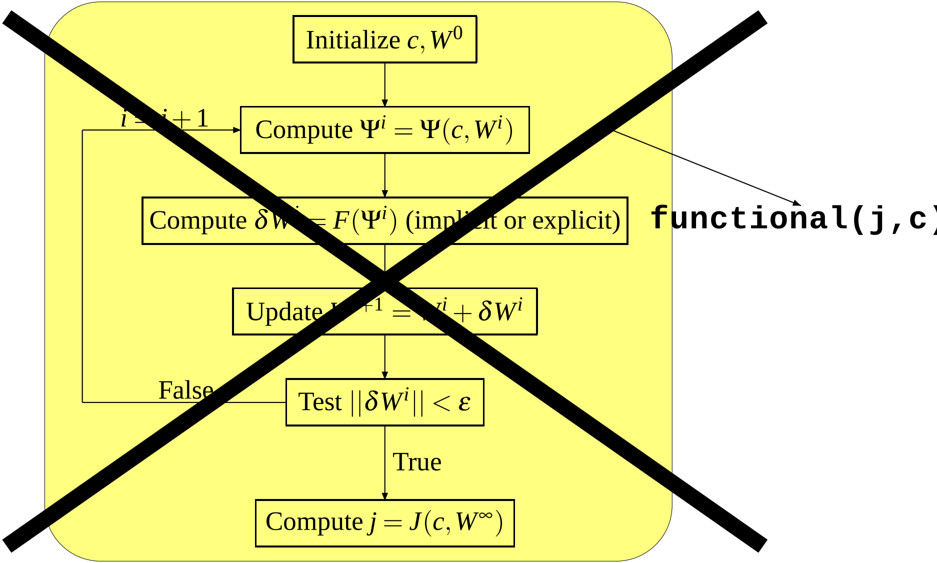
Direct/Tangent mode

- Differentiated code computes $j'(c).\delta c$
- Computational cost factor: $\alpha_T \approx 4$
- Does not store intermediate variables

Backward/Reverse mode

- Differentiated code computes $(j'(x))^*.\delta j$
- Computational cost factor: $\alpha_R \approx 5$
- Stores intermediate variables

Flow solver: basic algorithm



Initialize c, W^0

Compute $\Psi^i = \Psi(c, W^i)$

Compute $\delta W^i = F(\Psi^i)$ (implicit or explicit)

Update $W^{i+1} = W^i + \delta W^i$

Test $||\delta W^i|| < \varepsilon$

False

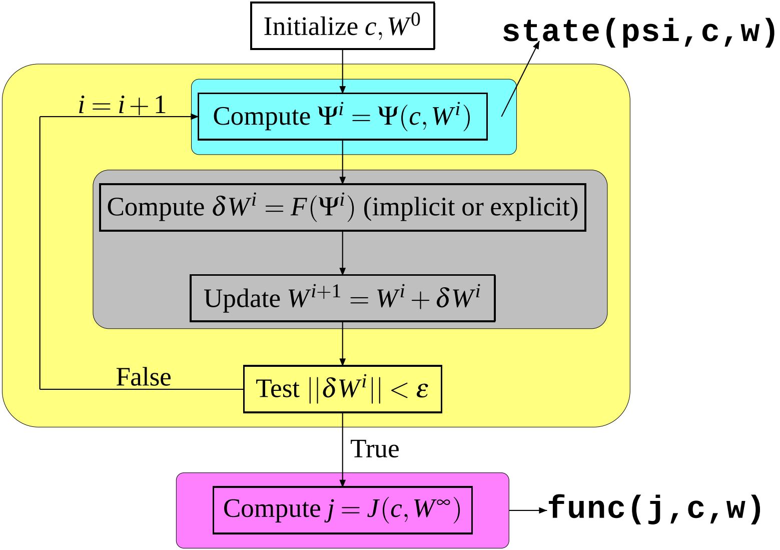
$i = i + 1$

True

Compute $j = J(c, W^\infty)$

`functional(j,c`

Flow solver: basic algorithm



Initialize c, W^0

`state(psi,c,w)`

$i = i + 1$

Compute $\Psi^i = \Psi(c, W^i)$

Compute $\delta W^i = F(\Psi^i)$ (implicit or explicit)

Update $W^{i+1} = W^i + \delta W^i$

False

Test $||\delta W^i|| < \varepsilon$

True

Compute $j = J(c, W^\infty)$

`func(j,c,w)`

Non-differentiated matrix-free iterative solver

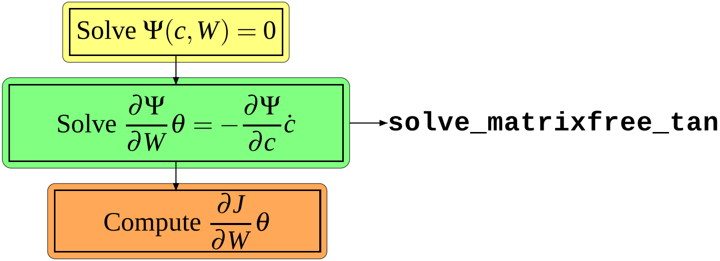
- We use iterative methods to solve $Ax = b$
- Loop of matrix-by-vector multiplications
- Re-engineering of preconditioner
- To compute matrix-by-vector multiplication:
  - if $A = \left(\frac{\partial \Psi}{\partial W}\right)^*$ we use backward mode: **`state_dw_b`**

$$\texttt{state\_dw\_b(}\underset{\Psi}{\texttt{psi}}, \overset{x}{\texttt{psib}}, \overset{c}{\texttt{c}}, \overset{W}{\texttt{w}}, \underset{\left(\frac{\partial \Psi}{\partial W}\right)^* x}{\texttt{wb}}\texttt{)}$$

  - if $A = \left(\frac{\partial \Psi}{\partial W}\right)$ we use direct mode: **`state_dw_d`**

$$\texttt{state\_dw\_d(}\underset{\Psi}{\texttt{psi}}, \underset{\left(\frac{\partial \Psi}{\partial W}\right) x}{\texttt{psid}}, \overset{c}{\texttt{c}}, \overset{W}{\texttt{w}}, \overset{x}{\texttt{wd}}\texttt{)}$$

First Derivatives: basic Tangent algorithm



Solve $\Psi(c, W) = 0$

Solve $\dfrac{\partial \Psi}{\partial W}\theta = -\dfrac{\partial \Psi}{\partial c}\dot{c}$ → **`solve_matrixfree_tan`**

Compute $\dfrac{\partial J}{\partial W}\theta$

First-order derivative: Tangent Mode

$$\texttt{state(}\,\texttt{psi},\overset{\downarrow}{\texttt{c}},\overset{\downarrow}{\texttt{w}}\,\texttt{)}$$

$$\texttt{state\_d(}\,\underset{\Psi}{\texttt{psi}},\underset{\dot{\Psi}}{\texttt{psid}},\overset{c}{\texttt{c}},\overset{\dot{c}}{\texttt{cd}},\overset{W}{\texttt{w}},\overset{\dot{W}}{\texttt{wd}}\,\texttt{)}$$

- Input variables: $\texttt{c} = c$, $\texttt{w} = W$, $\texttt{cd} = \dot{c}$, $\texttt{wd} = \dot{W}$
- Output variables: $\texttt{psi} = \Psi$, $\texttt{psid} = \dot{\Psi} = \left(\frac{\partial \Psi}{\partial c}\right)\dot{c} + \left(\frac{\partial \Psi}{\partial W}\right)\dot{W}$

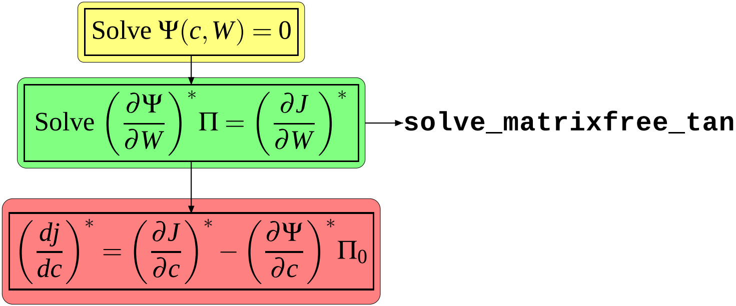
First-order derivative: Tangent Mode

$$\texttt{func(}\ \texttt{j}\,,\overset{\downarrow}{\texttt{c}}\,,\overset{\downarrow}{\texttt{w}}\ \texttt{)}$$

$$\texttt{func\_d(}\ \underset{J}{\texttt{j}}\,,\underset{\dot{J}}{\texttt{jd}}\,,\overset{c}{\texttt{c}}\,,\overset{\dot{c}}{\texttt{cd}}\,,\overset{W}{\texttt{w}}\,,\overset{\dot{W}}{\texttt{wd}}\ \texttt{)}$$

- Input variables: $\texttt{c} = c$, $\texttt{w} = W$, $\texttt{cd} = \dot{c}$, $\texttt{wd} = \dot{W}$
- Output variables: $\texttt{j} = J$, $\texttt{jd} = \dot{J} = \left(\frac{\partial J}{\partial c}\right)\dot{c} + \left(\frac{\partial J}{\partial W}\right)\dot{W}$

First Derivative: Reverse algorithm



Solve $\Psi(c, W) = 0$

$$\text{Solve } \left(\frac{\partial \Psi}{\partial W}\right)^* \Pi = \left(\frac{\partial J}{\partial W}\right)^*$$

`solve_matrixfree_tan`

$$\left(\frac{dj}{dc}\right)^* = \left(\frac{\partial J}{\partial c}\right)^* - \left(\frac{\partial \Psi}{\partial c}\right)^* \Pi_0$$

First-order derivative: Reverse Mode

$$\texttt{func(}\,\underset{\downarrow}{\texttt{j}}\,,\overset{\downarrow}{\texttt{c}}\,,\overset{\downarrow}{\texttt{w}}\,\texttt{)}$$

$$\texttt{func\_b(}\,\underset{J}{\texttt{j}}\,,\overset{\bar{J}}{\texttt{jb}}\,,\overset{c}{\texttt{c}}\,,\underset{\bar{c}}{\texttt{cb}}\,,\overset{W}{\texttt{w}}\,,\underset{\bar{W}}{\texttt{wb}}\,\texttt{)}$$

- Input variables: $\texttt{c} = c, \texttt{w} = W$, $\texttt{jb} = \bar{J}$
- Output variables: $\texttt{j} = J$,

$$\begin{cases} \texttt{cb} = \bar{c} = \left(\dfrac{\partial J}{\partial c}\right)^{*} \bar{J} \\ \texttt{wb} = \bar{W} = \left(\dfrac{\partial J}{\partial W}\right)^{*} \bar{J} \end{cases}$$

First-order derivative: Reverse Mode

$$\texttt{state(}\underset{\downarrow}{\texttt{psi}},\overset{\downarrow}{\texttt{c}},\overset{\downarrow}{\texttt{w}}\texttt{)}$$

$$\texttt{state\_b(}\underset{\Psi}{\texttt{psi}},\overset{\bar{\Psi}}{\texttt{psib}},\overset{c}{\texttt{c}},\underset{\bar{c}}{\texttt{cb}},\overset{W}{\texttt{w}},\underset{\bar{W}}{\texttt{wb}}\texttt{)}$$

- Input variables: $\texttt{c} = c, \texttt{w} = W, \texttt{psib} = \bar{\Psi}$
- Output variables: $\texttt{psi} = \Psi$,

$$\begin{cases} \texttt{cb} = \bar{c} = \left(\dfrac{\partial \Psi}{\partial c}\right)^{*} \bar{\Psi} \\ \texttt{wb} = \bar{W} = \left(\dfrac{\partial \Psi}{\partial W}\right)^{*} \bar{\Psi} \end{cases}$$

Second derivative: Tangent-on-Tangent approach

$$\frac{d^2j}{dc_idc_k} = -\Pi_0^* D^2_{i,k}\Psi + D^2_{i,k}J$$

with Π_0 solution of the adjoint system

$$\left(\frac{\partial \Psi}{\partial W}\right)^* \Pi = \left(\frac{\partial J}{\partial W}\right)^*$$

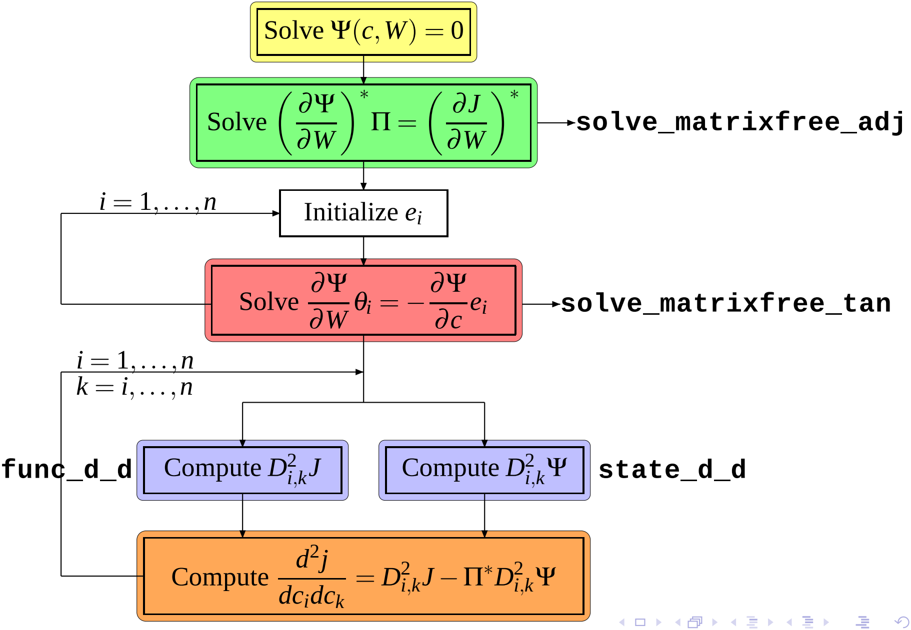
and

$$\begin{cases} D^2_{i,k}J = \frac{\partial}{\partial c}\left(\frac{\partial J}{\partial c}e_i\right)e_k + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial c}e_i\right)\theta_k + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial c}e_k\right)\theta_i + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\theta_i\right)\theta_k \\ D^2_{i,k}\Psi = \frac{\partial}{\partial c}\left(\frac{\partial \Psi}{\partial c}e_i\right)e_k + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial c}e_i\right)\theta_k + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial c}e_k\right)\theta_i + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial W}\theta_i\right)\theta_k \end{cases}$$

where $\theta_k = \frac{dW}{dc_k}$ is the solution of the system

$$\frac{\partial \Psi}{\partial W}\frac{dW}{dc_k} = -\frac{\partial \Psi}{\partial c}e_k$$

Second Derivatives: basic ToT algorithm



Solve $\Psi(c, W) = 0$

Solve $\left(\frac{\partial \Psi}{\partial W}\right)^{*} \Pi = \left(\frac{\partial J}{\partial W}\right)^{*}$ → **`solve_matrixfree_adj`**

$i = 1, \ldots, n$

Initialize e_i

Solve $\frac{\partial \Psi}{\partial W} \theta_i = -\frac{\partial \Psi}{\partial c} e_i$ → **`solve_matrixfree_tan`**

$i = 1, \ldots, n$

$k = i, \ldots, n$

`func_d_d` Compute $D^2_{i,k} J$

Compute $D^2_{i,k} \Psi$ **`state_d_d`**

Compute $\frac{d^2 j}{dc_i dc_k} = D^2_{i,k} J - \Pi^* D^2_{i,k} \Psi$

Second derivative: Tangent-on-Tangent Mode

$$\texttt{state\_d(}\,\texttt{psi},\texttt{psid},\overset{\downarrow}{\texttt{c}},\texttt{cd},\overset{\downarrow}{\texttt{w}},\texttt{wd}\,\texttt{)}$$

$$\downarrow$$

$$\texttt{state\_d\_d(}\,\underset{\Psi}{\texttt{psi}},\underset{\dot{\Psi}}{\texttt{psid}},\underset{\ddot{\Psi}}{\texttt{psidd}},\overset{c}{\texttt{c}},\overset{e_k}{\texttt{cd0}},\overset{e_i}{\texttt{cd}},\overset{W}{\texttt{w}},\overset{\theta_k}{\texttt{wd0}},\overset{\theta_i}{\texttt{wd}}\,\texttt{)}$$

- Input variables: **`c`** $= c$, **`cd`** $= e_i$, **`w`** $= W$, **`wd`** $= \theta_i$, **`cd0`** $= e_k$, **`wd0`** $= \theta_k$
- Output variables: **`psi`** $= \Psi$, **`psid`** $= \dot{\Psi} = \left(\frac{\partial \Psi}{\partial c}\right) e_i + \left(\frac{\partial \Psi}{\partial W}\right) \theta_i$

$$\texttt{psidd} = \frac{\partial}{\partial c}\left(\frac{\partial \Psi}{\partial c} e_i\right) e_k + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial c} e_i\right) \theta_k + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial c} e_k\right) \theta_i + \frac{\partial}{\partial W}\left(\frac{\partial \Psi}{\partial W} \theta_i\right) \theta_k$$
$$= D^2_{i,k}\Psi$$

Second derivative: Tangent-on-Tangent Mode

$$\texttt{func\_d(}\,\texttt{j}\,,\underset{\downarrow}{\texttt{jd}}\,,\overset{\downarrow}{\texttt{c}}\,,\texttt{cd}\,,\overset{\downarrow}{\texttt{w}}\,,\texttt{wd}\,\texttt{)}$$

$$\texttt{func\_d\_d(}\,\underset{J}{\texttt{j}}\,,\underset{\dot{J}}{\texttt{jd}}\,,\underset{\ddot{J}}{\texttt{jdd}}\,,\overset{c}{\texttt{c}}\,,\overset{e_k}{\texttt{cd0}}\,,\overset{e_i}{\texttt{cd}}\,,\overset{W}{\texttt{w}}\,,\overset{\theta_k}{\texttt{wd0}}\,,\overset{\theta_i}{\texttt{wd}}\,\texttt{)}$$

- Input variables: $\texttt{c} = c$, $\texttt{cd} = e_i$, $\texttt{w} = W$, $\texttt{wd} = \theta_i$, $\texttt{cd0} = e_k$, $\texttt{wd0} = \theta_k$
- Output variables: $\texttt{j} = J$, $\texttt{jd} = \dot{J} = \left(\frac{\partial J}{\partial c}\right) e_i + \left(\frac{\partial J}{\partial W}\right)\theta_i$

$$\texttt{jdd} = \frac{\partial}{\partial c}\left(\frac{\partial J}{\partial c} e_i\right) e_k + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial c} e_i\right)\theta_k + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial c} e_k\right)\theta_i + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\theta_i\right)\theta_k$$
$$= D^2_{i,k} J$$

Second derivative: Tangent-on-Reverse

$$\frac{\partial}{\partial c_i}\left(\frac{\partial j}{\partial c}\right)^* = \left(\frac{\partial^2 j}{\partial c^2}\right)e_i = \frac{\partial}{\partial c}\left(\frac{\partial J}{\partial c}\right)^* e_i + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial c}\right)^* \theta_i + \\ - \frac{\partial}{\partial c}\left[\left(\frac{\partial \Psi}{\partial c}\right)^* \Pi_0\right] e_i - \frac{\partial}{\partial W}\left[\left(\frac{\partial \Psi}{\partial c}\right)^* \Pi_0\right] \theta_i - \left(\frac{\partial \Psi}{\partial c}\right)^* \lambda_i$$

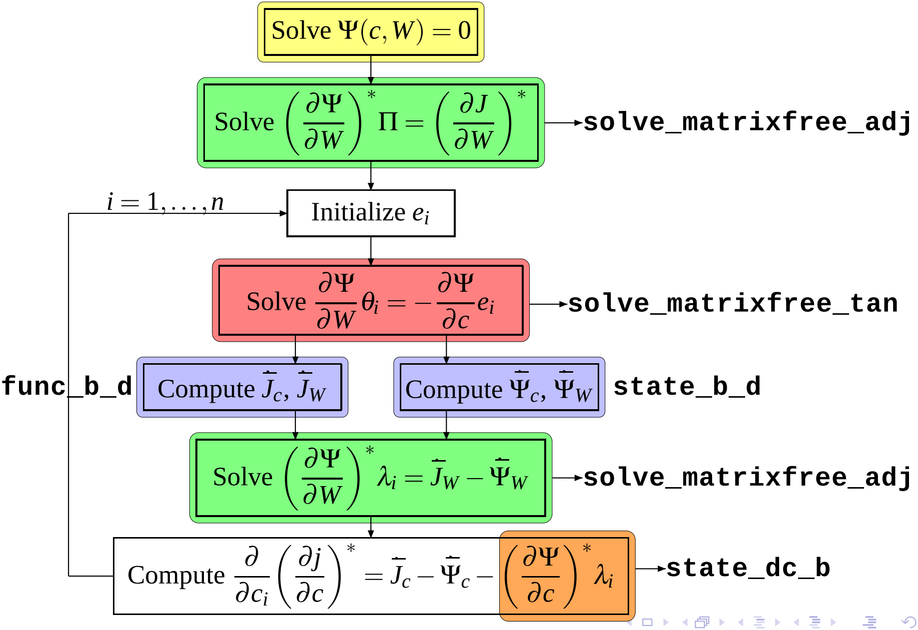
with Π_0 solution of the adjoint system

$$\left(\frac{\partial \Psi}{\partial W}\right)^* \Pi = \left(\frac{\partial J}{\partial W}\right)^*$$

θ_i and λ_i solution of the systems

$$\begin{cases} \dfrac{\partial \Psi}{\partial W}\theta_i = -\dfrac{\partial \Psi}{\partial c}e_i \\ \left(\dfrac{\partial \Psi}{\partial W}\right)^* \lambda_i = \dfrac{\partial}{\partial c}\left(\dfrac{\partial J}{\partial W}\right)^* e_i + \dfrac{\partial}{\partial W}\left(\dfrac{\partial J}{\partial W}\right)^* \theta_i + \\ \qquad - \dfrac{\partial}{\partial c}\left[\left(\dfrac{\partial \Psi}{\partial W}\right)^* \Pi_0\right] e_i - \dfrac{\partial}{\partial W}\left[\left(\dfrac{\partial \Psi}{\partial W}\right)^* \Pi_0\right] \theta_i \end{cases}$$

Second Derivatives: basic ToR algorithm



Solve $\Psi(c,W)=0$

Solve $\left(\frac{\partial \Psi}{\partial W}\right)^{*}\Pi=\left(\frac{\partial J}{\partial W}\right)^{*}$ → `solve_matrixfree_adj`

$i=1,\ldots,n$

Initialize e_i

Solve $\frac{\partial \Psi}{\partial W}\theta_i=-\frac{\partial \Psi}{\partial c}e_i$ → `solve_matrixfree_tan`

`func_b_d` Compute $\bar{J}_c,\bar{J}_W$

Compute $\bar{\Psi}_c,\bar{\Psi}_W$ `state_b_d`

Solve $\left(\frac{\partial \Psi}{\partial W}\right)^{*}\lambda_i=\bar{J}_W-\bar{\Psi}_W$ → `solve_matrixfree_adj`

Compute $\frac{\partial}{\partial c_i}\left(\frac{\partial j}{\partial c}\right)^{*}=\bar{J}_c-\bar{\Psi}_c-\left(\frac{\partial \Psi}{\partial c}\right)^{*}\lambda_i$ → `state_dc_b`

Second derivative: Tangent-on-Reverse Mode

$$\texttt{state\_b(}\ \texttt{psi},\texttt{psib},\overset{\downarrow}{\texttt{c}},\underset{\downarrow}{\texttt{cb}},\overset{\downarrow}{\texttt{w}},\underset{\downarrow}{\texttt{wb}}\ \texttt{)}$$

$$\texttt{state\_b\_d(}\ \underset{\Psi}{\texttt{psi}},\overset{\Pi_0}{\texttt{psib}},\overset{c}{\texttt{c}},\overset{e_i}{\texttt{cd}},\underset{\left(\frac{\partial\Psi}{\partial c}\right)^{*}\Pi_0}{\texttt{cb}},\underset{\dot{\bar{c}}_\Psi}{\texttt{cbd}},\overset{W}{\texttt{w}},\overset{\theta_i}{\texttt{wd}},\underset{\left(\frac{\partial\Psi}{\partial W}\right)^{*}\Pi_0}{\texttt{wb}},\underset{\dot{\bar{W}}_\Psi}{\texttt{wbd}}\ \texttt{)}$$

- Input variables: **`psib`** $= \Pi_0$, **`c`** $= c$, **`w`** $= W$, **`cd`** $= e_i$, **`wd`** $= \theta_i$
- Output variables: **`psi`** $= \Psi$, **`cb`** $= \left(\frac{\partial\Psi}{\partial c}\right)^{*}\Pi_0$, **`wb`** $= \left(\frac{\partial\Psi}{\partial W}\right)^{*}\Pi_0$,

  `cbd` $= \dot{\bar{c}}_\Psi$, **`wbd`** $= \dot{\bar{W}}_\Psi$

$$\begin{cases} \dot{\bar{c}}_\Psi = \dfrac{\partial}{\partial c}\left[\left(\dfrac{\partial\Psi}{\partial c}\right)^{*}\Pi_0\right]e_i + \dfrac{\partial}{\partial W}\left[\left(\dfrac{\partial\Psi}{\partial c}\right)^{*}\Pi_0\right]\theta_i \\ \dot{\bar{W}}_\Psi = \dfrac{\partial}{\partial c}\left[\left(\dfrac{\partial\Psi}{\partial W}\right)^{*}\Pi_0\right]e_i + \dfrac{\partial}{\partial W}\left[\left(\dfrac{\partial\Psi}{\partial W}\right)^{*}\Pi_0\right]\theta_i \end{cases}$$

Second derivative: Tangent-on-Reverse Mode

$$\texttt{func\_b(}\,\texttt{j},\texttt{jb},\overset{\downarrow}{\texttt{c}},\underset{\downarrow}{\texttt{cb}},\overset{\downarrow}{\texttt{w}},\underset{\downarrow}{\texttt{wb}}\,\texttt{)}$$

$$\texttt{func\_b\_d(}\,\underset{J}{\texttt{j}},\overset{1.0}{\texttt{jb}},\overset{c}{\texttt{c}},\overset{e_i}{\color{red}\texttt{cd}},\underset{\left(\frac{\partial J}{\partial c}\right)^*}{\texttt{cb}},\underset{\dot{\bar{c}}_J}{\color{red}\texttt{cbd}},\overset{W}{\texttt{w}},\overset{\theta_i}{\color{red}\texttt{wd}},\underset{\left(\frac{\partial J}{\partial W}\right)^*}{\texttt{wb}},\underset{\dot{\bar{W}}_J}{\color{red}\texttt{wbd}}\,\texttt{)}$$

- Input variables: $\texttt{jb} = 1.0$, $\texttt{c} = c$, $\texttt{w} = W$, $\texttt{cd} = e_i$, $\texttt{wd} = \theta_i$
- Output variables: $\texttt{j} = j$, $\texttt{cb} = \left(\frac{\partial J}{\partial c}\right)^*$, $\texttt{wb} = \left(\frac{\partial J}{\partial W}\right)^*$, $\texttt{cbd} = \dot{\bar{c}}_J$, $\texttt{wbd} = \dot{\bar{W}}_J$

$$\begin{cases} \dot{\bar{c}}_J = \frac{\partial}{\partial c}\left(\frac{\partial J}{\partial c}\right)^* e_i + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial c}\right)^* \theta_i \\ \dot{\bar{W}}_J = \frac{\partial}{\partial c}\left(\frac{\partial J}{\partial W}\right)^* e_i + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\right)^* \theta_i \end{cases}$$

Full Hessian: ToT vs. ToR

Common part

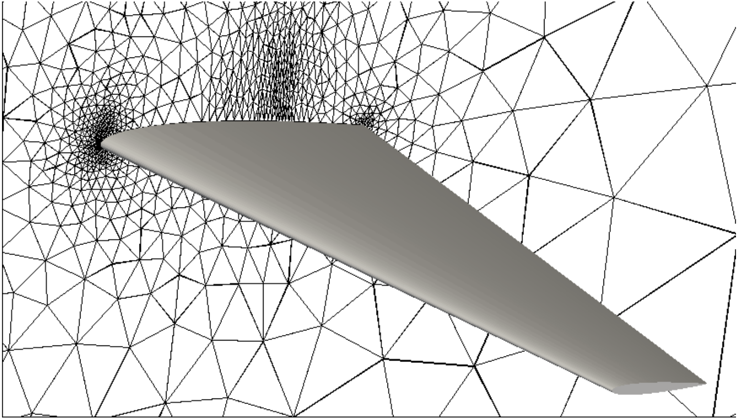
- Adjoint state $\left(\frac{\partial \Psi}{\partial W}\right)^{*} \Pi = \left(\frac{\partial J}{\partial W}\right)^{*}$
- n linear system to solve: $\frac{\partial \Psi}{\partial W} \frac{dW}{dc_i} = -\frac{\partial \Psi}{\partial c} e_i$

Specific part

- We assume unitary cost for a single residual evaluation $\Psi(c, W)$
- ToT: computation of $D^2_{ik}\Psi$ and $D^2_{ik}J \implies \frac{n(n+1)}{2}\alpha_T^2 \qquad (1 < \alpha_T < 4)$
- ToR: n (adjoint) linear system to solve: $\left(\frac{\partial \Psi}{\partial W}\right)^{*} \lambda_i = \bar{J}_W - \bar{\Psi}_W$
  $\implies n(n_{\text{iter}} + \alpha_T)\alpha_R \qquad (1 < \alpha_R \lesssim 5)$

$$\boxed{\text{Use ToT when } n < \frac{2\alpha_R(n_{\text{iter}} + \alpha_T)}{\alpha_T^2}}$$

Application to a 3D-Euler software

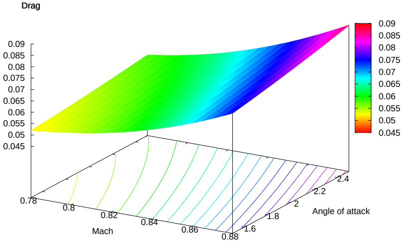


Numerics:

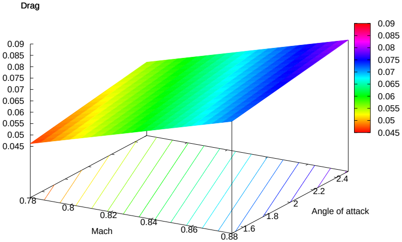
- Upwind vertex-centered finite-volume
- Implicit pseudo-time advancing with first-order Jacobian
- F77

Building the reduced quadratic model

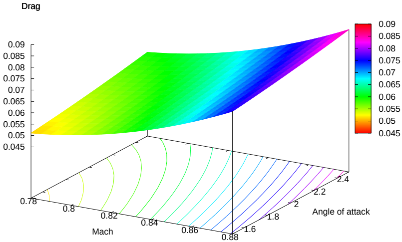
Nonlinear simulations



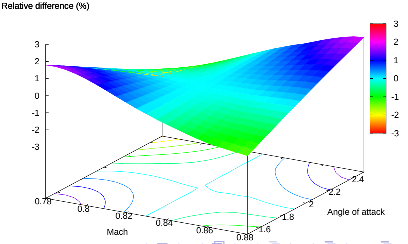
Taylor 1st order (α=2.0, M=0.83)



Taylor 2nd order (α=2.0, M=0.83)



Relative Difference: Nonlinear vs. Taylor 2nd order



Concluding remarks

Synthesis: non-black-box AD application

- Optimum use of modern iterative solvers.
- Avoid managing useless storage or recomputing in backward mode.
- First and second derivative architecture well identified.
- Validated by experiments.

Future works

- TAPENADE further improvements.
- Functional involved in robust optimisation and their gradients.
- Pathologies in relation with discontinuities.
- Very large instationary state systems.