Acceleration of Turbomachinery steady CFD Simulations on GPU

Mohamed H. Aissa\textsuperscript{1,2}
Dr. Tom Verstraete\textsuperscript{1}
Prof. Cornelis Vuik\textsuperscript{2}

\textsuperscript{1} Turbomachinery Department, Von Karman Institute, Belgium
\textsuperscript{2} Delft Institute of Applied Mathematics, TU Delft, the Netherlands
Topic of Interest:
Reduce Fuel Consumption and CO₂ Emission
Turbomachinery is about Performance and Efficiency
Axial Jet Engine

Source: Wikipedia.org
Content

• Multidisciplinary Optimization
• CFD simulations on GPU
  • Literature review
  • Implicit RANS Implementation
  • Benchmark
• Optimization Case
Optimization algorithm

Derivative-based optimization

- fast convergence but ..
- derivative evaluation could be complicated and problem specific (adjoint, automatic differentiation)

Derivative free methods: e.g. Population based

- Simplicity
- Black box approach of the evaluation but ..
- Large number of evaluations

\[
\min f(\mathbf{x}) \\
\text{subject to} \\
g(\mathbf{x}) \leq 0
\]
## Derivative-based optimization

- fast convergence but ..
- derivative evaluation could be complicated and problem specific (adjoint, automatic differentiation)

## Derivative free methods: e.g. Population based

- Simplicity
- Black box approach of the evaluation but ..
- Large number of evaluations

\[
\min f(x) \\
subject \ to \\
g(x) \leq 0
\]
CFD: Core of the Optimization

CFD much slower than CSM
Need for acceleration -> GPU

CADO: the VKI in-house optimizer
Steady CFD Simulations

- Simulation with a unique solution for given boundary Conditions.
- A start solution is advanced iteratively in time until convergence.
Steady CFD Simulations

- Simulation with a unique solution for given boundary Conditions.
- A start solution is advanced iteratively in time until convergence.
Numerical Scheme:

Explicit Time Stepping ($\beta=0$):

$$\left(\Omega \bar{M}\right)_I \frac{\Delta \bar{W}_I^n}{\Delta t_I} = -\beta \bar{R}_I^{(n+1)} - (1 - \beta) \bar{R}_I^n$$

$$\Delta W^n = -\frac{\Delta t}{\Omega} R^n$$

Implicit Time Stepping ($\beta=1$):

$$\bar{R}_I^{n+1} \approx \bar{R}_I^n + \left(\frac{\delta R}{\delta \bar{W}}\right)_I \Delta \bar{W}^n$$

$$\left[\frac{(\Omega I)}{\Delta t} + \left(\frac{\delta R}{\delta \bar{W}}\right)\right] \Delta W^n = -R^n$$

Implicit Time Stepping is more Stable but...

\[
\left[ \frac{(\Omega I)}{\Delta t} + \left( \frac{\delta R}{\delta W} \right) \right] \Delta W^n = -R^n
\]

\[
A \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = b
\]

\[
k(A) = ||A|| \times ||A^{-1}||
\]

\[
M^{-1} Ax = M^{-1} b
\]
Literature Review

• What to Port
  • only linear solver when it is dominant
  • both assembly and solve is optimal (no communication)

• Linear solver
  • Library: code maturity but restrictive
    (petsc-dev, Paralution, AmgX, ViennaCL ...)
  • Own code: flexibility

• Storage format
  • Standard (CSR, DIA ...)
  • New (hybrid)
CFD Solver (Standard)

\[
\begin{bmatrix}
\frac{(\Omega I)}{\Delta t} + \left( \frac{\delta R}{\delta W} \right)
\end{bmatrix}
\Delta W^n = -R^n
\]

Implicit Runge-Kutta scheme

Xu et Al. JCP 2015
Implicit Runge-Kutta scheme

\[
\begin{align*}
A^{(0)}[W^{(1)} - W^{(0)}] &= W^n - \alpha_1 R(W^{(0)}) \\
A^{(0)}[W^{(2)} - W^{(0)}] &= -\alpha_2 R(W^{(1)}) \\
& \vdots \\
A^{(0)}[W^{(m)} - W^{(0)}] &= -\alpha_m R(W^{(m-1)}) \\
W^{n+1} &= W^{(0)} + [W^{(m)} - W^{(0)}]
\end{align*}
\]
Implicit Runge-Kutta scheme

\[
A^{(0)}[W^{(1)} - W^{(0)}] = W^{n} - \alpha_{1} R(W^{(0)}) \\
A^{(0)}[W^{(2)} - W^{(0)}] = -\alpha_{2} R(W^{(1)}) \\
\vdots \\
A^{(0)}[W^{(m)} - W^{(0)}] = -\alpha_{m} R(W^{(m-1)}) \\
W^{n+1} = W^{(0)} + [W^{(m)} - W^{(0)}]
\]
CFD Solver (On-demand Factorization)

\[ r = Ax - b \]

Stop condition relative, absolute or a combination

\[ \frac{||r_k||}{||b||} < \tau_3 \quad ||r_k|| < \tau_2 \]

![Graph showing iterations per call vs. time with two lines representing different conditions.](image)
CFD Solver (On-demand Factorization)

\[ r = Ax - b \]

Stop condition relative, absolute or a combination

\[
\frac{\|r_k\|}{\|b\|} < \tau_3 \quad \frac{\|r_k\|}{\|r_k\|} < \tau_2
\]

MAX_ITR = \( \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} N_{ITR_i} \)
Benchmark: Flow around LS89
2-Stages Runge-Kutta
Assembly Acceleration

**Speedups on Coarse Mesh**
- 2xCores: Standard
- 3xCores: Standard
- 4xCores: Standard
- 4xCores: On-demand

**Speedups on Fine Mesh**
- 2xCores: Standard
- 3xCores: Standard
- 4xCores: Standard
- 4xCores: On-demand

Assembly speedup: 7.8x
Linear solve speedup: 12.2x
Global speedup: 10%

CPU: 70%
GPU: 90%

**Notes:**
- GPU assembly acceleration results in significant speedup compared to CPU.
- On-demand scheduling provides further speedup benefit.
- Linear solve and global speedup are also improved with GPU acceleration.
Linear Solver Acceleration

Speedups on Coarse Mesh

- 2xCores
- 3xCores
- 4xCores

Assembly speedup: x 0.7
Linear solve speedup: x 1.2
Global speedup: x 2

Speedups on Fine Mesh

- 2xCores
- 3xCores
- 4xCores

Assembly speedup: x 2
Linear solve speedup: x 1.8
Global speedup: x 5.7

CPU vs. GPU performance for different core counts and demand modes.
Global Acceleration

Speedups on Coarse Mesh

- 2xCores: x 2.0
- 3xCores CPU: x 3.2
- 4xCores: x 4.8

Speedups on Fine Mesh

- 2xCores: x 9.6
- 3xCores CPU: x 4.8
- 4xCores: x 9.6

Assembly speedup
Linear solve speedup
Global speedup

Suggestion for better Performance assessment are very welcome!
Increase of the Speedup for higher Numbers of Runge-Kutta Stages on Fine Mesh
Content

- Multidisciplinary Optimization
- CFD simulations on GPU
  - Literature review
  - Implicit RANS Implementation
  - Benchmark
- Optimization Case
Test Case 3: TU Berlin TurboLab Stator Optimization requirements

Objectives:
- Decrease outflow axial deviation
- Decrease total pressure loss

Considering 3 operating points
TurboLab
Manufacturing Constraints

- $N_{\text{blades}} = 15$
- Chord length fixed
- Casing fixture

![Diagram](image-url)
TurboLab: Boundary conditions and summary

Inlet $P_0$: 102713.0 Pa
Inlet $T_0$: 294.314 K

Objectives:
- Decrease outflow axial deviation
- Decrease total pressure loss

Considering 3 operating points

9 kg/s ± 0.1
Massflow imposed
$P_2$ adapted

Inlet whirl angle: 42°
Inlet pitch angle: 0°
Parametrization
21 Design variables
Turbolab Parameterization
Optimization Results

\[ \int_{\text{casing}}^{\text{hub}} \alpha_{\text{whirl}}^2 \]

\[ \text{Flow deviation [deg]} \]

\[ \text{Total pressure loss} \]

\[ \text{Loss}_{P_0} = \frac{p_{01} - p_{02}}{p_{01} - p_1} \]
Optimized Blade
Baseline Vs Optimized
Baseline Vs Optimized

![Graph showing baseline vs optimized comparison with pressure contour plot.](image-url)
Isentropic Mach Number at mid-span
Conclusion

• Optimization

• GPU Solver with implicit time stepping

• *On-demand* (incomplete) Factorization

• 10x speedup

• Aerodynamic shape optimization
Future Work

Benchmark Case: Transonic Turbine Stator T106c
Thanks for your attention

Mohamed Hassanine Aissa
Turbomachinery & Propulsion Department Department
72, chaussee de Waterloo
B1640 - Rhode Saint Genese - Belgium
Email: aissa@vki.ac.be

acknowledgements:

PARALUTION
Support

Hardware