Hybrid Parallel Simulation of Helicopter Rotor Dynamics

Melven Röhrig-Zöllner[1], Achim Basermann[1], Johannes Hofmann[2], Margrit Klitz[1], and Lukas Schmierer[1]

[1]DLR, Simulation and Software Technology
[2]DLR, Institute of Flight Systems

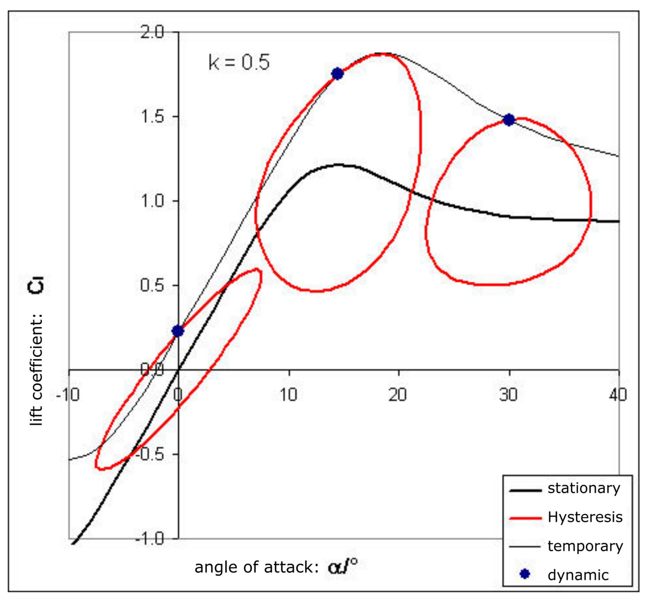
Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Coupled rotor simulation

Rotor simulation: the S4 code

- Simulates rotor blade movement, angles and forces
- High resolution
  default: 20 blade elements, 2° steps, or (much) finer
- Dynamic-response problem:
  - Sectional airloads: semi-empirical unsteady analytical model
    includes compressibility, yawed flow and dynamic stall
  - Blade dynamics: modal synthesis approach

→ fast



Rotor discretization

- Discretized **rotor positions** in space $\mathbf{x}_{rotor}(t) \in \mathbb{R}^{3n}$
- Change according to

$$\begin{aligned}\dot{\mathbf{x}}_{rotor}(t) &= \mathbf{F}_{rotor}\left(\mathbf{x}_{rotor}(t), \boldsymbol{\Gamma}_{rotor}(t)\right),\\ \boldsymbol{\Gamma}_{rotor}(t) &= \mathbf{f}_{circulation}\left(\mathbf{x}_{rotor}(t), \mathbf{v}_{inflow}(t)\right)\end{aligned} \qquad (1)$$

  with circulation on the blades $\boldsymbol{\Gamma}_{rotor}$ and inflow velocity $\mathbf{v}_{inflow}$

Wake simulation: the Freewake code

- Simulates the flow around a helicopter's rotor
- Vortex-lattice method:
  - Wake structure discretized by a set of elements
  - Circulation on the blades creates vortices
  - Calculates wake-lattice perturbation

→ explicit vortex tracing without numerical dissipation
→ very fast compared to CFD

Wake discretization

- **Rotor wake** modeled by Lagrangian markers in space $\mathbf{x}_{wake}(t) \in \mathbb{R}^{3m}$ with $m \gg n$
- Change according to

$$\begin{aligned}\dot{\mathbf{x}}_{wake}(t) &= \mathbf{F}_{wake}\left(\mathbf{x}_{wake}(t), \mathbf{x}_{rotor}, \boldsymbol{\Gamma}_{rotor}\right),\\ \mathbf{v}_{inflow}(t) &= \mathbf{f}_{inflow}\left(\mathbf{x}_{wake}(t), \boldsymbol{\Gamma}_{rotor}\right)\end{aligned} \qquad (2)$$

- Markers depend on the history of $\mathbf{x}_{rotor}$ and the circulation $\boldsymbol{\Gamma}_{rotor}$ leading to the integral equation

$$\mathbf{v}_{inflow}(t) = \int_0^t \widehat{\mathbf{F}}_{wake}\left(t, \mathbf{x}_{rotor}(\hat{t}), \boldsymbol{\Gamma}_{rotor}(\hat{t})\right)\, d\hat{t} \qquad (3)$$

Time integration schemes

Rotor mechanics: Runge-Kutta method

- *Runge-Kutta-Gill-4* (variant of classical RK4)
- Uses substep-predictions at $t + \frac{1}{2}$.
- Not all parts of the model $\mathbf{F}_{rotor}$ are updated in each substep:

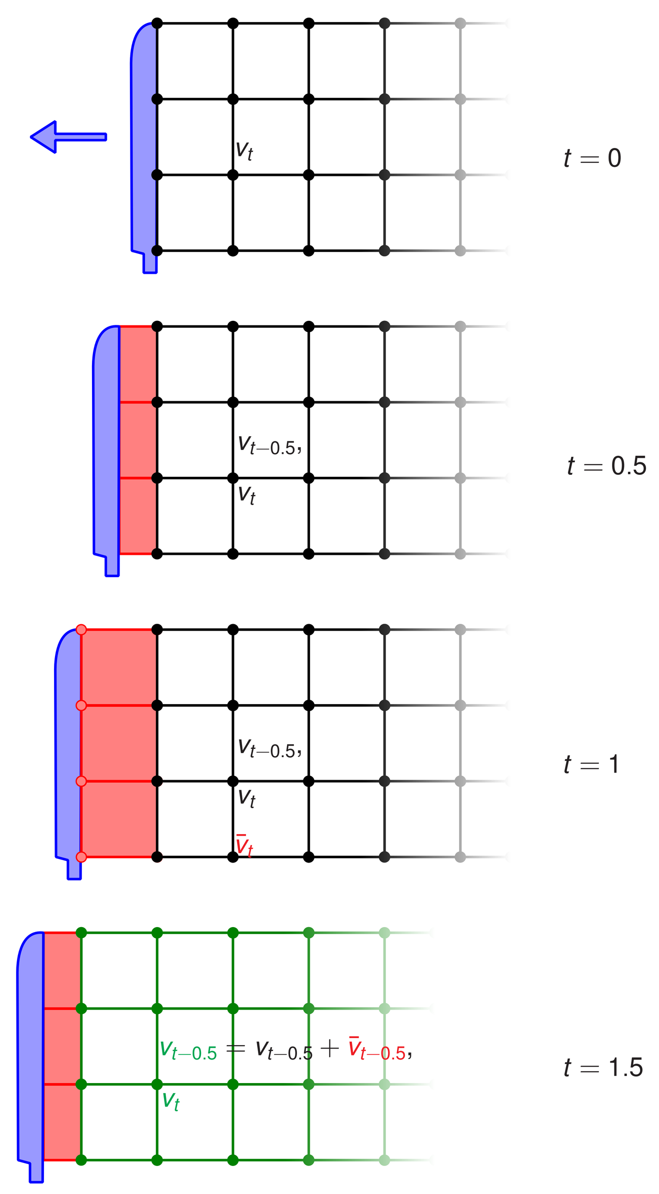
$$\begin{aligned}\mathbf{x}_{rotor}(t+1) &= \mathbf{x}_{rotor}(t) + b_1\mathbf{k}_1 + b_2\mathbf{k}_2 + b_3\mathbf{k}_3 + b_4\mathbf{k}_4,\\ \mathbf{k}_1 &= \widetilde{\mathbf{F}}_{rotor}(t, \dots)\\ \mathbf{k}_2 &= \mathbf{F}_{rotor}(t + \tfrac{1}{2}, \dots)\\ \mathbf{k}_3 &= \widetilde{\mathbf{F}}_{rotor}(t + \tfrac{1}{2}, \dots)\\ \mathbf{k}_4 &= \mathbf{F}_{rotor}(t, \dots)\end{aligned}$$

  (In steps with $\widetilde{\mathbf{F}}_{rotor}$, right hand side is updated approximately reusing results from the previous substep.)

Wake simulation: multistep method

- Variable step size *Adams-Bashforth-2* (two-step method)
- Requires induced velocity from previous timestep:
  - No data is available for new grid points.
    (directly behind the rotor blades)

  → Use explicit Euler with smaller timesteps there.
- Predictions (second order) for $\mathbf{v}_{inflow}$.
- Complicates calculating/tracking velocities in the wake:
  See figure on the right:
  $t = 0.0$: No old data available, start with explicit Euler.
  $t = 0.5$: Use the two-step method.
  $t = 1.0$: Use the two-step method for **black dots**, explicit Euler for **red dots**.
  $t = 1.5$: Combine **old data**, and use the two-step method.
  (Scheme more complex with variable step size and smaller explicit Euler steps!)



Rotor wake dynamics

Vorticity transport equation

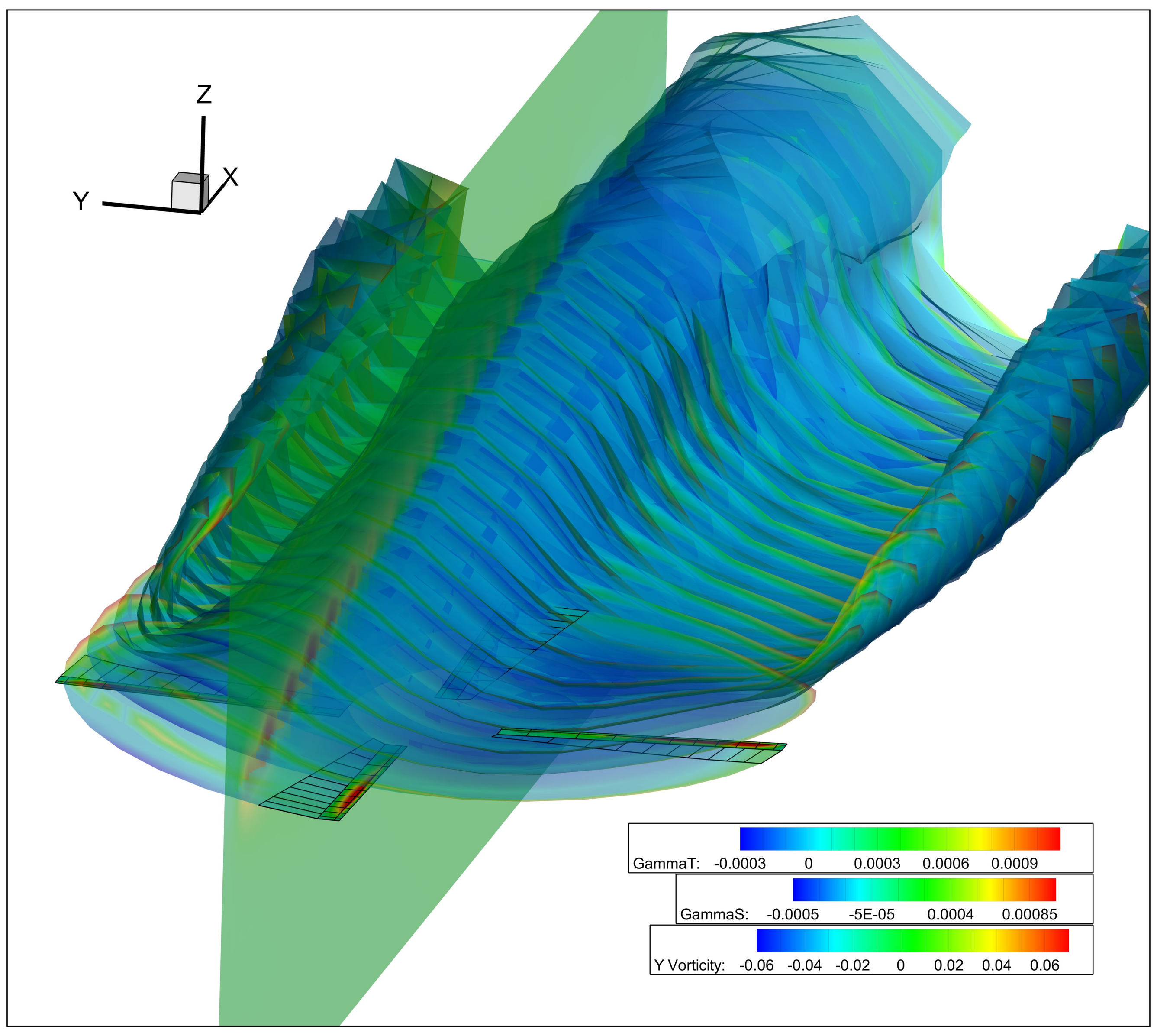
Vortices move with the flow except for stretching/tilting and dissipation:

$$\frac{D\vec{\omega}}{Dt} = \left(\vec{\omega}\cdot\vec{\nabla}\right)\vec{v} + \nu\vec{\nabla}^2\vec{\omega}$$

Stretching/tilting is implicitly respected in a moving wake grid.

Model assumptions

- Vorticity is concentrated in a thin layer.
- Coarse discretization requires subgrid modeling:
  - (Tip) vortex roll-up
    (vorticity concentrates radially at vortex' centers)
  - Vortex core radii (vorticity smoothly distributed)



Coupling schemes

Weak coupling

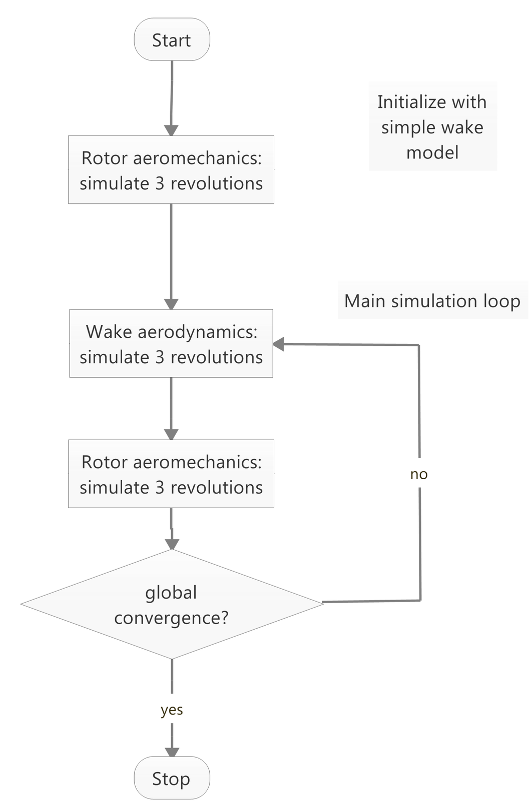
- Only for quasi-steady operational conditions
- Rotor and wake are updated in a loop.
  See figure on the right.
- Needs another, simple wake model for the first step.
- Coupling data captured over one revolution
  $\mathbf{v}_{inflow}$, resp. $\mathbf{x}_{rotor}$, $\boldsymbol{\Gamma}_{rotor}$

→ Apply low-pass filter to remove irregularities
  Frequencies not resolvable by discretization

Strong coupling

- Wake evaluated inside Runge-Kutta-scheme.
- Wake still uses its own time-stepping.
  (RK4 for the wake too costly!)
- Predict $\mathbf{v}_{inflow}(t + \Delta t)$.
- Circular dependence of $\mathbf{v}_{inflow}(t)$ and $\boldsymbol{\Gamma}_{rotor}(t)$

→ Idea: use small fixed-point iteration (untested)



Parallelization

Vortex methods

- Most expensive operation: Calculating induced velocities in the wake (Biot-Savart):

$$\vec{v}(\vec{x}_i) = \frac{1}{4\pi}\sum_j^{n_{gridcells}} \int_{Cell_j} \frac{\vec{\omega}\times(\vec{x}_i - \vec{y})}{\left(\|\vec{x}_i - \vec{y}\|^2 + r_c^2\right)^{\frac{3}{2}}}\, d\vec{y}$$

- Required at every grid point: naive runtime $O(n^2)$.
- Non-trivial integration formula from subgrid models.

→ Fast multipole methods (FMM) not easily applicable!
- Idea: Approximations of different accuracy depending on the distance $\|\vec{x} - \vec{y}\|$.
- Performance compute-bound, small data.

OpenMP and MPI parallelization for multicore CPUs

- Well encapsulated, all data replicated on all processes.
- Parallelization of $\vec{v}(\vec{x}_i)$ over grid points i.
- Some intermediate data recalculated on every thread.
- Dynamic load balancing over processes
  with static OpenMP scheduling

OpenACC parallelization for GPUs

- Dedicated code for OpenACC for specific data dimensions.
  Only calculations for the complete grid executed on GPUs.
- Testable on CPUs.
- High level code, but still more complex than CPU code.
- No vector-reductions in OpenACC

→ Bad loop nest ordering
- Hybrid calculation with GPUs+CPUs possible

→ Not useful, yet: data dimensions too small!

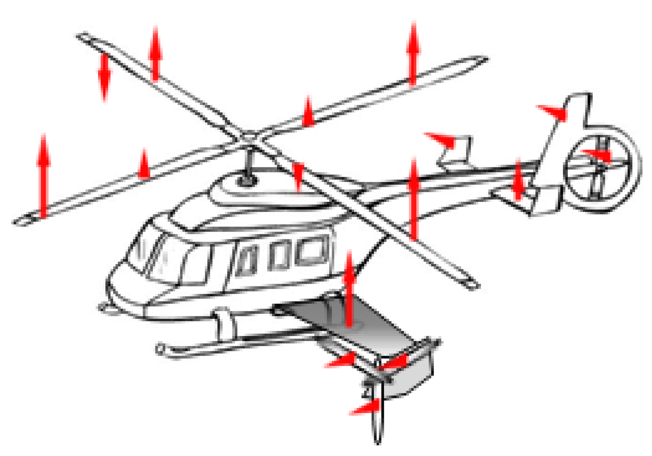
Table 1 : Timings for one revolution on a workstation with 2x12 core Intel Xeon E-2670 and NVidia Tesla K40m

Timestepping	Parallelization	Time [s]
AB-2	Single core	28.5
	OpenMP (12 cores)	9.2
	OpenMP (24 cores)	8.9
	MPI (24 proc.)	8.0
	MPI+OpenMP (2x12)	8.4
	OpenACC (K40m)	15.8
Expl. Euler	OpenMP (12 cores)	26.5

Plans for the future

Versatile Aeromechanics Simulation Tool (VAST)

- Simulation of freely flying helicopters
- Independence from commercial/proprietary codes.
- Modern, adaptive and future-proof high performance framework without simplifications.



Planned features

- Various rotor configurations
  multiple rotors, co-axial, tilt, ...
- Account for fuselage aerodynamics
  and fuselage-rotor interference
- Arbitrary arrangement of the rotors
  (even non-symmetric)
- Model for the pilot
- Support for wind turbines