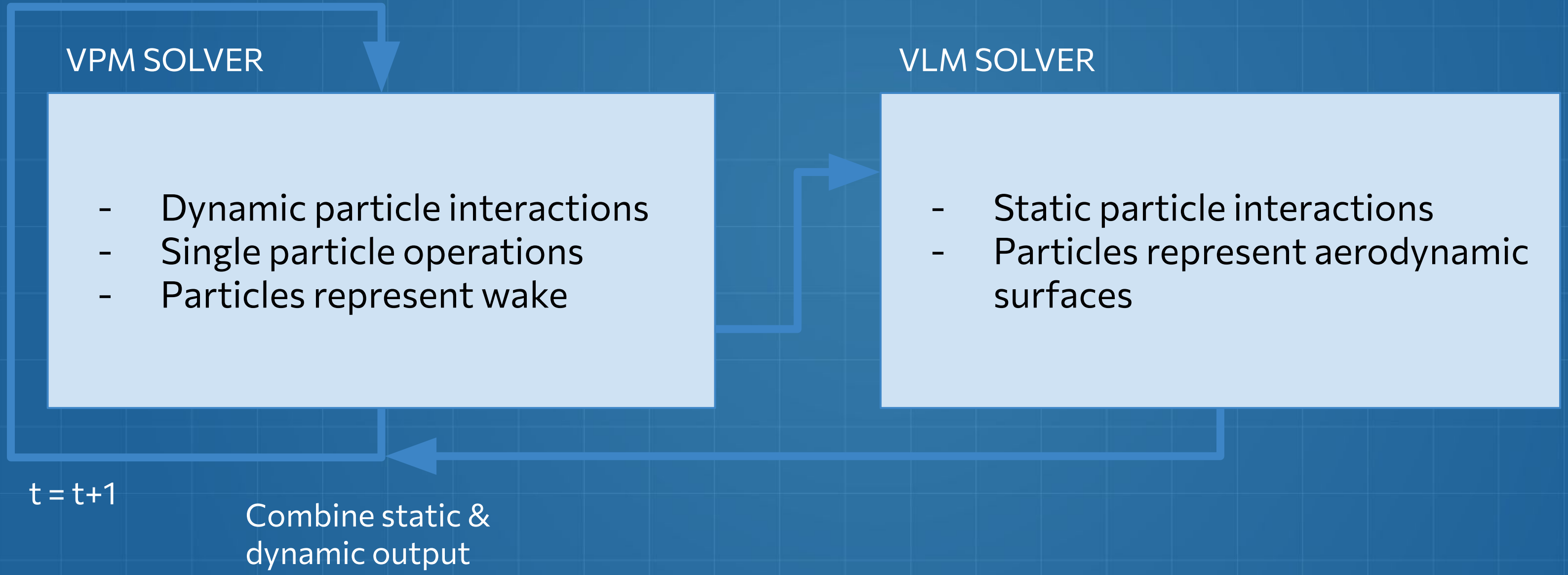
GPU-VPM

A GPU-based aerodynamics solver

MILESTONE 2

Nadine Adnane | Dominik Kau | Shreyas Singh

OVERVIEW



VPM SOLVER
- Dynamic particle interactions
- Single particle operations
- Particles represent wake
VLM SOLVER
- Static particle interactions
- Particles represent aerodynamic surfaces
t = t+1
Combine static & dynamic output

UPDATE

- FLOWVLM Implementation - Static particle solver
- Wing, WingSystem, Rotor, Solver etc.
- Wrapping up solver, performing tests
- Example Airfoil data evaluation:

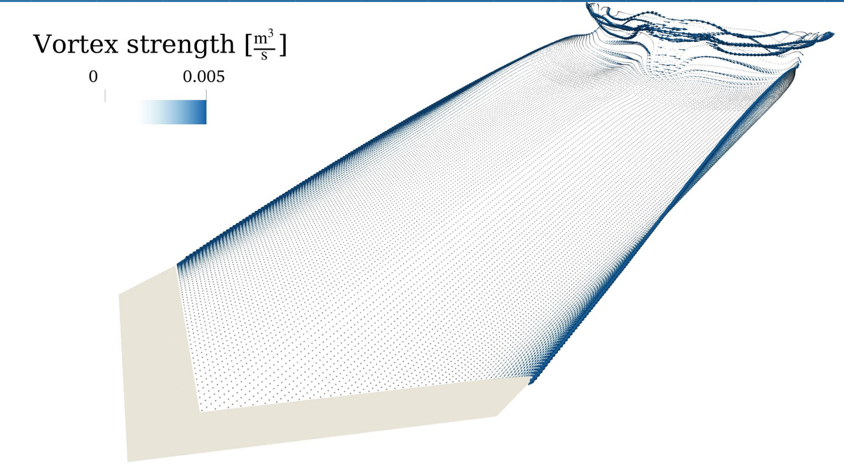
```cpp
// Example inputs
std::vector<double> alpha = { -10, 0, 10, 20, 30 };
std::vector<double> cl = { 0.1, 0.5, 0.8, 0.4, -0.1 };
std::vector<double> cd = { 0.02, 0.03, 0.05, 0.07, 0.1 };

// Create airfoil data
OCCBAirfoilData airfoil = occb_af_from_data(alpha, cl, cd);

// Evaluate airfoil properties at alpha = 15 degrees
auto [cl_val, cd_val] = occb_airfoil(airfoil, 15.0);

std::cout << "At alpha = 15 degrees:" << std::endl;
std::cout << "CL = " << cl_val << ", CD = " << cd_val << std::endl;
```

```
At alpha = 15 degrees:
CL = 0.675, CD = 0.0597917
```



GOAL VISUALIZATION

NEXT STEPS - MILESTONE 3

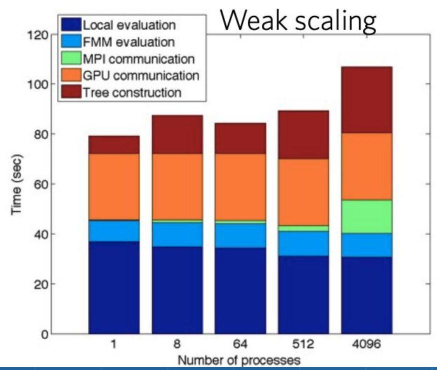
1. Complete VLM implementation
    - Finish solver, perform testing
2. Complete VPM implementation
    - Finish naive solver, perform testing
    - Start implementing Fast Multipole Method
3. Integration of VLM results into VPM
4. ParaView for visualization

    Obtain vorticity and velocity field

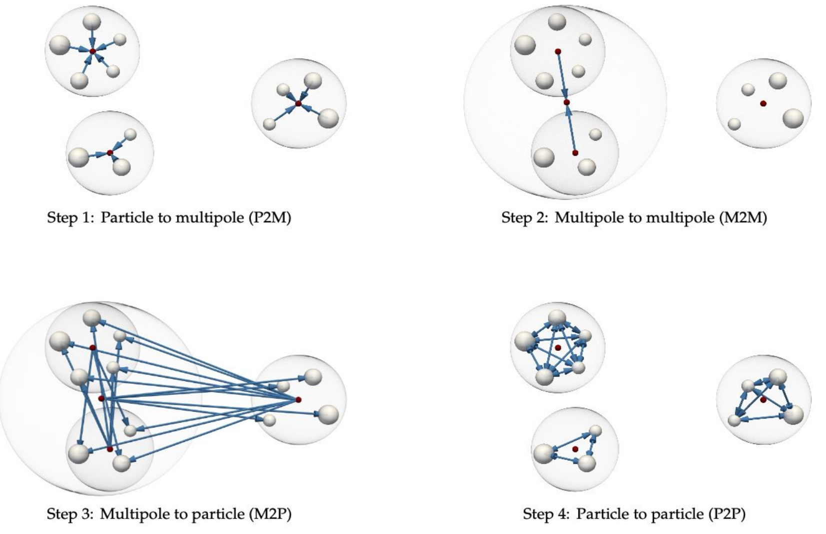
Fast Multipole Method (FMM)

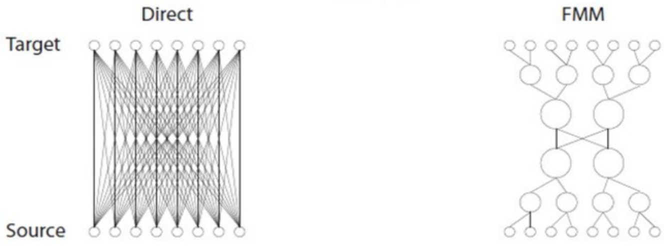
- Grouping particles that are far apart.
- Approximate interactions between distant particle groups
- ExaFMM: open-source package to utilize FMMs, in parallel, and with GPU capability.
- A treecode-FMM hybrid algorithm: auto-tuning capabilities; O(N).

- Shown to scale well to thousands of GPUs
- Performs in the order of Peta FLOPS

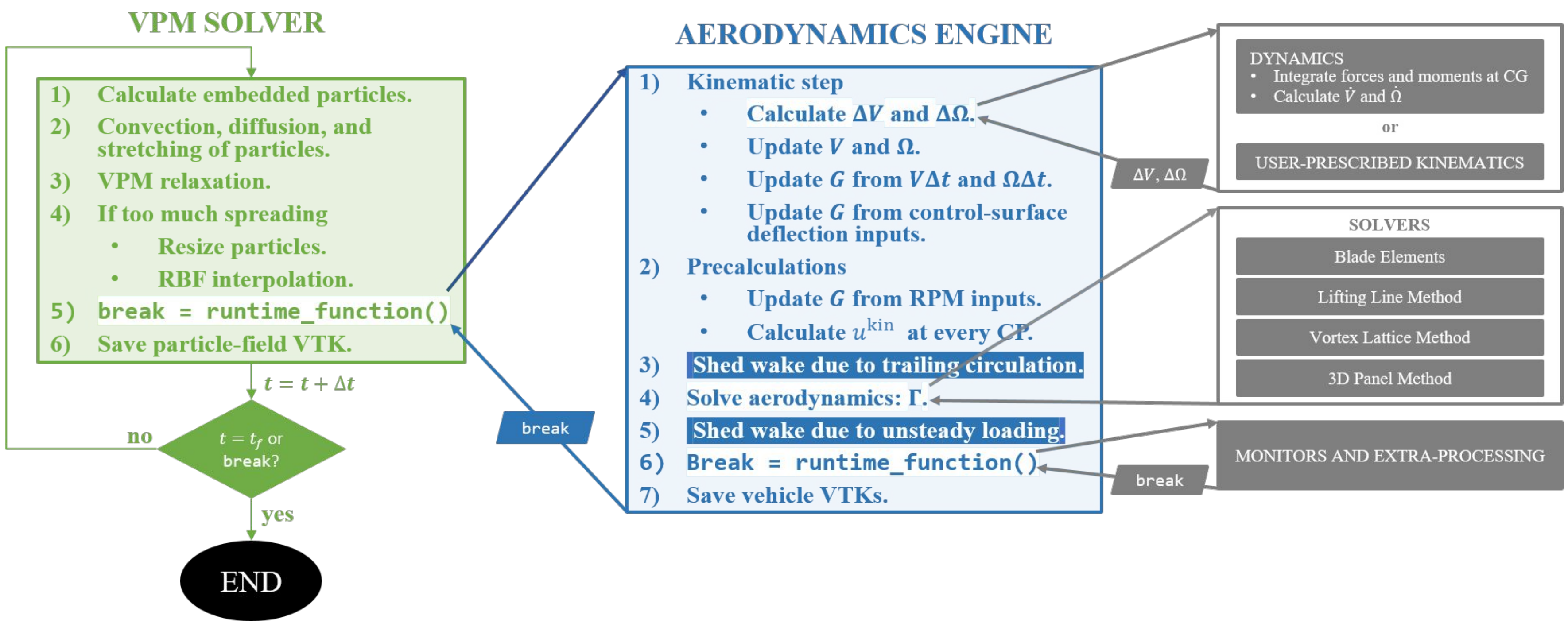


64 billion in 100 seconds
1.0 PFlops
4K GPUs on TSUBAME



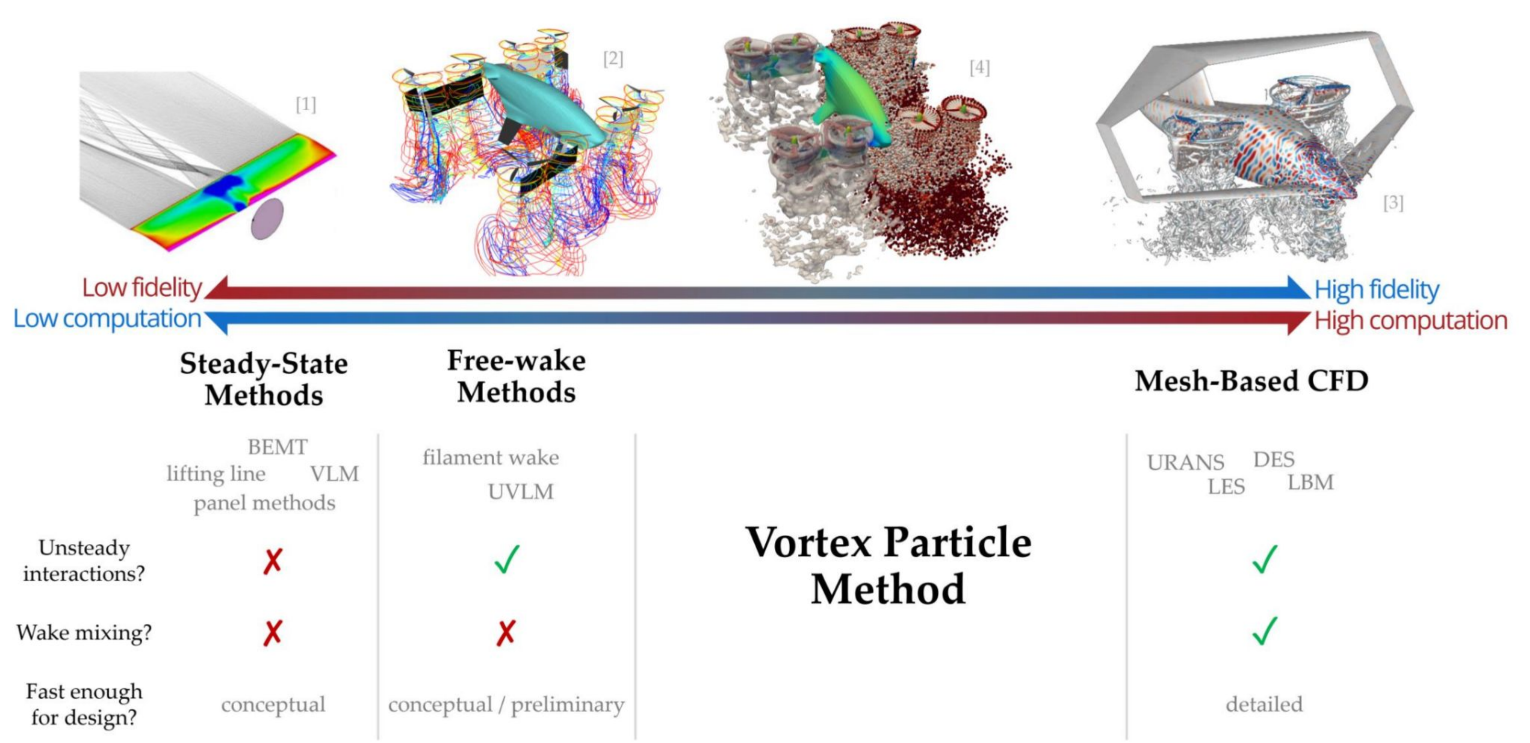


OVERVIEW



VPM SOLVER
1) Calculate embedded particles.
2) Convection, diffusion, and stretching of particles.
3) VPM relaxation.
4) If too much spreading
• Resize particles.
• RBF interpolation.
5) break = runtime_function()
6) Save particle-field VTK.
$t = t + \Delta t$
$t = t_f$ or break?
no
yes
END
break
AERODYNAMICS ENGINE
1) Kinematic step
• Calculate ΔV and $\Delta\Omega$.
• Update V and Ω.
• Update G from $V\Delta t$ and $\Omega\Delta t$.
• Update G from control-surface deflection inputs.
2) Precalculations
• Update G from RPM inputs.
• Calculate u^{kin} at every CP.
3) Shed wake due to trailing circulation.
4) Solve aerodynamics: Γ.
5) Shed wake due to unsteady loading.
6) Break = runtime_function()
7) Save vehicle VTKs.
ΔV, $\Delta\Omega$
break
DYNAMICS
• Integrate forces and moments at CG
• Calculate $\dot{V}$ and $\dot{\Omega}$
or
USER-PRESCRIBED KINEMATICS
SOLVERS
Blade Elements
Lifting Line Method
Vortex Lattice Method
3D Panel Method
MONITORS AND EXTRA-PROCESSING

PROJECT OVERVIEW



https://www.nas.nasa.gov/assets/nas/pdf/ams/2022/AMS_20220809_Alvarez.pdf