Accelerating Low-Fidelity Aerodynamic Codes

Marcin Chrust and Éric Laurendeau
Department of Mechanical Engineering, Ecole Polytechnique de Montréal, Canada

Objectives

Study the feasibility of porting low-fidelity aerodynamic codes, based on potential theory and used in the industry, to GPU using CUDA C/C++ and available libraries on the example of non-linear vortex lattice method and the determination of:

• Potential performance gains
• Overall performance to cost ratio
• Required programming effort

Introduction

Low-fidelity aerodynamic codes, including panel, lifting line and vortex lattice methods [1], are used in the preliminary aerodynamic studies in the early stages of the aircraft design and constitute an important and compute-intensive part of aircraft design process. This preliminary design phase is usually very time consuming as it involves parametric studies counting tens of thousands computations. Speeding-up low-fidelity aerodynamic codes should result in reduced research costs and faster product time to market.

Non-linear Vortex Lattice Method

Vortex lattice method is used for the simulation of incompressible, inviscid dynamic lifting surfaces and relies on potential flow theory stating that the flowfield is conservative vector field \( \mathbf{v} = \nabla \phi \) for which the continuity equation becomes \( \nabla \cdot \mathbf{v} = 0 \). The geometry is discretized using quadrilateral elements called panels and containing vortex ring singularities. A vortex ring is placed on the panel quarter chord line and its control (or collocation) point is placed at the center of three-quarter chord line.

Mathematical Details

Each straight vortex segment of strength \( \Gamma \) extending from point 1 to 2 induces velocity at an arbitrary point \( P \) given by Biot-Savart law, where \( r_1 \) and \( r_2 \) are the vectors that define position of point \( P \) in relation to points 1 and 2:

\[
\mathbf{v} = \frac{\Gamma}{4\pi} \left( \frac{r_1 - r_2}{|r_1 - r_2|^3} \right) \times \left( \frac{r_1}{|r_1|^2} - \frac{r_2}{|r_2|^2} \right)
\]

By assembling the influence of each vortex segment of each panel at a given collocation point one can write the condition of zero normal velocity in matrix form and solve for circulation at each control point. Further, Kutta-Joukowski theorem is applied to find the lift distribution. The algorithm involves therefore two major steps:

• Assembly of the influence coefficients matrix of size \( N_{\text{panels}} \times N_{\text{panels}} \)
• Inversion of the matrix.

Non-linear vortex lattice method involves coupling of the sectional data with data obtained using 3D Navier-Stokes solver in an iterative loop to account for non-linear effects such as shock waves.

Non-linear Vortex Lattice Method

Zero normal velocity on the wing surface is applied at each control point to guarantee the impermeability condition allowing the code to converge.

\[
(n_i + v_{n_i}) = 0, \quad i = 1, 2, \ldots, N_{\text{panels}}
\]

where \( n_i \) is the free stream velocity, \( v_{n_i} \) is the induced velocity and \( n_i \) the panel normal vector.

Results

All tests were carried out on Tesla K20 GPU in single precision, ECC on. In order to speed-up the code for problem of small size, Matrix Solve kernel was executed concurrently using 16 streams. Matrix Assembly speed-up was very high even for the problem of smallest considered size (\( \times 168 \)). Performance comparison with single thread execution: Intel Xeon CPU E5-1620 @ 3.6GHz.

GPU

Profiling of the code indicates that matrix assembly and matrix solve are responsible for > 98% of the total wall time in the whole range of typically encountered problem sizes. The optimization of the code was carried out with the help of NVIDIA Visual Profiler. The two functions were ported to GPU:

• Matrix Solve
  • CUDA kernel implementing bi-conjugate gradient stabilized solver using cuBLAS library.
• Matrix Solve kernel was executed concurrently in streams for problems of small size \( N_{\text{panels}} < 2048 \) (profiling indicates that the matrix inversion is the most important hotspot for problems of small size).

Important Result

Excellent code speed-ups were achieved for problems of medium to large size (\( \geq 2048 \) panels). As expected, the code speed-up for problems of small size was limited.

• Use of GPU acceleration is justified above certain problem size threshold.
• This threshold can be pushed higher by exploiting concurrent execution on GPU.
• Potential theory based low-fidelity aerodynamic codes can highly benefit from GPU acceleration.
• Panel methods, which are typically used for problems of large size (going to and above 20k panels), seem to be an especially good candidate.

References


Acknowledgments

We would like to acknowledge support of Natural Science and Engineering Research Council of Canada. We would like to thank NVIDIA Corporation for providing a Tesla K20 GPU. We would also like to acknowledge support from Calcul Québec.