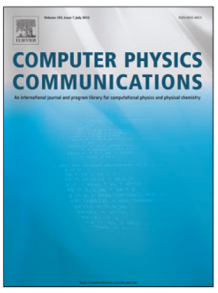
Accepted Manuscript



Accelerating high-order mesh optimisation with an architecture-independent programming model

Jan Eichstädt, Mashy Green, Michael Turner, Joaquim Peiró, David Moxey

PII: S0010-4655(18)30097-3
DOI: https://doi.org/10.1016/j.cpc.2018.03.025
Reference: COMPHY 6470

To appear in: *Computer Physics Communications*

Received date : 7 November 2017
Revised date : 24 January 2018
Accepted date : 27 March 2018

Please cite this article as: J. Eichstädt, M. Green, M. Turner, J. Peiró, D. Moxey, Accelerating high-order mesh optimisation with an architecture-independent programming model, *Computer Physics Communications* (2018), https://doi.org/10.1016/j.cpc.2018.03.025

Accelerating high-order mesh optimisation with an architecture-independent programming model

Jan Eichstädt[a], Mashy Green[a], Michael Turner[a], Joaquim Peiró[a], David Moxey[b]

[a] *Department of Aeronautics, Imperial College London*
[b] *College of Engineering, Mathematics and Physical Sciences, University of Exeter*

Abstract

Heterogeneous manycore performance-portable programming models and libraries, such as *Kokkos*, have been developed to facilitate portability and maintainability of high-performance computing codes and enhance their resilience to architectural changes. Here we investigate the suitability of the *Kokkos* programming model for optimizing the performance of the high-order mesh generator *NekMesh*, which has been developed to efficiently generate meshes containing millions of elements for industrial problem involving complex geometries. We describe the variational approach for *a posteriori* high-order mesh optimisation employed within *NekMesh* and its parallel implementation. We discuss its implementation for modern manycore massively parallel shared-memory CPU and GPU platforms using *Kokkos* and demonstrate that we achieve increased performance on multicore CPUs and accelerators compared with a native *Pthreads* implementation. Further, we show that we achieve additional speedup and cost reduction by running on GPUs without any hardware-specific code optimisation.

Keywords: high-order mesh optimisation, architecture-independent programming model, *Kokkos*, portability, parallel hardware, variational framework

1. Introduction

High-order spectral element methods are gaining support within the computational fluid dynamics (CFD) community. They offer improved solution accuracy for a given computational cost due to their exponential convergence and show very low dispersion and diffusion errors, giving these methods an edge over traditional low-order methods [1]. Although the use of high-order methods is becoming increasingly common in academic studies, a significant bottleneck in their more widespread adoption in industrial applications is the availability of robust high-order meshing capabilities for complex three-dimensional geometries, and their efficiency on current and future high-performance computing (HPC) systems [2].

The standard approach to generate a high-order mesh is to deform an initial coarse linear mesh, which can be obtained using one of the many available linear meshing tools, to conform with the curved boundary specified by the CAD geometry. This *a posteriori* process will likely yield very distorted or inverted elements close to the boundary, as the introduction of curvature into the element frequently leads to self-intersection. We therefore require a second step, that corrects invalid elements through a boundary-induced mesh deformation, so that curvature is introduced into elements connected and in close proximity to the curved surface. Several different techniques for this step have been proposed in the literature, which can be broadly classified into two categories: *elastic analogies* where the mesh is treated as a solid body and the curvature acts a force on the body, e.g. [3, 4, 5], and *energy minimisation techniques* in which a functional representing mesh distortion is minimised to optimise mesh quality and correct invalid elements, e.g. [6, 7]. Alternatively, high order meshes can be adapted by combining mesh curving and mesh topology changes, as presented for example in reference [8].

These techniques in general are computationally expensive, since they require either the solution of a partial differential equation or a non-linear optimisation to obtain the corrected mesh. In an industrial setting, where geometries are typically extremely complex and meshes can consist of millions or billions of elements, this process can be computationally prohibitive. Modern design lifecycles also demand the generation and optimisation of meshes in the order of minutes or hours, typically on only a single high-performance workstation.