# Final Project:
# Flexible-order finite difference computations

Ph.D. Course 2011:
Scientific GPU Computing

This project is concerned with both implementing and evaluating the performance of a flexible-order finite difference kernel for execution on a heterogenous CPU-GPU hardware system. The goal will be to try and maximize throughput performance of the kernel for execution on GPU architectures. Implementation should be both in CUDA and OpenCL.

Consider the general formula for flexible-order finite difference approximations of the $q$'th derivative of a function $f(x)$ in one space dimension

$$\frac{\partial^q f}{\partial x^q} \approx \sum_{n=-\alpha}^{\beta} c_n f(x_{i+n})$$

where $c_n$ is finite difference coefficients which can be computed using the supplied C function `fdcoeffF.c` and the function $f(x)$ is evaluated at a discrete grid $x_i = hi$, $i = 0, 1, ..., N-1$, with uniform spacing between grid points of size $h = \frac{1}{N-1}$. $\alpha$ and $\beta$ are integer values indicating the number of points, respectively, to the left and right of the expansion point $x_i$. Take $\alpha = \beta$ for all interior points sufficiently far from the boundaries. Near the domain boundaries at $x_0$ and $x_{N-1}$ the stencils will need to be off-centered.

- Familiarize yourself with the supplied sequential code for computing approximations of the $q$'th derivative on the discrete grid with $N$ spatial points in one space dimension on a CPU.
- Assess and describe the characteristics (fx. bandwidth, hardware limits, peak performance, etc.) of the heterogenous CPU-GPU hardware you will use for this project. Test correctness of output against output from the supplied CPU code version.
- Write a parallel GPU version of the sequential code using the CUDA programming model to investigate the potential for speeding up the computations relative to the CPU-only version.
- Carry out performance tests and try to maximize throughput for various sizes of stencils with rank $r = \alpha + \beta + 1$ with ranks $r = 3, 5, 7, ...$ while exploiting the memory hierarchy of the architecture. Any incremental improvements should be reported and documented (ranging from naive to most optimized GPU kernel). Include both absolute and relative measures of performance (fx. timings, throughput, transfers, etc.) and possibly other interesting performance indicators. [HINT: e.g. use the compute profiler, occupancy calculator, etc.]
- Write a parallel GPU version of the sequential code using the OpenCL programming model. Redo the tests you have done using the CUDA version for performance comparison. Test the the codes on available GPU architectures.
- Compare the performance of your code with the performance of a software library (e.g. use cusp-library). Discuss differences in performance.

The answers should be given in a short report in PDF file format containing your analysis, results and conclusions. The final code should be supplied in source and included in an appendix and include sufficient comments to understand your program code and reproduce your results.