HPC for Mathematicians

Suggested Final Projects

Feb, 2021

1 Heat equation on a plate with an arbitrarily shaped hole

This project is a good example of complex geometries that may come up in real world applications.

You solve a steady 2D heat equation

$$T_{xx} + T_{yy} = 0, \tag{1}$$

on rectangular plate which has a hole in it, but the shape of the hole may change from one case to another. Some examples of these geometries are given in figure 1. Your code should be able to handle arbitrary shapes of inner holes. The temperature at the outer edge (the red lines) is given by T_o and at the inner edge (the blue lines) by T_i, and they are different. You need to find the temperature distribution on this plate which comes from solving (1) subject to these boundary conditions. You can use any numerical scheme to solve this equation. If you are using a finite difference scheme, you can start with an initial guess (e.g. $T(x, y) = (T_i + T_o)/2$ for the points inside the boundaries) and then keep iterating until you reach the steady solution. To make sure that you have reached the steady state, you can set up a criteria (for example the maximum difference between two iterations should be less than ϵ).

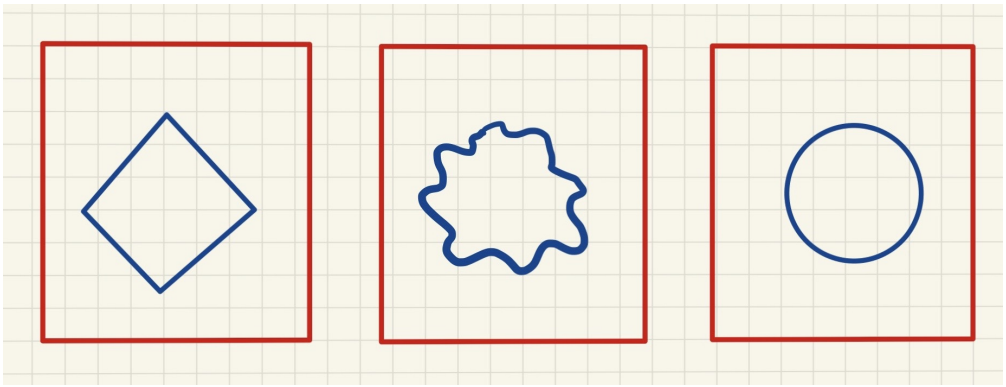

Figure 1: Examples of plates with different holes in them.

The tricky part of this project is decomposing the computational space. You should try to keep the load balance and have an algorithm that can handle any hole shape while keeping the load between different processes balance. You may split the domain in horizontal/vertical strips with varying size so all have similar number of grid points. Another approach could be solving the heat equation in polar coordinate and giving pie slices to different processes. Note that you still need halo swapping regardless of your geometric decomposition. You may also consider small blocks and use the task-farm model of parallelisation. Also some grid points may reach steady state faster

than the others and you may be able to stop the computations for them. Test your code for several arbitrarily shaped holes and compare the performance of different cases.

2 High-order parallel time-integrators

One of the challenging areas for parallelisation is time-integration, because calculation at one time-step usually depends on previous steps. There has been several efforts to parallelise time integration (e.g. look at Parareal https://en.wikipedia.org/wiki/Parareal), but one may argue that we have not been super successful! In this project, you will look at a scheme that is used to solve ODEs more accurately by pipe-lining a prediction process and several corrections. While the prediction process is finding the rough estimate for advanced time-steps, the correction processes try to correct the solution at previous time-steps by numerically solving an ODE for error terms. You will be given a python script for this project, where the basic idea is implemented. You need to parallelise this code in c++/FORTRAN, and potentially develop it.

These references explain the algorithm behind this scheme:

- Andrew J Christlieb, Colin B Macdonald, and Benjamin W Ong. Parallel high-order integrators. *SIAM Journal on Scientific Computing*, 32(2):818–835, 2010
- Andrew Christlieb, Benjamin Ong, and Jing-Mei Qiu. Comments on high-order integrators embedded within integral deferred correction methods. *Communications in Applied Mathematics and Computational Science*, 4(1):27–56, 2009 https://msp.org/camcos/2009/4-1/camcos-v4-n1-p02-s.pdf

3 Global minimum for benchmark optimisation problems

Consider the following optimisation test functions on the given domains.

- **Eggholder function**

$$f(x_1, x_2) = -(x_2 + 47)\sin\left(\sqrt{\left|\frac{x_1}{2} + x_2 + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - x_2 - 47|}\right), \tag{2}$$

  defined for $(x_1, x_2) \in [-512, 512]^2$.

- **Shekel function**

$$f(\mathbf{x}) = -\sum_{i=1}^{m}\left(\sum_{j=1}^{4}(x_j - C_{ji}) + \beta_i\right)^{-1}, \tag{3}$$

  where x_i are defined on the hypercube $x_i \in [0, 10]$ for $i = 1, 2, 3, 4$, and the coefficients are

$$m = 10 \tag{4}$$

$$\beta = \frac{1}{10}(1,\ 2,\ 2,\ 4,\ 4,\ 6,\ 3,\ 7,\ 5,\ 5)^T \tag{5}$$

$$C = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix} \tag{6}$$

The goal of this project is to present an efficient parallel algorithm that finds the global minima of these test functions. You may use some classical methods (like steepest gradient) or more modern ones (like swarm intelligence) or combination of both. You need to explore the space properly and then derive the exact minimum point up to at least 4 decimal accuracy. Think about the way you can explore the space better using parallel processes.

4 Parallel Molecular Dynamics

Molecular dynamics is one of the largest applications for high performance computing. In one of the most popular forms, the quantum mechanical interactions of a collection of atoms are replaced by classical forces derived from an empirical potential energy function with periodic boundary conditions used to keep the atoms restricted to a compact spatial domain. Atoms then move according to Newton's laws of motion, in a similar way to the heavenly bodies. Parallel computers can be employed in a variety of ways to speed up the process (or to allow the treatment of a larger number of atoms). The simplest idea is to use a multiprocessor to compute the forces at each time-step more rapidly than could be done using a uniprocessor. While molecular systems often have complicated force fields, testing of the parallel methods could be performed using a simple "Lennard-Jones fluid" in which all atoms interact in the same pair potential expressed in terms of the distance between them.

- Ben Leimkuhler and Charles Matthews. *Molecular Dynamics*. Springer, 2016
- Roman Trobec, Marián Vajteršic, and Peter Zinterhof. *Parallel computing: numerics, applications, and trends*. Springer Science & Business Media, 2009
- Dušanka Janežič, Urban Borštnik, and Matej Praprotnik. *Parallel Approaches in Molecular Dynamics Simulations*, pages 281–305. 06 2009 https://tinyurl.com/1756hxfu

5 Parallel Elastic Averaging for Bayesian Statistical Modelling

Many statistical problems consist of finding optimal parameters so that a prescribed statistical model represents a given data set; the starting point for such tasks is Bayes' formula which provides a probabilistic model for the parameters given the data. Many machine learning challenges can be cast in this form; a conceptually simple challenge is to fit a Gaussian mixture model (GMM) to a set of given points in Euclidean space, a sort of 'clustering' task. "Training" of a statistical model involves taking a sequence of steps using an optimisation method (typically "stochastic gradient descent - SGD), a task that can be accelerated by use of parallel computers. An example of an algorithm for parallel training is "elastic averaging stochastic gradient descent" (EASGD). In EASGD one performs a sort of stochastic optimisation by distributing the data set across a set of compute nodes; gradients are then computed separately on the different nodes and exchanged with a manager that adjusts the model parameters sequentially.

- SGD: https://en.wikipedia.org/wiki/Stochastic_gradient_descent
- Sixin Zhang, Anna Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd: 3rd international conference on learning representations, iclr 2015. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015
- GMMs: http://www.ceremade.dauphine.fr/%7Exian/mixo.pdf

- An application of a GMM to traffic modelling: https://academic.oup.com/tse/article/1/2/145/5618804
- Some (complicated) algorithms discussed here: https://static.epcc.ed.ac.uk/dissertations/hpc-msc/2017-2018/Nestor_Sanchez-dissertation_report.pdf