Further topics

We want to list some further topics of current interest in Scientific Computing that we did not cover in this module. The list can not be exhaustive and there will be things of importance that I am leaving out. But it should give some pointers for those who are interested in diving more into the research side.

Domain Decomposition Methods

We have not discussed domain decomposition methods at all. The idea is to decompose a computational problem into a number of subproblems that can be solved independently and exchange information via interface conditions. The individual subproblems can then be solved in parallel. This approach is important to achieve weak scaling for larger problems on massively parallel architectures. Research into domain decomposition methods combines techniques from Partial Differential Equations (coupled PDE problems, design of interface conditions), Numerical Linear Algebra (efficient preconditioning, adapted iterative solvers, etc.) and Computer Science (load balancing, network communication, and much more).

Randomized Linear Algebra

It might seem counterintuitive. But we can learn a lot about matrices by multiplication with random vectors. Indeed, we can learn so much that we can design efficient methods for problems such as singular value decompositions, dense linear solvers, eigenvalue computations, and many other standard problems of numerical linear algebra. Modern randomized methods approximate solutions to these problems in a probabilistic sense, with an error probability that is so small that in practice we need not worry about it. This area of research has become extremely prominent in the last 10 years and is more and more used for application problems of all sizes. A very recent overview is given in the article Randomized numerical linear algebra: Foundations and algorithms by Martinsson and Tropp.

Fast direct solvers

Fast direct solvers are a technology to compute approximate inverses to many relevant PDE problems in close to lnear complexity. This is made possible through compressing Green's function interactions between different points of the computational domain in the inversion. Fast direct solvers have shown tremendous success for certain types of linear systems arising for the solution of non-oscillatory stationary problems. A beautiful overview article is Fast direct solvers for integral equations in complex three-dimensional domains by Greengard, Gueyffier, Martinsson and Rohklin.

Space-Time Parallel Methods

We have seen very simple discretisation of time-stepping methods for first and second order systems. However, while we can easily parallelise the space discretisation, we have to compute one timestep after another, which limits parallelisation opportunities on large HPC machines. The idea of space-time parallel methods is to form a large linear system that contains all space variables at all time steps and then to develop efficient preconditioned methods that solve these problems. More information is given in the article 50 Years of Time Parallel Time Integration by Martin Gander.

Reproducibility in Computational Sciences using containers

Reproducibility means that we want to provide means that make it easy for other people to run our codes and data to reproduce the output that we are using in our publications. This sounds straight forward, but is fiendishly complex due to widely varying hardware configurations, operating systems, tool libraries, etc. In recent years Docker container have become more and more established as a tool to achieve reproducibility. Docker containers allow us to pack all required libraries and software into one image that can be run on various operating systems. A great introduction to reproduciblity and use of containers in computational sciences can be found at https://lorenabarba.com/tag/reproducibility/.

Machine Learning meets Scientific Computing

The tremendous importance of machine learning is undisputed and there are a number of modules at UCL, which teach all aspects of machine learning. However, machine learning has developed quite independent from traditional scientific computing with its own tools and libraries. Recently more and more researchers have become interested in merging more traditional scientific computing and modelling with machine learning. The idea is to mix statistical and PDE based models to significantly improve the predictive power of computational simulations. This is an emerging area with a huge need for mathematical and computational and natural sciences research together with the development of suitable computational tools.

The Julia Programming Language

We have focused in this module on using Python. It is a very powerful programming language, which is used in a number of HPC projects. However, Julia is growing significantly as programming language and it is a number of novel ideas and concepts. A few years ago it was demonstrated that a pure Julia application was able to scale to petascale performance, which was a huge breakthrough. For new projects that do not need to be developed in a low-level language such as C++ or Fortran I strongly consider to evaluate both, Julia and Python as environments.