

Numerical Methods for Reliable GPU Acceleration in Scientific Computing

Finn Smith

PhD research proposal, discussion draft, 27 October 2026

Abstract

This proposal develops numerical methods for scientific computation on graphics processing units, with constrained optimisation as the first application. An optimisation algorithm repeatedly solves linear systems whose coefficients and conditioning change as the solution is approached. Lower precision and reuse of previous work can reduce the cost of these solves. Their benefit depends on whether the resulting steps still carry the computation to an acceptable solution. The proposed research studies how precision, iterative refinement and factorisation refresh should adapt along this sequence. It will develop a numerical decision rule, analyse its effect on convergence under stated assumptions, and evaluate complete solution time at common accuracy requirements. The intended contribution is a method for deciding where computational effort is needed, supported by reproducible evidence of the problems and regimes in which it is useful.

1. Research motivation

Scientific computing often involves solving a sequence of related problems. In a primal-dual optimisation method, each iteration requires a linear system that determines the next step. Its sparsity pattern may remain fixed while its numerical values change substantially. Approaching the solution can also make the system increasingly ill-conditioned. These features create opportunities to reuse computation, but they also determine when reuse ceases to be effective.

GPUs make the numerical formulation particularly important. The precision of a factorisation, the work assigned to iterative corrections and the movement of data between CPU and GPU all affect the time required to obtain a solution. An inexpensive linear solve may produce a poor optimisation step, while an unnecessarily accurate solve can spend work that the current iteration does not need. My interest is in how those decisions can be made from the progress of the computation itself.

When should a GPU solver reuse, refine or rebuild its numerical approximation so that the required solution accuracy is reached with less total computation?

The first study will address sequences of sparse systems arising in constrained control or engineering design problems. These provide nonfinancial applications with explicit feasibility and optimality criteria. The wider ambition is to make larger scientific calculations practical within a stated time and memory budget, with numerical quality measured alongside performance.

2. Research position and preparation

Mixed-precision iterative refinement already has an established error analysis [1]. GPU-resident optimisation methods also combine approximate factors with iterative corrections [2], while condensed interior-point methods study the interaction between KKT structure, conditioning and GPU execution [3]. CuClarebel supplies a further GPU solver and mixed-precision comparison [4]. These results establish the starting point. The proposed contribution concerns joint decisions about precision and factorisation refresh across a defined sequence, with the cost of failed corrections and delayed convergence included in the comparison.

My preparation is in numerical optimisation and statistical computation. I have implemented damped Newton and quasi-Newton methods, line searches and a Mehrotra predictor-corrector method for a constrained quadratic programme. My MSc dissertation developed simulation, estimation and inference under limited observability. This work provides experience tracing numerical choices through to the result an application uses. The GPU study carries that numerical foundation into a new computational setting.

3. Problem formulation and proposed method

The initial application is a convex quadratic programme,

$$\min_x \frac{1}{2}x^\top Qx + c^\top x, \quad Ax = b, \quad x \geq 0, \quad Q \succeq 0.$$

The initial analysis assumes primal and dual strict feasibility, full row rank of A and a finite optimum. A primal-dual method produces a sequence of linear systems

$$K_k d_k = b_k,$$

where K_k is the chosen KKT formulation and d_k determines the next optimisation step. The research will begin with one specified formulation, scaling procedure and solver implementation. This fixes the numerical problem before introducing adaptation.

Let $\mathcal{H}_k$ contain information available at iteration k , including recent residual reduction, refinement effort, step acceptance and indicators of matrix change. A proposed policy $\pi(\mathcal{H}_k)$ will select the working precision and decide whether an existing approximate factorisation should be reused or replaced. Residuals will be recomputed against the current system in a higher verification precision. Reused factors can serve as a preconditioner for a correction method even when they no longer approximate the current matrix closely enough for a direct solve.

A normalised linear backward-error diagnostic is

$$\eta_k = \frac{\|b_k - K_k \hat{d}_k\|}{\|K_k\| \|\hat{d}_k\| + \|b_k\|}.$$

This measures how well the computed step solves the linear system. Conditioning determines how that relates to forward error. Step acceptance will be tied to an inexact-Newton forcing condition for the original primal-dual equations, after recovering the full step from any scaling or condensation. Final termination will use feasibility, complementarity and optimality residuals for the original problem. Regularisation and scaling will be recorded explicitly.

The central hypothesis is that combining observed refinement behaviour with changes in the system and progress of the outer iteration can reduce avoidable refactorisation and over-solving. The method will include explicit triggers for rebuilding factors or increasing precision when progress deteriorates. Its analysis will seek sufficient conditions on the accepted inexact steps that preserve convergence for the selected problem class. The novelty to establish is the specific decision rule and its analysis relative to existing adaptive solvers.

4. Research experiments

1. **Establish the numerical regimes.** Construct controlled sequences and use public control or engineering design instances to vary conditioning, sparsity, dimension and the rate of matrix change. Compare a reliable double-precision reference with fixed mixed-precision and fixed-refresh policies. Measure where corrections succeed, where reuse becomes expensive and where the outer method loses progress.
2. **Develop and test the adaptive method.** Compare the proposed policy with established GPU approaches at the same final acceptance criteria. Separate the effects of precision selection, refinement and refresh through controlled comparisons. Assess complete runtime, successful solves, primal and dual residuals, complementarity gap, peak memory and the cost of fallback. Include transfers, setup and additional outer iterations in the timing.
3. **Evaluate transfer to an application.** Freeze the policy and test on distinct problem instances with a domain-relevant output. For control this could include constraint satisfaction and the resulting control objective. Report performance across the full test set, including failures and cases where the reference remains preferable. The first paper will connect the numerical rule, its analysis and this application evidence.

5. Scientific contribution and industrial use

The proposed thesis has three connected contributions. The first is a numerical analysis of when approximate factors remain useful along a changing sequence of KKT systems. The second is an adaptive method that uses this information to allocate precision, corrections and refactorisation. The third is a reproducible implementation showing how these decisions affect the time and memory needed to reach a solution of stated quality. The analysis will explain useful and failing regimes, providing a basis for choosing the method in an application.

For an industrial GPU platform, this work creates an application-level account of performance. A user needs to know which problem can be solved, at what accuracy and cost, and how those outcomes change as the problem grows. I propose to own the numerical formulation, reference implementation and validation of one such application, working with the team’s GPU implementation. The same application can provide a demonstrable pilot and the scientific setting for the doctoral question. Commercial value would follow from a verified reduction in the resources required for a useful result.

Quantum simulation is a possible subsequent application if it matches the venture’s actual work. Classical GPUs can simulate quantum circuits and device dynamics, as illustrated by NVIDIA’s cuQuantum libraries [5]. That is distinct from operating physical qubits. Any extension would require a specific quantum problem and suitable domain supervision, with error measured in the physical quantities of interest as well as numerical residuals. The first study remains focused on constrained optimisation.

6. Research setting and first stage

I propose to discuss this programme with Professor Philip Treleaven as a possible part-time PhD, supported by appropriate numerical-analysis and GPU expertise. A prospective industrial collaboration would provide representative workloads, implementation knowledge and access to the target NVIDIA hardware. Academic approval, funding, hardware access and agreements on IP and publication remain to be established.

Two or three MSc projects could investigate self-contained supporting questions, such as a reference application, an independent problem family or a replication study. My doctoral work would comprise the central method, its analysis and the synthesis of the scientific evidence. Each student’s contribution would be attributed, with UCL academic supervision and agreed rights to use the resulting work.

The first stage will reproduce the relevant solver baselines, select one application and identify a specific failure or cost pattern that existing policies do not handle well. It will then develop and analyse one adaptive rule and test it on reserved instances. This provides a bounded route to a first methods paper and an application that the team can evaluate. Further thesis work would follow from the numerical questions exposed by that study.

References

- [1] E. Carson and N. J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM Journal on Scientific Computing*, 40(2):A817–A847, 2018. Published manuscript.
- [2] K. Świrydowicz et al. Iterative methods in GPU-resident linear solvers for nonlinear constrained optimization. 2024. arXiv:2401.13926.
- [3] F. Pacaud et al. Condensed interior-point methods for scalable nonlinear programming on GPUs. Revised July 2026. arXiv:2405.14236v3.
- [4] Y. Chen, D. Tse, P. Nobel, P. Goulart and S. Boyd. CuClarabel: GPU acceleration for a conic optimization solver. Revised November 2025. arXiv:2412.19027v3.
- [5] NVIDIA. *cuQuantum: quantum simulation libraries for GPUs*. Official documentation. Accessed 27 October 2026.