Example Solutions
cs542g Final Exam
December 5, 2006

1. . . . .

In some cases the Moving Least-Squares (MLS) approximation to scattered data points is undefined. **Describe when this can happen.**

One possible solution is to use the pseudo-inverse of the least-squares matrix involved. Another possible solution is to dynamically adjust the width of the kernel to include k sample points for some suitably large and fixed k: i.e. to get the estimate at point x, weights $W(\alpha|x - x_i|)$ are used for sample points x_i, with α chosen so that at least k weights are nonzero. **Discuss the relative merits of these two possibilities.**

. . . .

MLS is undefined when the least-squares problem is underdetermined: for example, if not enough points are within the support of the kernel function, or the points are arranged in a degenerate geometry (e.g. all on a line) that cause rank-deficiency in the matrix.

Using the pseudo-inverse will select a minimum norm solution in all of these cases, giving an interpolant defined everywhere, but at the expense of having to compute it (perhaps using the SVD, considerably more expensive than solving a well-posed linear least-squares problem). The pseudo-inverse can also cause discontinuities in the interpolant: e.g. far enough away from the data points, it jumps to zero (when $A = 0$).

Adjusting the width of the kernel protects against some break-downs (not enough sample points) but not all (if all k points are on a line, the matrix can still be rank-deficient). It also may be somewhat expensive to determine the radius requried (the k nearest-neighbour problem).

2. . . . .

The PDE $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$ for time $t > 0$ and one-dimensional space x is first discretized in space with one-sided finite differences as:

$$\left(\frac{\partial u}{\partial t}\right)_j = -\frac{u_{j+1} - u_j}{\Delta x}$$

What are the eigenvalues of the Jacobian? (hint: with boundary conditions unspecified, the eigenvectors are of the form $v_j = e^{i\zeta j}$ where $i = \sqrt{-1}$ and ζ is a real constant) **What sort of time integration scheme will stably**

converge? Would you suggest any changes to the spatial discretization?

. . . .

Plugging in the suggested eigenvector to the finite difference expression to find the eigenvalue gives:

$$\begin{aligned} -\frac{v_{j+1}-v_j}{\Delta x} &= \lambda v_j \\ -\frac{1}{\Delta x}(e^{i\zeta(j+1)}-e^{i\zeta j}) &= \lambda e^{i\zeta j} \\ -\frac{1}{\Delta x}(e^{i\zeta}-1) &= \lambda \end{aligned}$$

Writing out λ in terms of its real and imaginary parts gives:

$$\lambda = \frac{1-\cos\zeta}{\Delta x} - i\frac{\sin\zeta}{\Delta x}$$

Note that this means $\Re(\lambda) \geq 0$: we cannot hope for a stable convergent solution, no matter the time integration scheme, because even exact integration gives exponentially increasing solutions.

The problem is that the spatial discretization is "down-winded": one-sided but only letting information flow the wrong way (opposite the actual flow of information in the PDE). The CFL condition will never be met. Switching to an up-wind difference such as $-(u_i - u_{i-1})/\Delta x$ solves this issue.

3 . . . .

The heat equation $\frac{\partial u}{\partial t} = \Delta u$ is to be solved using standard second order finite differences in space on a large 3D uniform grid. A massively parallel architecture is used, whose communication costs mean the best available solver for linear systems is Jacobi iteration. **Discuss the relative merits between an explicit time integration scheme and an implicit scheme for this problem.**

. . . .

The critical question is how expensive is it to get a solution of desired accuracy at some final time T. Note that roughly the same amount of work is incurred in an evaluation of an explicit method as in one iteration of Jacobi, so we can use number of steps (explicit) or number of total iterations (implicit) as our gauge of cost.

Let A be the matrix representing the second-order accurate finite difference approximation of Δ in 3D. It's not hard to see the eigenvalues are negative and real, and range between $\lambda_{\max} = -O(1/\Delta x^2)$ (with a checkerboard-like eigenvector:

$v_{ijk} = (-1)^{i+j+k}$) and $\lambda_{\min} = -O(1)$ (with a very smooth eigenvector). For example, on a square domain with Dirichlet boundary conditions, the Fourier modes are the eigenvectors and these eigenvalues are apparent.

For a typical explicit method, such as Runge-Kutta or multi-step methods, the stability time restriction is then $\Delta t < O(\Delta x^2)$, based on requiring $|\lambda_{\max}\Delta t| < O(1)$. Beyond the scope of this course, but even for some specialty explicit methods along the lines of Dufort-Frankel (which are unconditionally stable) the CFL condition will require that Δt approaches zero faster than Δx for convergence: there too we can expect to require $O(n^2)$ time steps.

For an implicit method, such as Backwards Euler or BDF methods, an arbitrarily large time step can be taken stably, with the need to solve a system of linear equations of the form $(I - O(\Delta t)A)x = b$. (For Backwards Euler, the matrix is precisely $I - \Delta tA$.) Similar eigenvector analysis shows that the spectral radius (i.e. maximum eigenvalue) of the Jacobi iteration matrix here is

$$\rho = O\left(\frac{1}{1 + \Delta t/\Delta x^2}\frac{\Delta t}{\Delta x^2}(1 - 1/n^2)\right)$$

To get reasonable convergence for Jacobi, $O(-1/\log\rho$ iterations must be taken. This can be approximated as

$$\begin{aligned}
\frac{-1}{\log\rho} &\sim \frac{-1}{\log\left(\frac{1-1/n^2}{1+\Delta x^2/\Delta t}\right)} \\
&= \frac{1}{\log(1 + \Delta x^2/\Delta t) - \log(1 - 1/n^2)} \\
&\sim \frac{1}{\Delta x^2/\Delta t + 1/n^2} \\
&\sim \frac{n^2}{1 + 1/\Delta t}
\end{aligned}$$

Multiplying this by $O(1/\Delta t)$ time steps gives a total work of $O(n^2)$ iterations. If a large time step is used, lots of Jacobi iterations per step are required; if a small time step is used, only a few are needed per step; no matter what Δt you pick it balances out to $O(n^2)$ iterations total.

Thus asymptotically, implicit and explicit methods require the same amount of work in this case! Without knowing the constants hidden in the $O()$ notation, which would require detailed timings of different methods on the hardware, one might as well then use a simpler explicit method which will probably have a smaller time truncation error.

4. . . . .

Consider the linear least squares problem $\min_x ||Ax-b||_2$ when A is large, sparse, and nearly rank-deficient. The QR decomposition is selected for solution. **How might you go about exploiting sparsity to improve the computational efficiency?** (hint: what is R in terms of A^TA?) **What is the impact of one fully dense column in A? What is the impact of one fully dense row in A?**

. . . .

Any of the algorithms we considered (Modified Gram-Schmidt, Householder, and Givens) can be implemented in "sparse" mode (i.e. only storing nonzeros). However, fill-in during the factorization can be problematic. Storing the Q part implicitly as a sequence of Householder or Givens matrices eliminates questions of needless fill-in for Q. The R part is just the transpose of the Cholesky factor of A^TA (since $A^TA = R^TQ^TQR = R^TR$) and thus fill-reducing orderings of A^TA can be applied to permute the columns of A to reduce fill in R.

One fully dense column of A, as can readily be seen by looking at the MGS definition of QR (even if that is not how we compute it), causes R to be fully dense to the right of that column. Thus we certainly want to order those columns last.

One fully dense row of A causes A^TA to be fully dense, and thus (unless we have spectacularly lucky cancellations) R will be dense no matter what we do. As an advanced solution (well beyond the extent of this course) one could view this as a low rank perturbation from a sparse $\hat{A}$ (that is, A without the dense row), and from the sparse QR factorization of $\hat{A}$ and the Sherman-Morrison formula for how low rank perturbations perturb the inverse of a matrix derive an efficient solution method.