MATH 676

–

Finite element methods in scientific computing

Wolfgang Bangerth, Texas A&M University

Lecture 34:

What solver to use

Solver questions

The finite element method provides us with a linear system

$$A\,x \;=\; b$$

We know:

- A is *large*: typically a few 1,000 up to a few billions
- A is *sparse*: typically no more than a few 100 entries per row
- A is typically *ill-conditioned*: condition numbers up to 10^9

Question:

How do we go about solving such linear systems?

Direct solvers

Direct solvers – compute a decomposition of *A*:

- Can be thought of as variant of LU decomposition that finds triangular factors *L, U* so that

$$A = LU$$

- *Sparse direct* solvers save memory and CPU time by considering the sparsity pattern of *A*
- Very robust
- Work grows as $O(N^{1+2(d-1)/d})$, i.e.,
  – $O(N^2)$ in 2d
  – $O(N^{7/3})$ in 3d
- Memory grows as $O(N^{1+(d-1)/d})$, i.e.,
  – $O(N^{3/2})$ in 2d
  – $O(N^{5/3})$ in 3d

Direct solvers

Where to get a direct solver:

- Several very high quality, open source packages
- Most widely used ones are
  - UMFPACK
  - SuperLU
  - MUMPS
- The latter two are even parallelized

As before:

Don't get stuck with some poorly supported package found somewhere on the internet!

Choose widely used, high quality software!

Iterative solvers

Iterative solvers improve the solution in each iteration:

- Start with an initial guess x_0
- Continue iterations till a stopping criterion is satisfied (typically that the error/residual is less than a tolerance)
- Return final guess x_k

- Depending on solver and preconditioner type, work can be *O(N)* or (much) worse
- Memory is typically linear, i.e., *O(N)*

Note: The final guess does not solve $Ax=b$ exactly!

Iterative solvers

There is a wide variety of iterative solvers:

- CG, MinRes, GMRES, ...
- All of them are actually rather simple to implement: They usually need less than 200 lines of code
- Consequently, many high quality implementations

Advantage: Only need multiplication with the matrix, no access to matrix elements required.

Disadvantage: Efficiency hinges on availability of good preconditioners.

Direct vs iterative

As a general rule:

By and large, in order for your program to be correct, you need to

- compute the correct system matrix and right hand side
- solve the system correctly.

The first version of your program will almost never produce the correct result. So: *Which of the two above is it?*

To eliminate the second possible source of errors, use a direct solver in the first version of a code!

Note: See step-29 for how to do that.

Direct vs iterative

Guidelines for direct solvers vs iterative solvers:

Direct solvers:

- ✔ *Always* work, for any invertible matrix
- ✔ No need to think about preconditioners
- ✔ Faster for problems with <100k unknowns
- ✗ Need too much memory + CPU time for larger problems

Iterative solvers:

- ✔ Need *O(N)* memory
- ✔ Can solve *very* large problems
- ✔ Often parallelize well
- ✗ Choice of solver/preconditioner depends on problem

Advice for direct vs iterative

For most “small” problems:

- Direct solvers are fast enough
- Direct solvers fit into memory

You should only start thinking about iterative solvers:

- If you know that your system is correctly assembled
- Your problem is large enough
- You want to parallelize the solver

Advice for iterative solvers

There is a wide variety of iterative solvers:

- CG: Conjugate gradients
- MinRes: Minimal residuals
- GMRES: Generalized minimal residuals
- F-GMRES: Flexible GMRES
- SymmLQ: Symmetric LQ decomposition
- BiCGStab: Biconjugate gradients stabilized
- QMR: Quasi-minimal residual
- TF-QMR: Transpose-free QMR
- ...

Which solver to choose depends on the properties of the matrix, primarily *symmetry* and *definiteness*!

Advice for iterative solvers

Guidelines for use:

- CG: Matrix is symmetric, positive definite
- MinRes: –
- GMRES: Catch-all
- F-GMRES: Catch-all with variable preconditioners
- SymmLQ: –
- BiCGStab: Matrix is non-symmetric but positive definite
- QMR: –
- TF-QMR: –
- All others: –

In reality, only CG, BiCGStab and (F-)GMRES are used much.

Advice for iterative solvers

Note:

All iterative solvers are bad without a good preconditioner!

The art of devising a good iterative solver is to devise a good preconditioner!

MATH 676

–

Finite element methods in scientific computing

Wolfgang Bangerth, Texas A&M University