Bigger linear systems

KSP, preconditioning, and performance

Ed Bueler
Dept. of Mathematics and Statistics, UAF

2 February 2016

recap: vector norms

- $\mathbb{R}^N$ is space of N-dimensional column vectors
- $\|\cdot\|$ denotes a vector norm; it measures the length of a vector; for example

$$\|\mathbf{v}\|_2 = \left(\sum_{j=0}^{N-1} |v_j|^2 \right)^{1/2}$$

- a vector norm satisfies
  - $\|a\mathbf{v}\| = |a|\|\mathbf{v}\|$
  - $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$
- $\|\mathbf{v}\|_2$ in PETSc:

```
VecNorm(v,NORM_2,&result)
```

matrix norms and condition number

- $\|\cdot\|$ also denotes the *induced matrix norm*

$$\|A\| = \sup_{\mathbf{v}\neq 0} \frac{\|A\mathbf{v}\|}{\|\mathbf{v}\|}$$

- PETSc only does easily-computed matrix norms
  - in PETSc: `MatNorm(A,type,&result)` where `type` is one of

    `NORM_1,NORM_FROBENIUS,NORM_INFINITY`
  - see Trefethen & Bau, chapter 3, on norms
- the *condition number* $\kappa(A)$ of a matrix measures how much it distorts the unit ball:

$$\kappa(A) = \|A\|\|A^{-1}\|$$

  - see Trefethen & Bau, chapter 4, 5, 12, 13, 14, 15

linear systems

- suppose $A \in \mathbb{R}^{N \times N}$ and $\mathbf{b} \in \mathbb{R}^N$
- also suppose $\kappa(A) < \infty$, i.e. A is invertible
- we want to solve

$$A\mathbf{u} = \mathbf{b}$$

“direct” numerical linear algebra

- we can use *direct* algorithms which get the exact answer in finitely-many steps (if using exact arithmetic)
  - LU decomposition, a.k.a. Gauss elimination
  - Cholesky decomposition ($A = R^\top R$) if A is Hermitian and positive-definite
  - QR decomposition
- LU and Cholesky are in PETSc

some numerical facts of life

1. *limits to accuracy*: regardless of the algorithm, the solution of $A\mathbf{u} = \mathbf{b}$, can only be computed to within $\kappa(A)\epsilon$ of the correct answer if ϵ is the accuracy of the floating point operations
   - algorithms *can* do worse but they cannot do better
   - we should diagnose accuracy of results with $\kappa(A)$ in mind
2. *direct solutions are expensive*: LU and Cholesky cost $O(N^3)$ operations on general dense matrices
   - banded matrices faster if no pivoting
     - note Cholesky needs no pivoting
   - graph-theoretic reordering can speed up
     - PETSc includes nested-dissection and minimum-degree orderings
3. *do not form the inverse* A^{-1}
   - requires extra work and memory
   - often dense even if A sparse
   - . . . and not needed to solve $A\mathbf{u} = \mathbf{b}$ anyway!

residuals

- to start describing the alternative to direct methods, measure how wrong is a guess at the solution of solve $A\mathbf{u} = \mathbf{b}$
- given $\mathbf{u}_0$, the *residual* of the linear system is the vector

$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$$

  - $\|\mathbf{r}_0\|$ a measure of how "wrong" is $\mathbf{u}_0$
- compute $\mathbf{r}_0$ in time of matrix-vector product: $O(N^2)$
  - often $O(N)$ if A is sparse

Richardson (simple) iteration

- the residual is basis of (pre-computer) *Richardson iteration*:

$$\begin{aligned}\mathbf{u}_{k+1} &= \mathbf{u}_k + \omega(\mathbf{b} - A\mathbf{u}_k) \\ &= \mathbf{u}_k + \omega\mathbf{r}_k\end{aligned}$$

  - add fraction of residual in hope that $\mathbf{u}_{k+1}$ is better
- this example system has solution $\mathbf{u} = [1\ 2]^\top$:

$$\begin{bmatrix} 10 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \end{bmatrix}$$

- but if $\mathbf{u}_0 = [0\ 0]^\top$ then the $\omega = 1$ Richardson iteration gives

$$\mathbf{u}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{u}_1 = \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \mathbf{u}_2 = \begin{bmatrix} -63 \\ 9 \end{bmatrix}, \mathbf{u}_3 = \begin{bmatrix} 584 \\ -62 \end{bmatrix}, \cdots$$

- not converging . . . bad $\mathbf{u}_0$? . . . hopeless for this system?

convergence if good "spectral properties"

- rewrite Richardson as

$$\mathbf{u}_{k+1} = (I - \omega A)\mathbf{u}_k + \omega \mathbf{b}$$

- "size" of matrix $I - \omega A$ determines whether $\mathbf{u}_k$ converge
- recall *spectral radius* $\rho(B)$ is the maximum magnitude of the eigenvalues of B
- easy theorem:

Richardson converges if and only if $\rho(I - \omega A) < 1$

- also: $\rho(B) \leq \|B\|$ in any induced matrix norm, so Richardson converges if $\|I - \omega A\| < 1$
- generically: success of iterative method is controlled by "spectral properties" of the matrix . . . e.g. spectral radius or 2-norm

preconditioning (first of many slides on this . . .)

- and we have some control of the matrix!
- assume we are solving $A\mathbf{u} = \mathbf{b}$ (A and $\mathbf{b}$ fixed!)
- if $M \in \mathbb{R}^{N \times N}$ is an invertible matrix then the systems

$$(M^{-1}A)\mathbf{u} = M^{-1}\mathbf{b}$$

and

$$(AM^{-1})(M\mathbf{u}) = \mathbf{b}$$

have exactly the same set of solutions as $A\mathbf{u} = \mathbf{b}$

- referred to as *preconditioned* versions of the linear system
  - *left-preconditioning* and *right-preconditioning*, respectively
- easy to use if M and M^{-1} can be applied fast

preconditioning main idea

- solution algorithms can use better spectral properties of $M^{-1}A$ or AM^{-1}, compared to those of A, to generate approximations to solution of $A\mathbf{u} = \mathbf{b}$ more quickly
    - though *accuracy* of the approximate solution cannot be improved beyond the $\kappa(A)\epsilon$ level

redo 2×2 example

- extract the diagonal of A as M:

$$M = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \tag{1}$$

- being diagonal, M is easy to invert and apply
- consider left-preconditioned Richardson iteration using M:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \omega(M^{-1}\mathbf{b} - M^{-1}A\mathbf{u}_k) \tag{2}$$

- again $\mathbf{u}_0 = [0\ 0]^*$ and $\omega = 1$
- get

$$\mathbf{u}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{u}_1 = \begin{bmatrix} 0.8 \\ 1.0 \end{bmatrix}, \mathbf{u}_2 = \begin{bmatrix} 0.9 \\ 1.8 \end{bmatrix}, \mathbf{u}_3 = \begin{bmatrix} 0.98 \\ 1.90 \end{bmatrix}, \ldots \tag{3}$$

- apparently converging to solution $\mathbf{u} = [1\ 2]^*$
- easy explanation:

$$\rho(I - A) = 9.1 \quad \text{and} \quad \rho(I - M^{-1}A) = 0.32, \tag{4}$$

iterates = polynomials

- recall Richardson iteration with $\omega = 1$:
  $\mathbf{u}_{k+1} = \mathbf{u}_k + (\mathbf{b} - A\mathbf{u}_k)$
- get started with $\mathbf{u}_0 = \mathbf{b}$:

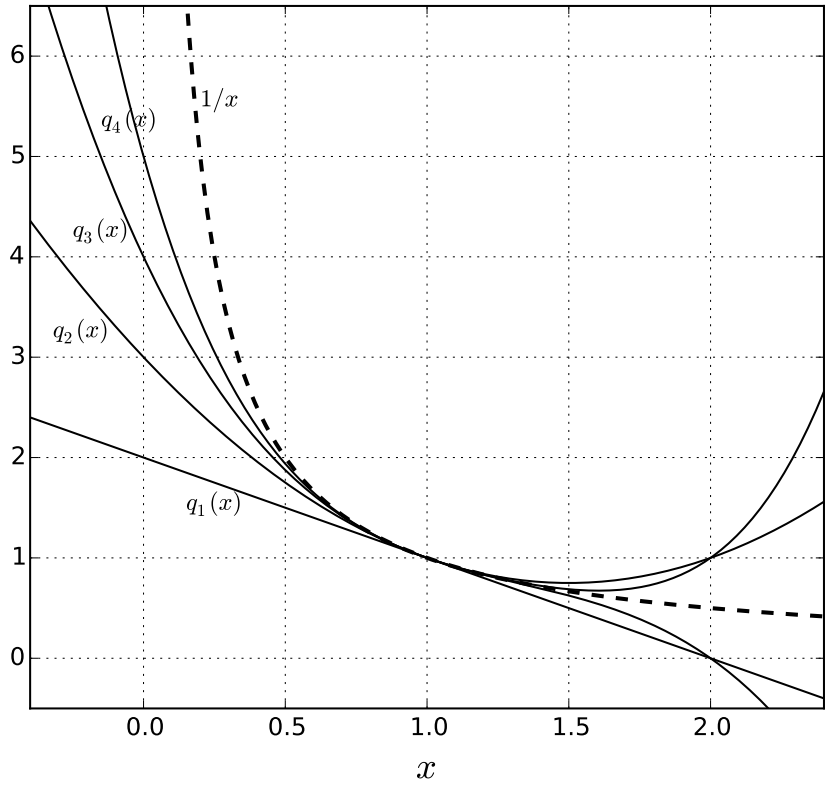
$$\begin{aligned}
\mathbf{u}_1 &= 2\mathbf{b} - A\mathbf{b} \\
\mathbf{u}_2 &= 3\mathbf{b} - 3A\mathbf{b} + A^2\mathbf{b} \\
\mathbf{u}_3 &= 4\mathbf{b} - 6A\mathbf{b} + 4A^2\mathbf{b} - A^3\mathbf{b} \\
&\dots
\end{aligned}$$

- easy to rewrite using *polynomials* applied to A:

$$\mathbf{u}_k = q_k(A)\mathbf{b} \quad \text{with} \quad q_{k+1}(x) = 1+(1-x)q_k(x), \quad q_0(x) = 1$$

iterates = polynomials 2

- in fact $q_k(x) \to 1/x$ for $x \in (0, 2)$



- on the other hand: $q_k(A)\mathbf{b} \to A^{-1}\mathbf{b}$ if $\rho(I - A) < 1$
- *summary*: Richardson iteration approximates $\mathbf{u} = A^{-1}\mathbf{b}$ by $\mathbf{u}_k = q_k(A)\mathbf{b}$ for polynomials $q_k(x)$ which approximate $1/x$ on interval $(0, 2)$

Krylov methods

- Krylov methods generalize the above idea
- given initial vector $\mathbf{v}$, the N-dimensional Krylov space is the span of N vectors:

$$\mathcal{K}_N(\mathbf{v}) = \text{span}\left\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{N-1}\mathbf{v}\right\}$$

- example: Richardson iterates are in $\mathcal{K}_{N+1}(\mathbf{b})$ if $\mathbf{u}_0 = \mathbf{b}$
- *key idea*: element of $\mathcal{K}_N(\mathbf{v})$ is a degree $N-1$ polynomial in A applied to $\mathbf{v}$

$$c_0\mathbf{v} + c_1 A\mathbf{v} + \cdots + c_{N-1}A^{N-1}\mathbf{v} = p_{N-1}(A)\mathbf{v}$$

where

$$p(z) = c_0 + c_1 z + \cdots + c_{N-1}z^{N-1}$$

Krylov methods

- iterative linear algebra
  =
  look for approximate solution of $A\mathbf{u} = \mathbf{b}$ from Krylov space
- examples:
  - Richardson ∼1910
  - CG = conjugate gradient ∼1955
  - GMRES = generalized minimum residual ∼1992
- many more:

```
$ ./vecmatksp -help |grep ksp_type
  -ksp_type <gmres>: Krylov method (one of) cg groppcg pipecg
cgne nash stcg gltr richardson chebyshev gmres tcqmr fcg bcgs
ibcgs fbcgs fbcgsr bcgsl cgs tfqmr cr pipecr lsqr preonly qcg
bicg fgmres minres symmlq lgmres lcd gcr pgmres dgmres tsirm
cgls (KSPSetType)
```

- "good" Krylov methods like CG and GMRES generate polynomial approximations which are optimal (in various senses)

code `vecmatksp.c`: runtime

- show using `-ksp_` options:

```
$ ./vecmatksp -help |grep ksp_
$ ./vecmatksp -help |grep ksp_type
$ ./vecmatksp -ksp_view
$ mpiexec -n 2 ./vecmatksp -ksp_view
$ ./vecmatksp -ksp_monitor
```

- unpreconditioned Richardson fails

```
$ ./vecmatksp -ksp_type richardson -pc_type none
```

- diagonal preconditioned Richardson . . . also fails

```
$ ./vecmatksp -ksp_type richardson -pc_type jacobi
```

- direct linear algebra by turning-off the Krylov method and setting the preconditioner to direct method:

```
$ ./vecmatksp -ksp_type preonly -pc_type lu
```

arbitrary-size linear system

- look at `c/ch2/tri.c`
  - `PetscOptions...()`
  - `MatGetOwnershipRange(A,&Istart,&Iend)`
  - generic row of A is

$$-1 \quad 3 \quad -1$$

  - "manufacture" exact solution: `MatMult(A,xexact,b)`
- performance = execution time (for now)

```
$ time ./tri -tri_m 10000
$ timer ./tri -tri_m 10000
$ alias timer
```

preformance

- try:

```
$ timer mpiexec -n N ./tri -tri_m 20000000 \
  -ksp_rtol 1.0e-10 -ksp_type KSP -pc_type PC
```

- get

KSP	PC	N=1 time (s)	N=4 time (s)
preonly	lu	10.74	
	cholesky	5.84	
richardson	jacobi	13.48	5.45
gmres	none	9.99	5.30
	jacobi	10.23	4.49
	ilu	4.77	
	bjacobi+ilu		2.99
cg	none	7.22	3.18
	jacobi	7.49	3.31
	icc	4.81	
	bjacobi+icc		2.87

Table 2.2: Times for `tri.c` to solve systems of dimension $m = 2 \times 10^7$. In this case the matrix is *tridiagonal*, *symmetric*, *diagonally-dominant*, and *positive definite*. All runs were on WORKSTATION (see page 41).