Numerical Methods I: Linear solvers and least squares

Georg Stadler
Courant Institute, NYU
stadler@cims.nyu.edu

Sep. 24, 2015
Main reference:
Sections 8 and 3 in Deuflhard/Hohmann.
Solving linear systems

We study the solution of linear systems of the form

$$Ax = b$$

with $A \in \mathbb{R}^{n \times n}$, $x, b \in \mathbb{R}^n$. We assume that this system has a unique solution, i.e., $A$ is invertible.

Solving linear systems is needed in many applications. Often, we have to solve

- large systems (can be up to millions of unknowns, and more)
- as fast as possible, and
- accurately and reliably.
Kinds of linear systems

Solvers such as MATLAB’s \ take advantage of matrix properties:

- **Dense matrix storage:** Only entries are stored as 1D array (column or row wise)
- **Sparse matrix storage:** Most $a_{ij} = 0$: only store nonzero entries; stores indices and value; occur in many applications
Kinds of linear systems

Solvers such as MATLAB’s take advantage of matrix properties:

- **Dense matrix storage:** Only entries are stored as 1D array (column or row wise)
- **Sparse matrix storage:** Most $a_{ij} = 0$: only store nonzero entries; stores indices and value; occur in many applications

Matrix application/system solution:

- **Dense:** the best way to compute $Ax$ is direct summation
- **Fast algorithms for special matrices** for computing $Ax$, FFT, FMM, ...
- **Sparse:** Most $a_{ij} = 0$: avoid fill in in factorizations
- **Structured/unstructured:** is the sparsity pattern easy to describe without storing it explicitly?
Kinds of linear systems and solvers

Symmetry, positivity . . .

- Special factorizations for (skew) symmetric matrices
- Special factorizations for positive definite matrices (Choleski)
- Diagonally dominant matrices don’t need pivoting

MATLAB’s \ (i.e., UMFPACK) chooses the optimal algorithms after studying properties of the matrix (details in the “backslash” book: Tim Davis: *Direct methods for sparse linear systems*, SIAM, 2006.)
Kinds of linear systems and solvers

UMFPACK’s decision tree for dense matrices
Kinds of linear systems and solvers

UMFPACK's decision tree for dense matrices
Kinds of linear systems and solvers

UMFPACK’s decision tree for sparse matrices

- Is A square?
  - Yes: Compute the bandwidth of A
  - No: Use QR solver

- Is A diagonal?
  - Yes: Use diagonal solver
  - No: Does A look triangular? (Upper or lower bandwidth of 0)
    - Yes: Use tridiagonal solver
    - No: Is A tridiagonal?
      - Yes: Use tridiagonal solver
      - No: Use QR solver
Kinds of linear systems and solvers

UMFPACK's decision tree for sparse matrices

1. Use banded solver
2. Is the band density of A > banded density threshold? NO
3. Is A actually triangular? (diagonal is numerically nonzero?) NO
   - Is A permuted triangular? YES
     - Use permuted triangular solver
     - Use LU solver
   - Use Cholesky solver
   - Use LDL solver
   - Use band solver
4. Does A have a real and positive diagonal? YES
5. Is A Hermitian? YES
   - Use LDL solver
   - Use LU solver
   - Use band solver
6. Does Cholesky succeed? YES
   - Use Cholesky solver
   - Use LDL solver
   - Use band solver
7. Is A real? YES
   - Use LDL solver
   - Use LU solver
   - Use band solver
8. Use triangular solver
Kinds of linear systems and solvers

**Factorization-based/direct solvers** (dense/sparse LU, Choleski) require the matrix
- to fit into memory,
- to be explicitly available (sometimes only a function that applies the matrix to a vector is available) and to fit in memory,
+ but compute exact (besides rounding error) solution

**Iterative solvers**
- find an \( \varepsilon \)-approximation of the solution,
+ able to solve very large problems,
+ often only require a function that computes \( Ax \) for given \( x \)
\( \pm \) might be faster or slower than a factorization-based method
Kinds of linear systems and solvers
MATLAB demo

- What are the different storage formats (sparse/dense)? Is it always better to use one of them?
- How long does it take to solve sparse/dense systems?
- What is fill in and how to avoid it?
Kinds of linear systems and solvers
MATLAB demo

**Sparse/sense storage:**
A=rand(2,2);
B=sparse(A);
whos

**Fill-in:**
A=bucky + 4*speye(60);
r = symrcm(A);
spy(A); spy(A(r,r)); spy(chol(A)); spy(chol(A(r,r)));

**Which sparse solver?**
spparms('spumoni',1);
A=gallery('poisson',8);
b=randn(64,1);
A\b;
Iterative solution of (symmetric) linear systems

Target problems: very large \((n = 10^5, 10^6, \ldots)\), \(A\) is usually sparse has specific properties.
To solve

\[ Ax = b \]

we construct a sequence

\[ x_1, x_2, \ldots \]

of iterates that converges as fast as possible to the solution \(x\), where \(x_{k+1}\) can be computed from \(\{x_1, \ldots, x_k\}\) with as little cost as possible (e.g., one matrix-vector multiplication).
Iterative solution of (symmetric) linear systems

Let $Q$ be invertible, then

$$Ax = b \iff Q^{-1}(b - Ax) = 0$$

$$\iff (I - Q^{-1}A)x + Q^{-1}b = x$$

$$\iff Gx + c = x$$

**Theorem:** The fixed point method $x_{k+1} = Gx_k + c$ with an invertible $G$ converges for each starting point $x_o$ if and only if

$$\rho(G) < 1,$$

where $\rho(G)$ is the largest eigenvalue of $G$ (i.e., the spectral radius).
Iterative solution of (symmetric) linear systems

Choices for $Q$:
- $Q = I \ldots$ Richardson method

Consider $A = L + D + U$, where $D$ is diagonal, $L$ and $U$ are lower and upper triangular with zero diagonal. Then:
- $Q = D \ldots$ Jacobi method
- $Q = D + L \ldots$ Gauss-Seidel method

Convergence depends on properties of $A$: Richardson converges if all eigenvalues of $A$ are in $(0, 2)$, Jacobi converges for diagonally dominant matrices, and Gauss Seidel for spd matrices.
Relaxation methods: Use linear combination between new and previous iterate:

\[ x_{k+1} = \omega (G x_k + c) + (1 - \omega) x_k = G_\omega x_k + \omega c, \]

where \( \omega \in [0, 1] \) is a damping parameter. Target is to choose the best damping parameter such that \( \rho(G_\omega) \) is as small as possible. Optimal damping parameters can be computed for Richardson and Jacobi using the eigenvalues of \( G \) (see Deuflhard/Hohmann).
Iterative solution of (symmetric) linear systems

Chebyshev acceleration

So far, the new iterate $x_{k+1}$ only depended on $x_k$. This can be improved by using all previous iterates when computing $x_{k+1}$.

The resulting schemes are called **Chebyshev accelerated methods**, and they usually converge faster than the original iterative schemes.

Chebyshev methods are based on computing linear combinations

$$y_k := \sum_{j=0}^{k} v_{k,j} x_j$$

with suitable coefficients $v_{k,j}$ such that $y_0, y_1, \ldots$ converges faster than $x_0, x_1, \ldots$ Computation of coefficient requires knowledge of the eigenvalues of $G$. 


Iterative solution of (symmetric) linear systems

Krylov methods:
Idea: Build a basis for the Krylov subspace \( \{ r_0, Ar_0, A^2r_0 \ldots \} \) and reduce residual optimally in that space.

- spd matrices: Conjugate gradient (CG) method
- symmetric matrices: Minimal residual method (MINRES)
- general matrices: Generalized residual method (GMRES), BiCG, BiCGSTAB
Iterative solution of (symmetric) linear systems

Krylov methods:
Idea: Build a basis for the Krylov subspace \( \{r_0, Ar_0, A^2r_0 \ldots \} \) and reduce residual optimally in that space.

- spd matrices: Conjugate gradient (CG) method
- symmetric matrices: Minimal residual method (MINRES)
- general matrices: Generalized residual method (GMRES), BiCG, BiCGSTAB

Properties:
Do not require eigenvalue estimates; require usually one matrix-vector multiplication per iteration; convergence depends on eigenvalue structure of matrix (clustering of eigenvalues aids convergence). Availability of a good preconditioner is often important. Some methods require storage of iteration vectors.
Least-squares problems

Given data points/measurements

\[(t_i, b_i), \quad i = 1, \ldots, m\]

and a model function \(\phi\) that relates \(t\) and \(b\):

\[b = \phi(t; x_1, \ldots, x_n),\]

where \(x_1, \ldots, x_n\) are model function parameters. If the model is supposed to describe the data, the deviations/errors

\[\Delta_i = b_i - \phi(t_i, x_1, \ldots, x_n)\]

should be small. Thus, to fit the model to the measurements, one must choose \(x_1, \ldots, x_n\) appropriately.
Least-squares problems

Measuring deviations

Least squares: Find $x_1, \ldots, x_n$ such that

$$\frac{1}{2} \sum_{i=1}^{m} \Delta_i^2 \rightarrow \min$$

From a probabilistic perspective, this corresponds to an underlying
Gaussian noise model.

Weighted least squares: Find $x_1, \ldots, x_n$ such that

$$\frac{1}{2} \sum_{i=1}^{m} \left( \frac{\Delta_i}{\delta b_i} \right)^2 \rightarrow \min,$$

where $\delta b_i > 0$ contain information about how much we trust the
$i$th data point.
Least-squares problems
Measuring deviations

Alternatives to using squares:

$L^1$ error: Find $x_1, \ldots, x_n$ such that

$$\sum_{i=1}^{m} |\Delta_i| \rightarrow \min$$

Result can be very different, other statistical interpretation, more stable with respect to outliers.

$L^\infty$ error: Find $x_1, \ldots, x_n$ such that

$$\max_{1 \leq i \leq m} |\Delta_i| \rightarrow \min$$
Linear least-squares

We assume (for now) that the model depends linearly on $x_1, \ldots, x_n$, e.g.:

$$\phi(t; x_1, \ldots x_n) = a_1(t)x_1 + \ldots + a_n(t)x_n$$

Choosing the least square error, this results in

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|,$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$, $\mathbf{b} = (b_1, \ldots, b_m)^T$, and $a_{ij} = a_j(t_i)$.

In the following, we study the overdetermined case, i.e., $m \geq n$. 


Linear least-squares problems–QR factorization

Consider non-square matrices $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and rank($A$) = $n$. Then the system

$$Ax = b$$

does, in general, not have a solution (more equations than unknowns). We thus instead solve a minimization problem

$$\min_{x} \|Ax - b\|^2.$$ 

The minimum $\bar{x}$ of this optimization problem is characterized by the normal equations:

$$A^T A \bar{x} = A^T b.$$
Solving the normal equations

\[ A^T A \bar{x} = A^T \mathbf{b} \]

requires:

- computing \( A^T A \) (which is \( O(mn^2) \))
- condition number of \( A^T A \) is square of condition number of \( A \); (problematic for the Choleski factorization)
Solving the normal equation is equivalent to computing $Pb$, the orthogonal projection of $b$ onto the subspace $V$ spanned by columns of $A$.

Let $x$ be the solution of the least square problem and denote the residual by $r = b - Ax$, and

$$\sin(\theta) = \frac{\|r\|_2}{\|b\|_2}.$$
Linear least-squares problems–QR factorization

Conditioning

The relative condition number $\kappa$ of $x$ in the Euclidean norm is bounded by

- With respect to perturbations in $b$:
  \[ \kappa \leq \frac{\kappa_2(A)}{\cos(\theta)} \]

- With respect to perturbations in $A$:
  \[ \kappa \leq \kappa_2(A) + \kappa_2(A)^2 \tan(\theta) \]

Small residual problems $\cos(\theta) \approx 1$, $\tan(\theta) \approx 0$: behavior similar to linear system.

Large residual problems $\cos(\theta) \ll 1$, $\tan(\theta) > 1$: behavior essentially different from linear system.
One would like to avoid the multiplication $A^T A$ and use a suitable factorization of $A$ that aids in solving the normal equation, the QR-factorization:

$$A = QR = [Q_1, Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

where $Q \in \mathbb{R}^{m \times m}$ is an orthonormal matrix ($QQ^T = I$), and $R \in \mathbb{R}^{m \times n}$ consists of an upper triangular matrix and a block of zeros.
How can the $QR$ factorization be used to solve the normal equation?

$$\min_x \| A x - b \|^2 = \min_x \| Q^T (A x - b) \|^2 = \min_x \| \begin{bmatrix} b_1 - R_1 x \\ b_2 \end{bmatrix} \|^2,$$

where $Q^T b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$.

Thus, the least squares solution is $x = R^{-1} b_1$ and the residual is $\| b_2 \|$. 
How can we compute the QR factorization?

**Givens rotations**
Use sequence of rotations in 2D subspaces:
For $m \approx n$: $\sim n^2/2$ square roots, and $4/3n^3$ multiplications
For $m \gg n$: $\sim nm$ square roots, and $2mn^2$ multiplications

**Householder reflections**
Use sequence of reflections in 2D subspaces
For $m \approx n$: $2/3n^3$ multiplications
For $m \gg n$: $2mn^2$ multiplications