Tridiagonal matrices If no pivoting is needed, then solving systems with a tridiagonal matrix is very efficient ($\mathcal{O}(n)$ flops)

Here is an example of such a system:

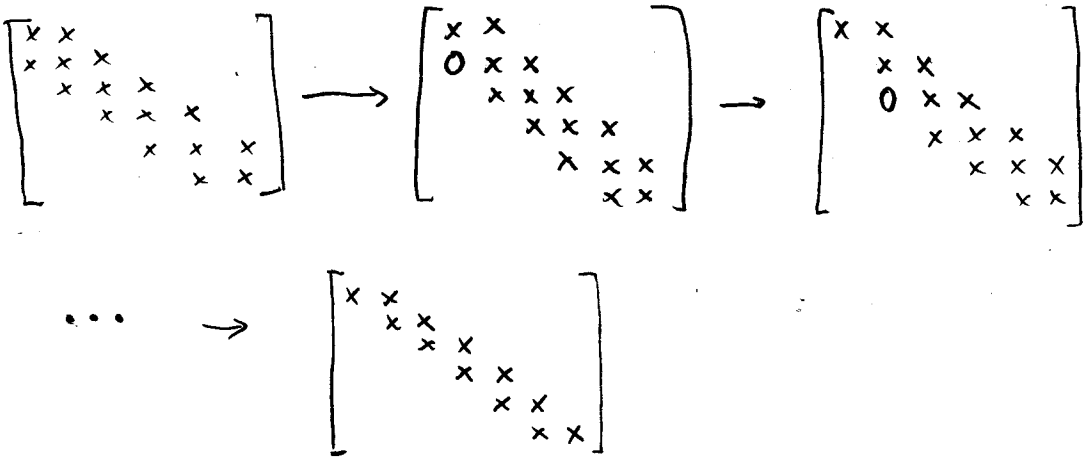
$$\begin{bmatrix} d_1 & c_1 & & & \\ a_1 & d_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & c_{n-1} & \\ & & a_{n-1} & d_n & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \\ b_n \end{bmatrix}$$

In matlab we can write A using vectors a, d, c:

```
A = diag(a,-1) + diag(d,0) + diag(c,1);
```

In real life we only need to store 3 vectors.

Here is how Gaussian elimination looks like:



Introducing zeros below first equation:

$$d_2 = d_2 - \frac{a_1 c_1}{d_1}$$

$$b_2 = b_2 - \frac{a_1 b_1}{d_1}$$

(note: here we carry operations for L at the same time on RHS we are not computing factor L !!)

for $i = 2, \ldots, n$

$$d_i = d_i - \frac{a_{i-1} c_{i-1}}{d_{i-1}}$$

$$b_i = b_i - \frac{a_{i-1} b_{i-1}}{d_{i-1}}$$

(update formulas have only one term)

Then to find x we do back substitution, which also simplifies:

$$x_n = b_n / d_n$$

for $i = n-1 : -1 : 1$

$$x_i = (b_i - c_i x_{i+1}) / d_i$$

Similar algorithms exist for <u>band matrices</u> (i.e. only a few sub, super diagonals are non zero)

DIRECT SPARSE SOLVERS (brief intro)

A *sparse matrix* is a matrix with most of its entries being zero.

- saves memory as only *non-zero* elements are stored.
- saves also computation time as algorithms need only to access non-zero elements.

Here are two common ways of storing a sparse matrix

Coordinate format

iA = row index of nz entries
jA = column " " " "
vA = values of nz entries.

$$a_{iA(k),jA(k)} = vA(k)$$

Compressed sparse column format

pA = pointer to all nz elements in a column.
iA = row indices of nz entries in column
vA = values " " " " "

Example:
$$A = \begin{bmatrix} 4 & 0 & 3 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 2 & 3 & 0 \\ 5 & 3 & 0 & 6 \end{bmatrix}$$

Coordinate

iA	jA	vA
1	1	4
4	1	5
2	2	1
3	2	2
4	2	3
1	3	3
3	3	3
2	4	1
4	4	6

CSC

pA	iA	vA
1 →	1	4
3 →	4	5
	2	1
6 →	3	2
8 →	4	3
10	1	3
	3	3
	2	1
	4	6

Some popular direct sparse solvers:

SuperLU (Lawrence Berkeley National Lab)

MUMPS (French group: CERFACS, CNRS, INPT, INRIA)

UMFPACK (University of Florida, what is used by Matlab's backslash for solving sparse systems)

...

These are sophisticated libraries that can use a graph representation of a matrix to run the LU (or Cholesky) factorization symbolically and automatically switch between sparse and dense algorithms.

We are not going to see these algorithms in detail. Consider only the following examples to see how sparse structure reduces amount of computations. (these come from http://www.cise.ufl.edu/~davis the creator of UMFPACK homepage)

Matrix vector product: $y = Ax$

Dense version:

```
y = 0
for j = 1..n
    for i = 1..n
        yi = yi + aij xj
```

$O(n^2)$

Sparse version:

```
y = 0
for j = 1..n
    for each i for which aij ≠ 0:
        yi = yi + aij xj
```

$O(n \times (\text{nz in a column}))$

Triangular solve: $Lx = b$ w.o.l.g: $L_{ii} = 1$ (unit L.T.)

$(\Delta)|=1$

Dense version:

$x = b$
for $j = 1 \ldots n$
  for $i = j+1 \ldots n$
    $x_i = x_i - l_{ij} x_j$

$O(n^2)$

Sparse version #1

$x = b$
for $j = 1 \ldots n$
  if $x_j \neq 0$
    for each $i > j$ with $l_{ij} \neq 0$
      $x_i = x_i - l_{ij} x_j$

$O(n + |b|)$

Can be made even more efficient if we know in advance the sparsity pattern of x.

let $\mathcal{X} = \{ i \mid x_i \neq 0 \}$ ≡ sparsity pattern of sol x.

$\mathcal{B} = \{ i \mid b_i \neq 0 \}$ ≡ sparsity pattern of RHS b.

If we knew $\mathcal{X}$ then alg. becomes:

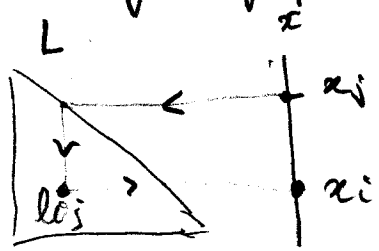
Sparse version #2

$x = b$
for j in $\mathcal{X}$
  for each $i > j$ w/ $l_{ij} \neq 0$
    $x_i = x_i - l_{ij} x_j$

$O(|b|)$ flops

(which can be better than version 1).

The idea to obtain $\mathcal{X}$ is to run triangular solve symbolically in graph representing L.



① $b_j \neq 0 \Rightarrow x_j \neq 0$

② $x_j \neq 0$ and $l_{ij} \neq 0 \Rightarrow x_i \neq 0$

→ Replace L by an oriented graph with an edge $j \to i$ if $l_{ij} \neq 0$.

→ $\mathcal{X} \equiv$ closure (i.e. fixed point) of neighbor (②)

The enemy of sparse direct solvers is <u>fill-in</u>

<u>For example:</u>
A sparse matrix may not necessarily have a sparse Cholesky factor L.

→ See Sparse matrices Matlab demo.

→ See also: "Direct Methods for Sparse Linear Systems"
T. Davis
→ Has sample code which is simple enough to read.

Norms and the analysis of errors (§7.1 in textbook)

How are we sure that after doing $O(n^3)$ operations to solve a linear system using LU factor + backward substitution that errors we make don't pile up?

~> we need notion of error in $\mathbb{R}^n$.

Def (norm) $\|x\|$ is a norm if

i) $\|x\| \geq 0 \quad \forall x \in \mathbb{R}^n$

ii) $\|x\| = 0 \Rightarrow x = 0$

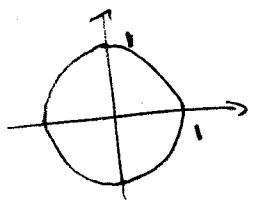
iii) $\|\lambda x\| = |\lambda| \|x\|$, $\forall x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$

iv) $\|x + y\| \leq \|x\| + \|y\|$ (triangle ineq)

Examples:

Euclidean norm

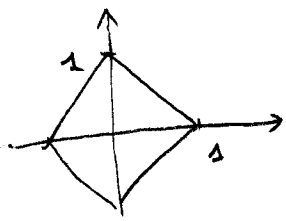
$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$



unit ball.

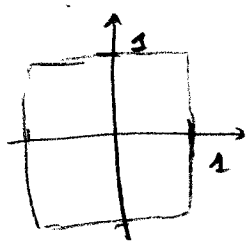
ℓ_1 norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$



ℓ_∞ norm

$$\|x\|_\infty = \max_{i=1..n} |x_i|$$



Induced matrix norm

Let $\|\cdot\|$ be a norm in $\mathbb{R}^n$ then we defined the induced matrix norm

$$\|A\| = \sup_{\|u\|=1} \|Au\| = \sup_{u \neq 0} \frac{\|Au\|}{\|u\|}$$

$\|\cdot\|$ satisfies all axioms of a norm and:

$$\|I\| = 1$$
$$\|AB\| \leq \|A\| \|B\|$$

Examples:

- $$\|A\|_2^2 = \sup_{x \neq 0} \frac{\|Ax\|^2}{\|x\|^2} = \sup_{x \neq 0} \frac{x^T A^T A x}{x^T x}$$
$$= \lambda_1(A^T A) = \text{largest eigenvalue of } A^T A.$$
$$\Rightarrow \boxed{\|A\|_2 = \sqrt{\lambda_1(A^T A)}}$$

(recall λ is an eigenvalue of matrix B if there exists $u \neq 0$ s.t. $Bu = \lambda u$.)

- $$\|A\|_\infty = \sup_{\|u\|_\infty = 1} \|Au\|_\infty$$
$$= \sup_{\|u\|_\infty = 1} \max_{1 \leq i \leq n} |(Au)_i|$$
$$= \sup_{\|u\|_\infty = 1} \max_{1 \leq i \leq n} \sum_j |a_{ij} u_j|$$
$$= \max_{1 \leq i \leq n} \sup_{\|u\|_\infty = 1} \|$$
$$= \max_{1 \leq i \leq n} \sum_j |a_{ij}| = \max_{1 \leq i \leq n} \ell_1\text{norm}(i\text{-th row of } A)$$

← sup attained when $u_j = \text{sgn } a_{ij}$

Condition number

Suppose $\tilde{b}$ is a perturbation of b.

If $x, \tilde{x}$ satisfy:

$$Ax = b$$
$$A\tilde{x} = \tilde{b}$$

then how close are x and $\tilde{x}$?

$$\|x - \tilde{x}\| = \|A^{-1}b - A^{-1}\tilde{b}\| = \|A^{-1}(b - \tilde{b})\|$$
$$\leq \|A^{-1}\| \, \|b - \tilde{b}\|.$$

We know giving a relative error is more informative so here it is.

$$\|x - \tilde{x}\| \leq \|A^{-1}\| \, \|b - \tilde{b}\| = \|A^{-1}\| \, \frac{\|Ax\|}{\|b\|} \, \|b - \tilde{b}\|$$

$$\leq \|A^{-1}\| \, \|A\| \, \|x\| \, \frac{\|b - \tilde{b}\|}{\|b\|}$$

$$\boxed{\frac{\|x - \tilde{x}\|}{\|x\|} \leq K(A) \, \frac{\|b - \tilde{b}\|}{\|b\|}}$$

Here $K(A)$ = condition number of A

$= \|A\| \, \|A^{-1}\|$ (depends on choice of norm usually $\|\cdot\|_2$ is used)

≥ 1

⇝ quantifies how much we can trust solution x if there are errors in RHS.

⇝ There are ways of estimating $K(A)$ without computing A^{-1} (in a fast way)

Example:

$$A = \begin{bmatrix} 1 & 1+\varepsilon \\ 1-\varepsilon & 1 \end{bmatrix} \qquad A^{-1} = \varepsilon^{-2} \begin{bmatrix} 1 & -1-\varepsilon \\ -1+\varepsilon & 1 \end{bmatrix}$$

$$\|A\|_\infty = 2+\varepsilon \qquad \|A^{-1}\|_\infty = \varepsilon^{-2}(2+\varepsilon)$$

$$\Rightarrow \quad \mathcal{K}(A) = \frac{(2+\varepsilon)^2}{\varepsilon^2}.$$

If $\varepsilon \leq 0.01$ then $\mathcal{K}(A) \geq 40000$.

$\rightsquigarrow$ we loose 4 digits of precision!

As a rule of thumb:

$$\log_{10}(\mathcal{K}(A)) = \#\text{ of digits of precision lost in solution.}$$

Example: $\dfrac{\|b-\tilde{b}\|}{\|b\|} \approx 10^{-16}$ (Machine precision)

$$\mathcal{K}(A) \approx 10^{10}$$

$\rightsquigarrow$ $\dfrac{\|x-\tilde{x}\|}{\|x\|} \lesssim 10^{-6}$ lost 10 digits of precision!

Convergence in $\mathbb{R}^n$

We say a sequence of vectors $v^{(k)} \in \mathbb{R}^n$ converges to $v \in \mathbb{R}^n$ as $k \to \infty$ if

$$\lim_{k\to\infty} \|v^{(k)} - v\| = 0$$

In $\mathbb{R}^n$ the choice of norm does not matter as all <u>norms are equivalent</u>: (this is true for any finite dimensional space)

Two norms $\|\cdot\|$ and $\|\cdot\|$ are equivalent if there are two constants c_1 and $c_2 > 0$ s.t.

$$c_1\|x\| < \|x\| \leq c_2\|x\|.$$

Also $\mathbb{R}^n$ is <u>complete</u> meaning any sequence satisfying the Cauchy criterion converges.

$$\forall \varepsilon > 0 \quad \exists N \text{ s.t. } \forall i,j \geq N \quad \|v^{(i)} - v^{(j)}\| < \varepsilon$$

Neumann Series

Let $A \in \mathbb{R}^{n\times n}$ be such that $\|A\| < 1$, then

i) $I - A$ is invertible

ii) $(I-A)^{-1} = \sum_{k=0}^{\infty} A^k$

<u>Proof</u>: Assume for contradiction that $I - A$ is singular

$\Rightarrow \exists x \in \mathbb{R}^n$ s.t. $(I-A)x = 0$, take w.l.o.g. x s.t. $\|x\| = 1$

then:

$$1 = \|x\| = \|Ax\| \leq \underbrace{\|A\|}_{<1}\|x\| < \|x\| = 1. \quad \text{contradiction!}$$

Now we need to show that

$$\sum_{k=0}^{m} A^k \to (I-A)^{-1} \quad \text{(i.e. partial sums converge)}$$

Notice: (telescoping series)

$$(I-A)\sum_{k=0}^{m} A^k = \sum_{k=0}^{m} A^k - A^{k+1} = A^0 - A^{m+1}$$
$$= I - A^{m+1}$$

Now using properties of induced matrix norm: ($\|AB\| \le \|A\|\|B\|$)

$$\|A^{m+1}\| \le \|A\|^m \to 0$$

Thus:

$$\left\| (I-A)\sum_{k=0}^{m} A^k - I \right\| \le \|A\|^{m+1} \to 0 \text{ as } m \to \infty.$$

QED

Note: This theorem is a very intuitive generalization of what you already know about geometric series.

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots, \text{ when } |x| < 1.$$

Here is the kind of result we can show with Neumann Series:

$$\|(I-A)^{-1}\| \le \sum_{k=0}^{\infty} \|A^k\| \le \sum_{k=0}^{\infty} \|A\|^k = \frac{1}{1-\|A\|}.$$

Note: This theorem is very useful for the theory, however it is seldom used in practice. (There are faster iterative methods for solving $Ax = b$).

Here is an easy but handy generalization of Neumann Series:

Theorem

If A and B are matrices s.t. $\|I - AB\| < 1$ then A, B are invertible and

$$A^{-1} = B \sum_{k=0}^{\infty} (I - AB)^k$$

$$B^{-1} = \left[\sum_{k=0}^{\infty} (I - AB)^k \right] A$$

Proof: By Neumann series:

$I - (I - AB) = AB$ is invertible and:

$$(AB)^{-1} = \sum_{k=0}^{\infty} (I - AB)^k$$

$$\Rightarrow \quad A^{-1} = B B^{-1} A^{-1} = B \sum_{k=0}^{\infty} (I - AB)^k$$

$$B^{-1} = B^{-1} A^{-1} A = \left[\sum_{k=0}^{\infty} (I - AB)^k \right] A$$

Iterative refinement

Let $x^{(0)}$ be an approx. sol to

$$Ax = b.$$

then:

$$x = x^{(0)} + A^{-1}(b - Ax^{(0)})$$

We often call

$r^{(0)} = b - Ax^{(0)}$ = residual vector

$e^{(0)} = A^{-1} r^{(0)}$ = error vector.

Algorithm

$x^{(0)}$ given

for $k = 0, 1, \ldots$

$$r^{(k)} = b - Ax^{(k)}$$

$$Ae^{(k)} = r^{(k)} \quad \leftarrow \text{usually inexact solve}$$

$$x^{(k+1)} = x^{(k)} + e^{(k)}$$

Solution can be improved even if the solves are inexact for example:

→ LU factorization is stopped prematurely (= Incomplete LU fact.)

→ $Ae^{(k)} = r^{(k)}$ is solved with an iterative method

→ A is not known exactly.

To be more precise about what we mean by "solving approximatively"

Let $B \equiv$ "approximate" inverse of A

$$\approx A^{-1} \quad \text{(in some sense we shall see)}$$

The iterates are:

$$x^{(0)} = Bb$$
$$x^{(k+1)} = x^{(k)} + B(b - Ax^{(k)}), \quad k \geq 0.$$

Looking at a few iterates we see a pattern emerge.

$$x^{(0)} = Bb$$
$$x^{(1)} = x^{(0)} + B(b - Ax^{(0)})$$
$$= Bb + B(b - ABb)$$
$$= Bb + B(I - AB)b$$
$$x^{(2)} = x^{(1)} + B(b - Ax^{(1)})$$
$$= Bb + B(I-AB)b + B(b - A(Bb + B(I-AB)b))$$
$$= Bb + B(I-AB)b + B((I-AB)b - AB(I-AB)b)$$
$$= Bb + B(I-AB)b + B(I-AB)^2 b$$
$$\vdots$$

$$\boxed{x^{(m)} = B\sum_{k=0}^{m} (I-AB)^k b}$$

Theorem (Iterative refinements)

Iterates from iterative refinement are:

$$x^{(m)} = B\sum_{k=0}^{m} (I-AB)^k b \quad (m \geq 0)$$

and because of the generalized Neuman series:

$$x^{(m)} \to x \text{ as } m \to \infty \text{ when } \|I - AB\| < 1.$$

Proof: Can be done by induction on m.

Solving equations with iterative methods

Idea: Instead of having a direct method which finds solution in finitely many steps, use an iterative method to find succesive approx that converge to sol.

Motivational example

$$\begin{bmatrix} 7 & -6 \\ -8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$

Jacobi method: Solve i-th equation for i-th unknown:

$$\begin{cases} x_1^{(k)} = \frac{3}{7} + \frac{6}{7} x_2^{(k-1)} \\ x_2^{(k)} = \frac{-4}{9} + \frac{8}{9} x_1^{(k-1)} \end{cases}$$

Gauss Seidel method: use improved value of $x_1^{(k)}$ in second equation.

$$\begin{cases} x_1^{(k)} = \frac{3}{7} + \frac{6}{7} x_2^{(k-1)} \\ x_2^{(k)} = \frac{-4}{9} + \frac{8}{9} x_1^{(k)} \end{cases}$$

converges faster!

Note: convergence for these methods depends on initial guess $(x_1^{(0)}, x_2^{(0)})$!

Matrix splitting

Consider the system $Ax = b$.

Introduce a non-singular matrix Q and rewrite system as:

$$Qx = (Q - A)x + b$$

Get iteration:

$$Q x^{(k)} = (Q - A) x^{(k-1)} + b, \quad k \geq 1, \quad x^{(0)} \text{ given}$$

Q should have the desirable properties:

① $x^{(k)} \to x$ where x is a sol. of $Ax = b$

② Solving systems with Q is cheap.

If $x^{(k)} \to x$ then the limit x must satisfy:

$$Qx = (Q - A)x + b \implies Ax = b$$

Assuming A & Q are non-singular:

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

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

$$x^{(k)} - x = (I - Q^{-1}A)(x^{(k-1)} - x)$$

$$\Rightarrow \|x^{(k)} - x\| \leq \|I - Q^{-1}A\| \, \|x^{(k-1)} - x\|$$

$$\vdots$$

$$\leq \|I - Q^{-1}A\|^k \, \|x^{(0)} - x\|$$

Thus when $\|I - Q^{-1}A\| < 1$ we have $x^{(k)} \to x$, where $Ax = b$.

Theorem If $\|I - Q^{-1}A\| < 1$ for some induced matrix norm then seq.

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b$$

converges to sol. of $Ax = b$ for any starting vector $x^{(0)}$