

Examples of solving symmetric linear systems

```
>> rng(0);
>> B = sprandsym(4000, 0.001);
>> A = speye(size(B)) - 0.1*B;
>> tic; R = chol(full(A)); toc
Elapsed time is 1.059630 seconds.
>> tic; R = chol(A); toc
Elapsed time is 0.910553 seconds.
>> nnz(R)
ans =
    2360808
>> nnz(A)
ans =
    19984
```

for a posdef matrix, chol(A)
lu(A)
qr(A)

$A = I - 0.1B$
if B has eigh. λ_i ,
 A has eigh.

$$1 - 0.1\lambda_i$$

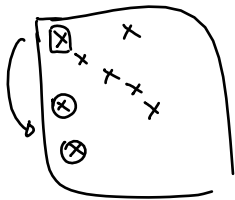
It's ≥ 0 if
 $|\lambda_i| \leq 10$ for all i

$$\frac{1}{3}n^3$$
$$\frac{2}{3}n^3$$
$$\frac{4}{3}n^3$$

$\text{chol}(A)$ returns $R = \begin{bmatrix} \text{diag} & \\ & \text{diag} \\ & & 0 \end{bmatrix}$ s.t.

$$A = R^T R$$

$$Ax = b \quad x = A^{-1}b = (R^T R)^{-1}b = R^{-1}(R^T)^{-1}b$$



Reordering entries to reduce bandwidth

The instruction `symrcm` reorders the entries of A according to a (sort of) BFS on its adjacency graph. This often reduces the bandwidth and the number of nonzeros in `chol(A)`.

```
>> p = symrcm(A);  
>> tic; R = chol(A(p, p)); toc  
Elapsed time is 0.374971 seconds.  
>> nnz(R)  
ans =  
    1576535  
>> spy(R)
```

All these factorizations can be used to solve linear systems, e.g.,

```
>> x = R \ (R' \ b(p));  
>> x(p) = x; %inverse permutation  
>> norm(A*x-b)  
ans =  
    7.8569e-14
```

$$P^T A P = R^T R \quad A = P R^T R P^T$$

↑ ↑
permut. matrix upper tri.

$$x = A^{-1} b = \underbrace{P R^{-1} R^T P^T}_{\text{inverse}} b$$

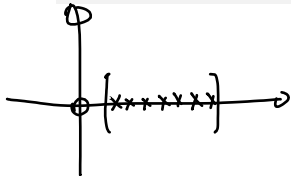
Iterative methods

The function `pcg` solves a system with conjugate gradient (without need for a factorization).

```
>> tic; [x, ~, ~, ~, resvec] = pcg(A, b); toc
Elapsed time is 0.014492 seconds.
>> norm(A*x-b)
ans =
    6.1736e-05
>> semilogy(resvec)
```

A is a reasonably well-conditioned matrix, and CG converges fast on it.

```
>> ev = eig(full(A)); plot(real(ev), imag(ev), 'x');
```



Don't use `inv`

Direct inversion with `inv` is not competitive by any metric.

```
>> tic; P = inv(A); toc
Elapsed time is 2.178261 seconds.
>> tic; x = P*b; toc
Elapsed time is 0.027226 seconds.
>> norm(A*x - b)
ans =
    1.4211e-13
```

A different example

This 'thin-band' matrix comes from discretization of a differential equation; it is more suitable to direct solvers than iterative ones.

```
[>> A = delsq(numgrid('S',50));  
>> size(A)  
ans =  
      2304 2304  
>> spy(A)  
>> b = randn(length(A), 1);  
>> tic; R = chol(A); toc;  
Elapsed time is 0.028630 seconds.  
>> pcg(A, b, 1e-8, 100);  
pcg stopped at iteration 100 without converging to the desired accuracy  
because the maximum number of iterations was reached.  
The iterate returned (number 100) has relative residual 8.1e-08.  
>> [x, ~, ~, ~, resvec] = pcg(A, b, 1e-8, 100); semilogy(resvec)
```

Preconditioning

A powerful idea: **preconditioning**. Since the performance of CG / GMRES on $Ax = b$ depends a lot on where the eigenvalues of A are located, we may replace $Ax = b$ with $PAx = Pb$ (or $(P^TAP)P^{-1}x = P^Tb$ to keep symmetry) to try to 'improve' its eigenvalues.

Among many possibilities: `ichol(A)` computes the 'incomplete Cholesky' factor, a sort of approximated Cholesky factorization which does not increase the number of nonzeros.

```
>> L = ichol(A);  
>> cond(A)  
ans =  
    1.4136e+03  
>> ev = cond(L \ A / L')  
ev =  
    2.0379e+02
```


$$Ax = b$$

$$PAx = Pb \quad \text{for nonsingular } P \text{ of choice}$$

$\neq \neq$ our choice

$$\underbrace{PAP^T}_{\text{symmetric and posdef}} (P^{-1}x) = Pb$$

Hopefully, the eigenvalues of PA (or PAP^T) are "better located" than those of A

$$\text{If } P = \underline{A^{-1}}, \quad PA = I$$

"Incomplete Cholesky"

Produces $R^T R \approx A$

$$R^{-T} A R^{-1} \approx I$$

$$\underbrace{R^{-T} A R^{-1}} \underbrace{R x}^u = R^{-T} b$$

$$z = R^{-T} A R^{-1} v \rightarrow$$

$$u = R x$$

1. $w = R^{-1} v$
2. $y = A w$
3. $z = R^{-T} y$

Using a preconditioner

We wish to solve $L^{-1}AL^{-T}(L^T x) = L^{-1}b$.

```
>> L = ichol(A);  
>> matvec = @(v) L \ (A*(L' \ v))  
matvec =  
    function_handle with value:  
    @(v)L\ (A*(L'\v))  
>> x = L' \ pcg(matvec, L \ b, 1e-8, 100);  
pcg converged at iteration 51 to a solution with relative r  
>> norm(A*x - b)  
ans =  
    2.9117e-07
```