Symmetric elimination

If A is a symmetric matrix, we can still use LU/QR, but there are more efficient methods.

Idea: choose L_1 as before, but now take $L_1 A L_1^T$. This is symmetric (check $(L_1 A L_1^T)^T$ and block upper triangular (because both $L_1 A$ and L_1^T are so):

$$\begin{bmatrix} 1 & & & & \\ * & 1 & & & \\ * & & 1 & & \\ * & & & 1 & \\ * & & & & 1 \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \begin{bmatrix} 1 & * & * & * & * \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} = \begin{bmatrix} * & 0 & 0 & 0 & 0 \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}$$

and continue in the same fashion:

$$\begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & * & 1 & & \\ & * & & 1 & \\ & * & & & 1 \end{bmatrix} \begin{bmatrix} * & 0 & 0 & 0 & 0 \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & 1 & * & * & * \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} = \begin{bmatrix} * & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}$$

Example

```
>> A = randn(5,5); A = A*A';
>> L1 = eye(size(A));
>> L1(2:end, 1) = -A(2:end, 1) / A(1, 1);
>> A1 = L1*A*L1'
A1 =
   4.3142e+00             0             0             0
             0   1.1666e+01  -5.0430e+00  -4.8709e+00  -2.59
             0  -5.0430e+00   7.0707e+00   6.0102e-01   7.07
             0  -4.8709e+00   6.0102e-01   1.5633e+01   9.56
             0  -2.5910e+00   7.0771e-01   9.5604e+00   6.08
```

Example

```
>> L2 = eye(size(A1));
>> L2(3:end, 2) = -A1(3:end, 2) / A1(2, 2);
>> L2 * A1 * L2'
ans =
   4.3142e+00            0            0            0
            0   1.1666e+01            0            0  -8.88
            0            0   4.8907e+00  -1.5045e+00  -4.12
            0            0  -1.5045e+00   1.3599e+01   8.47
            0            0  -4.1231e-01   8.4786e+00   5.51
```

LDL^T factorization

In the end, we get

$$L_{m-1}L_{m-2}\dots L_1 A L1^T \dots L_{m-2}^T L_{m-1}^T = D,$$

where D is diagonal, or

$$A = L_1 L_2 \dots L_{m-1} D L_{m-1}^T \dots L_2^T L_1^T = LDL^T.$$

Any symmetric matrix $A = A^T \in \mathbb{R}^{m\times m}$ for which we do not encounter zero pivots in the algorithm admits a factorization $A = LDL^T$, where L is lower triangular with ones on its diagonal, and D is diagonal.

Formulas and symmetry

$$\begin{bmatrix} 1 & \\ w & I \end{bmatrix} \begin{bmatrix} \alpha & a^T \\ a^T & \widehat{A} \end{bmatrix} \begin{bmatrix} 1 & w^T \\ & I \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & B \end{bmatrix},$$

$$w = -\tfrac{1}{\alpha} a, \quad B = \widehat{A} - wa^T = \widehat{A} - w\tfrac{1}{\alpha} w^T.$$

From the last expression, we see that B is symmetric (we did not prove it earlier...).

```
function [L, D] = ldl_factorization(A)
m = size(A, 1);
L = eye(m); D = zeros(m);
for k = 1:m-1
        D(k, k) = A(k, k);
        L(k+1:end, k) = A(k+1:end, k) / A(k, k);
        A(k+1:end, k+1:end) = A(k+1:end, k+1:end) ...
                    - L(k+1:end, k) * A(k, k+1:end);
end
D(m, m) = A(m, m);
```

LDL - details

Cost $\frac{1}{3}m^3 + O(m^2)$, half as much as LU — provided we compute only half of the entries and fill the rest in by symmetry (our implementation above doesn't).

Stable? Absolutely not, unless we do some form of pivoting. Pivoting must be symmetric, too: PAP^T.

Matlab's `[L, D, P] = ldl(A)` produces matrices such that $P^T AP = LDL^T$, where D may have 2×2 diagonal blocks.

`[M, D] = ldl(A)` returns $M = PL$ and D (so that $A = MDM^T$).

Positive definite factorization

However, things work better for positive definite matrices.

Lemma

- $A = A^T \in \mathbb{R}^{m \times m}$ is positive definite if and only if MAM^T is so, for some invertible $M \in \mathbb{R}^{m \times m}$.
- If $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ is symmetric positive definite, then A_{11} and A_{22} are, too.

(Proof: use the definition $z^T Az > 0$, and for the second bullet take $z = \begin{bmatrix} z_1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ z_2 \end{bmatrix}$).

Using this result, one can prove that at each step $D_{kk} > 0$.

Cholesky factorization

For positive definite matrices, usually a slightly different form is used (Cholesky factorization):

$$A = LDL^T = LD^{1/2}(D^{1/2\,T}L^T) = CC^T,$$

where $D^{1/2} = \mathrm{diag}(D_{11}^{1/2}, D_{22}^{1/2}, \ldots, D_{mm}^{1/2})$, and C is lower triangular (but not anymore with ones on the diagonal).

Matlab's `Ct = chol(A)` returns C^T (upper triangular).

We won't discuss stability further, but Cholesky is always backward stable even without pivoting ($\|C\| = \|A\|^{1/2}$, see exercise).

In a sparse matrix, we can choose the (symmetric) permutation with the only goal of reducing fill-in.

Summary

- Another weapon in our arsenal: symmetric variants of Gaussian elimination.
- Reduce cost and storage by 1/2.
- Again, we can use them to solve linear systems, e.g.,
  `[L, D] = ldl(A); L' \ ((L\b) ./ diag(D));`

Exercises

- Let C be the Cholesky factor of a positive definite $A \in \mathbb{R}^{m \times m}$. Show that $\|C\| = \|A\|^{1/2}$. Hint: use the SVD of C.
- Implement Cholesky factorization.
- Add a form of symmetric pivoting to your implementation of Cholesky factorization: at each step, find $\max((A_{k-1})_{kk}, (A_{k-1})_{k+1,k+1}, \ldots, (A_{k-1})_{mm})$, and bring it in position (k, k) by swapping rows and columns.