

MATH 2200: PLU decomposition / Conjugate Gradient

A/Prof Geoffrey Goodhill, Semester 2, 2008

Applications of the PLU decomposition

The decomposition $A = P'LU$ is especially useful for computing determinants and inverses (note $P' = P^{-1}$ where P' is the transpose of P).

Determinants

In general $\det(AB) = \det(A)\det(B)$, so

$$\det(A) = \det(P')\det(L)\det(U)$$

The 3 determinants on the rhs are now trivial to calculate:

- For a permutation matrix $\det(P) = (-1)^\sigma$, where σ is the number of swops in P .
- The determinant of a triangular matrix is the product of the elements on the leading diagonal (so $\det(L) = 1$).

Inverses

We want to find X such that $AX = I$, or equivalently $A\underline{x}_j = \underline{e}_j$, where \underline{x}_j is the j th column of X and \underline{e}_j is the j th column of the identity matrix. Write this as

$$P'LU\underline{x}_j = \underline{e}_j, \quad \text{or} \quad LU\underline{x}_j = P\underline{e}_j$$

This is equivalent to solving

$$L\underline{y}_j = P\underline{e}_j, \quad U\underline{x}_j = \underline{y}_j$$

But the first equation is trivially solved by forward substitution, and the second equation is trivially solved by backward substitution. This is faster and less susceptible to round-off error than calculating A^{-1} directly.

Conjugate Gradient

In the Conjugate Gradient method the problem is expressed as one of minimization. Performing gradient descent in this case ensures that each iteration brings one closer to the solution, and that the solution will be reached in n steps. It only applies to symmetric matrices which are also positive definite (i.e. $\underline{x}'A\underline{x} > 0$ for all $\underline{x} \neq 0$).

Let $\langle \underline{a}, \underline{b} \rangle$ denote the inner product of \underline{a} and \underline{b} . It can be shown that \underline{x}^* is a solution to the positive definite linear system $A\underline{x} = \underline{b}$ (of size $n \times n$) iff \underline{x}^* minimizes

$$g(\underline{x}) = \langle \underline{x}, A\underline{x} \rangle - 2 \langle \underline{x}, \underline{b} \rangle$$

We now choose an initial guess \underline{x} which has residual \underline{r} , and a search direction \underline{v} (initially $\underline{v} = \underline{r}$). It can be shown that the following update equations obtain the exact solution x^* in n steps:

$$\begin{aligned} t_k &= \frac{\langle \underline{r}^{(k-1)}, \underline{r}^{(k-1)} \rangle}{\langle \underline{v}^{(k)}, A\underline{v}^{(k)} \rangle} \\ \underline{x}^{(k)} &= \underline{x}^{(k-1)} + t_k \underline{v}^{(k)} \\ \underline{r}^{(k)} &= \underline{r}^{(k-1)} - t_k A\underline{v}^{(k)} \\ s_k &= \frac{\langle \underline{r}^{(k)}, \underline{r}^{(k)} \rangle}{\langle \underline{r}^{(k-1)}, \underline{r}^{(k-1)} \rangle} \\ \underline{v}^{(k+1)} &= \underline{r}^{(k)} + s_k \underline{v}^{(k)} \end{aligned}$$

As a direct method this is not as good as Gaussian elimination since it takes n^3 multiplies, as compared to $n^3/3$ for Gaussian elimination, and is more susceptible to round-off errors. Rather, its usefulness is that for matrices A which are well-conditioned it produces a very close approximation to x^* in only about \sqrt{n} steps. If A is not well-conditioned, it can be multiplied by a **conditioning matrix** C and a slightly revised version of the above equations applied.