```
for  k = 1,...,n-1 do
    find ℓ such that |a_{ℓ,k}| = max_{k≤i≤n} |a_{i,k}| ;
    save ℓ in ipvt(k) ;
    if a_{ℓ,k} = 0 then skip to end of k loop;
    interchange a_{k,k} and a_{ℓ,k} ;
    for i = k+1,...,n do
        m_{i,k} = -a_{i,k}/a_{k,k}  (save in a_{i,k} );
    for j = k+1,...,n do
        interchange a_{k,j} and a_{ℓ,j} ;
        for i = k+1,...,n do
            a_{i,j} ← a_{i,j} + m_{i,k}*a_{k,j} ;
    end j
end k .
```

In terms of this outline, P_k is the identity with the k-th and ℓ-th rows interchanged and L_k is the identity with $m_{i,k}$, $i = k+1,\ldots,n$ inserted below the diagonal. The upper triangular matrix U is stored in the upper triangle of A , including the diagonal, when the algorithm has been completed. The quantity $a_{\ell,k}$ which is moved into $a_{k,k}$, is called the pivot. If it is zero, then at the k-th step of the elimination, the entire k-th column must be zero below the diagonal and $P_k = L_k = I$. A zero pivot does not indicate failure of the factorization, but simply that the factor U is singular and cannot be used for solving linear systems.

The only modification necessary for complex arithmetic is to define $|z| = |\mathrm{real}(z)| + |\mathrm{imag}(z)|$. This can be computed more rapidly than the conventional modulus and has all the necessary numerical properties.

The condition estimate. The condition number of A is

$$\kappa(A) = \|A\|\,\|A^{-1}\|$$

using the ℓ_1 norm, that is

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

$$\|A\| = \max_x \|Ax\|/\|x\|$$

$$\|A^{-1}\| = \max_z \|z\|/\|Az\|$$

The ℓ_1 vector norm is chosen because it can be computed rapidly and because the subordinate matrix norm can be computed directly from the columns a_j by

$$\|A\| = \max_j \|a_j\| \ .$$

The basic task of the condition estimator is to obtain a good approximation for $\|A^{-1}\|$ without computing all the columns of A^{-1} . This is accomplished by choosing a certain vector y , solving a single system $Az = y$, and then estimating

$$\|A^{-1}\| \approx \|z\|/\|y\| \ .$$

In order to avoid overflow, an estimate of $1/\kappa(A)$ is computed, namely

$$\mathrm{RCOND} = \frac{\|y\|}{\|A\|\,\|z\|} \ .$$

Since $\|A^{-1}\| \geq \|z\|/\|y\|$, the estimate actually satisfies

$$1/\mathrm{RCOND} \leq \kappa(A) \ .$$

If this estimate is to be reasonably accurate, it is necessary that y be chosen in such a way that $\|z\|/\|y\|$ is nearly as large as possible. The technique used is described in a paper by Cline, Moler, Stewart and Wilkinson.

In the condition estimation subroutines, y is obtained by solving $A^Ty = e$ where e is a scalar multiple of a vector with components ± 1 . Using the factorization $A = LU$ this involves solving $U^Tw = e$ and then solving $L^Ty = w$. The components of e are determined during the calculation of w so that $\|w\|$ is large. Suppose that $e_1,\ldots,e_{k-1}$ and $w_1,\ldots,w_{k-1}$ have already been obtained. In the process, the quantities

$$t_j = \sum_{i=1}^{k-1} u_{ij}w_j \ , \quad j = k,\ldots,n$$

are also computed. The equation determining w_k is

$$u_{kk}w_k = e_k - t_k \ .$$

The two possible choices for e_k are

$$e_k^+ = \mathrm{sign}(-t_k) \quad \text{and} \quad e_k^- = -e_k^+$$

which give

$$w_k^+ = (e_k^+ - t_k)/u_{kk} \quad \text{and} \quad w_k^- = (e_k^- - t_k)/u_{kk} \ .$$

The t_j's are temporarily updated, $t_k^+ = e_k^+ - t_k$, $t_k^- = e_k^- - t_k$,

$$t_j^+ = t_j + u_{kj}w_k^+ \quad \text{and} \quad t_j^- = t_j + u_{kj}w_k^- \ , \quad j = k+1,\ldots,n \ .$$

The probable growth in $\|w\|$ is then predicted by comparing $\sum_{j=k}^{n} |t_j^+|$ with $\sum_{j=k}^{n} |t_j^-|$ and choosing the larger. The resulting + or - is used to specify the choice of e_k and hence w_k. These steps are repeated for $k = 1,\ldots,n$.

The algorithm may produce overflow during division by u_{kk} if it is small and breaks down completely if any u_{kk} is zero. This indicates that $\kappa(A)$ is large or possibly that A is singular. The overflows and divisions by zero are avoided by rescaling e so that $|w_k^+| \le 1$ and $|z_k| \le 1$. The resulting value of RCOND may underflow and be set to zero. With this scaling it is not necessary to handle exact singularity as a special case. A singular A will merely produce a small, possibly zero, value of RCOND.

The vector z satisfies

$$\|Az\| = \text{RCOND}\cdot\|A\|\cdot\|z\| .$$

Consequently, if RCOND is small, then z is an approximate null vector of the nearly singular matrix A.

The general outline of the entire process is

compute $\|A\|$

factor $A = LU$

solve $U^T w = e$, choosing e_k as described

solve $L^T y = w$

solve $Lv = y$

solve $Uz = v$

$\text{RCOND} = \|y\|/(\|A\|\,\|z\|)$.

Solving linear systems. With the factorization $A = LU$ the linear system $Ax = b$ is equivalent to the two triangular systems, $Ly = b$ and $Ux = y$. The algorithm thus involves two stages. The first stage, the forward elimination, produces $y = L^{-1}b$ by applying the same permutation and elimination operations to b that were applied to the columns of A during the factorization. The second stage, the back substitution, consists of solving the upper triangular system $Ux = y$. To obtain an algorithm which involves operations on the columns of U, this system can be written

$$x_1 \begin{bmatrix} u_{11} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} u_{12} \\ u_{22} \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \cdots + x_n \begin{bmatrix} u_{1n} \\ u_{2n} \\ \vdots \\ u_{nn} \end{bmatrix} = y .$$