```
function [L,U,P,g] = lupp(Ao)
%lupp  Computes the LU decomposition of A with partial pivoting
%
%   [A,p] = lupp(A) Computes the LU decomposition of A with partial
%   pivoting using vectorization. The factors L and U are returned in the
%   output A, and the permutation of the rows from partial pivoting are
%   recorded in the vector p. Here L is assumed to be unit lower
%   triangular, meaning it has ones on its diagonal. The resulting
%   decomposition can be extracted from A and p as follows:
%       L = eye(length(LU))+tril(LU,-1);     % L with ones on diagonal
%       U = triu(LU);                        % U
%       P = p*ones(1,n) == ones(n,1)*(1:n);  % Permutation matrix
%   A is then given as L*U = P*A, or P'*L*U = A.
%
%   Use this function in conjuction with backsub and forsub to solve a
%   linear system Ax = b.
A = Ao;
n = size(A,1);
p = (1:n)';

for k=1:n-1
    % Find the row in column k that contains the largest entry in magnitude
    [~,pos] = max(abs(A(k:n,k)));
    row2swap = k-1+pos;
    % Swap the rows of A and p (perform partial pivoting)
    A([row2swap, k],:) = A([k, row2swap],:);
    p([row2swap, k]) = p([k, row2swap]);
    % Perform the kth step of Gaussian elimination
    J = k+1:n;
    A(J,k) = A(J,k)/A(k,k);
    A(J,J) = A(J,J) - A(J,k)*A(k,J);
end

%Permutation matrix
P = p*ones(1,n) == ones(n,1)*(1:n);

LU = A;

%unit lower triangular matrix L
L = eye(length(LU))+tril(LU,-1);

%upper triangular matrix U
U = triu(LU);

%growth factor
u = abs(U);
%A = P'*L*U;
a = abs(Ao);
u1 = max(u,[],'all');
a1 = max(a,[],'all');
g = u1/a1;

end
```

```
Not enough input arguments.
```

```
Error in lupp (line 17)
A = Ao;
```

Published with MATLAB® R2020b