Operation Counts and MATLAB Code for Gaussian Elimination

Here we explain what the MATLAB command `A\b` actually does when A is an n by n nonsingular matrix and b is a given n-vector. We count the number of operations (additions, subtractions, multiplications, and divisions) required to solve the linear system $Ax = b$.

Step 1. If $a_{11} = 0$, exchange rows so that the $(1,1)$ element is nonzero. For each row $i = 2, \ldots, n$, replace row i by itself minus a_{i1}/a_{11} times row 1:

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} & b_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} & b_n \end{array}\right) \longrightarrow \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & b_1 \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \ldots & \tilde{a}_{2n} & \tilde{b}_2 \\ 0 & \tilde{a}_{32} & \tilde{a}_{33} & \ldots & \tilde{a}_{3n} & \tilde{b}_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & \tilde{a}_{n2} & \tilde{a}_{n3} & \ldots & \tilde{a}_{nn} & \tilde{b}_n \end{array}\right)$$

Operation Count:
$n-1$ divisions to compute the numbers a_{i1}/a_{11}; $n(n-1)$ multiplications and $n(n-1)$ subtractions to do the elimination. Total: approximately $2n^2$ operations. [Note: We are interested in the operation count for large n. Hence it will not matter much whether it is $2n^2$ or $2n(n-1)$. The coefficient of the largest term n^2 is still 2.]

MATLAB code for this step (after rows have been exchanged so that the $(1,1)$ element is nonzero):

```
% Eliminate first column using first row.
for i=2:n,
  mult = A(i,1)/A(1,1);
  for k=1:n,
    A(i,k) = A(i,k) - mult*A(1,k);
  end;
  b(i) = b(i) - mult*b(1);
end;
```

Note that the inner *for loop* can be replaced by the statement:

```
A(i,:) = A(i,:) - mult*A(1,:);    % This does the same thing as the inner
                                  % loop above.
```

Step 2. If $\tilde{a}_{22} = 0$, exchange rows so that the $(2,2)$ element is nonzero. For each row $i = 3, \ldots, n$, replace row i by itself minus $\tilde{a}_{i2}/\tilde{a}_{22}$ times row 2.

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & b_1 \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \ldots & \tilde{a}_{2n} & \tilde{b}_2 \\ 0 & \tilde{a}_{32} & \tilde{a}_{33} & \ldots & \tilde{a}_{3n} & \tilde{b}_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & \tilde{a}_{n2} & \tilde{a}_{n3} & \ldots & \tilde{a}_{nn} & \tilde{b}_n \end{array}\right) \longrightarrow \left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & b_1 \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \ldots & \tilde{a}_{2n} & \tilde{b}_2 \\ 0 & 0 & \tilde{\tilde{a}}_{33} & \ldots & \tilde{\tilde{a}}_{3n} & \tilde{\tilde{b}}_3 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \tilde{\tilde{a}}_{n3} & \ldots & \tilde{\tilde{a}}_{nn} & \tilde{\tilde{b}}_n \end{array}\right)$$

Operation Count: $n-2$ divisions to compute the numbers $\tilde{a}_{i2}/\tilde{a}_{22}$; $(n-1)(n-2)$ multiplications and $(n-1)(n-2)$ subtractions to do the elimination. Total: approximately $2(n-1)^2$ operations.

MATLAB code for this step (after rows have been exchanged so that the $(2,2)$ element is nonzero):

```
for i=3:n,
  mult = A(i,2)/A(2,2);
  A(i,2:n) = A(i,2:n) - mult*A(2,2:n);  % Note that we work only with
                                        % columns 2 through n.
end;
```

Step 3. If $\tilde{\tilde{a}}_{33} = 0$, exchange rows so that the $(3,3)$ element is nonzero. For each row $i = 4, \ldots, n$, replace row i by itself minus $\tilde{\tilde{a}}_{i3}/\tilde{\tilde{a}}_{33}$ times row 3.

Operation Count: $2(n-2)^2$.

Etc.

Total Operation Count for Gaussian Elimination:

$$2[n^2 + (n-1)^2 + (n-2)^2 + \ldots + 1^2] = 2\frac{n(n+1)(2n+1)}{6} \approx 2\frac{n^3}{3}$$

MATLAB Code:

```
% Given an n by n nonsingular matrix A and an n-vector b, does Gaussian
% elimination (with pivoting) to bring A to upper triangular form.

for j=1:n-1,
  [cmax,jmax] = max(abs(A(j:n,j))); % Find index of maximum element in column j.
  temp = A(j,:);                    % Exchange rows j and j+jmax-1.
  A(j,:) = A(j+jmax-1,:);
  A(j+jmax-1,:) = temp;
  tempb = b(j);                    % Also exchange elements j and j+jmax-1 of b.
  b(j) = b(j+jmax-1);
  b(j+jmax-1) = tempb;
  for i=j+1:n,                      % Eliminate in column j using row j.
    mult = A(i,j)/A(j,j);
    A(i,j:n) = A(i,j:n) - mult*A(j,j:n);
    b(i) = b(i) - mult*b(j);
  end;
end;
```

The matrix is now upper triangular. We still must backsolve to find the solution of the linear system:

$$\begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ & & \ddots & \\ & & & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}.$$

For $i = n, n-1, \dots, 1$,

$$x_i = (f_i - \sum_{j=i+1}^{n} u_{ij} x_j)/u_{ii}.$$

This requires approximately

$$2\sum_{i=1}^{n}(n-i+1) = 2\frac{n(n+1)}{2} \approx n^2$$

This is lower order (n^2) than the work required to get A into upper triangular form (n^3).

The MATLAB code for backsolving is:

```
% Solve the linear system Ax=b once A has been reduced to upper triangular
% form.
x = zeros(n,1);      % Force x to be a column vector.
x(n) = b(n)/A(n,n);
for i=n-1:-1:1,
  x(i) = (b(i) - A(i,i+1:n)*x(i+1:n))/A(i,i);
end;
```