Solving Linear Systems

We have given general conditions for a system of linear equations defined by the matrix identity

$$Ax = b$$

where $A$ is an $n \times n$ matrix and $x$ and $b$ are $n$ vectors. $A$ must be nonsingular, i.e., its columns must be linearly independent.

How do we compute this in practice?

We first add structure to the pattern of the elements in the matrix to simplify the problem
Simplest case: \( A = I = [e_1, e_2, \cdots, e_n] \)

\[
\begin{align*}
Ax &= b \\
Ix &= b \\
x &= b
\end{align*}
\]

Next simplest case \( A \) is a nonsingular diagonal matrix \( \alpha_{ij} = 0, \text{ for } i \neq j \) and \( \alpha_{ii} \neq 0, \text{ for } i = 1, \ldots, n \)
Example: \( n = 4 \)

\[
\begin{pmatrix}
\alpha_{11} & \alpha_{22} \\
\alpha_{22} & \alpha_{33} \\
\alpha_{33} & \alpha_{44}
\end{pmatrix}
\begin{pmatrix}
\xi_1 \\
\xi_2 \\
\xi_3 \\
\xi_4
\end{pmatrix}
= 
\begin{pmatrix}
\phi_1 \\
\phi_2 \\
\phi_3 \\
\phi_4
\end{pmatrix}
\]

This defines the following identities:

\[
\begin{align*}
\alpha_{11}\xi_1 &= \phi_1 \\
\alpha_{22}\xi_2 &= \phi_2 \\
\alpha_{33}\xi_3 &= \phi_3 \\
\alpha_{44}\xi_4 &= \phi_4
\end{align*}
\]

And the linearly independence condition guarantees it can be solved.
Triangular Systems

Consider the following set of equations with $n = 4$

\[
\begin{align*}
\lambda_{11}\xi_1 &= \phi_1 \\
\lambda_{21}\xi_1 + \lambda_{22}\xi_2 &= \phi_2 \\
\lambda_{31}\xi_1 + \lambda_{32}\xi_2 + \lambda_{33}\xi_3 &= \phi_3 \\
\lambda_{41}\xi_1 + \lambda_{42}\xi_2 + \lambda_{43}\xi_3 + \lambda_{44}\xi_4 &= \phi_4
\end{align*}
\]

This can be written using matrix notation as the lower triangular system

\[Lx = f\]

where $L = [\lambda_{ij}] \in \mathbb{R}^{n \times n}$, $\lambda_{ij} = 0$ if $i < j$, $f = [\phi_i] \in \mathbb{R}^n$, $x = [\xi_i] \in \mathbb{R}^n$ $\lambda_{ii} \neq 0$ if $i = 1, \ldots, n$,

\[
\begin{pmatrix}
\lambda_{11} \\
\lambda_{21} & \lambda_{22} \\
\lambda_{31} & \lambda_{32} & \lambda_{33} \\
\lambda_{41} & \lambda_{42} & \lambda_{43} & \lambda_{44}
\end{pmatrix}
\begin{pmatrix}
\xi_1 \\
\xi_2 \\
\xi_3 \\
\xi_4
\end{pmatrix}
= 
\begin{pmatrix}
\phi_1 \\
\phi_2 \\
\phi_3 \\
\phi_4
\end{pmatrix}
\]
Rewriting the equations gives the following identities:

\[
\begin{align*}
\lambda_{11} & \xi_1 = \phi_1 \\
\lambda_{22} \xi_2 & = \phi_2 - \lambda_{21} \xi_1 \\
\lambda_{33} \xi_3 & = \phi_3 - \lambda_{31} \xi_1 - \lambda_{32} \xi_2 \\
\lambda_{44} \xi_4 & = \phi_4 - \lambda_{41} \xi_1 - \lambda_{42} \xi_2 - \lambda_{43} \xi_3
\end{align*}
\]

As a result we can derive two standard sequential methods for solving these systems in \(O(n^2)\) operations. They differ in the fact that one is oriented towards rows, and the other columns.

First we must choose data structures to store the mathematical objects \(L\), \(x\) and \(f\) We let

\[
\begin{align*}
L(I, J) & = \lambda_{ij} \\
X(I) & = \xi_i \\
F(I) & = \phi_i
\end{align*}
\]
Row oriented:

\[
X(1) = F(1) / L(1,1)
\]

\[
\text{do } I = 2, N \\
\text{ do } J = 1, I - 1 \\
\quad F(I) = F(I) - L(I,J) \times (J) \\
\text{ enddo} \\
X(I) = F(I) / L(I,I)
\]

enddo
\textit{Column oriented}:
\begin{verbatim}
do \hspace{1em} J = 1, N - 1 \\
\hspace{2em} X(J) = F(J) / L(J,J) \\
\hspace{2em} do \hspace{1em} I = J + 1, N \\
\hspace{3em} F(I) = F(I) - L(I,J) \times (J) \\
\hspace{2em} end do \\
end do \\
X(N) = F(N) / L(N,N)
\end{verbatim}
We can derive these algorithms from another point of view: matrix partitioning.

Partition to solve a simple problem and reduce the size of the overall problem to be solved by 1.

$$\begin{pmatrix} \lambda_{11} & 0^T \\ l_{n-1} & L_{n-1} \end{pmatrix} \begin{pmatrix} \xi_1 \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} \phi_1 \\ f_{n-1} \end{pmatrix}$$

where $L_{n-1}$ is an $n-1 \times n-1$ lower triangular matrix, $f_{n-1}, l_{n-1}$ and $x_{n-1}$ are $n-1$ vectors.
Multiplying out the partitioned equations yields the identities:

\[
\begin{align*}
\lambda_{11} \xi_1 &= \phi_1 \\
\xi_1 l_{n-1} + L_{n-1} x_{n-1} &= f_{n-1}
\end{align*}
\]

Therefore we can solve the 1 \times 1 problem and then produce a problem with the same structure as the \( n \times n \) problem but of dimension \( n - 1 \).

\[
\begin{align*}
\xi_1 &= \phi_1 / \lambda_{11} \\
L_{n-1} x_{n-1} &= f_{n-1} - \xi_1 l_{n-1} = \bar{f}_{n-1}
\end{align*}
\]

Note this is the column oriented algorithm.
Partition to solve the problem by assuming the solution is available for an $n-1$ order problem and expand the system to get the form of the solution for a problem of order $n$.

$$
\begin{pmatrix}
L_{n-1} & 0 \\
T_{n-1} & \lambda_{nn}
\end{pmatrix}
\begin{pmatrix}
x_{n-1} \\
\xi_n
\end{pmatrix}
=
\begin{pmatrix}
f_{n-1} \\
\phi_n
\end{pmatrix}
$$
Multiplying out the partitioned equations yields the identities:

\[ \begin{align*}
L_{n-1} x_{n-1} &= f_{n-1} \\
l^T_{n-1} x_{n-1} + \lambda_{nn} \xi_n &= \phi_n
\end{align*} \]

Assuming the solution \( x_{n-1} \) is available we get the solution to the problem of order \( n \) by determining \( \xi_n \)

\[ \xi_n = (\phi_n - l^T_{n-1} x_{n-1})/\lambda_{nn} \]

Note this is the row oriented version and it can also generate a recursive version if the solution \( x_{n-1} \) is done via recursive function call.
Elementary triangular matrices and solving triangular systems

Definition: An elementary triangular matrix (column form) is a matrix of the form:

\[ M_i = I + l_i e_i^T \]

where \( l_i \) is 0 in the first \( i \) positions and \( e_i \) is the standard basis vector which has 0 in all positions with the exception of a 1 in the i-th position.

Example: \( n = 4, \ i = 2, \ x \) denotes a nonzero position

\[
M_2 = \begin{bmatrix}
1 & 1 \\
0 & x & 1 \\
0 & x & 0 & 1
\end{bmatrix},
\begin{bmatrix}
l_2 \\
1
\end{bmatrix},
\begin{bmatrix}
e_2 \\
x
\end{bmatrix}
\]

Note a row form of \( I + e_i l_i^T \) can also be defined.
Consider $M_i M_j$

$$M_i M_j = (I + l_i e_i^T)(I + l_j e_j^T) = I + l_i e_i^T + l_j e_j^T + \alpha l_i e_j^T$$

where $\alpha = e_i^T l_j$.

If $i \leq j$ then $\alpha = 0$ and the product merely copies the nonzero elements below the diagonal of $M_i$ and $M_j$ into the corresponding positions in the product, i.e., no operations need to be performed.

Example: $n = 4$

$$\begin{bmatrix} 1 & 1 \\ \alpha_1 & 0 \\ \alpha_2 & 1 \\ \alpha_3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \beta_1 \\ 0 & \beta_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \alpha_3 & 0 \end{bmatrix}$$
If $i > j$ then $\alpha \neq 0$ but,

$$M_i M_j = I + l_i e_i^T + (l_j + \alpha l_i) e_j^T$$

Therefore, the nonzero pattern of the product is the union of the nonzero patterns of $M_i$ and $M_j$, i.e., 1 on the diagonal and nonzeros below the diagonal in columns $i$ and $j$ only. Note also that the computations required are a single vector triad $l_j + \alpha l_i$.

These facts about multiplication of elementary triangular matrices extend immediately to the case where $M_i$ and $M_j$ have a set of contiguous columns which are nonzero below the diagonal. For example, if $M_i$ and $M_j$ have $k$ nontrivial columns each then $M_i M_j$ has $2k$ nontrivial columns in the corresponding positions.
The following results follow in a straightforward fashion from the previous facts:

- A unit lower triangular matrix $L$ can be written as
  
  $$L = \prod_{i=1}^{n-1} C_i$$

  where $C_i = I + l_i e_i^T$ is the elementary triangular matrix in column form defined by the nonzeros below the diagonal in the $i$-th column of $L$.

- The multiplication of an elementary triangular matrix in column form by a vector is equivalent to a triad.

  $$C_i x = (I + l_i e_i^T)x = x + \xi_i l_i$$

  Note that due to the nonzero structure in $l_i$ the triad is of length $n - i$. 
• The multiplication of an elementary triangular matrix in row form by a vector is equivalent to a dot-product

\[ R_ix = (I + e_il_i^T)x = x + e_il_i^T x \]

Only one element of \( x \) is updated, \( \xi_i \leftarrow \xi_i + l_i^T x \). Note that due to the nonzero structure in \( l_i \) the dotproduct is of length \( n - i \).

• Inversion of elementary triangular matrices is trivial. \( C_i^{-1} = I - l_ie_i^T \). Therefore, solving a system \( C_ix = b \) is equivalent to a triad (or a dotproduct for the row version) and the inverses of \( C_i \) and \( R_i \) are elementary triangular matrices.

• The inverse of a (unit) lower triangular matrix is a (unit) lower triangular matrix.
An algebraic approach to the column and row algorithms

The column and row algorithms to solve $Lx = f$ can be derived in a straightforward manner based on this algebraic characterization.

Let

$$L = \prod_{i=1}^{n-1} C_i = \prod_{j=2}^{n} R_j,$$

where $C_i = I + l_i e_i^T$, $R_j = I + e_j v_j^T$, $l_i$ is the vector corresponding to column $i$ in $L$ with the 1 on the diagonal removed and $v_j$ is similarly constructed from row $j$ of $L$. Clearly,

$$x = \left( \prod_{i=n-1}^{1} N_i \right) f = \left( \prod_{j=n}^{2} M_j \right) f$$

where $N_i = C_i^{-1}$ and $M_j = R_j^{-1}$. Note that no computation is involved in defining this factorization — unlike more standard factorizations we will discuss later.
The column and row sweeps are generated by a particular grouping of the computations. For \( n = 8 \) the grouping is

\[
(N_7(N_6(N_5(N_4(N_3(N_2(N_1 f)))))))
\]

and

\[
(M_8(M_7(M_6(M_5(M_4(M_3(M_2 f))))))).
\]