6.5.2 Summary

Linear systems of equations

A linear system of equations with m equations in n unknowns is given by

$$\begin{array}{ccccccccc}
\alpha_{0,0}\chi_0 & + & \alpha_{0,1}\chi_1 & + & \cdots & + & \alpha_{0,n-1}\chi_{n-1} & = & \beta_0 \\
\alpha_{1,0}\chi_0 & + & \alpha_{1,1}\chi_1 & + & \cdots & + & \alpha_{1,n-1}\chi_{n-1} & = & \beta_1 \\
\vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
\alpha_{m-1,0}\chi_0 & + & \alpha_{m-1,1}\chi_1 & + & \cdots & + & \alpha_{m-1,n-1}\chi_{n-1} & = & \beta_{m-1}
\end{array}$$

Variables $\chi_0, \chi_1, \ldots, \chi_{n-1}$ are the unknowns.

This Week, we only considered the case where $m = n$:

$$\begin{array}{ccccccccc}
\alpha_{0,0}\chi_0 & + & \alpha_{0,1}\chi_1 & + & \cdots & + & \alpha_{0,n-1}\chi_{n-1} & = & \beta_0 \\
\alpha_{1,0}\chi_0 & + & \alpha_{1,1}\chi_1 & + & \cdots & + & \alpha_{1,n-1}\chi_{n-1} & = & \beta_1 \\
\vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\
\alpha_{n-1,0}\chi_0 & + & \alpha_{n-1,1}\chi_1 & + & \cdots & + & \alpha_{n-1,n-1}\chi_{n-1} & = & \beta_{n-1}
\end{array}$$

Here the $\alpha_{i,j}$s are the coefficients in the linear system. The β_is are the right-hand side values.

Basic tools

Solving the above linear system relies on the fact that its solution does not change if

1. Equations are reordered (not used until next week);
2. An equation in the system is modified by subtracting a multiple of another equation in the system from it; and/or
3. Both sides of an equation in the system are scaled by a nonzero.

Gaussian elimination is a method for solving systems of linear equations that employs these three basic rules in an effort to reduce the system to an upper triangular system, which is easier to solve.

Appended matrices

The above system of n linear equations in n unknowns is more concisely represented as an appended matrix:

$$\left(\begin{array}{cccc|c}
\alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} & \beta_0 \\
\alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n-1} & \beta_1 \\
\vdots & \vdots & & \vdots & \vdots \\
\alpha_{n-1,0} & \alpha_{n-1,1} & \cdots & \alpha_{n-1,n-1} & \beta_{n-1}
\end{array}\right)$$

This representation allows one to just work with the coefficients and right-hand side values of the system.

Matrix-vector notation

The linear system can also be represented as $Ax = b$ where

$$\begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \end{pmatrix} \quad \begin{pmatrix} \chi_0 \end{pmatrix} \quad \begin{pmatrix} \beta_0 \end{pmatrix}$$

$$A = \left(\begin{array}{cccc} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \\ \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n-1} \\ \vdots & \vdots & & \vdots \\ \alpha_{n-1,0} & \alpha_{n-1,1} & \cdots & \alpha_{n-1,n-1} \end{array} \right), \quad x = \left(\begin{array}{c} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{array} \right), \quad \text{and} \quad \left(\begin{array}{c} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{array} \right).$$

Here, A is the matrix of coefficients from the linear system, x is the solution vector, and b is the right-hand side vector.

Gauss transforms

A Gauss transform is a matrix of the form

$$L_j = \begin{pmatrix} I_j & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -\lambda_{j+1,j} & 1 & 0 & \cdots & 0 \\ 0 & -\lambda_{j+2,j} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -\lambda_{n-1,j} & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

When applied to a matrix (or a vector, which we think of as a special case of a matrix), it subtracts $\lambda_{i,j}$ times the jth row from the ith row, for $i = j+1,\ldots,n-1$. Gauss transforms can be used to express the operations that are inherently performed as part of Gaussian elimination with an appended system of equations.

The action of a Gauss transform on a matrix, $A := L_j A$ can be described as follows:

$$\begin{pmatrix} I_j & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -\lambda_{j+1,j} & 1 & 0 & \cdots & 0 \\ 0 & -\lambda_{j+2,j} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -\lambda_{n-1,j} & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} A_0 \\ \widetilde{a}_j^T \\ \widetilde{a}_{j+1}^T \\ \vdots \\ \widetilde{a}_{n-1}^T \end{pmatrix} = \begin{pmatrix} A_0 \\ \widetilde{a}_j^T \\ \widetilde{a}_{j+1}^T - \lambda_{j+1,j}\widetilde{a}_j^T \\ \vdots \\ \widetilde{a}_{n-1}^T - \lambda_{n-1,j}\widetilde{a}_j^T \end{pmatrix}.$$

An important observation that was NOT made well enough this week is that the rows of A are updates with an AXPY! A multiple of the jth row is subtracted from the ith row.

A more concise way to describe a Gauss transforms is

$$\widetilde{L} = \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right).$$

Now, applying to a matrix A, $\widetilde{L}A$ yields

$$\left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & -l_{21} & I \end{array}\right) \left(\begin{array}{c} A_0 \\ \hline a_1^T \\ \hline A_2 \end{array}\right) = \left(\begin{array}{c} A_0 \\ \hline a_1^T \\ \hline A_2 - l_{21}a_1^T \end{array}\right).$$

In other words, A_2 is updated with a rank-1 update. **An important observation that was NOT made well enough this week is that a rank-1 update can be used to simultaneously subtract multiples of a row of A from other rows of A.**

Forward substitution

Forward substitution applies the same transformations that were applied to the matrix to a right-hand side vector.

Back(ward) substitution

Backward substitution solves the upper triangular system

$$\begin{array}{ccccccccc}
\alpha_{0,0}\chi_0 & + & \alpha_{0,1}\chi_1 & + & \cdots & + & \alpha_{0,n-1}\chi_{n-1} & = & \beta_0 \\
 & & \alpha_{1,1}\chi_1 & + & \cdots & + & \alpha_{1,n-1}\chi_{n-1} & = & \beta_1 \\
 & & & & \ddots & & \vdots & \vdots & \vdots \\
 & & & & & & \alpha_{n-1,n-1}\chi_{n-1} & = & \beta_{n-1}
\end{array}$$

This algorithm overwrites b with the solution x.

LU factorization

The LU factorization factorization of a square matrix A is given by $A = LU$, where L is a unit lower triangular matrix and U is an upper triangular matrix. An algorithm for computing the LU factorization is given by

Algorithm: $A := \text{LU\_UNB\_VAR5}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

$a_{21} := a_{21}/\alpha_{11} \quad (= l_{21})$

$A_{22} := A_{22} - a_{21}a_{12}^T \quad (= A_{22} - l_{21}a_{12}^T)$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right)$$

endwhile

This algorithm overwrites A with the matrices L and U. Since L is unit lower triangular, its diagonal needs not be stored.

The operations that compute an LU factorization are the same as the operations that are performed when reducing a system of linear equations to an upper triangular system of equations.

Solving $Lz = b$

Forward substitution is equivalent to solving a unit lower triangular system $Lz = b$. An algorithm for this is given by

Algorithm: $[b] := \text{LTRSV\_UNB\_VAR1}(L, b)$

Partition $L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$, $b \rightarrow \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right)$

where L_{TL} is 0×0, b_T has 0 rows

while $m(L_{TL}) < m(L)$ **do**

Repartition

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right) \rightarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \\ \hline b_2 \end{array}\right)$$

$b_2 := b_2 - \beta_1 l_{21}$

Continue with

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline l_{10}^T & \lambda_{11} & 0 \\ \hline L_{20} & l_{21} & L_{22} \end{array}\right), \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right) \leftarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \\ \hline b_2 \end{array}\right)$$

endwhile

This algorithm overwrites b with the solution z.

Solving $Ux = b$

Back(ward) substitution is equivalent to solving an upper triangular system $Ux = b$. An algorithm for this is given by

Algorithm: $[b] := \text{UTRSV\_UNB\_VAR1}(U, b)$

Partition $U \rightarrow \left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline U_{BL} & U_{BR} \end{array}\right)$, $b \rightarrow \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right)$

where U_{BR} is 0×0, b_B has 0 rows

while $m(U_{BR}) < m(U)$ **do**

Repartition

$$\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} U_{00} & u_{01} & U_{02} \\ \hline 0 & \upsilon_{11} & u_{12}^T \\ \hline 0 & 0 & U_{22} \end{array}\right), \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right) \rightarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \\ \hline b_2 \end{array}\right)$$

$\beta_1 := \beta_1 - u_{12}^T b_2$

$\beta_1 := \beta_1 / \upsilon_{11}$

Continue with

$$\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} U_{00} & u_{01} & U_{02} \\ \hline 0 & \upsilon_{11} & u_{12}^T \\ \hline 0 & 0 & U_{22} \end{array}\right), \left(\begin{array}{c} b_T \\ \hline b_B \end{array}\right) \leftarrow \left(\begin{array}{c} b_0 \\ \hline \beta_1 \\ \hline b_2 \end{array}\right)$$

endwhile

This algorithm overwrites b with the solution x.

Solving $Ax = b$

If LU factorization completes with an upper triangular matrix U that does not have zeroes on its diagonal, then the following three steps can be used to solve $Ax = b$:

- Factor $A = LU$.
- Solve $Lz = b$.
- Solve $Ux = z$.

Cost

- Factoring $A = LU$ requires, approximately, $\frac{2}{3}n^3$ floating point operations.
- Solve $Lz = b$ requires, approximately, n^2 floating point operations.
- Solve $Ux = z$ requires, approximately, n^2 floating point operations.