L5: Linear Algebra, Matrices, Gauss Elimination (Chapters 8 & 9)

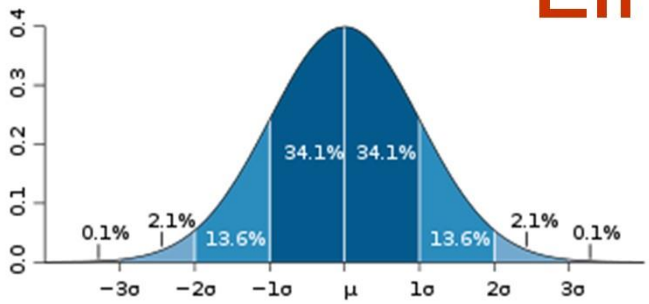
BME 313L
Introduction to Numerical Methods in BME

Tim Yeh

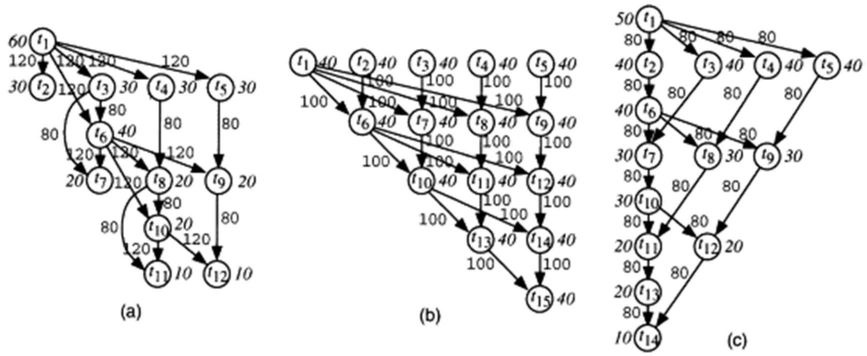
Department of Biomedical Engineering

University of Texas at Austin

Tim.Yeh@austin.utexas.edu



Linear Algebraic Equations, Matrices, Gauss Elimination



$$
A^{(k)} = \left[\begin{array}{cccc|ccc}
a_{11}^{(k)} & \cdot & \cdots & a_{1,k-1}^{(k)} & a_{1k}^{(k)} & \cdots & a_{1n}^{(k)} \\
0 & a_{22}^{(k)} & \cdots & a_{2,k-1}^{(k)} & \cdot & & \cdot \\
\vdots & & \ddots & \vdots & \vdots & & \vdots \\
0 & \cdot & \cdots & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \cdots & a_{k-1,n}^{(k)} \\
\hline
0 & \cdot & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\
\vdots & & & & \vdots & & \vdots \\
0 & \cdot & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)}
\end{array}\right].
$$

- Matrix algebra
- Linear algebraic equations (LAEs)
- Solving LAEs with MATLAB
- Graphical methods
- Naïve Gauss elimination
- Pivoting
- Diagonal systems

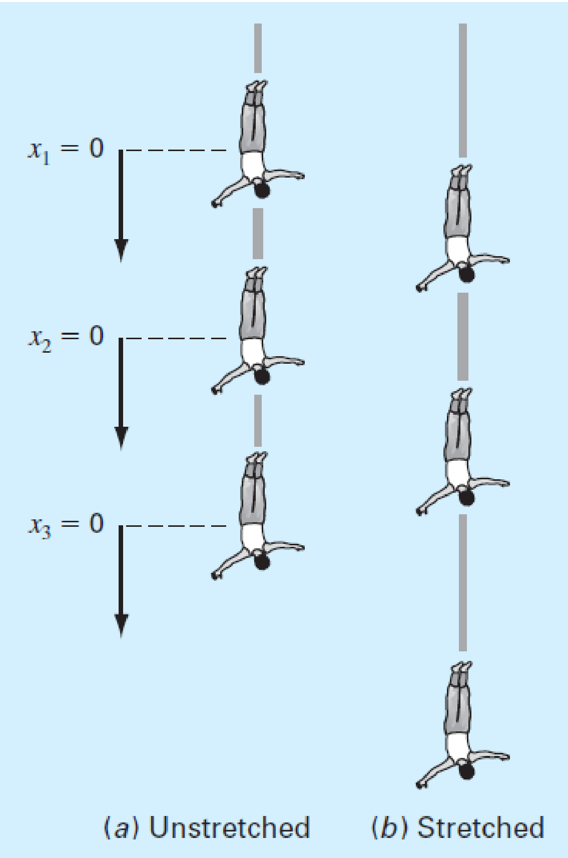
Learning Objectives (Chapter 8) Linear Algebra and Matrices

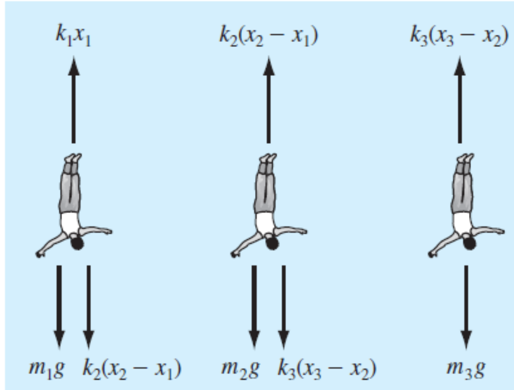
- Understanding matrix notation.
- Being able to identify the following types of matrices: identity, diagonal, symmetric, triangular, and tridiagonal.
- Knowing how to perform matrix **multiplication** and being able to assess when it is feasible.
- Knowing how to **represent** a system of linear equations **in matrix form**.
- Knowing how to solve linear algebraic equations with left division and matrix inversion in MATLAB.

Learning Objectives (Chapter 9) Gauss Elimination

- Knowing how to solve small sets of linear equations with the graphical method and Cramer's rule.
- Understanding how to implement forward elimination and back substitution as in Gauss elimination.
- Understanding how to count **flops** to evaluate the efficiency of an algorithm.
- Understanding the concepts of **singularity** and **ill-condition**.
- Understanding how partial pivoting is implemented and how it differs from complete pivoting.
- Recognizing how the banded structure of a **tridiagonal** system can be exploited to obtain extremely efficient solutions.

Large Linear Algebraic Equations



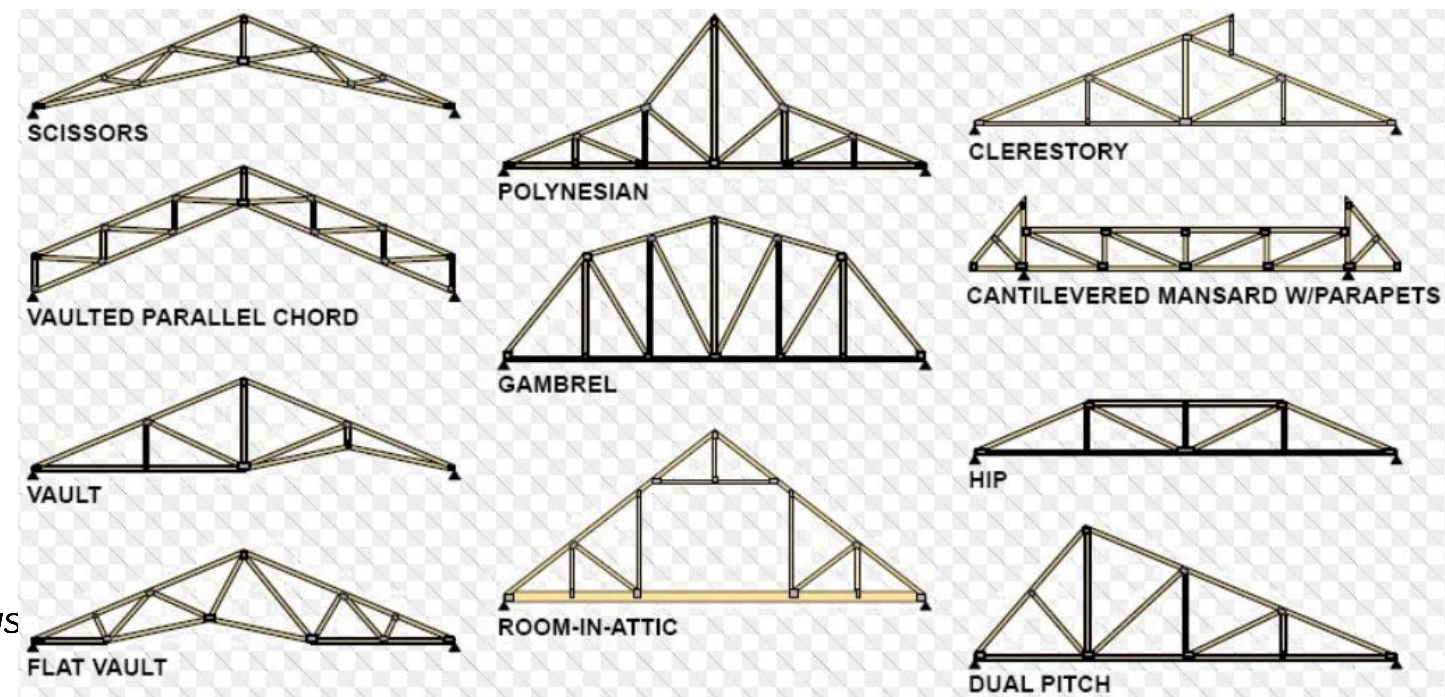


$$m_1\frac{d^2x_1}{dt^2} = m_1g + k_2(x_2 - x_1) - k_1x_1$$

$$m_2\frac{d^2x_2}{dt^2} = m_2g + k_3(x_3 - x_2) + k_2(x_1 - x_2)$$

$$m_3\frac{d^2x_3}{dt^2} = m_3g + k_3(x_2 - x_3)$$

$$\begin{aligned}(k_1 + k_2)x_1 \quad\quad - k_2x_2 \quad\quad\quad &= m_1g \\ -k_2x_1 + (k_2 + k_3)x_2 - k_3x_3 &= m_2g \\ -k_3x_2 + k_3x_3 &= m_3g\end{aligned}$$



Linear Algebra, MATLAB, Gaus

Overview

- A *matrix* consists of a rectangular array of elements represented by a single symbol (example: [*A*]).
- An individual entry of a matrix is an *element* (example: $\mathbf{a_{23}}$)

$$\begin{array}{ccc} & \begin{array}{ccccc} \phantom{a_{11}} & \phantom{a_{12}} & \text{Column 3} & & \phantom{a_{1n}} \\ & & \downarrow & & \end{array} & \\ [A] = & \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & \boxed{a_{23}} & \cdots & a_{2n} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} & \begin{array}{l} \\ \leftarrow \text{Row 2} \\ \\ \\ \\ \\ \end{array} \end{array}$$

Overview (cont)

- A horizontal set of elements is called a *row* and a vertical set of elements is called a *column*.
- The first subscript of an element indicates the row while the second indicates the column.
- The size of a matrix is given as m rows by n columns, or simply m by n (or m x n).
- 1 x n matrices are ***row*** *vectors*.
- m x 1 matrices are ***column*** *vectors.*

Special Matrices

- Matrices where *m=n* are called ***square matrices***.
- There are a number of special forms of square matrices:

Symmetric	Diagonal	Identity
$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$	$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$	$[A] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Upper Triangular	**Lower Triangular**	**Banded**
$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ & & a_{33} \end{bmatrix}$	$[A] = \begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$	$[A] = \begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{bmatrix}$

In this example, the "bandwidth" is 3
$\Rightarrow$ **Tridiagonal** matrix

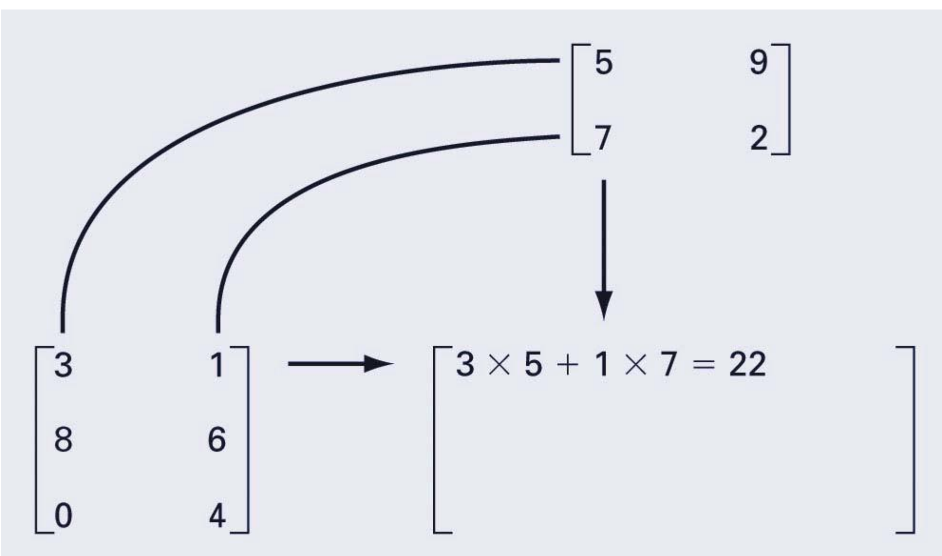
Matrix Operations

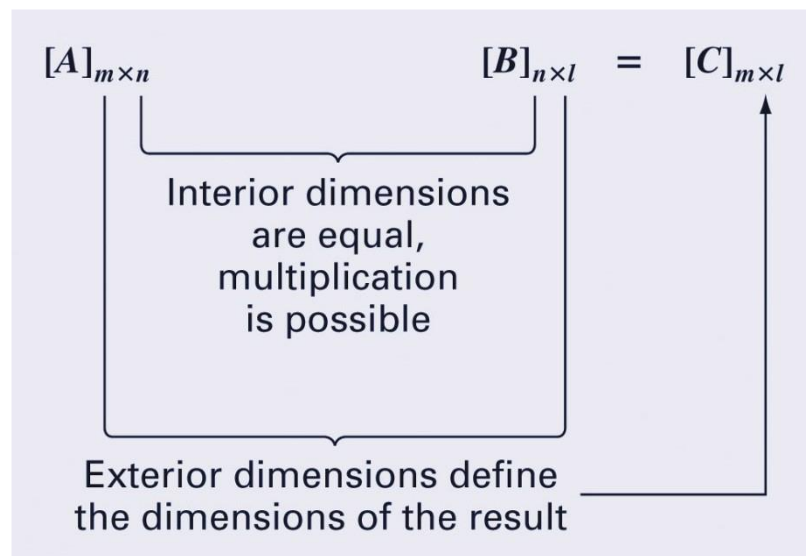
- Two matrices are considered "equal" if and only if every element in the first matrix is equal to every corresponding element in the second matrix
  - this also means the two matrices must have the same size.
- Matrix addition and subtraction are performed by adding or subtracting the corresponding elements. This requires that the two matrices be the same size.
- Scalar matrix multiplication is performed by multiplying each element by the same scalar.

Matrix Multiplication

- The elements in the matrix [C] that results from multiplying matrices [A] and [B] are calculated using:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$





Matrix Operating Rules

- Both addition and subtraction are **commutative** and **associative**

- What about multiplication?

That is, the order of matrix multiplication is important!

Inverse of a Matrix

- The ***inverse*** of a square, nonsingular matrix $[A]$ is another matrix $[A]^{-1}$ which, when multiplied by $[A]$, yields the identity matrix.

$$[A][A]^{-1}=[A]^{-1}[A]=[I]$$

$$[A]^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

Why do we need inverse of a matrix?

- There is really no matrix division – not a defined operation.

- However, the multiplication of a matrix by the inverse is analogous to **division**.

Transpose of a Matrix

- The ***transpose*** of a matrix involves transforming its rows into columns and its columns into rows.

$$(a_{ij})^T = a_{ji}$$

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\{c\} = \begin{Bmatrix} c_1 \\ c_1 \\ c_1 \end{Bmatrix}$$

$$[A]^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

$$\{c\}^T = \lfloor c_1 \quad c_2 \quad c_3 \rfloor$$

Augmentation

- A matrix is **augmented** by the addition of columns to the original matrix.
- Suppose we have a 3x3 coefficient matrix. We might wish to augment it with a 3x3 identity matrix to yield a 3x6 new matrix.

$$\left[\begin{array}{ccc|ccc} a_{11} & a_{11} & a_{11} & 1 & 0 & 0 \\ a_{21} & a_{21} & a_{21} & 0 & 1 & 0 \\ a_{31} & a_{31} & a_{31} & 0 & 0 & 1 \end{array}\right]$$

Representing Linear Algebra

- Matrices provide a "concise notation" for representing and solving simultaneous linear equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix}$$

$$[A]\{x\} = \{b\}$$

Solving With MATLAB

- How to solve this linear algebraic equations in matrix form?

$$[A]\{x\} = \{b\}$$

Solving With MATLAB

- MATLAB provides two direct ways to solve systems of linear algebraic equations [A]{x}={b}:

  - Left-division (use backslash)

    ```
    x = A\b
    ```

  - Matrix inversion

    ```
    x = inv(A)*b
    ```

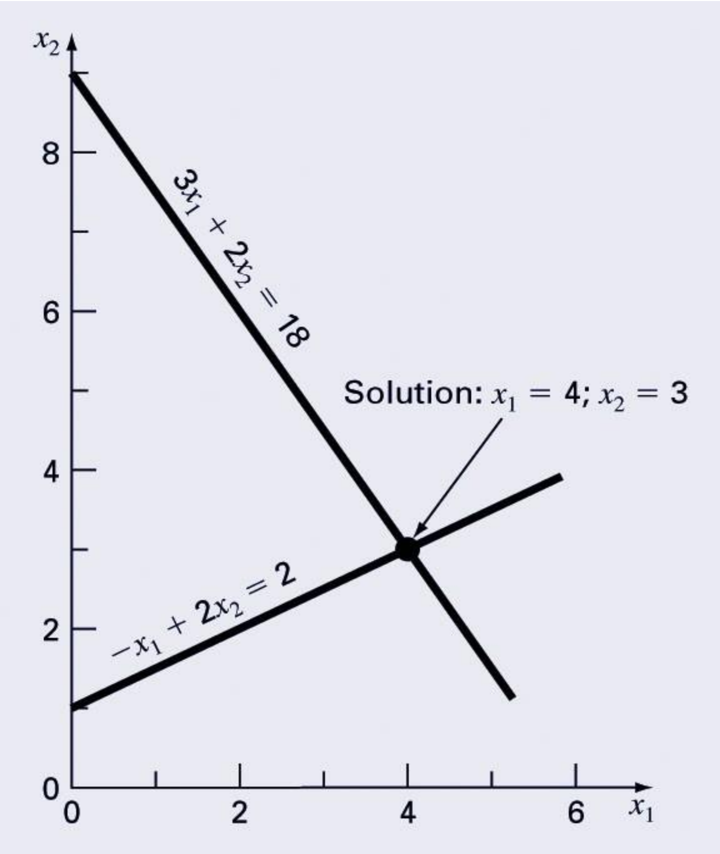
Try “help \” and “help /” in command window and compare their difference.

- The matrix inverse is less efficient than left-division and also only works for square, non-singular systems.

Graphical Method (chapter 9)

- For small sets of simultaneous equations, graphing them and determining the location of the intercept provides a solution.

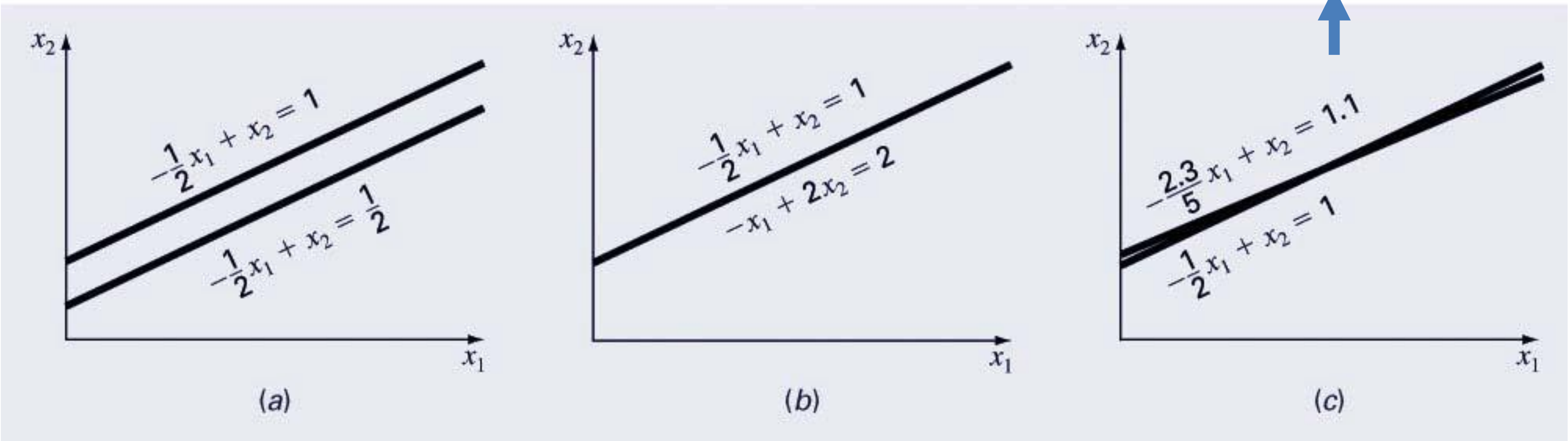
$$3x_1 + 2x_2 = 18$$
$$-x_1 + 2x_2 = 2$$



Graphical Method (cont)

- Graphing the equations can also show systems where:
  a) No solution exists
  b) Infinite solutions exist
  c) System is ill-conditioned

Posing potential problem in numerical solutions: Extremely sensitive to roundoff error (i.e. subtractive cancellation)



Determinants

- The *determinant D*=|*A*| of a matrix is calculated from the elements of [A].
- Determinants for small matrices are:

1×1	$\left\lvert a_{11} \right\rvert = a_{11}$
2×2	$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$
3×3	$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$

- Determinants for matrices larger than 3 x 3 can be very complicated.

Cramer's Rule

- ***Cramer's Rule*** states that each unknown in a system of linear algebraic equations may be expressed **as a fraction of two determinants** with denominator *D* and with the numerator obtained from *D* by replacing the column of coefficients of the unknown in question by the constants $b_1, b_2, \ldots, b_n$.

$$[A]\{x\} = \{b\}$$

- *b* is called "right-hand-side vector" in the textbook
- Proof for a 2x2 system is on page 253.

Cramer's Rule Example

- Find x_2 in the following system of equations:

$$
\begin{aligned}
0.3x_1 + 0.52x_2 + x_3 &= -0.01 \\
0.5x_1 + x_2 + 1.9x_3 &= 0.67 \\
0.1x_1 + 0.3x_2 + 0.5x_3 &= -0.44
\end{aligned}
$$

- Find the determinant *D*

$$D = \begin{vmatrix} 0.3 & 0.52 & 1 \\ 0.5 & 1 & 1.9 \\ 0.1 & 0.3 & 0.5 \end{vmatrix} = 0.3\begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} - 0.52\begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} + 1\begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.4 \end{vmatrix} = -0.0022$$

- Find determinant D_2 by replacing *D*'s second column with *b*

$$D_2 = \begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix} = 0.3\begin{vmatrix} 0.67 & 1.9 \\ -0.44 & 0.5 \end{vmatrix} - 0.01\begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} + 1\begin{vmatrix} 0.5 & 0.67 \\ 0.1 & -0.44 \end{vmatrix} = 0.0649$$

- Divide

$$x_2 = \frac{D_2}{D} = \frac{0.0649}{-0.0022} = -29.5$$

Naïve Gauss Elimination

- For larger systems, Cramer's Rule can become unwieldy.
- Instead, a sequential process of removing unknowns from equations using ***forward elimination*** followed by ***back substitution*** **may** be used - this is Gauss elimination.
- "Naïve" Gauss elimination simply means the process <u>does not check for potential problems resulting from division by zero</u>.

Naïve Gauss Elimination (p. 255)

- Forward elimination
  - Starting with the first row, add or subtract multiples of that row to eliminate the first coefficient from the second row and beyond.
  - Continue this process with the second row to remove the second coefficient from the third row and beyond.
  - Stop when an upper triangular matrix remains.
- Back substitution
  - Starting with the ***last* row**, solve for the unknown, then substitute that value into the next highest row.
  - Because of the upper-triangular nature of the matrix, each row will contain only one more unknown.

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]$$

$$\downarrow$$

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ & a'_{22} & a'_{23} & b'_2 \\ & & a''_{33} & b''_3 \end{array}\right]$$

(*a*) Forward elimination

$$\downarrow$$

$$x_3 = b''_3/a''_{33}$$

$$x_2 = (b'_2 - a'_{23}x_3)/a'_{22}$$

$$x_1 = (b_1 - a_{13}x_3 - a_{12}x_2)/a_{11}$$

(*b*) Back substitution

Naïve Gauss Elimination (p. 254)

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
\vdots \qquad\qquad \vdots & \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a'_{32}x_2 + a'_{33}x_3 + \cdots + a'_{3n}x_n &= b'_3 \\
\vdots \qquad\qquad \vdots & \\
a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n &= b'_n
\end{aligned}
$$

Forward elimination of unknowns:

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \frac{a_{21}}{a_{11}}a_{13}x_3 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1$$

This equation can be subtracted from Eq. (9.8*b*) to give

$$\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_n = b_2 - \frac{a_{21}}{a_{11}}b_1$$

$$a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2$$

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_3 \\
\vdots \qquad\qquad \vdots & \\
a''_{n3}x_3 + \cdots + a''_{nn}x_n &= b''_n
\end{aligned}
$$

Double prime means that the elements have been modified twice.

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n &= b'_2 \\
a''_{33}x_3 + \cdots + a''_{3n}x_n &= b''_3 \\
\ddots \qquad\qquad \vdots & \\
a_{nn}^{(n-1)}x_n &= b_n^{(n-1)}
\end{aligned}
$$

Naïve Gauss Elimination Program (p. 258)

```
function x = GaussNaive(A,b)
% GaussNaive: naive Gauss elimination
%   x = GaussNaive(A,b): Gauss elimination without pivoting.
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector

[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
nb = n+1;
Aug = [A b];
% forward elimination
for k = 1:n-1
  for i = k+1:n
    factor = Aug(i,k)/Aug(k,k);
    Aug(i,k:nb) = Aug(i,k:nb)-factor*Aug(k,k:nb);
  end
end
% back substitution
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
  x(i) = (Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

- Aug: augmented matrix
- Implement forward elimination steps on p. 254-256

Subtract multiples of the “first” row to eliminate the “first” coefficients of the remaining rows

- Calculate from the bottom

Gauss Elimination Efficiency (p. 260)

- The execution of Gauss elimination depends on the number of ***floating-point operations*** (or flops). The flop count for forward elimination of an *n* x *n* system is:

Outer Loop k	Inner Loop i	Addition/Subtraction Flops	Multiplication/Division Flops
1	2, n	$(n-1)(n)$	$(n-1)(n+1)$
2	3, n	$(n-2)(n-1)$	$(n-2)(n)$
$\vdots$	$\vdots$		
k	$k+1$, n	$(n-k)(n+1-k)$	$(n-k)(n+2-k)$
$\vdots$	$\vdots$		
$n-1$	n, n	$(1)(2)$	$(1)(3)$

Σ.... + Σ....

Forward Elimination	$\dfrac{2n^3}{3} + O(n^2)$
Back Substitution	$n^2 + O(n)$
Total	$\dfrac{2n^3}{3} + O(n^2)$

- Conclusions:
  - As the system gets larger, the computation time **increases rapidly**.
  - Most of the effort is incurred in the **elimination step**.

Partial Pivoting

- Problems arise with naïve Gauss elimination if a coefficient along the diagonal is 0 (problem: division by 0) or close to 0 (problem: round-off error)

$$
\begin{aligned}
2x_2 + 3x_3 &= 8 \\
4x_1 + 6x_2 + 7x_3 &= -3 \\
2x_1 - 3x_2 + 6x_3 &= 5
\end{aligned}
$$

- One way to combat these issues is to determine the coefficient with the largest absolute value in the column below the pivot element. The rows can then be **switched** so that the largest element is the **pivot element**. This is called ***partial pivoting.***

- If the columns as well as rows are searched for the largest element and then switched, this is called ***complete pivoting*** (rarely used).

Partial Pivoting Program (p. 263)

```
function x = GaussPivot(A,b)
% GaussPivot: Gauss elimination pivoting
%   x = GaussPivot(A,b): Gauss elimination with pivoting.
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector

[m,n]=size(A);
if m~=n, error('Matrix A must be square'); end
nb=n+1;
Aug=[A b];
% forward elimination
for k = 1:n-1
  % partial pivoting
  [big,i]=max(abs(Aug(k:n,k)));
  ipr=i+k-1;
  if ipr~=k
    Aug([k,ipr],:)=Aug([ipr,k],:);
  end
  for i = k+1:n
    factor=Aug(i,k)/Aug(k,k);
    Aug(i,k:nb)=Aug(i,k:nb)-factor*Aug(k,k:nb);
  end
end
% back substitution
x=zeros(n,1);
x(n)=Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
  x(i)=(Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

How to swap two rows?

Tridiagonal Systems

- A *tridiagonal* system is a banded system with a bandwidth of 3:

$$
\begin{bmatrix}
f_1 & g_1 & & & & & \\
e_2 & f_2 & g_2 & & & & \\
 & e_3 & f_3 & g_3 & & & \\
 & & \cdot & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \cdot & \\
 & & & & \cdot & \cdot\ \cdot & \\
 & & & & e_{n-1} & f_{n-1} & g_{n-1} \\
 & & & & & e_n & f_n
\end{bmatrix}
\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_{n-1} \\ x_n \end{Bmatrix}
=
\begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \cdot \\ \cdot \\ \cdot \\ r_{n-1} \\ r_n \end{Bmatrix}
$$

- Tridiagonal systems can be solved using the same method as Gauss elimination, but with much less effort because most of the matrix elements are already 0.

Tridiagonal System Solver

```
function x = Tridiag(e,f,g,r)
% Tridiag: Tridiagonal equation solver banded system
%   x = Tridiag(e,f,g,r): Tridiagonal system solver.
% input:
%   e = subdiagonal vector
%   f = diagonal vector
%   g = superdiagonal vector
%   r = right hand side vector
% output:
%   x = solution vector
n=length(f);
% forward elimination
for k = 2:n
  factor = e(k)/f(k-1);
  f(k) = f(k) - factor*g(k-1);
  r(k) = r(k) - factor*r(k-1);
end
% back substitution
x(n) = r(n)/f(n);
for k = n-1:-1:1
  x(k) = (r(k)-g(k)*x(k+1))/f(k);
end
```

Example 1 (Problem 8.9)

Five reactors are linked as shown. The **rate of mass flow** through each pipe is computed as the product of flow (Q) and concentration (c).

At steady state, the mass flow into and out of each reactor must be equal. For example, for the first reactor, a mass balance can be written as follows (see equation).

Write mass balances for the remaining reactors (see figure) and express the equations in matrix form. Then use MATLAB to solve for the concentrations in each reactor.

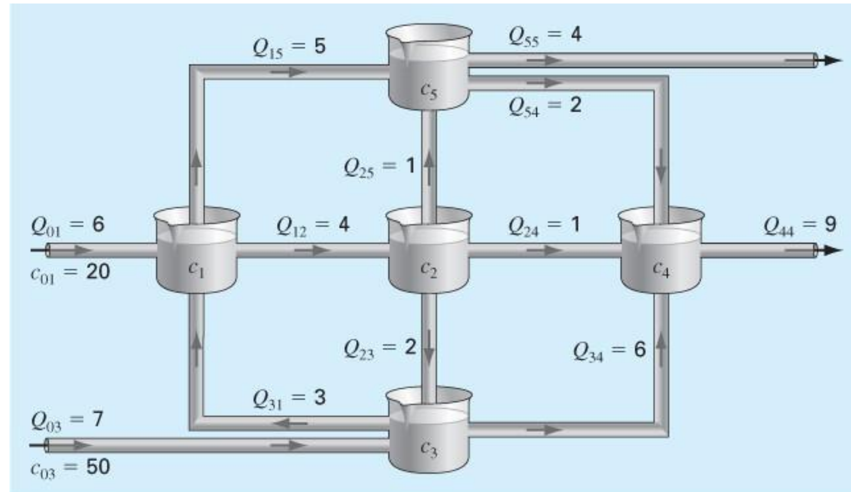


FIGURE P8.9

$$Q_{01}c_{01} + Q_{31}c_3 =$$
$$= Q_{15}c_1 + Q_{12}c_1$$

Example 1 (cont)

The mass balances can be written as

$$
\begin{aligned}
&(Q_{15}+Q_{12})c_1 + 0\cdot c_2 && -Q_{31}c_3 && +0\cdot c_4 && +0\cdot c_5 && = Q_{01}c_{01} \\
&-Q_{12}c_1 + (Q_{23}+Q_{24}+Q_{25})c_2 && +0\cdot c_3 && +0\cdot c_4 && +0\cdot c_5 && = 0 \\
&0\cdot c_1 - Q_{23}c_2 && +(Q_{31}+Q_{34})c_3 && +0\cdot c_4 && +0\cdot c_5 && = Q_{03}c_{03} \\
&0\cdot c_1 - Q_{24}c_2 && -Q_{34}c_3 && +Q_{44}c_4 && -Q_{54}c_5 && = 0 \\
&-Q_{15}c_1 - Q_{25}c_2 && +0\cdot c_3 && +0\cdot c_4 && +(Q_{54}+Q_{55})c_5 && = 0
\end{aligned}
$$

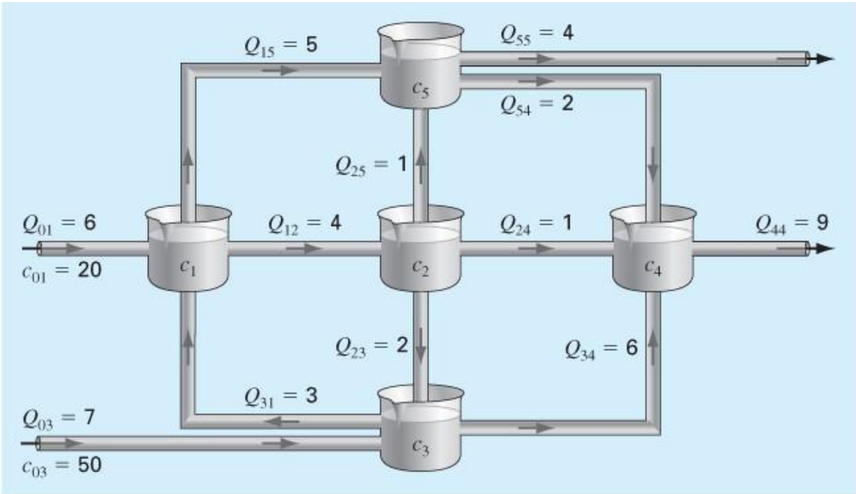


FIGURE P8.9

Example 1 (cont)

The mass balances can be written as

$$\begin{bmatrix} Q_{15}+Q_{12} & 0 & -Q_{31} & 0 & 0 \\ -Q_{12} & Q_{23}+Q_{24}+Q_{25} & 0 & 0 & 0 \\ 0 & -Q_{23} & Q_{31}+Q_{34} & 0 & 0 \\ 0 & -Q_{24} & -Q_{34} & Q_{44} & -Q_{54} \\ -Q_{15} & -Q_{25} & 0 & 0 & Q_{54}+Q_{55} \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{Bmatrix} = \begin{Bmatrix} Q_{01}c_{01} \\ 0 \\ Q_{03}c_{03} \\ 0 \\ 0 \end{Bmatrix}$$

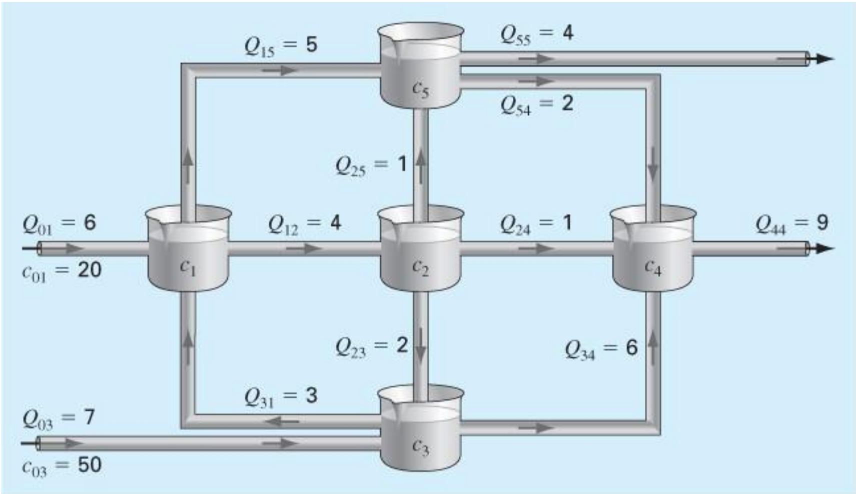


FIGURE P8.9

Example 1 (cont)

```
%
%   First, let define variable
%
q01=6; q03=7; q15=5; q12=4; q31=3; q25=1; q23=2; q55=4; q54=2; …
q24=1; q34=6; q44=9;
c01=20; c03=50;
Q = [q15+q12  0 -q31  0  0;
    -q12  q23+q24+q25  0  0  0;
     0 -q23  q31+q34  0  0;
     0 -q24 -q34  q44 -q54;
    -q15 -q25  0  0  q54+q55];
Qc = [q01*c01; 0; q03*c03; 0; 0];
%
%   Now solve for concentrations in each reactor
%
c = Q\Qc
%
%   Let check the solution
%
am_i_zero=Q*c-Qc
```

Example 2 (Problem 9.9)

Three reactors (see figure) are linked by pipes. As indicated, the **rate of transfer of chemicals** through each pipe is equal to a flow rate Q (m^3/s) multiplied by concentration c (mg/m^3) of the reactor from which the flow originates. If the system is at a steady state, the transfer into each reactor will balance the transfer out. Develop mass-balance equations for the reactors and solve the three simultaneous linear algebraic equations for their concentrations.

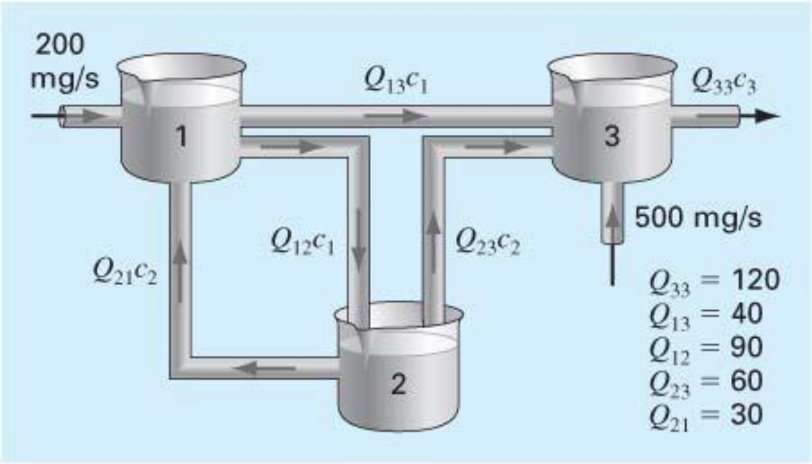
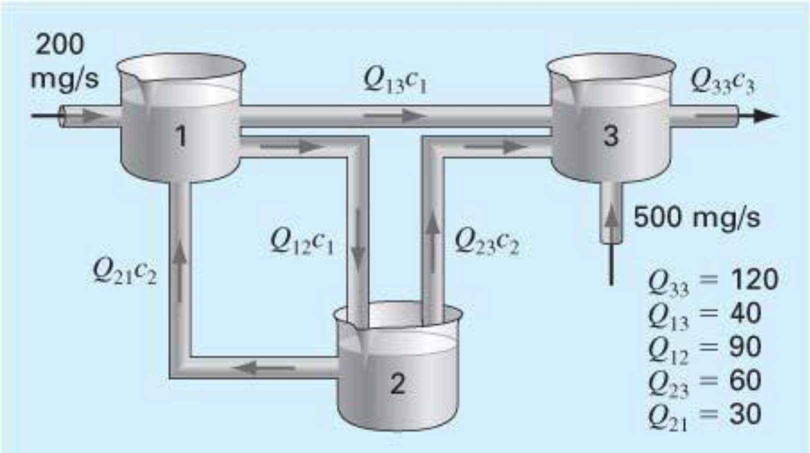


FIGURE P9.9
Three reactors linked by pipes. The rate of mass transfer through each pipe is equal to the product of flow Q and concentration c of the reactor from which the flow originates.

Example 2 (cont)

The mass balances can be written as

$$200 - Q_{13}c_1 - Q_{12}c_1 + Q_{21}c_2 = 0$$
$$Q_{12}c_1 - Q_{21}c_2 - Q_{23}c_2 = 0$$
$$500 + Q_{13}c_1 + Q_{23}c_2 - Q_{33}c_3 = 0$$



$$(Q_{13} + Q_{12})c_1 - Q_{21}c_2 = 200$$
$$Q_{12}c_1 - (Q_{21} + Q_{23})c_2 = 0$$
$$Q_{13}c_1 + Q_{23}c_2 - Q_{33}c_3 = -500$$

$$\begin{bmatrix} Q_{13} + Q_{12} & -Q_{21} & 0 \\ Q_{12} & -(Q_{21} + Q_{23}) & 0 \\ Q_{13} & Q_{23} & -Q_{33} \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} 200 \\ 0 \\ -500 \end{Bmatrix}$$

Example 2 (cont)

```
%
%   First, let define variable
%
q01=200; q03=500;
q33=120; q13=40; q12=90; q23=60; q21=30;
c01=20; c03=50;
Q = [q13+q12  -q21   0;
     q12  -(q21+q23) 0;
     q13  q23  -q33];
Qc = [q01; 0; q03];
%
%   Now solve for concentrations in each reactor
%
c = Q\Qc
%
%   Let check the solution
%
am_i_zero=Q*c-Qc
```

Example 3 (Problem 9.5)

Given the equation:

$$0.5x_1 - x_2 = -9.5$$

$$1.02x_1 - 2x_2 = -18.8$$

```
%
%   First, let define variable
%
a11=0.5; a12=-1;
a21=1.02; a22=-2;
d1=-9.5; d2=-18.8;
```

(a) Solve graphically

(b) Compute the determinant

(c) Considering (a) and (b), what would you expect regarding the system's condition

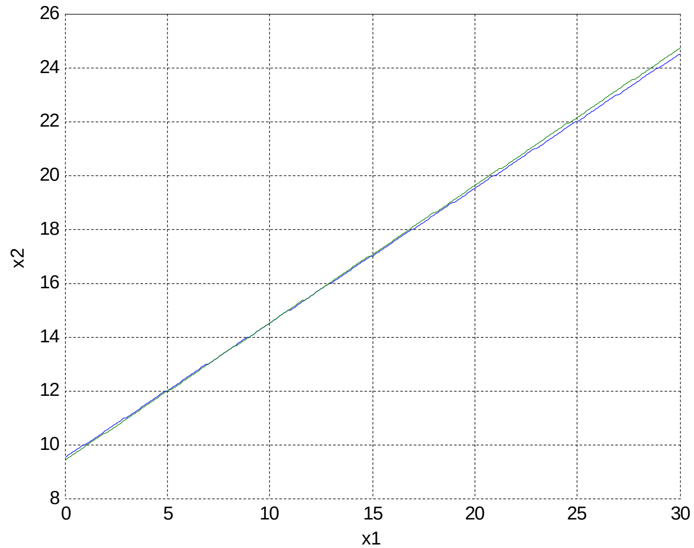
(d) Solve by elimination of unknowns

(e) Solve again but with a_{11} modified slightly to 0.52. Interpret your results.

Example 3 (cont)

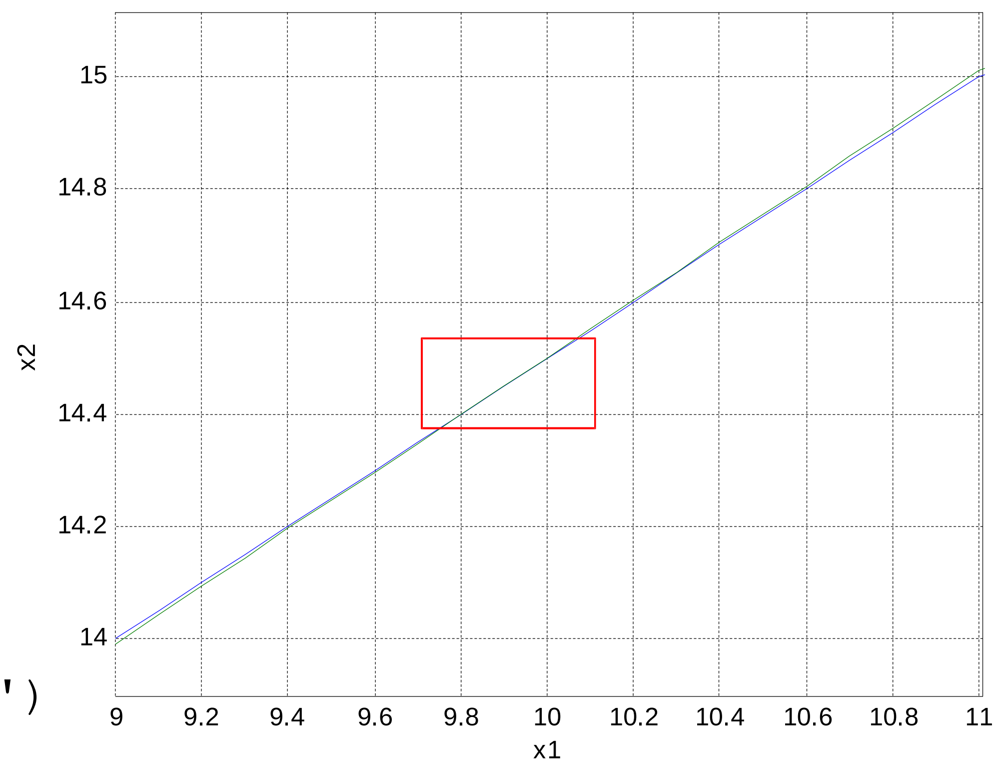
$$0.5x_1 - x_2 = -9.5$$
$$1.02x_1 - 2x_2 = -18.8$$

(a) Solve graphically



$$x_2 = 0.5x_1 + 9.5$$
$$x_2 = 0.51x_1 + 9.4$$

```
%
%   Graphical solution
%
x1=0:0.1:30;
x21=0.5*x1+9.5;
x22=0.51*x1+9.4;
  figure(1);
  plot(x1,x21,x1,x22); grid
  xlabel('x1'); ylabel ('x2')
```



Example 3 (cont)

(b) Compute the determinant

```
%
%   Determinant of A
%
A=[a11 a12;
    a21 a22]
detA=det(A)

A =

    0.5000   -1.0000
    1.0200   -2.0000


detA =

    0.0200
```

(c) Considering (a) and (b), what would you expect regarding the system's condition

Example 3 (cont)

(d) Solve by elimination of unknowns

$$x_2 = 0.5x_1 + 9.5$$
$$x_2 = 0.51x_1 + 9.4$$

$$0 = 0.01x_1 - 0.1$$
$$0.01x_1 = 0.1$$

$$x_1 = 10$$
$$x_2 = 0.5x_1 + 9.5 = 14.5$$

(e) Solve again but with a_{11} modified slightly to 0.52. Interpret your results

$$x_2 = \boxed{0.52x_1} + 9.5$$
$$x_2 = 0.51x_1 + 9.4$$

$$0 = 0.01x_1 + 0.1$$
$$0.01x_1 = -0.1$$

$$x_1 = -10$$
$$x_2 = 0.52x_1 + 9.5 = -4.3$$

Example 3 (cont)

(f) Solve in MATLAB

```
%
%   Solve Ax=D
%
x=A\D

x =

   10.0000
   14.5000

A =

    0.5000   -1.0000
    1.0200   -2.0000
```

(g) Investigate a bit more

```
%
%   Stability of the solution
%
a0=0.5 : 0.001 : 0.52;
n=length(a0);
x1=zeros(1,n); x2=zeros(1,n);
for i=1:n
    a11=a0(i);
    A=[a11 a12; a21 a22];
    foo=A\D;
    x1(1,i)=foo(1,1);
    x2(1,i)=foo(2,1);
end
figure(2);
plot(a0, x1), hold;
plot(a0, x2,'r'); grid;
xlabel('a11');
ylabel('Solution: x1 and x2')
```

A loop to add a small increment to a11!