Solving Linear Systems of Equations

Gerald Recktenwald
Portland State University
Mechanical Engineering Department
gerry@me.pdx.edu

Version 0.88 August 22, 2006

Primary Topics

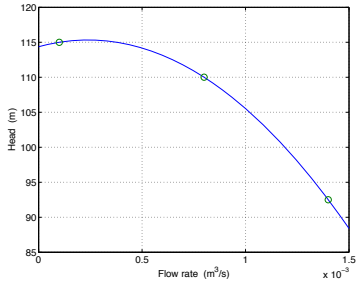
- Basic Concepts
- Gaussian Elimination
- Limitations on Numerical Solutions to $Ax = b$
- Factorization Methods
- Nonlinear Systems of Equations

Basic Concepts

- Matrix Formulation
- Requirements for a Solution
  - ➢ Consistency
  - ➢ The Role of rank(A)
  - ➢ Formal Solution when A is $n \times n$

Pump Curve Model (1)

Objective: Find the coefficients of the quadratic equation that approximates the pump curve data.



Model equation:

$$h = c_1 q^2 + c_2 q + c_3$$

Write the model equation for three points on the curve. This gives three equations for the three unknowns c_1, c_2, and c_3.

Pump Curve Model (2)

Points from the pump curve:

$q\,(\mathrm{m}^3/\mathrm{s})$	1×10^{-4}	8×10^{-4}	1.4×10^{-3}
$h\,(\mathrm{m})$	115	110	92.5

Substitute each pair of data points into the model equation

$$\begin{aligned} 115 &= 1 \times 10^{-8}\, c_1 + 1 \times 10^{-4}\, c_2 + c_3, \\ 110 &= 64 \times 10^{-8}\, c_1 + 8 \times 10^{-4}\, c_2 + c_3, \\ 92.5 &= 196 \times 10^{-8}\, c_1 + 14 \times 10^{-4}\, c_2 + c_3, \end{aligned}$$

Rewrite in matrix form as

$$\begin{bmatrix} 1 \times 10^{-8} & 1 \times 10^{-4} & 1 \\ 64 \times 10^{-8} & 8 \times 10^{-4} & 1 \\ 196 \times 10^{-8} & 14 \times 10^{-4} & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 115 \\ 110 \\ 92.5 \end{bmatrix}.$$

Pump Curve Model (3)

Using more compact symbolic notation

$$Ax = b$$

where

$$A = \begin{bmatrix} 1 \times 10^{-8} & 1 \times 10^{-4} & 1 \\ 64 \times 10^{-8} & 8 \times 10^{-4} & 1 \\ 196 \times 10^{-8} & 14 \times 10^{-4} & 1 \end{bmatrix},$$

$$x = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}, \qquad b = \begin{bmatrix} 115 \\ 110 \\ 92.5 \end{bmatrix}.$$

Pump Curve Model (4)

In general, for any three (q, h) pairs the system is still $Ax = b$ with

$$A = \begin{bmatrix} q_1^2 & q_1 & 1 \\ q_2^2 & q_2 & 1 \\ q_3^2 & q_3 & 1 \end{bmatrix}, \quad x = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}, \quad b = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}.$$

Matrix Formulation

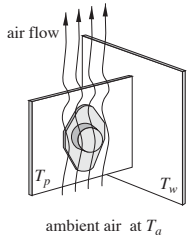
Recommended Procedure

1. Write the equations in natural form.
2. Identify unknowns, and order them.
3. Isolate the unknowns.
4. Write equations in matrix form.

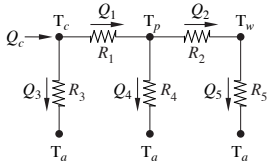
Thermal Model of an IC Package (1)

Objective: Find the temperature of an integrated circuit (IC) package mounted on a heat spreader. The system of equations is obtained from a thermal resistive network model.

Physical Model:



Mathematical Model:



Thermal Model of an IC Package (2)

1. Write the equations in natural form:

Use resistive model of heat flow between nodes to get

$$Q_1 = \frac{1}{R_1}(T_c - T_p) \qquad Q_2 = \frac{1}{R_2}(T_p - T_w)$$

$$Q_3 = \frac{1}{R_3}(T_c - T_a) \qquad Q_4 = \frac{1}{R_4}(T_p - T_a)$$

$$Q_2 = \frac{1}{R_5}(T_w - T_a)$$

$$Q_c = Q_1 + Q_3 \qquad Q_1 = Q_2 + Q_4$$

Thermal Model of an IC Package (3)

2. Identify unknowns, and order them:

Unknowns are Q_1, Q_2, Q_3, Q_4, T_c, T_p, and T_w. Thus,

$$x = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ T_c \\ T_p \\ T_w \end{bmatrix}.$$

3. Isolate the Unknowns

$$\begin{aligned} R_1Q_1 - T_c + T_p &= 0, \\ R_2Q_2 - T_p + T_w &= 0, \\ R_3Q_3 - T_c &= -T_a, \\ R_4Q_4 - T_p &= -T_a, \\ R_5Q_2 - T_w &= -T_a, \\ Q_1 + Q_3 &= Q_c, \\ Q_1 - Q_2 - Q_4 &= 0. \end{aligned}$$

Thermal Model of an IC Package (4)

4. Write in Matrix Form

$$\begin{bmatrix} R_1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & R_2 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & R_3 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & R_4 & 0 & -1 & 0 \\ 0 & R_5 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ T_c \\ T_p \\ T_w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -T_a \\ -T_a \\ -T_a \\ Q_c \\ 0 \end{bmatrix}$$

Note: The coefficient matrix has many more zeros than non-zeros. This is an example of a *sparse* matrix.

Requirements for a Solution

1. Consistency
2. The Role of $\mathrm{rank}(A)$
3. Formal Solution when A is $n \times n$
4. Summary

Consistency (1)

If an exact solution to $Ax = b$ exists, b must lie in the column space of A. If it does, then the system is said to be **consistent**.

If the system is consistent, an exact solution exists.

Consistency (2)

$\mathrm{rank}(A)$ gives the number of linearly independent columns in A

$[A\ b]$ is the *augmented* matrix formed by combining the b vector with the columns of A.

$$[A\ \ b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1,n} & b_1 \\ a_{21} & a_{22} & & a_{2,n} & b_2 \\ \vdots & & \ddots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{n,n} & b_n \end{array}\right]$$

If $\mathrm{rank}([A\ b]) > \mathrm{rank}(A)$ then b does not lie in the column space of A. In other words, since $[A\ b]$ has a larger set of basis vectors than A, and since the difference in the size of the basis set is solely due to the b vector, the b vector cannot be constructed from the column vectors of A.

Role of $\mathrm{rank}(A)$

- If A is $m \times n$, and z is an n-element column vector, then $Az = 0$ has a nontrivial solution only if the columns of A are linearly dependent[1].
- In other words, the *only* solution to $Az = 0$ is $z = 0$ when A is full rank.
- Given the $m \times n$ matrix A, the system $Ax = b$ has a unique solution if the system is consistent and if $\mathrm{rank}(A) = n$.

[1] "0" is the m-element column vector filled with zeros

Summary of Solution to $Ax = b$ where A is $m \times n$

For the general case where A is $m \times n$ and $m \geq n$,

- If $\mathrm{rank}(A) = n$ and the system is consistent, the solution exists and it is unique.
- If $\mathrm{rank}(A) = n$ and the system is inconsistent, no solution exists.
- If $\mathrm{rank}(A) < n$ and the system is consistent, an infinite number of solutions exist.

If A is $n \times n$ and $\mathrm{rank}(A) = n$, then the system is consistent and the solution is unique.

Formal Solution when A is $n \times n$

The *formal solution* to $Ax = b$ is

$$x = A^{-1}b$$

where A is $n \times n$.

If A^{-1} exists then A is said to be **nonsingular**.

If A^{-1} does not exist then A is said to be **singular**.

Formal Solution when A is $n \times n$

If A^{-1} exists then

$$Ax = b \qquad \Longrightarrow \qquad x = A^{-1}b$$

but

Do not compute the solution to $Ax = b$ by finding A^{-1}, and then multiplying b by A^{-1}!

We see: $x = A^{-1}b$

We do: **Solve $Ax = b$ by Gaussian elimination or an equivalent algorithm**

Singularity of A

If an $n \times n$ matrix, A, is **singular** then

- ➢ the columns of A are linearly dependent
- ➢ the rows of A are linearly dependent
- ➢ $\mathrm{rank}(A) < n$
- ➢ $\det(A) = 0$
- ➢ A^{-1} does not exist
- ➢ a solution to $Ax = b$ may not exist
- ➢ If a solution to $Ax = b$ exists, it is not unique

Summary of Requirements for Solution of $Ax = b$

Given the $n \times n$ matrix A and the $n \times 1$ vector, b

- the solution to $Ax = b$ exists and is unique for any b if and only if $\mathrm{rank}(A) = n$.
- $\mathrm{rank}(A) = n$ automatically guarantees that the system is consistent.

Gaussian Elimination

- Solving Diagonal Systems
- Solving Triangular Systems
- Gaussian Elimination Without Pivoting
  - ▷ Hand Calculations
  - ▷ Cartoon Version
  - ▷ The Algorithm
- Gaussian Elimination with Pivoting
  - ▷ Row or Column Interchanges, or Both
  - ▷ Implementation
- Solving Systems with the Backslash Operator

Solving Diagonal Systems (1)

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

Solving Diagonal Systems (1)

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

is equivalent to

$$\begin{aligned} x_1 &= -1 \\ 3x_2 &= 6 \\ 5x_3 &= -15 \end{aligned}$$

Solving Diagonal Systems (1)

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

is equivalent to

$$\begin{aligned} x_1 &= -1 \\ 3x_2 &= 6 \\ 5x_3 &= -15 \end{aligned}$$

The solution is

$$x_1 = -1 \qquad x_2 = \frac{6}{3} = 2 \qquad x_3 = \frac{-15}{5} = -3$$

Solving Diagonal Systems (2)

Algorithm 8.1

given A, b
for $i = 1 \ldots n$
  $x_i = b_i / a_{i,i}$
end

In MATLAB**:**

```
>> A = ...          %  A is a diagonal matrix
>> b = ...
>> x = b./diag(A)
```

This is the *only* place where element-by-element division (`.*`) has anything to do with solving linear systems of equations.

Triangular Systems (1)

The generic lower and upper triangular matrices are

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & & \cdots & l_{nn} \end{bmatrix}$$

and

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & u_{nn} \end{bmatrix}$$

The triangular systems

$$Ly = b \qquad Ux = c$$

are easily solved by **forward substitution** and **backward substitution**, respectively

Solving Triangular Systems (2)

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

Solving Triangular Systems (3)

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

Solving Triangular Systems (4)

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

Solve in backward order (last equation is solved first)

$$x_3 = \frac{8}{4} = 2$$

Solving Triangular Systems (5)

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

Solve in backward order (last equation is solved first)

$$x_3 = \frac{8}{4} = 2 \qquad x_2 = \frac{1}{3}\left(-1 + 2x_3\right) = \frac{3}{3} = 1$$

Solving Triangular Systems (6)

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

Solve in backward order (last equation is solved first)

$$x_3 = \frac{8}{4} = 2 \qquad x_2 = \frac{1}{3}(-1 + 2x_3) = \frac{3}{3} = 1$$

$$x_1 = \frac{1}{-2}(9 - x_2 - 2x_3) = \frac{4}{-2} = -2$$

Solving Triangular Systems (7)

Solving for $x_n, x_{n-1}, \ldots, x_1$ for an upper triangular system is called **backward substitution**.

Algorithm 8.2

given U, b
$x_n = b_n / u_{nn}$
for $i = n - 1 \ldots 1$
  $s = b_i$
  for $j = i + 1 \ldots n$
    $s = s - u_{i,j} x_j$
  end
  $x_i = s / u_{i,i}$
end

Solving Triangular Systems (8)

Solving for $x_1, x_2, \ldots, x_n$ for a lower triangular system is called **forward substitution**.

Algorithm 8.3

given L, b
$x_1 = b_1 / \ell_{11}$
for $i = 2 \ldots n$
  $s = b_i$
  for $j = 1 \ldots i - 1$
    $s = s - \ell_{i,j} x_j$
  end
  $x_i = s / \ell_{i,i}$
end

Using forward or backward substitution is sometimes referred to as performing a **triangular solve**.

Gaussian Elimination

Goal is to transform an arbitrary, square system into the equivalent upper triangular system so that it may be easily solved with backward substitution.

The *formal solution* to $Ax = b$, where A is an $n \times n$ matrix is

$$x = A^{-1} b$$

In MATLAB**:**

```
>> A = ...
>> b = ...
>> x = A\b
```

Gaussian Elimination — Hand Calculations (1)

Solve

$$x_1 + 3x_2 = 5$$
$$2x_1 + 4x_2 = 6$$

Subtract 2 times the first equation from the second equation

$$x_1 + 3x_2 = 5$$
$$-2x_2 = -4$$

This equation is now in triangular form, and can be solved by backward substitution.

Gaussian Elimination — Hand Calculations (2)

The elimination phase transforms the matrix and right hand side to an equivalent system

$$\begin{aligned} x_1 + 3x_2 &= 5 \\ 2x_1 + 4x_2 &= 6 \end{aligned} \quad \longrightarrow \quad \begin{aligned} x_1 + 3x_2 &= 5 \\ -2x_2 &= -4 \end{aligned}$$

The two systems have the same solution. The right hand system is upper triangular.

Solve the second equation for x_2

$$x_2 = \frac{-4}{-2} = 2$$

Substitute the newly found value of x_2 into the first equation and solve for x_1.

$$x_1 = 5 - (3)(2) = -1$$

Gaussian Elimination — Hand Calculations (3)

When performing Gaussian Elimination by hand, we can avoid copying the x_i by using a shorthand notation.

For example, to solve:

$$A = \begin{bmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ -7 \\ -6 \end{bmatrix}$$

Form the *augmented* system

$$\tilde{A} = [A \;\; b] = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 6 & -6 & 7 & -7 \\ 3 & -4 & 4 & -6 \end{array}\right]$$

The vertical bar inside the augmented matrix is just a reminder that the last column is the b vector.

Gaussian Elimination — Hand Calculations (4)

Add 2 times row 1 to row 2, and add (1 times) row 1 to row 3

$$\tilde{A}_{(1)} = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{array}\right]$$

Subtract (1 times) row 2 from row 3

$$\tilde{A}_{(2)} = \left[\begin{array}{ccc|c} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{array}\right]$$

Gaussian Elimination — Hand Calculations (5)

The transformed system is now in upper triangular form

$$\tilde{A}_{(2)} = \left[\begin{array}{rrr|r} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{array}\right]$$

Solve by back substitution to get

$$x_3 = \frac{2}{-2} = -1$$

$$x_2 = \frac{1}{-2}\left(-9 - 5x_3\right) = 2$$

$$x_1 = \frac{1}{-3}\left(-1 - 2x_2 + x_3\right) = 2$$

Gaussian Elimination — Cartoon Version (1)

Start with the augmented system

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

The xs represent numbers, they are not necessarily the same values.

Begin elimination using the first row as the *pivot row* and the first element of the first row as the pivot element

$$\begin{bmatrix} \boxed{x} & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

Gaussian Elimination — Cartoon Version (2)

Eliminate elements under the pivot element in the first column. x' indicates a value that has been changed once.

$$\begin{bmatrix} \boxed{x} & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \longrightarrow \begin{bmatrix} \boxed{x} & x & x & x & x \\ 0 & x' & x' & x' & x' \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \boxed{x} & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \\ x & x & x & x & x \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \boxed{x} & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \end{bmatrix}$$

Gaussian Elimination — Cartoon Version (3)

The pivot element is now the diagonal element in the second row. Eliminate elements under the pivot element in the second column. x'' indicates a value that has been changed twice.

$$\begin{bmatrix} x & x & x & x & x \\ 0 & \boxed{x'} & x' & x' & x' \\ 0 & x' & x' & x' & x' \\ 0 & x' & x' & x' & x' \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & \boxed{x'} & x' & x' & x' \\ 0 & 0 & x'' & x'' & x'' \\ 0 & x' & x' & x' & x' \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & \boxed{x'} & x' & x' & x' \\ 0 & 0 & x'' & x'' & x'' \\ 0 & 0 & x'' & x'' & x'' \end{bmatrix}$$

Gaussian Elimination — Cartoon Version (4)

The pivot element is now the diagonal element in the third row. Eliminate elements under the pivot element in the third column. x''' indicates a value that has been changed three times.

$$\begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & \boxed{x''} & x'' & x'' \\ 0 & 0 & x'' & x'' & x'' \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & \boxed{x''} & x'' & x'' \\ 0 & 0 & 0 & x''' & x''' \end{bmatrix}$$

Gaussian Elimination — Cartoon Version (5)

Summary

- Gaussian Elimination is an orderly process of transforming an augmented matrix into an equivalent upper triangular form.
- The elimination operation is

$$\tilde{a}_{kj} = \tilde{a}_{kj} - (\tilde{a}_{ki}/\tilde{a}_{ii})\tilde{a}_{ij}$$

- Elimination requires three nested loops.
- The result of the elimination phase is represented by the image below.

$$\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \longrightarrow \begin{bmatrix} x & x & x & x & x \\ 0 & x' & x' & x' & x' \\ 0 & 0 & x'' & x'' & x'' \\ 0 & 0 & 0 & x''' & x''' \end{bmatrix}$$

Gaussian Elimination Algorithm

Algorithm 8.4

form $\tilde{A} = [A \;\; b]$

for $i = 1 \ldots n-1$
  for $k = i+1 \ldots n$
    for $j = i \ldots n+1$
      $\tilde{a}_{kj} = \tilde{a}_{kj} - (\tilde{a}_{ki}/\tilde{a}_{ii})\tilde{a}_{ij}$
    end
  end
end

GEshow: The `GEshow` function in the NMM toolbox uses Gaussian elimination to solve a system of equations. `GEshow` is intended for demonstration purposes only.

The Need for Pivoting (1)

Solve:

$$A = \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix} \qquad b = \begin{bmatrix} -4 \\ 5 \\ 7 \\ 7 \end{bmatrix}$$

Note that there is nothing "wrong" with this system. A is full rank. The solution exists and is unique.

Form the augmented system.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 1 & 2 & 4 & -3 & 5 \\ -3 & -3 & 8 & -2 & 7 \\ -1 & 1 & 6 & -3 & 7 \end{array}\right]$$

The Need for Pivoting (2)

Subtract $1/2$ times the first row from the second row,
add $3/2$ times the first row to the third row,
add $1/2$ times the first row to the fourth row.

The result of these operations is:

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 0 & 5 & -2 & 7 \\ 0 & 3 & 5 & -5 & 1 \\ 0 & 3 & 5 & -4 & 5 \end{array}\right]$$

The *next* stage of Gaussian elimination will not work because there is a zero in the *pivot* location, $\tilde{a}_{22}$.

The Need for Pivoting (3)

Swap second and fourth rows of the augmented matrix.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 3 & 5 & -4 & 5 \\ 0 & 3 & 5 & -5 & 1 \\ 0 & 0 & 5 & -2 & 7 \end{array}\right]$$

Continue with elimination: subtract (1 times) row 2 from row 3.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 3 & 5 & -4 & 5 \\ 0 & 0 & 0 & -1 & -4 \\ 0 & 0 & 5 & -2 & 7 \end{array}\right]$$

The Need for Pivoting (4)

Another zero has appear in the pivot position. Swap row 3 and row 4.

$$\left[\begin{array}{cccc|c} 2 & 4 & -2 & -2 & -4 \\ 0 & 3 & 5 & -4 & 5 \\ 0 & 0 & 5 & -2 & 7 \\ 0 & 0 & 0 & -1 & -4 \end{array}\right]$$

The augmented system is now ready for backward substitution.

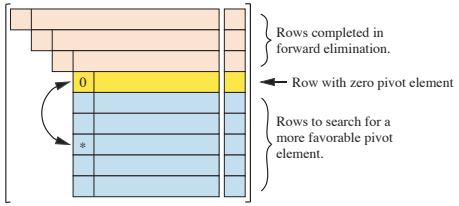
Pivoting Strategies

Partial Pivoting: Exchange only rows

- ➢ Exchanging rows does not affect the order of the x_i
- ➢ For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the partial column below the pivot element.
- ➢ Partial pivoting is usually sufficient.

Partial Pivoting

To avoid division by zero, swap the row having the zero pivot with one of the rows below it.



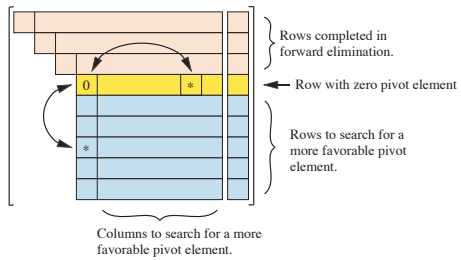
To minimize the effect of roundoff, always choose the row that puts the largest pivot element on the diagonal, i.e., find i_p such that $|a_{i_p,i}| = \max(|a_{k,i}|)$ for $k = i, \ldots, n$

Pivoting Strategies (2)

Full (or Complete) Pivoting: Exchange *both* rows and columns

- Column exchange requires changing the order of the x_i
- For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the pivot row, *and* in all rows below the pivot row, starting the pivot column.
- Full pivoting is less susceptible to roundoff, but the increase in stability comes at a cost of more complex programming (not a problem if you use a library routine) and an increase in work associated with searching and data movement.

Full Pivoting



Gauss Elimination with Partial Pivoting

Algorithm 8.5

form $\tilde{A} = [A \;\; b]$
for $i = 1 \ldots n-1$
  find i_p such that
    $\max(|\tilde{a}_{i_p i}|) \geq \max(|\tilde{a}_{ki}|)$ for $k = i \ldots n$
  exchange row i_p with row i
  for $k = i+1 \ldots n$
    for $j = i \ldots n+1$
      $\tilde{a}_{k,j} = \tilde{a}_{k,j} - (\tilde{a}_{k,i}/\tilde{a}_{i,i})\tilde{a}_{i,j}$
    end
  end
end

Gauss Elimination with Partial Pivoting

GEshow: The `GEshow` function in the NMM toolbox uses naive Gaussian elimination without pivoting to solve a system of equations.

GEpivshow: The `GEpivshow` function in the NMM toolbox uses Gaussian elimination with partial pivoting to solve a system of equations. `GEpivshow` is intended for demonstration purposes only.

`GEshow` and `GEpivshow` are for demonstration purposes only. They are also a convenient way to check your hand calculations.

The Backslash Operator (1)

Consider the scalar equation

$$5x = 20 \qquad \Longrightarrow \qquad x = (5)^{-1}20$$

The extension to a system of equations is, of course

$$Ax = b \qquad \Longrightarrow \qquad x = A^{-1}b$$

where $A^{-1}b$ is the formal solution to $Ax = b$

In MATLAB notation the system is solved with

```
x = A\b
```


The Backslash Operator (2)

Given an $n \times n$ matrix A, and an $n \times 1$ vector b the \ operator performs a sequence of tests on the A matrix. MATLAB attempts to solve the system with the method that gives the least roundoff and the fewest operations.

When A is an $n \times n$ matrix:

1. MATLAB examines A to see if it is a permutation of a triangular system
   If so, the appropriate triangular solve is used.

2. MATLAB examines A to see if it *appears* to be symmetric and positive definite.
   If so, MATLAB attempts a Cholesky factorization
   and two triangular solves.

3. If the Cholesky factorization fails, or if A does not appear to be symmetric,
   MATLAB attempts an LU factorization
   and two triangular solves.

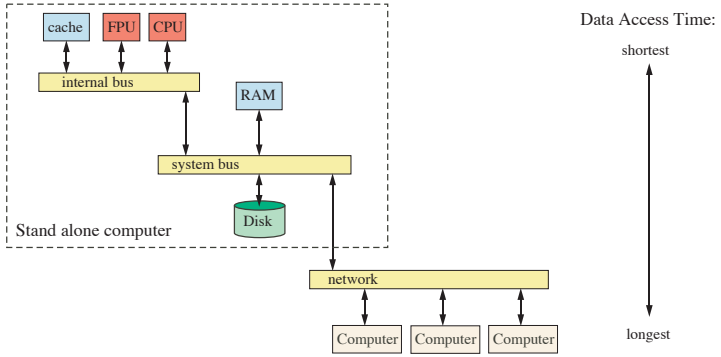
Limits on Numerical Solution to $Ax = b$

Machine Limitations

- RAM requirements grow as $\mathcal{O}(n^2)$
- flop count grows as $\mathcal{O}(n^3)$
- The time for data movement is an important speed bottleneck on modern systems

Computer Architecture Affecting the Speed of Data Access

(a highly simplified view)



Limits on Numerical Solution to $Ax = b$

Limits of Floating Point Arithmetic

- Exact singularity
- Effect of perturbations to b
- Effect of perturbations to A
- The condition number

Geometric Interpretation of Singularity (1)

Consider a 2×2 system describing two lines that intersect

$$y = -2x + 6$$
$$y = \frac{1}{2}x + 1$$

The matrix form of this equation is

$$\begin{bmatrix} 2 & 1 \\ -1/2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 1 \end{bmatrix}$$

The equations for two **parallel** but **not intersecting** lines are

$$\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \end{bmatrix}$$

Here the coefficient matrix is singular ($\mathrm{rank}(A) = 1$), and the system is inconsistent

Geometric Interpretation of Singularity (2)

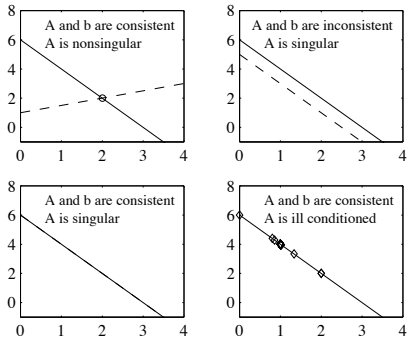
The equations for two **parallel** and **coincident** lines are

$$\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

The equations for two **nearly parallel** lines are

$$\begin{bmatrix} 2 & 1 \\ 2+\delta & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6+\delta \end{bmatrix}$$

Geometric Interpretation of Singularity (3)



Effect of Perturbations to b

Consider the solution of a 2×2 system where

$$b = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}$$

One expects that the *exact* solutions to

$$Ax = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix} \qquad \text{and} \qquad Ax = \begin{bmatrix} 1 \\ 0.6667 \end{bmatrix}$$

will be different. Should these solutions be a **lot different** or a **little different**?

Effect of Perturbations to b

Perturb b with δb such that

$$\frac{\|\delta b\|}{\|b\|} \ll 1,$$

The perturbed system is

$$A(x + \delta x_b) = b + \delta b$$

The perturbations satisfy

$$A\delta x_b = \delta b$$

Analysis shows (see next two slides for proof) that

$$\frac{\|\delta x_b\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x_b\|}{\|x\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Effect of Perturbations to b (Proof)

Let $x + \delta x_b$ be the *exact* solution to the perturbed system

$$A(x + \delta x_b) = b + \delta b \tag{1}$$

Expand

$$Ax + A\delta x_b = b + \delta b$$

Subtract Ax from left side and b from right side since $Ax = b$

$$A\delta x_b = \delta b$$

Left multiply by A^{-1}

$$\delta x_b = A^{-1}\delta b \tag{2}$$

Effect of Perturbations to b (Proof, p. 2)

Take norm of equation (2)

$$\|\delta x_b\| = \|A^{-1}\,\delta b\|$$

Applying consistency requirement of matrix norms

$$\|\delta x\| \leq \|A^{-1}\|\|\delta b\| \tag{3}$$

Similarly, $Ax = b$ gives $\|b\| = \|Ax\|$, and

$$\|b\| \leq \|A\|\|x\| \tag{4}$$

Rearrangement of equation (4) yields

$$\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|} \tag{5}$$

Effect of Perturbations to b (Proof)

Multiply Equation (4) by Equation (3) to get

$$\frac{\|\delta x_b\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|} \tag{6}$$

Summary:

If $x + \delta x_b$ is the *exact* solution to the perturbed system

$$A(x + \delta x_b) = b + \delta b$$

then

$$\frac{\|\delta x_b\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

Effect of Perturbations to A

Perturb A with δA such that

$$\frac{\|\delta A\|}{\|A\|} \ll 1,$$

The perturbed system is

$$(A + \delta A)(x + \delta x_A) = b$$

Analysis shows that

$$\frac{\|\delta x_A\|}{\|x + \delta x_A\|} \leq \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x_A\|}{\|x + \delta x_A\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Effect of Perturbations to both A and b

Perturb both A with δA and b with δb such that

$$\frac{\|\delta A\|}{\|A\|} \ll 1 \quad \text{and} \quad \frac{\|\delta b\|}{\|b\|} \ll 1$$

The perturbation satisfies

$$(A + \delta A)(x + \delta x) = b + \delta b$$

Analysis shows that

$$\frac{\|\delta x\|}{\|x + \delta x\|} \quad \leq \quad \frac{\|A\|\|A^{-1}\|}{1 - \|A\|\|A^{-1}\|\frac{\|\delta A\|}{\|A\|}}\left[\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right]$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x\|}{\|x + \delta x\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Condition number of A

The **condition number**

$$\kappa(A) \equiv \|A\|\|A^{-1}\|$$

indicates the sensitivity of the solution to perturbations in A and b. The condition number can be measured with any p-norm.

The condition number is always in the range

$$1 \leq \kappa(A) \leq \infty$$

- $\kappa(A)$ is a mathematical property of A
- *Any* algorithm will produce a solution that is sensitive to perturbations in A and b if $\kappa(A)$ is large.
- In exact math a matrix is either singular or non-singular. $\kappa(A) = \infty$ for a singular matrix
- $\kappa(A)$ indicates how close A is to being *numerically* singular.
- A matrix with large κ is said to be **ill-conditioned**

Computational Stability

In Practice, applying Gaussian elimination with partial pivoting and back substitution to $Ax = b$ gives the **exact solution**, $\hat{x}$, to the **nearby problem**

$$(A + E)\hat{x} = b \quad \text{where} \quad \|E\|_\infty \leq \varepsilon_m \|A\|_\infty$$

> Gaussian elimination with partial pivoting and back substitution "gives exactly the right answer to nearly the right question."
>
> — Trefethen and Bau

Computational Stability

An algorithm that gives the exact answer to a problem that is near to the original problem is said to be **backward stable**. Algorithms that are not backward stable will tend to amplify roundoff errors present in the original data. As a result, the solution produced by an algorithm that is not backward stable will not necessarily be the solution to a problem that is close to the original problem.

Gaussian elimination without partial pivoting is *not* backward stable for arbitrary A. If A is symmetric and positive definite, then Gaussian elimination without pivoting in backward stable.

The Residual

Let $\hat{x}$ be the numerical solution to $Ax = b$. $\hat{x} \neq x$ (x is the exact solution) because of roundoff.

The **residual** measures how close $\hat{x}$ is to satisfying the original equation

$$r = b - A\hat{x}$$

It is not hard to show that

$$\frac{\|\hat{x} - x\|}{\|\hat{x}\|} \leq \kappa(A)\frac{\|r\|}{\|b\|}$$

Small $\|r\|$ does not guarantee a small $\|\hat{x} - x\|$.

If $\kappa(A)$ is large the $\hat{x}$ returned by Gaussian elimination and back substitution (or any other solution method) is not guaranteed to be anywhere near the true solution to $Ax = b$.

Rules of Thumb (1)

- Applying Gaussian elimination with partial pivoting and back substitution to $Ax = b$ yields a numerical solution $\hat{x}$ such that the residual vector $r = b - A\hat{x}$ is small *even if* the $\kappa(A)$ is large.
- If A and b are stored to machine precision ε_m, the numerical solution to $Ax = b$ by any variant of Gaussian elimination is correct to d digits where

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A))$$

Rules of Thumb (2)

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A))$$

Example:

MATLAB computations have $\varepsilon_m \approx 2.2 \times 10^{-16}$. For a system with $\kappa(A) \sim 10^{10}$ the elements of the solution vector will have

$$\begin{aligned} d &= |\log_{10}(2.2 \times 10^{-16})| - \log_{10}\left(10^{10}\right) \\ &= 16 - 11 \\ &= 5 \end{aligned}$$

correct digits

Summary of Limits to Numerical Solution of $Ax = b$

1. $\kappa(A)$ indicates how close A is to being numerically singular
2. If $\kappa(A)$ is "large", A is **ill-conditioned** and *even the best* numerical algorithms will produce a solution, $\hat{x}$ that cannot be guaranteed to be close to the true solution, x
3. In practice, Gaussian elimination with partial pivoting and back substitution produces a solution with a small residual
$$r = b - A\hat{x}$$
even if $\kappa(A)$ is large.

Factorization Methods

- LU factorization
- Cholesky factorization
- Use of the backslash operator

LU Factorization (1)

Find L and U such that

$$A = LU$$

and L is lower triangular, and U is upper triangular.

$$L = \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ \ell_{2,1} & 1 & 0 & & 0 \\ \ell_{3,1} & \ell_{3,2} & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ \ell_{n,1} & \ell_{n,2} & \cdots & \ell n-1,n & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ 0 & u_{2,2} & u_{2,3} & \cdots & u_{2,n} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & & & u_{n-1,n} \\ 0 & 0 & & & u_{n,n} \end{bmatrix}$$

Since L and U are triangular, it is easy to apply their inverses.

LU Factorization (2)

Since L and U are triangular, it is easy to apply their inverses.

Consider the solution to $Ax = b$.

$$A = LU \Longrightarrow (LU)x = b$$

Regroup, matrix multiplication is associative

$$L(Ux) = b$$

Let $Ux = y$, then

$$Ly = b$$

Since L is triangular it is easy (without Gaussian elimination) to compute

$$y = L^{-1}b$$

This expression should be interpreted as "Solve $Ly = b$ with a forward substitution."

LU Factorization (3)

Now, since y is known, solve for x

$$x = U^{-1}y$$

which is interpreted as "Solve $Ux = y$ with a backward substitution."

LU Factorization (4)

Algorithm 8.6 Solve $Ax = b$ with LU factorization

Factor A into L and U	
Solve $Ly = b$ for y	use forward substitution
Solve $Ux = y$ for x	use backward substitution

The Built-in `lu` Function

- Refer to `luNopiv` and `luPiv` functions in the NMM toolbox for expository implementations of LU factorization
- Use the built-in `lu` function for routine work

Cholesky Factorization (1)

- A must be symmetric and positive definite (SPD)
- For SPD matrices, pivoting is not required
- Cholesky factorization requires one half as many flops as LU factorization. Since pivoting is not required, Cholesky factorization will be more than twice as fast as LU factorization since data movement is avoided.
- Refer to the `Cholesky` function in NMM Toolbox for a view of the algorithm
- Use built-in `chol` function for routine work

Backslash Redux

The \ operator examines the coefficient matrix before attempting to solve the system.

\ uses:

- A triangular solve if A is triangular, or a permutation of a triangular matrix
- Cholesky factorization and triangular solves if A is symmetric and the diagonal elements of A are positive (*and* if the subsequent Cholesky factorization does not fail.)
- LU factorization if A is square and the preceding conditions are not met.
- QR factorization to obtain the least squares solution if A is not square.

Nonlinear Systems of Equations

The system of equations

$$Ax = b$$

is nonlinear if $A = A(x)$ or $b = b(x)$

Characteristics of nonlinear systems

- Solution *requires* iteration
- Each iteration involves the work of solving a related *linearized* system of equations
- For strongly nonlinear systems it may be difficult to get the iterations to converge
- Multiple solutions might exist

Nonlinear Systems of Equations

Example: Intersection of a parabola and a line

$$y = \alpha x + \beta$$
$$y = x^2 + \sigma x + \tau$$

or, using $x_1 = x$, $x_2 = y$

$$\alpha x_1 - x_2 = -\beta$$
$$(x_1 + \sigma)x_1 - x_2 = -\tau$$

which can be expressed in matrix notation as

$$\begin{bmatrix} \alpha & -1 \\ x_1 + \sigma & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -\beta \\ -\tau \end{bmatrix}$$

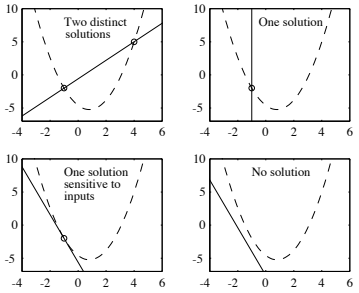
The coefficient matrix, A depends on x

Nonlinear Systems of Equations

Graphical Interpretation of solutions to:

$$\begin{bmatrix} \alpha & -1 \\ x_1 + \sigma & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -\beta \\ -\tau \end{bmatrix}$$

Change values of α and β to get



Newton's Method for Nonlinear Systems (1)

Given

$$Ax = b$$

where A is $n \times n$, write

$$f = Ax - b$$

Note: $f = -r$

The solution is obtained when

$$f(x) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n)) \\ f_2(x_1, x_2, \ldots, x_n)) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n)) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Newton's Method for Nonlinear Systems (2)

Let $x^{(k)}$ be the guess at the solution for iteration k

Look for $\Delta x^{(k)}$

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$$

so that

$$f(x^{(k+1)}) = 0$$

Expand f with the multidimensional Taylor Theorem

$$f(x^{(k+1)}) = f(x^{(k)}) + f'(x^{(k)})\Delta x^{(k)} + \mathcal{O}\left(\|\Delta x^{(k)}\|^2\right)$$

Newton's Method for Nonlinear Systems (3)

$f'(x^{(k)})$ is the *Jacobian* of the system of equations

$$f'(x) \equiv J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_3}{\partial x_n} \\ \vdots & & \ddots & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Neglect the higher order terms in the Taylor expansion

$$f(x^{(k+1)}) = f(x^{(k)}) + J(x^{(k)})\Delta x^{(k)}$$

Newton's Method for Nonlinear Systems (4)

Now, assume that we can find the $\Delta x^{(k)}$ that gives $f(x^{(k+1)}) = 0$

$$0 = f(x^{(k)}) + J(x^{(k)})\Delta x^{(k)} \qquad \Longrightarrow \qquad J(x^{(k)})\Delta x^{(k)} = -f(x^{(k)})$$

The essence of Newton's method for systems of equations is

1. Make a guess at x
2. Evaluate f
3. If $\|f\|$ is small enough, stop
4. Evaluate J
5. solve $J\,\Delta x = -f$ for Δx
6. update: $x \leftarrow x + \Delta x$
7. Go back to step 2

Newton's Method for Nonlinear Systems (5)

Example: Intersection of a line and parabola

$$y = \alpha x + \beta$$
$$y = x^2 + \sigma x + \tau$$

Recast as

$$\alpha x_1 - x_2 + \beta = 0$$
$$x_1^2 + \sigma x_1 - x_2 + \tau = 0$$

or

$$f = \begin{bmatrix} \alpha x_1 - x_2 + \beta \\ x_1^2 + \sigma x_1 - x_2 + \tau \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Newton's Method for Nonlinear Systems (6)

Evaluate each term in the Jacobian

$$\frac{\partial f_1}{\partial x_1} = \alpha \qquad \frac{\partial f_1}{\partial x_2} = -1$$

$$\frac{\partial f_2}{\partial x_1} = 2x_1 + \sigma \qquad \frac{\partial f_2}{\partial x_2} = -1$$

therefore

$$J = \begin{bmatrix} \alpha & -1 \\ (2x_1 + \sigma) & -1 \end{bmatrix}$$