Introduction to Square Linear Systems

Contents

Opening Example: Polynomial Interpolation

Polynomial Interpolation

Formal Statement

Given a set of n data points $\{(x_j, y_j) \mid j \in \mathbb{N}[1,n]\}$ with distinct x_j's, not necessarily sorted, find a polynomial of degree $n-1$,

$$p(x) = c_1 + c_2 x + c_3 x^2 + \cdots + c_n x^{n-1}, \qquad (\star)$$

(ascending)

which interpolates the given points, *i.e.*,

$$p(x_j) = y_j, \quad \text{for } j = 1, 2, \ldots, n\,.$$

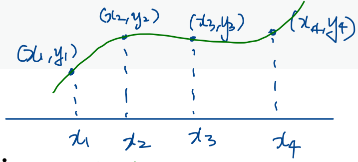
(n eqns)

- The goal is to determine the coefficients $c_1, c_2, \ldots, c_n$.
- Note that the total number of data point is 1 larger than the degree of the interpolating polynomial.

Why Do We Care?

- to find the values between the discrete data points;
- to approximate a (complicated) function by a polynomial, which makes such computations as differentiation or integration easier.

Interpolation to Linear System



Writing out the n *interpolating conditions* $p(x_j) = y_j$:

Goal: Find $p(x)$ polynom. going through all data points.

Equations

$$
\begin{aligned}
p(x_1) = \quad & c_1 + c_2 x_1 + \cdots + c_n x_1^{n-1} = y_1 \\
p(x_2) = \quad & c_1 + c_2 x_2 + \cdots + c_n x_2^{n-1} = y_2 \\
\vdots \quad & \quad \vdots \qquad \vdots \qquad\qquad\quad \vdots \qquad\quad \vdots \\
p(x_n) = \quad & c_1 + c_2 x_n + \cdots + c_n x_n^{n-1} = y_n
\end{aligned}
\quad \rightarrow
$$

Matrix equation

$$
\underbrace{\begin{bmatrix} 1 & \boxed{x_1} & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix}}_{V \ (n \times n)}
\underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{c}}
=
\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}}
$$

x-data

unknowns

y-data

- This is a linear system of n equations with n unknowns.
- The matrix V is called a **Vandermonde matrix**.

$$
V = \left[\; x.\wedge 0 \; \middle| \; x.\wedge 1 \; \middle| \; x.\wedge 2 \; \middle| \; \cdots \; \middle| \; x.\wedge (n-1) \; \right] \qquad \text{where } x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
$$

must be a column!

```
>> V = x.^(0:n-1);
```

Example: Fitting Population Data

U.S. Census data are collected every 10 years.

Year	Population (millions)
1980	226.546
1990	248.710
2000	281.422
2010	308.746
2020	332.639

Question. How do we estimate population in other years?

- Interpolate available data to compute population in intervening years.

Example: Fitting Population Data (cont')

- Input data.
- Match up notation (optional).
- Note the shift in Line 7.
- Construct the Vandermonde matrix V by *broadcasting*.
- Solve the system using the backslash (\\) operator.

```
1   year = (1980:10:2020)';
2   pop = [226.546;
3          248.710;
4          281.422;
5          308.746;
6          332.639];
7   x = year - 1980;
8   y = pop;
9   n = length(x);
10  V = x.^(0:n-1);
11  c = V \ y;
```

years since 1980.

efficient construction of Vandermonde matrix

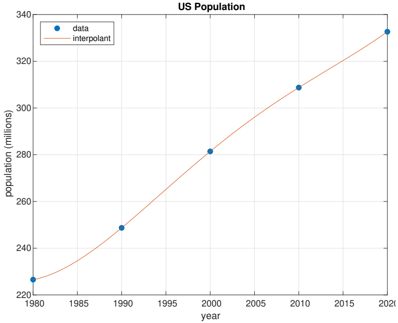
$$V\vec{c} = \vec{y} \implies \vec{c} = V^{-1}\vec{y}$$

Post-Processing

MATLAB stores/reads polynom. coeffs.
in descending order.

```
xx = linspace(0, 40, 100)';
yy = polyval(flip(c), xx);
clf
plot(1980+x, y, '.', 1980+xx, yy)
title('US Population'),
xlabel('year'), ylabel('population (millions)')
legend('data', 'interpolant', 'location', 'northwest')
```

- Use the `polyval` function to evaluate the polynomial.
- MATLAB expects coefficients to be in descending order. (`flip`)



Square Linear Systems

Given $A \in \mathbb{R}^{n \times n}$, $\vec{y} = \mathbb{R}^{n}$ ($= \mathbb{R}^{n \times 1}$ column vector),

find $\vec{x} \in \mathbb{R}^{n}$ satisfying

$$A\vec{x} = \vec{y}.$$

Analytically: $\vec{x} = A^{-1}\vec{y}$

Numerically: $x = A \backslash y;$

Overview

Square matrix.

Let $A \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. Then the equation $A\mathbf{x} = \mathbf{b}$ has the following possibilities:

- If A is invertible (or nonsingular), then $A\mathbf{x} = \mathbf{b}$ has a unique solution $\mathbf{x} = A^{-1}\mathbf{b}$, or $\Leftrightarrow \det(A) \neq 0$
- If A is not invertible (or singular), then $A\mathbf{x} = \mathbf{b}$ has either no solution or infinitely many solutions. $\Leftrightarrow \det(A) = 0$

The Backslash Operator " \ "

To solve for $\mathbf{x}$ in MATLAB, we use the backslash symbol " \ ":

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

This produces the solution without explicitly forming the inverse of A.

- not fast
- not accurate

Warning: Even though $\mathbf{x} = A^{-1}\mathbf{b}$ analytically, don't use `x = inv(A)*b`!

Triangular Systems

Systems involving triangular matrices are easy to solve.

- A matrix $U \in \mathbb{R}^{n\times n}$ is **upper triangular** if all entries below main diagonal are zero:

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}.$$

- A matrix $L \in \mathbb{R}^{n\times n}$ is **lower triangular** if all entries above main diagonal are zero:

$$L = \begin{bmatrix} \ell_{11} & 0 & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & \ell_{nn} \end{bmatrix}.$$

Example: Upper Triangular Systems

Solve the following 4×4 system

unknown

$$\underbrace{\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}}_{\text{given}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}}_{\text{given}}.$$

Strategy: Tackle from simple to complex.

(Backward Substitution)

Write out components

$$u_{11}x_1 + u_{12}x_2 + u_{13}x_3 + u_{14}x_4 = b_1$$

$$u_{22}x_2 + u_{23}x_3 + u_{24}x_4 = b_2$$

$$u_{33}x_3 + u_{34}x_4 = b_3$$

$$u_{44}x_4 = b_4$$

- Simplest

- $u_{44}x_4 = b_4 \Rightarrow \boxed{x_4 = \dfrac{b_4}{u_{44}}}$
- $\boxed{x_3 = \dfrac{b_3 - u_{34}x_4}{u_{33}}}$
- $\boxed{x_2 = \dfrac{b_2 - u_{23}x_3 - u_{24}x_4}{u_{22}}}$
- $\boxed{x_1 = \dfrac{b_1 - u_{12}x_2 - u_{13}x_3 - u_{14}x_4}{u_{11}}}$

General Results

efficient coding : (inner product)

$$[u_{i,i+1}\ u_{i,i+2} \cdots u_{i,n}] \begin{bmatrix} x_{i+1} \\ x_{i+2} \\ \vdots \\ x_n \end{bmatrix}$$

- **Backward Substitution.** To solve a general $n \times n$ upper triangular system $U\mathbf{x} = \mathbf{b}$:

$$\begin{cases} x_n = \dfrac{b_n}{u_{nn}} \quad \text{and} \\ x_i = \dfrac{1}{u_{ii}} \left(b_i - \boxed{\displaystyle\sum_{j=i+1}^{n} u_{ij} x_j} \right) \end{cases}$$

can be combined w/ convention

$$\sum_{j=n+1}^{n} u_{ij} x_j = 0.$$

for $i = n-1, n-2, \ldots, 1$.

- **Forward Elimination.** To solve a general $n \times n$ lower triangular system $L\mathbf{x} = \mathbf{b}$:

$$\begin{cases} x_1 = \dfrac{b_1}{\ell_{11}} \quad \text{and} \\ x_i = \dfrac{1}{\ell_{ii}} \left(b_i - \displaystyle\sum_{j=1}^{i-1} \ell_{ij} x_j \right) \end{cases}$$

$L =$ 

Simple

for $i = 2, 3, \ldots, n$.

Implementation: Backward Substitution

```
function x = backsub(U,b)
% BACKSUB x = backsub(U,b)
% Solve an upper triangular linear system.
% Input:
%    U    upper triangular square matrix (n by n)
%    b    right-hand side vector (n by 1)
% Output:
%    x    solution of Ux=b (n by 1 vector)
    n = length(U);
    x = zeros(n,1); % preallocate
    for i = n:-1:1
        x(i) = ( b(i) - U(i,i+1:n)*x(i+1:n) ) / U(i,i);
    end
end
```

$$\sum_{j=i+1}^{n} u_{ij} x_j$$

Implementation: Forward Elimination

Exercise. Complete the code below.

```
function x = forelim(L,b)
% FORELIM x = forelim(L,b)
% Solve a lower triangular linear system.
% Input:
%   L    lower triangular square matrix (n by n)
%   b    right-hand side vector (n by 1)
% Output:
%   x    solution of Lx=b (n by 1 vector)






end
```

Does It Always Work?

Singularity of Triangular Matrix

A triangular matrix is singular if and only if at least one of its diagonal elements is zero.