Use of Scilab to demonstrate concepts in linear algebra and polynomials

Dr. Madhu N. Belur

Control & Computing
Department of Electrical Engineering
Indian Institute of Technology Bombay
Email: belur@ee.iitb.ac.in

Outline

Introduction

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- This talk focus: linear algebra and polynomials.

Defining a matrix

- `A=[1 3 4 6]`
- `B=[1 3 4 6;5 6 7 8]`
- `size(A), length(A), ones(A), zeros(B), zeros(3,5)`

determinant/eigenvalues/trace

- `A=rand(3,3)`
- `det(A), spec(A), trace(A)`
- `sum(spec(A))`
-
```
if sum(spec(A))==trace(A) then
      disp('yes, trace equals sum')
else
      disp('no, trace is not sum ')
end
```
- `prod(spec(A))-det(A)`

(Block) diagonalize A?

Let A be a square matrix ($n \times n$) with distinct eigenvalues $\lambda_1, \ldots \lambda_n$. Eigenvectors (column vectors) v_1 to v_n are then independent.

$$Av_1 = \lambda_1 v_1 \quad Av_2 = \lambda_2 v_2 \quad \ldots Av_n = \lambda_n v_n$$

$$A \left[v_1 v_2 \ldots v_n \right] = \left[v_1 v_2 \ldots v_n \right] \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix}$$

(Column scaling of vectors v_1, etc is just post-multiplication.)

- $[spe, vect] = spec(A)$
- `inv(vect)*A*vect`

Inverse exists because of independence assumption on eigenvectors.
Use ‘bdiag’ command for block diagonalization (when non diagonalizable).

Rank, SVD

- `rank(A) svd(A)`
- $[u, s, v] = svd(A)$
- check `u'-inv(u)` `u*s*v-A`

Example: Income tax

Income tax for a man earning Rs. NET (after exempted deductions) is

0%	for the first 1,50,000
10%	for the part between 1,50,000 and 3,00,000
20%	for the part between 3,00,000 and 5,00,000
30%	for the part above 5,00,000

-
- $[u, s, v] = svd(A)$
- check `u'-inv(u)` `u*s*v-A`

Defining polynomials

Polynomials play a very central role in control theory: transfer functions are ratio of polynomials.

- `s=poly(0,'s')` `s=poly(0,'s','roots')`
- `p=s^2+3*s+2` `p=poly([2 3 1],'s','coeff')`
- `roots(p)` `horner(p,5)`
- $a = [1\ \ 2\ \ 3]$ `horner(p,a)` `horner(p,a')`
- `w=poly(0,'w')` `horner(p,%i*w)`

Differentiation

- `p=poly([1 2 3 4 -3],'s','coeff')`
- `cfp=coeff(p)`
- `diffpcoff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoff,'s','coeff')`
- `degree(p)` can be used instead of `length(cfp)-1`

More about horner

- `w=poly(0,'w')`   `horner(p,(1+w)/(1-w))`
- `a=-rand(1,4); p=poly(a,'s');`
- `q=horner(p,(w-1)/(1+w)) // bilinear(Cayley transform)`
- `abs(roots(numer(q)))`

Multiplication and convolution

Output of a (linear and time-invariant) dynamical system is the convolution of the input signal with the 'impulse response'.
Convolution: central role.
Polynomial multiplication is related to convolution of their coefficients

- `a=[1 2 3]; b=[4 5 6]; convol(a,b)`
- `pa=poly(a,'s','coeff'); pb=poly(b,'s','coeff'); coeff(pa*pb)`

To convolve $u(\cdot)$ by $h(\cdot)$ is a linear operation on $u(\cdot)$.
Write $h(s) = h_0 + h_1 s + h_2 s^2 + \cdots + h_n s^n$ (similarly $u(s)$)
convolution $y := h * u$ (convolution of h and u).
$y(k) = \sum_{j=0}^{n+m} h(j)u(k-j)$ (u has degree m).

Matrix for convolution

$$[y_0 \ y_1 \ \cdots \ y_{n+m}] = [u_0 \ u_1 \ \cdots \ u_m] C_h$$

where the matrix C_h with $m+1$ rows and $n+m+1$ columns is defined as

$$\begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_n & 0 & \cdots & 0 \\ 0 & h_0 & h_1 & \cdots & h_{n-1} & h_n & \cdots & 0 \\ \vdots & \ddots & \ddots & \dots & \ddots & \ddots & \dots & \vdots \\ 0 & 0 & 0 & \dots & & & & h_n \end{bmatrix}$$

Coprime polynomials

Numerator and denominator polynomials of a transfer function being coprime is critical for controllability and observability of dynamical systems: Kalman

- polynomials $a(s)$ and $b(s)$ are called coprime if they have no common root.
- Equivalently, their gcd (greatest common divisor) is 1.

Problem: given $a(s)$ and $b(s)$, find polynomials $p(s)$ and $q(s)$ such that $ap + bq = 0$.
Easy: take $p := -b$ and $q = a$. Relation to coprimeness??

Coprimeness: equivalent statements

Consider polynomials $a(s)$ (of degree m) and $b(s)$ (of degree n). Following statements are equivalent.

- a and b are coprime.
- there exist no polynomials p and q with degree$(p) < n$ degree$(q) < m$ such that $ap + bq = 0$.
- there exist polynomials u and v such that $au + bv = 1$ (their gcd).

In fact, u and v having degrees at most $n - 1$ and $m - 1$ respectively can be found. Then they are unique.

Equivalent matrix formulations

$ap + bq$ can be considered as having coefficients obtained from

$$[p_0 \ p_1 \ \cdots \ p_{m-1} \ q_0 \ q_1 \ \cdots \ q_{n-1}] \begin{bmatrix} C_a \\ C_b \end{bmatrix}$$

C_a and C_b have m rows and n rows respectively, and both have $m+n$ columns each.
Hence, following are equivalent

- a and b are coprime
- $\begin{bmatrix} C_a \\ C_b \end{bmatrix}$ is nonsingular
- $\begin{bmatrix} C_a \\ C_b \end{bmatrix}$ has $[1 \ 0 \ \cdots \ 0]$ in its (left)-image

Above matrix: Sylvester resultant matrix, its determinant: resultant of two polynomials

Check this in scilab

- *sylvester_mat.sci* function constructs required matrix
- `linsolve(sy',`$[1\ 0\ 0\ \cdots\ 0]'$`) // for coprime and non-coprime`
- $[x, kern]$`=linsolve(sy',`$[0\ 0\ 0\ \cdots\ 0]'$`)`

Uncontrollable and unobservable modes are related to eigenvectors corresponding to the eigenvalue which is the ‘common root’.

Command 'find' : extracts TRUE indices

Consider the following problem: polynomials a and b might have roots 'close by'.
(Very difficult to control/observe: very high energy or input levels needed to control, or measurement very sensitive to noise. This is due to 'close to' uncontrollable/unobservable).
Find which are close to each other.

- Find roots of a and b. For each root of a, check if a root of b is within specified tolerance 'toler'.
- Two for loops?
- find allows extraction of indices satisfying a boolean expression

Coefficients and powers as vectors

Evaluation of a polynomial $p(s)$ at a value $s = a$ is a linear map on the coefficients.
$p(a) = [p_0 \; p_1 \; \cdots \; p_n][1 \; a \; a^2 \cdots \; a^n]'$
Moreover, if n and m are the degrees of p and q respectively,

$$p(s)q(s) = [1 \; s \; s^2 \; s^n] \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} [q_0 \; q_1 \; \cdots \; q_m] \begin{bmatrix} 1 \\ s \\ \vdots \\ s^m \end{bmatrix}$$

$(n+1) \times (m+1)$ matrix.

Bezoutian matrix

Bezoutian of a pair of polynomials $p(s)$ and $q(s)$ is defined as the symmetric matrix B such that (suppose $n \geqslant m$)

$$[1\ x\ x^2\ x^{n-1}]B \begin{bmatrix} 1 \\ y \\ \vdots \\ y^{n-1} \end{bmatrix} = \frac{p(x)q(y) - p(y)q(x)}{x - y}$$

B is $n \times n$ matrix. B is nonsingular if and only if p and q are coprime.

Size of B is roughly half the size of the Sylvester resultant matrix.

B is symmetric.

(Both can have polynomials (in γ) as their coefficients.)

Bezoutian matrix

Bezoutian of a pair of polynomials $p(s)$ and $q(s)$ is defined as the symmetric matrix B such that (suppose $n \geqslant m$)

$$[1 \ x \ x^2 \ x^{n-1}]B \begin{bmatrix} 1 \\ y \\ \vdots \\ y^{n-1} \end{bmatrix} = \frac{p(x)q(y) - p(y)q(x)}{x - y}$$

B is $n \times n$ matrix. B is nonsingular if and only if p and q are coprime.
Size of B is roughly half the size of the Sylvester resultant matrix.
B is symmetric.
(Both can have polynomials (in γ) as their coefficients.)

Discrete Fourier Transform

For a periodic sequence: DFT (Discrete Fourier Transform) gives the frequency content.
Linear transformation on the input sequence.
Take signal values of just one period: finite dimensional signal (due to periodicity of N).

$$X(k) := \sum_{n=0}^{N-1} x(n) e^{\frac{-2\pi ik}{N} n} \text{ for } k = 0, \ldots, N-1 \text{ (analysis equation)}$$

$e^{-2\pi ikN}$ is the N^{th} root of unity.
Inverse DFT for the synthesis equation. Normalization constants vary in the literature.

Discrete Fourier Transform

What is the matrix defining relating the DFT $X(k)$ of the signal $x(n)$? Define $\omega := e^{\frac{-2\pi ik}{N}n}$.

$$\begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2N-2} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2N-2} & \cdots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

(Note: $\omega^N = 1$, etc.)
Check that the above $N \times N$ matrix has nonzero determinant.
(Change of basis.) Moreover, columns are orthogonal.
Orthonormal? (Normalization (by $\sqrt{N}$) not done yet.)

Discrete Fourier Transform and interpolation

Van der monde matrix: closely related to interpolation problems
Of course, inverse DFT is nothing but interpolation! Used in computation of determinant of a polynomial matrix.
Construct $p(s) := x_0 + x_1 s + x_2 s^2 \cdots + x_{N-1} s^{N-1}$
To obtain $X(k)$, evaluate p at $s = \omega^k$.
$X(k) = p(\omega^k)$ `horner` command
Given values of $p(\omega^k)$ for various ω^k (i.e., $X(k)$), find the coefficients of the polynomial $p(s)$: inverse DFT: interpolation of a polynomial to 'fit' given values at specified (complex) numbers.

Discrete Fourier Transform and interpolation

Van der monde matrix: closely related to interpolation problems
Of course, inverse DFT is nothing but interpolation! Used in computation of determinant of a polynomial matrix.
Construct $p(s) := x_0 + x_1 s + x_2 s^2 \cdots + x_{N-1} s^{N-1}$
To obtain $X(k)$, evaluate p at $s = \omega^k$.
$X(k) = p(\omega^k)$ `horner` command
Given values of $p(\omega^k)$ for various ω^k (i.e., $X(k)$), find the coefficients of the polynomial $p(s)$: inverse DFT: interpolation of a polynomial to 'fit' given values at specified (complex) numbers.

Discrete Fourier Transform and interpolation

Van der monde matrix: closely related to interpolation problems
Of course, inverse DFT is nothing but interpolation! Used in computation of determinant of a polynomial matrix.
Construct $p(s) := x_0 + x_1 s + x_2 s^2 \cdots + x_{N-1} s^{N-1}$
To obtain $X(k)$, evaluate p at $s = \omega^k$.
$X(k) = p(\omega^k)$ `horner` command
Given values of $p(\omega^k)$ for various ω^k (i.e., $X(k)$), find the coefficients of the polynomial $p(s)$: inverse DFT: interpolation of a polynomial to 'fit' given values at specified (complex) numbers.

Discrete Fourier Transform and interpolation

Van der monde matrix: closely related to interpolation problems
Of course, inverse DFT is nothing but interpolation! Used in computation of determinant of a polynomial matrix.
Construct $p(s) := x_0 + x_1 s + x_2 s^2 \cdots + x_{N-1} s^{N-1}$
To obtain $X(k)$, evaluate p at $s = \omega^k$.
$X(k) = p(\omega^k)$ `horner` command
Given values of $p(\omega^k)$ for various ω^k (i.e., $X(k)$), find the coefficients of the polynomial $p(s)$: inverse DFT: interpolation of a polynomial to 'fit' given values at specified (complex) numbers.

FFT

Since many powers of ω are repeated in that matrix (only $N-1$ powers are different, many real/imaginary parts are repeated for even N), redundancy can be drastically decreased.

Length of the signal is a power of 2: recursive algorithm possible.

FFT

Since many powers of ω are repeated in that matrix (only $N-1$ powers are different, many real/imaginary parts are repeated for even N), redundancy can be drastically decreased.
Length of the signal is a power of 2: recursive algorithm possible.

FFT: recursive implementation

- Separate $p(s)$ (coefficients $x_0, \ldots, x_{N-1}$) into its even and odd powers (even and odd indices k). N is divisible by 2.
- Compute DFT of p_{odd} and p_{even} separately. (Do same separation, if possible.)
- Let X_{odd} and X_{even} denote the individual DFT's. (Same length.)
- Define $D := \mathrm{diag}(1, \omega, \omega^2, \ldots, \omega^{\frac{N}{2}-1})$
- Combine the two separate DFT's using the formula

$$\begin{aligned} X(k) &= X_{\mathrm{even}} + DX_{\mathrm{odd}} \text{ for } k = 0, \ldots, \frac{N}{2} - 1 \\ X(k) &= X_{\mathrm{even}} - DX_{\mathrm{odd}} \text{ for } k = \frac{N}{2}, \ldots, N - 1 \end{aligned}$$

Conclusions

- Matrices and polynomials provide rich source of problems
- With good computational tools, the future lies in computational techniques
- Scilab provides handy tools
- We saw: `if elseif else end for horner poly coeff`
- Recursive use of `function`
- `find conv max min`