Use of Scilab for SVD, linear algebra

Madhu N. Belur

Control & Computing, Department of Electrical Engineering, IITB
Email: belur@iitb.ac.in

November 2019

Codes and other material (for this workshop):
www.ee.iitb.ac.in/∼belur/scilab
Learning material: www.spoken-tutorial.org:
28 Scilab tutorials: each in upto 16 languages

Outline

- Vector/Matrix syntax
- Eigenvalues, singular values
- Solving Ax = b
- Linear independence/dependence
- How to read in data from a file, define matrix, etc.

Introduction

- Scilab is free and open source (unlike expensive/proprietary: Matlab)
- Matrix/loops syntax is same as for Matlab
- Accuracy and computation time: same: both use (FOSS) Lapack/Linpack within
- In many ways, Scilab is better
- This talk focus: linear algebra, SVD, solving Ax=b

Introduction

- Scilab is free and open source (unlike expensive/proprietary: Matlab)
- Matrix/loops syntax is same as for Matlab
- Accuracy and computation time: same: both use (FOSS) Lapack/Linpack within
- In many ways, Scilab is better
- This talk focus: linear algebra, SVD, solving Ax=b

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)`

Note: in numerical computation, all values are 'floating point': Hence == is not the right way to check. Use 'norm'.

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)`

Note: in numerical computation, all values are 'floating point': Hence == is not the right way to check. Use 'norm'.

Rank, SVD

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

Solving Ax = b

- For a matrix A, and vector b:
  linsolve(A,b) gives
  'compatible' x (if exists) such that $Ax + b = 0$
- [x,kerA] = linsolve(A,b) // to get all solutions
- If no 'compatible' one exists, gives
  ''Warning: Conflicting linear constraints''
- A\b
- x = A\b

Solving Ax = b

- For a matrix A, and vector b:
  linsolve(A,b) gives
  'compatible' x (if exists) such that $Ax + b = 0$
- [x,kerA] = linsolve(A,b) // to get all solutions
- If no 'compatible' one exists, gives
  ''Warning: Conflicting linear constraints''
- A\b
- x = A\b

Rank, solvability, linear dependence

- A = [1 2;3 6] // A has two ROWS
- b = [1 3]' // b is a column vector
- Check rank(A), and rank([A b])

Adding column: rank: might increase, but CANNOT decrease

- rank(A) and rank([A b]) are same

$\Updownarrow$

b is some linear combination of A

$\Updownarrow$

$Ax = b$ has a solution x

- Check for: A = [1 2;3 6]; b = [1 2]'
- Check for: A = [1 2;0 6]; b = [1 2]'

Rank, solvability, linear dependence

- `A = [1 2;3 6] // A has two ROWS`
- `b = [1 3]' // b is a column vector`
- `Check rank(A), and rank([A b])`

`Adding column: rank: might increase, but CANNOT decrease`

- `rank(A) and rank([A b]) are same`

  $\Updownarrow$

  `b is some linear combination of A`

  $\Updownarrow$

  $Ax = b$ `has a solution` x
- `Check for: A = [1 2;3 6]; b = [1 2]'`
- `Check for: A = [1 2;0 6]; b = [1 2]'`

Inbuilt parameters

Scilab provides inbuilt parameters: Need % to get them

- %pi for π
- %i for imaginary unit i
- %j for imaginary unit j
- Verify: %pi - 3.14
- Verify: %i^2
- Avoid defining your own variables starting with %
- %s, %z, %eps
- 13e14 = 13E14 = 13E+14 = 1.3D+15 = 1.3×10^{15}
  D $\equiv$ Decimal
- 190e-37 = 19E-36 = 1.9D-35
  1.9×10^{-35}
- Check value of %eps: machine precision: relative accuracy

Inbuilt parameters

Scilab provides inbuilt parameters: Need % to get them

- %pi for π
- %i for imaginary unit i
- %j for imaginary unit j
- Verify: %pi - 3.14
- Verify: %i^2
- Avoid defining your own variables starting with %
- %s, %z, %eps
- 13e14 = 13E14 = 13E+14 = 1.3D+15 = 1.3×10^{15}
  D $\equiv$ Decimal
- 190e-37 = 19E-36 = 1.9D-35
  1.9×10^{-35}
- Check value of %eps: machine precision: relative accuracy

Inbuilt parameters

Scilab provides inbuilt parameters: Need % to get them

- %pi for π
- %i for imaginary unit i
- %j for imaginary unit j
- Verify: %pi - 3.14
- Verify: %i^2
- Avoid defining your own variables starting with %
- %s, %z, %eps
- 13e14 = 13E14 = 13E+14 = 1.3D+15 = 1.3×10^{15}
  D $\equiv$ Decimal
- 190e-37 = 19E-36 = 1.9D-35
  1.9×10^{-35}
- Check value of %eps: machine precision: relative accuracy

Inbuilt parameters

Scilab provides inbuilt parameters: Need % to get them

- %pi for π
- %i for imaginary unit i
- %j for imaginary unit j
- Verify: %pi - 3.14
- Verify: %i^2
- Avoid defining your own variables starting with %
- %s, %z, %eps
- 13e14 = 13E14 = 13E+14 = 1.3D+15 = 1.3×10^{15}
  D $\equiv$ Decimal
- 190e-37 = 19E-36 = 1.9D-35
  1.9×10^{-35}
- Check value of %eps: machine precision: relative accuracy

Defining polynomials

- `s=poly(0,'s');` Same as: `s=poly(0,'s','roots')`

`p=2+3*s+s^2;` Alternatively: `p=poly([2 3 1],'s','coeff')`

- `roots(p)`
- $a = [1\ 2\ 3]$
- `w=poly(0,'w')`

Defining your own function

- Often we need to have our own function
- Good for modular and systematic design of large/complex programs:

```
function out1 = some_function_name(inp1)
   // Avoid using a name that is ALREADY a function
   // Recommended to have some explanatory text
   // Recommended to give an indent
   out1 = inp1^3
endfunction
```

This talk available at

- Can have many functions all in one file:
- Execute that file once and these functions are in the memory.
- Re-execute to overwrite (if functions have been changed)
- Some codes and other material (for this workshop): www.ee.iitb.ac.in/%7Ebelur/scilab

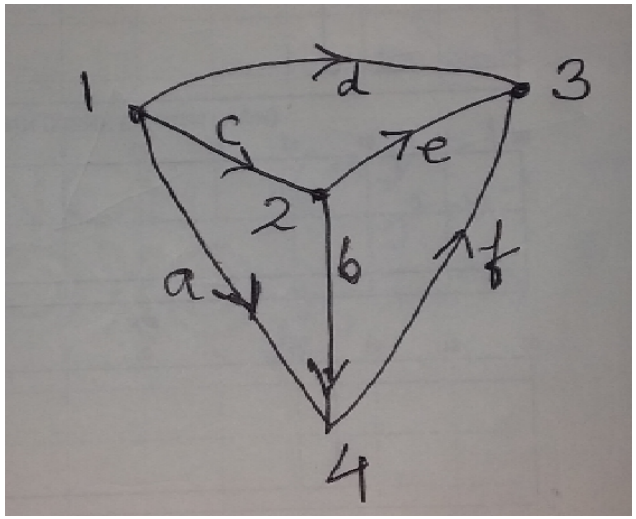
This talk available at

- Can have many functions all in one file:
- Execute that file once and these functions are in the memory.
- Re-execute to overwrite (if functions have been changed)
- Some codes and other material (for this workshop): www.ee.iitb.ac.in/%7Ebelur/scilab

SciNotes editor and exercise

Use SciNotes editor: syntax highlighting, quick execute, etc.

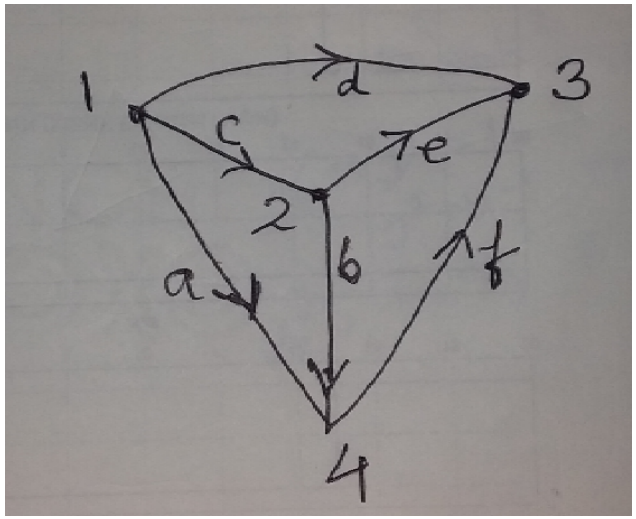
- Construct (full) 'incidence matrix' A for the graph below
- Find rank, kernel, image of A
  (see help colcomp, help rowcomp). (Note use of tolerance)
- Check that any row of A is in span of other rows



SciNotes editor and exercise

Use SciNotes editor: syntax highlighting, quick execute, etc.

- Construct (full) 'incidence matrix' A for the graph below
- Find rank, kernel, image of A
  (see help colcomp, help rowcomp). (Note use of tolerance)
- Check that any row of A is in span of other rows



SciNotes editor and exercise

Use SciNotes editor: syntax highlighting, quick execute, etc.

- Construct (full) 'incidence matrix' A for the graph below
- Find rank, kernel, image of A
  (see help colcomp, help rowcomp). (Note use of tolerance)
- Check that any row of A is in span of other rows