Good references....

- Lecture notes on numerical linear algebra:

  http://gbenthien.net/tutorials.html

- Classic text: Golub and Van Loan
- Excellent lectures on Linear Algebra:

  http://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/

Important properties of matrices

- Condition number
- Norm
- Decompositions:
  - Eigen-decomposition
  - SVD
- Classifications
  - General
  - Symmetric
  - Positive definite
  - Etc.

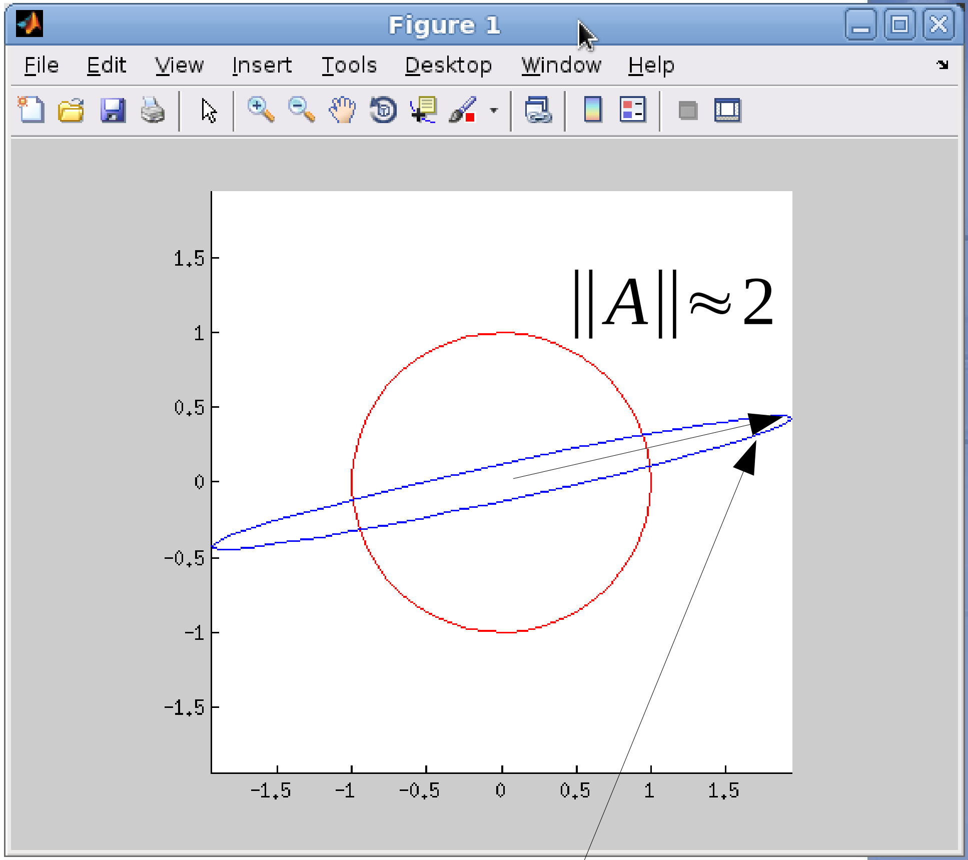
Induced matrix norm

- Start with vector norm:

$$||x||$$

- Define matrix norm by considering action of matrix on all vectors:

$$||A|| = max\left(\frac{||A\,x||}{||x||} : x \in K^n\right)$$



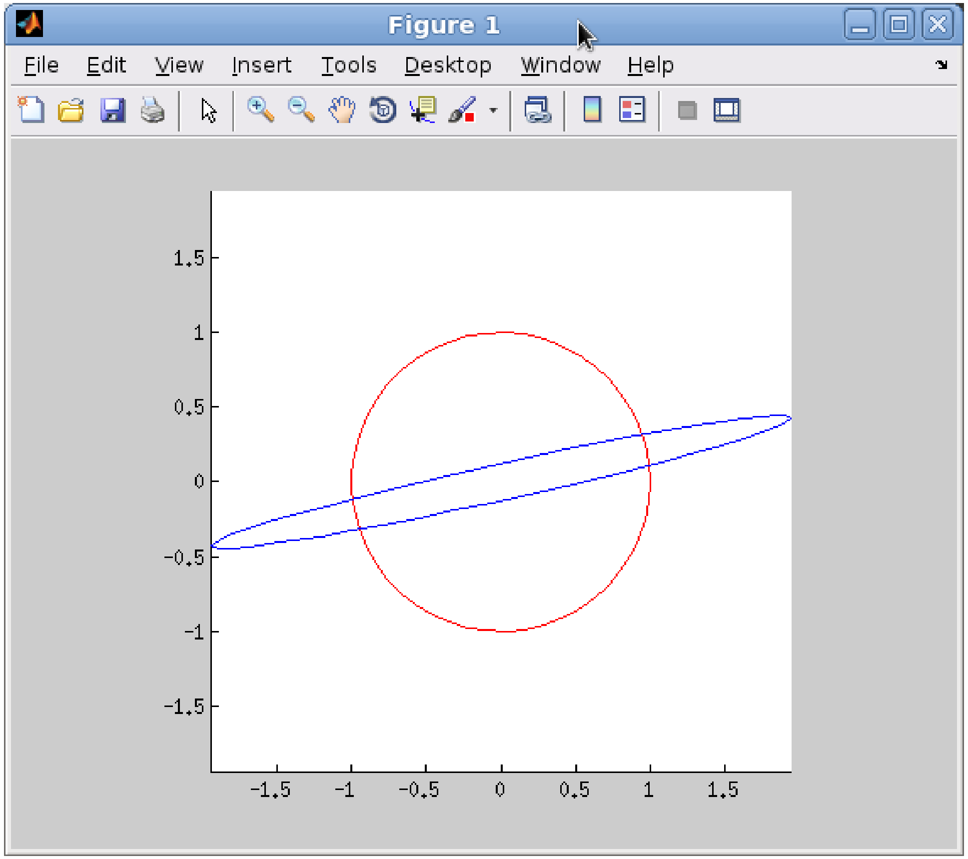
Find largest extension of unit circle induced by matrix.

Extension: Visualizing a matrix

- One way: Action of A on vector x where $||x|| = 1$. (That is, action of A on unit circle.)
  - Largest point on resulting ellipse is induced norm ("spectral norm")
  - Ratio of two axes lengths is condition number
  - Ratio of singular values is also condition number
- "Looking at shadow cast by the matrix."

/home/sdb/Northeastern1/Class3/ellipses.m

Action of matrix A on circle



- Plot of

  $$y = A x$$

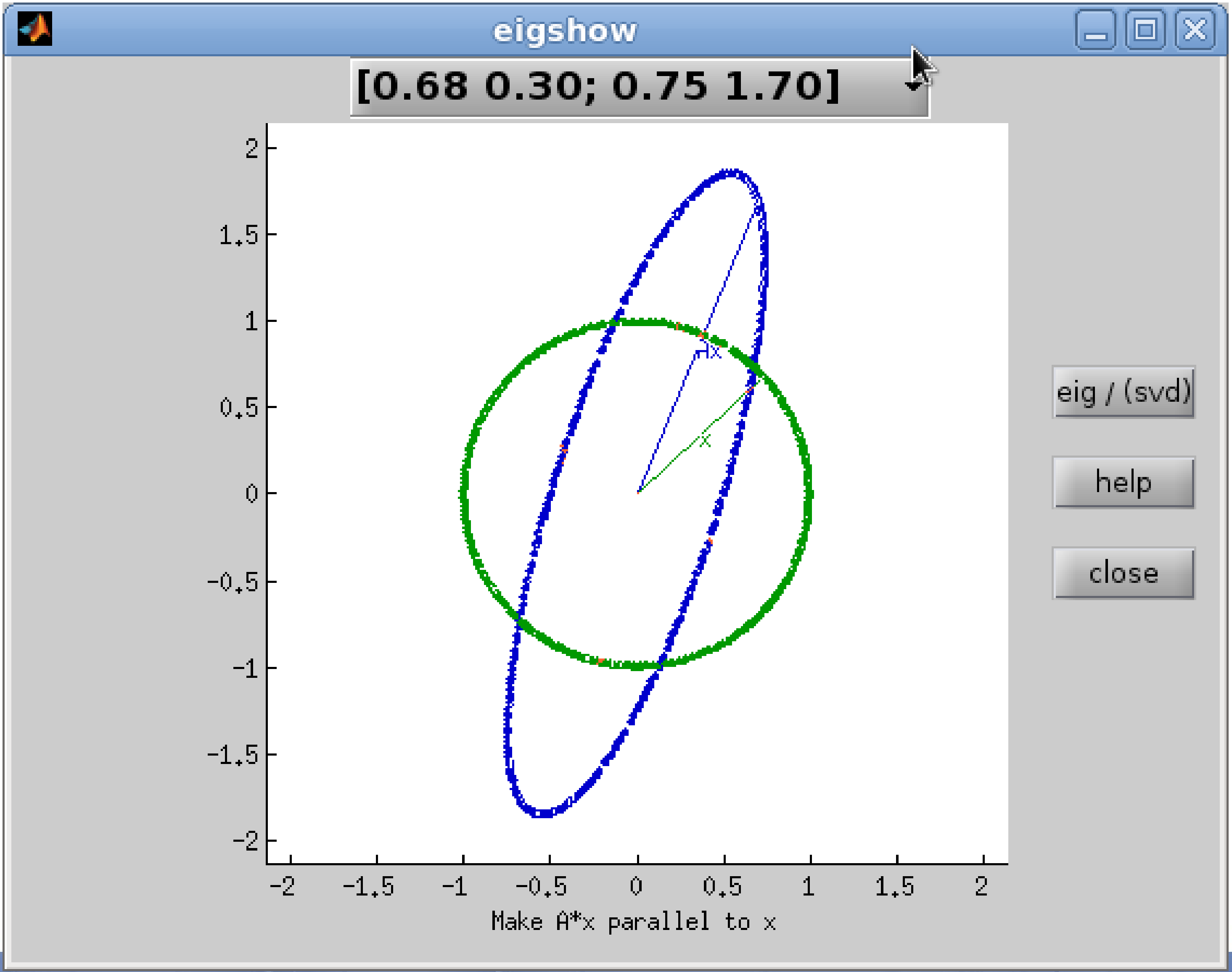
  where

  $$x = \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix}$$

  $$0 \leq t \leq 2\pi$$

```
>> ellipses
cond(A) = 16.369359, norm(A) = 1.989071 svd = [1.989071, 0.121512]

ans =

    0.1873   -1.9330
   -0.0825   -0.4390
```

Matlab "eigshow"



Matrix norm, eigenvalues and SVD

- Eigenvalue decomposition: Square matrix

$$A = Q \Lambda Q^{-1}$$

$$\boxed{A} = \boxed{Q}\ \boxed{\Lambda}\ \boxed{Q^{-1}}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 & \\ & \lambda_2 & & \\ 0 & & \lambda_3 & \\ & & & \ddots \end{pmatrix}$$

- Singular value decomposition: Arbitrary rectangular matrix

$$A = U \Sigma V^T$$

$$\underset{n \times p}{\boxed{A}} = \underset{n \times n}{\boxed{U}}\ \underset{n \times p}{\boxed{\Sigma}}\ \underset{p \times p}{\boxed{V^T}}$$

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \cdots \\ 0 & & \sigma_3 & \end{pmatrix}$$

Where does the SVD come from?

- Eigenvalue equation:

$$A\vec{x}_i = \lambda_i \vec{x}_i$$

Transformed vector lies in same space as x. (Only stretching along eigenvectors)

- SVD equation (works for rectangular matrix):

$$A\vec{v}_i = \sigma_i \vec{u}_i$$

u, v are unit vectors

Transformed vector lies in different space

- Rearranging:

$$A = \vec{u}_i \sigma_i \vec{v}_i^T$$

Because $\vec{v}_i^T = \vec{v}_i^{-1}$
($\vec{v}_i$ is unit vector)

SVD

$$A = U \Sigma V^T$$

$$\underset{n\times p}{A} = \underset{n\times n}{U}\;\underset{n\times p}{\Sigma}\;\underset{p\times p}{V^T}$$

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \cdots \\ 0 & & \sigma_3 & \end{pmatrix}$$

- *U*, *V* are unitary. ← Composed of unit column vectors.
- Σ is diagonal. Diagonal elements are the "singular values".
  - By convention, they are written in decreasing order, from largest to smallest.
  - Non diagonal entries are zero.

```
>> A = rand(4,6)

A =

    0.3161    0.3424    0.3774    0.1260    0.6010    0.9383
    0.1267    0.2041    0.0862    0.6835    0.8032    0.1889
    0.6724    0.5746    0.4193    0.8314    0.7251    0.9041
    0.9151    0.5423    0.6413    0.8550    0.7012    0.9235

>> [U, S, V] = svd(A)

U =

   -0.3928   -0.5424    0.7297    0.1378
   -0.3069    0.8323    0.4155    0.2012
   -0.5858    0.0431   -0.1325   -0.7984
   -0.6390   -0.1059   -0.5266    0.5506


S =

    2.9439         0         0         0         0         0
         0    0.7147         0         0         0         0
         0         0    0.4932         0         0         0
         0         0         0    0.1428         0         0


V =

   -0.3878   -0.1874   -0.5835    0.2524   -0.3358   -0.5455
   -0.2990   -0.0679   -0.0548   -0.5037    0.7121   -0.3770
   -0.2820   -0.2558   -0.1665    0.6139    0.5097    0.4366
   -0.4391    0.6239   -0.3741   -0.2669   -0.1087    0.4415
   -0.4604    0.4190    0.6224    0.3611   -0.0128   -0.3074
   -0.5252   -0.5745    0.3185   -0.3226   -0.3290    0.2833
```

Singular values & eigenvalues

- Singular values related to eigenvalues of $A^T A$
- Proof: $A = \vec{u}_i \sigma_i \vec{v}_i^T \qquad A^T = \vec{v}_i \sigma_i^T \vec{u}_i^T$

$$A^T A = (\vec{v}_i \sigma_i^T \underbrace{\vec{u}_i^T)(\vec{u}_i}_{I} \sigma_i \vec{v}_i^T) \qquad A A^T = (\vec{u}_i \sigma_i \underbrace{\vec{v}_i^T)(\vec{v}_i}_{I} \sigma_i^T \vec{u}_i^T)$$

$$= \vec{v}_i \sigma_i^T \sigma_i \vec{v}_i^T \qquad = \vec{u}_i \sigma_i \sigma_i^T \vec{u}_i^T$$

$$(A^T A) \vec{v}_i = \vec{v}_i \sigma_i^T \sigma_i \underbrace{\vec{v}_i^T \vec{v}_i}_{I} \qquad (A A^T) \vec{u}_i = \vec{u}_i \sigma_i \sigma_i^T \underbrace{\vec{u}_i^T \vec{u}_i}_{I}$$

$$(A^T A) \vec{v}_i = (\sigma_i^T \sigma_i) \vec{v}_i \qquad (A A^T) \vec{u}_i = (\sigma_i \sigma_i^T) \vec{u}_i$$

- Therefore: $\sigma_i = \sqrt{eig_i(A^T A)} = \sqrt{eig_i(A A^T)}$

Singular values corollary

- For positive, symmetric definite matrix:

$$\sigma_i = \sqrt{eig_i(A^T A)}$$

I

$$= \sqrt{eig_i((\vec{v}_i \sigma_i^T \vec{u}_i^T)(\vec{u}_i \sigma_i \vec{v}_i^T))}$$

$$= \sqrt{eig_i(\vec{v}_i \sigma_i^T \sigma_i \vec{v}_i^T)}$$

I

$$= \sqrt{eig_i(\sigma_i^T \sigma_i \vec{v}_i \vec{v}_i^T)}$$

$$= \sqrt{eig_i(\sigma_i^T \sigma_i)}$$

- Therefore:

Singular values

Eigenvalues

$$\sigma_i = |\lambda_i|$$

Action of a matrix on unit ball

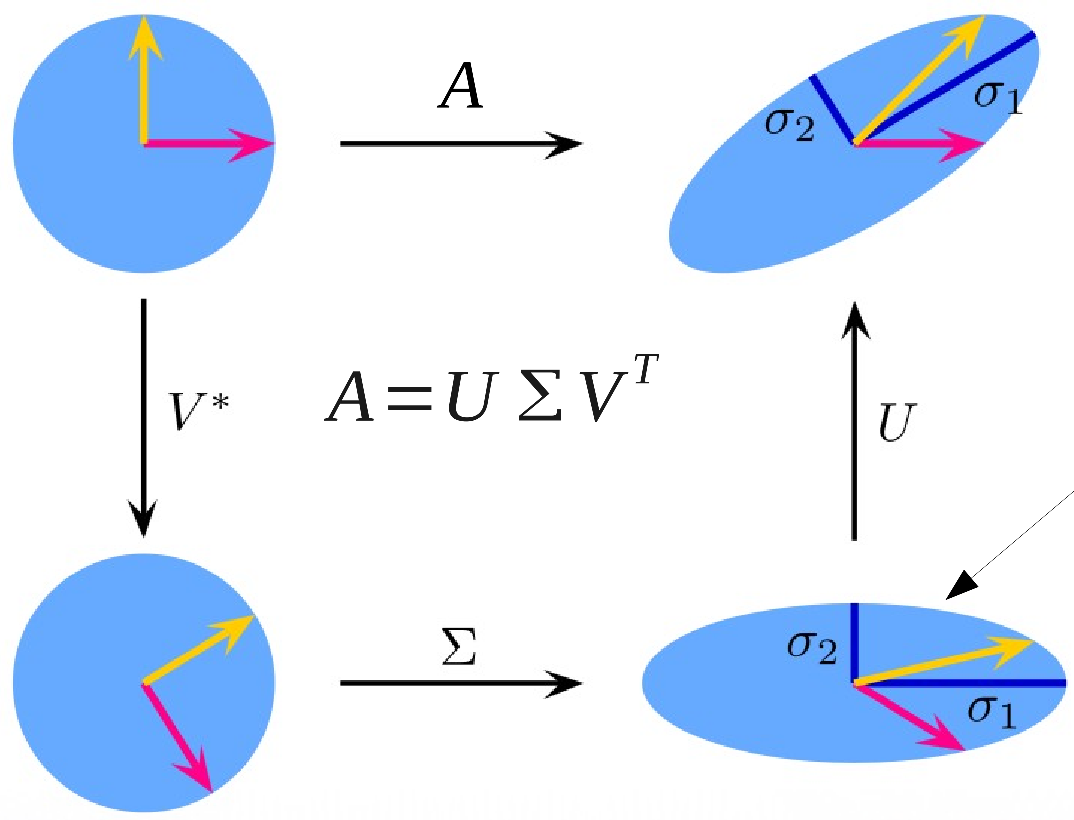
$$A\vec{e} = (\vec{u}_i \sigma_i \vec{v}_i^T)\vec{e}$$

Unit ball

Rotation

Stretching

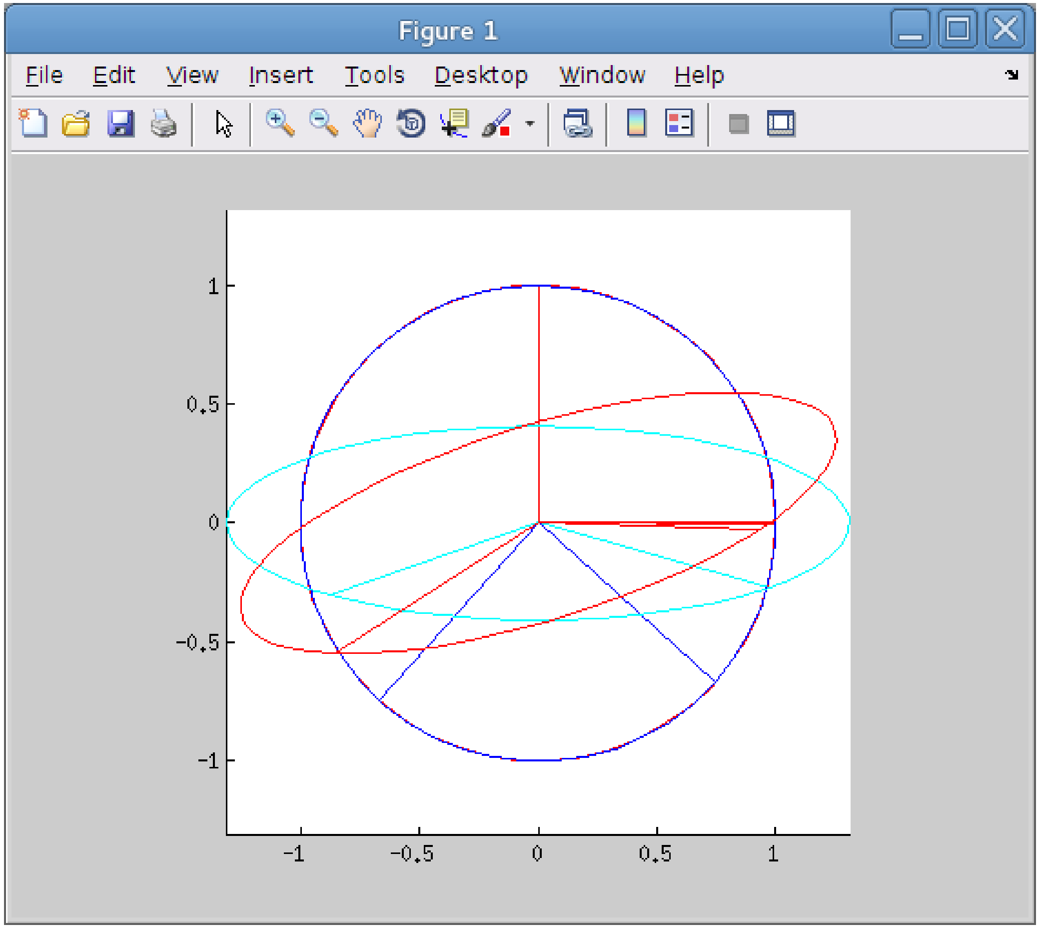
Rotation



Singular values are axes of ellipse

Demonstration

1. Red: Unit circle x
2. Blue: $V^T x$
3. Light blue: $S V^T x$
4. Black: $U S V^T x$
5. Red: $A x$

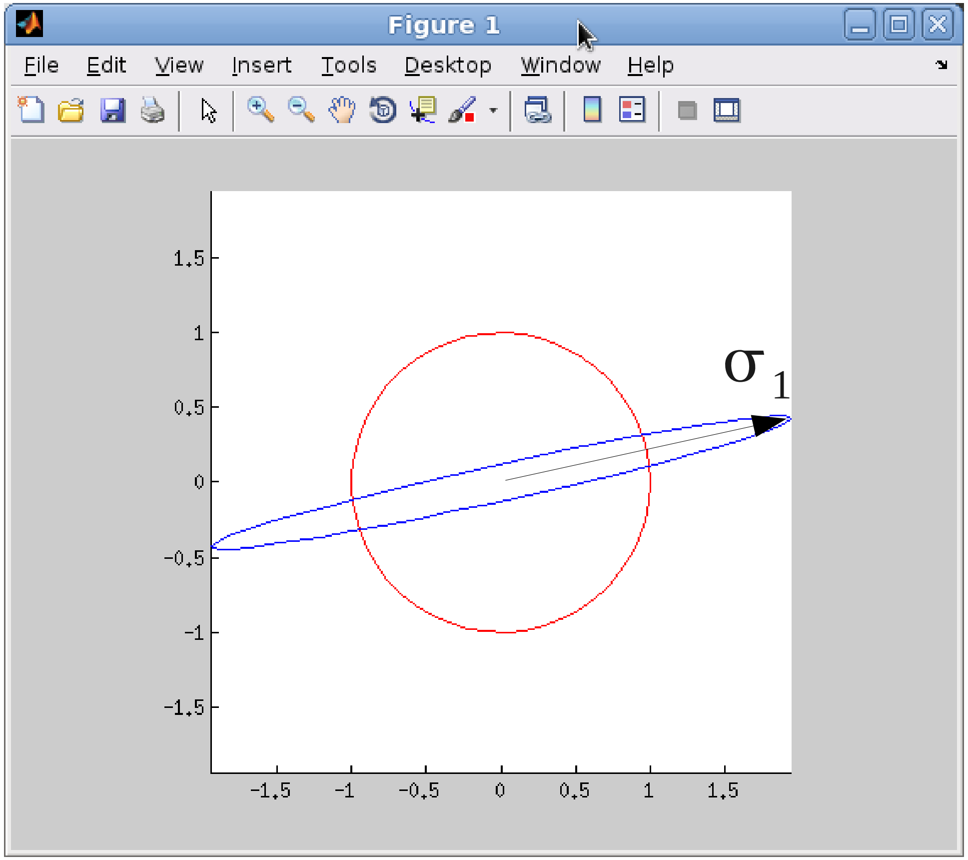


~/SVDTransform/svd_transform.m

Induced matrix norm & SVD

- "Induced norm" = largest singular value (see picture).
- SVD: $A = U \Sigma V^T$
- Singular values:

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \dots \\ 0 & & \sigma_3 & \end{pmatrix}$$



- Norm often written $\|A\| = max\left(\frac{\|A x\|}{\|x\|} : x \in K^n\right)$

Condition number

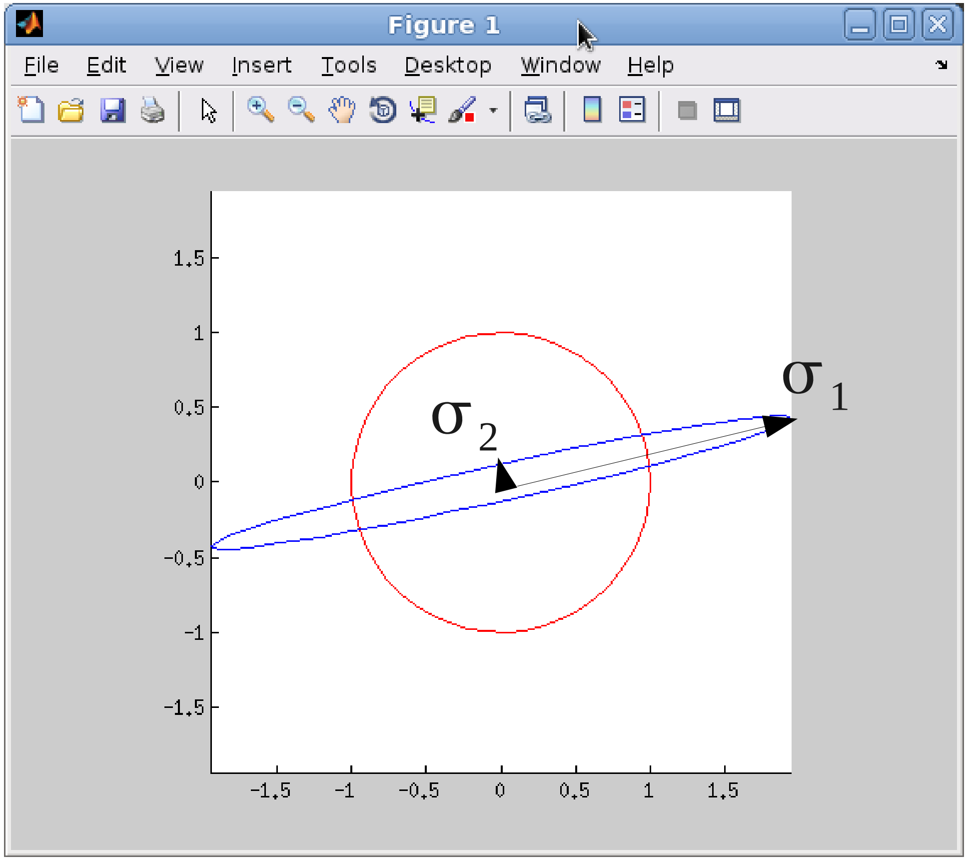
- SVD: $A = U \Sigma V^T$

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \cdots \\ 0 & & \sigma_3 & \end{pmatrix}$$

- Condition number:

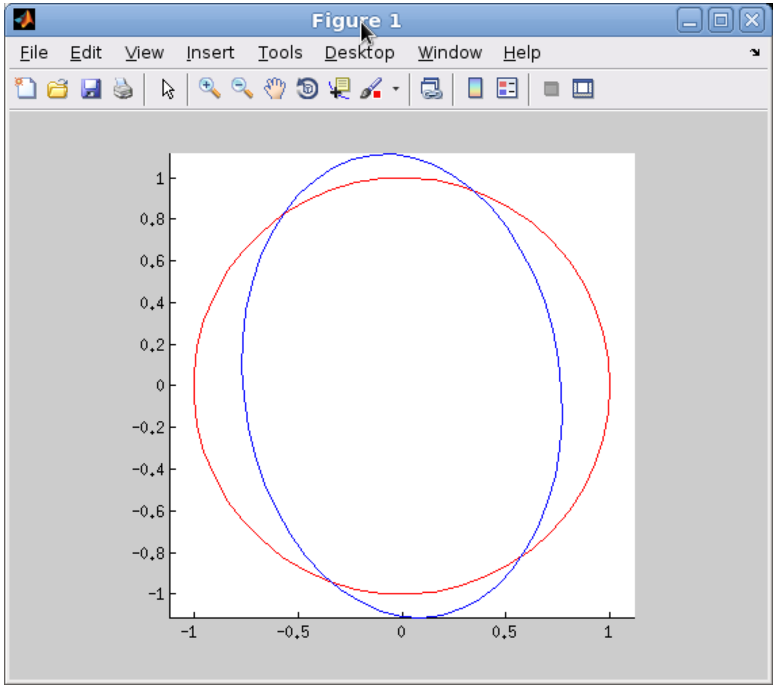
$$k = \frac{\sigma_{max}}{\sigma_{min}}$$

$$= \frac{\sigma_1}{\sigma_2}$$ In this example



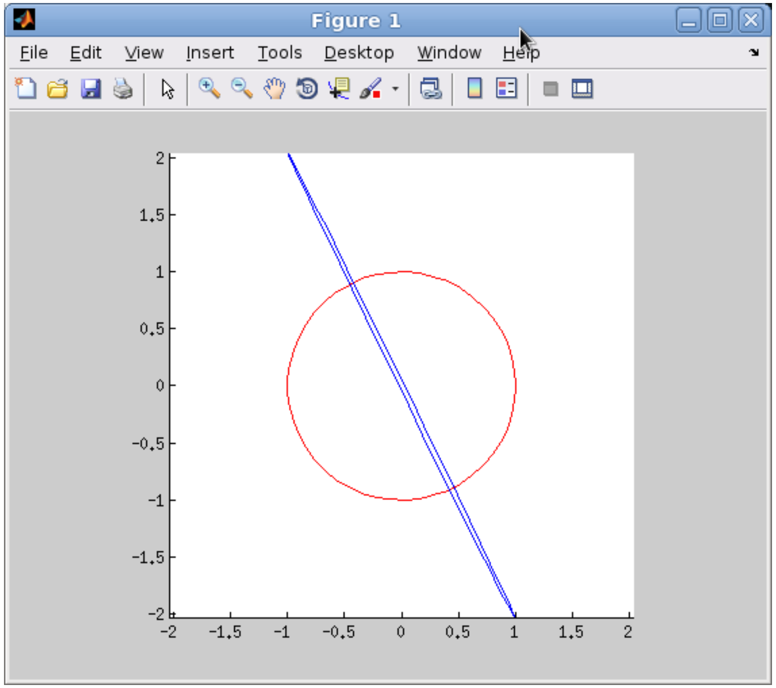
- For general matrix, the stretching occurs in N dimensions.

Why is high condition number bad?



Condition number near 1

Small change in x => small change in Ax. Robust to perturbations.



High condition number

Small change in x => large change in Ax for some values of x. Result is sensitive to small perturbations in x.

Rule of thumb for error in A\b

- Compute k = cond(A). You will lose ***maximum*** log10(k) digits of accuracy when solving Ax = b. (This is an ***upper bound***.)
- Examples:
  - Consider doubles. They provide 16 digits. Take A with cond(A) = 1000. Solving Ax = b provides x values good to roughly 13 digits.
  - Consider floats. They provide 7 digits. Take A with cond(A) = 1000. Solving Ax = b provides x good to 4 digits.

Demonstration

/home/sdb/Northeastern1/Class4/ConditionError

```
octave:21> test_condition_error(10)
ans =  6.5403e-16
octave:22> test_condition_error(10)
ans =  8.2444e-16
octave:23> test_condition_error(10)
ans =  9.0921e-16
octave:24> test_condition_error(10)
ans =  1.1156e-15
octave:25> test_condition_error(10)
ans =  1.5464e-15
octave:26>
octave:26>
octave:26>
octave:26> test_condition_error(1000)
ans =  3.1365e-14
octave:27> test_condition_error(1000)
ans =  8.3084e-14
octave:28> test_condition_error(1000)
ans =  1.2118e-13
octave:29> test_condition_error(1000)
ans =  1.6971e-14
octave:30> test_condition_error(1000)
ans =  1.3272e-14
octave:31> test_condition_error(1000)
ans =  1.6214e-13
```

1. Generate random A, x
2. Compute $b = Ax$
3. Compute and print residual

$$r = Ax - b$$

Computing the condition number

- Never directly compute $\|A\| \cdot \|A^{-1}\|$
- Condition numbers are generally computed as a byproduct to a solver algorithm (e.g. LU).
- As an end-user, you should use call a condition number routine.
  - LAPACK has several condition number routines for different types of matrices, using different norms.
  - MATLAB has several routines which wrap LAPACK's implementations, including cond().

Summary: Triangle of concepts

SVD

$$A = U \Sigma V^T$$

$$\Sigma = \begin{pmatrix} \sigma_1 & & 0 & \\ & \sigma_2 & & \cdots \\ 0 & & \sigma_3 & \end{pmatrix}$$

Matrix norm (induced norm)

$$\|A\| = max\left(\frac{\|Ax\|}{\|x\|} : x \in K^n\right)$$
$$= \sigma_{max}$$

Matrix condition number

$$k = \frac{\sigma_{max}}{\sigma_{min}}$$
$$= \|A\| \cdot \|A^{-1}\|$$

Concept: Matrix rank

- Consider matrix A of size [N, M].
- rank(A) is number of linearly independent cols/cols in A.
- rank(A) is number of non-zero singular values of A.

Non-singular matrix

```
>> A = rand(3,4)

A =

    0.7248    0.1833    0.6014    0.2990
    0.3741    0.9401    0.0266    0.3543
    0.7022    0.2276    0.7808    0.1509

>> rank(A)

ans =

     3

>> svd(A)

ans =

    1.6103
    0.8492
    0.1516
```

Matrix has full rank – all columns/rows are linearly independent

All singular values are non-zero

Singular matrix

```
>> B = [1 2 3 4; -2 4 -6 8; 2 4 6 8]

B =

     1     2     3     4
    -2     4    -6     8
     2     4     6     8

>> rank(B)

ans =

     2

>> [U, S, V] = svd(B)

U =

   -0.3630    0.2612    0.8944
   -0.5840   -0.8118         0
   -0.7261    0.5223   -0.4472


S =

   13.4970         0         0         0
         0    9.3718         0         0
         0         0    0.0000         0


V =

   -0.0480    0.3126    0.9486    0.0099
   -0.4420   -0.0678    0.0093   -0.8944
   -0.1439    0.9377   -0.3162   -0.0033
   -0.8841   -0.1356   -0.0047    0.4472
```

Matrix is not full rank – only two of three rows are linearly independent

Notice zero singular value on diagonal

Matlab implementation of rank

- Count the number of singular values larger than some tolerance.

```
s = svd(A);
tol = max(size(A))*eps(max(s));
r = sum(s > tol);
```

- Tolerance depends upon:
  - Largest sized dimension of matrix
  - Scaling factor deduced using max singular value.

- Tolerance algorithm needed to handle matrices with high condition number.

SVD and matrix inverse

- A square matrix has an inverse if all singular values are non-zero.

$$A = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_N \end{bmatrix} V^T$$

$$A^{-1} = V \begin{bmatrix} 1/\sigma_1 & & \\ & \ddots & \\ & & 1/\sigma_N \end{bmatrix} U^T$$

Matrix inverse

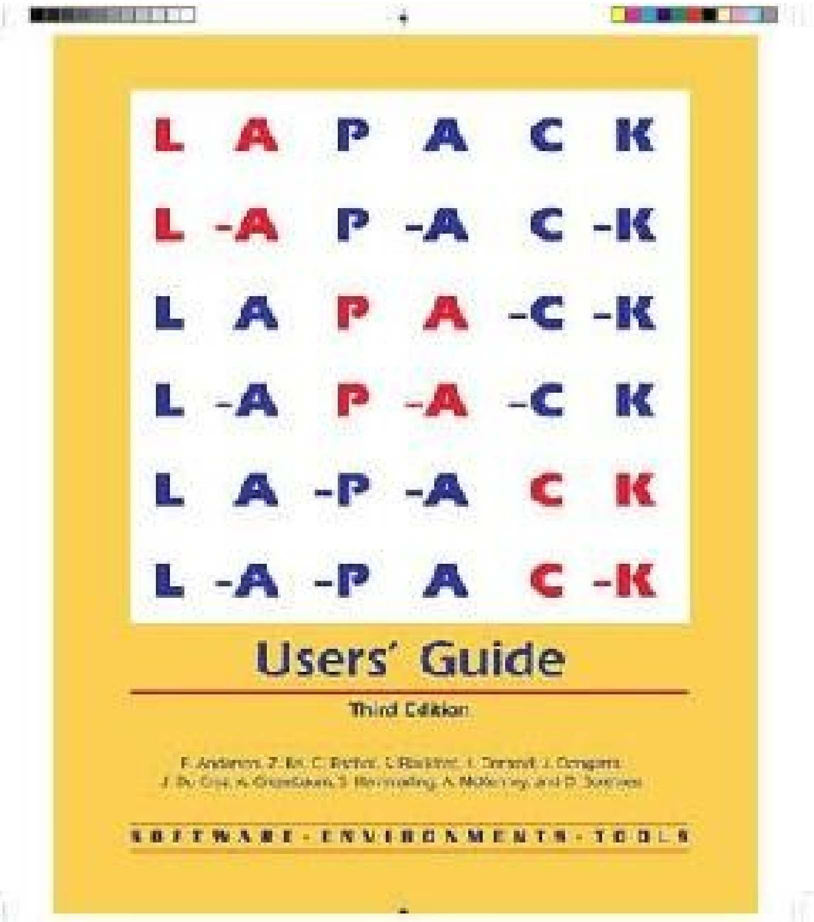
- If one or more singular values are zero, the matrix has no inverse (i.e. it is singular)

$$A = U \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & 0 \end{pmatrix} V^T$$

$$A^{-1} = V \begin{pmatrix} 1/\sigma_1 & & \\ & \ddots & \\ & & 1/0 \end{pmatrix} U^T$$

Problem

Important library LAPACK

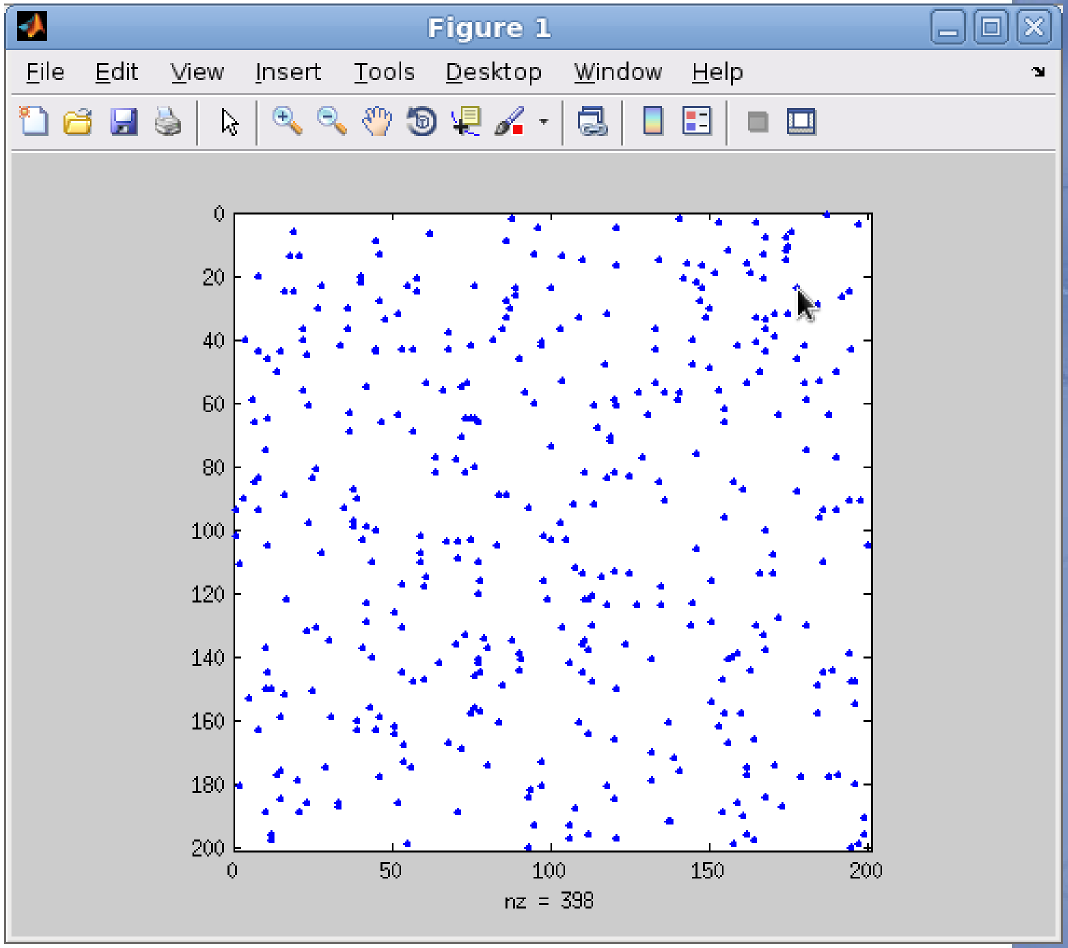


- LAPACK contains routines used for many linear algebra computations:
  - SVD
  - Solvers
  - Eigendecompositions
  - Etc.
- Built on top of BLAS.

http://www.netlib.org/lapack/lug/lapack_lug.html

More matrix visualizations: Nonzeros of a large, sparse matrix

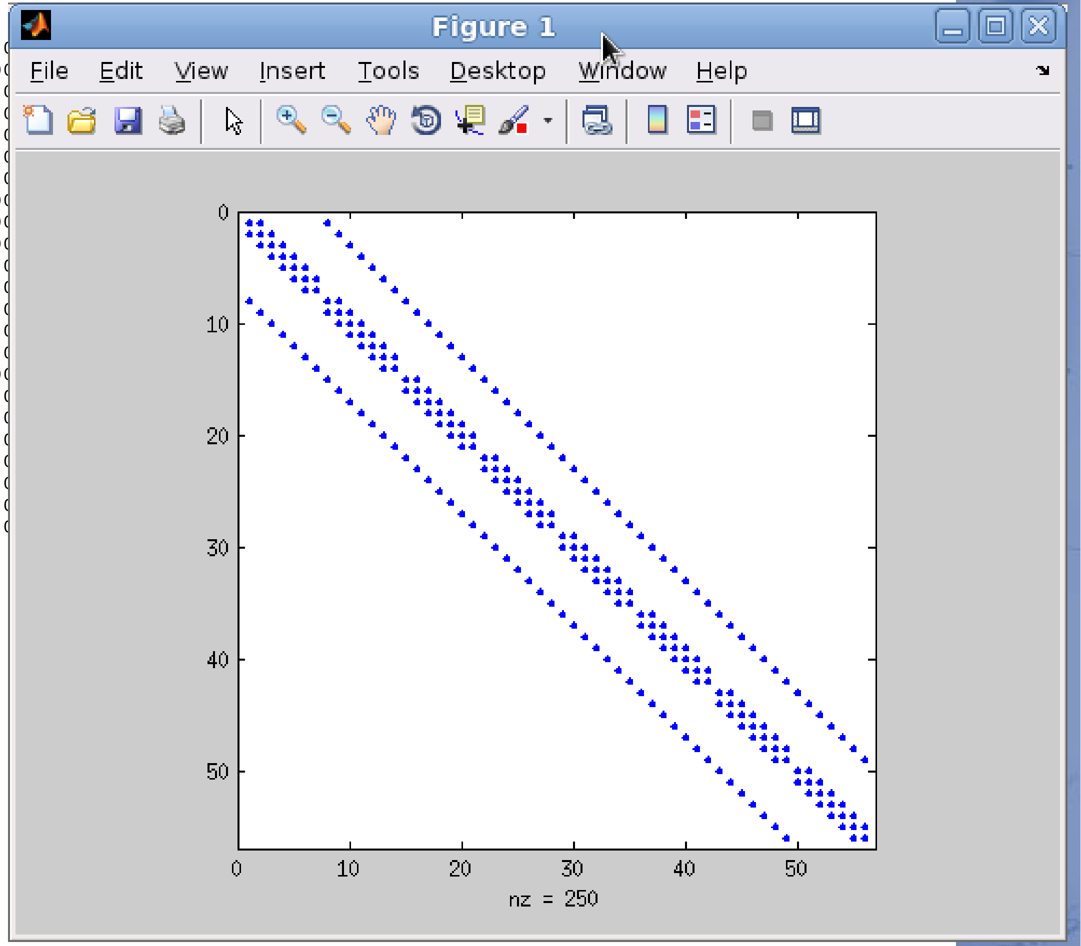
```
>> A = sprandn(200, 200, .01);
>> nnz(A)

ans =

   398

>> 200*200

ans =

       40000

>> spy(A)
```



Matlab “spy” command

Sparsity patterns of common matrices

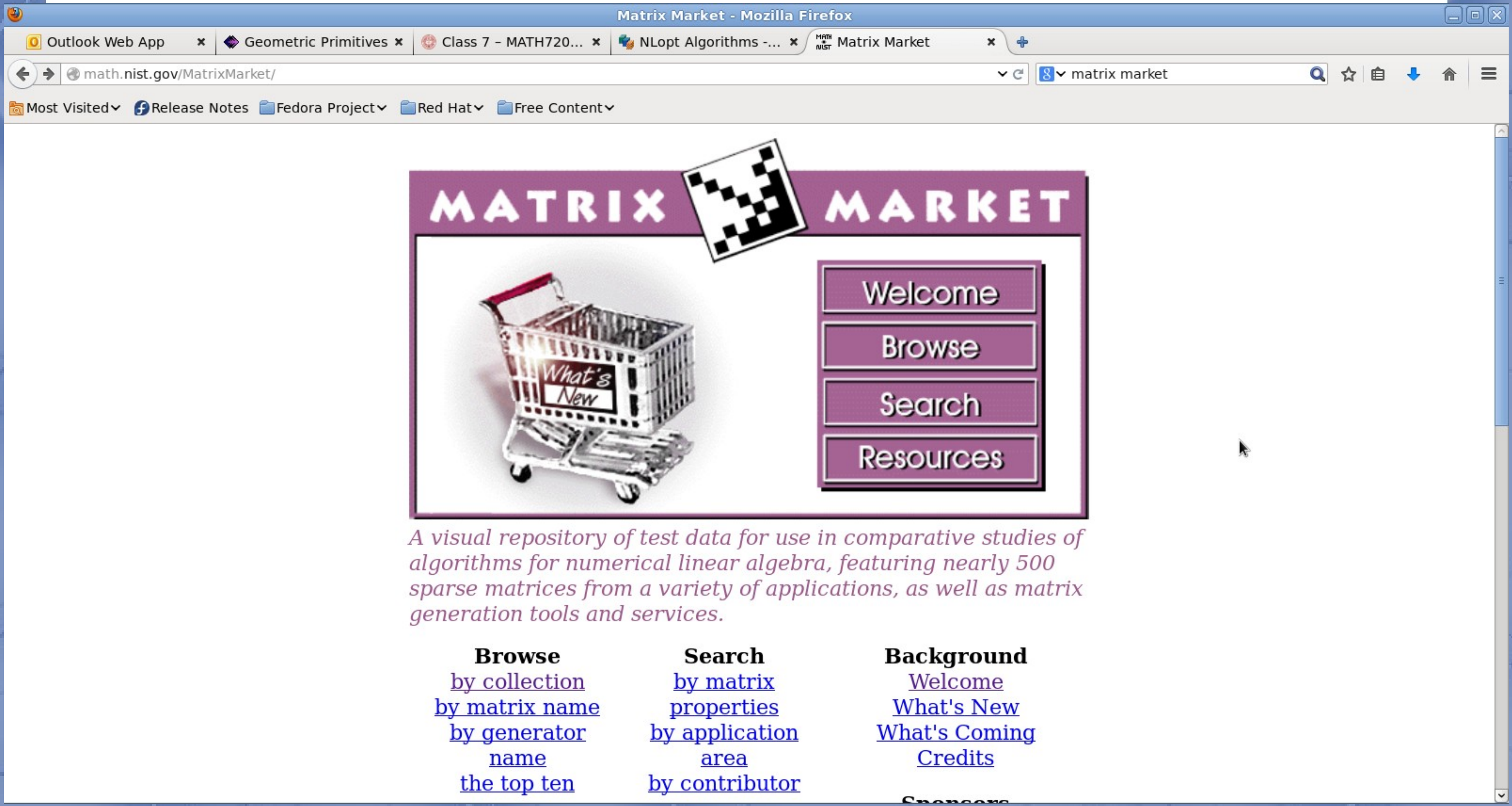
```
>> full(A)

ans =

  Columns 1 through 14

    0.4000    0.4000         0         0         0         0         0    0.4000         0
    0.4000    0.4000    0.4000         0         0         0         0         0    0.400
         0    0.4000    0.4000    0.4000         0         0         0         0         0
         0         0    0.4000    0.4000    0.4000         0         0         0         0
         0         0         0    0.4000    0.4000    0.4000         0         0         0
         0         0         0         0    0.4000    0.4000    0.4000         0         0
         0         0         0         0         0    0.4000    0.4000         0         0
    0.4000         0         0         0         0         0         0    0.4000    0.400
         0    0.4000         0         0         0         0         0    0.4000    0.400
         0         0    0.4000         0         0         0         0         0    0.400
         0         0         0    0.4000         0         0         0         0         0
         0         0         0         0    0.4000         0         0         0         0
         0         0         0         0         0    0.4000         0         0         0
         0         0         0         0         0         0    0.4000         0         0
         0         0         0         0         0         0         0    0.4000         0
         0         0         0         0         0         0         0         0    0.400
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0         0
```



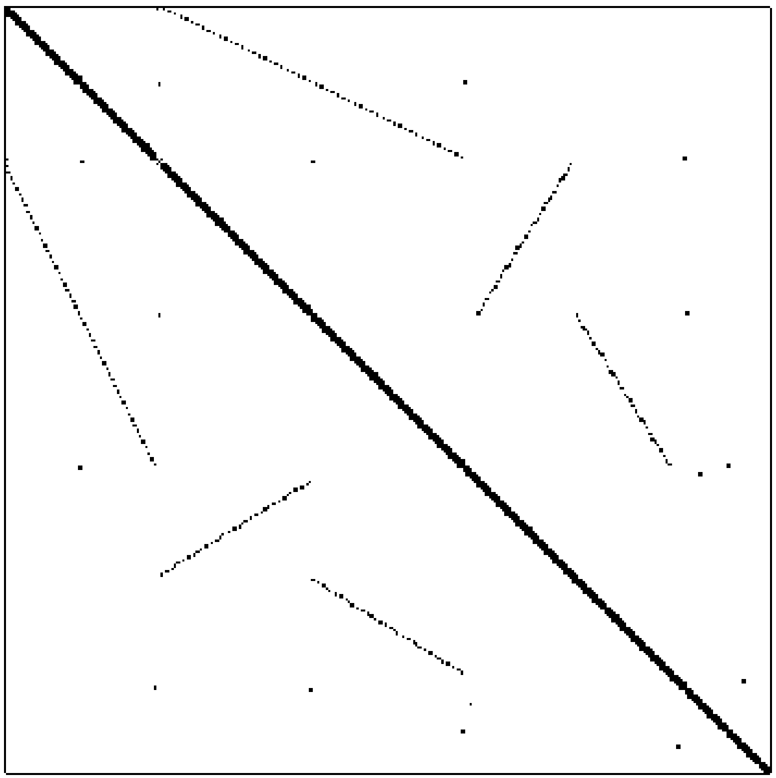
Finite difference propagator matrix for 2D wave equation

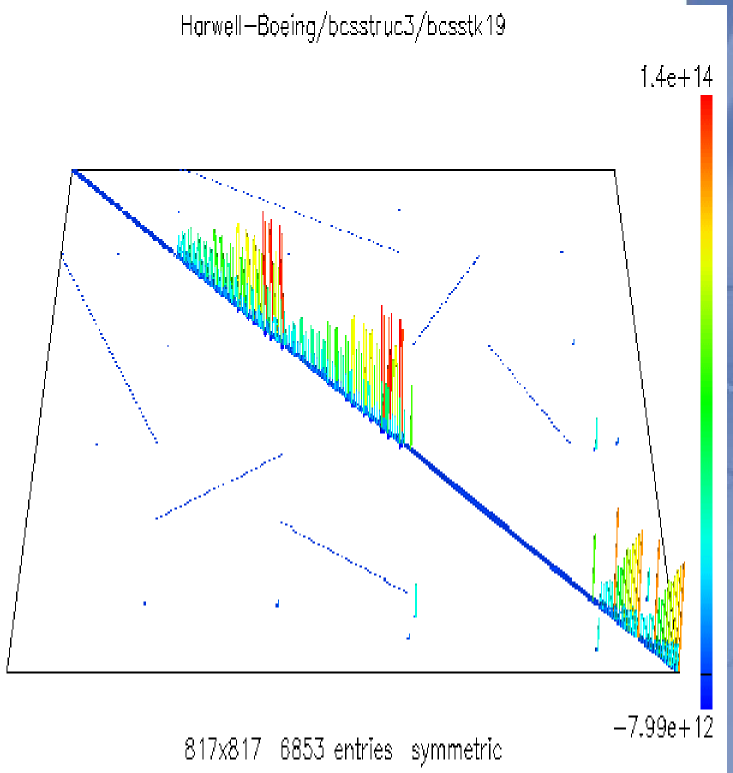
Browse others at: http://math.nist.gov/MatrixMarket/

NIST Matrix Market – sparse matrix collection



Typical FEM matrices





- Matrix BCSSTK19: BCS Structural Engineering Matrices (eigenvalue problems)
  - Part of a suspension bridge

Next topic: Visualizing a matrix as a quadratic form

- Another visualization: Consider quadratic forms.

$$f(u)=u^T B u$$

where *f* is scalar function of input vector *u*.

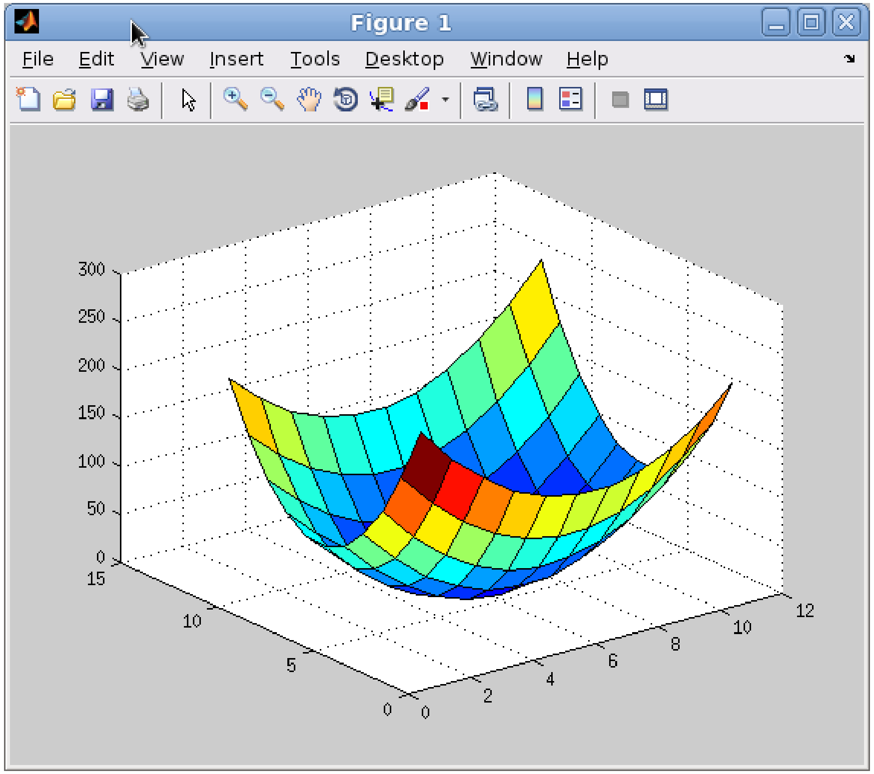
- Visualization works for *square* matrix.
- The idea is to plot the values of *f(u)* vs *u*.
- If all eigenvalues of *B* are positive, then *f(u)* is a parabola opening upward.

Positive definite matrix

- All eigenvalues positive <=> matrix is positive definite.
- Consider $f(u)=u^T B u$ where *u* = *[x; y]*
- If *B* is positive definite, then *f(u)* is a parabola opening upward.
- Points obeying

$$f(u)=u^T B u=1$$

form an ellipse.



/home/sdb/Northeastern/Class5/PositiveDefinite

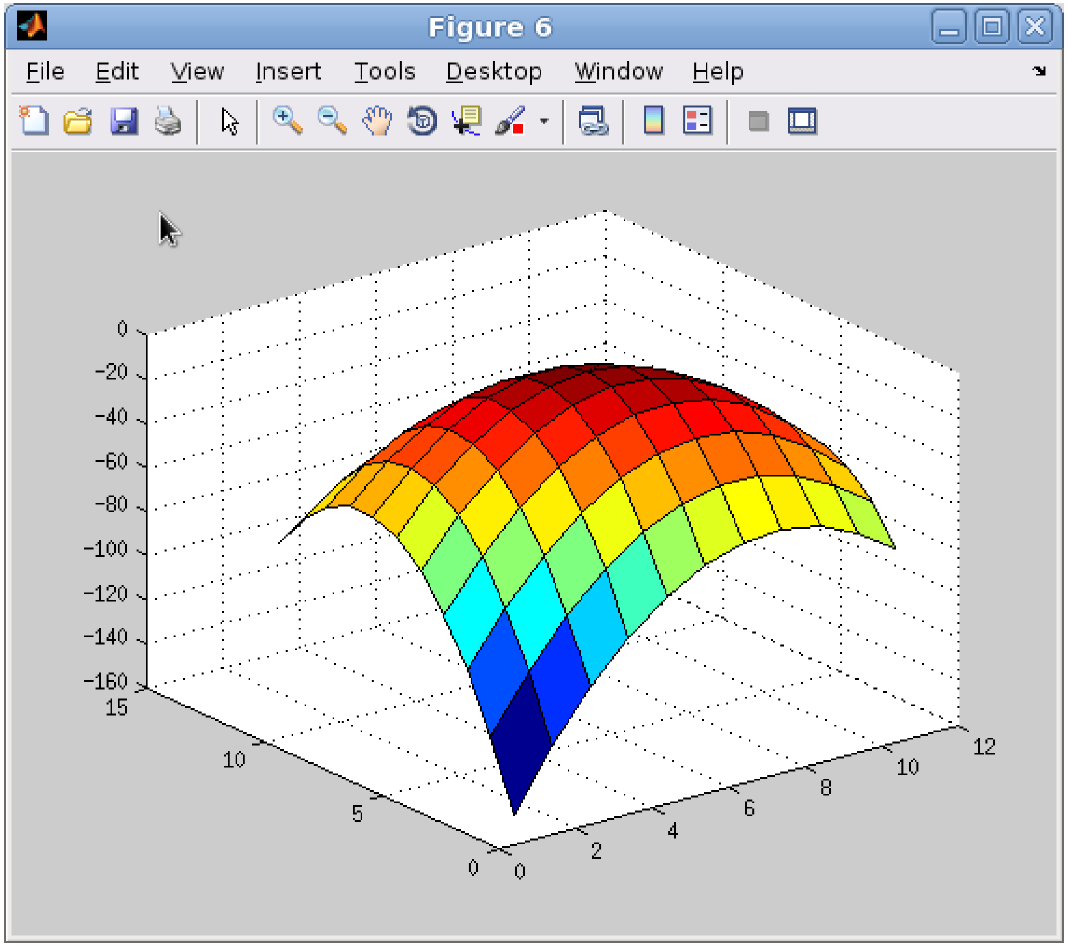
What if A is negative definite?

```
>> B = randn_cond(2, 2, 1.3)

B =

    1.4193    0.1978
    0.2916    1.5877

>> A = -B'*B

A =

   -2.0995   -0.7437
   -0.7437   -2.5599

>> eig(A)

ans =

   -3.1082
   -1.5512

>> plot_surface(A)
```

What surface corresponds to a negative definite matrix?

What if A is negative definite?

```
>> B = randn_cond(2, 2, 1.3)

B =

    1.4193    0.1978
    0.2916    1.5877

>> A = -B'*B

A =

   -2.0995   -0.7437
   -0.7437   -2.5599

>> eig(A)

ans =

   -3.1082
   -1.5512

>> plot_surface(A)
```



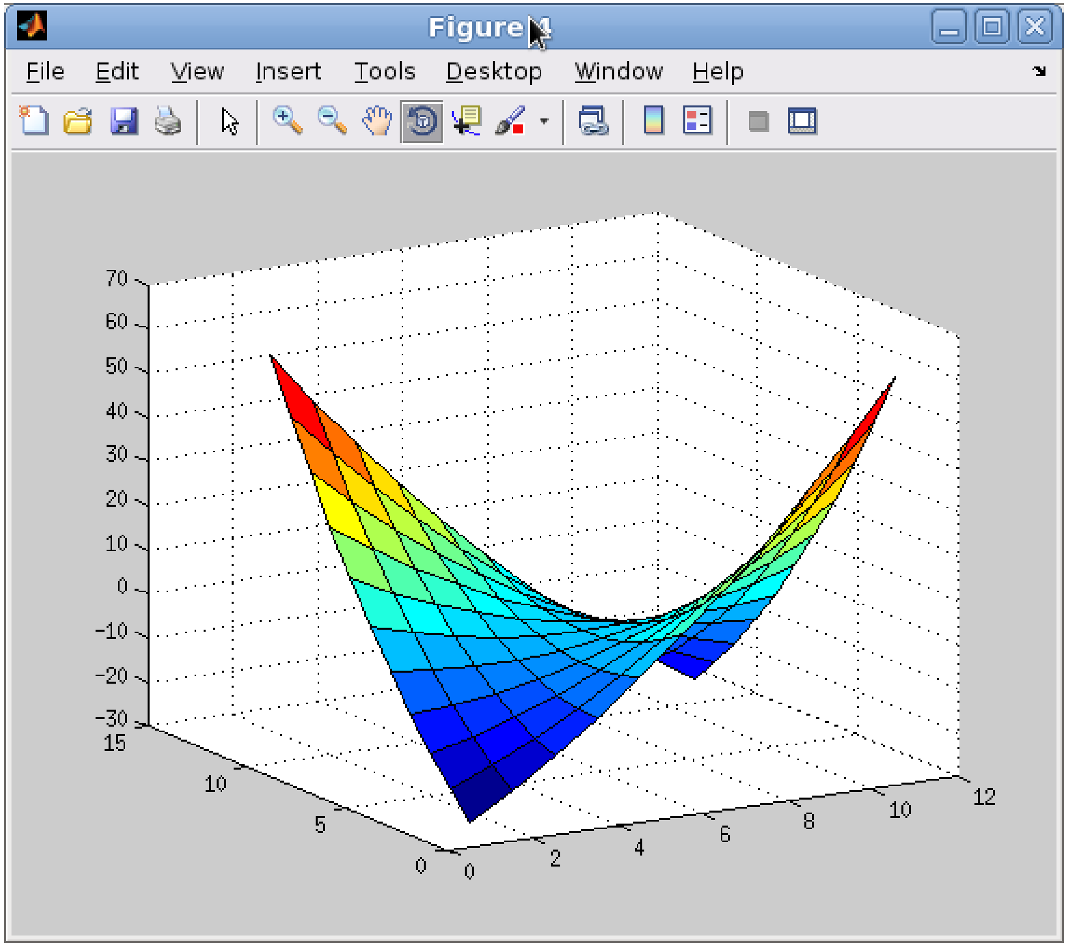
If matrix is neither positive nor negative definite?

```
>> cd PositiveDefinite/
>> B = randn(2)

B =

   0.3188   -0.4336
  -1.3077    0.3426

>> eig(B)

ans =

  -0.4224
   1.0838

>> plot_surface(B)
```

What surface corresponds to an indefinite matrix?

If matrix is neither positive nor negative definite?

```
>> cd PositiveDefinite/
>> B = randn(2)

B =

    0.3188   -0.4336
   -1.3077    0.3426

>> eig(B)

ans =

   -0.4224
    1.0838

>> plot_surface(B)
```

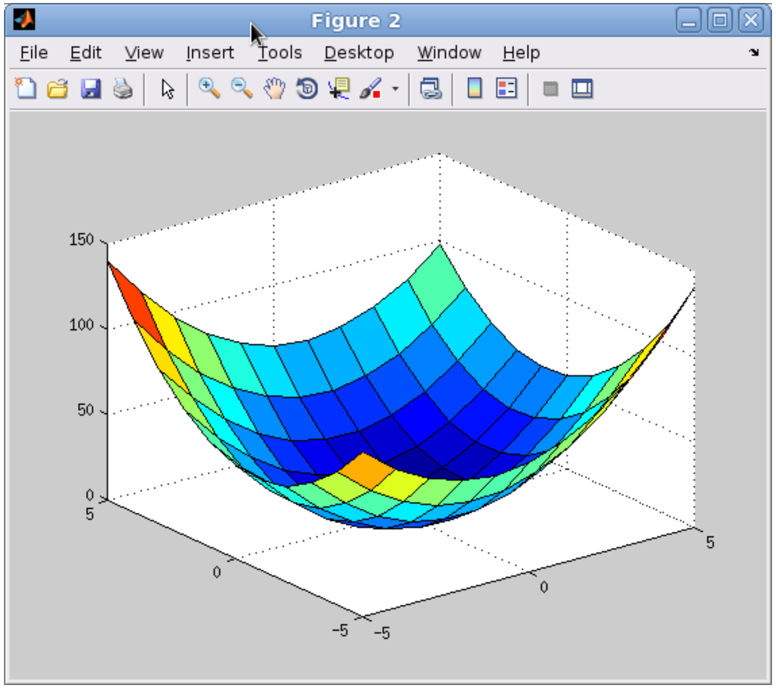


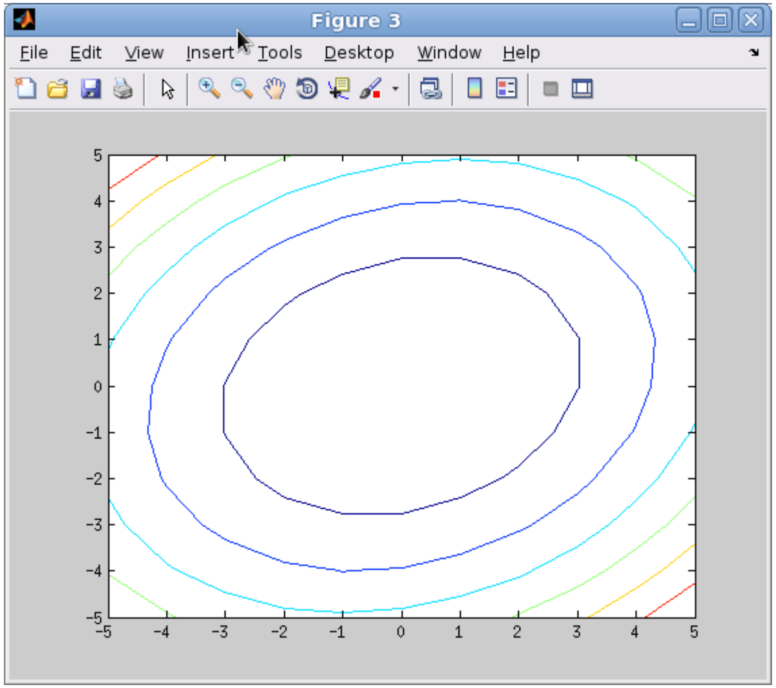
Solution set of $f(u)=u^T B u=1$ is a hyperbola

Visualization

- Consider $f(u)=u^T B u=c$ for some c.
- If NxN matrix is positive definite (which many useful matrices are), then think of the solution set *u* as an N-Dimensional ellipsoid.
  - Condition number is ratio of longest to shortest semi-axis of ellipse.
- If NxN matrix is not positive definite (positive and negative eigenvalues), think of the surface as a complicated mixture of hyperboloids and ellipsoids in some N-dimensional space.

Positive definite matrix





Parabola from quadratic form

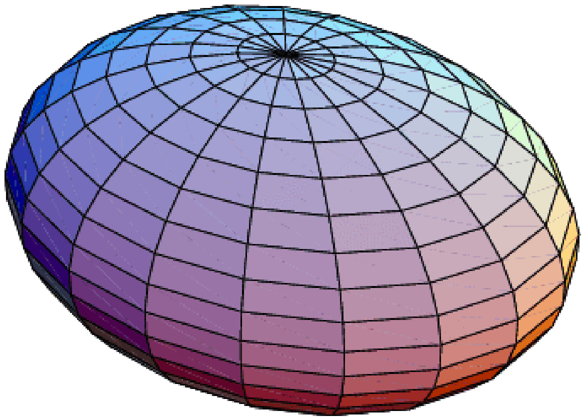
$f(u) = u^T B u$

Contours of equal height

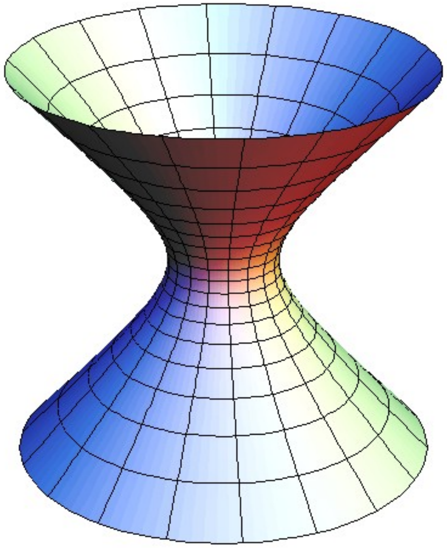
$f(u) = u^T B u = c$

Extension to 3D (and beyond...)

- For SPD matrix, think of ellipses in ND space



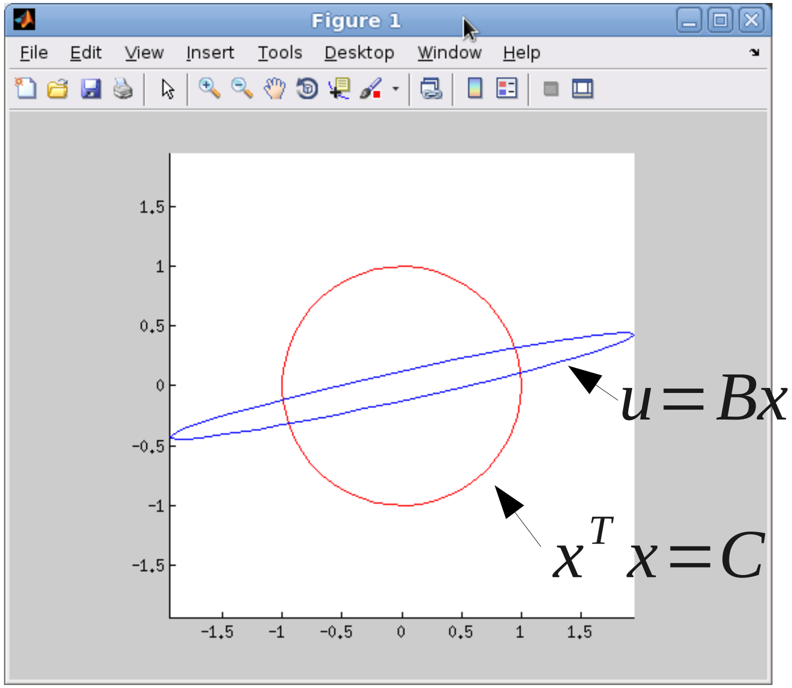
- For symmetric-indefinite matrix, think of combinations of ellipses and hyperbolas



Applicable to symmetric matrices.

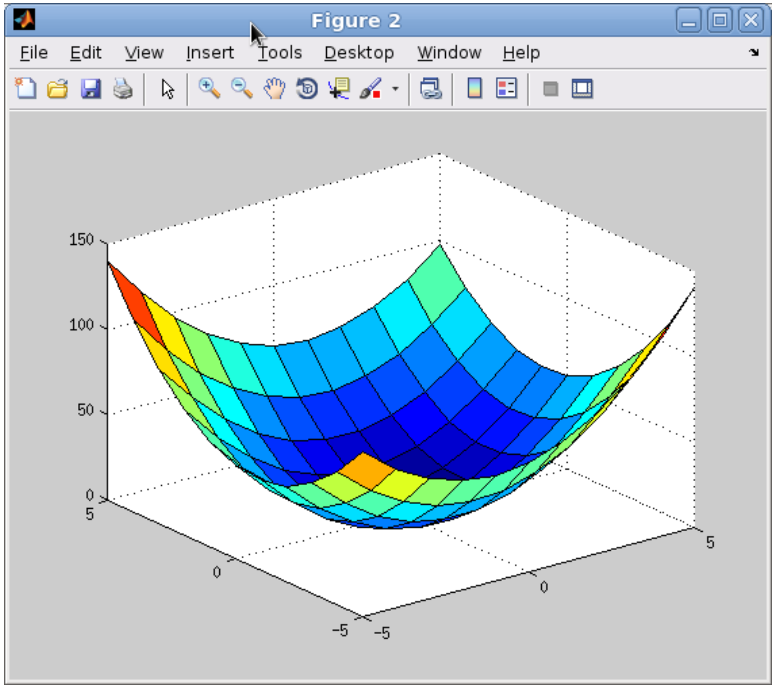
Now tie the two visualizations together

- Matrix as transform (any matrix)
- Matrix as quadratic form (symmetric only)



$u = Bx$

$x^T x = C$



$u^T A u = C$

- I claim: $A = (B^{-1})^T (B^{-1})$

Proof

- I claim: $A = (B^{-1})^T (B^{-1})$
- Start with: $u^T A u$
- Recall: $u = Bx$
- So:

$$(x^T B^T) A (B x)$$

$$(x^T B^T)(B^{-1})^T (B^{-1})(B x)$$

- Use: $B^T (B^{-1})^T = (B^{-1} B)^T = I$

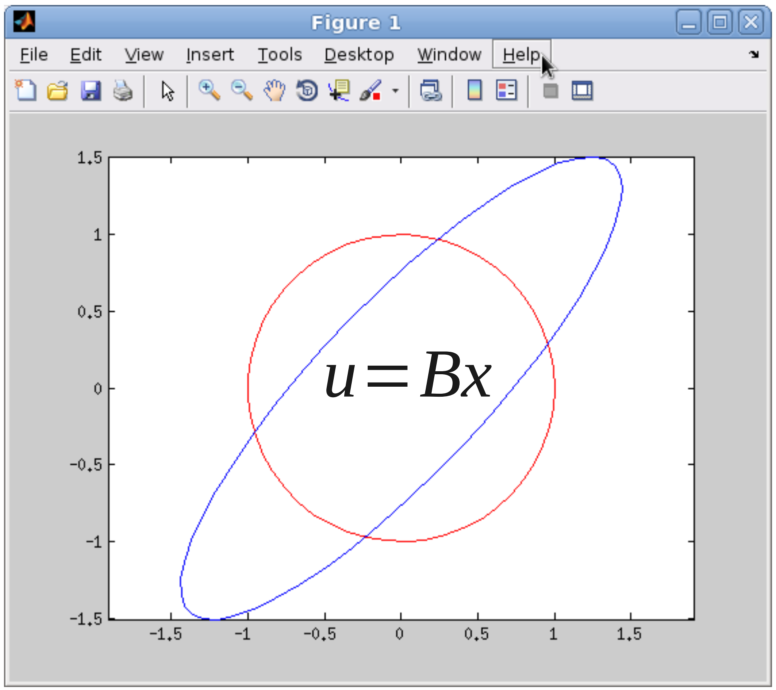
$$x^T (I)(I) x$$

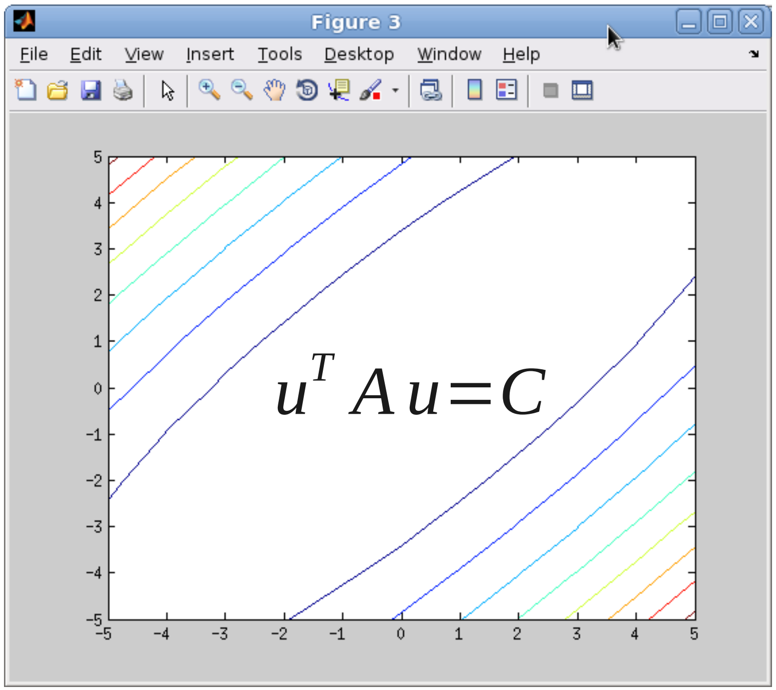
$$= x^T x = C$$

Q.E.D.

Comparison of B & A

- Output of program which plots $u=Bx$ and levelsets of $u^T A = C$





$$A=(B^{-1})^T(B^{-1})$$

- Note angle of ellipses is the same

~/MatrixVisualizations/PositiveDefinite

Main points made in this session

- Concepts: matrix norm, SVD, condition number, and rank.
- These concepts are all linked by the SVD.
- Visualize matrix by its effect on a circle.
  - Works for any matrix.
- Visualize matrix via quadratic form.
  - Works for square, symmetric, positive definite matrix.