Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 1: This week, I attended all the three lecture of Applied Math I and the major topic was going over linear algebra.

References or papers consulted:

- *Convergence Analysis of the Nonlinear Coarse-Mesh Finite Difference Method for One-Dimensional Fixed-Source Neutron Diffusion Problem*
- *A Comparison of Coarse Mesh Rebalance (CMR) and Coarse Mesh Finite Difference (CMFD) Acceleration Methods for the Neutron Transport Calculations*

Things I learned:

Went Over

- Eigenvalue and Eigenvector
- Finite element method
- Matrix Factorization
- Gauss Elimination
- LU factorization Jordan decomposition and Schur decomposition $A = QUQ^{-1}$

New

- Condition number
  We compute $f(x + \delta x) - f(x) \approx |\delta x| \cdot |f'(x)|$ and $|f'(x)|$ is absolute condition number.
  We compute $\frac{|f(x+\delta x)-f(x)|}{|f(x)|} \approx \frac{|\delta x|}{|x|} \cdot \frac{|f'(x)|\cdot|x|}{|f(x)|}$ and $|f'(x)|$ is relative condition number.
- Absolute error and relative error: $|y - \bar{y}|$ and $\frac{|y-\bar{y}|}{|y|}$
- Backward error δx could be defined by $alg(x) = f(x + \delta x)$. This error is used for verify the stability of an algorithm.
- Round-off effects The results could lose its digits by a bad algorithm,etc switch rows before using Gauss elimination.
- Floating Point Algorithm A float number stored in computer included sign, fraction, base and exponent. An IEEE double precision has 64 total bits.
- overflow/underflow threshold are the maximum and minimum value of an certain standard float number. etc, the underflow of IEEE float is 2^{1022} and overflow is 2^1024
- machine epsilon ϵ Machine epsilon could be define as $fl(a) = a(1 + \delta)$ $|\delta| \leq \epsilon$ when all the numbers are not beyond the overflow/underflow.

Software Code:

- Create a repository on Github for all the assignments and Journals in the rest of the semester.

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Also, only for this week: Personal Intro

Hi, my name is Leidong Xu. And I have been studying mechanical and nuclear engineering in K-state for five years. This academic year will be the second year for my master degree. My thesis is related to using numerical method solving engineering and science problems on Computational Fluid Dynamics and Nuclear systems, which include preconditioner(sparse matrix) and geometric multigrid method.

I did not take any pure mathematic class in the past 3 years, and almost all the courses I took last year were using programming to solve PDEs and ODEs. Things get boring when I continue stayed in finite difference all the time. This also the main reason I come to this class, I hope I can have a better understanding of linear algebra and have a better understanding of those numerical methods in a view of applied math.

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 2: This week, I studied the rest parts of Chapter 1 which included vector and matrix norm and also a good start of Chapter

References or papers consulted:

- *Linear Independence Oracles and Applications to Rectangular and Low Rank Linear Systems*

Things I learned:

- $\mathbb{R}^{(3x3)}$ means real number matrix and $\mathbb{C}^{(3x3)}$ means complex number matrix.
- some norm axioms
- P-norm could be defined as $||x||_p = (\Sigma_i |x_i|^p)^{1/p}$.
- Euclidean norm: p-norm when p = 2
- taxicab norm: p-norm when p = 1
- $\infty - norm$ or infinity norm: p-norm p = ∞
- Cauchy-Schwartz inequality: $| < x, y > | \leq \sqrt{< x, x > \cdot < y, y >}$
- Inner product: Hard to give a definition, but could be think as a space created by vectors and scalars, related Legendre Polynomial.
- General strategy to show a = 0: a= -a
- Max-norm: $max_{ij}|a_{ij}$ and Frobenius norm :$(\Sigma|a_{ij}|^2)^{1/2}$
- Spectral theory for symmetric matrix
- An operator norm is a matrix norm.
- Matrix norm could be think as the maximum strectch.
- How to compute singular value of an matrix
- residual: $r = A\hat{x} - b$
- condition number of a matrix: $\kappa(A) = ||A^{-1}|| \cdot ||A||$

Software Code:

- Finished programming hw 1. The questions I picked are Problem.32 Most nonzero elements in row and Problem 17. Find all elements less than 0 or greater than 10 and replace them with NaN.
- These problems are pretty easy, and good to know we can define NaN by float('nan').

```
'''
Problem 32. Most nonzero elements in row
Given the matrix a, return the index r of the row with the most nonzero
elements. Assume there will always be exactly one row that matches this
```

```
criterion.
'''

import numpy as np


def mostnonzero(matrix):
    row = []
    max_zero = 0
    for i in range(matrix.shape[0]):
        N_nonzero = 0
        for j in range(matrix.shape[1]):
            if matrix[i, j] != 0.:
                N_nonzero += 1
        if max_zero < N_nonzero:
            row = [i]
            max_zero = N_nonzero
        elif max_zero == N_nonzero:
            row.append(i)
    row = np.array(row) + 1
    print('the' + ' ' + str(row) + 'th row has the maxinum nonzero element')
    return row


'''
Find all elements less than 0 or greater than 10 and replace them with NaN
'''


def replacenan(matrix):
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            if matrix[i, j] > 10 or matrix[i, j] < 0:
                matrix[i, j] = float('nan')
    return matrix
```

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 3: This week, I went over some knowledge about series and condition number. I was lost those content belong to which chapter on the book.

References or papers consulted:

- *What every computer scientist should know about floating-point arithmetic*

Things I learned:

- Pi notation: $\prod_{n}^{i=m} x_i = x_m \cdot x_{m+1} \cdot x_{m+2} \cdot \cdots \cdot x_n$
- Vandermonde matrix is a matrix have a geometric series in each row, $V_{i,j} = \alpha_i^{j-1}$ and the determinant of Vandermonde matrix is $\prod_{1 \le i < j \le n} (\alpha_j - \alpha_i)$, for example

```
det (  [[  1,   1,   1,   1],
        [  1,   2,   4,   8],
        [  1,   3,   9,  27],
        [  1,   5,  25, 125]]) = (5-3)*(5-2)*(5-1)*(3-2)*(3-1)*(2-1)
```

- ill-condition: A problem with a low condition number is said to be well-conditioned, while a problem with a high condition number is said to be ill-conditioned.
- geometric series:$1 + x + x^2 + x^3 + x^4 + \cdots x^n = \sum_{k=0}^{\infty} x^k$
- geometric interpretation
- If the vector space X is finite dimensional, all norms are equivalent.

Software Code:

- I have not finished PA 2, but there are two new numpy functions and some latex functions.
- numpy.finfo: Machine limits for floating point types. For example:

```
print(np.finfo(float).eps)
# 2.22044604925e-16

print(np.finfo(np.float32).eps)
# 1.19209e-07
```

- numpy.vander: Generate a Vandermonde matrix. Most common example:

```
>>> x = np.array([1, 2, 3, 5])
>>> np.vander(x, increasing=True)
array([[  1,   1,   1,   1],
       [  1,   2,   4,   8],
       [  1,   3,   9,  27],
       [  1,   5,  25, 125]])
```

- $\mathbb{R}$ by

```
\mathbb{}
```

- $\sqrt{< x, x >}$ by

```
\sqrt{}
```

- Pi notation $\prod_{i=1}^{n} a_i$ by

```
\prod_{i = 1}^{n} a_{i}
```

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 4: This week, I went over some knowledge about different matrix factorization.

References or papers consulted:

- *Model order reduction using DMD modes and adjoint DMD modes*

Things I learned:

- SPD: symmetric positive-defined matrix.
- Singular value can be thought dimensional, etc, largest singular value is $||A||_2$, which could be treated as the length of semi-major axis.
- GramSchmidt process
- LU factorization
- leading principal: The determinant of a principal submatrix is called the principal minor of A. The leading principal submatrix of order k of an n n matrix is obtained by deleting the last n k rows and column of the matrix.
- Matrix Minor: the remain matrix from delecting some rows and columns from original matrix.
- Permutation matrix: a matrix have a '1' at every row and column, all others are zero.
- Cholesky factorization(Cholesky decomposition): a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose.

Software Code:

- N/A

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 5: This week, I went over some knowledge about Cholesky factorization and finite difference method. I'm pretty familiar with this method but I have never think about adjust the matrix feature to achieve a better performance when using directly method.

References or papers consulted:

- *On Dynamic Mode Decomposition: Theory and Applications*

Things I learned:

- ONB: Orthonormal Basis
- spectral theorem for symmetric matrices
- Two different ways to think matrix multiplication
- Change of Variables
- Gershgorin circle theorem
- Eigenvalue of A is real $\rightarrow$ All the element of A are strictly positive

Software Code:

- The machine epsilon of float 64 (double precision) is around 2.220446049250313e-16, and the smallest nonzero number could be shown is around 1.1102230246251565e-16.
- np.log could directly apply on a array, so do not use for loop.
- machine epsilon could be import by np.finfo(float).eps
- numpy.arange could be a alternative of np.linspace. Instead of input how many points needed, we could define the start point, end point and the space.

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 6: This week, I went over some knowledge about Cholesky factorization and finite difference method. I'm pretty familiar with this method but I have never think about adjust the matrix feature to achieve a better performance when using directly method.

References or papers consulted:

- *Dynamic mode decomposition of numerical and experimental data*

Things I learned:

- How to solve Least Squares:
- (1)Form $A^TA \& A^Tb$
- (2)Cholesky $A^TA = L^TL$
- (3)Solve $L^Tw = A^Tb$
- (4)Solve $Lx = w$
- rank nullity
- dim(A) denotes the dimension of a vector space A.
- QR decomposition : a decomposition of a matrix A into a product A = QR of an orthogonal matrix Q and an upper triangular matrix R.
- Normal Equation: $A^TAx = A^Tb$ where A^TA is a normal matrix.

Software Code:

- **numpy.polyfit(x, y, deg, rcond=None, full=False, w=None, cov=False)**:Least squares polynomial fit.Fit a polynomial p(x) = p[0] * x**deg + ... + p[deg] of degree deg to points (x, y). Returns a vector of coefficients p that minimises the squared error.

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 7: This week, we talked about mainly SVD-singular value decomposition and some other related topics which included least square with SVD, lower rank approximation and pseudo-inverse.

References or papers consulted:

- Continue read *Dynamic mode decomposition of numerical and experimental data*

Things I learned:

- null space: If T is a linear transformation of R^n, then the null space $Null(T)$, also called the kernel $Ker(T)$, is the set of all vectors X such that$T(X) = 0$
- column space: a matrix A is the span (set of all possible linear combinations) of its column vectors
- low-rank approximation: In mathematics, low-rank approximation is a minimization problem, in which the cost function measures the fit between a given matrix (the data) and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank. $A = \Sigma \sigma_j u_j v_j^T$
- pseudo-inverse

Software Code:

- array[xxxx], where xxxx could be an statement instead of index. I have used python for over one year and feel so surprised about this feature. This feature achieve restriction or selection in a single line of code. etc. Diag[Diag==np.amin(Diag)] *= 10**(-u*16), replaced the mininum values in this array.
- **numpy.random.rand :** Create an array of the given shape and populate it with random samples from a uniform distribution over [0, 1).
- **numpy.random.normal :** Draw random samples from a normal (Gaussian) distribution.
- **matplotlib.pyplot.subplot :** Plot multiple pictures in a specific location and order.
- Plot np.array([value1, value2]) and np.array([value3, value4]) directly could plot a straight line from (value1,value3) to (value2,value4).

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 8: This week, we talked about mainly pseudo-inverse, gradient of linear operator and some other concepts of numerical example of applying SVD.

References or papers consulted:

- *None*

Things I learned:

- if web define $A = \sum \sigma_j u_j v_j$ and $A^+ = \sum \sigma_j^{-1} u_j v_j$, then $AA^+ = \sum u_j u_j^T$.
- Gradient matrix
- full rank: every row and column is not independent of each other.
- rank deficient matrix: A matrix is said to have full rank if its rank is either equal to its number of columns or to its number of rows (or to both). A matrix that does not have full rank is said to be rank deficient.
- $div_x(f) = \sum f(x, y)$
- algebraic connectivity: The algebraic connectivity (also known as Fiedler value or Fiedler eigenvalue) of a graph G is the second-smallest eigenvalue of the Laplacian matrix of G.[1] This eigenvalue is greater than 0 if and only if G is a connected graph. This is a corollary to the fact that the number of times 0 appears as an eigenvalue in the Laplacian is the number of connected components in the graph. The magnitude of this value reflects how well connected the overall graph is. It has been used in analysing the robustness and synchronizability of networks.
-

Software Code:

- None

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 9: This week, the main work I have done is review the basic linear algebra knowledge for the close-book mideterm.

References or papers consulted:

- N/A

Things I learned:

- det(I) = 1
- det(singular matrix) = 0
- det((A^T) = det(A)
- det(A^{-1}) = 1/ det(A)
- det(AB) = det(A)det(B)
- det(cA) = $c^n det(A)$ where n is the matrix demension
- det(triangular matrix) = $\prod_{i=1}^n a_{ii}$
- det (I + uv^T) = 1 + uv
- det (X + AB) = det(X)det(T+$BX^{-1}A$)
- cramer rule: $x_i = \frac{det(A_i)}{det(A)}$ where is A_i is A replace $A[:, i] = b$
- trace(A) = $\sum_1^n a_{ii}$
- trace(A+B) = trace(A) + trace(B)
- trace(A^T) = trace(A)
- trace(AB) = trace(A)trace(B)
- trace(ABC) = trace(BCA) = trace(CAB)
- $||x||_p = (\sum |x_i|^p)^{1/p}$
- $||A||_1 = max_{1 \leq j \leq n} \sum_{i=1} |a_{ij}|$ the maximum value of the sum of column vector
- $||A||_\infty = max_{1 \leq i \leq n} \sum_{j=1} |a_{ij}|$ the maximum value of the sum of row vector
- $||A||_2 = trace(A^T A) = \sigma_{max}(A) = \sqrt{\lambda_{max}(A^T A)}$
- $||A||_F \sqrt{\sum \sum |a_{ij}|^2}$
- cond = $\frac{\sigma_1(A)}{\sigma_n(A)} = = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$
- cholosky: $A = LL^T$
- QR decomposition: $A = QR$, where $v_1 = [a_1, a_2 - (q_1^T a_2) \cdot q_1, a_3 - (q_1^T a_3) \cdot q_1 - (q_2^T a_3) \cdot q_2]$, , which is equivalent to $c = \frac{A_T c}{A^T A} A - \frac{B^T c}{B^T B} B$,then $q_n = normalize(v_n)$

Software Code:

- Not Applied

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 10: This week, we only had two lectures and started Chapter 4: Nonsymmetric eigenvalue problems.

References or papers consulted:

- **Calculating Time Eigenvalues of the Neutron Transport Equation with Dynamic Mode Decomposition**

Things I learned:

- Jordan Matrix Decomposition: The Jordan matrix decomposition is the decomposition of a square matrix M into the form $M = SJS^1$ where M and J are similar matrix, J is a matrix of Jordan canonical form, and $S^(- 1)$ is the matrix inverse of S.
- Jordan normal form (Jordan canonical form): a linear operator on a finite-dimensional vector space is an upper triangular matrix of a particular form called a Jordan matrix, representing the operator with respect to some basis. Such a matrix has each non-zero off-diagonal entry equal to 1, immediately above the main diagonal (on the superdiagonal), and with identical diagonal entries to the left and below them.
- A system is called under damped, if damping ratio (denoted by zeta) is less than 1. Here the system oscillates with a gradual decrements to zero.
- A system is called over damped, if the damping ratio is greater than one, here system shows tendencies to achieve equilibrium without oscillating.
- A system is called critically damped if damping ration for the system is exactly one. Here system shows the tendencies to come to equilibrium as quickly as possible without damping.

Software Code:

- plt.scatter(w.real,w.imag): this command could plot the complex number w on a 2d picture where x is real number axis and y axis is the imaginary axis.
- plt.figure could set the picture size and plt.subplot could set the order of pictures.
- plt.hist could give a histogram in python and we can use

  ```
  fig, ax = plt.subplots()
  plt.hist(singular_min_list,bins= x)
  plt.xticks(x)
  ax.set_xscale('log', basex=2)
  ```

  to give a log scale on x axis and set the beam range by a list.

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 11: This week, major topic was still Chapter 4. We introduce many new concepts include Schur Decomposition, quasi upper triangular and so on.

References or papers consulted:

- N/A

Things I learned:

- Schur decomposition: A is a n n square matrix with complex entries, then A can be expressed as $A = QUQ^{-1}$ where Q is a unitary matrix and U is an upper triangular matrix, which is called a Schur form of A. Since U is similar to A, it has the same spectrum, and since it is triangular, its eigenvalues are the diagonal entries of U.
- Quasi-triangular matrices: Real Schur canonical form. IF A is real, there exists a real orthogonal matrix V such that $V^T AV = T$ is quasi-upper triangular. This means that T is block upper triangular with 1-by1 and 2-by-2 blocks on the diagonal. Its eigenvalues are the eigenvalues of the diagonal blocks. The 1-by-1 blocks correspond to real eigenvalues, and the 2-by-2 blocks to complex conjugate pairs.

Software Code:

- Not Applied

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 12: This week(11/5-11/11), we continue to study chapter 4. And there are some new concepts include power method, spectral mapping theorem and so on..

References or papers consulted:

- **N/A**

Things I learned:

- power method: given a diagonalizable matrix A, the algorithm will produce a number λ , which is the greatest (in absolute value) eigenvalue of A, and a nonzero vector v, the corresponding eigenvector of λ , such that $Av = \lambda v$. The algorithm is also known as the Von Mises iteration
- Spectral mapping theorem: Let A be an operator on an n-dimensional real or complex vector space V and let q(x) be any polynomial. Then the spectrum of the polynomial operator q(A) is the image of the spectrum of A under q, i.e $sp(q(A)) = q(sp(A))$
- Distinct eigenvalues: Suppose that the eigenvalues $\lambda_1, \ldots, \lambda_n$ of A are distinct. Then the corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly independent.
- Propertites of similar matrix: If two matrix are similar, then they share same Rank, Determinant, Trace, Eigenvalues (though the eigenvectors will in general be different), Characteristic polynomial, Minimal polynomial (among the other similarity invariants in the Smith normal form), Elementary divisors.
-

Software Code:

- zip function example at python:

```
name = [ "Manjeet", "Nikhil", "Shambhavi", "Astha" ]
roll_no = [ 4, 1, 3, 2 ]
marks = [ 40, 50, 60, 70 ]

# using zip() to map values
mapped = zip(name, roll_no, marks)

# converting values to print as set
mapped = set(mapped)

# printing resultant values
print ("The zipped result is : ",end="")
print (mapped)
```

Student: Leidong Xu
Journal: Fall 2018
Class: Applied Math I
Instructor: Pietro Poggi-Corradini

Week 13: This week(11/12-11/18), the major topic was still Chapter 4.

References or papers consulted:

- **DATA-DRIVEN SURROGATE MODEL TO PREDICT ISOTOPIC COMPOSITION USING DYNAMIC MODE DECOMPOSITION**

Things I learned:

- courant fischer theorem: also known as min-max theorem or variational theorem, which is a result that gives a variational characterization of eigenvalues of compact Hermitian operators on Hilbert spaces. Suppose $A \in M_n$ is Hermitian, and for each $1 \leq k \leq n$, let $\{S_k^\alpha\}_{\in I_k}$ denote the set of all kdimensional linear subspaces of $\mathbb{C}^n$. Also, enumerate the n eigenvalues $\lambda_1, \cdots, \lambda_n$ (counting multiplicity) in increasing order, i.e. $\lambda_1 \leq \cdots \leq \lambda_n$. Then, we have
- $min\ max \frac{<Ax,x>}{||x||^2} = \lambda_k$
- Weyl's theorem on complete reducibility: $|\alpha_j - \hat{\alpha_j}| \leq ||E||_2$
- I think PA6 is doable. However,I think it is important to let the students know the objective of the PA if we want to do something numerical to a verify a lemma.

Software Code:

- Python dictionary example:

```
## Can build up a dict by starting with the the empty dict {}
## and storing key/value pairs into the dict like this:
## dict[key] = value-for-that-key
dict = {}
dict['a'] = 'alpha'
dict['g'] = 'gamma'
dict['o'] = 'omega'

print dict  ## {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}

print dict['a']     ## Simple lookup, returns 'alpha'
dict['a'] = 6       ## Put new key/value into dict
'a' in dict         ## True
## print dict['z']                 ## Throws KeyError
if 'z' in dict: print dict['z']     ## Avoid KeyError
print dict.get('z')  ## None (instead of KeyError)
```