MANE 3351

LECTURE 24

Classroom Management

Agenda

- Linear Algebra using software
- Homework 7 (assigned 11/24/25, due 12/1/25 - no late submissions)
- Turn in Raspberry Pi and Arduino

RESOURCES

Handouts

- Lecture 24 slides
- Lecture 24 slides marked

Calendar

Week	Monday Lecture	Wednesday Lecture
14	**12/1:** Lecture 24 - Software	**12/3:** Lecture 25 - Review
15	**12/8:** Lecture 26 - no class	~~**12/10:**~~ Lecture 27 - no class

Final Exam is Monday 12/15/2025 8:00 - 9:45 AM

I will be off-campus on an ABET visit and a proctor will be arranged for the final exam.

Assignments

- Homework 7 (assigned 11/40, due 12/1 - no late submissions)

Solving Linear Algebra using Python/Jupyter Notebook

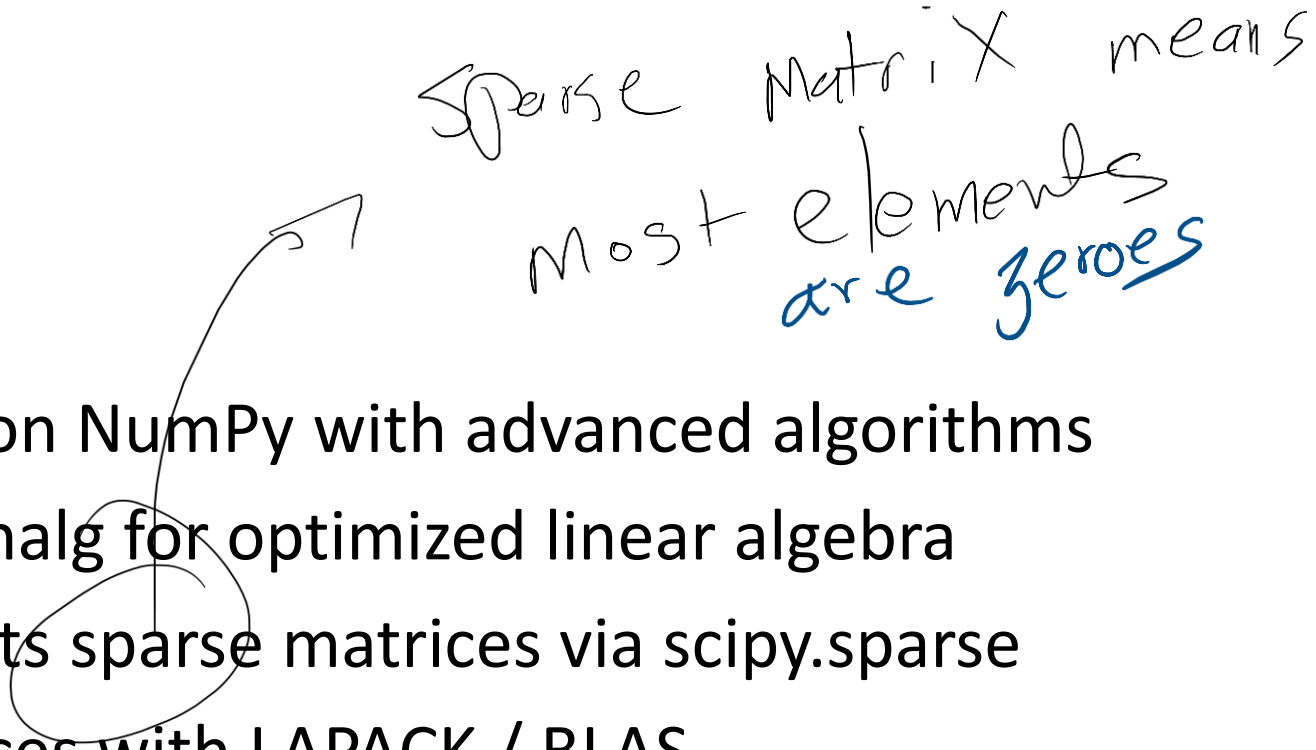
- Slides prepared by ChatGPT
- Options: NumPy, SciPy, SymPy, scikit-learn, CVXPY, PyTorch, TensorFlow, Jax, GNU Octave

NumPy

- Core package for numerical computing
- Supports vectors, matrices, broadcasting
- Linear algebra routines via numpy.linalg
- Matrix multiplication
- Eigenvalues/eigenvectors
- Determinants
- Solving linear systems

SciPy

- Builds on NumPy with advanced algorithms
- scipy.linalg for optimized linear algebra
- Supports sparse matrices via scipy.sparse
- Interfaces with LAPACK / BLAS
- Advanced solvers (iterative, sparse solvers)



SymPy

- Symbolic mathematics engine
- Exact arithmetic (non-floating point)
- Matrix algebra:
  - Inverse, rank, determinant
  - Eigenvalues symbolically
  - Jordan normal form
- Useful for teaching and derivations

scikit-learn

- Machine learning library built on NumPy/SciPy
- Uses linear algebra heavily for:
  - PCA
  - SVD
  - Least-squares models
- Efficient sparse matrix utilities

CVXPY

- Convex optimization modeling library
- Expresses problems using linear algebra primitives
- Solves:
  - LP, QP
  - QCQP, SOCP
  - SDP (semidefinite programs)
- Works with NumPy/SciPy as backend

PyTorch

- GPU-accelerated tensor computation
- Automatic differentiation
- Linear algebra routines:
  - Matrix multiplication
  - Solve linear systems
  - Decompositions: SVD, QR, Cholesky

AI → neural network

TensorFlow

- Large-scale tensor computing
- GPU/TPU support
- tf.linalg includes:
  - Determinant
  - Matrix inverse
  - SVD / QR
  - Matrix solve
- Automatic differentiation support

JAX

- NumPy-compatible high-performance computing
- JIT compilation via XLA
- Autodiff built in
- jax.numpy.linalg for fast linear algebra
- Excellent for simulation, optimization, ML research

GNU Octave

- MATLAB-compatible numerical computing environment
- Supports vectors, matrices, advanced linear algebra
- Key features:
  - Built-in solvers for linear systems (A
  - Eigenvalue/eigenvector computations
  - Matrix decompositions: LU, QR, SVD
  - Symbolic math via optional packages
- Jupyter Support:
  - Can be used through the Octave Kernel for Jupyter notebooks
  - Integrates smoothly into teaching workflows
- Excellent for users transitioning between MATLAB and Python

Example Problems (Generated by ChatGPT)

A=

$$\begin{bmatrix} 2 & 1 & -1 \\ -1 & 3 & 2 \\ 3 & -2 & 4 \end{bmatrix}$$

x=

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

b=

$$\begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

C=

$$\begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 3 & -1 \end{bmatrix}$$

Jupyter Demonstration 1 - Numpy

– Numpy Cheat Sheet

Numpy Linalg Commands

```
# Numpy
import numpy as np
A=np.array([[2,1,-1],[-1,3,2],[3,2,4]])
print(A)
D=2*A
print(D) print(np.linalg.det(A))
print(A.T)
print(A+A)
b=np.array([3,4,5])
print(np.linalg.solve(A,b))
print(np.linalg.inv(A))
B=np.array([[1,2],[0,1],[3,-1]])
print(A@B)
# print(B@A) array size mismatch
```

Jupyter Demonstration 2 - SciPy

– SciPy Cheat Sheet

SciPy linalg Commands

```
# Cell 2
from scipy import linalg
A=np.array([[2,1,-1],[-1,3,2],[3,-2,4]])
print(A)
D=2*A
print(D)
print(linalg.det(A))
print(A.T)
print(A+A)
b=np.array([3,4,5])
print(linalg.solve(A,b))
print(linalg.inv(A))
B=np.array([[1,2],[0,1],[3,-1]])
print(A@B)
# print(B@A) array size mismatch
# matrix creation commands
I4=np.array(np.eye(4))
print(I4)
J=np.array(np.ones((4,3)))
print(J)
```

Octave

GNU Octave is an open-source, MATLAB-compatible numerical computing environment used for:

- Linear algebra
- Numerical analysis
- Simulation & modeling
- Signal processing
- Optimization
- Teaching & engineering computations

Octave syntax is highly similar to MATLAB, making it ideal for students and engineers who want a free alternative.

Key Features

- Fully MATLAB-compatible language (scripts & functions)
- Powerful matrix operations and linear algebra tools
- Built-in solvers (`A\b`, `pinv`, `eig`, `svd`, `qr`, etc.)
- Supports plotting and visualization (`plot`, `surf`, etc.)
- Interactive command-line and GUI
- Works inside **Jupyter Notebooks** via the *Octave Kernel*

Installing GNU Octave & Jupyter Notebook Support (Windows)

Download & Install GNU Octave

1. Go to the official Octave download page: **https://octave.org/download**
2. Under **Windows**, download the latest **.exe installer**.
3. Run the installer:
   - Accept default settings
   - Ensure *Add Octave to system PATH* is selected (if available)
4. Launch Octave to verify installation.

Install the Octave Kernel for Jupyter

In Anaconda Prompt (or cmd):

A. Install via pip

```
pip install octave_kernel
```

B. Install Octave support (oct2py)

```
pip install oct2py
```

C. Verify kernel installation

```
python -m octave_kernel.check
```

You should see Octave executable found.

Jupyter Demonstration 3 - Octave within Jupyter Notebook

– SciPy Cheat Sheet

Matlab Commands within Jupyter Notebook

```
A=[2 1 -1;-1 3 2;3 -2 4]
b=[3;4;5]
A\b
det(A)
inv(A)
A'
B=[1 2; 0 1;3 -1]
A*B
```