Python for Linear Algebra

> ***Relevant exam portion:*** *WeBWorK (Thursday 3/5) — Python is allowed. Written (3/3) is no electronics. All examples below are drawn directly from the `python-material/` notebooks.*

Libraries: NumPy vs SymPy

Two libraries are used in this course. Know when to use each.

Library	Use for	Answers
`numpy`	Floating-point / decimal calculations	Approximate decimals
`sympy`	Symbolic / exact calculations	Exact fractions, radicals

```python
import numpy as np
import sympy as sp
from sympy import Matrix, S, Rational, linsolve, symbols, Eq
```

Example — same calculation, different outputs:

```python
# numpy: decimal
e = 2 / np.sqrt(2)
print(e)            # 1.414213562373095

# sympy: exact
f = 2 / sp.sqrt(2)
f                   # sqrt(2)
```

Example — exact fractions with sympy:

```python
c = sp.S("9/4")
d = sp.S("5/6")
print(c + d)        # 37/12  (exact)
```

> ***Rule of thumb:*** *Use sympy for everything in this course. It gives exact answers and handles fractions correctly. Only use numpy if you specifically need a decimal result.*

Creating Matrices with SymPy

```python
from sympy import Matrix

# General form: Matrix([[row1], [row2], ...])
A = Matrix([[6, 3, 0, -3],
            [3, 2, -2, 12],
            [0, 6, 8, 18]])
```

Each row is a Python list. The whole matrix is a list of rows.

Augmented matrix `[A | b]` — just include the RHS as an extra column:

```
# System: x1 + 2x2 - x3 = 6, -3x1 + 3x3 = 10
A = Matrix([[1,  2, -1,  6],
            [-3, 0,  3, 10]])
```

Finding RREF

```
A.rref()[0]   # The [0] is required — rref() returns a tuple (matrix, pivot_cols)
```

Example (from `04 - Using Python to find RREF.ipynb`):

```
A = Matrix([[6, 3, 0, -3],
            [3, 2, -2, 12],
            [0, 6, 8, 18]])
A.rref()[0]
# Matrix([
# [1, 0, 0,  -5],
# [0, 1, 0,   9],
# [0, 0, 1, -9/2]])
```

Echelon form (not fully reduced):

```
A.echelon_form()
```

> ***Watch Out:** Always use `A.rref()[0]`, not `A.rref()`. The second element `[1]` is a tuple of pivot column indices, not the matrix.*

The Decimal / Fraction Problem

Problem: When a matrix contains Python floats (e.g., `1/2` written as `0.5`), SymPy stores them as floating-point numbers, which causes rounding errors in RREF.

```
# BAD — uses Python float division, causes decimal output
B = Matrix([[6, 3, 0, -3], [3, 2, -2, 1/2], [0, 6, 8, 18]])
B.rref()[0]
# Matrix([[1, 0, 0, -2.125], [0, 1, 0, 3.25], [0, 0, 1, -0.1875]])
```

Rounding error can even give wrong pivot structure:

```
# D contains decimals — echelon_form() shows a near-zero pivot (should be 0)
D.echelon_form()
# [0, 0, -9.09e-13, ...]   ← should be exactly 0
```

Fix 1 — Use `S("...")` to parse the matrix string (fractions in input)

```
# GOOD — parses "1/2" as exact rational
B = S("""Matrix([[6, 3, 0, -3], [3, 2, -2, 1/2], [0, 6, 8, 18]])""")
B.rref()[0]
# Matrix([[1, 0, 0, -17/8], [0, 1, 0, 13/4], [0, 0, 1, -3/16]])
```

Fix 2 — Convert decimals to exact fractions with `applyfunc` + `Rational`

Use this when the matrix already contains Python floats (e.g., from a problem with decimals):

```
from sympy import Rational

D_frac = D.applyfunc(lambda x: Rational(str(x)))
D_frac.rref()[0]   # Now gives exact answer
```

Real example (from Week 4 WebWork):

```
A = Matrix([[-6.3, -9.72, 7.3, -1.1],
            [-1.6, -2.07, -0.3, 2.7],
            [-2,   0.03, -7.4, 8.7],
            [9.9,  16.41, -8.7, -7.6]])

D_frac = A.applyfunc(lambda x: Rational(str(x)))
D_frac.rref()[0]
# Matrix([
# [1, 0, 0,     3],
# [0, 1, 0, -10/3],
# [0, 0, 1,    -2],
# [0, 0, 0,     0]])
```

Getting Decimal Output from Exact Answers

```
C.rref()[0].evalf()      # Full decimal precision
C.rref()[0].evalf(3)     # Round to 3 significant figures
```

Solving Linear Systems with `linsolve`

Two input styles: equations or augmented matrix.

Style 1 — Augmented Matrix (most common for this course)

```
from sympy import linsolve, symbols, Matrix

x, y, z = symbols('x, y, z')
A = Matrix([[1, 2, -1, 6],
            [-3, 0,  3, 10]])
```

```
linsolve(A, (x, y, z))
# {(z - 10/3, 14/3, z)}   ← z is free
```

Style 2 — List of Equations

```
from sympy import linsolve, symbols, Eq

x1, x2, x3 = symbols('x1, x2, x3')
eq1 = Eq(x1 + 2*x2 - x3, 6)
eq2 = Eq(-3*x1 + 3*x3, 10)
linsolve((eq1, eq2), (x1, x2, x3))
# {(x3 - 10/3, 14/3, x3)}
```

Reading `linsolve` Output

Output	Meaning
`{(-3, 3, 2)}`	Unique solution: x=−3, y=3, z=2
`{(z - 10/3, 14/3, z)}`	Infinitely many; z is free parameter
`{(-w - 1, w - 1, -w - 2, w)}`	Infinitely many; w is free parameter
`EmptySet`	No solution (inconsistent system)

No solution example:

```
A = Matrix([[1, 2, -1, 6],
            [1, 2, -1, 5]])   # Same LHS, different RHS
linsolve(A, (x, y, z))
# EmptySet
```

Fraction Issue in Equations — Use `S("Eq(...)")`

```
# BAD — Python evaluates 3/2 as float 1.5
eq1 = Eq(x1 + (3/2)*x2 - x3, 6)
linsolve((eq1, eq2), (x1, x2, x3))
# {(1.0*x3 - 3.333..., 6.222..., 1.0*x3)}   ← decimals

# GOOD — parses 3/2 as exact rational
eq1 = S("Eq(x1 + (3/2)*x2 - x3, 6)")
linsolve((eq1, eq2), (x1, x2, x3))
# {(x3 - 10/3, 56/9, x3)}   ← exact fractions
```

Practical Patterns from the Notebooks

Check if a vector is in the null space (Week 5 — Conceptual Quiz)

```
# A = [3 0 -2 1; 4 1 0 -1]
A = Matrix([[3, 0, -2, 1],
            [4, 1,  0, -1]])

# Check if v = [-4, 0, -14, -16] is in null(A): compute A*v
v = Matrix([-4, 0, -14, -16])
A * v   # If result is [0, 0], v is in null(A)
```

Or find the RREF to identify all null space vectors:

```
D_frac = A.applyfunc(lambda x: Rational(str(x)))
D_frac.rref()[0]
```

Find bases for col(A), row(A), null(A) (Week 5 WebWork)

```
A = Matrix([[1, 4, -1, 1],
            [3, 14, -1, 6],
            [2, 12,  2, 8]])
rref, pivots = A.rref()
print("Pivot columns:", pivots)   # e.g., (0, 1) → columns 0 and 1 are pivot cols
# col(A) basis: columns 0 and 1 of ORIGINAL A (not rref)
# row(A) basis: nonzero rows of rref
# null(A) basis: solve linsolve with homogeneous system
```

Solve a system from WebWork (Week 2 WebWork)

```
# Problem 8: x + 2y + 3z = 9, -3x - 2y + 4z = 11, -6x + 2y + 3z = 30
x, y, z = symbols('x, y, z')
A = Matrix([[1,  2, 3,  9],
            [-3, -2, 4, 11],
            [-6,  2, 3, 30]])
linsolve(A, (x, y, z))
# {(-3, 3, 2)}
```

Solve a 4-variable system (Week 2 WebWork Problem 9)

```
x, y, z, w = symbols('x, y, z, w')
A = Matrix([[0,  0, -1, 1, -2],
            [1, -1,  1, 1,  0],
            [0,  1,  2, 1,  0],
            [-1, 2, -3, 5, -9]])
linsolve(A, (x, y, z, w))
# {(-1, -1, 1, -1)}
```

Quick Reference

```python
# Setup
from sympy import Matrix, S, Rational, linsolve, symbols, Eq
import sympy as sp

# Create matrix
A = Matrix([[1, 2, 3], [4, 5, 6]])

# RREF (always [0])
A.rref()[0]

# Echelon form
A.echelon_form()

# Fix decimal rounding errors
A_exact = A.applyfunc(lambda x: Rational(str(x)))
A_exact.rref()[0]

# Solve system (augmented matrix)
x, y, z = symbols('x, y, z')
linsolve(A, (x, y, z))

# Solve system (equations)
linsolve((Eq(x+y, 3), Eq(x-y, 1)), (x, y))

# Exact fractions in equations
eq = S("Eq(x + (3/2)*y, 5)")

# Decimals from exact answer
A.rref()[0].evalf()     # full precision
A.rref()[0].evalf(3)    # 3 significant figures
```