Extending Python For Numerical Computation

The Python Matrix Object

(based on Jim Fulton's original implementation of a matrix object as a container type)

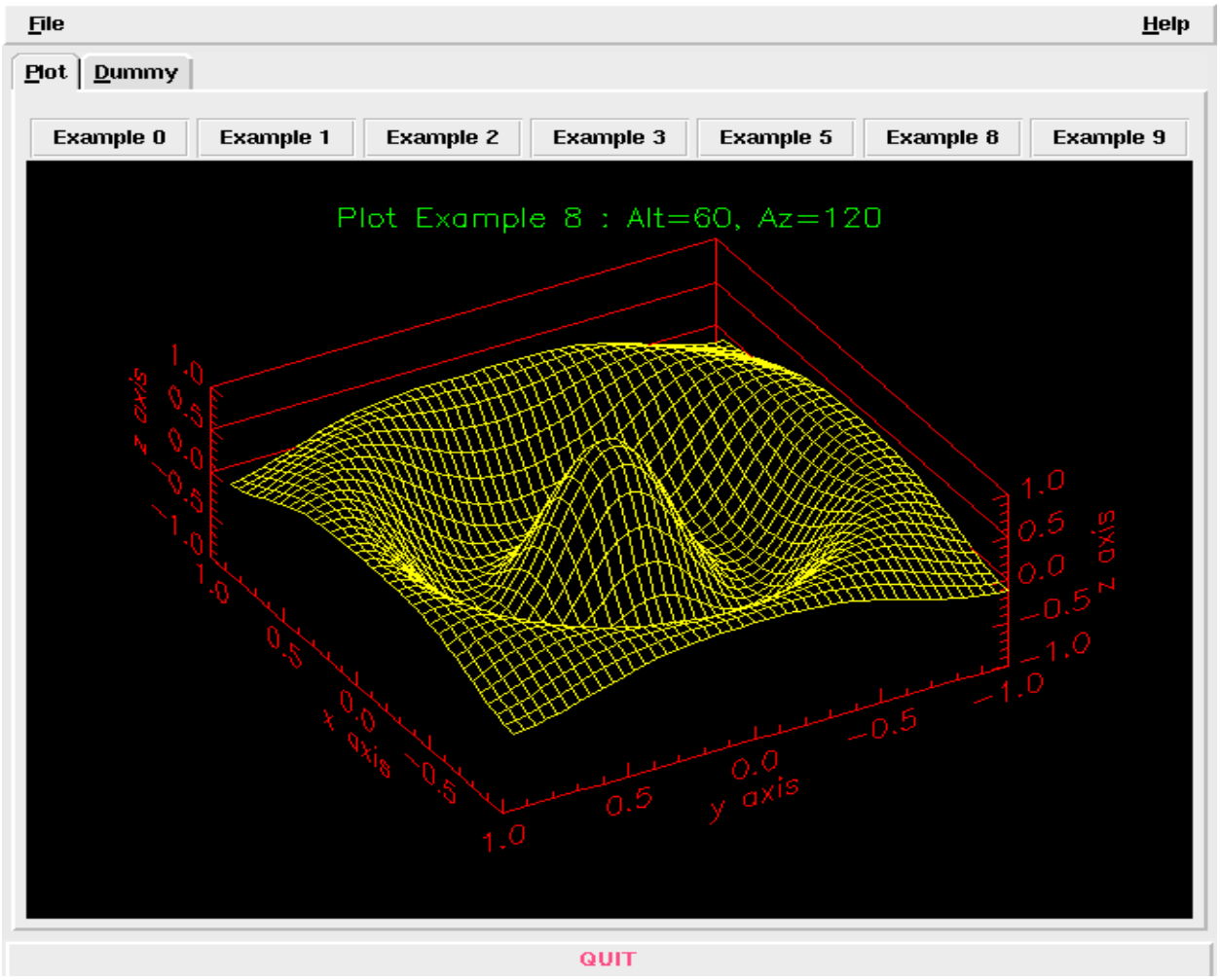
Jim Hugunin

December 6, 1995

Massachusetts Institute of Technology
Laboratory for Computer Science

Current Status

- It works!



m, n = 45, 40
x,y=Matrix.mesh((-1, 1, 2.0/m), (-1, 1, 2.0/n))
r=sqrt(x**2+y**2)
z=exp(-r**2)*cos(2*pi*r)

Yet Another Numeric Language?

“The eval function now provides an optional mechanism for detecting and trapping error conditions that occur during the evaluation of the argument expression.”

- Matlab 4.1 Release Notes

“Octave has a real mechanism for handling functions that take an unspecified number of arguments, so it is no longer necessary to place an upper bound on the number of optional arguments that a function can accept.”

- Octave FAQ

“However, the most significant improvement is RLaB's list class. A list is a heterogeneous associative array that can contain any data type, including other lists. The list gives RLaB users the opportunity to structure their data as necessary.”

- Why Use RLaB?

“S-PLUS now supports a number of features from the object-oriented programming paradigm, including classes, inheritance, and methods.”

- S-Plus 3.0 Release Notes

Death to little languages

Olin Shivers - Scheme Shell

- Learning curve

  - Instead of 5 or 6 special purpose languages, just learn Python

- Power

  - The full power of Python is always available

- Flexibility

  - Can run numerical code in my web browser, and vice versa

What is a Numeric Language?

- Multi-dimensional arrays of numbers

  - ints, floats, doubles, complex, ...

- Efficient simple numeric operations

  - +, *, <, sin, ...

- More complicated array operations

  - sum, outer product, ...

- Powerful indexing

  - slices, transposes, arbitrary index lists, ...

- Large numeric library

  - fft, matrix inversion, ...

- Visualization Tools

  - line plots, surface plots, contour plots, ...

Multi-dimensional arrays

- Arbitrary number of dimensions

  - vs. only allowing 2d matrices

- All basic C types plus complex floats and doubles

  - vs. only complex doubles

- Arrays of Python Objects as well

  - Curious feature

- Easily extensible with new types

  - vs. not extensible at all

  - 3 hours to add complex numbers

Efficient numeric operations

- All basic operators
- Trigonometric functions
- Basic operations on large arrays are fast (but could be faster)
- Squaring a 1000 element array

  - on a Sparc10

Language	MFLOPS w/ doubles	MFLOPS w/ floats
Python w/ for loop	0.01	X
Python w/ matrix of PyObjects	0.1	X
Octave 1.2	0.2	X
gcc 2.7.0 w/ -g	1.0	1.0
Matlab (tm) 4.2	1.5	X
Python w/matrix of raw numbers	1.5	2.1
Python w/raw matrix and tricks	2.6	3.2
gcc 2.6.3 w/ -O4	2.6	3.8
gcc 2.7.0 w/ -O4	6.8	6.8

More complicated operations

- General purpose reduce
  - Essentially equivalent to the built-in python function “reduce”, but very efficient for basic operations on matrices
  - vs. special purpose sum and product functions
- Accumulate as well
  - vs. cumsum and cumprod
- Yorick style pseudo indices
  - Any dimension of size 1 is broadcast
  - Easily create a dimension of size 1 by using None as an index
  - Can express outer products, plus much more

Powerful indexing

- Can get at any arbitrary subarray
- Indexing returns a new array by reference
- Strides are used to represent arbitrary transposes, and slices with steps
- take method will select an arbitrary list of indices

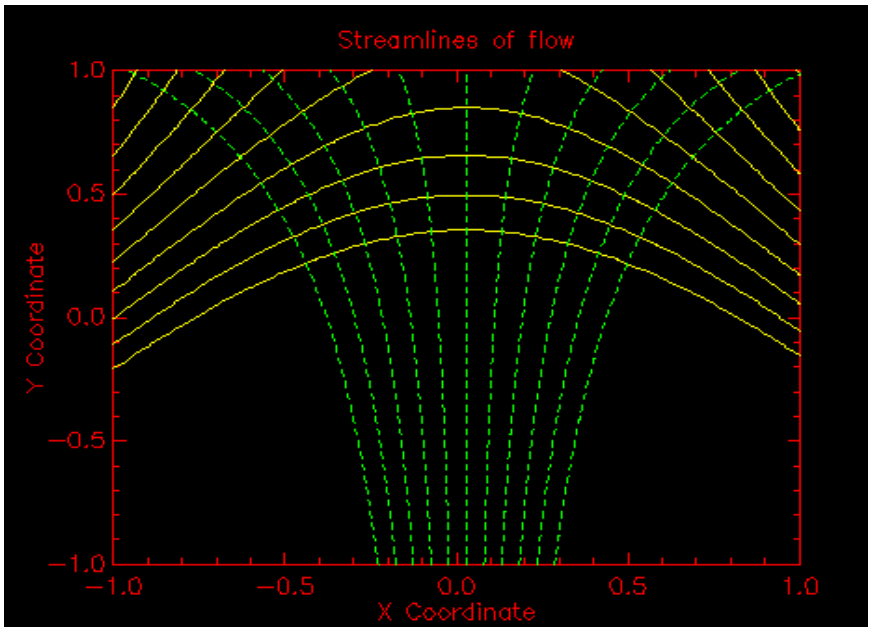
Large numeric library

- Basic structure is equivalent to both C and FORTRAN style arrays
- Can interface easily to existing numerical libraries
- LAPACK, FFTPACK, ...

Visualization Tools

- Tom Schwaller's plmodule

  - interface to plplot library (seen in opening slide)



- Tom's also working on an OpenGL module

- There are great possibilities here!

Why Python?

- New object types in C

  - sine qua non

- Python is a great language

- Wouldn't it be nice to be able to write a fluid-dynamics package in python that could calculate the average wind-speed velocity of a European Swallow?

Major Open Questions

- Matrix Multiplication
  - A LinearAlgebraMatrix class?
  - Steal % from modulo?
  - Use current a.matrixMultiply(b) syntax?
- Numeric Exceptions/Overflows
  - This is machine specific to do efficiently
  - Some people want IEEE values instead of exceptions
- Boolean operations
  - Why can I implement an __add__ method, but not a __lessThan__ method?
  - This is the only frustration of mine regarding the python core.
- By Reference vs. By Copy

Matrix SIG Accomplishments

- Created working matrix object v0.20
  - Close to the result of consensus from the list (except for a few open questions)
- Konrad Hinsen's complex numbers
- Two modules based on prototype matrix object are already available
  - Paul DuBois' URNG (Uniform Random Number Generator)
  - Tom Schwaller's plplot (interface to a good quality plotting library)
- Several small patches to the python grammar are being tested
  - a**b == pow(a,b) -with Guido's blessing
  - a[1,2,3] == a[(1,2,3)] - with Guido's blessing
  - 1+2j == complex(1,2) - not blessed yet
  - a[1:3, :4, 6] == a[slice(1,3), slice(None,4), 6] - not blessed yet