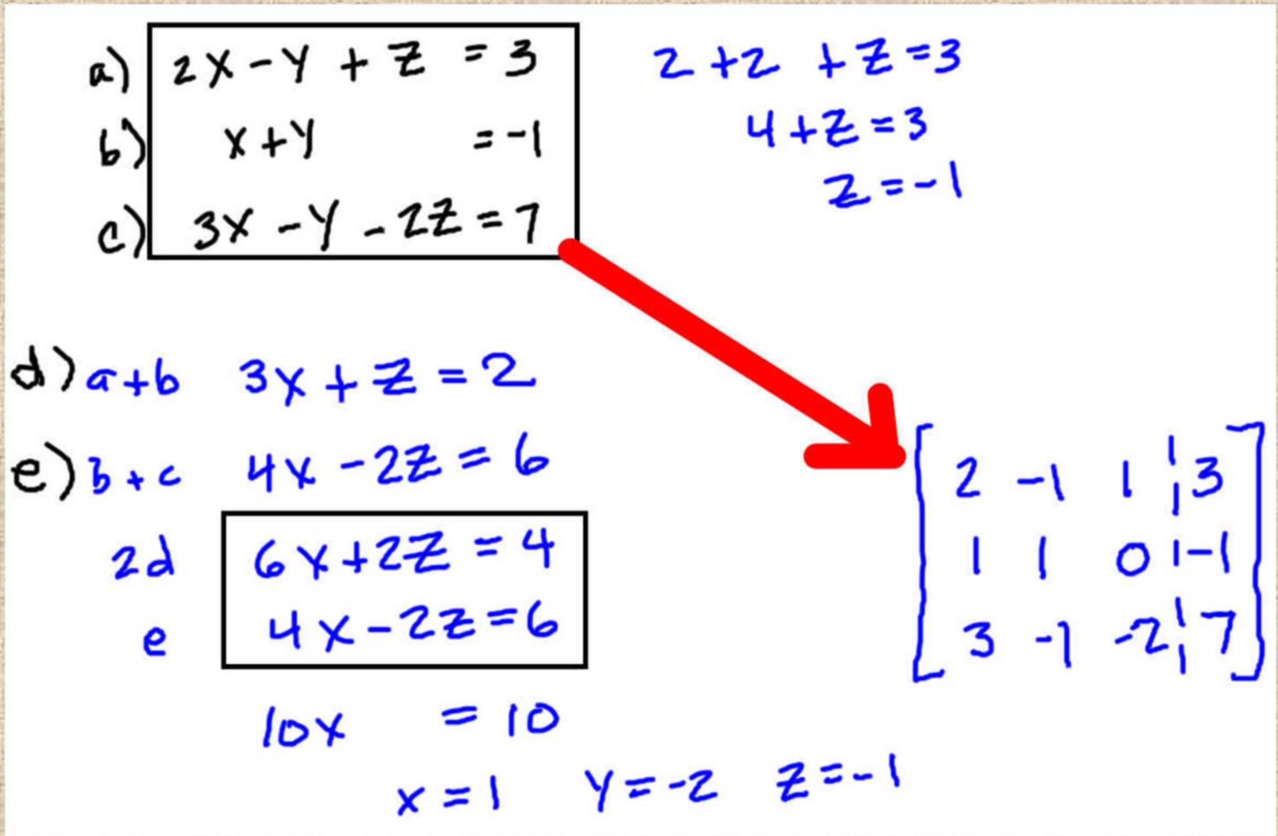
LAME
(Linear Algebra Made Easy)

David Golub
Carmine Elvezio
Muhammad Akbar
Ariel Deitcher

MOTIVATION

Performing calculations and operations by hand is tedious, time consuming and error prone.



However, computers love this stuff!

Why LAME isn't (lame)

- LAME allows for basic control flow operations (if, while)
- Performs matrix/vector operations (resizing, transpose, multiplication, exponentiation) so you don't have to
- Imperative language with C and MATLAB like syntax

The Basics

Basic Types

- Boolean
- String
- Scalar
  - 64 bit signed double precision float
- Matrix
  - Dynamically sized 2-D array
  - Elements can be scalars or other matrices

Basic Operations

- Addition
- Negation
- Multiplication
- Division
- Exponentiation

Basic Operators

- `print`
- `if`
- `while`
- `dim`
- Relational Operators
- Logical Operators
- Transpose

Example time!

Let us implement an algorithm using LAME to solve a system of simultaneous linear equations. The equations are:

$$3x_1 + x_2 = 3$$

$$9x_1 + 4x_2 = 6$$

We use the following algorithm to solve this problem.

$$A = \begin{bmatrix} 3 & 1 \\ 9 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = A^{-1}B$$

$$A^{-1} = \frac{1}{|A|}\mathrm{Adj}(A) = \frac{1}{A_{0,0}A_{1,1} - A_{0,1}A_{1,0}} \begin{bmatrix} A_{1,1} & -A_{0,1} \\ -A_{1,0} & A_{0,0} \end{bmatrix}$$

Doing This (fingers crossed)

```
matrix A = { 3, 1; 9, 4 };
matrix B = { 3; 6 };
matrix X;

print "\nSolving system of simultaneous linear
equations:\n";
print A[0,0] + " x1 + " + A[0,1] + " x2 = " + B[0] +
"\n";
print A[1,0] + " x1 + " + A[1,1] + " x2 = " + B[1] +
"\n";
print "\nA = \n" + A + "\n";
print "\nB = \n" + B + "\n";

scalar det_of_A = A[0,0]*A[1,1] - A[0,1]*A[1,0];
print "\nDeterminant(A) = " + det_of_A + "\n";
```

Outputs This

Solving system of simultaneous linear
equations:

3 x1 + 1 x2 = 3

9 x1 + 4 x2 = 6

A =

3 1

9 4

B =

3

6

Determinant(A) = 3

Doing This

```
if(det_of_A != 0) {
…  //see next slide
} else {
print "A is singular and its inverse doesn't
exist.\n";
}
```

Doing This (continued)

```
matrix inv_of_A;

inv_of_A [0,0] = A[1,1];
inv_of_A [0,1] = -1*A[0,1];
inv_of_A [1,0] = -1*A[1,0];
inv_of_A [1,1] = A[0,0];
inv_of_A = inv_of_A / det_of_A;

X = inv_of_A * B;

print "\nInverse(A) = \n" + inv_of_A + "\n";
print "X = Inverse(A) * B = \n" + X + "\n";
print "Solution:\n";
print "x1 = " + X[0] + "\n";
print "x2 = " + X[1] + "\n";
```

Outputs This (Yay!)

```
Inverse(A) =
1.33333 -0.333333
-3 1

X = Inverse(A) * B =
2
-3

Solution:
x1 = 2
x2 = -3
4
```

The Whole Shebang

```
matrix A = { 3, 1; 9, 4 };
matrix B = { 3; 6 };
matrix X;
print "\nSolving system of simultaneous linear equations:\n";
print A[0,0] + " x1 + " + A[0,1] + " x2 = " + B[0] + "\n";
print A[1,0] + " x1 + " + A[1,1] + " x2 = " + B[1] + "\n";
print "\nA = \n" + A + "\n";
print "\nB = \n" + B + "\n";
scalar det_of_A = A[0,0]*A[1,1] - A[0,1]*A[1,0];
print "\nDeterminant(A) = " + det_of_A + "\n";
if(det_of_A != 0) {
matrix inv_of_A;
inv_of_A [0,0] = A[1,1];
inv_of_A [0,1] = -1*A[0,1];
inv_of_A [1,0] = -1*A[1,0];
inv_of_A [1,1] = A[0,0];
inv_of_A = inv_of_A / det_of_A;
X = inv_of_A * B;
print "\nInverse(A) = \n" + inv_of_A + "\n";
print "X = Inverse(A) * B = \n" + X + "\n";
print "Solution:\n";
print "x1 = " + X[0] + "\n";
print "x2 = " + X[1] + "\n";
} else {
print "A is singular and its inverse doesn't exist.\n";
}
```

All Together Now

Solving system of simultaneous linear equations:

3 x1 + 1 x2 = 3

9 x1 + 4 x2 = 6

A =

3 1

9 4

B =

3

6

Determinant(A) = 3

Inverse(A) =

1.33333 -0.333333

-3 1

X = Inverse(A) * B =

2

-3

Solution:

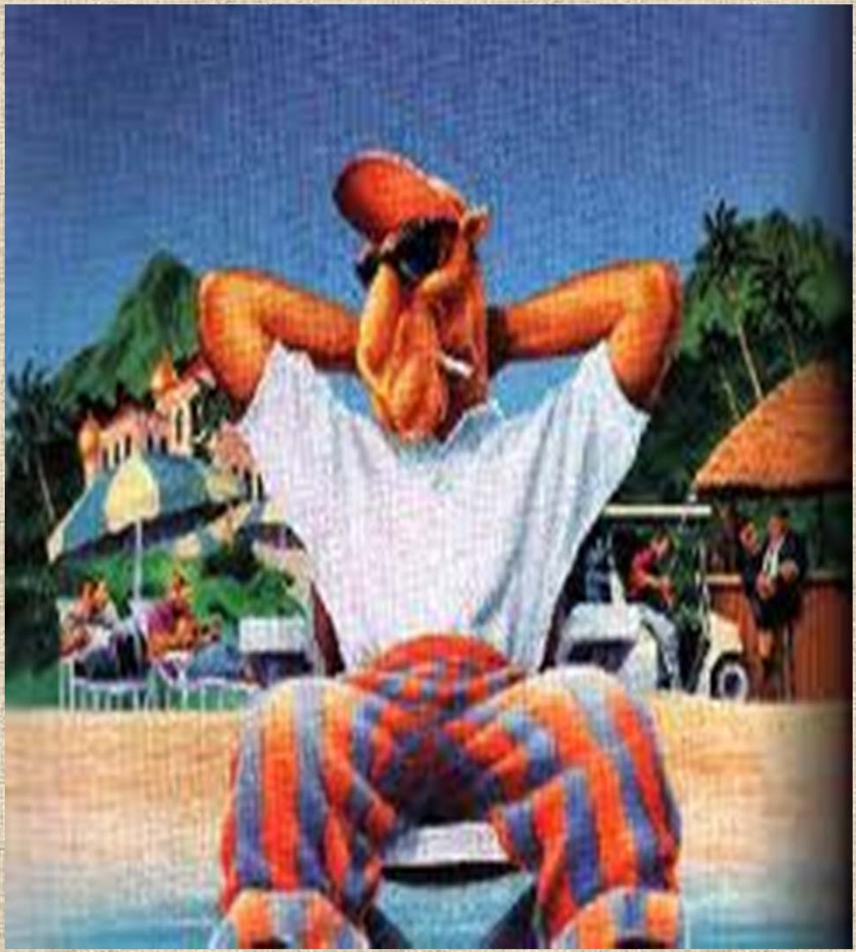
x1 = 2

x2 = -3

4

Implementation

- O'Caml takes care of the hard work
- iLAME 3-op code is converted into C ++
- C++ is then used to perform matrix operations and output the result



Lessons Learned

- Get started early!
- Make sure the problem is well defined – math really is your friend
- Set and keep to deadlines
- Be open to revision
- Formal interfaces - document/code style, module interaction
- AWK is better than O'Caml (kidding)

Questions?