Core Loops in Native Code
Octave

Tyson Whitehead

SHARCNET
The University of Western Ontario

June 5, 2009

Introduction

Interpreted languages such as Octave are only efficient if the majority of a programs time is spent inside built-in functions.

- This requires the core loop of the program to map well onto matrix primitives (e.g., solving a system of equations).
- Performance will be a problem if the core loop has to be iterate through individual matrix/vector entries.
- Efficiency can be recovered in these cases by rewriting the core loop in C/C++ and calling it from Octave.

Octave Manual (Appendix A: Dynamically Linked Functions)

`http://www.gnu.org/software/octave/doc/interpreter`

Compiling for Octave

Compiling (Appendix A.1.1)

- Give the cpp file the same base name as the function.
- Compile the cpp file with the mkoctfile command (a wrapper around the C++ compiler).
- C++ compiler flags will be passed through.

Example

```
$ mkoctfile helloworld_cpp.cpp
$
```

Octave Types Overview (1/2)

Octave uses to C++ type system to present various interfaces to a chunk of underlying memory. The classes derived from the underlying Array type to provide specific interpretations include

- Cell — array of octave_value pointers
- ∗RowVector/∗ColVector — row/column vector
- ∗Matrix — matrix
- ∗NDArray — a multidimensional array

where * is {bool,ch,int,f,,fC,C,{int,uint}{8,16,32,64}} (not all of these are supported by all variants).

Special Types/Representations

The dim_vector, ∗DiagMatrix, ∗Sparse, and PermMatrix types have their own special representation for efficiency. Most of this is opaque unless the underlying memory is accessed directly.

Octave Types Overview (2/2)

Type are passed around wrapped in an introspective hierarchy accessed through octave_value/octave_value_list (the former being a list of the latter). Features include

- is_∗ — various type test
- ∗_value — convert to specific type
- save_∗/load_∗ — save/load to/from various formats

Various other special classes existing for encapsulating/interfacing with various algorithms/operations (e.g., decomposition).

Call Throughs

Wrappers for most of the general functions on the specific types are provided so they can also be called on octave_value.

Array Types

The basic array types include ∗RowVector/∗ColVector, ∗Matrix, and ∗NDArray. They are mostly identical apart from the availability of specific math operations (e.g., matrix inversion).

- Underlying memory is in column-major order (i.e., the left most index address adjacent values as in Fortran), and is accessed via the data or fortran_vec functions.
- Dimensioning is a dim_vector and can be manipulated/queried by various functions including ndims, dims, and resize.
- Elements accessed via () (either of the next two depending on whether BOUNDS_CHECKING is defined), checkelem (check indices and unique), elem (check unique), or xelem.
- Most math functions and operators overloaded to work; various is_∗, any_∗, and all_∗ tests are available; and map and fast_map can be used.

Gotchas

- Use OCTAVE_QUIT to check for CTRL+C.
- Octave indexes from one, C++ indexes from zero.
- Octave arrays are in column-major order, C++ arrays are row-major-order.