

Numlua: a numerical package for Lua

Luis Carvalho
Brown University
carvalho@dam.brown.edu

Lua Workshop
July, 2008

Introduction

- A lot of problems in scientific computing need high performance numerical routines

Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)

Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)
- Promising candidate: *Lua!*

Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)
- Promising candidate: *Lua!*
 - *Performance*: fast execution and small footprint



Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)
- Promising candidate: *Lua!*
 - *Performance*: fast execution and small footprint
 - *Powerful*: GC, functional facilities, coroutines, metamethods

Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)
- Promising candidate: *Lua!*
 - *Performance*: fast execution and small footprint
 - *Powerful*: GC, functional facilities, coroutines, metamethods
 - *Extensible*: simple and well defined API in ANSI C

Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)
- Promising candidate: *Lua!*
 - *Performance*: fast execution and small footprint
 - *Powerful*: GC, functional facilities, coroutines, metamethods
 - *Extensible*: simple and well defined API in ANSI C
 - *Easy*: semantically rich but simple

Introduction

- A lot of problems in scientific computing need high performance numerical routines
- General approach: heavy computations in low level routines (FORTRAN or C) wrapped in a *scripting* or OO language (high level)
- Promising candidate: *Lua!*
 - *Performance*: fast execution and small footprint
 - *Powerful*: GC, functional facilities, coroutines, metamethods
 - *Extensible*: simple and well defined API in ANSI C
 - *Easy*: semantically rich but simple
 - *Free*: MIT license

- Based on “classical” numerical packages



Numeric Lua

- Based on “classical” numerical packages
- Motivation: keep minimalist approach from Lua in order to provide a basis for more elaborated scientific packages

Numeric Lua

- Based on “classical” numerical packages
- Motivation: keep minimalist approach from Lua in order to provide a basis for more elaborated scientific packages
- Components
 - Complex numbers
 - Special functions (Exps + CDFs + PDFs)
 - Random number generation (MT + Ranlib)
 - Numerical linear algebra (BLAS + LAPACK)

Numlua – Features

- Some OO and functional flavor, but no paradigm compromise

Numlua – Features

- Some OO and functional flavor, but no paradigm compromise
- Matrix type system: general, triangular, symmetric, and posdef

Numlua – Features

- Some OO and functional flavor, but no paradigm compromise
- Matrix type system: general, triangular, symmetric, and posdef
- Small: interpreter + libs \approx 1.25Mb (static link) or 200Kb (dynamic link)



Numlua – Features

- Some OO and functional flavor, but no paradigm compromise
- Matrix type system: general, triangular, symmetric, and posdef
- Small: interpreter + libs \approx 1.25Mb (static link) or 200Kb (dynamic link)
- Implemented in ANSI C and thus as portable as Lua

Numlua – Features

- Some OO and functional flavor, but no paradigm compromise
- Matrix type system: general, triangular, symmetric, and posdef
- Small: interpreter + libs \approx 1.25Mb (static link) or 200Kb (dynamic link)
- Implemented in ANSI C and thus as portable as Lua
- Standard interface to BLAS/LAPACK: can be linked to many optimized libs

Numlua – Features

- Some OO and functional flavor, but no paradigm compromise
- Matrix type system: general, triangular, symmetric, and posdef
- Small: interpreter + libs \approx 1.25Mb (static link) or 200Kb (dynamic link)
- Implemented in ANSI C and thus as portable as Lua
- Standard interface to BLAS/LAPACK: can be linked to many optimized libs
- MIT license

Numlua – Examples

```
function circ1(v)
  local n = table.getn(v)
  local m = matrix.zeros(n)
  for i = 1, n do
    for j = 1, n do
      m[i][j] = v[(j - i) % n + 1]
    end
  end
  return m
end
```

Numlua – Examples

```
function circ1(v)
    local n = table.getn(v)
    local m = matrix.zeros(n)
    for i = 1, n do
        for j = 1, n do
            m[i][j] = v[(j - i) % n + 1]
        end
    end
    return m
end

function circ2(v)
    local n = table.getn(v)
    return matrix.zeros(n):apply(
        function(i, j) return v[(j - i) % n + 1] end)
end
```

Numlua – Examples

```
local r = rng(os.clock())
function mvnorm(mean, var, n)
    local u, info = matrix.chol(var) -- u' * u = var
    assert(info == 0, "matrix is not positive definite")
    -- y:col(i) ~ N(0_m, I_m)
    local m = mean:size()
    local y = matrix(m, n):map(
        function(e) return r:norm(0, 1) end)
    -- u' * y + mean ~ N(mean, var)
    y = #u * y
    for i, j, e in y:entries() do
        y[i][j] = e + mean[i]
    end
    return y
end
```

Future work

- Documentation!

Future work

- Documentation!
- Move to Luarocks

Future work

- Documentation!
- Move to Luarocks
- New internal representation for matrices

Future work

- Documentation!
- Move to Luarocks
- New internal representation for matrices
- Use token filters to make matrix declaration easier (less painful)



Future work

- Documentation!
- Move to Luarocks
- New internal representation for matrices
- Use token filters to make matrix declaration easier (less painful)
- Extensions: HDF5, sparse matrices, stat library, plotting library, ...

Thank you!

www.mccours.com

Site N°1 des Cours et Exercices Email: contact@mccours.com

