A short course on object-oriented numerics

Sylwester Arabas
Faculty of Physics, University of Warsaw

local organisers:
Juan A. Añel, **Orlando García**

EPhysLab, Universidade de Vigo, Ourense

day 3 (June 5th 2014)

plan for today: yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

Example 6: Boost.MPI basics

Boost.MPI

http://boost.org/libs/mpi/

```
 1 #include <boost/mpi.hpp>
 2 #include <iostream>
 3
 4 int main()
 5 {
 6   boost::mpi::environment env;
 7   boost::mpi::communicator com;
 8
 9   std::ostringstream tmp;
10   tmp
11     << com.rank() << "/"
12     << com.size() << std::endl;
13   std::cerr << tmp.str();
14 };
```

```
$ sudo apt-get install \
    libboost-mpi-dev
$ mpic++ -lboost_mpi \
    -std=c++11 ex5.cpp
$ ./a.out
0/1
$ mpirun -np 2 ./a.out
0/2
1/2
$ mpirun -np 4 ./a.out
3/4
0/4
1/4
2/4
```

Example 7: Boost.MPI/Blitz++ “hello world”

```
 1 #include <boost/mpi.hpp>
 2 #define BZ_HAVE_BOOST_SERIALIZATION
 3 #include <blitz/array.h>
 4
 5 template <class arr_t>
 6 void output(
 7   const std::string &str,
 8   const int &rank,
 9   const arr_t &arr
10 )
11 {
12   std::ostringstream tmp;
13   tmp << str << " @" << rank  << ":" << arr << std::endl;
14   std::cerr << tmp.str();
15 }
16
17 int main()
18 {
19   boost::mpi::environment env;
20   boost::mpi::communicator com;
21
```

Example 7: Boost.MPI/Blitz++ “hello world”

```
22    // each process fills an array with its rank number
23    blitz::Array<int, 1> arr(4);
24    arr = com.rank();
25
26    output("initial data", com.rank(), arr);
27
28    // wait for all nodes to get here
29    com.barrier();
30
31    // locating neighbours
32    int from_left_nghbr = (com.rank() - 1 + com.size()) % com.size(),
33          to_rght_nghbr = (com.rank() + 1 + com.size()) % com.size();
34
35    // sending/receiving a Blitz++ array with MPI non-blocking comm.
36    std::list<boost::mpi::request> reqs;
37    enum { msgid };
38    reqs.push_back(com.isend(to_rght_nghbr,   msgid, arr));
39    reqs.push_back(com.irecv(from_left_nghbr, msgid, arr));
40    boost::mpi::wait_all(reqs.begin(), reqs.end());
41
42    output("received data", com.rank(), arr);
43 }
```

Example 7: Boost.MPI/Blitz++ “hello world”

```
$ mpic++ -lboost_mpi -lboost_serialization -std=c++11 ex6.cpp

$ mpirun -np 3 ./a.out
initial data @2:(0,3)
[ 2 2 2 2 ]

initial data @0:(0,3)
[ 0 0 0 0 ]

initial data @1:(0,3)
[ 1 1 1 1 ]

received data @2:(0,3)
[ 1 1 1 1 ]

received data @1:(0,3)
[ 0 0 0 0 ]

received data @0:(0,3)
[ 2 2 2 2 ]
```

Example 9: gnuplot-iostream with Blitz++

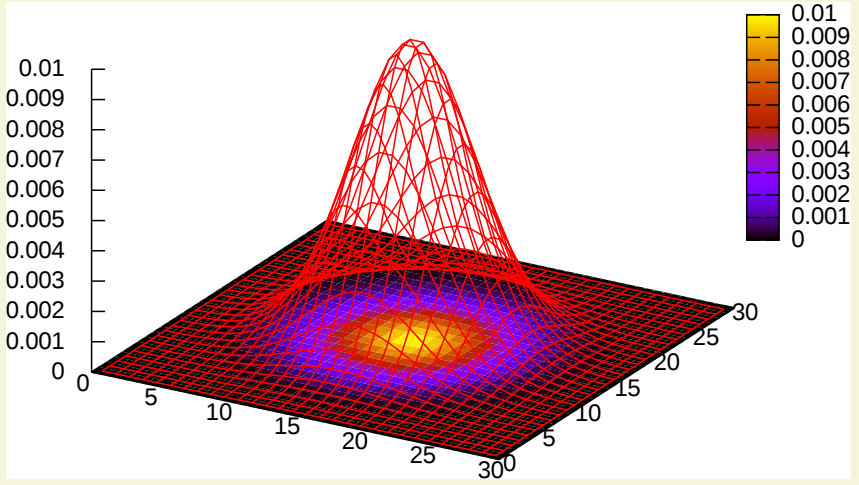
```cpp
#include <blitz/array.h>
#include <gnuplot-iostream.h>
#include <boost/math/constants/constants.hpp>

int main()
{
  int n = 30;
  double sgma = 4;
  blitz::Array<double, 2> arr(n,n);

  {
    using namespace blitz::tensor;
    using boost::math::constants::pi;
    arr = pow(
      2 * pi<double>() * pow(sgma, 2), -1
    ) * exp(-(
      pow(i+.5 - n/2., 2) +
      pow(j+.5 - n/2., 2)
    ) / 2 / pow(sgma, 2));
  }

  Gnuplot gp;
  gp
    << "set term svg dynamic"       << "\n"
    << "set xrange [0:" << n << "]" << "\n"
    << "set yrange [0:" << n << "]" << "\n";
```

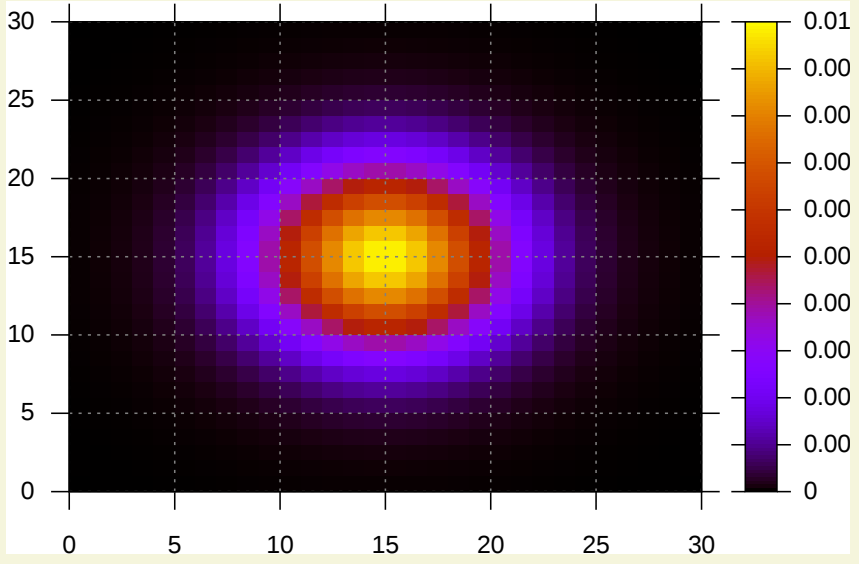
$$\frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}$$

Example 9: gnuplot-iostream with Blitz++

```
27
28    gp
29      << "set output 'plot_lines.svg'"<< "\n"
30      << "set grid"                   << "\n"
31      << "set xyplane at 0"           << "\n"
32      << "splot"
33      << " '-' binary " << gp.binfmt(arr)
34      << "   origin=(.5,.5,0)"
35      << "   with image failsafe notitle"
36      << ",'-' binary " << gp.binfmt(arr)
37      << "   origin=(.5,.5,0)"
38      << "   with lines notitle"      << "\n";
39    gp.sendBinary(arr);
40    gp.sendBinary(arr);
41
42    gp
43      << "set output 'plot_image.svg'"<< "\n"
44      << "set view map"               << "\n"
45      << "set tics out"               << "\n"
46      << "splot"
47      << " '-' binary " << gp.binfmt(arr)
48      << "   origin=(.5,.5,0)"
49      << "   with image failsafe not" << "\n";
50    gp.sendBinary(arr);
51  }
```

```
$ clang++ -lboost_iostreams -lboost_system \
    -lblitz -std=c++11 ex7.cpp
$ ./a.out
```





yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)

- libraries usable via the Blitz++ C-pointer API

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)

- libraries usable via the Blitz++ C-pointer API
  - **netCDF-cxx4**[1]

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)

- libraries usable via the Blitz++ C-pointer API
  - **netCDF-cxx4**[1]
  - **HDF5 C++ API**

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)
- libraries usable via the Blitz++ C-pointer API
  - **netCDF-cxx4**[1]
  - **HDF5 C++ API**
- other potentially useful libraries:

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)
- libraries usable via the Blitz++ C-pointer API
  - **netCDF-cxx4**[1]
  - **HDF5 C++ API**
- other potentially useful libraries:
  - **Boost.odeint** – ordinary diff. eq. solvers (supports Boost.units out of the box)

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)
- libraries usable via the Blitz++ C-pointer API
  - **netCDF-cxx4**[1]
  - **HDF5 C++ API**
- other potentially useful libraries:
  - **Boost.odeint** – ordinary diff. eq. solvers (supports Boost.units out of the box)
  - **Boost.Python** – two-way C++/Python bridge

[1] http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/

[2] http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo

yet more libraries

- libraries ready-to-use with Blitz++ OOP API
  - **Boost.MPI** & **Boost.Serialization**
  - **gnuplot-iostream** (also supports Armadillo)
- libraries usable via the Blitz++ C-pointer API
  - **netCDF-cxx4**[1]
  - **HDF5 C++ API**
- other potentially useful libraries:
  - **Boost.odeint** – ordinary diff. eq. solvers (supports Boost.units out of the box)
  - **Boost.Python** – two-way C++/Python bridge
  - **Thrust** – nVidia's free/libre & open-source STL-like library for CUDA, OpenMP (reportedly[2] incl. Xeon Phi/MIC!), Intel TBB as well as serial execution

[1] `http://www.unidata.ucar.edu/software/netcdf/docs/cxx4/`

[2] `http://groups.google.com/forum/#!topic/thrust-users/dOEy0qhslIo`

Summary / take-home messages

- how OOP fits geoscientific model development:
  - separation of concerns
  - blackboard abstractions

Summary / take-home messages

- how OOP fits geoscientific model development:
  - separation of concerns
  - blackboard abstractions

- potential benefits:
  - libraries $\rightsquigarrow$ not reinventing the wheel
  - succinct OOP syntax $\rightsquigarrow$ readability
  - OOP modularity $\rightsquigarrow$ maintainability

Summary / take-home messages

- how OOP fits geoscientific model development:
  - separation of concerns
  - blackboard abstractions
- potential benefits:
  - libraries $\rightsquigarrow$ not reinventing the wheel
  - succinct OOP syntax $\rightsquigarrow$ readability
  - OOP modularity $\rightsquigarrow$ maintainability
- C++ advantages:
  - rich set of mature OOP libraries (unlike Fortran)
  - no inherent performance limitations (unlike Python)

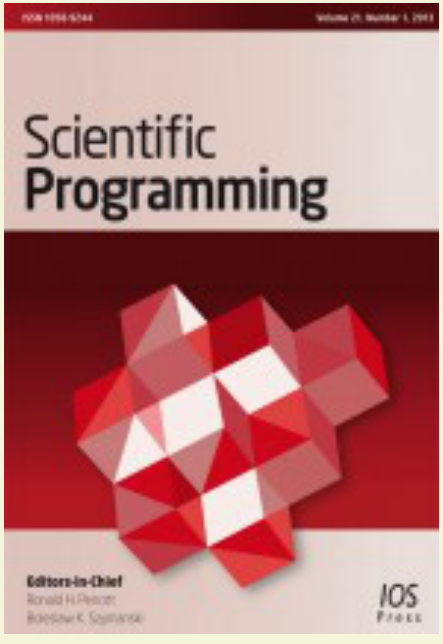
Summary / take-home messages

- how OOP fits geoscientific model development:
  - separation of concerns
  - blackboard abstractions
- potential benefits:
  - libraries $\rightsquigarrow$ not reinventing the wheel
  - succinct OOP syntax $\rightsquigarrow$ readability
  - OOP modularity $\rightsquigarrow$ maintainability
- C++ advantages:
  - rich set of mature OOP libraries (unlike Fortran)
  - no inherent performance limitations (unlike Python)
- C++ drawbacks:
  - lengthy unintelligible error messages
  - numerous Python-like features only since recently (C++11)

That's it for today.
Thanks for your attention!

That's it for today.
Thanks for your attention!

Blitz++ / NumPy / NumPyPy / Fortran comparison study



Formula translation in Blitz++, NumPy and modern Fortran: A case study of the language choice tradeoffs

Sylwester Arabas[1], Dorota Jarecka[1], Anna Jaruga[1], Maciej Fijałkowski[2]

[1]Institute of Geophysics, Faculty of Physics, University of Warsaw
[2]PyPy Team

Journal	Scientific Programming
DOI	10.3233/SPR-140379
Online Date	Monday, March 24, 2014