Languages for numerical simulations

Roman Putanowicz
R.Putanowicz@L5.pk.edu.pl

Programming languages

There are multitude of programming languages and several ways of categorising them depending on their characteristics. Without going much into details we will distinguish two categories:

System programming languages (C, C++, Fortran, Java, Ada) – associated with the tags like: efficiency safety, static type control.

Scripting languages (Python, Ruby, Tcl, Guile, Ch) – associated with the tags like: rapid prototyping, flexibility, advanced introspection features

Programming languages for numerical simulations

Some languages are considered as better suited for writing numerical simulation codes. However picking the right language is a difficult thing often depending on non-technical issues (like available human resources in terms of programmers or local experts).
As most to the numerical algorithms utilize vector and matrix abstractions one important factor when evaluating a language is to what extend the language support direct use of these abstractions. Support for vector and matrices can be either built-in into a language (Matlab, Octave, Fortran) or can be provided by a set of libraries.
Some popular choices are: Ada, C, C++, Fortran (both 77 and 90 and above), Matlab, Octave, Python, Ch.

Python

What is Python

- Scripting, object-oriented, rapid prototyping, general purpose language
- Quite popular for writing scientific codes, especially when supported by C/C++/Fortran extension libraries
- Extensible and embeddable in applications

Selected Python features

- Portability – UNIX, Windows, Mac, BeOS, VMS, Cray, ...
- Compiles to interpreted byte code
- Automatic memeory management through reference counting
- Many GUI libraries
- Several extension modules (NumPy for numerics)

Python – language properties

Selected language properties:

- everything is an object,
- packages, modules, classes, functions,
- exceptions handling,
- dynamic typing, polymorphism,
- operator overloading,
- high level data types – lists, dictionaries, sets

Python – application area

Python is general programming language, but it shows its strength in some application areas:

- rapid prototyping
- throw-away programming
- web scripting
- steering scientific applications
- extension language
- GUI applications
- XML processing

Python – a glimpse at syntax

Code formating is a part of the syntax

```
1  def bisect(fun, a, b, tol):
2      "bisection algorithm"
3      if (fun(a)*fun(b) > 0):
4          raise ValueError, \
5          "No solut. in (%f, %f)" % (a,b)
6      while (abs(a-b) > tol):
7          x = (a+b)/2.0
8          print x
9          if (fun(a)*fun(x)<0):
10             a,b = a,x
11         else:
12             a,b = x,b
13     return x
14
15 def f(x):
16     return (x-2.0)*(x+1.0)
17
18 import re
19 s = raw_input("Give starting range : ")
20 a,b = map(float, re.split('[, ]+',s))
21 tol = float(raw_input("Give tolerance : "))
22 sol = bisect(f,a,b,tol)
23 print "Solution: ", sol
```

Introduction to GNU Octave

GNU Octave `http://www.octave.org`

- computing environment and programming language for numerical computing,
- matrix language – matrix operations built into the language
- Open Source software distributed on GPL licese, available for most operating systems.

History

Octave conceived as a textbook companion software, 1988.
John W. Eaton joins the development team, 1992.
First alpha release on January 4, 1993.
Version 1.0 published on February 17, 1994.
The newest version is 3.2.4.

Octave application areas

- Numerical modelling and simulations
- Data analysis
- Visualizations

Octave components

- command line interpreter
- scripting high level language
- numerical libraries
- external components for 2D and 3D visualisation

Working with Octave

Octave can be operated in two modes:

- interactive,
- batch processing.

- in both modes the programs are interpreted,
- both modes support the same instruction set,
- Octave scripts have suffix ".m".
- Script is a sequence of instructions.
- Instructions are separated by " " or " " , or newline.
- Line comments start with "%" lub "#"
- Octave interpreter is case sensitive

Operating Octave in interactive mode

- Starting Octave – **octave -q**.
- Octave prompt – **octave:1**>.
- Commands history ↑↓.
- Function **history**.
- Finishing Octave session: **quit** lub **exit**.

Some Octave applications

- Matrix calculations,
- Interpolation and approximation,
- Delaunay triangulation
- Convex hull and Voronoi diagrms

Matrix calculations

```
octave:2> A = [1,2;3 4]
A =
   1 2
   3 4

octave:3> A^-1
ans =
  -2.00000 1.00000
   1.50000 -0.50000

octave:4> det(A)
ans = -2
octave:5> b = [2;1]
b =
   2
   1

octave:6> x = A\b
x =
  -3.0000
   2.5000
```

Interpolation and approximation

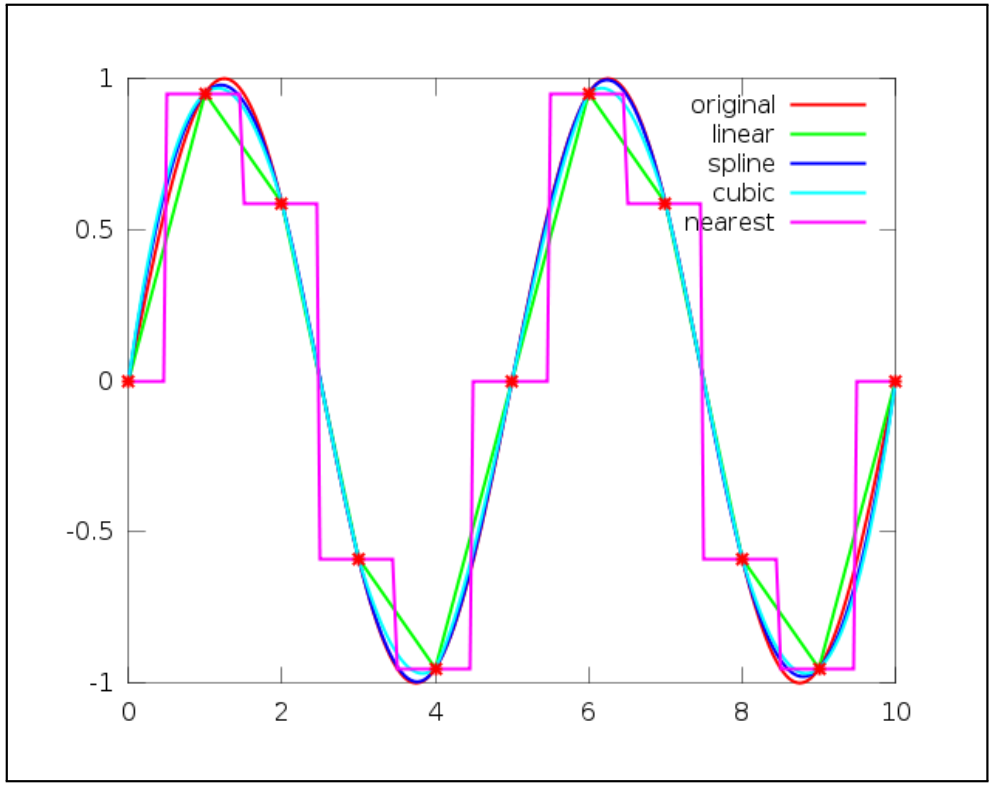
Octave provides several functions for interpolation and approximation.

- Scalar function interpolation with various methods, **iterp1**. Supported methods

  nearest – nearest neighbour,
  linear – linear interpolation from nearest neighbors,
  pchip – piece-wise cubic Hermite interpolating polynomial
  cubic – cubic interpolation from four nearest neighbors
  spline – cubic spline interpolation-smooth first and second derivatives

- Fourier interpolation, function: **interpft**
- Least square polynomial fitting, function: **polyfit**
- Interpolation on Scattered Data, 2D, 3D, nD, functions: **griddata, griddata3, griddatan**

Interpolation and approximation.

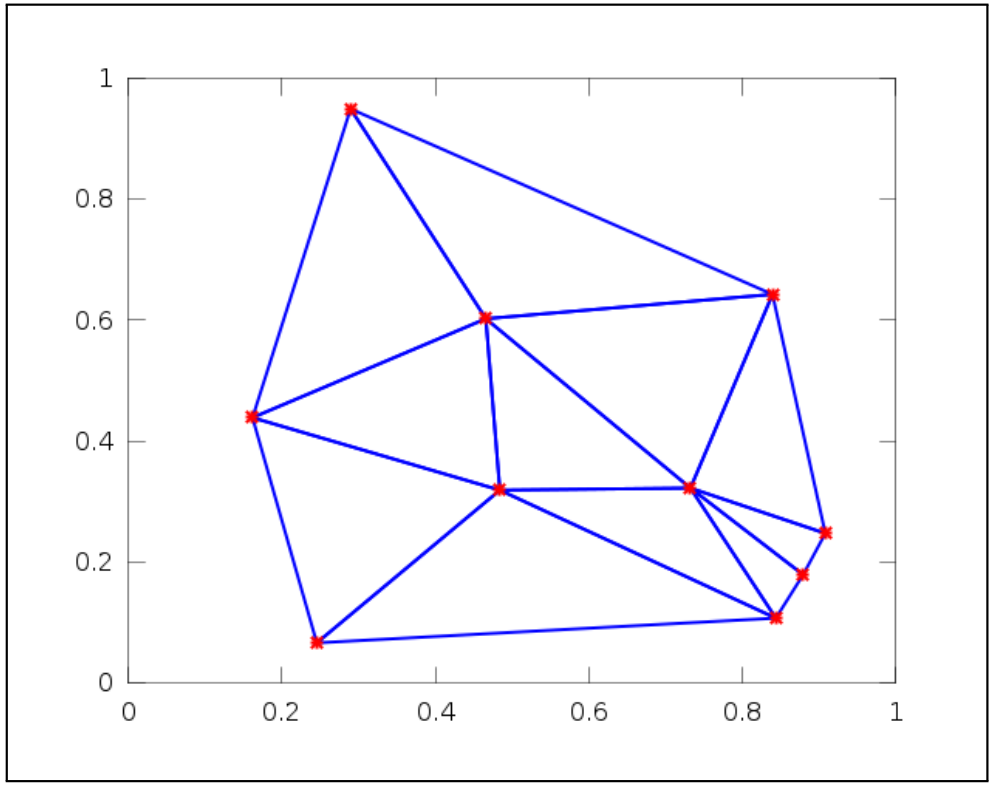



```
1   xf = [0:0.05:10];
2   yf = sin (2*pi*xf/5);
3   xp = [0:10];
4   yp = sin (2*pi*xp/5);
5   lin = interp1 (xp, yp, xf);
6   spl = interp1 (xp, yp, xf,
    "spline");
7   cub = interp1 (xp, yp, xf,
    "cubic");
8   near = interp1 (xp, yp, xf, "nearest");
9   h = plot (xf, yf, "r", xf, lin, "g", xf, spl, "b",
    xf, cub, "c", xf, near, "m", xp, yp, "r*");
10  legend ("original", "linear", "spline", "cubic", "nearest")
11  set(h, "linewidth", 2.0)
12  print("interp.png", "-S640,480")
```

Example from Octave documentation, Chapter 28.1

Mesh generation

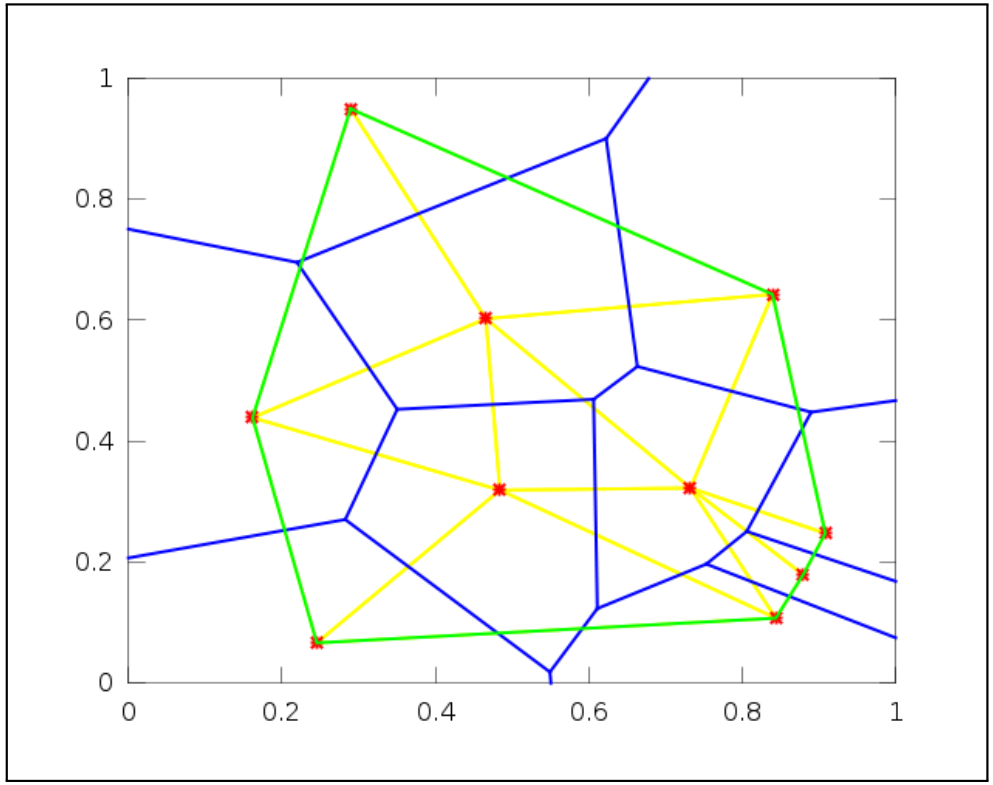
Octave provides Delaunay triangulation for 2D, 3D and nD point sets.
The file `coords` consists in 10 points coordinates, saved in vectors `x` i `y`, respectively




```
1  load "coords"
2  T = delaunay (x, y);
3  X = [x(T(:,1)); x(T(:,2)); x(T(:,3)); x(T(:,1))];
4  Y = [y(T(:,1)); y(T(:,2)); y(T(:,3)); y(T(:,1))];
5  axis ([0,1,0,1]);
6  h = plot (X, Y, "b", x, y, "r*");
7  set(h, "linewidth", 2.0)
8  print("triang.png", "-S640,480")
```

Geometric algorithms

Convex hull and Voronoi diagram for given Delaunay triangulation.




```
load "coords"
T = delaunay (x, y);
X = [x(T(:,1)); x(T(:,2)); x(T(:,3)); x(T(:,1))];
Y = [y(T(:,1)); y(T(:,2)); y(T(:,3)); y(T(:,1))];
ch = convhull (x, y);
[vx, vy] = voronoi (x, y);
h=plot (X, Y, "-y", vx, vy, "-b", x, y, "r*", x(ch), y(ch), "-g")
set(h, "linewidth", 2.0)
axis([0,1,0,1])
print("dual.png", "-S640,480")
```

References

1. GNU Octave web page `http://www.gnu.org`
2. GNU Octave. A high-level interactive language for numerical computations, by John W. Eaton, David Bateman, Søren Hauberg, edition 3 for Octave version 3.0.2, Network Theory Ltd, 2008
3. Python web page `http://www.python.org/`
4. Own materials

Thank you for your attention

License