Computational Physics (6810): Session 10

Dick Furnstahl
Nuclear Theory Group
OSU Physics Department

March 22, 2017

Mathematica Notebooks (cf. Session 8)

Session 10 Stuff

Mathematica Notebooks (cf. Session 8)

- Note: it is not necessary to understand the Mathematica code
  - But look at the results carefully!
- `nonlinear.nb` looks at the Duffing equation
  - Revisit the ingredients of `diffeq_pendulum` from Session 8
  - Poincaré sections: once every period of driving force
  - Power spectrum (Fourier transform)
    - How is the power (energy per time) distributed among different frequencies?
    - What does a peak at zero frequency mean? How can you get rid of it?
- `pendulum.nb` recreates nonlinear pendulum results
  - Select from Session 8 table parameters
  - `p2` is a limit cycle; compare Poincaré map and power spectrum $\Longrightarrow$ do they correspond?
  - Identify different frequencies in theta(t) plots

Session 10 stuff

- GSL adaptive ODE solver
  - Use when a specified accuracy is required and you can afford the extra computing time
  - Version using classes next time: see Session 10 notes
    - Works, but not likely to be an optimal implementation
  - Van der Pol oscillator (nonlinear differential equation)
    - plot from three different initial points using gnuplot "with lines"
    - "isolated attractors" $\Longrightarrow$ explain what this means
- Interpolation: sample problem using theoretical model
  - E.g., imagine that calculating every point is too expensive
  - Usually based on idea that every *differentiable* function looks like a polynomial locally
  - Be careful of applying to *data*: fit rather than interpolate
  - *Global* polynomials are problematic (as you'll see) but putting together local (low order) polynomials is fine. Why?

Session 10 stuff (cont.)

- Interpolation (cont.)
  - A spline pieces together *mostly* smooth polynomial sections.
    - Cubic spline $\Longrightarrow$ continuous up through the 2nd derivative
    - Boundary conditions (end points) can be a problem
    - Generally good, but often other polynomial methods better
  - Other ways not considered here:
    - Barycentric Lagrange interpolation $\Longrightarrow$ improved! (see ref.)
    - Rational function interpolation (and particularly extrapolation)
    - Orthogonal polynomial interpolation
    - Multidimensional interpolation (usually iterative)
- Python scripts to run C++ programs
  - A "script" is just a program that acts externally on a system
  - Here: set up input, run a C++ code, and process output
  - Replace interactive input with command-line arguments
    - conventional: `main (int argc, char *argv[])`
    - `argc` is # of arguments, including program name
    - `argv[i]` is i'th argument (0 is name)
  - Python by example! Use generic scripts and adapt to needs