ACS2: A “yet another” toolbox for stellar dynamics

Jun Makino

Dec9, 2020 Internal seminar

Talk overview

- What is a toolbox?
- What are available?
- Why yet another toolbox?
- Current status

What is a toolbox?

For numerical experiments, a set of useful programs to perform experiments.

- prepare initial conditions
- perform numerical integration
- display/analyze results

What are available?

- NEMO
- (Starlab)
- AMUSE
- ACS

NEMO

https://teuben.github.io/nemo/

- Development started by Barnes and Hut, around 1985.
- Currently maintained by Peter Teuben
- Contains many, many useful programs,
  - Initial models: Plummer, Hernquist, King. Also contain interfaces to GalactICS
  - Many utilities to modify snapshot files: rotate, move, scale, add, ...
  - Integrator: tree, interfaces to NBODYx, gyrfalcON
  - Visualization/analysis: snapplot, glnemo, ...

Basic structure of NEMO

- Command-line interface: get[id]param functions (not Gnu getopt...)
- Nbody snapshot file format built on “structured file”
- Flexible processing of particle data using bodytrans library (using on-the-fly compiling and dynamic loading)

What you can do: some example

```
% mkplummer out=pl1k.snap nbody=1024 # make a plummer model with
                                     #1024 particles
% tsf                                # show the file content
% snapplot pl1k.snap
% snapplot pl1k.snap xvar=x yvar="vx*vx" # vx*vx is compiled
                                        # at runtime
```

“Problems” with NEMO

- Writing NEMO programs is not easy. CLI and File I/O are well designed, but both require fairly long and complex user code. (Same is true for Gnu getopt)
- File I/O is limited to predefined data structure (such as Nbody or SPH. No simple way to add new variable to these data, or to let old programs talk to new data structure

StarLab

- Started as the replacement of NBODY3/4/5/6 (collisional N-body code). Initially by Hut and JM, McMillan and Portegies Zwart joined. Used with GRAPE-4, 6.
- Written in C++
- File structure changed to text-base, particle-wise structure
- I/O library can handle “unknown” data, but as array of strings.
- Rely on NEMO for setup/analysis
- Sort of superseded by AMUSE.

AMUSE

https://amusecode.github.io/

- The outcome of MODEST (Modeling Dense Stellar Systems) collaboration, started in 2000?
- Portegies Zwart and McMillan
- I must say I do not know much about the inside of AMUSE. I only know it is hard to install..
- Python based.

ACS

The Art of Computational Science
http://www.artcompsci.org

- Hut and JM started in somewhere in 1999-2002
- Effort to write up everything you need to know to do research in stellar dynamics
- Have not dealt reached that far...
- Ruby based
- Fairly fancy CLI and File I/O, designed to remove everything we did not like about NEMO and Starlab.

CLI example NEMO mkplummer

```
string  defv[] = {                          /* DEFAULT INPUT PARAMETERS */
    "nbody=???\n       Number of particles",
    "mfrac=0.999\n             Mass fraction used of Plummer distribution",
    NULL,
};
void nemo_main(void)
{
  int     nbody;
  real    mfrac;
  Bbody = getiparam("nbody");
  mfrac = getdparam("mfrac");
   ...
```

This is actually much simpler than using getopt.

The getopt way

```
void usage()
{
    fprintf(stderr,"  -n: Number of particles\n");
    fprintf(stderr,"  -m: Mass fraction used of Plummer distribution\n");
}

int main(int argc, char * argv[])
{
    int ch;
    int nbody;
    double mfrac;
    static struct option longopts[] = {
        { "nbody", optional_argument, NULL, 'n' },
        {"mfrac", optional_argument, NULL, 'm'},
        { NULL,         0,                      NULL,           0 }
    };
    mfrac=0.999;
    while((ch=getopt_long(argc,argv,"m:n:",longopts, NULL))!= -1){
        fprintf(stderr,"optchar = %c optarg=%s\n", ch,optarg);
        switch (ch) {
            case 'n': nbody= atoi(optarg); break;
            case 'm': mfrac= atof(optarg); break;
            default:break;
        }
    }
    ...
```

The ACS way

```
options_text= <<-END
...
  Short name: -n
  Long name:            --n_particles
  Value type:           int
  Default value:        1
  Variable name:        n
  Description:          Number of particles
  Long description:
    Number of particles in a realization of Plummer's Model.

    Each particles is drawn at random from the Plummer distribution,
    and therefore there are no correlations between the particles.

  Short name:           -m
  Long name:            --mfrac
  Value type:           real
  Default value:        0.999
  Description:          Mass fraction used of Plummer distribution
  Variable name:        mfrac
  Long description: Mass fraction used of Plummer distribution
  END

c=parse_command_line(options_text, true)
c.nbody ...
c.mfrac ...
```

Comparison

- With gnu getopt, one option character and its associated variable appears in five different places.
- With NEMO getparam only in two places.
- With ACS parse_command_line in just one place. Also, it allows very long (man page like) documentation.
- ACS relies on dynamic nature of Ruby, where the class for options is created at runtime.
- For data structure, dynamic class creation is used. Thus, any ACS program can read/write snapshot data which contains “new” variables, and plotting/analysis programs can use them.

Why yet another toolbox?

- ACS is (in my opinion...) reasonably designed and easy to use and easy to write new programs.
- Major problem is the performance penalty of Ruby. We hoped (in 2004 or so) that some reasonable compiler for Ruby would appear.
- Real compiler is not there yet... Ruby 3.0 will feature JIT compiler but its expected performance would not match native C/C++
- Crystal, a new compiler language, inspired by Ruby, seemed to be promising.
- So I decided to “modernize” ACS using Crystal.
- FDPS can be called from Crystal. So we can write MPI parallel programs, not only for time integration but also for anslysis, using Crystal.

Crystal CLI

```
optionstr = <<-END
  Description: Plummer's Model Builder

  Short name: -n
  Long name:            --n_particles
  Value type:           int
  Default value:        1
  Variable name:        n
  Description:          Number of particles
  Long description:
    Number of particles in a realization of Plummer's Model.

    Each particles is drawn at random from the Plummer distribution,
    and therefore there are no correlations between the particles.

  Short name:           -m
  Long name:            --mfrac
  Value type:           real
  Default value:        0.999
  Description:          Mass fraction used of Plummer distribution
  Variable name:        mfrac
  Long description: Mass fraction used of Plummer distribution
END
clop_init(__LINE__, __FILE__, __DIR__, "optionstr")
c=CLOP.new(optionstr,ARGV)
```

The text part is the same as that for ACS. But now implemented using YAML. File IO is also based on YAML.

Current status

- Basic tools (CLI, File IO, dynamic evaluation) are there
- Can work with FDPS (Both MPI/OpenMP)
- source code at: https://github.com/jmakino/numerical-calculation-with-crystal
- “Documentation” at http://jun-makino.sakura.ne.jp/articles/intro_crystal/face.html (Sorry, curently only in Japanese).
- Need more compact documents for basic tools and tutorials.
- Will add more functions/documents.