Software design of the 3d BEPS3 PIC codes

Viktor K. Decyk
Department of Physics and Astronomy
University of California, Los Angeles
Los Angles, California 90095-1547

Introduction

This Particle-in-Cell (PIC) code is intended for research and teaching in plasma physics. Plasmas are ionized gases interacting with electromagnetic fields they generate. It is an example of a many body system described by statistical mechanics. PIC codes model plasmas as discrete particles and this code serves as a lab to perform numerical experiments.

This software consists of three separate Particle-in-Cell codes: an electrostatic, and electromagnetic, and a darwin code. They are based on the 3d OpenMP/MPI skeleton codes (mppic3, mpbpic3, mpdpic3) available at: http://picksc.idre.ucla.edu/software/skeleton-code/, with additional diagnostics and initial conditions added. The codes are written in layers. They are intended to be run on a large variety of platforms, from student laptops to supercomputers. The lowest, most compute intensive layer is mostly written in a Fortran77 subset of Fortran90. This layer uses only basic types, without array syntax and compiles to very fast code. It is easy to replace this layer with a C language layer. It contains about 100 procedures and 23,000 lines of code. The middle layer consists of Fortran90 wrappers, which simplifies the argument lists, introduces some polymorphism with case statements, adds safety checks, and is designed to be easily called from Python. It consists of 85 procedures and 3,000 lines of code organized in 10 libraries. An upper layer is planned for both Fortran and Python.

Low level libraries

The libraries are organized according the type of code, electrostatic, electromagnetic, and darwin. Seven of the libraries are used by all 3 codes:

```
libmpinit3.f: initializes particles
libmppush3.f: pushes electrostatic particles, deposits charge,
              and provides utility functions
libmpsort3.f: reorders particles for parallelization
libmpgard3.f: provides functions to process guard cells
libmpfield3.f: provides spectral field solvers and diagnostics
libmpfft3.f: provides 1D FFTs for scalar and vector arrays
```

`mpplib3.f90 provides communications procedures using MPI/OpenMP`

Two of the libraries are used by the electromagnetic and darwin codes:
`libmpcurd3.f: deposits current density`
`libmpbpush3.f: pushes electromagnetic particles`

One library is used by darwin code:
`libmpdpush3.f: deposits time derivative of current density`

Comments at the beginning of each library describe what each individual procedure does and comments at the beginning of each function give additional details as well as information about the input and output variables.

Middle level libraries

The ten middle level libraries provide an easier to use interface to the low level libraries and can be called by Python. They provide error checking but do not process errors (which may need to be processed in another language.) They also provide some level of polymorphism, such as whether a relativistic version of procedures should be used.

The middle level wrapper libraries have the same structure as the low level libraries. They are written to conform to the Fortran 90 standard. The names are similar, except that they usually start with `mod...` and end in `...f90` instead of `lib...` and `...f`. For example, the wrapper for `libmpinit3.f` is `modmpinit3.f90`. In addition, there are libraries that provide interfaces to the low level procedures to support argument checking for procedures (similar to header files in C). Their names are the same as the low level libraries, except they terminate with `_h.f90` instead of `.f`.

Comments at the beginning of each library describe what each individual procedure does and what low level procedures are called.

The Python wrappers can be created automatically from the middle layer by the numpy tool f2py. This required that the middle layer avoid certain Fortran90 language features, such as derived types and function overloading, and required that they provide the `intent` attribute for dummy arguments. The attribute `intent(inout)` was used whenever a variable or array was modified, and `intent(in)` otherwise. All communication between Python and Fortran will occur only in the middle layer,

Input to the code currently consists of about 50 variables which are defined with short descriptions and given default values in the main codes. Organizing them into namelists is planned.

Main codes

Three Fortran90 main codes were written for each code, `mpbeps3.f90`, `mpbbeps3.f90`, and `mpdbeps3.f90`. The libraries and main codes are compiled by a Makefile.