Software design of the 2d BEPS2 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 2d OpenMP/MPI skeleton codes (mppic2, mpbpic2, mpdpic2) available at: http://picksc.idre.ucla.edu/software/skeleton-code/, with additional diagnostics and initial conditions added. The codes are written in layers. It is 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 200 procedures and 31,000 lines of code organized in 16 libraries. 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 about 170 procedures and 5,800 lines of code organized in 11 libraries. The upper layer consists of 3 high level libraries which conform to the Fortran2003 standard and allows implementation of such features as object oriented programming, interoperability with C, and stream IO. They are not intended to be interoperable with Python. The three high level libraries currently contain about 12 procedures and 700 lines of code.

Low level libraries

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

```
libmpinit2.f: initializes particles
libmppush2.f: pushes electrostatic particles, deposits charge,
```

```
              and provides utility functions
libvmppush2.f: pushes electrostatic particles, deposits charge,
               and provides utility functions with vectorizable
               algorithms
libmpsort2.f: reorders particles for parallelization
libvmpsort2.f: reorders particles for parallelization with
               vectorizable algorithms
libmpgard2.f: provides functions to process guard cells
libmpfield2.f: provides spectral field solvers and diagnostics
libvmpfft2.f: provides 2D FFTs for scalar and vector arrays
libmpdiag2.f: provides diagnostic procedures such as
              distribution functions, frequency analysis, and
              others
mpplib2.f90 provides communications procedures using MPI/OpenMP
```

Four of the libraries are used by the electromagnetic and darwin codes:

```
libmpcurd2.f: deposits current density
libvmpcurd2.f: deposits current density with vectorizable
               algorithms
libmpbpush2.f: pushes electromagnetic particles
libvmpbpush2.f: pushes electromagnetic particles with
                vectorizable algorithms
```

Two libraries are used by the darwin code:

```
libmpdpush2.f: deposits time derivative of current density
libvmpdpush2.f: deposits time derivative of current density with
                vectorizable algorithms
```

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 eleven 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 `libmpinit2.f` is `modmpinit2.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 codes currently consists of about 176 `namelist` variables in 4 different `namelists`. The variables are defined and default values are given in the Fortran90 file `input2mod.f90`, which defines a module called `in2`. The `namelist input2` is used by all three codes. The `namelist input2b` is used by the electromagnetic and darwin codes, and the `namelist input2d` is used only by the darwin code. There is also a `namelist ions2` which is used by all 3 codes if ions are mobile. Comments in the module `input2mod.f90` give short descriptions of each input variable and its usage.

High Level libraries

Three Fortran2003 high level libraries are defined. The main purpose of these libraries is to encapsulate the data and the main steps of the PIC code and the major diagnostics. Currently, only the restart procedures are encapsulated.

The library `mpsimul2.f03` is used by all three codes. This library defines a module `f2` which provides support for the following:

1. Support for restart files for electrostatic code

The library `mpbsimul2.f03` is used by the electromagnetic and darwin codes. This library defines a module `fb2` which provides support for the following:

1. Support for restart files for electromagnetic code

The library `mpdsimul2.f03` is used by the darwin code. This library defines a module `fd2` which provides support for the following:

1. Support for restart files for darwin code

Comments at the beginning of each library describe what each individual procedure does.

The three high level Fortran modules f2, fb2, and fd2 were translated line by line to three Python scripts entitled s2.py, sb2,py, and sd2.py, respectively, using numpy arrays and procedures to replace Fortran90 array syntax. There are no GUI elements in these Python scripts and they make use of OpenMP only.

Main codes

Three Fortran90 main codes were written for each code, `mpbeps2.f90`, `mpbbeps2.f90`, and `mpdbeps2.f90`. Three Python main scripts were written by translating the Fortran main codes line by line, creating mbeps2.py, mbbeps2.py, and mdbeps2.py. The libraries and main codes are compiled by a Makefile.