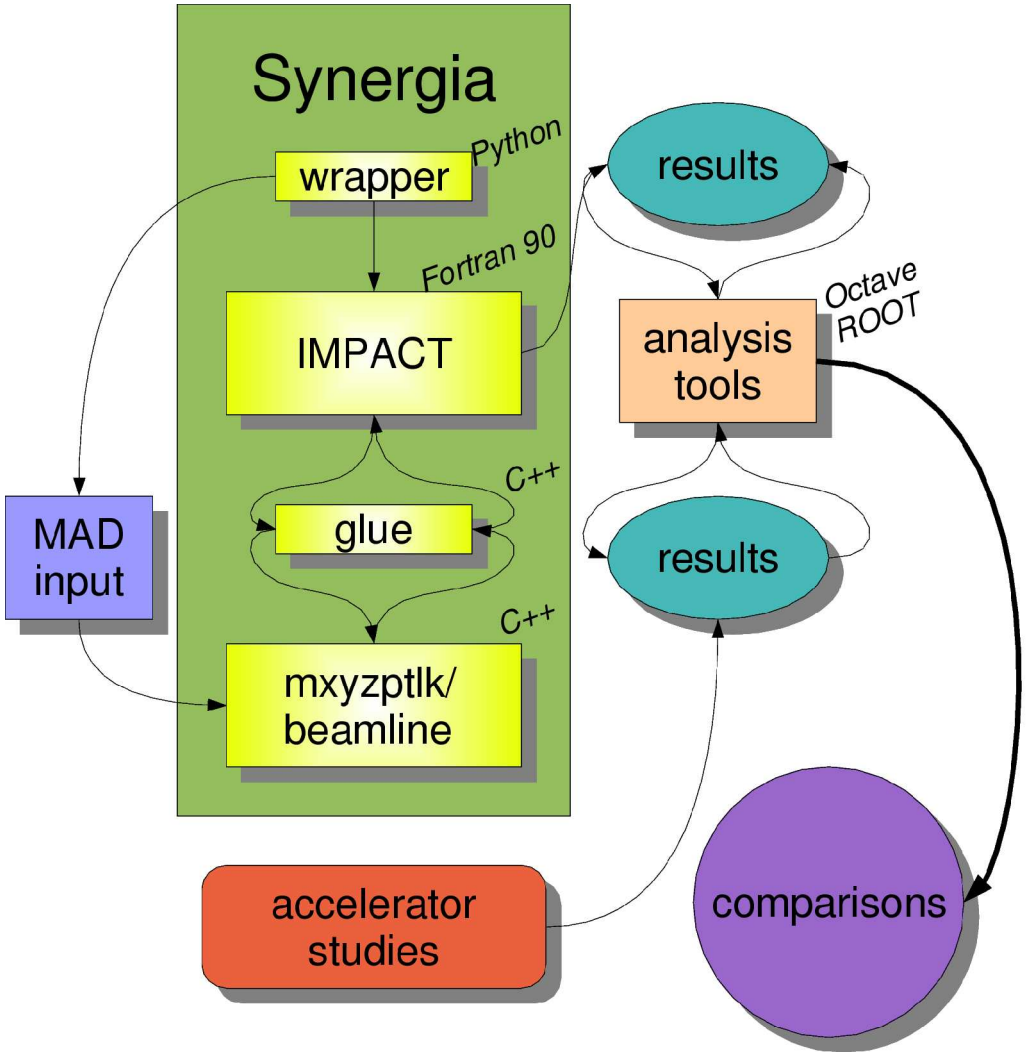
Συνεργεια

Synergia Code Development

- Overview
- IMPACT
- mxyzptlk / beamline
- Synergia
  - combining codes
  - portable compilation
  - interface
  - documentation
- Periodic Boundary Conditions
- Performance
- Next Steps

Συνεργεια

Overview



- Synergia is a combination of IMPACT, mxyzptlk/beamline, glue code and a wrapper
- IMPACT
  - program flow
  - space charge
- mxyzptlk/beamline
  - mad parser
  - linear maps

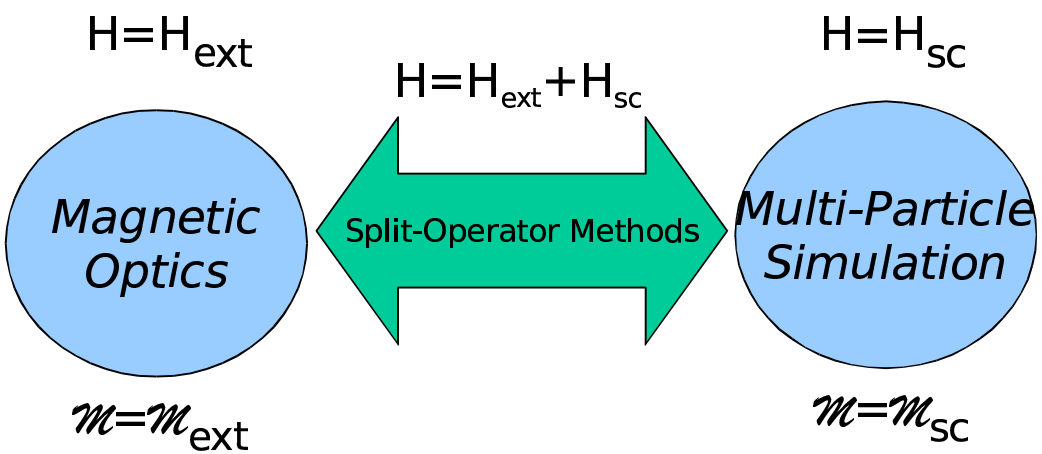
Συνεργεια

IMPACT

- J. Qiang, R. Ryne and S. Habib
- Fortran 90
- Linear accelerators
- Full 3D Space Charge calculation
- Fully parallel
  - Space Charge
  - Particle Manager

Space Charge in IMPACT

- Solve Poisson-Vlasov Equation
  - particle-in-cell (PIC)
- Split Operator Method



$$\mathcal{M}(t) = \mathcal{M}_{ext}(t/2)\, \mathcal{M}_{sc}(t)\, \mathcal{M}_{ext}(t/2) + O(t^3)$$

Συνεργεια

Space Charge in IMPACT (2)

- Fixed-time calculation using ballistic approximation
  - $$x^t = x^z - \frac{c}{\omega} p_x \psi / \gamma \qquad y^t = y^z - \frac{c}{\omega} p_y \psi / \gamma$$

$$\delta z = \frac{-c}{\omega} \beta_z \psi$$

Συνεργεια

mxyzptlk/beamline library suite

- L. Michelotti
- C++
  - first C++ library for accelerator physics
- Flexible libraries
- Many features
  - Relevant for us
    - Propagator functors
    - MAD file parser

Συνεργεια

mxyzptlk/beamline libraries

- basic_toolkit
  - Useful utility classes: Vector, Matrix...
- mxyzptlk
  - Automatic differentiation and differential algebra
- beamline
  - Objects for modeling elements of a beamline
- physics_toolkit
  - analysis and computation
- Machines
  - FNAL models

Συνεργεια

Language mixing

- Mixing F90 and C++ can be done
  - Requires some glue
  - Always platform/compiler-dependent
    - Macros for general code
    - Compiling and linking
      - see configuration managment

Συνεργεια

Glue in Synergia

- Added a new element to IMPACT, “external”
  - F90 modifications to IMPACT
  - New skeleton F90 module
- Created a C++ class, External_class, to get maps from a MAD file using mxyzptlk
- “F90 module” really C++ code that talks to External_class

Συνεργεια

Configuration Management

- Building and installing code is not glamorous
  - In a mixed-language environment it is also not trivial
- Configuration management can rapidly become a bottleneck
- GNU Autotools
  - “The worst possible choice, except for all the others”

Συνεργεια

Building Synergia with GNU Autotools

- In principle
  - Get mxyzptlk
    - ./configure && make && make install
  - Get IMPACT
    - ./configure && make
- In practice, there are more decisions to make...

Συνεργεια

In practice

```
> ./configure --help
<snip>
  --with-glib-prefix=PFX   Prefix where GLIB is installed (optional)
  --with-glib-exec-prefix=PFX Exec prefix where GLIB is installed (optional)
  --with-mxyzptlk-prefix=<dir>  Prefix directory for mxyzptlk.
    Default is to search /usr/local, \$HOME, \$HOME/opt, \$HOME/mxyzptlk
  --with-mpi-prefix=<dir>   Prefix directory for MPI.
  --with-mpi-include-dir=<dir>  MPI include directory.
    Default is -I\$MPI_PREFIX/include
  --with-mpi-ldflags=<flags>    LDFLAGS for linking with MPI.
  --with-mpi-libs=<libs>    Libraries for linking with MPI.

Some influential environment variables:
  CXX         C++ compiler command
  CXXFLAGS    C++ compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  CPPFLAGS    C/C++ preprocessor flags, e.g. -I<include dir> if you have
              headers in a nonstandard directory <include dir>
  F77         Fortran 77 compiler command
  FFLAGS      Fortran 77 compiler flags
  CC          C compiler command
  CFLAGS      C compiler flags
```

Συνεργεια

Synergia Build System

- Default choices are stored in installation scripts (configure.in)
- Compile-time choices are recorded (config.status)
- No source-code changes needed for configuration choices (preprocessor macros)
- New platforms are anticipated (Autotools)

Συνεργεια

Synergia Interface

- The program flow in Synergia is controlled by IMPACT
  - IMPACT's interface is something less than human-friendly
- Synergia wraps IMPACT with a Python Layer
  - Unit conversions
  - Defaults
  - Full power of Python
  - Job creation/submission

Συνεργεια

IMPACT Interface Example

```
16 4
6 2746880 1 0 1
65 65 65 4 0.04 0.04 1.51692
2 0 0
0.00268186, 0.000106579, 0 1.000000 1.000000 0.000000 0.000000
0.00268186, 0.000106579, 0 1.000000 1.000000 0.000000 0.000000
0.0940268, 0.000427895, 0 1.000000 1.000000 0.000000 0.000000
0.042 4e+08 9.38272e+08 1.000000 2.01e+08 0.000000
  0 0 81 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 83 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 84 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 85 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 86 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 87 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 88 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 89 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 90 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 91 -2 270.000000/
  474.2 100 1 91 1.0 0. 0.040000 0. 0. 0. 0. 0. /
  0 0 92 -2 270.000000/
  0 0 82 -2 270.000000/
```

Synergia Interface Example

```
ip = impact_parameters.Impact_parameters()
ip.processors(16,4)
ip.space_charge_BC("trans finite, long periodic round")
ip.input_distribution("6d gaussian")
ip.pipe_dimensions(0.04,0.04)
ip.kinetic_energy(0.400)
ip.scaling_frequency(201.0e6)
ip.x_params(sigma = .004 , lam = 1.0e-4)
ip.y_params(sigma = .004 , lam = 1.0e-4)
pz = ip.gamma() * ip.beta()*ip.mass_GeV
ip.z_params(sigma = 0.10, lam = 3.0e-4 * pz)
ip.particles(2700000)
ip.space_charge_grid(65,65,65)
booster = impact_elements.External_element(length=474.2,
                                           kicks=100, steps=1, radius=0.04,
                                           mad_file_name="booster.mad")
for turn in range(1,11):
    ip.add(booster)
my_impact = impact.Impact(ip)
currents = ( 1.0e-10, 0.011, 0.020, 0.042 )
for current in currents:
    ip.current(current)
    my_impact.prepare_run("bs1_I%0.3f" % current,
                          clock_limit = "01:30:00")
    my_impact.submit()
```

Identifiable names

Beam parameters specified in m, GeV

Trivial use of Python

multiple runs in one script

Συνεργεια

Documentation

- Documentation is in the form of web pages
  - Examples
  - API
- New external users will be able to help us refine the documentation

Synergia Documentation

Example Synergia scripts

- Example 1: FNAL Booster, no space charge
- Example 2: FNAL Booster with space charge
- Example 3: FNAL Booster, multiple turns

Synergia API references

- impact.py *Methods for generating and submitting jobs*
- impact_parameters.py *Methods for specifying simulation parameters*
- impact_elements.py *Methods for specifying beam line elements*

IMPACT documentation

Συνεργεια

API Documentation

Classes

BasicRF_element
Drift_element
External_element
Injection_element
Output_element

class **BasicRF_element**

__init__(self, length, steps, map_steps, field_scaling, rf_freq, driven_phase_deg, file_name, radius_m)
describe(self, file=<open file '<stdout>', mode 'w'>)
preprocess(self)
write_line(self, file)

class **Drift_element**

__init__(self, length, kicks, steps, radius)

- Documentation is automatically generated from source
  - always accurate
  - extensible
  - takes advantage of the power of Python

Συνεργεια

API Documentation, cont.

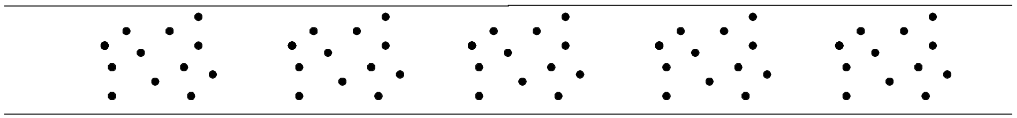
- Methods have descriptive names
  - optional text
- Variables have descriptive names
  - including units

class **Impact_parameters**

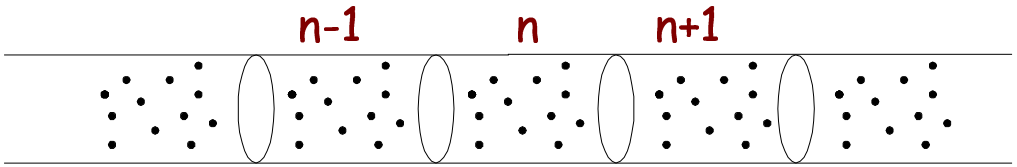
```
__init__(self, sampling_period=1)
_adjust_particles(self)
add(self, element, final=0)
beta(self)
    returns the velocity of the reference particle in natural units
charge(self, charge_e)
current(self, current_A)
describe(self, file=<open file '<stdout>', mode 'w'>)
dimensions(self, dim)
error_studies(self, bool)
gamma(self)
initial_phase(self, phase)
input_distribution(self, name)
integrator(self, name)
kinetic_energy(self, energy_GeV)
mass(self, mass_GeV)
omega(self)
output_type(self, name)
particles(self, number)
pipe_dimensions(self, width_x_m, width_y_m)
processors(self, procs_per_node, nodes)
restart(self, bool)
scaling_frequency(self, frequency_Hz)
space_charge_BC(self, name)
space_charge_grid(self, nx, ny, nz)
```

Periodic boundary conditions and long beams

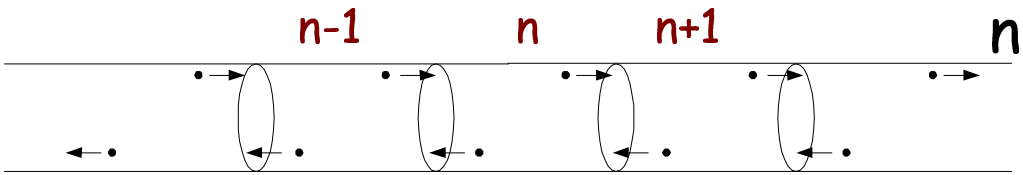
Take a long periodic (or uniform) beam



Look at period n



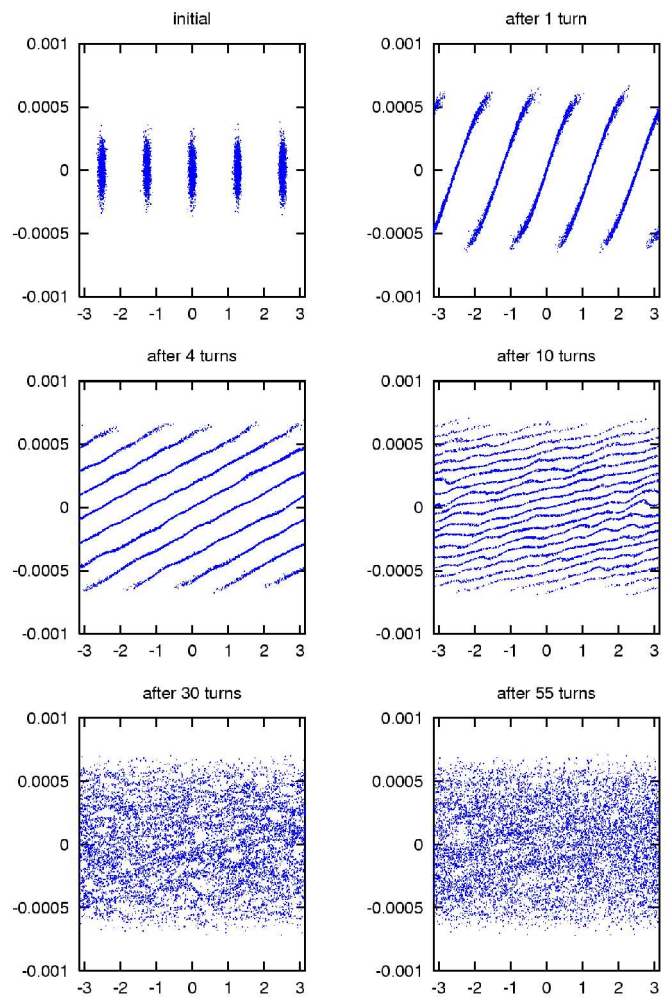
Each particle leaving n corresponds to another particle entering n from the other slide



We simulate a single slice (n) with periodic boundary conditions

Example: FNAL Booster Periodic BCs with RF

- Without RF, natural choice for period is 200 MHz Linac structure
- With RF, natural choice for period is 37.8 MHz RF structure
  - Need to start with 5 “200” MHz bunchlets
  - Tools are in place to do this



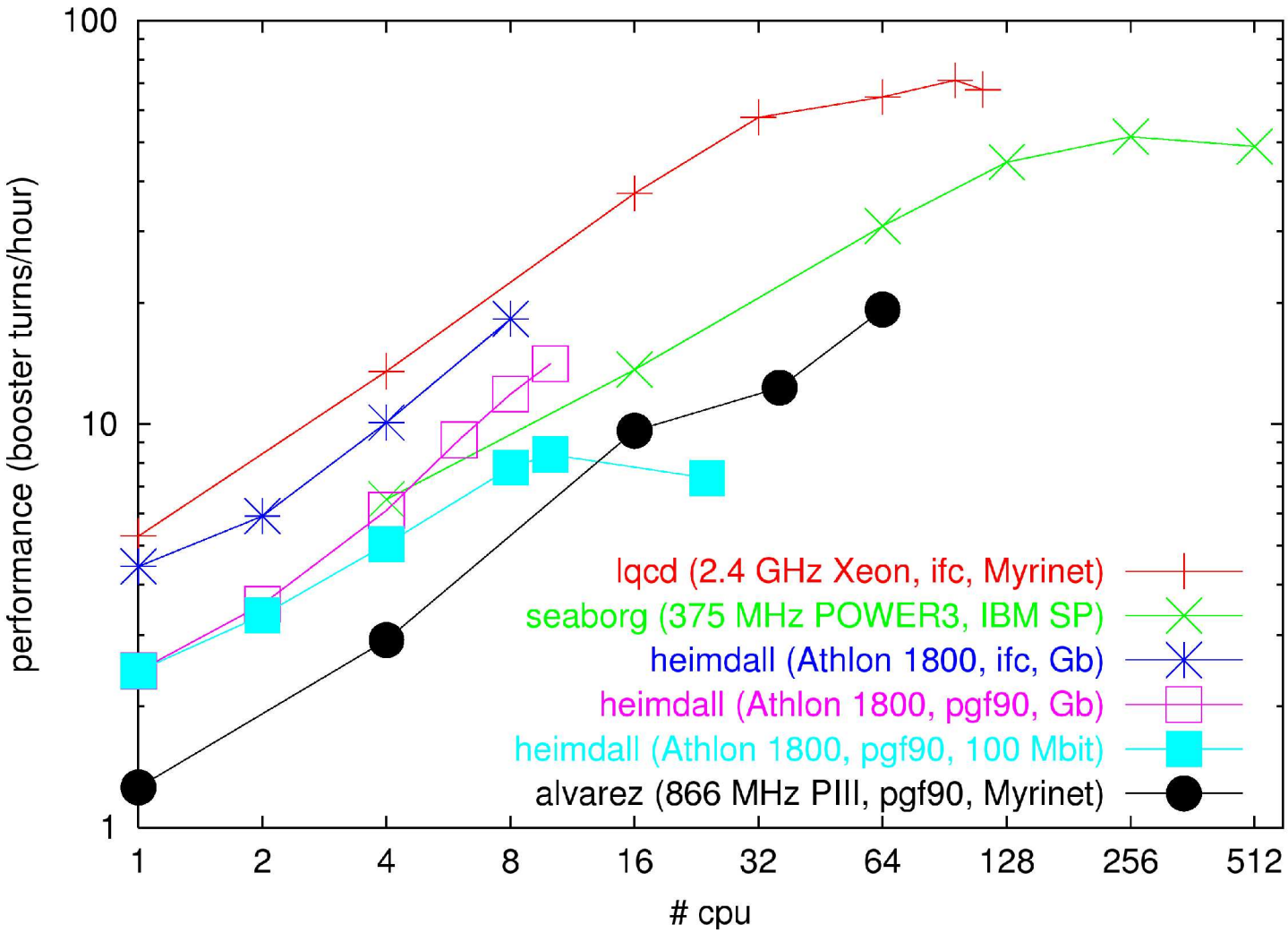
Συνεργεια

Performance and Parallelism

- Ran benchmarks on five machines
  - abacus
    - my 800 MHz laptop
  - heimdall (FNAL)
    - 32 dual 1.4 GHz Athlons, 100 Mbit/s and Gigabit interfaces
  - alvarez (NERSC)
    - 80 dual 866 MHz PIII, Myrinet
  - Seaborg (NERSC)
    - 6,000+ 375 MHz POWER3 processors
  - Lattice QCD (FNAL)
    - 128 dual 2.4 GHz Xeons, Myrinet

Συνεργεια

Results with space charge

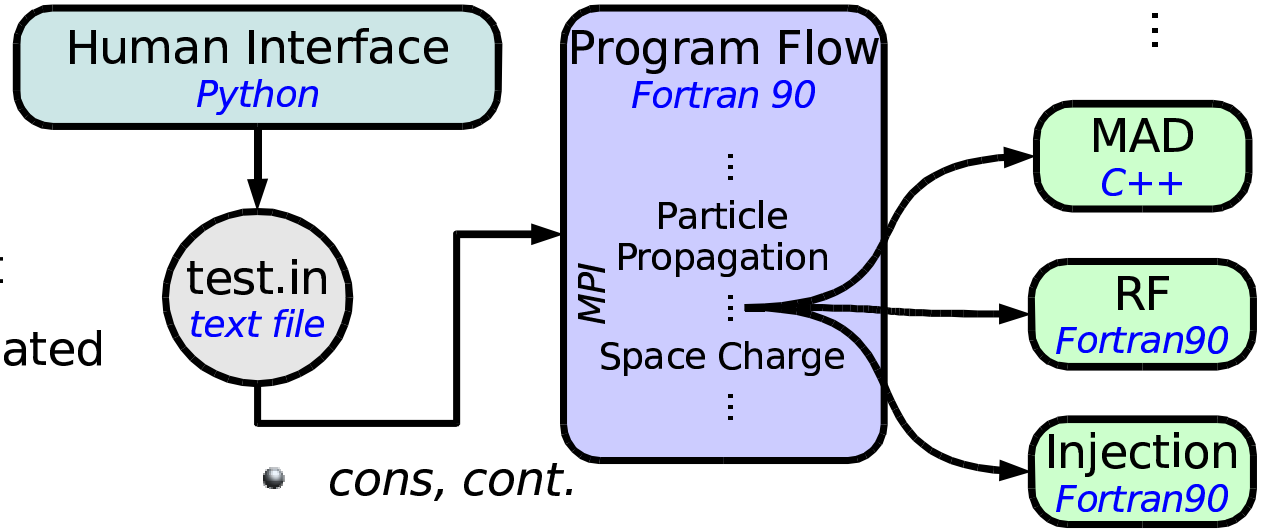


Συνεργεια

Next steps in development

- Synergia development to date has been driven by FNAL Booster applications
- Further applications will be easier if the program flow is made more flexible
  - New physics can be made much easier to add
- Peter Stoltz has SBIR funding for six months (in phase one) of development

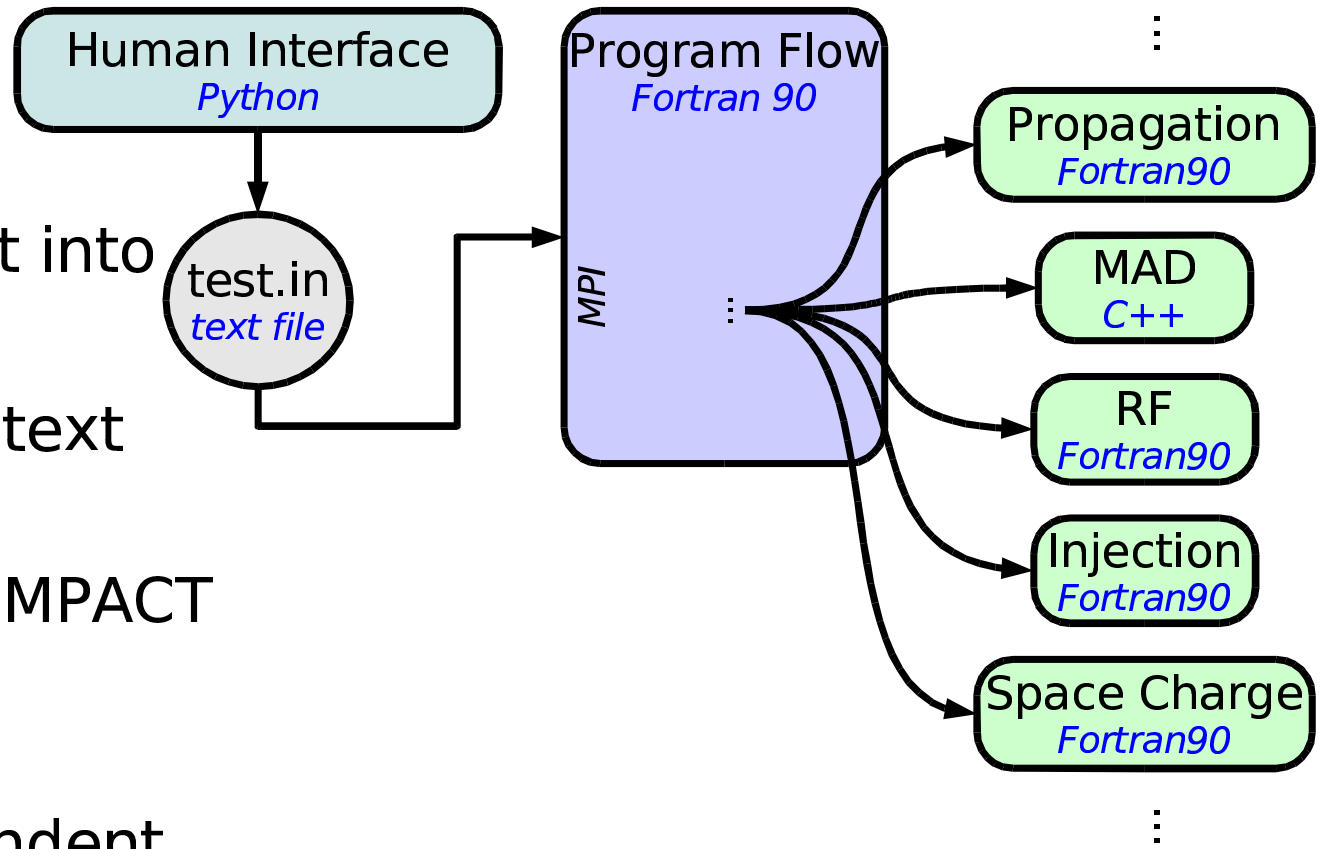
Current Scheme



- *pros*
  - simple to implement
  - minimal platform-related problems
- *cons*
  - difficult to extend for new physics
    - must extend IMPACT object system with N^2 connections
- *cons, cont.*
  - interfaces pass through text file
    - ...and must be parsed by Fortran code
  - IMPACT features (parallelism, SC) not exportable

Next Step

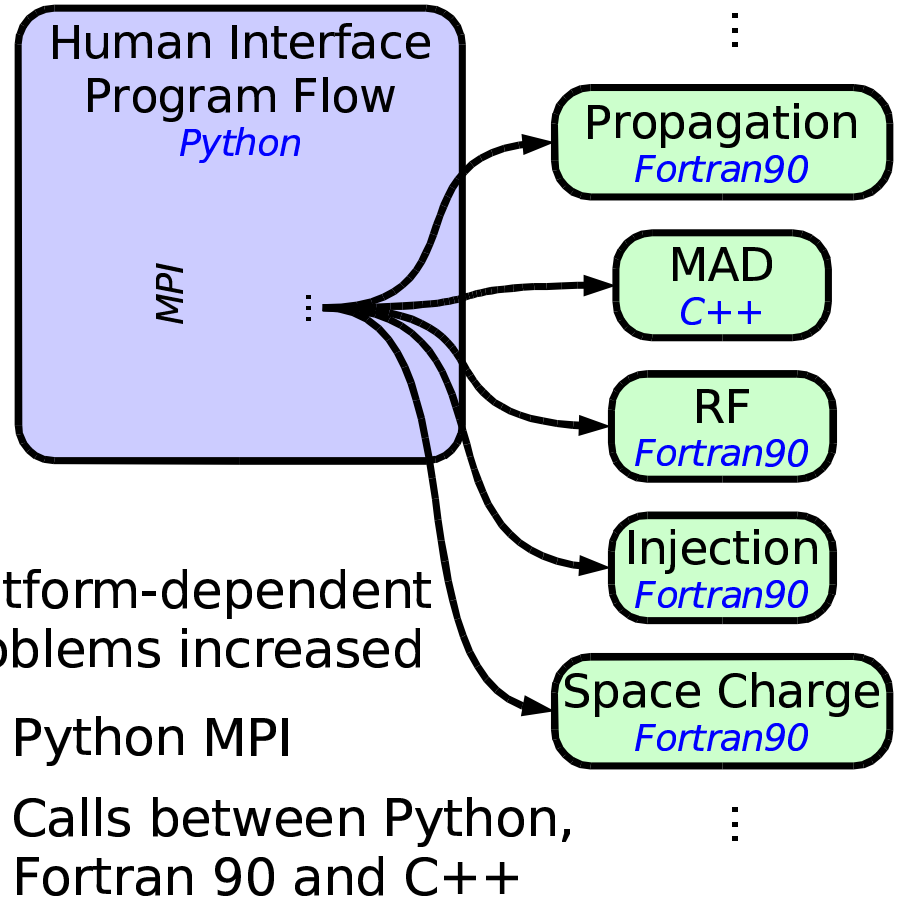
- program flow split into modules
  - still limited by text file
  - still bound by IMPACT object system
  - no additional platform-dependent problems



Συνεργεια

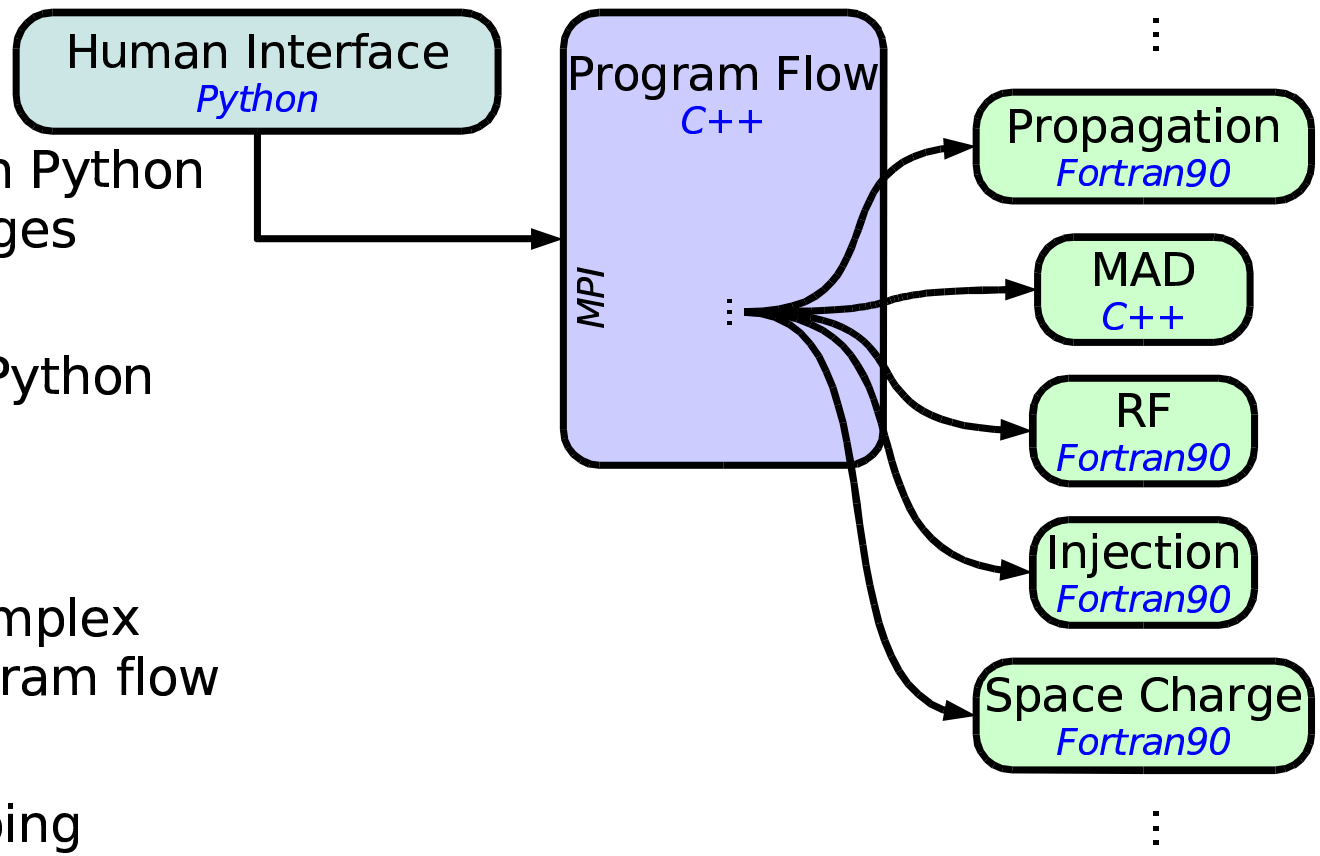
Final Goal

- *pros*
  - greatly enhanced flexibility
    - new physics modules can be trivially added
      - beam-beam
      - etc.
    - dynamic loading possible
  - better interface definition
- *cons*
  - platform-dependent problems increased
    - Python MPI
    - Calls between Python, Fortran 90 and C++



Alternate Plan

- *pros*
  - interface between Python and other languages limited
  - does not rely on Python MPI
- cons
  - requires more complex Python-C++ program flow interface
    - easier than doing same in Fortran90



Συνεργεια

Summary

- Synergia is a framework for combining accelerator physics codes
- The code has been designed to be portable and distributable
- Synergia is extensible
  - 3D Space Charge was first important physics module
  - Other physics modules, including beam-beam, to come