Implementing a ponderomotive guiding center solver: a case study

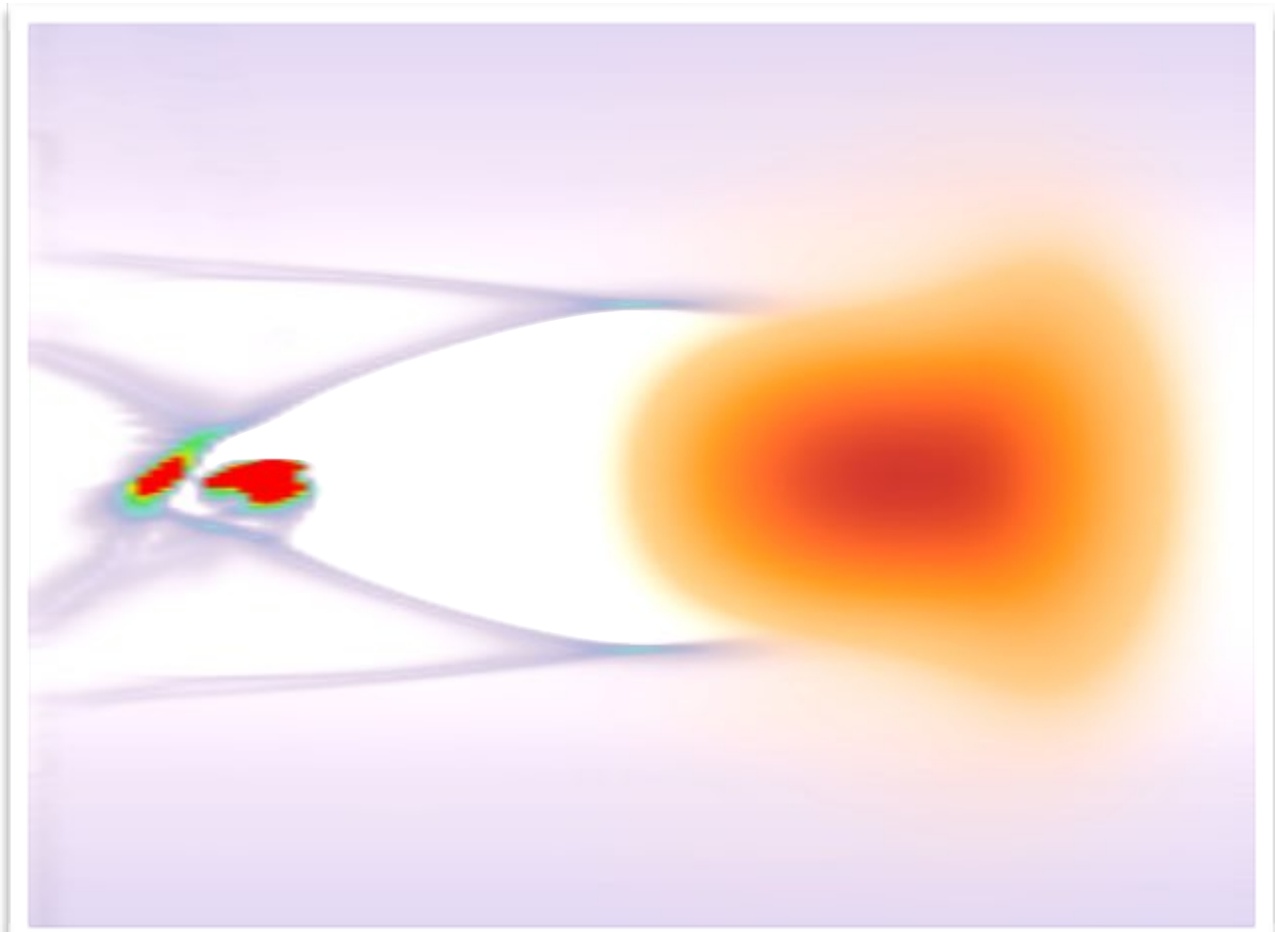
Anton Helm[1]
ahelm@ipfn.tecnico.ulisboa.pt

Ricardo Fonseca[1,2]

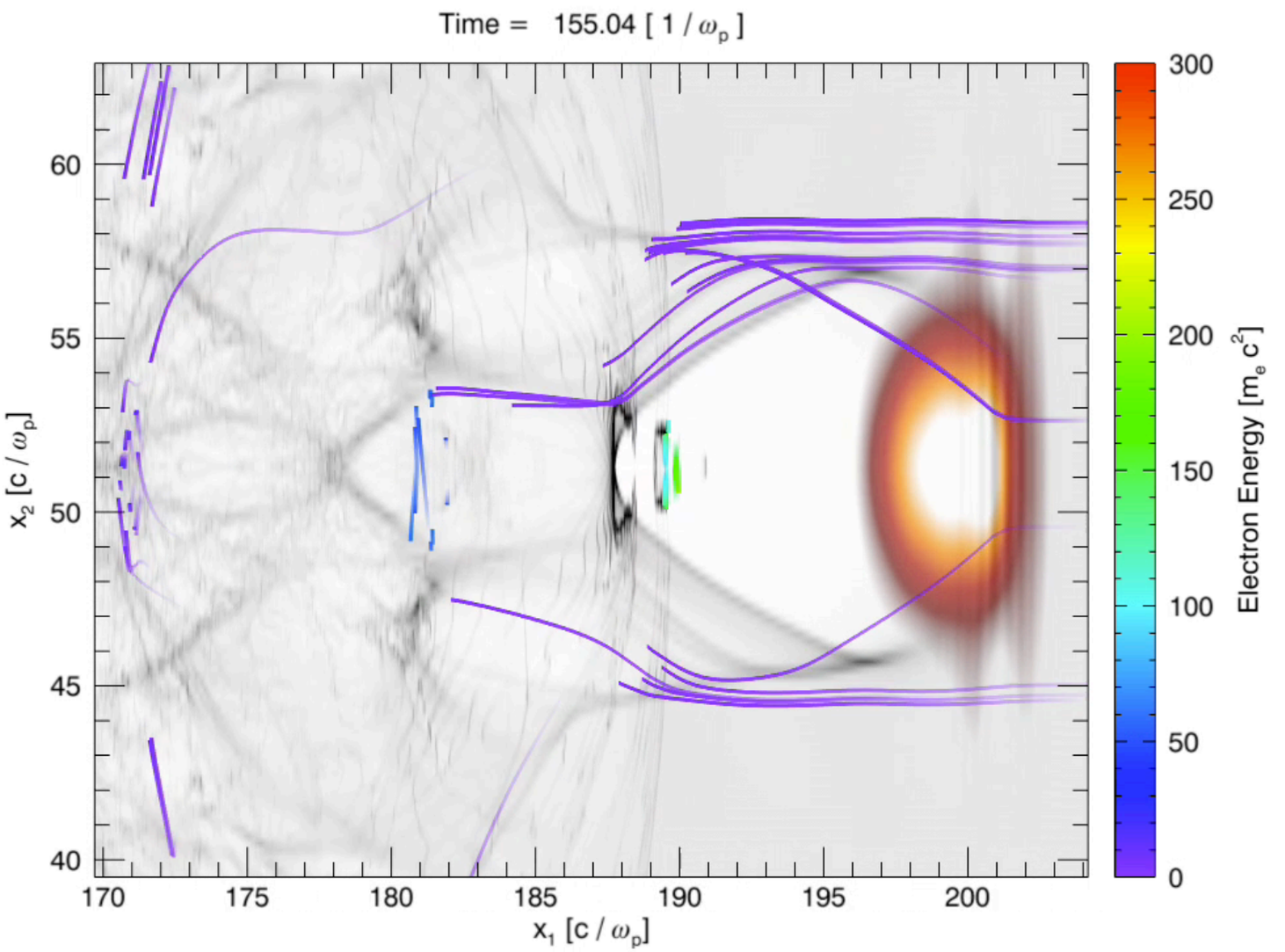
[1] GoLP / Instituto de Plasmas e Fusão Nuclear
Instituto Superior Técnico, Lisbon, Portugal

[2] ISCTE - Instituto Universitário de Lisboa,
Lisbon, Portugal

epp.tecnico.ulisboa.pt || **golp**.tecnico.ulisboa.pt



Scale disparities in LWFA modeling



R. Fonseca, EPS Meeting (2008)

scale disparity in modeling

multi-scale problems

- large disparity of spatial/temporal scales

sample problem: 50 GeV LWFA stage

- λ_0 ~ 1 μm / λ_p ~ 17 μm
- L ~ 1.5 m

computational requirements (moving window)

- ~10^9 grid cells
- ~10^{10} particles
- ~10^6 - 10^7 iterations

requirement for reduced models

Envelope approximation reduces spatial resolution

TÉCNICO LISBOA

particle-in-cell (PIC)

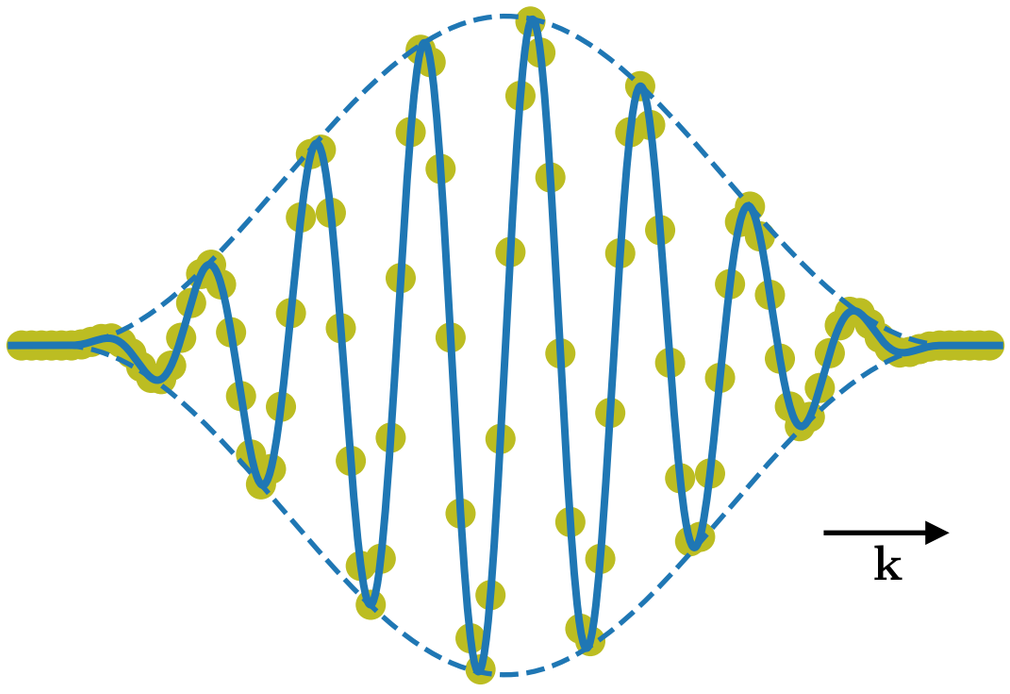
ponderomotive guiding center (PGC)

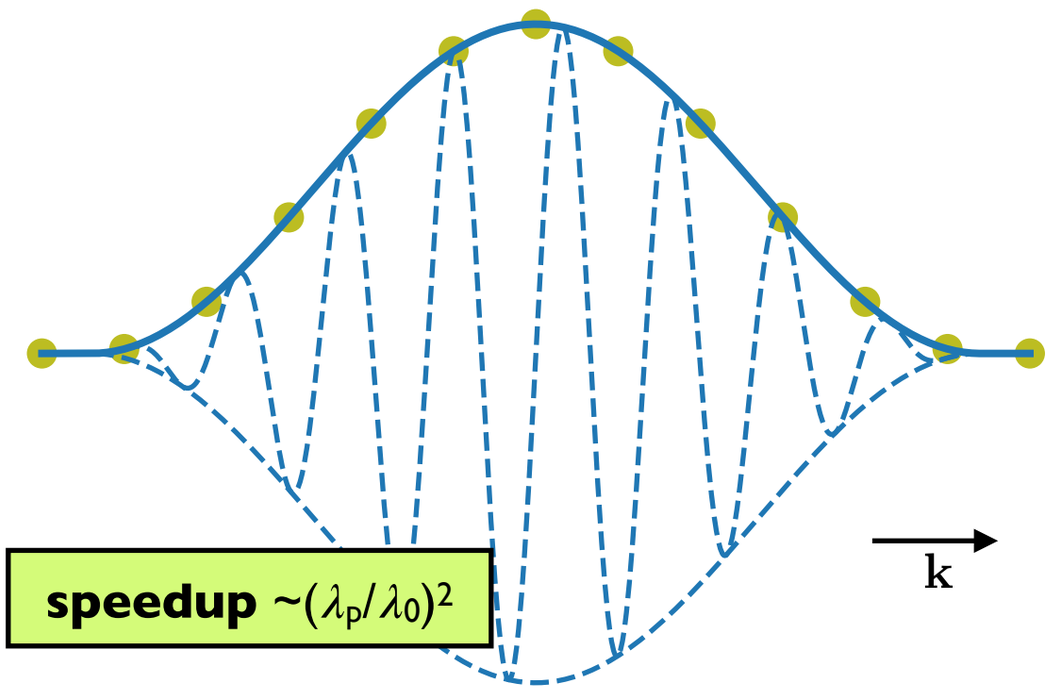
$$\frac{\partial \mathbf{E}}{\partial \tau} = c\nabla \times \mathbf{B} - 4\pi \mathbf{j}$$

$$\frac{\partial \mathbf{B}}{\partial \tau} = -c\nabla \times \mathbf{E}$$

spatial resolution:
laser wavelength

spatial resolution:
plasma skin depth





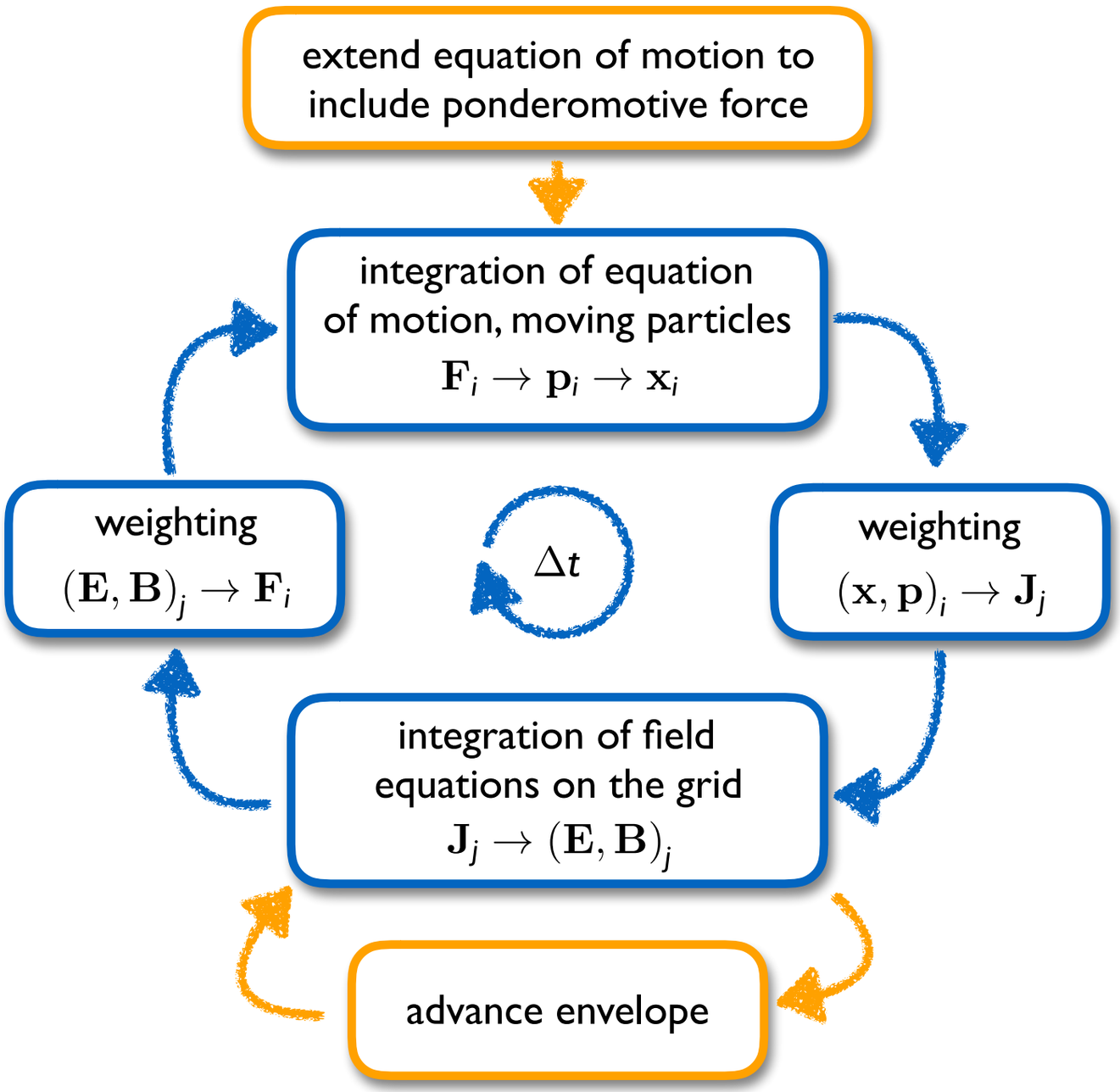
- resolve laser wavelength over propagation distance
- particle advancing is based on Lorentz force

- requires model for laser envelope propagation
- push particles using self consistent plasma fields and ponderomotive force

Incorporation of PGC into PIC cycle

TÉCNICO LISBOA

extended PIC algorithm



PGC extension

- time-averaged equation for laser evolution*,** in a co-moving frame

$$\partial_\tau a = \frac{1}{2i\omega_0}\left[\underbrace{\left(1+\frac{\partial_\xi}{i\omega_0}\right)}_{=:\hat{D}}\underbrace{(\chi a)}_{=:p} + \Delta_T a\right]$$

laser frequency — laser envelope

- particle advancing through ponderomotive force

$$\mathbf{F}_p = -\frac{1}{4}\frac{q^2}{\langle m\rangle}\nabla|a|^2$$

- coupling parameters

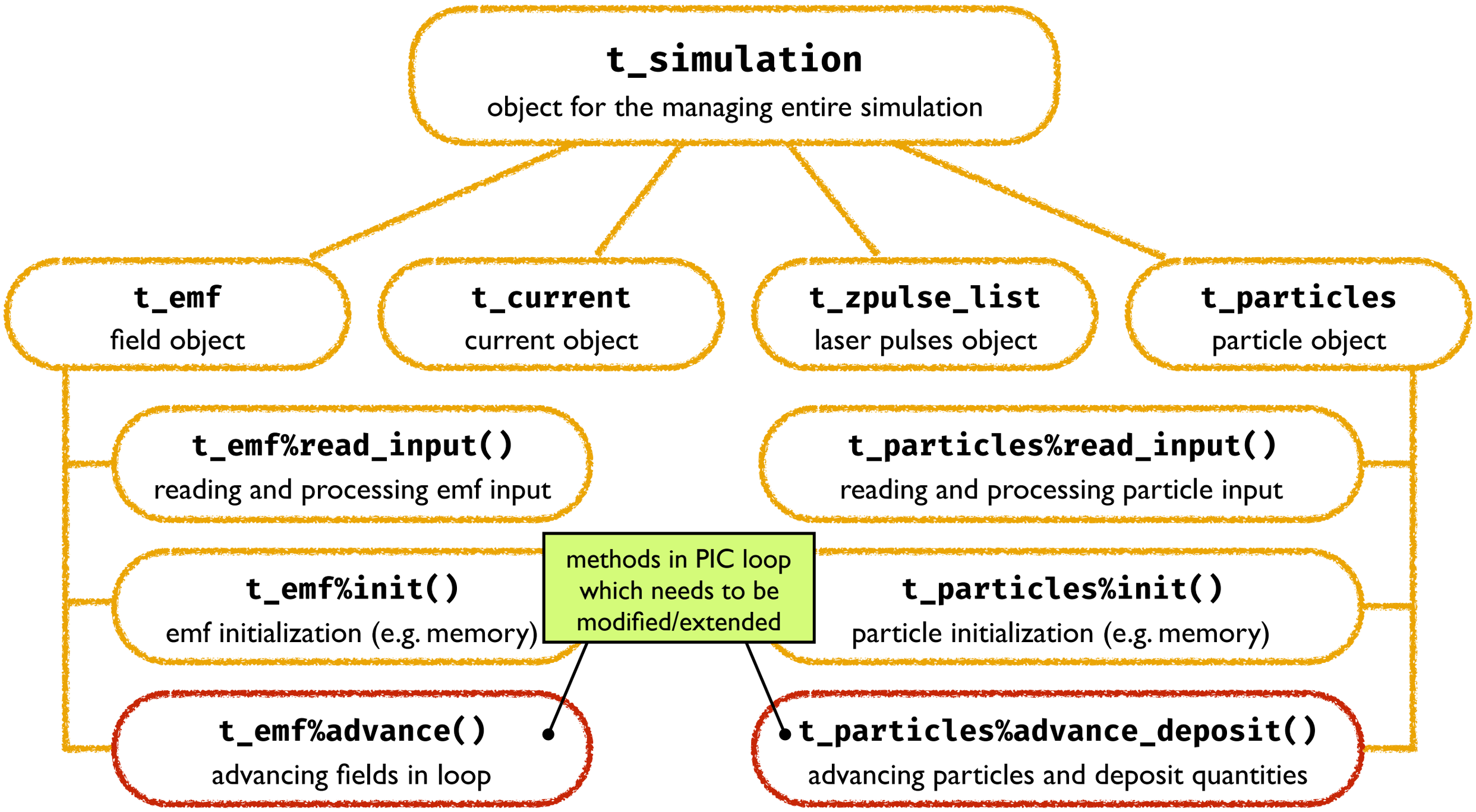
particle deposit

$$\chi = -\sum_i \frac{q_i \rho_i}{\langle m_i\rangle}$$

$$\langle m\rangle = \sqrt{m_0^2 + \mathbf{p}^2 + (q|a|)^2/2}$$

* P. Mora and T. M, Antonsen, PRL 53, R2068 (1996)
** P. Mora and T. M, Antonsen, AIP 4, 217 (1997)

Object tree in OSIRIS



Making simulation object aware of PGC

TÉCNICO LISBOA

```
! content of os-main.f03

use m_simulation_pgc

subroutine initialize( opts, sim )
  class( t_simulation ), pointer :: sim
  call read_sim_options( opts, input_file )

  ! Allocate simulation object based on algorithm
  select case ( opts%algorithm )

    case( p_sim_pgc )
      ! Ponderomotive guiding center simulation
      allocate( t_simulation_pgc :: sim )

  end select

  ! Allocate polymorphic object
  call sim%allocate_objs( )

end subroutine initialize
```

provides "t_simulation_pgc" type specification

pointer: not allocated at initialization and can be modified to change to modified type

reads inputdeck with algorithm specifications

`allocate ( [ type-spec :: ] obj )`

allocate sub-objects of simulation object

PGC extension for OSIRIS

TÉCNICO LISBOA

new simulation object can be used to rewrite other object definition

t_simulation_pgc
class for the managing entire simulation

t_emf
field object

t_current
current object

t_zpulse_list
laser pulses object

t_particles
particle object

t_emf%read_input()
reading and processing emf input

t_emf%init()
emf initialization (e.g. memory)

t_emf%advance()
advancing fields in loop

t_particles%read_input()
reading and processing particle input

t_particles%init()
particle initialization (e.g. memory)

t_particles%advance_deposit()
advancing particles and deposit quantities

Allocating emf and particle object for simulation object

TÉCNICO LISBOA

```
! content of source/pgc/os-simulation-pgc.f03
module m_simulation_pgc

use m_simulation

type, extends( t_simulation ) :: t_simulation_pgc
contains
  procedure :: allocate_objs ⇒ allocate_objs_pgc
end type t_simulation_pgc

contains

subroutine allocate_objs_pgc( sim )

  use m_emf_pgc
  use m_particles_pgc

  class( t_simulation_pgc ), intent(inout) :: sim

  allocate( t_emf_pgc :: sim%emf )
  allocate( t_particles_pgc :: sim%part )

  call sim%t_simulation%allocate_objs()

end subroutine allocate_objs_pgc
end module m_simulation_pgc
```

provides **t_simulation** object

t_simulation_pgc inherits from **t_simulation**

allocate_objs refers to new **allocate_objs_pgc**

includes PGC specific **emf** and **particles** object

sim%emf and **sim%part** are changed objects

for other object allocations use parent object

PGC extension for OSIRIS

TÉCNICO LISBOA

`t_simulation_pgc`
class for the managing entire simulation

new emf object can be used to rewrite **advance()** procedure

new simulation object can be used to rewrite **advance_deposit()**

`t_emf_pgc`
field object

`t_current`
current object

`t_zpulse_list`
laser pulses object

`t_particles_pgc`
particle object

`t_emf%read_input()`
reading and processing emf input

`t_emf%init()`
emf initialization (e.g. memory)

`t_emf%advance()`
advancing fields in loop

`t_particles%read_input()`
reading and processing particle input

`t_particles%init()`
particle initialization (e.g. memory)

`t_particles%advance_deposit()`
advancing particles and deposit quantities

Extending emf object for advancing the envelope

```
! content of source/pgc/os-emf-pgc.f03
module m_emf_pgc

public :: t_emf

type, extends( t_emf ) :: t_emf_pgc
  ! pgc specific variables
contains
  procedure :: read_input ⇒ read_input_pgc
  procedure :: init       ⇒ init_pgc
  procedure :: advance    ⇒ advance_pgc
  procedure :: report     ⇒ report_diag_pgc
  procedure :: cleanup    ⇒ cleanup_pgc
  procedure :: write_checkpoint ⇒ \
               write_checkpoint_pgc
  procedure :: allocate_objs ⇒ allocate_objs_pgc
  ! local methods
  procedure :: reset_chi ⇒ reset_chi_pgc
end type t_emf_pgc


contains
! specific subroutines, e.g. 'advance_pgc'
end module m_emf_pgc
```

IBM XL compilers: has to be explicitly made public to access superclass

repoint to specific routines required by PGC

class specific method to be called by species

write routines here to be used by your solver

Advancing the fields and advancing the envelope

```
! content of source/pgc/os-emf-pgc.f03
subroutine advance_pgc(this, jay, charge, dt, no_co,
coordinates)

  class( t_emf_pgc ), intent( inout )  ::  this
  class( t_current ),   intent(in) :: jay
  type( t_vdf ),   pointer :: charge
  real(p_double),  intent(in) :: dt
  type( t_node_conf ), intent(in) :: no_co
  integer , intent(in) :: coordinates

  ! Advance the EMF fields - fields at new time step
  call this%t_emf%advance(jay, charge, dt, no_co,
coordinates)

  ! solving envelope equation
  ! call advance_envelope(this)

end subroutine advance_pgc
```

same parameters as **emf%advance** (will be called from the **sim** object)

passed object requires to be the polymorphic object

by accessing "advance" routine of superclass the advance of superclass emf is triggered (similar for input, …)

advance envelope

Reset chi before pushing particles

```
! content of source/pgc/os-particles-pgc.f03
module m_particles_pgc
type, extends( t_particles ) :: t_particles_pgc
  ! no additional class members
contains
  procedure :: advance_deposit ⇒ advance_deposit_pgc
  procedure :: allocate_objs ⇒ allocate_objs_part_pgc
end type t_particles_pgc
contains
subroutine advance_deposit_pgc(this, emf, jay, tstep, t,
no_co, options)
  class( t_particles_pgc ), intent(inout) :: this
  class( t_emf ), intent( inout )  ::  emf

  select type( emf )
  class is ( t_emf_pgc )
    call emf%reset_chi()
  class default
    call abort_program( p_err_invalid )
  end select

  ! call superclass pusher
  call this%t_particles%advance_deposit(emf, jay, tstep,
t, no_co, options)

end subroutine advance_deposit_pgc
end module m_particles_pgc
```

reset chi quantity before advancing and check if **emf** is type of **t_emf_pgc** as it is defined as general **t_emf**

call **advance_deposit** of superclass

but it is not using the ponderomotive force, because using advance deposit of superclass?

Allocate new species object to be used for species push

TÉCNICO LISBOA

```
! content of source/pgc/os-particles-pgc.f03
subroutine allocate_objs_part_pgc(this, num_species, num_cathode, num_neutral,
num_neutral_mov_ions)

  class(t_particles_pgc), intent(inout) :: this

  integer, intent(in) :: num_species, num_cathode
  integer, intent(in) :: num_neutral
  integer, intent(in) :: num_neutral_mov_ions

  allocate(t_species_pgc :: this%species(total_species))

end subroutine allocate_objs_part_pgc
```

each species will be of type **t_species_pgc** - pushing of particles can be specified by modifying **t_species_pgc**

```
! content of source/particles/os-particles.f03
subroutine advance_deposit_particles( this, emf, jay, tstep, t, no_co, options)

  class( t_particles ), intent(inout) :: this
  class( t_emf ), intent( inout )  ::  emf
  class( t_current ), intent(inout) :: jay

  do i=1, this%num_species
    call this%species(i)%push(emf, jay%pf(tid+1), t, tstep, tid, nt, options)
  enddo

end subroutine advance_deposit_particles
```

general OSIRIS objects which might have been mutated/polymorphed

uses push for each species - for PGC, species is of type **t_species_pgc**

Conclusion

Ponderomotive guiding center solver

reduced solver for laser wakefield acceleration

advancing fields on plasma skin depth, envelope model for laser and ponderomotive force

OSIRIS main objects for extension

objects to modify to implement PGC in OSIRIS

step-by-step show case of implantation in OSIRIS

Objects in OSIRIS can be modified using object-oriented approach

type extension and redefining type procedures

customization through tree traversing