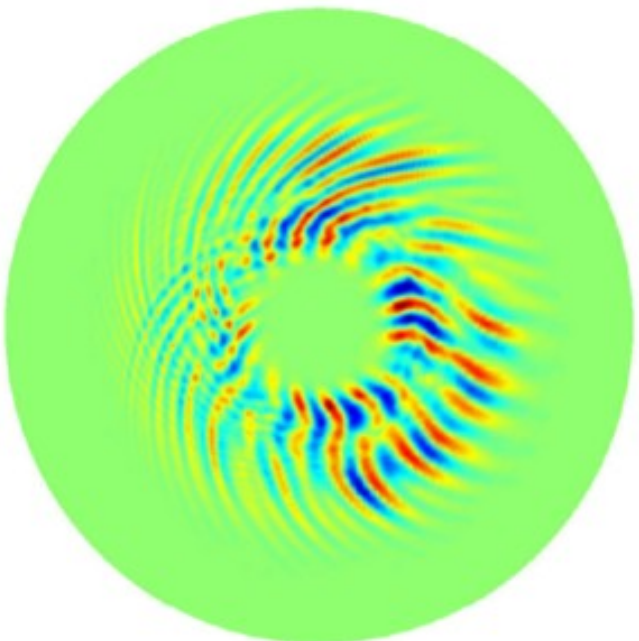
Consolidating ORB5 Bsplines with the SPClibs library

Huw Leggate

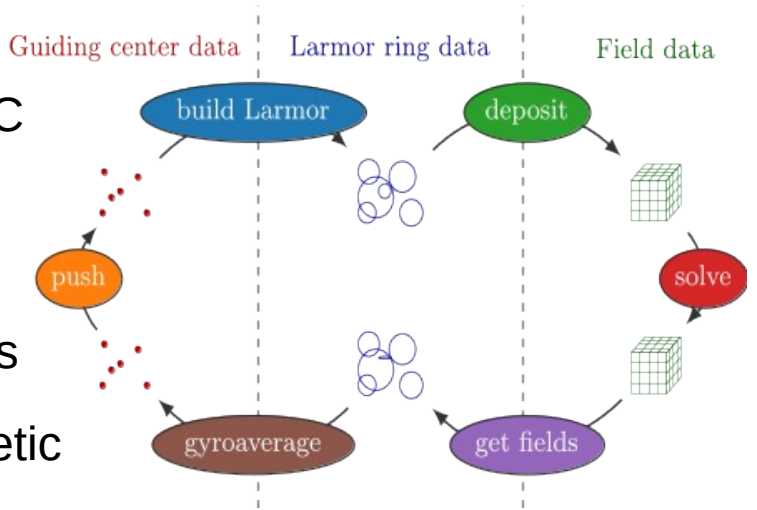
Dublin City University/Advanced Computing Hub - Garching

ORB5

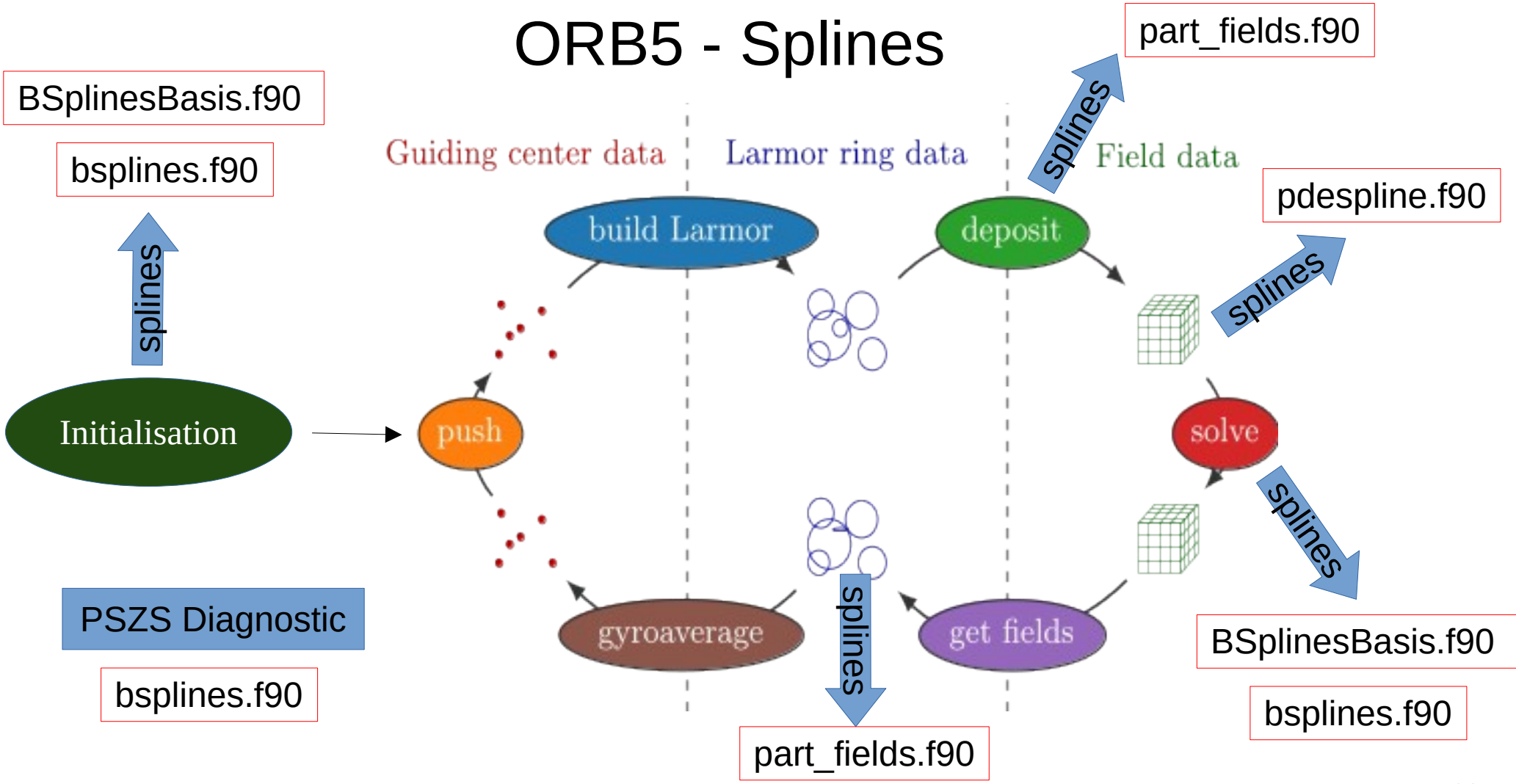
"ORB5: a global electromagnetic gyrokinetic code using the PIC approach in toroidal geometry" [for details, see Lanti 2020]



- δ-f modified distribution function discretized with PIC
- Fields solved using finite elements
- Fields stored using Bsplines
- Global kinetic electromagnetic (EM) simulations



ORB5 - Splines



Consolidation into bsplines_int

- Code contains splines from three different libraries
- Goal to base all spline computations on a single library
  - spclibs/bsplines.f90
- New module **bsplines_int.F90**

```
call set_spline((/nidbas,nidbas,nidbas/), (/nq,nq,nq/), &
                gridx, gridy, gridz, spl2d1d, &
                period=(/.True.,.True.,.True./), &
                nlppform=.True., nlequid=(/.True.,.True.,.True./))
```

Spline Deposition

- Charge and current are deposited using B-Splines, cubic by default
  - Spline weights (Basis functions) are given by

$$W_0(x) = (1-x)^3/6$$

$$W_1(x) = (3x^3 - 6x^2 + 4)/6$$

$$W_2(x) = (-3x^3 + 3x^2 + 3x + 1)/6$$

$$W_3(x) = x^3/6$$

$$S_t(x) = \sum_i c_i B_{i,k}(x)$$

  - Where *x* is the normalised particle location relative to the cell
  - So that

$$\rho(i) = \rho(i) + W_{(i)}\rho_p$$

Spline Deposition

- Charge and current are deposited using B-Splines, cubic by default
  - Spline weights (Basis functions) are given by

$$W_0(x) = (1-x)^3/6$$

$$W_1(x) = 2/3 + x^2(x/2-1)$$

$$W_2(x) = 1/6 + 0.5x(1 + x \times (1-x))$$

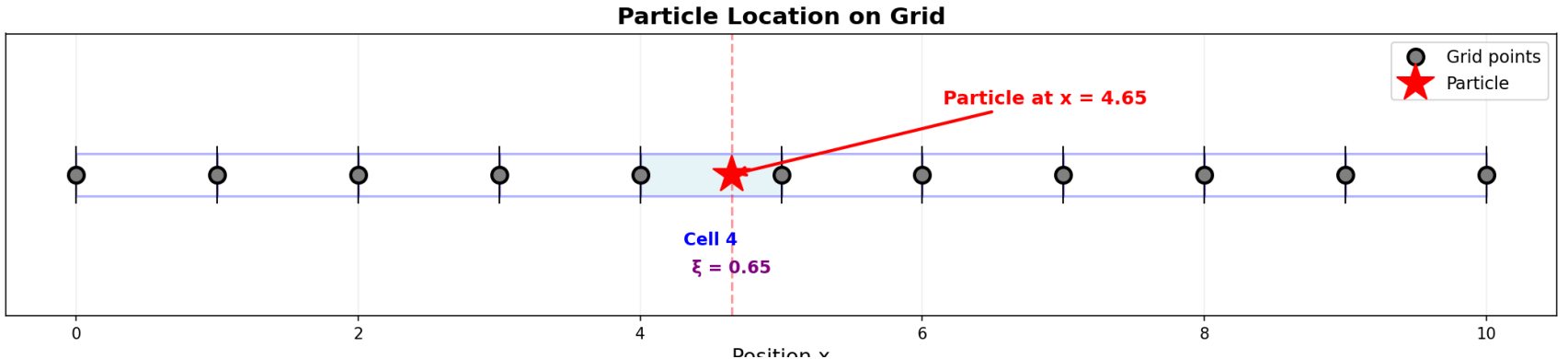
$$W_3(x) = x^3/6$$

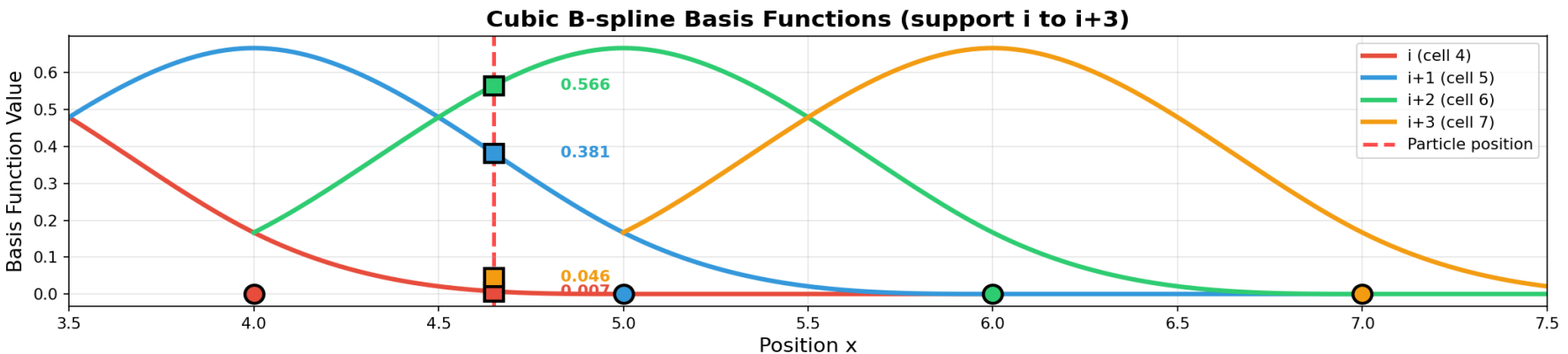
$$S_t(x) = \sum_i c_i B_{i,k}(x)$$

  - Where *x* is the normalised particle location relative to the cell
  - So that

$$\rho(i) = \rho(i) + W_{(i)}\rho_p$$

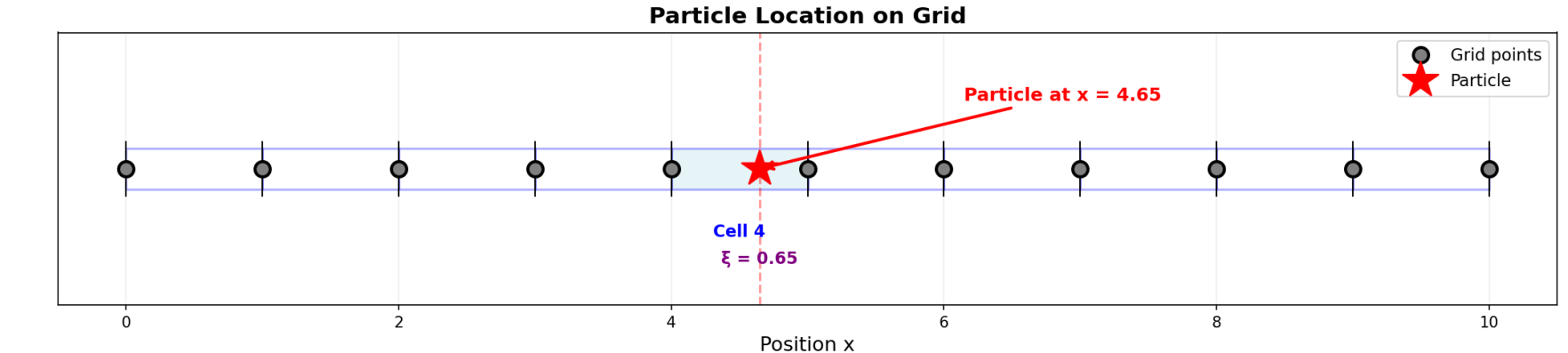
Spline Deposition

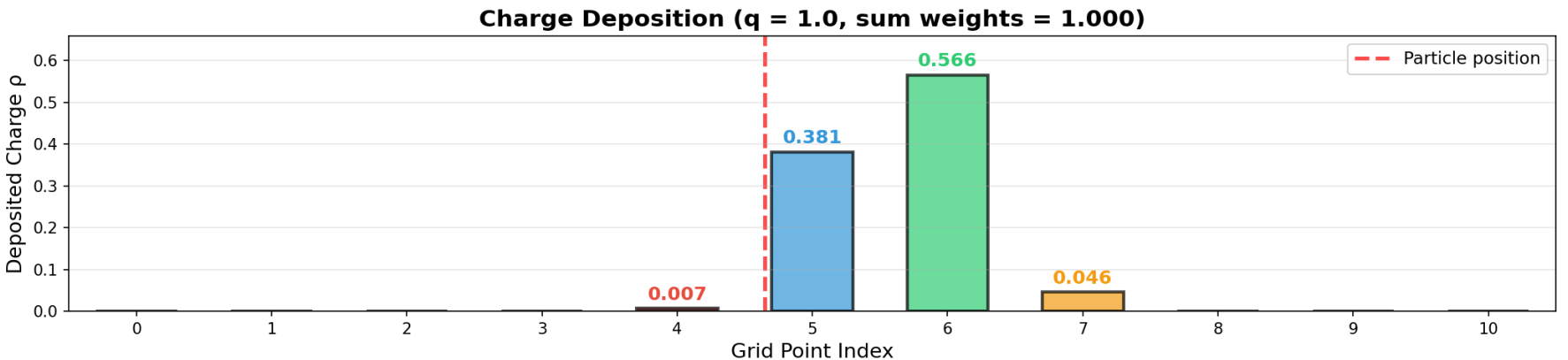




Generated by Claude AI

Spline Deposition



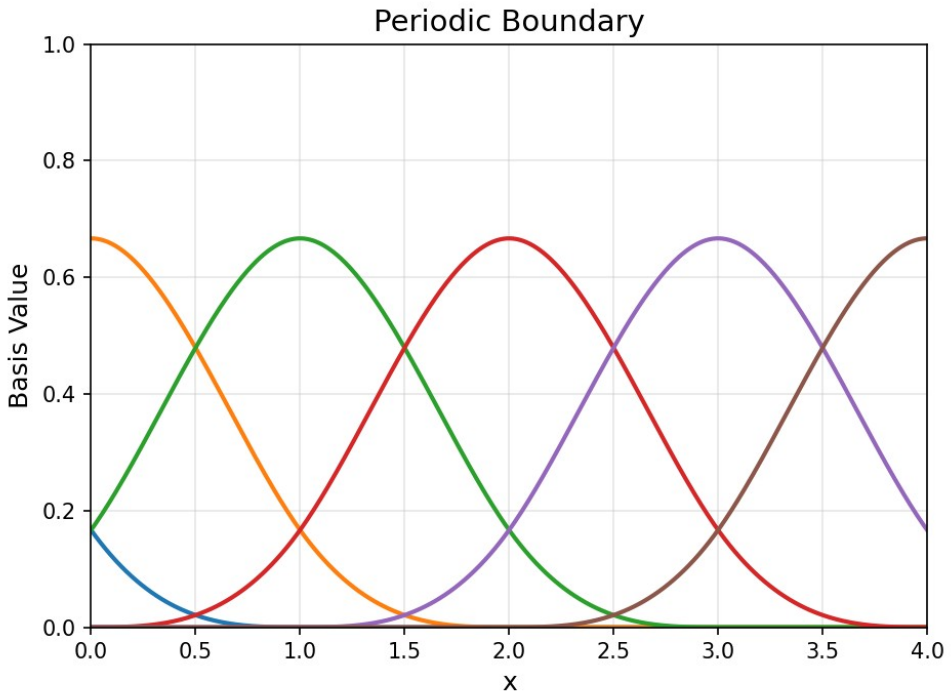


Generated by Claude AI

B-Spline Boundaries

- At fixed (non-periodic) boundaries splines are clamped

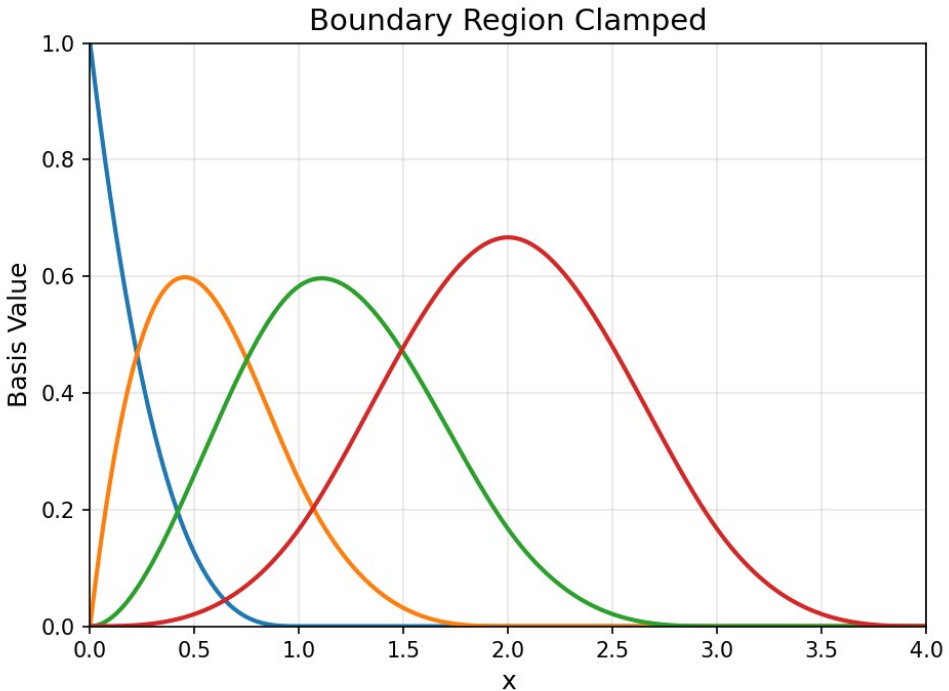
$$[0,0,0,0,1,2,3,4,5,6,7,8,8,8,8]$$



B-Spline Boundaries

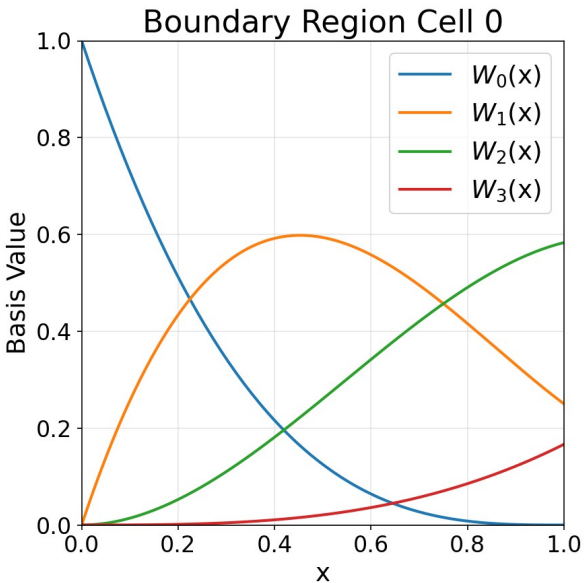
- At fixed (non-periodic) boundaries splines are clamped

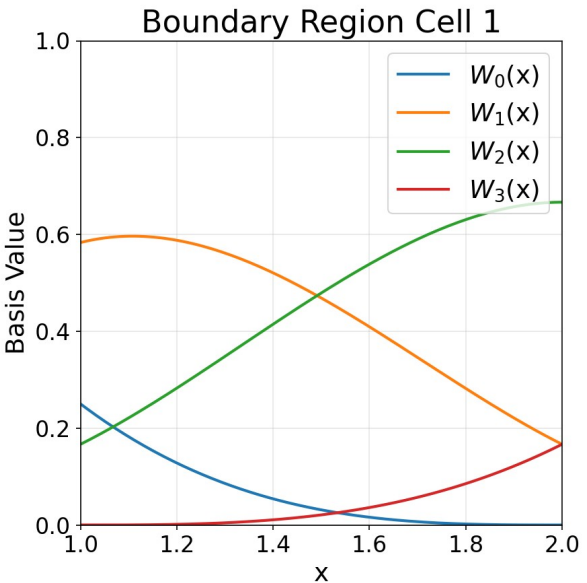
$$[0,0,0,0,1,2,3,4,5,6,7,8,8,8,8]$$

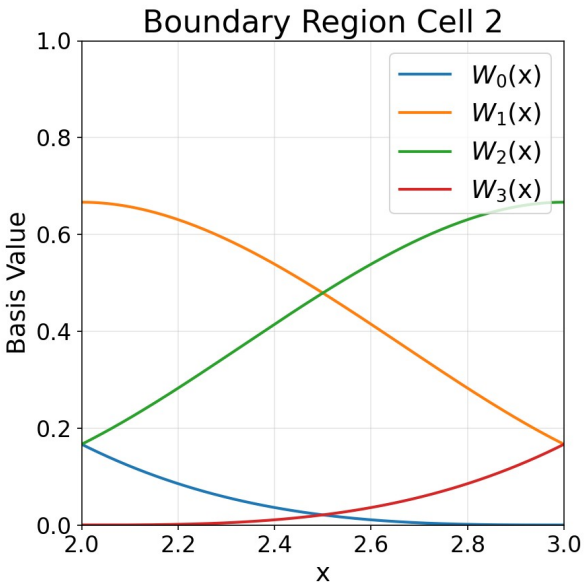


B-Spline Boundaries

- Looking at individual cells

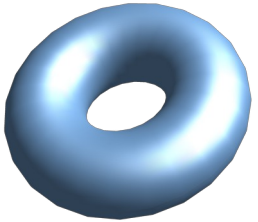






Correction for Periodic Splines

- ORB5 dimensions have 1 fixed and 2 periodic boundaries
  - However, deposition does not account for this
  - Deposited spline values are treated as periodic
  - Requires rebase of 2D rhs
  - For simplicity the 3D splines are created **periodic** in all dimensions



$$rhs = \begin{pmatrix} 1/c_1 & 0 & 0 & \cdots & 0 \\ c'_2 & 1 & 0 & \cdots & 0 \\ c'_3 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ c'_p & 0 & \cdots & 0 & 1 \end{pmatrix} \times \begin{pmatrix} v & v & v & \cdots & v \\ v & v & v & \cdots & v \\ v & v & v & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & v \\ v & v & \cdots & v & v \end{pmatrix}$$

Where c are the spline basis functions

bsplines_int

part

spline_bas

spclibs

bsplines
matrix
...

ORB5

solver
deposition
gyroaverage
...

pde_splines

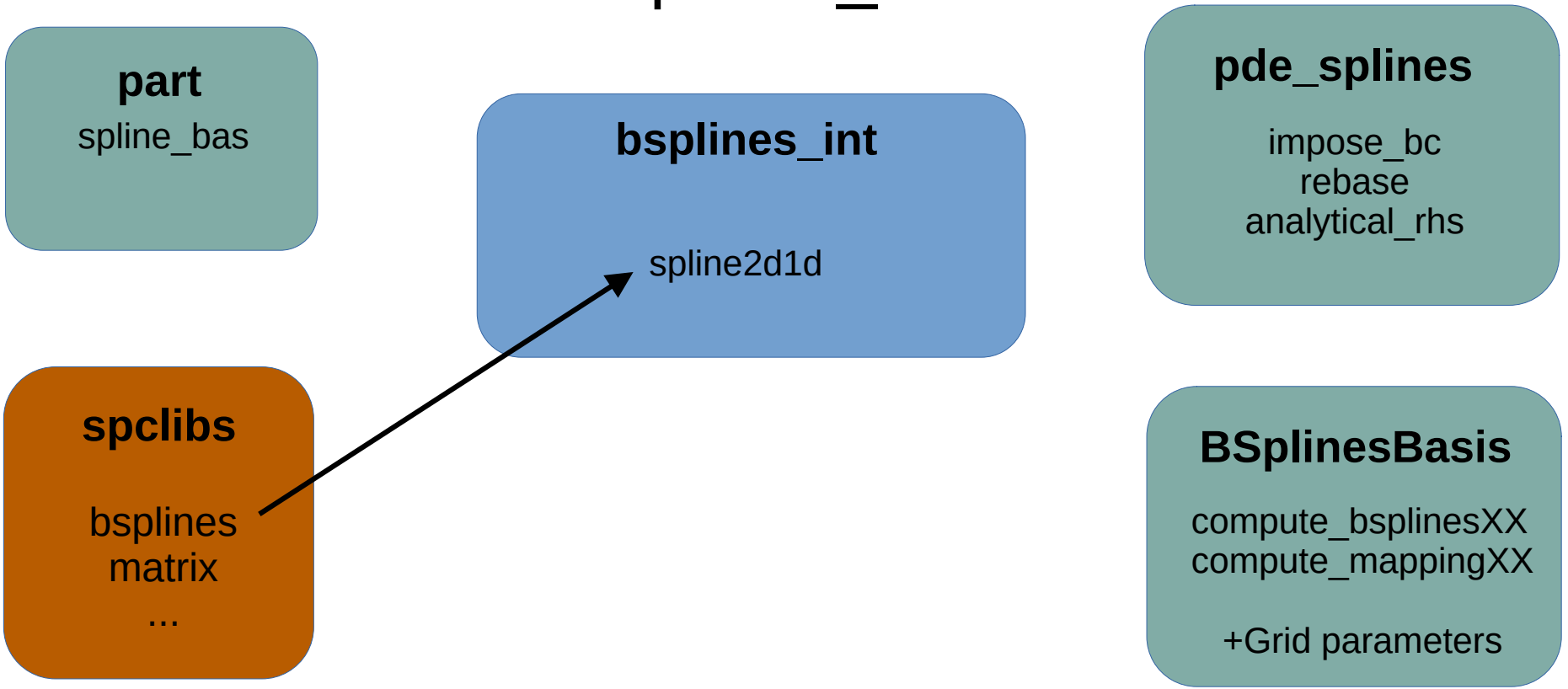
impose_bc
rebase
analytical_rhs

BSplinesBasis

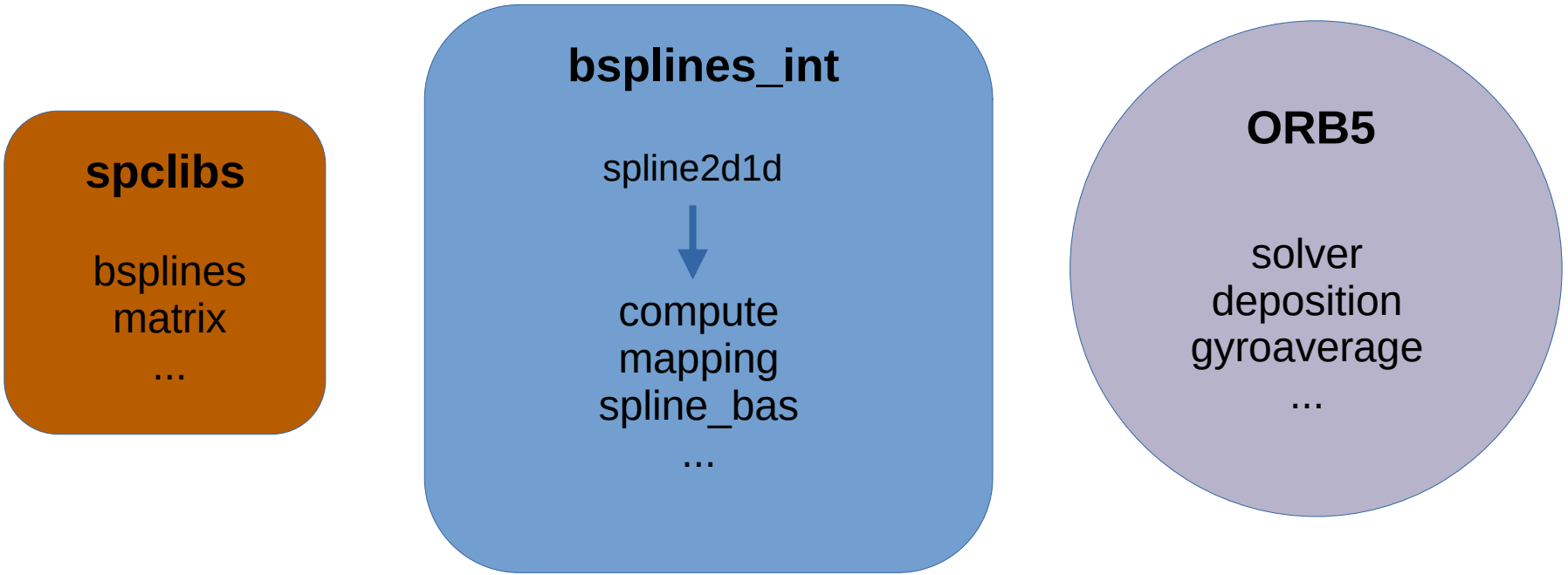
compute_bsplinesXX
compute_mappingXX

+Grid parameters

bsplines_int



bsplines_int



bsplines_int

- Interface to **spclibs bsplines** module
- Public routines imported from **pdespline** module
  - impose_bc, rebase, analytical_rhs
- Public routines imported from **BsplinesBasis**
  - compute_bsplines_XX+fft, compute_mapping_XX
- Public spline basis functions moved from **part_fields**
- Public grid parameters imported from **BsplinesBasis** used in the solver
- Spline routines now obtain values from **bsplines spline2d1d** type

Phase Space Zonal Structure Diagnostic

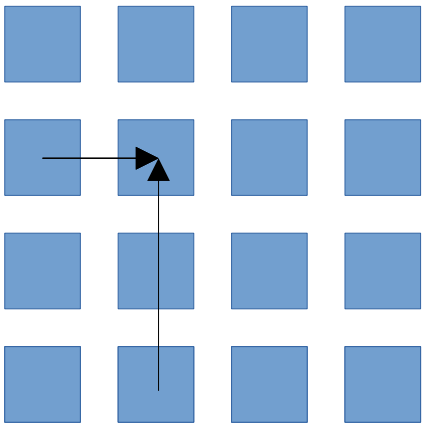
- Allow detailed treatment of energetic particles
- Defined as angle average of gyrocenter distribution function
- Transform each dimensions into either:
  - Kinetic Energy, Toroidal Angular Momentum or Magnetic Moment
  - Choice of which made during initialisation
  - All 3 dimensions now have fixed boundaries
  - Property is deposited on a PSZS spline grid
  - Carried out using SPCLIB Bspline routines

PSZS splines

- All three dimensions are bounded
  - Stencil as applied in rebase will not work
  - Need to account for boundaries during deposition
- Bsplines **basfun** routines are not GPU enabled
  - Need similar approach to local spline basis functions

```
pure function splines_bas3(x)
    !$acc routine seq
    real, intent(in) :: x
    real, dimension(0:3) :: splines_bas3
    splines_bas3(0) = (1.-x)**3 / 6.
    splines_bas3(1) = 2./3. + x**2*(x/2. - 1.)
    splines_bas3(2) = 1./6. + .5*x*(1.+x*(1.-x))
    splines_bas3(3) = x**3 / 6.
end function splines_bas3
```

Inner knot affected by both faces



Cannot retrieve original values

PSZS splines

- Solution - Store polynomial coefficients at left boundary
  - Now can use same approach as bsplines.F90

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$

$$x \in [0,1]$$

```
pure function splines_ppform_bas3(x,ppform)
    !$acc routine seq
    real, intent(in) :: x
    real, intent(in) :: ppform(4,4)
    real, dimension(0:3) :: splines_ppform_bas3
    splines_ppform_bas3(0) = ppform(1,1) + x*(ppform(2,1)+x*(ppform(3,1)+x*ppform(4,1)))
    splines_ppform_bas3(1) = ppform(1,2) + x*(ppform(2,2)+x*(ppform(3,2)+x*ppform(4,2)))
    splines_ppform_bas3(2) = ppform(1,3) + x*(ppform(2,3)+x*(ppform(3,3)+x*ppform(4,3)))
    splines_ppform_bas3(3) = 1.0d0 - sum(splines_ppform_bas3(0:2))
end function splines_ppform_bas3
```

- Could be used to remove rebase requirement

PSZS splines

- Deposition GPU enabled
  - Different metrics available for each dimension
  - Deposition routine split between position calculation and deposition
  - Different possible metrics in each dimension are computed during initialisation and stored as an integer in **pszs_coord**
  - **pszs** work arrays declared in module header contain storage for **position** and **weight**

```
subroutine pszs_deposit_bsplines(pszs_, …)
    call pszs_calc_position(pszs_,…)
        !$acc parallell loop gang worker vector...present(pos...
    call pszs_deposit(pszs_,…,)
        !$acc parallell loop gang worker vector...present(pos...
end subroutine pszs_deposit_bsplines
```

Summary

- All spline routines now contained in **bsplines_int** module
- **BsplinesBasis** and **pdespline** removed
  - All routines added to **bsplines_int** – minimal code changes elsewhere
- Now uses **spclibs bsplines** module
- Splines stored in **spline2d1d** derived type
- Splines are stored with **PERIODIC** boundary conditions
- New **splines_ppform** routines can account for boundary during deposition while running on the GPU