EFDA
EUROPEAN FUSION DEVELOPMENT AGREEMENT
Task Force
INTEGRATED TOKAMAK MODELLING

IMP5 / ACT4
RF Monte Carlo library for orbit following codes

T Johnson, A Salmi, L-G Eriksson, J Höök, T. Hellsten, R Dumont, M Schneider, D Zarzoso

Background

Lack of Fokker-Planck code in community able to model fast ions from ICRF, NBI and fusion reactions, including wide guiding centre orbits. Priority support was allocated to develop such a code. But what type of code should be built?

- Monte Carlo / Finite difference / Finite element
- Orbit averaged / Orbit following

Decision: Extend available codes by adding ICRF

- SPOT & ASCOT: Orbit following Monte Carlo – add ICRF ← This poster!
- FIDIT: Orbit averaged finite difference – add ICRF

This poster concerns:

- Progress on generic RF operator; *RFOF - **R**adio **F**requency Monte Carlo Library for **O**rbit **F**ollowing Codes*
- First results from coupling with ASCOT

Traditional disadvantage with orbit following Monte Carlo for RF:

- Quasilinear evolution on slowing down time scales (~1s), while orbit tracing resolves fractions of bounce time (10^{-5}s)
- Orbit averaged codes can take up to 1000 longer time steps

However, a new Monte Carlo operator [L.-G. Eriksson *et al*, Phys. Plasmas 12, 072524 (2005)] will allow RFOF to work with accelerated orbit time integration, i.e. let one orbit represent N_{ACC} orbits, as is commonly done in NBI codes like NUBEAM/ASCOT/SPOT. This will make CPU times for orbit following / orbit averaged codes of the same order.

Integration of RF operator in orbit following code

ASCOT and SPOT solve Fokker-Planck equation by separate *orbit tracing / collision / RF interaction*:

$$\frac{Df}{Dt} = C(f) + Q_{RF}(f)$$

$$\frac{Df}{Dt} = 0 \quad + \quad \frac{\partial f}{\partial t} = C(f) \quad + \quad \frac{\partial f}{\partial t} = Q_{RF}(f)$$

Orbit tracing ***Collisions*** ***RF acceleration***

These codes assume time step to be small and operators commute and can be added sequentially

This allows RFOF to be independent of both orbit tracing and collisions

RFOF is being written to be a generic library; should be possible to couple to any orbit code

INPUT:

- Marker/particle properties (**r,v**); set at initialisation to be pointer to marker, i.e. set only once per marker!
- RF-wave field; only pointers passed between routines, RF-wave stored as ITM cpo Waves
- Magnetic field; provided to RFOF through interface to subroutine, i.e. wrapper needed to interpolation routines inside orbit code

OUTPUT:

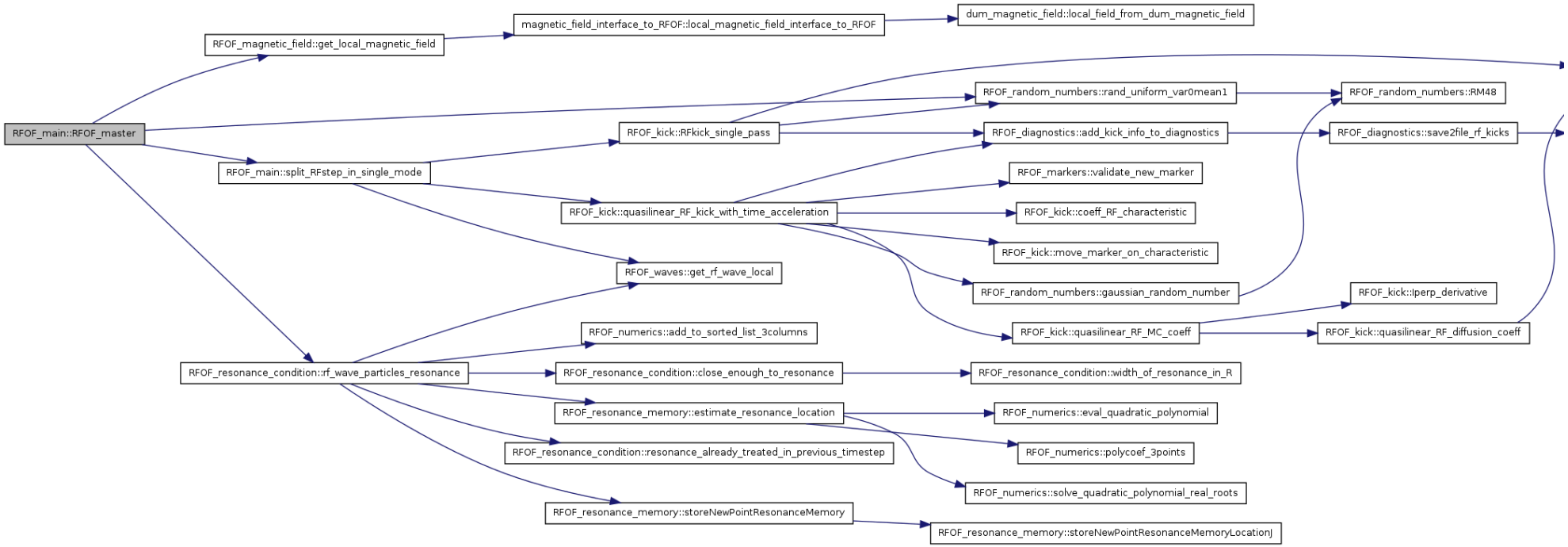
- Updated marker properties, after acceleration by RF wave
- Prediction of the time of future resonances; allow orbit traces to adjust time step to end at the resonance
  - In case the orbit step was too long and marker crossed resonance without RF kick; then return expected time of resonance to allow orbit solver to redo time step

Workflow of RFOF

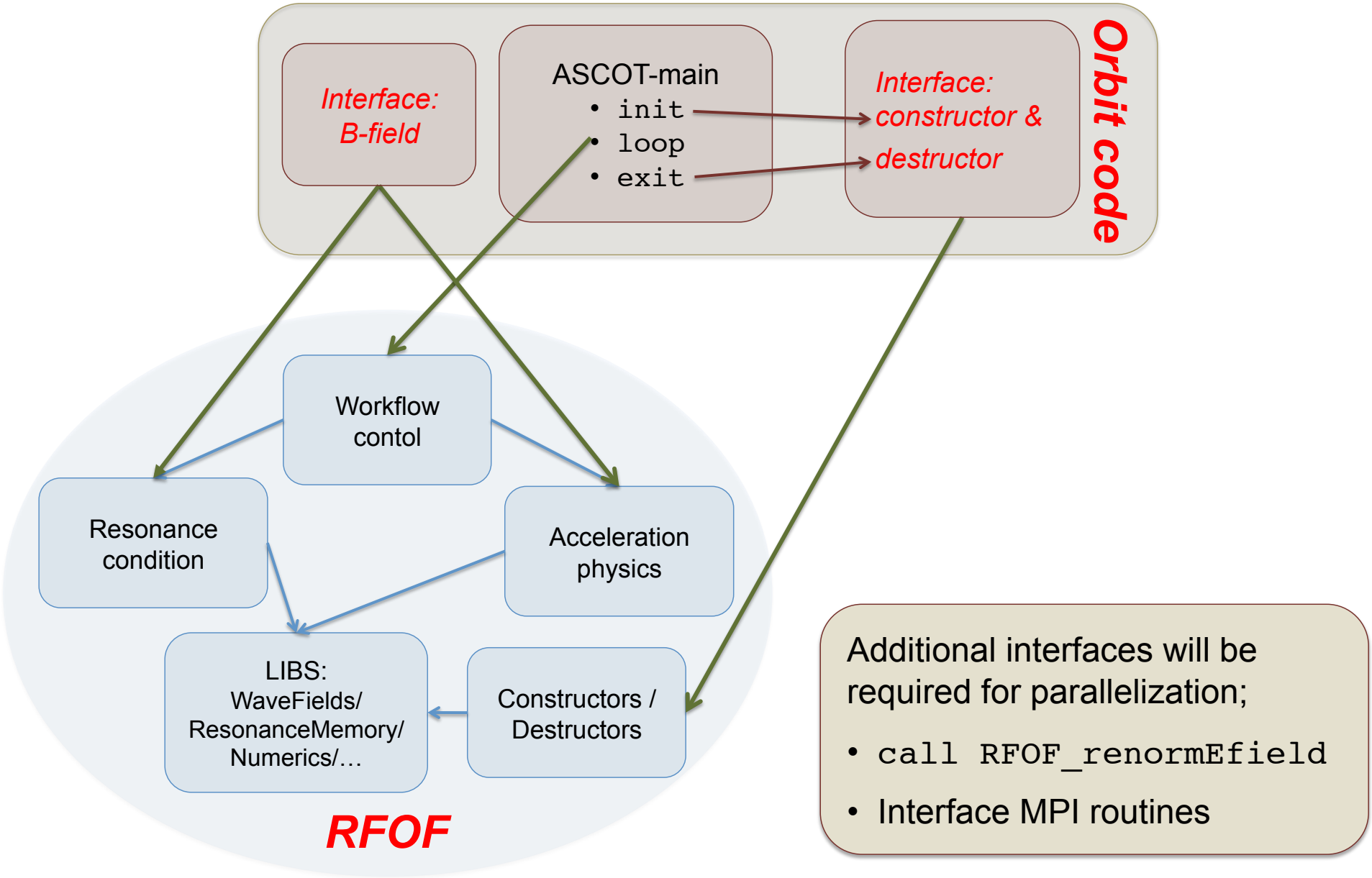
- Initialise and allocate data structures for RFOF
- In orbit code: after each step on orbit call RFOF
  1. RFOF evaluates resonance condition
  2. FOR ”waves resonant with particle”
     - ✓ Give RF kick
  3. ENDFOR
  4. RETURN:
     - ✓ MARKER: Updated particle
     - ✓ TIME_AT_RESONANCE (prediction: next resonance / missed resonance)
- Output diagnostics & deallocate memory

CODE STATUS (from documentation)

- The code can find the resonance position for non-accelerated schemes; i.e. judge if particle is in resonance and predict the time and position of future and past resonance. For time-acceleration we have to separate orbit-relative time and simulation time. Orbit code will provide simulation time, while derivatives are on orbit-relative time.
- The code can give RF Monte Carlo kicks, both for accelerated and non-accelerated orbit integration. The accelerated scheme is particularly badly tested.
- The interpolation of the RF wave field is still not implemented; as a temporary solution the wave field is always taken from coordinates with index (1,1).
- The renormalisation of the electric field to keep RF power constant is not implemented. This will require communication between MPI-nodes.



Design and coupling to orbit code



Additional interfaces will be required for parallelization;

- `call RFOF_renormEfield`
- Interface MPI routines

Coupling to ASCOT

RFOF now runs in ASCOT – thanks to Antti Salmi from ASCOT team!
Coupling was relatively easy: < 1 week of work

RFOF is generic – can't use ASCOT specific types, instead:

- markers in RFOF are pointers to markers in ASCOT
  (done once; then the same memory is used in RFOF and ASCOT)
- B-field is transferred via interface – wrapper to ASCOT routines
- wave fields transferred using pointers
- NOTE: No global variables in RFOF (except static variables)

Interpolation of wave field still not available
– instead ASCOT calls a dummy-constructor.

Changes to ASCOT:

1. Minor changes to main control routine (as shown to the right)
2. Implement initialise_RFOF and deallocate_RFOF
3. Implement interface for the magnetic field
4. Minor changes to the particle structure
   (a few missing `TARGETS` added)

ASCOT: from MAIN

```
CALL initialise_RFOF(prt(1),marker,mem,RFglobal,RFdiagno)
 ...

!! CALL RFOF module before collisions. Note: this is not
!! included in ASCINT(3) as bookkeeping of 'mem' and 'RFglobal'
!! and 'marker' is required. It seems also that initialisation
!! and time step control through ascint would become cumbersome.

prt%gyrof= prt%chargeSI * magn_bmagn / ( prt%loc%lorenz * prt%m0 )
CALL RFOF_master(prt(1)%time%t+TSTEP, prt(1)%time%t, &
     RFglobal,marker,mem,RFdiagno,mpivar_id, &
     interactionFailedParticleOvershotResonance, &
     timeAtResonance)
! RFOF can predict with high accuracy the time at the resonance.
! Use it to set additional limit on TSTEP.
RFtstep=timeAtResonance-(prt(1)%time%t+TSTEP)
IF (interactionFailedParticleOvershotResonance) THEN
   ! this still needs revision..
   RFtstep=timeAtResonance-prt(1)%time%t
   ERRFLG=50 ! go back to previous step and use TSTEP=RFtstep
   GOTO 1600 ! jump to a place where redirection to 850 occurs
END IF

 ...

CALL store2file_RFOF_cumlative_diagnostics(RFdiagno,RFglobal)
CALL deallocate_RFOF(RFglobal,mem,RFdiagno)
```

ASCOT: from magnetic_field_...

```
MODULE magnetic_field_interface_to_RFOF
CONTAINS
  !> \fn local_magnetic_field_interface_to_RFOF
  !> interface for RFOF to call magnetic field quantities
  SUBROUTINE local_magnetic_field_interface_to_RFOF(R,phi,z, &
       psi,theta,Bmod,F,psi_Estatic,dBmod_dpsi,dF_dpsi,dBmod_dtheta,dF_dtheta)
```

ASCOT: from interface_to_RFOF

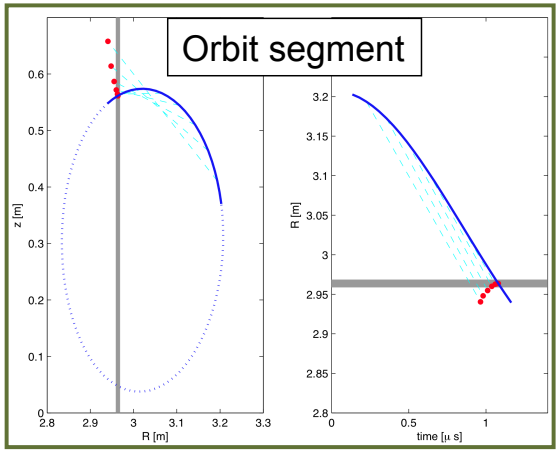
```
MODULE interface_to_RFOF
  IMPLICIT NONE

CONTAINS

  !> subroutine for initialising data structures used by RFOF
  SUBROUTINE initialise_RFOF(prt,marker,mem,RFglobal,RFdiagno)
    ...

  !> subroutine for deallocating data structures used by RFOF
  SUBROUTINE deallocate_RFOF(RFglobal,mem,RFdiagno)
```

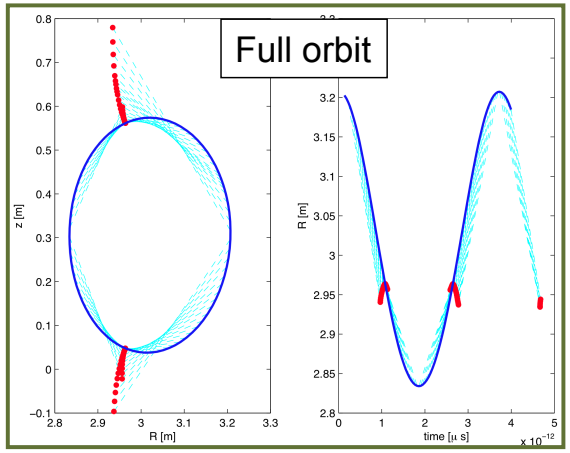
Examples with ASCOT: resonance condition

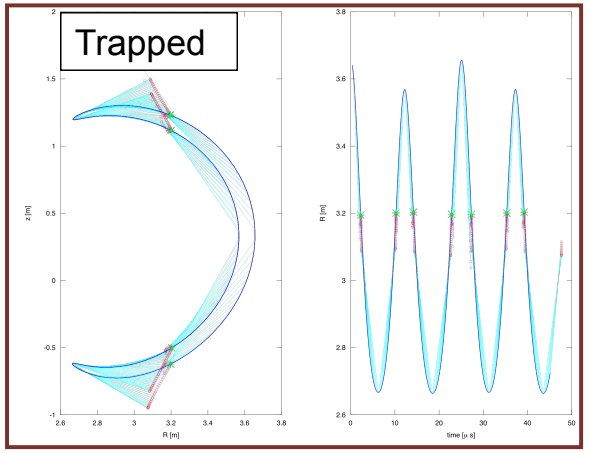
The resonance condition, including the Dopper shift, is $\nu = 0$, where: $\nu \equiv \omega - n\Omega_c - \mathbf{k} \bullet \mathbf{v}$

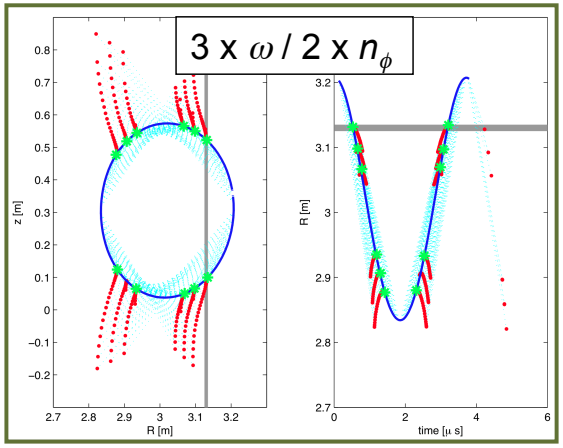
Remember the time history $\nu(t)$ allows one to extrapolate to find t_{res} : $\nu(t_{res}) = 0$.
RFOF calculates t_{res}, R_{res}, and z_{res} ; see below how prediction gets better when we approach resonance. Predictions of t_{res} allow ASCOT to adjust the time step to stop at the resonance.

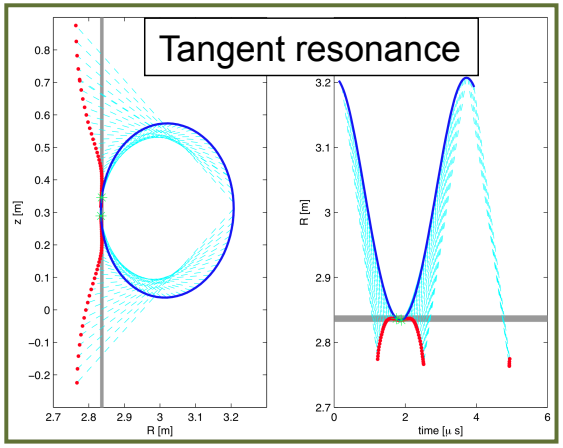
Red dots: predictions of $R_{res}(z_{res})$ and $R_{res}(t_{res})$. **Dark blue**: orbit $(R(t),z(t),t)$. **Light blue**: lines from orbit to predicted resonance.

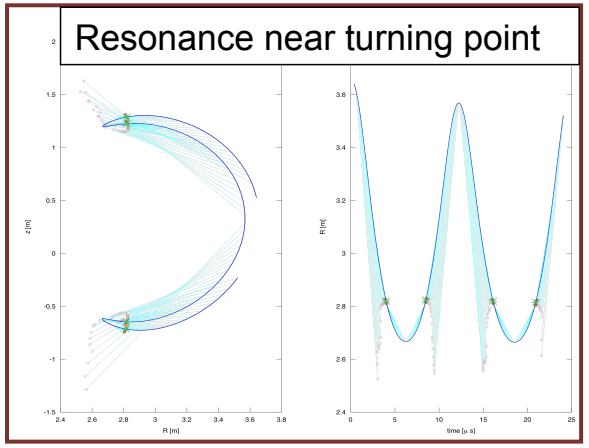












Examples with ASCOT: RF acceleration

- Traditionally RF in orbit following codes has been computationally expensive; collisional time ~1s / bounce times ~10^{-5} s
- Monte Carlo formulation by Eriksson and Schneider allows orbit code to work in accelerated time

$$\begin{cases} d\mu = \dfrac{\partial}{\partial I_\perp}\left(\dfrac{Ze}{m}D\right)N_{ACC} + \xi\dfrac{Ze}{m}\sqrt{2DN_{ACC}} \\ \dfrac{dE}{E} = \dfrac{n\Omega_c}{\omega}\dfrac{d\mu}{\mu} \\ dP_\phi = \dfrac{n_\phi}{\omega}dE \\ d\left(\omega - n\Omega_c - \mathbf{k}\bullet\mathbf{v}\right) = 0 \end{cases}$$ (condition for spatial displacement)

where

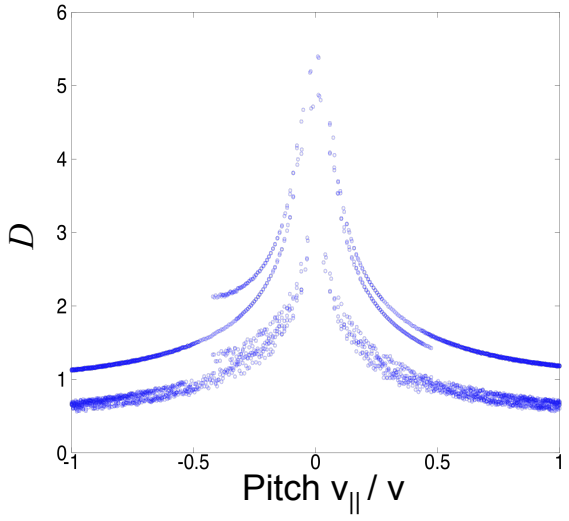
$$D = \left|\tau_{RF} v_\perp Ze E_{eff}\right|^2$$

$$E_{eff} = E_+ J_{n-1} + E_- J_{n+1}$$

$$I_\perp = \frac{\omega}{n\Omega} B\mu$$

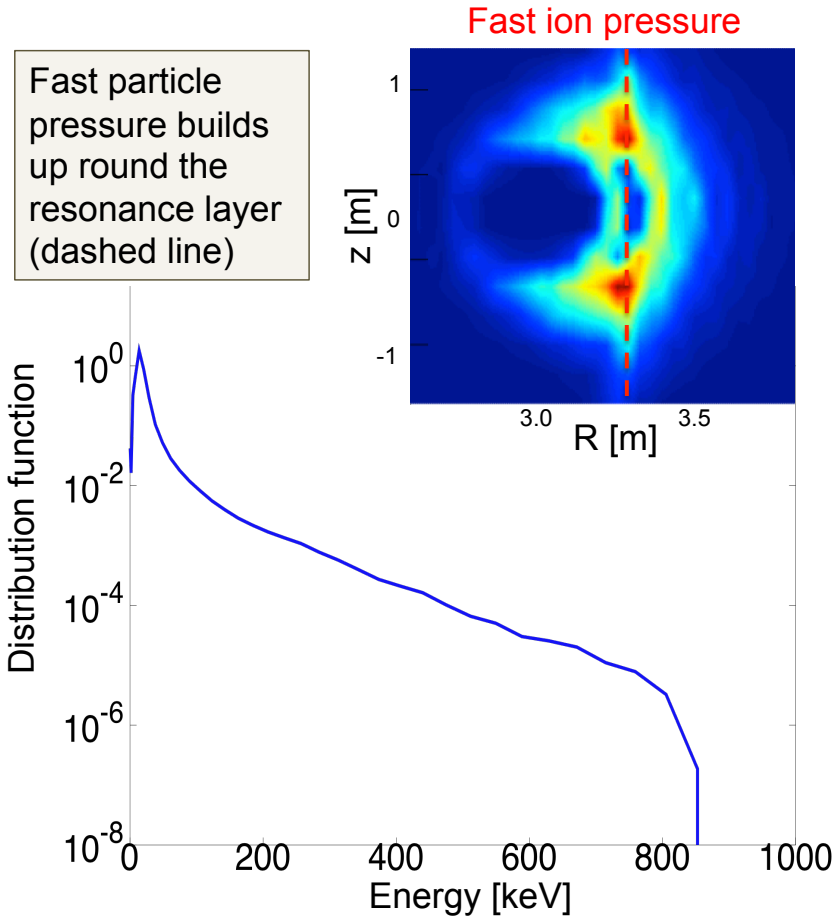
$$J_k = J_k(k_\perp \rho)$$

$$\xi \in N(0,1)$$



Simulations with ASCOT-RFOF

- ASCOT-RFOF simulations have been performed
- Example below: H minority in JET; time simulated=4ms; Prf=20MW
- Unrealistic, *but* shows how RFOF drives fast ion tail and turning points accumulate around the resonance



DOXYGEN documentation

How to couple your orbit following code to RFOF *from man-page*

1. Generate the data structure needed in RFOF. These are NOT global and your code has to keep track of them (keep your pointers until data is deallcocated). RFOF provides constructors for these data structures:
   - **THE MARKERS:** The marker structure are set in the subroutine `set_marker_pointers` in the module RFOF_markers. The markers data, e.g. v-parallel and psi, are all stored as pointer to your internal markers; i.e. both your code RFOF access the same memory and changes set in RFOF are immediately seen in your code, ad vice versa. This routine therefore require you to provide TARGETs for all marker variables. It also means that this structure is set only once per marker.
   - **THE RESONANCE MEMORY:** In order to evaluate the wave-particle interaction strength you need to know the time derivative of the cyclotron frequency of the marker, thus the short term time-history of the particle needs to be stored. This is the perpose of the "resonance-memory". The memory is contructed by the subroutine `constructor_rf_resonance_memory_matrix` in the module RFOF_resonance_memory. Note that this memory is specific for each particle and has to be reset every time you switch to tracing a new particle. To reset the memory use the subroutine `reset_rf_resonance_memory_matrix` in the module RFOF_resonance_memory.
   - THE RF WAVE FIELD: The RF wave field is handled entirely inside RFOF. The contructor is `dummy_rf_wave_field` in the module RFOF_resonance_memory. At the moment this constructor generates a dummy field with constant components over the entire plasma.
   - THE RF-INTERACTION DIAGNOSTICS: The diagnostics is handled entirely inside RFOF. The contructor is `contructor_RFOF_cumlative_diagnostics` in the module RFOF_diagnostics.
2. Write a subroutine to provide RFOF with your magnetic field data. The subroutine should be called local_magnetic_field_interface_to_RFOF and the interface for this routine is given below

```
real(8) RFOF_kick::quasilinear_RF_diffusion_coeff ( type(particle),intent(in)          marker,
                                                    type(rf_wave_local),intent(in)     RFlocal,
                                                    type(resonance_memory),intent(in)  mem
                                                  )
```

function quasilinear_RF_diffusion_coeff(marker,RFlocal,mem) result(diffusion)

Quasilinear diffusion coefficient for resonant interactions between RF wave field and charged particle in an inhomogeous magnetic field. This is a local operator given by the local magnetic fields and wave quantities and their time derivatives in the frame of the particles.

The diffusion coefficient in $I_\perp$ is:

$$D = \frac{1}{2}|\tau Z e v_\perp E_{eff}|^2$$

See [L.-G. Eriksson, Nuclear Fusion, 2005]

Attention:
`dIperp` is in units [MeV].

INPUT

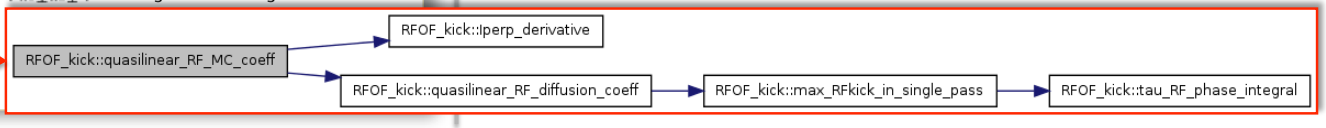
Parameters:
marker Properties of the marker (weight, mass, charge, velocity...)
RFlocal Local properties of the RF wave field ($|B|, RB_\phi, \ldots$)
mem Short term memory of the resonance condition along the orbit

OUTPUT

Parameters:
diffusion (out) Quasilinear scattering coefficient : $D = < dI_\perp dI_\perp >$ during one crossing of the resonance.

Ex. documentation of function

Here is the call graph for this function:



Here is the caller graph for this function:

Summary

RFOF (RF library for Orbit Following Monte Carlo codes) is under development within IMP5/ACT4 to provide detailed RF model

•The model allows time-accelerated orbit tracing; thus much faster than previous orbit-following RF codes

•RFOF has a generic interface to allow coupling to different orbit codes

•Much of RFOF is already written

- Finding resonance positions, give RF kicks
- To do: resonance positions with time acceleration; renormalize E-field to keep power constant…

•Coupling to ASCOT well advanced

- Resonance found
- Predicted resonances used to set future time steps in ASCOT
- First tests of RF-kicks were promising
- First simulation showing RFOF driving fast ion tail

•Future work:

- Finish missing parts of physics model / testing
- Improved diagnostics
- Coupling to orbit code SPOT