Modeling of spin-orbital dynamics in a FS lattice with E+B in the same element

Alexander Aksentev

April 16, 2018

Tasks from supervisor

- Study effects of WF tilts (preserves Lorentz force) in FS lattice on S_x, S_y, S_z;
- Same for quadrupole shifts (doesn't preserve LF);
- Study decoherence as a function of the inital beam distribution $(x, y, \delta W)$;
- Study optimal sextupole placement for the suppression of decoherence and chromaticity;
- Modeling of field calibration by effective gamma in the horizontal plane (CW/CCW procedure);

Conventional ODE integrator

Python prototype

- Classes defining most commonly utilized accelerator elements (dipoles, quadrupoles, Wien filters, etc);
- Vectorized RHS computation;
- Two versions of element positioning imperfections (tilting):
  - via computing the tilt matrix, and applying it to the computed field at run time (more general but time-consuming, doesn't preserve guiding field strength by default);
  - customized tilting for dipole, WF (less time-consuming, preserves the Lorentz force acting on the particle), and shift for quadrupole;
- Reason: needed a tool whose output could be exactly interpreted (source code could be understood by *me*)

Decoherence test

- 1000 initial conditions from $\Delta K \sim N(0, 10^{-4})$;
- $\vec{S}(0) = (0, 0, 1)$;
- Tracking for the first 100 turns;
- Statistics: $\Omega_x = S_y(100)/\Delta t(100),\ \Omega_y = S_x(100)/\Delta t(100)$.

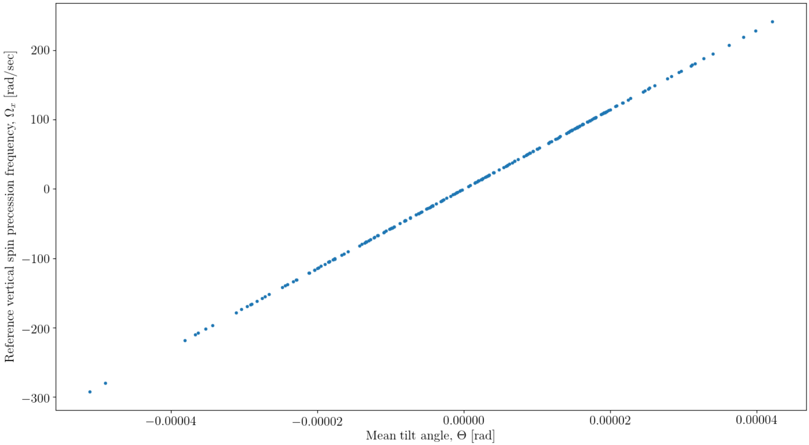


Figure: Reference particle spin precession frequency (vertical plane) vs mean tilt angle

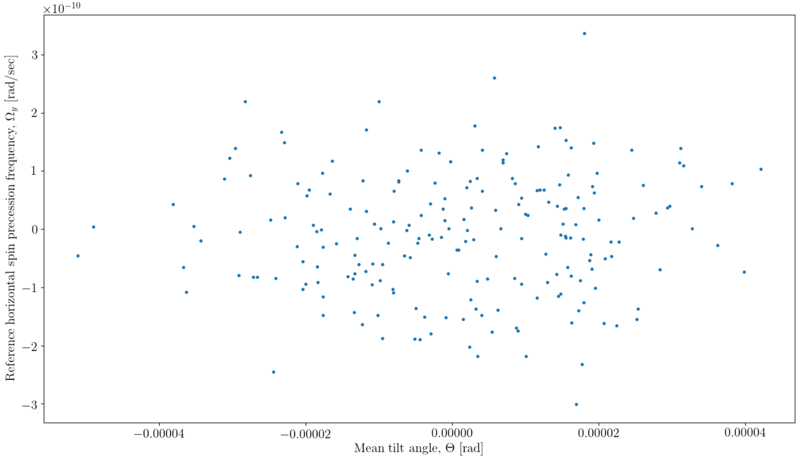


Figure: Reference particle spin precession frequency (horizontal plane) vs mean tilt angle

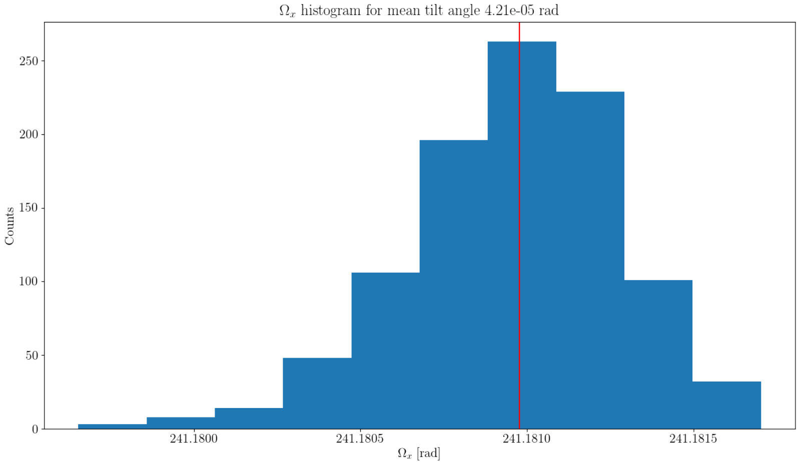


Figure: Vertical spin precession frequencies at the most positive mean tilt; red: reference particle

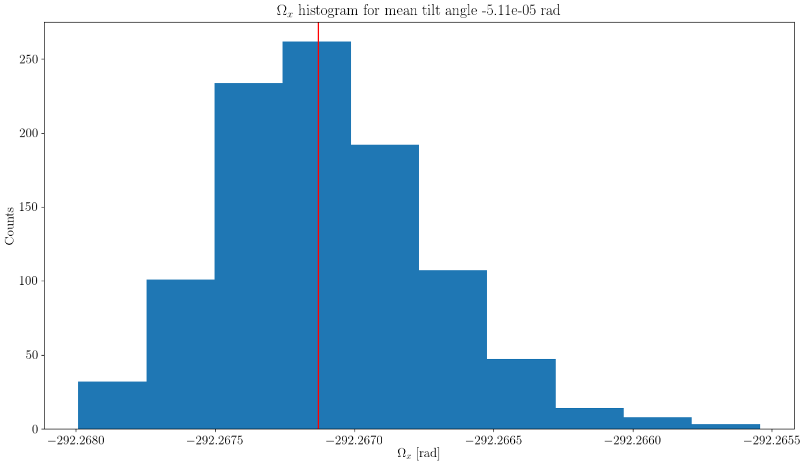


Figure: Vertical spin precession frequencies at the most negative mean tilt; red: reference particle

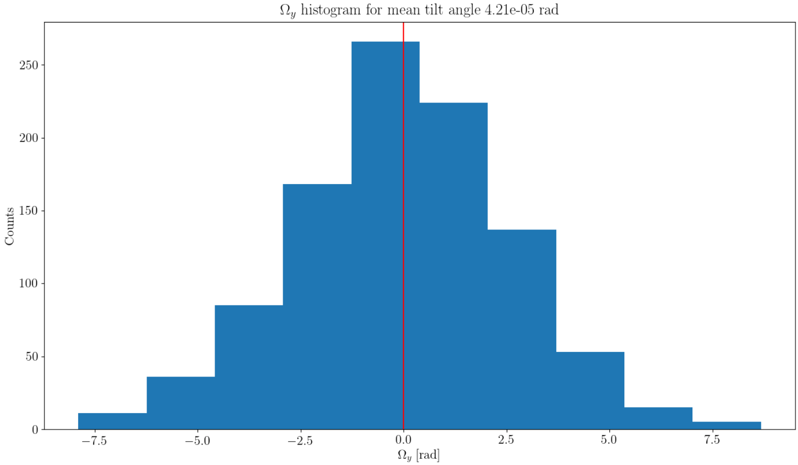


Figure: Horizontal spin precession frequencies at the most positive mean tilt; red: reference particle

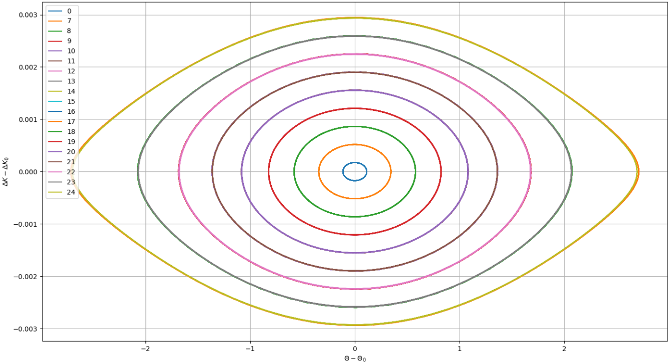
Conventional ODE integrator

C++ extension

- Because python isn't fast enough, rewrote parts of the integrator in c++;
- Still problems with speed (and precision) — working on that now;
- However, even if those problems are resolved, step-by-step integration is not a viable option for the type of analysis required:

What is required?

- $\omega_i = \omega_0 + G_6 \cdot \Delta\gamma_i^2$, where $\Delta\gamma_i^2$ is the average gamma in phase space, due to synchrotron oscillations.
- $Q_s = \frac{\omega_s}{\omega_{rev}} \ll 1$ (like $1/35$) $=>$ require thousands of turns.



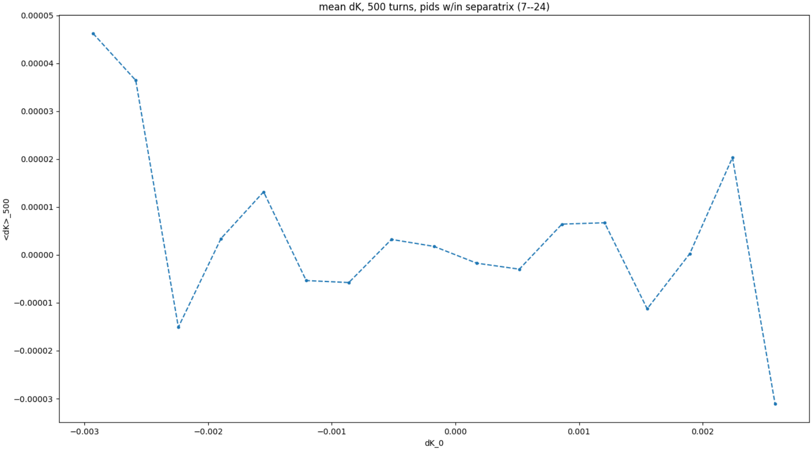


Figure: $\langle \Delta K \rangle$ vs ΔK_0 after 500 turns (14 synchrotron oscillations)

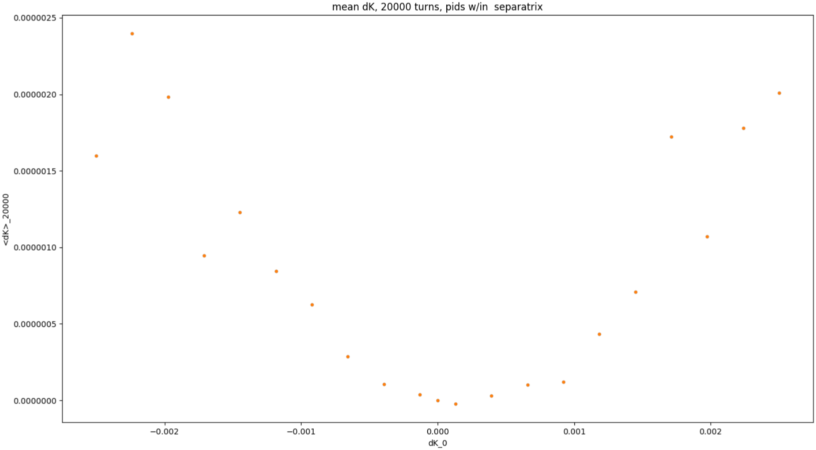


Figure: $\langle \Delta K \rangle$ vs ΔK_0 after 20,000 turns (571 synchrotron oscillations)

Therefore...

- Dr Valetov compares a conventional Runge-Kutta 8-th order, step-size calibrated integrator (MSURK89) with COSY INFINITY with regard to run time;
- By my estimation, that integrator would take 32 hours to run a **single** simulation with just one realization of an imperfect 397-element FS lattice with the beam of 1000 particles for initial distribution;
- COSY INFINITY is an order of magnitude faster; that's manageable.