Self-consistent beam simulation methods with Particle In Cell codes.

Christian Ratcliff

Michigan State University

December 13, 2023

Presentation Overview

Runge-Kutta Method

- Requires functions evaluated at beginning, middle, and end of interval
- Single-step method, utilizing the newly obtained information at each step to calculate the value
- Commonly used at 4th order, and rarely used beyond 7th order as the number of functions to be computed rises and is not as efficient as other methods

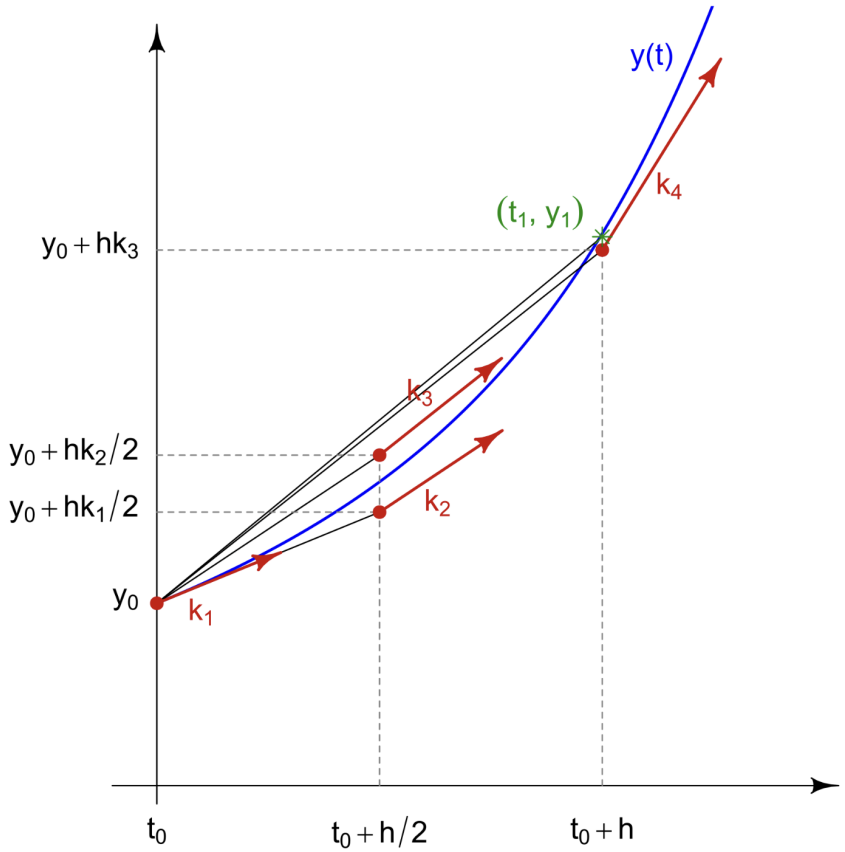

Figure: Fourth order Runge-Kutta method of integration diagram [Traum]

1. Predictor/Corrector
   - Multi-step, essentially approximating as polynomial
   - Used for high accuracy

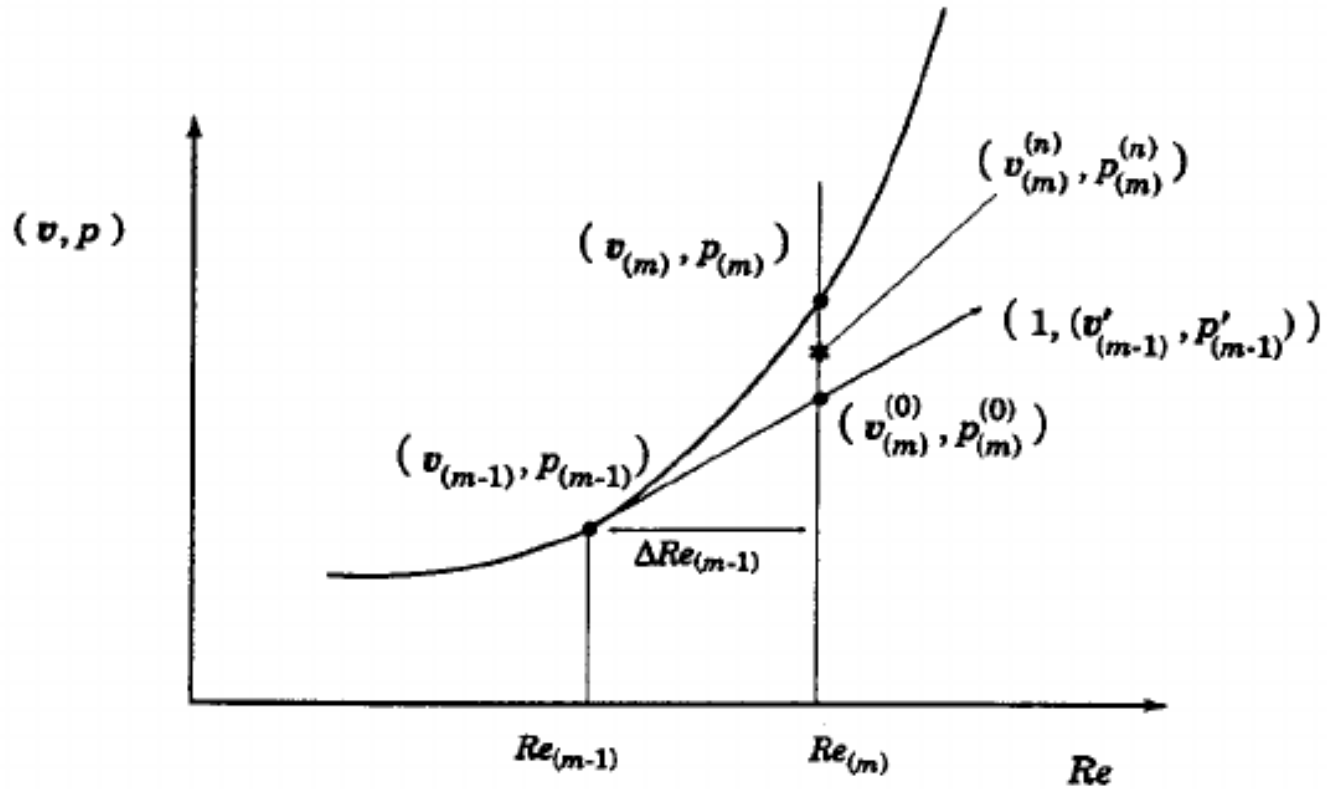

Figure: Diagram of predictor corrector method [Derby, 2000]

Predictor-Corrector Math

Single Particle Dynamics

$f(y,t) = y'(y,t)$, $y_{n+1} = y_n + \sum_{k=0}^{M} b_k^{(M)} f^{(n-k)}$ (predictor)

$y_{n+1} = y_n + \sum_{k=0}^{M} a_k^{(M)} f^{(n-k+1)}$ (corrector)

1. Use RK method to get initial prediction of y_M
2. Plug that value into Predictor Formula
3. Take value from predictor for y_{M+1}, evaluate to obtain f^{M+1}
4. Using this value for f^{M+1}, place into corrector formula
5. Using this new y_{M+1}, evaluate again, repeating steps 3 and 4, getting the final y_{M+1} value
6. Store $f^{(M+1)}$ and y_{M+1}
7. Repeat 2-6 as needed (PECEC)

Symplectic Integration

Single Particle Dynamics

Useful for magnetic optics as they are composed of symplectic matrices

$$D_H z = \{z, H\}; A \equiv D_T, B \equiv D_V$$

$$z(\tau) = [e^{\tau(A+B)}]z_0$$

$$[e^{\tau(A+B)}] = \prod_{i=1}^{k} e^{c_i \tau A} e^{d_i \tau B} = S$$

$$S^{(2)}(\tau) = e^{(1/2)\tau A} e^{\tau B} e^{(1/2)\tau A}$$

$$S^{(4)}(\tau) = S^{(2)}(x_1 A) \cdot S^{(2)}(x_0 B) \cdot S^{(2)}(x_1 A)$$

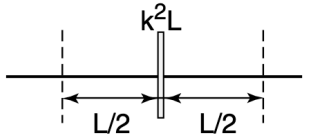

Figure: 2nd Order Symplectic Integrator [Chao, 2002]

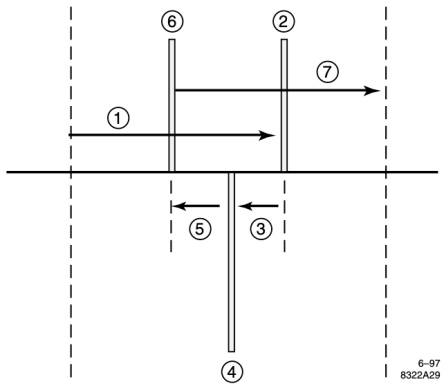

Figure: 4th Order Symplectic Integrator [Chao, 2002]

Tells us about the shape of the phase space ellipse at a given s value.

$$\gamma = \frac{1 + \alpha^2}{\beta}$$

$$\beta' = 2\alpha$$

$$\alpha' = k\beta - \gamma$$

$$\psi = \int_0^s \frac{1}{\beta(s')} \mathrm{d}s'$$

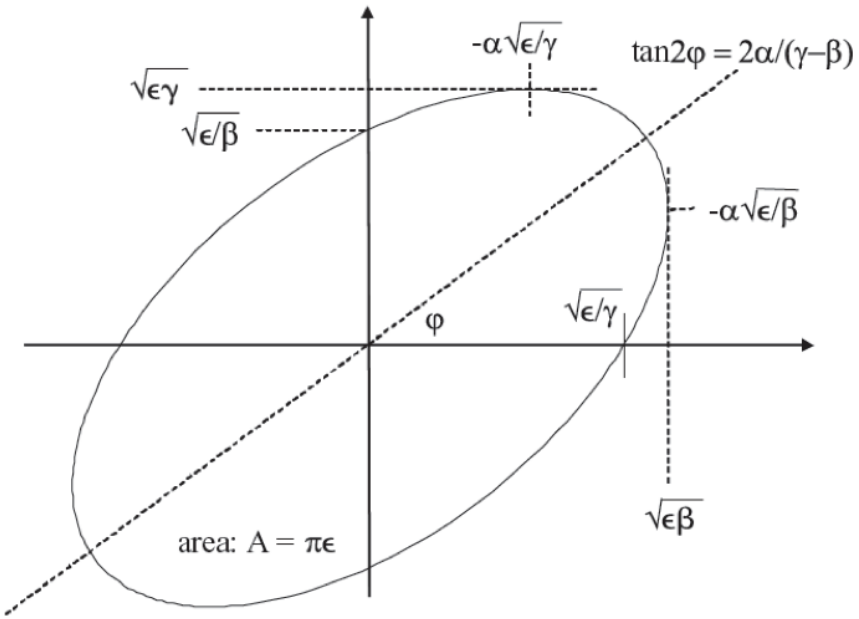

Figure: Phase space ellipse with the Twiss Parameters [Biscari, 2014]

The matrices representing these elements can be found by solving Hill's Equation,

$$x''(s) + k(s)x(s) = 0 \tag{1}$$

$$\begin{pmatrix} x \\ x' \end{pmatrix}_s = M \begin{pmatrix} x \\ x' \end{pmatrix}_0 \tag{2}$$

$$M(s) = \begin{bmatrix} \sqrt{\frac{\beta}{\beta_0}}\left(\cos\psi + \alpha_0 \sin\psi\right) & \sqrt{\beta_0\beta}\sin\psi \\ \sqrt{\frac{1}{\beta_0\beta}}\left((\alpha_0 - \alpha)\cos\psi - (1 + \alpha_0\alpha)\sin\psi\right) & \sqrt{\frac{\beta}{\beta_0}}\left(\cos\psi + \alpha_0\sin\psi\right) \end{bmatrix} \tag{3}$$

where α, β, and ψ are functions of s

Simulation of Beam

Particle Simulation Techniques

- Easy to treat the beam in this manner if it is round (azimuthally symmetric)
- Traditionally broken up into two parts; the H_{self} and H_{ext}
- Without any tricks, it is extremely inefficient for non-azimuthally symmetric beams
  - First must loop over all the particles applying the force
  - Then must loop through all of the particles receiving the force
  - Results in time complexity of $\mathcal{O}(N^2)$
- With 10^4 particles, it requires GB of memory
- With 10^5 particles, it requires 100 GB of memory

Grid Based

Symplecticity

- Nowadays there solutions to Poisson's Equations without the $\mathcal{O}(N^2)$ complexity.
- It is **NOT** symplectic, if it is not specifically enforced, which is a complicated process

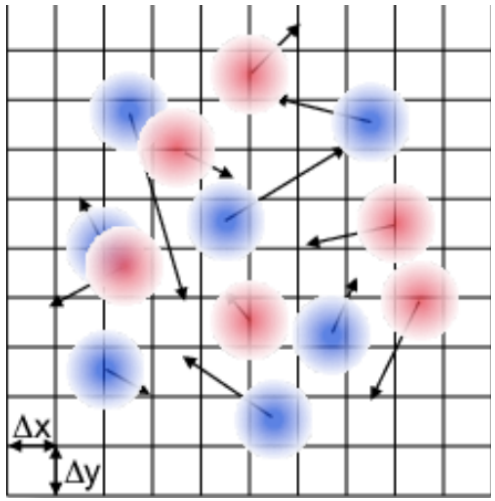

Figure: Example of the grid that is overlayed with the particles [WarpX, 2017]

PIC part 2

1. **Making the Grid:** Break up the cross-section of the beam into a 2D grid
2. **Charge Placing:** Place charges on the grid, with all of the charge be depositing on the nearest grid point
3. **Field Equations:** Solve the field equations on the grid
4. **Interpolation:** Interpolate the solutions of the field equations at the grid point nearest the particle using the values at the grid intersections

Auto-Differentiation

- Currently, PIC uses numeric integration for solving equations, which can be slow
- Goal is to integrate AD into these PIC codes
- AD breaks an equation into elementary steps
- Forward Mode:
  - Calculates derivatives w.r.t. each input independently (must be stored)
  - Efficient for few inputs and many outputs
- Reverse Mode:
  - Calculates derivatives w.r.t. each input in single backward pass
  - Efficient for many inputs and few outputs
  - Much more useful for optimization

Auto-Differentiation Pt. 2

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Primal Trace

$v_{-1} = x_1$	$= 2$	
$v_0 = x_2$	$= 5$	
$v_1 = \ln v_{-1}$	$= \ln 2$	
$v_2 = v_{-1} \times v_0$	$= 2 \times 5$	
$v_3 = \sin v_0$	$= \sin 5$	
$v_4 = v_1 + v_2$	$= 0.693 + 10$	
$v_5 = v_4 - v_3$	$= 10.693 + 0.959$	
$y = v_5$	$= 11.652$	

Reverse Adjoint (Derivative) Trace

$\bar{x}_1 = \bar{v}_{-1}$		$= \mathbf{5.5}$
$\bar{x}_2 = \bar{v}_0$		$= \mathbf{1.716}$
$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5 = \bar{y}$	$= 1$	

Figure: Auto-Differentiation table showing both forward and reverse modes. [Bayden, 2018]

References

Robert D. Ryne (2009)

Computational Methods in Accelerator Physics
USPAS UNM, 2009

Georg Hoffstaetter (2023)

USPAS Graduate Accelerator Physics
USPAS Cornell, 2023

Haruo Yoshida (1990)

Construction of higher order symplectic integrators
Phys. Lett. A. 150 (5–7): 262–268

C. Biscari (2023)

Transverse Beam Dynamics
UAB 2014-15

Hilber Traum

Runge-Kutta 4th Order
CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=64361270

Alex Chao (2002)

Lecture Notes on Topics in Accelerator Physics
SLAC-PUB-9574

Jinyu Wan (2023)

Towards auto-differentiable modeling in Julia
Journal Name 12(3), 45 – 678.

R.E. Wengert (1964)

A Simple Automatic Derivative Evaluation Program
Communications of the ACM 7(8), 463 – 464.

Baydin, Atilim Gunes, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind (2018)

Automatic Differentiation in Machine Learning: A Survey
The Journal of Machine Learning Research 18(153), 1– 43.

WarpX Documentation (2017)

Particle in Cell Method
warpx.readthedocs.io

Jeffery Derby (2000)

Construction of Solution Curves for Large Two-Dimensional Problems of Steady-State Flows of Incompressible Fluids
SIAM Journal on Scientific Computing 22(1)

The End

Questions? Comments?