Simulation of Charged Particles Motion with the Guiding Center Approximation

Stefano Markidis

KTH Royal Institute of Technology

The Guiding Center (GC) Approximation

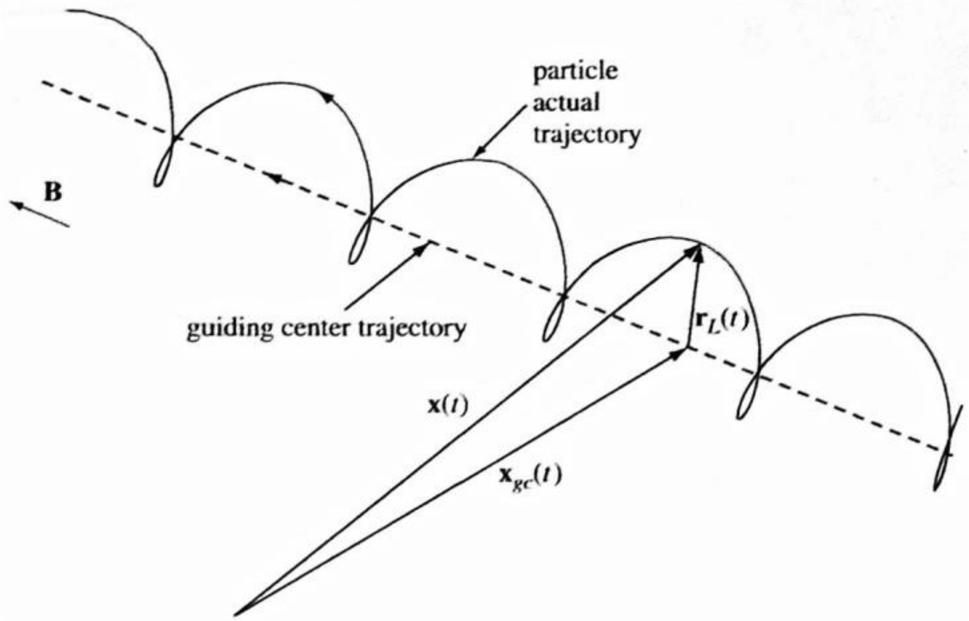
- The guiding center is the geometric center of cyclotron motion.
- We will calculate the trajectory of the guiding center.
- The particle position r is substituted with:

$$\mathbf{r} = \mathbf{R} + \rho$$

- Assuming that the cyclotron radius is much smaller than the length scale of the Field, we can expand B around R to the first order in the Taylor series:

$$\mathbf{B}(\mathbf{r}) = \mathbf{B}(\mathbf{R}) + (\rho \cdot \nabla)\mathbf{B}$$

- This expression is substituted into the Newton-Lorenz equation, and the equation is **averaged over a gyro-period**, eliminating rapidly oscillating terms containing ρ and its derivatives.



Fundamentals of Plasma Physics,
Dr. Paul Bellan, Cambridge Press 2006

GC Approximation

1

$$\frac{d\mathbf{R}}{dt} = \frac{mv^2}{2qB^2}\left(1 + \frac{v_{//}^2}{v^2}\right)\hat{\mathbf{b}} \times \nabla B + v_{//}\hat{\mathbf{b}}$$

2

$$\frac{dv_{//}}{dt} = -\frac{\mu}{m}\hat{\mathbf{b}} \cdot \nabla B$$

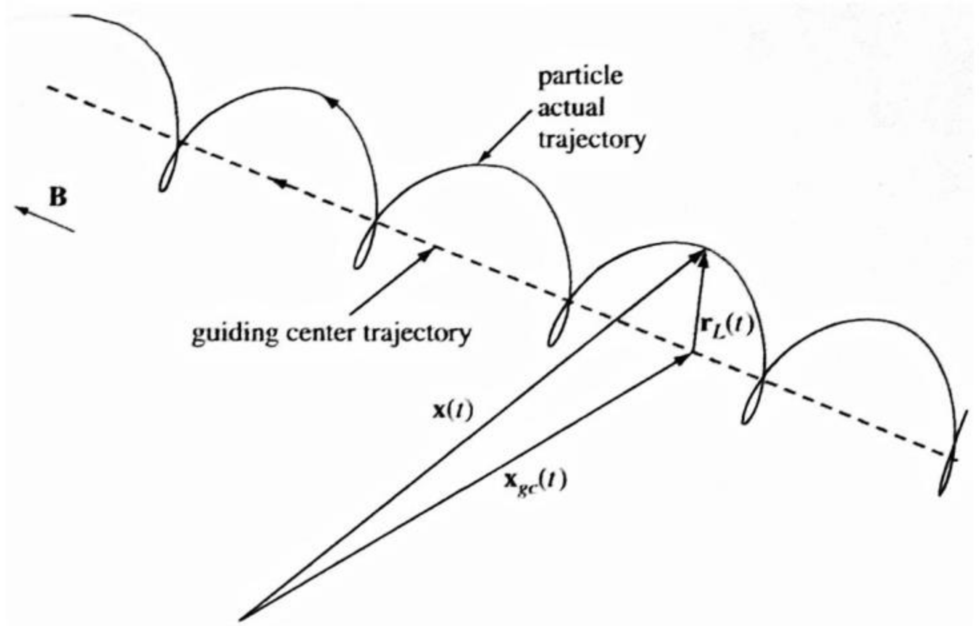
$$\mu = \frac{mv_\perp^2}{2B}$$

Magnetic Moment

The unknowns are the three coordinates of the guiding center and the parallel (to **B**) velocity

Advantages of GC approach

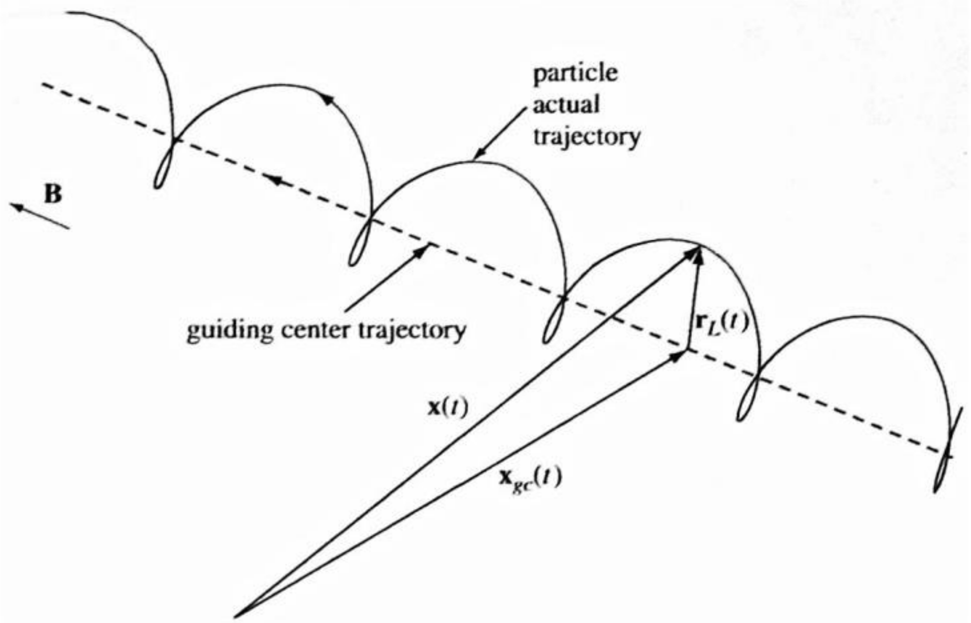
- For numerical stability reasons, the time step in the computer simulations needs to be a fraction of the fastest time scale in the system (in this case, the gyro-period).
  - T = 2*pi*m/(abs(e)*B);
- When we simulate electrons, the gyro-period can become very small, imposing a tiny simulation time step (simulation will take a long time).
- Because the GC approximation removes the gyration motion from the model, we take time steps larger than the gyro period, and the simulation will take less time.



Fundamentals of Plasma Physics,
Dr. Paul Bellan, Cambridge Press 2006

Validity of GC Approximation

- We derived the GC equation assuming that the <u>Larmor radius is much smaller than the length scale of the field.</u>
  - This approximation is often used in the simulation of fusion devices where you have a very strong magnetic field.
- GC approximation is not valid in the presence of magnetic null or weak magnetic field.
- When you have highly energetic particles, the Larmor radius might become comparable or larger than the length scale of the magnetic field (in the case of Earth's magnetic dipole).
  - In this case, the GC approximation is not valid.



Fundamentals of Plasma Physics,
Dr. Paul Bellan, Cambridge Press 2006

```python
def newton_lorentz_gc(t, x_vect):

x, y, z, vpar = x_vect
vsq = v_mod**2
fac1 = -B0 * Re**3 / (x**2 + y**2 + z**2)**2.5
Bx = 3 * x * z * fac1
By = 3 * y * z * fac1
Bz = (2 * z**2 - x**2 - y**2) * fac1
B_mod = np.sqrt(Bx**2 + By**2 + Bz**2)
# Magnetic moment (adiabatic invariant)
mu = m * (vsq - vpar**2) / (2 * B_mod)

# Gradient of B_mod (numerical)
d = 1e-3 * Re # Small perturbation for finite difference
gradB_x = (getBmod(x + d, y, z) - getBmod(x - d, y, z)) / (2 * d)
gradB_y = (getBmod(x, y + d, z) - getBmod(x, y - d, z)) / (2 * d)
gradB_z = (getBmod(x, y, z + d) - getBmod(x, y, z - d)) / (2 * d)
# Unit vector along B
b_unit_x = Bx / B_mod
b_unit_y = By / B_mod
b_unit_z = Bz / B_mod

# Cross product b_unit x grad(B)
bxgB_x = b_unit_y * gradB_z - b_unit_z * gradB_y
bxgB_y = b_unit_z * gradB_x - b_unit_x * gradB_z
bxgB_z = b_unit_x * gradB_y - b_unit_y * gradB_x
# Dot product b_unit · grad(B)
dotpr = b_unit_x * gradB_x + b_unit_y * gradB_y + b_unit_z * gradB_z

# Guiding center velocity components
fac = m / (2 * q * B_mod**2) * (vsq + vpar**2)
dxdt = fac * bxgB_x + vpar * b_unit_x
dydt = fac * bxgB_y + vpar * b_unit_y
dzdt = fac * bxgB_z + vpar * b_unit_z
dvpar_dt = -mu / m * dotpr

return [dxdt, dydt, dzdt, dvpar_dt]
```

ODE to Solve

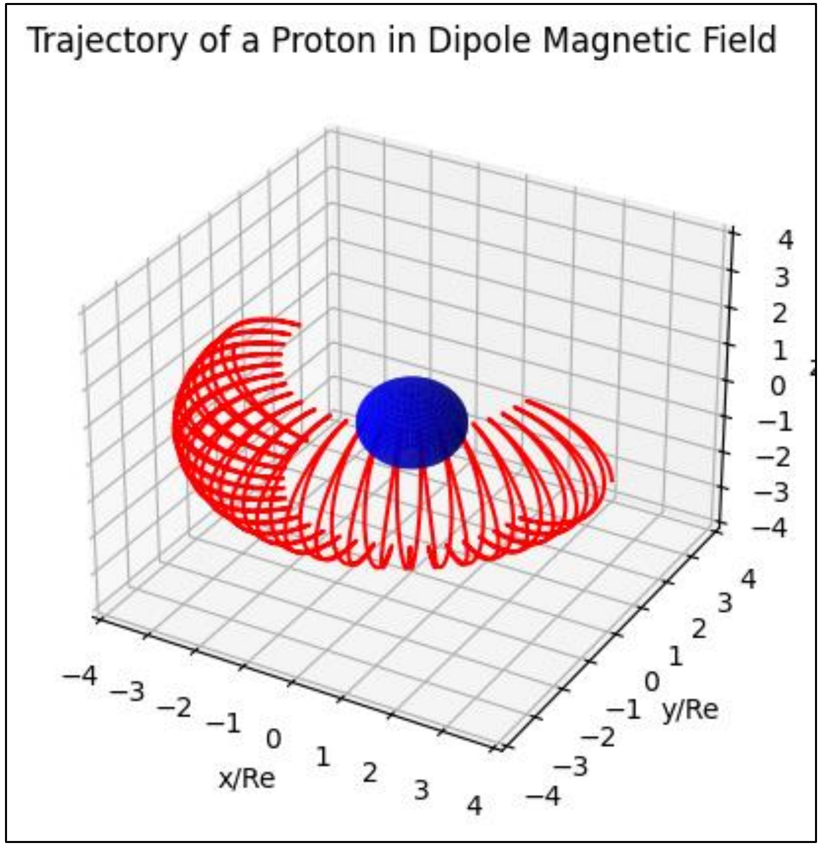
```python
# Magnetic field computation
def getBmod(x, y, z):
fac1 = -B0 * Re**3 / (x**2 + y**2 + z**2)**2.5
Bx = 3 * x * z * fac1
By = 3 * y * z * fac1
Bz = (2 * z**2 - x**2 - y**2) * fac1
return np.sqrt(Bx**2 + By**2 + Bz**2)
```

$$\frac{d\mathbf{R}}{dt} = \frac{mv^2}{2qB^2}\left(1 + \frac{v_{//}^2}{v^2}\right)\hat{\mathbf{b}} \times \nabla B + v_{//}\hat{\mathbf{b}}$$

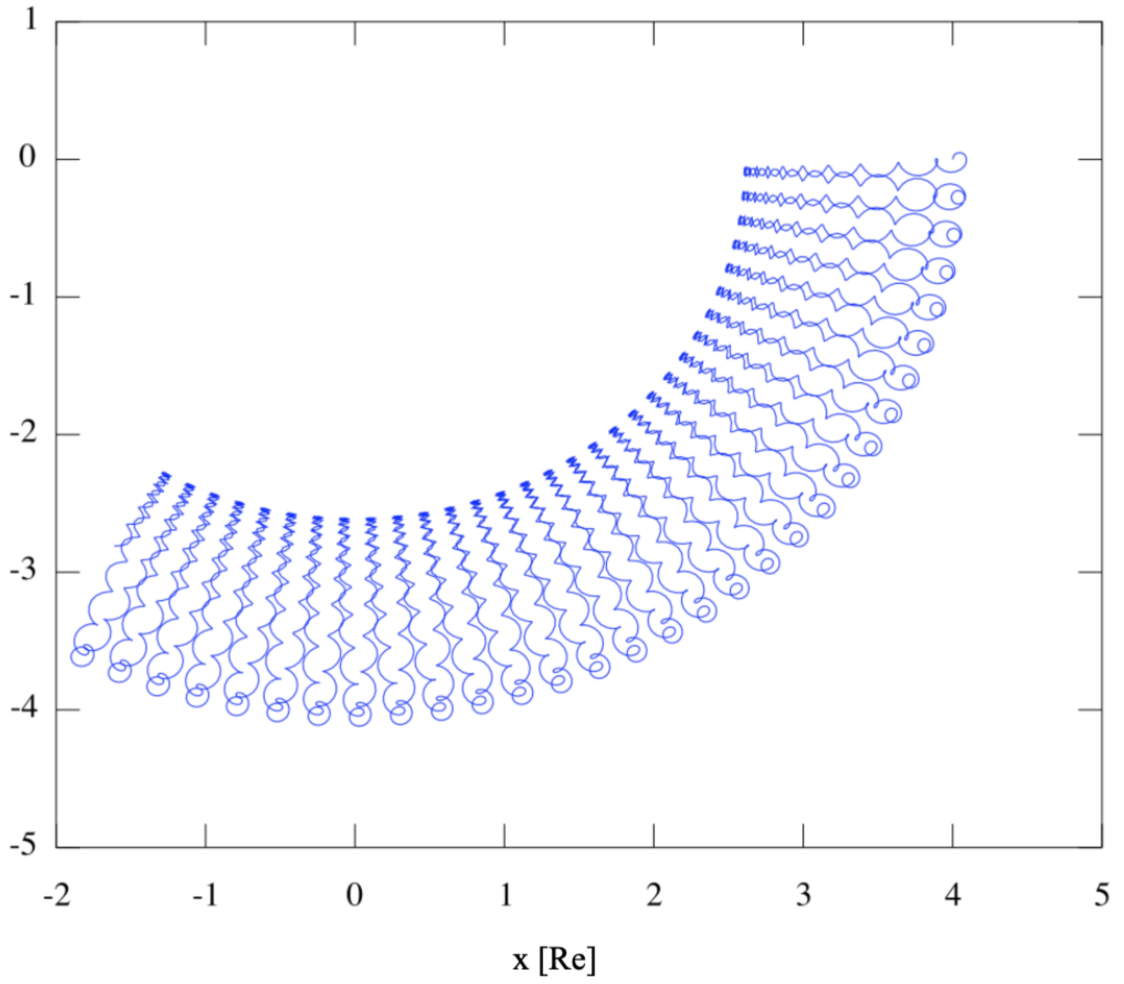
$$\frac{dv_{//}}{dt} = -\frac{\mu}{m}\hat{\mathbf{b}} \cdot \nabla B$$

Main Code

```python
# Initial conditions
x0 = 4 * Re
y0 = 0
z0 = 0
initial_conditions = [x0, y0, z0, v_par0]

# Time span
tfin = 80.0 # seconds
time = np.arange(0, tfin, 0.01)

# Solve the ODE
solution = solve_ivp(newton_lorentz_gc, [0, tfin], initial_conditions,
t_eval=time, method='RK45')

# Extract solution
x_sol = solution.y.T # Transpose for easier handling

# Plot trajectory
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x_sol[:, 0] / Re, x_sol[:, 1] / Re, x_sol[:, 2] / Re, 'r')
ax.set_xlabel('x / Re')
ax.set_ylabel('y / Re')
ax.set_zlabel('z / Re')
ax.set_title('Guiding Center Trajectory')
plt.show()
```



No Approximation



GC Approximation