


Project 3

Cory Alexander Balaton & Janita Ovidie Sandtr en Willumsen

 <https://github.uio.no/FYS3150-G2-2023/Project-3>

(Dated: October 16, 2023)

Add an abstract for the project?

I. INTRODUCTION

Newton's second law for a particle i is then

II. METHODS

$$\frac{d^2 \mathbf{r}_i}{dt^2} = \frac{\mathbf{F}_i(t, \frac{d\mathbf{r}_i}{dt}, \mathbf{r}_i)}{m_i}, \quad (5)$$

A. Units and constants

Before continuing, we need to define the units we'll be working with. Since we are working with particles, we need small units to work with so the numbers we are working with aren't so small that they could potentially lead to large round-off errors in our simulation. The units that we will use are listed in Table I.

Dimension	Unit	Symbol
Length	micrometer	μm
Time	microseconds	μs
Mass	atomic mass unit	u
Charge	the elementary charge	e

TABLE I. The set of units we'll be working with.

With these base units, we get

$$k_e = 1.3893533 \cdot 10^5 \frac{u(\mu m)^3}{(\mu s)^2 e}, \quad (1)$$

and we get that the unit for magnetic field strength (Tesla, T) and electric potential (Volt, V) are

$$T = 9.64852558 \cdot 10^1 \frac{u}{(\mu s)e} \quad (2)$$

$$V = 9.64852558 \cdot 10^7 \frac{u(\mu m)^2}{(\mu s)^2 e}.$$

B. Dealing with a multi-particle system

For a multi-particles system, we need to modify \mathbf{F} to account for the force of other particles in the system acting upon each other. To do that, we add another term to \mathbf{F}

$$\mathbf{F}_i(t, \mathbf{v}_i, \mathbf{r}_i) = q_i \mathbf{E}(t, \mathbf{r}_i) + q_i \mathbf{v}_i \times \mathbf{B} - \mathbf{E}_p(t, \mathbf{r}_i), \quad (3)$$

where i and j are particle indices and

$$\mathbf{E}_p(t, \mathbf{r}_i) = q_i k_e \sum_{j \neq i} q_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3}. \quad (4)$$

We can then rewrite the second order ODE from equation 5 into a set of coupled first order ODEs. We now rewrite Newton's second law of motion as

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \mathbf{v}_i \\ \frac{d\mathbf{v}_i}{dt} &= \frac{\mathbf{F}_i(t, \mathbf{v}_i, \mathbf{r}_i)}{m_i}. \end{aligned} \quad (6)$$

C. Forward Euler

For a particle i , the forward Euler method for a coupled system is expressed as

$$\begin{aligned} \mathbf{r}_{i,j+1} &= \mathbf{r}_{i,j} + h \cdot \frac{d\mathbf{r}_{i,j}}{dt} = \mathbf{r}_{i,j} + h \cdot \mathbf{v}_{i,j} \\ \mathbf{v}_{i,j+1} &= \mathbf{v}_{i,j} + h \cdot \frac{d\mathbf{v}_{i,j}}{dt} = \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{F}(t_j, \mathbf{v}_{i,j}, \mathbf{r}_{i,j})}{m}, \end{aligned} \quad (7)$$

for particle i where j is the current time step of the particle, m is the mass of the particle, and h is the step length.

When dealing with a multi-particle system, we need to ensure that we do not update the position of any particles until every particle has calculated their next step. An easy way of doing this is to create a copy of all the particles, then update the copy, and when all the particles have calculated their next step, simply replace the particles with the copies. Algorithm 1 provides an overview on how that can be achieved.

Algorithm 1 Forward Euler method

```

procedure EVOLVE FORWARD EULER(particles, dt)
   $N \leftarrow$  Number of particles in particles
   $a \leftarrow$  Calculate  $\frac{\mathbf{F}_i}{m_i}$  for each particle in particles
  for  $i = 1, 2, \dots, N$  do
     $\text{particles}_i.\mathbf{r} \leftarrow \text{particles}_i.\mathbf{r} + dt \cdot \text{particles}_i.\mathbf{v}$ 
     $\text{particles}_i.\mathbf{v} \leftarrow \text{particles}_i.\mathbf{v} + dt \cdot a_i$ 

```

D. 4th order Runge-Kutta

For a particle i , we can express the 4th order Runge-Kutta (RK4) method as

$$\begin{aligned} \mathbf{v}_{i,j+1} &= \mathbf{v}_{i,j} + \frac{h}{6} (\mathbf{k}_{\mathbf{v},1,i} + 2\mathbf{k}_{\mathbf{v},2,i} + 2\mathbf{k}_{\mathbf{v},3,i} + \mathbf{k}_{\mathbf{v},4,i}) \\ \mathbf{r}_{i,j+1} &= \mathbf{r}_{i,j} + \frac{h}{6} (\mathbf{k}_{\mathbf{r},1,i} + 2\mathbf{k}_{\mathbf{r},2,i} + 2\mathbf{k}_{\mathbf{r},3,i} + \mathbf{k}_{\mathbf{r},4,i}), \end{aligned} \quad (8)$$

where

$$\begin{aligned} \mathbf{k}_{\mathbf{v},1,i} &= \frac{\mathbf{F}_i(t_j, \mathbf{v}_{i,j}, \mathbf{r}_{i,j})}{m} \\ \mathbf{k}_{\mathbf{r},1,i} &= \mathbf{v}_{i,j} \\ \mathbf{k}_{\mathbf{v},2,i} &= \frac{\mathbf{F}_i(t_j + \frac{h}{2}, \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},1,i}}{2}, \mathbf{r}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{r},1,i}}{2})}{m} \\ \mathbf{k}_{\mathbf{r},2,i} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},1,i}}{2} \\ \mathbf{k}_{\mathbf{v},3,i} &= \frac{\mathbf{F}_i(t_j + \frac{h}{2}, \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},2,i}}{2}, \mathbf{r}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{r},2,i}}{2})}{m} \\ \mathbf{k}_{\mathbf{r},3,i} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},2,i}}{2} \\ \mathbf{k}_{\mathbf{v},4,i} &= \frac{\mathbf{F}_i(t_j + h, \mathbf{v}_{i,j} + h \cdot \mathbf{k}_{\mathbf{v},3,i}, \mathbf{r}_{i,j} + h \cdot \mathbf{k}_{\mathbf{r},3,i})}{m} \\ \mathbf{k}_{\mathbf{r},4,i} &= \mathbf{v}_{i,j} + h \cdot \frac{\mathbf{k}_{\mathbf{v},3,i}}{2}. \end{aligned} \quad (9)$$

In order to find each $\mathbf{k}_{\mathbf{r},i}$ and $\mathbf{k}_{\mathbf{v},i}$, we need to first compute all $\mathbf{k}_{\mathbf{r},i}$ and $\mathbf{k}_{\mathbf{v},i}$ for all particles, then we can update the particles in order to compute $\mathbf{k}_{\mathbf{r},i+1}$ and $\mathbf{k}_{\mathbf{v},i+1}$. In order for the algorithm to work, we need to save a copy of each particle before starting so that we can update the particles correctly for each step.

This approach would require 8 loops to be able to complete the calculation since we cannot update the particles until after all \mathbf{k} values have been computed, however if we create a temporary array that holds particles, we can put the updated particles in there, and then use that array in the next loop, and would reduce the required amount of loops down to 4.

III.

IV. RESULTS

V. CONCLUSION

Algorithm 2 RK4 method

```

procedure EVOLVE RK4(particles, dt)
  N  $\leftarrow$  Number of particles inside the Penning trap
  orig_p  $\leftarrow$  Copy of particles
  tmp_p  $\leftarrow$  Array of particles of size N
  kr  $\leftarrow$  2D array of vectors of size  $4 \times N$ 
  kv  $\leftarrow$  2D array of vectors of size  $4 \times N$ 
  for i = 1, 2, ..., N do
    kr,1,i  $\leftarrow$  particlesi.v
    kv,1,i  $\leftarrow$   $\frac{\mathbf{F}_i}{m_i}$ 
    tmp_pi.r  $\leftarrow$  orig_pi.r +  $\frac{dt}{2} \cdot \mathbf{k}_{r,1,i}$ 
    tmp_pi.v  $\leftarrow$  orig_pi.v +  $\frac{dt}{2} \cdot \mathbf{k}_{v,1,i}$ 
  particles  $\leftarrow$  tmp_p ▷ Update particles
  for i = 1, 2, ..., N do
    kr,2,i  $\leftarrow$  particlesi.v
    kv,2,i  $\leftarrow$   $\frac{\mathbf{F}_i}{m_i}$ 
    tmp_pi.r  $\leftarrow$  orig_pi.r +  $\frac{dt}{2} \cdot \mathbf{k}_{r,2,i}$ 
    tmp_pi.v  $\leftarrow$  orig_pi.v +  $\frac{dt}{2} \cdot \mathbf{k}_{v,2,i}$ 
  particles  $\leftarrow$  tmp_p ▷ Update particles
  for i = 1, 2, ..., N do
    kr,3,i  $\leftarrow$  particlesi.v
    kv,3,i  $\leftarrow$   $\frac{\mathbf{F}_i}{m}$ 
    tmp_pi.r  $\leftarrow$  orig_pi.r + dt · kr,3,i
    tmp_pi.v  $\leftarrow$  orig_pi.v + dt · kv,3,i
  particles  $\leftarrow$  tmp_p ▷ Update particles
  for i = 1, 2, ..., N do
    kr,4,i  $\leftarrow$  particlesi.v
    kv,4,i  $\leftarrow$   $\frac{\mathbf{F}_i}{m}$ 
    tmp_pi.r  $\leftarrow$  orig_pi.r +  $\frac{dt}{6} \cdot (\mathbf{k}_{r,1,i} + \mathbf{k}_{r,2,i} + \mathbf{k}_{r,3,i} + \mathbf{k}_{r,4,i})$ 
    tmp_pi.v  $\leftarrow$  orig_pi.v +  $\frac{dt}{6} \cdot (\mathbf{k}_{v,1,i} + \mathbf{k}_{v,2,i} + \mathbf{k}_{v,3,i} + \mathbf{k}_{v,4,i})$ 
  particles  $\leftarrow$  tmp_p ▷ Final update

```

F in the algorithm does not take any arguments as it uses the velocities and positions of the particles inside the array *particles* to calculate the total force acting on particle *i*.