How physics simulations work: simulating particles in electric and magnetic fields

Nikolaj Roager Christensen

Student Colloquium in Physics and Astronomy, Aarhus University

May 2022 CE

How physics simulations work: simulating particles in electric and magnetic fields.

Motivation and introductions.

- Numerical simulations are important.

Motivation and introductions.

- Numerical simulations are important.
- Testing setups, non-analytical systems.

Motivation and introductions.

- Numerical simulations are important.
- Testing setups, non-analytical systems.
- Engineering: testing integrity of buildings/machines.

Motivation and introductions.

- Numerical simulations are important.
- Testing setups, non-analytical systems.
- Engineering: testing integrity of buildings/machines.
- Entertainment: Video-games and CGI effects in movies.

Motivation and introductions.

- Numerical simulations are important.
- Testing setups, non-analytical systems.
- Engineering: testing integrity of buildings/machines.
- Entertainment: Video-games and CGI effects in movies.
- Demonstration, charged particles in electric and magnetic fields.

Motivation and introductions.

- Numerical simulations are important.
- Testing setups, non-analytical systems.
- Engineering: testing integrity of buildings/machines.
- Entertainment: Video-games and CGI effects in movies.
- Demonstration, charged particles in electric and magnetic fields.
- Simulations are not experiments!

Theory: Classical non-relativistic particles.

- Some repetition from Electrodynamics.

Theory: Classical non-relativistic particles.

- Some repetition from Electrodynamics.
- The Lorentz force+N2:

$$\vec{F} = q(\vec{v} \times \vec{B} + \vec{E}).$$

$$\ddot{\vec{r}}(t) = \frac{q}{m}(\dot{\vec{r}}(t) \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)).$$

Theory: Classical non-relativistic particles.

- Some repetition from Electrodynamics.
- The Lorentz force+N2:

$$\vec{F} = q(\vec{v} \times \vec{B} + \vec{E}).$$
$$\ddot{\vec{r}}(t) = \frac{q}{m}(\dot{\vec{r}}(t) \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)).$$

- Could use potentials $\phi(\vec{r}, t)$ $\vec{A}(\vec{r}, t)$ and Hamiltonian, or Lagrangian.

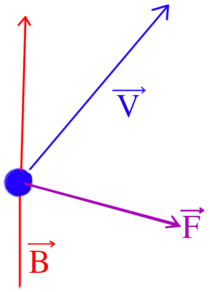
Theory: Classical non-relativistic particles.

- Some repetition from Electrodynamics.
- The Lorentz force+N2:

$$\vec{F} = q(\vec{v} \times \vec{B} + \vec{E}).$$
$$\ddot{\vec{r}}(t) = \frac{q}{m}(\dot{\vec{r}}(t) \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)).$$

- Could use potentials $\phi(\vec{r}, t)$ $\vec{A}(\vec{r}, t)$ and Hamiltonian, or Lagrangian.
- Other systems would have other differential equations.

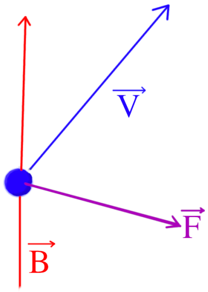
Known results, cyclotron motion $\vec{B}$ fields.



Known results, cyclotron motion $\vec{B}$ fields.

- Magnetic forces do no work:

$$dW_{\vec{B}} = \vec{F}_B \cdot d\vec{r} \propto (\vec{v} \times \vec{B}) \cdot \vec{v} = 0.$$



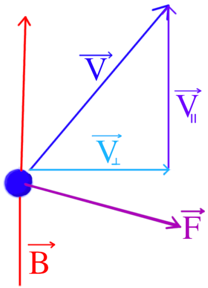
Known results, cyclotron motion $\vec{B}$ fields.

- Magnetic forces do no work:

$$dW_{\vec{B}} = \vec{F}_B \cdot d\vec{r} \propto (\vec{v} \times \vec{B}) \cdot \vec{v} = 0.$$

- $(\vec{v} = \vec{v}_\perp + \vec{v}_{||})$:

$$|\vec{F}_B| = |q(\vec{v} \times \vec{B})| = |qv_\perp B|.$$



Known results, cyclotron motion $\vec{B}$ fields.

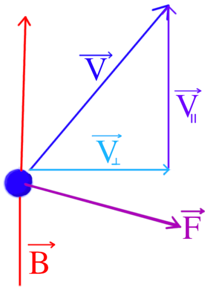
- Magnetic forces do no work:

$$dW_{\vec{B}} = \vec{F}_B \cdot d\vec{r} \propto (\vec{v} \times \vec{B}) \cdot \vec{v} = 0.$$

- $(\vec{v} = \vec{v}_\perp + \vec{v}_{||})$:

$$|\vec{F}_B| = |q(\vec{v} \times \vec{B})| = |qv_\perp B|.$$

- Same as Centripetal force: Cyclotron motion.



Known results, cyclotron motion $\vec{B}$ fields.

- Magnetic forces do no work:

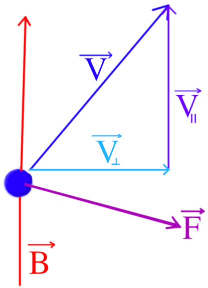
$$dW_{\vec{B}} = \vec{F}_B \cdot d\vec{r} \propto (\vec{v} \times \vec{B}) \cdot \vec{v} = 0.$$

- $(\vec{v} = \vec{v}_\perp + \vec{v}_\parallel)$:

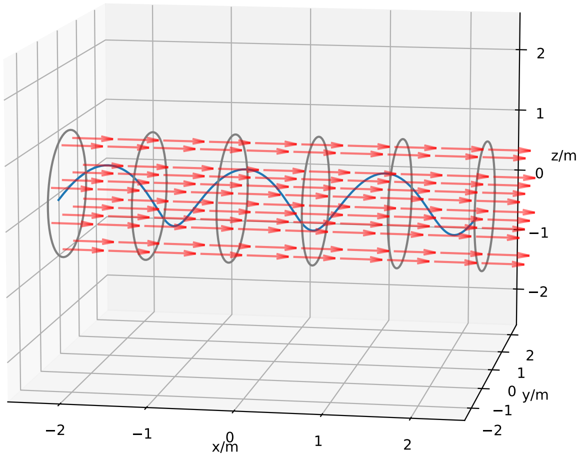
$$|\vec{F}_B| = |q(\vec{v} \times \vec{B})| = |q v_\perp B|.$$

- Same as Centripetal force: Cyclotron motion.
- Cyclotron radius and frequency:

$$R = \frac{v_\perp m}{|q|B} \quad \omega_c = \frac{|q|B}{m}.$$



What we expect to see.

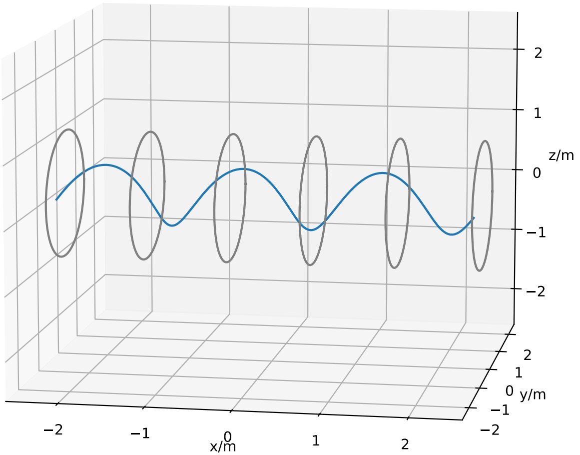


“Cyclotron motion”

Solenoid with $N = 1000$ turns per m, $I = 5\,\mathrm{A}$, $r = 1\,\mathrm{m}$, $|\vec{B}| \approx 6\,\mathrm{mT}$.

Proton with $|v| \approx 3.195 \times 10^5\,\mathrm{m/s}$

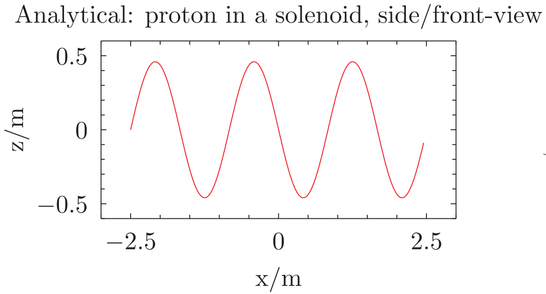
What we expect to see.

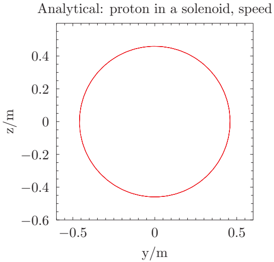


"Cyclotron motion"

$$R \approx 0.5\,\mathrm{m}\sin(\theta) \qquad T_c = \frac{2\pi}{\omega_c} \approx 10\,\mu\mathrm{s}$$

What we expect to see.

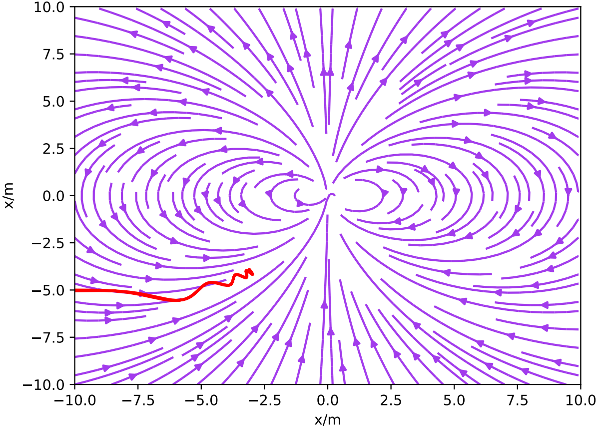




"Cyclotron motion"

$$R \approx 0.5\,\mathrm{m}\sin(\theta) \qquad T_c = \frac{2\pi}{\omega_c} \approx 10\,\mu\mathrm{s}$$

What we expect to see.



(Actually from my simulation)

Ordinary differential equation's.

- Sources: Zeigler et al. Theory of Modeling and Simulation (Third edition) chapter 3.
- We have:

$$\ddot{\vec{r}}(t) = \frac{q}{m}(\dot{\vec{r}}(t) \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)).$$

Ordinary differential equation's.

- Sources: Zeigler et al. Theory of Modeling and Simulation (Third edition) chapter 3.
- We have:

$$\ddot{\vec{r}}(t) = \frac{q}{m}(\dot{\vec{r}}(t) \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)).$$

- Algorithms exists for ODEs:

$$\dot{\mathbf{X}} = \mathbf{f}_{ode}(\mathbf{X}(t), t).$$

Ordinary differential equation's.

- Sources: Zeigler et al. Theory of Modeling and Simulation (Third edition) chapter 3.
- We have:

$$\ddot{\vec{r}}(t) = \frac{q}{m}(\dot{\vec{r}}(t) \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)).$$

- Algorithms exists for ODEs:

$$\dot{\mathbf{X}} = \mathbf{f}_{ode}(\mathbf{X}(t), t).$$

- Here:

$$\mathbf{X} = \begin{pmatrix} \vec{r} \\ \dot{\vec{r}} \end{pmatrix} \quad \mathbf{f}_{ode}(\mathbf{X}, t) = \begin{pmatrix} \dot{\vec{r}} \\ \frac{q}{m}(\dot{\vec{r}} \times \vec{B}(\vec{r}, t) + \vec{E}(\vec{r}, t)) \end{pmatrix}.$$

Solving differential equations.

- We know only $\mathbf{X}(t_i)$ and $t_i = t_0 + i\Delta t$ and $\mathbf{f}_{ode}$.

- Bernard P. Zeigler et al. Theory of Modeling and Simulation (Third edition), chapter 3.

Solving differential equations.

- We know only $\mathbf{X}(t_i)$ and $t_i = t_0 + i\Delta t$ and $\mathbf{f}_{ode}$.
- Can we find $\mathbf{X}(t_i + \Delta t)$ for $\Delta t > 0$?

- Bernard P. Zeigler et al. Theory of Modeling and Simulation (Third edition), chapter 3.

Solving differential equations.

- We know only $\mathbf{X}(t_i)$ and $t_i = t_0 + i\Delta t$ and $\mathbf{f}_{ode}$.
- Can we find $\mathbf{X}(t_i + \Delta t)$ for $\Delta t > 0$?
- Euler's Method:

$$\mathbf{X}(t_i + \Delta t) = \mathbf{X}(t_i) + \Delta t \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i).$$

- Multiple steps $t_0, t_1 = t_0 + \Delta t, \ldots t_i, \ldots$.

- Bernard P. Zeigler et al. Theory of Modeling and Simulation (Third edition), chapter 3.

Solving differential equations.

- We know only $\mathbf{X}(t_i)$ and $t_i = t_0 + i\Delta t$ and $\mathbf{f}_{ode}$.
- Can we find $\mathbf{X}(t_i + \Delta t)$ for $\Delta t > 0$?
- Euler's Method:

$$\mathbf{X}(t_i + \Delta t) = \mathbf{X}(t_i) + \Delta t \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i).$$

- Multiple steps $t_0, t_1 = t_0 + \Delta t, \ldots t_i, \ldots$.
- Can this be justified?
- Bernard P. Zeigler et al. Theory of Modeling and Simulation (Third edition), chapter 3.

Why does this work? What is the error.

- Multiple justifications for why.

Why does this work? What is the error.

- Multiple justifications for why.
- First 2 terms in Taylor series Zeigler et al.:

$$\mathbf{X}(t_i + \Delta t) = \mathbf{X}(t_i) + \Delta t \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) + \Delta t^2 \ldots + \ldots$$

Why does this work? What is the error.

- Multiple justifications for why.
- First 2 terms in Taylor series Zeigler et al.:

$$\mathbf{X}(t_i + \Delta t) = \mathbf{X}(t_i) + \Delta t \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) + \Delta t^2 \ldots + \ldots$$

- Local error $O((\Delta t)^2)$.

Why does this work? What is the error.

- Multiple justifications for why.
- First 2 terms in Taylor series Zeigler et al.:

$$\mathbf{X}(t_i + \Delta t) = \mathbf{X}(t_i) + \Delta t \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) + \Delta t^2 \ldots + \ldots$$

- Local error $O((\Delta t)^2)$.
- We want "better" (larger Δt (fewer steps, fewer calls), same error).

Why does this work? What is the error.

- Multiple justifications for why.
- First 2 terms in Taylor series Zeigler et al.:

$$\mathbf{X}(t_i + \Delta t) = \mathbf{X}(t_i) + \Delta t \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) + \Delta t^2 \ldots + \ldots$$

- Local error $O((\Delta t)^2)$.
- We want "better" (larger Δt (fewer steps, fewer calls), same error).
- Argument suggests we need $\dot{\mathbf{f}}_{ode}, \ddot{\mathbf{f}}_{ode}, \ldots$, we don't!

The Runge Kutta steppers.

- In general:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t_{i+1}} \dot{\mathbf{X}}(t)dt.$$

- L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, 2017, chapter 8:

The Runge Kutta steppers.

- In general:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt.$$

- L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, 2017, chapter 8:

The Runge Kutta steppers.

- In general:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt = \Delta t \mathbf{f}_{ode}(\mathbf{X}(\tau), \tau).$$

- *Mean Value theorem for integrals* $t_i \leq \tau \leq t_{i+1}$.

- L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, 2017, chapter 8:

The Runge Kutta steppers.

- In general:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt = \Delta t \mathbf{f}_{ode}(\mathbf{X}(\tau), \tau).$$

- *Mean Value theorem for integrals* $t_i \leq \tau \leq t_{i+1}$.
- More generally, (*Explicit* and *single step*), Runge-Kutta family:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t'} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt + \ldots \int_{t^{(m)}}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt.$$

- L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, 2017, chapter 8:

The Runge Kutta steppers.

- In general:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt = \Delta t \mathbf{f}_{ode}(\mathbf{X}(\tau), \tau).$$

- *Mean Value theorem for integrals* $t_i \leq \tau \leq t_{i+1}$.
- More generally, (*Explicit* and *single step*), Runge-Kutta family:

$$\begin{aligned}\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) &= \int_{t_i}^{t'} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt + \dots \int_{t^{(m)}}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt, \\ &= \sum_{j=1}^{m} \Delta t_j \mathbf{f}_{ode}(\mathbf{X}(\tau_j), \tau_j).\end{aligned}$$

- L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, 2017, chapter 8:

The Runge Kutta steppers.

- In general:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \int_{t_i}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt = \Delta t \mathbf{f}_{ode}(\mathbf{X}(\tau), \tau).$$

- *Mean Value theorem for integrals* $t_i \leq \tau \leq t_{i+1}$.
- More generally, (*Explicit* and *single step*), Runge-Kutta family:

$$\begin{aligned}\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) &= \int_{t_i}^{t'} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt + \dots \int_{t^{(m)}}^{t_{i+1}} \mathbf{f}_{ode}(\mathbf{X}(t), t)dt, \\ &= \sum_{j=1}^{m} \Delta t_j \mathbf{f}_{ode}(\mathbf{X}(\tau_j), \tau_j).\end{aligned}$$

- Cant guess anything ... unless $\mathbf{X}(t)$ is polynomial.
- L. Zheng, X. Zhang, Modeling and Analysis of Modern Fluid Problems, 2017, chapter 8:

Runge Kutta methods.

- More commonly written:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j \mathbf{K}_j$$

- Martha L. Abell, James P. Braselton, Differential Equations with Mathematica (Fourth Edition), 2016.

Runge Kutta methods.

- More commonly written:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j \mathbf{K}_j$$

- Pick $\mathbf{K}_j$ so and pick an m:

$$\begin{aligned}
\mathbf{K}_1 &= \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) \\
\mathbf{K}_2 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t a_{21}\mathbf{K}_1, t_i + c_2 \Delta t) \\
\mathbf{K}_3 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t a_{31}\mathbf{K}_1 + \Delta t a_{32}\mathbf{K}_2, t_i + c_3 \Delta t) \\
&\vdots
\end{aligned}$$

- Martha L. Abell, James P. Braselton, Differential Equations with Mathematica (Fourth Edition), 2016.

Runge Kutta methods.

- More commonly written:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j \mathbf{K}_j$$

- Pick $\mathbf{K}_j$ so and pick an m:

$$\begin{aligned}
\mathbf{K}_1 &= \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) \\
\mathbf{K}_2 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t a_{21}\mathbf{K}_1, t_i + c_2 \Delta t) \\
\mathbf{K}_3 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t a_{31}\mathbf{K}_1 + \Delta t a_{32}\mathbf{K}_2, t_i + c_3 \Delta t) \\
&\vdots
\end{aligned}$$

- Pick parameters to be *exact* for p'th order polynomial.

- Martha L. Abell, James P. Braselton, Differential Equations with Mathematica (Fourth Edition), 2016.

Runge Kutta methods.

- More commonly written:

$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j \mathbf{K}_j$$

- Pick $\mathbf{K}_j$ so and pick an m:

$$\begin{aligned}
\mathbf{K}_1 &= \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i) \\
\mathbf{K}_2 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t a_{21}\mathbf{K}_1, t_i + c_2 \Delta t) \\
\mathbf{K}_3 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t a_{31}\mathbf{K}_1 + \Delta t a_{32}\mathbf{K}_2, t_i + c_3 \Delta t) \\
&\vdots
\end{aligned}$$

- Pick parameters to be *exact* for p'th order polynomial.
- Taylor series analogy: Local error $O((\Delta t)^{p+1})$.
- Martha L. Abell, James P. Braselton, Differential Equations with Mathematica (Fourth Edition), 2016.

The General explicit Runge Kutta method.

- Expressed in Butcher tableu:

$$
\begin{array}{c|cccc}
c_1 = 0 & & & \\
c_2 & a_{21} & & \\
c_3 & a_{31} & a_{32} & \\
c_n & a_{n1} & a_{n2} & \dots \\
\hline
 & b_1 & b_2 & \dots
\end{array}
$$

Higher order methods.

- 2nd order (Heun's method):

$$\begin{aligned}
\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) &= \frac{\Delta t}{2}(\mathbf{K}_1 + \mathbf{K}_2), \\
\mathbf{K}_1 &= \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i), \\
\mathbf{K}_2 &= \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t\mathbf{K}_1, t_i + \Delta t).
\end{aligned}$$

Higher order methods.

- 4th order, often simply called the Runge Kutta method:

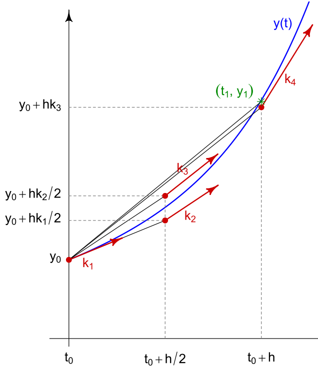
$$\mathbf{X}(t_{i+1}) - \mathbf{X}(t_i) = \frac{\Delta t}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4)$$

$$\mathbf{K}_1 = \mathbf{f}_{ode}(\mathbf{X}(t_i), t_i)$$

$$\mathbf{K}_2 = \mathbf{f}_{ode}(\mathbf{X}(t_i) + \frac{\Delta t}{2}\mathbf{K}_1, t_i + \frac{\Delta t}{2})$$

$$\mathbf{K}_3 = \mathbf{f}_{ode}(\mathbf{X}(t_i) + \frac{\Delta t}{2}\mathbf{K}_2, t_i + \frac{\Delta t}{2})$$

$$\mathbf{K}_4 = \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t\mathbf{K}_3, t_i + \Delta t).$$



Does it work.

- Test, same proton in a solenoid use same starting conditions with:

$$R \approx 0.45\,\mathrm{m} \quad T_c = \frac{2\pi}{\omega_c} \approx 10\,\mu\mathrm{s}.$$

Does it work.

- Test, same proton in a solenoid use same starting conditions with:

$$R \approx 0.45\,\mathrm{m} \quad T_c = \frac{2\pi}{\omega_c} \approx 10\,\mu\mathrm{s}.$$

- Compare Analytic, Euler, Runge-Kutta 4. At different Δt_{equiv}.

Does it work.

- Test, same proton in a solenoid use same starting conditions with:

$$R \approx 0.45\,\mathrm{m} \quad T_c = \frac{2\pi}{\omega_c} \approx 10\,\mu\mathrm{s}.$$

- Compare Analytic, Euler, Runge-Kutta 4. At different Δt_{equiv}.
- $\Delta t_{equiv} = 4\Delta t$ for 4th order method.

Does it work.

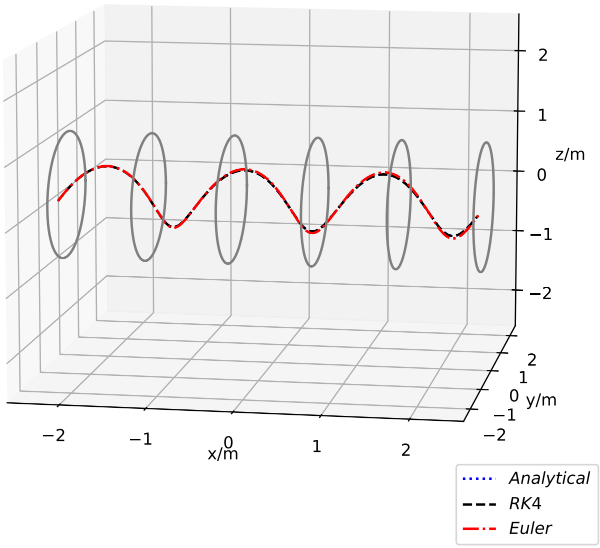
- Test, same proton in a solenoid use same starting conditions with:

$$R \approx 0.45\,\mathrm{m} \quad T_c = \frac{2\pi}{\omega_c} \approx 10\,\mu\mathrm{s}.$$

- Compare Analytic, Euler, Runge-Kutta 4. At different Δt_{equiv}.
- $\Delta t_{equiv} = 4\Delta t$ for 4th order method.
- Check error on $|\vec{v}|$, $R = \sqrt{y^2 + z^2}$.

At a glance, 3D view.

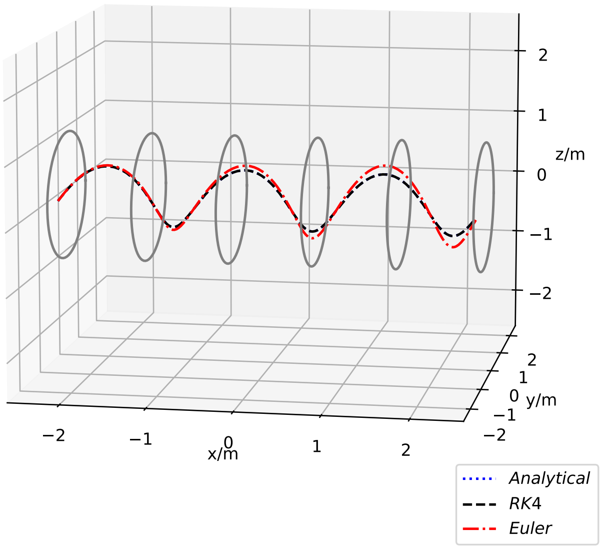
$\Delta t_{equiv} \approx 1/1000\, T_c$



3129 steps

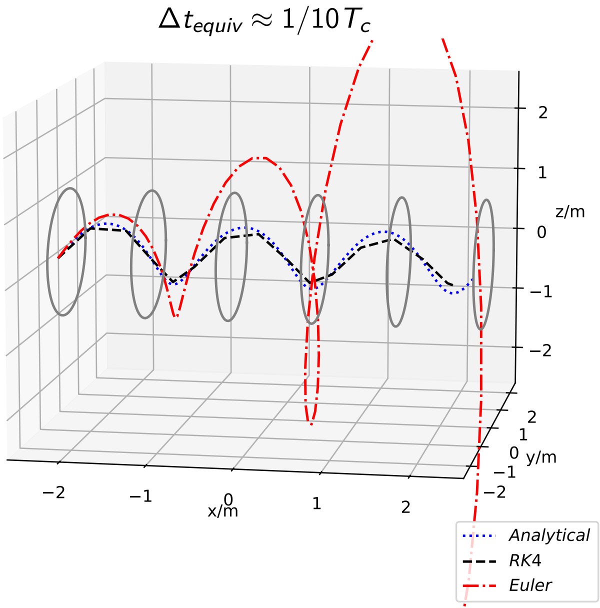
At a glance, 3D view.

$$\Delta t_{equiv} \approx 1/100\, T_c$$



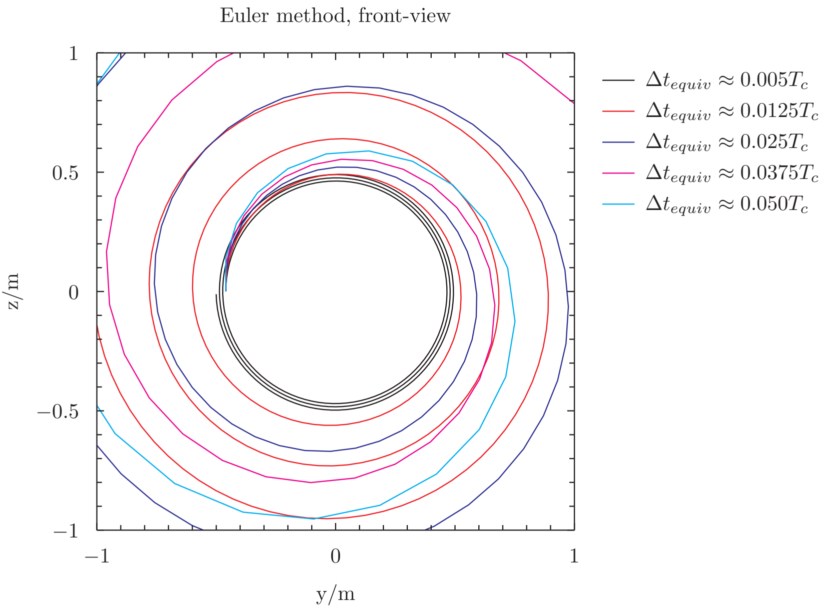
312 steps

At a glance, 3D view.

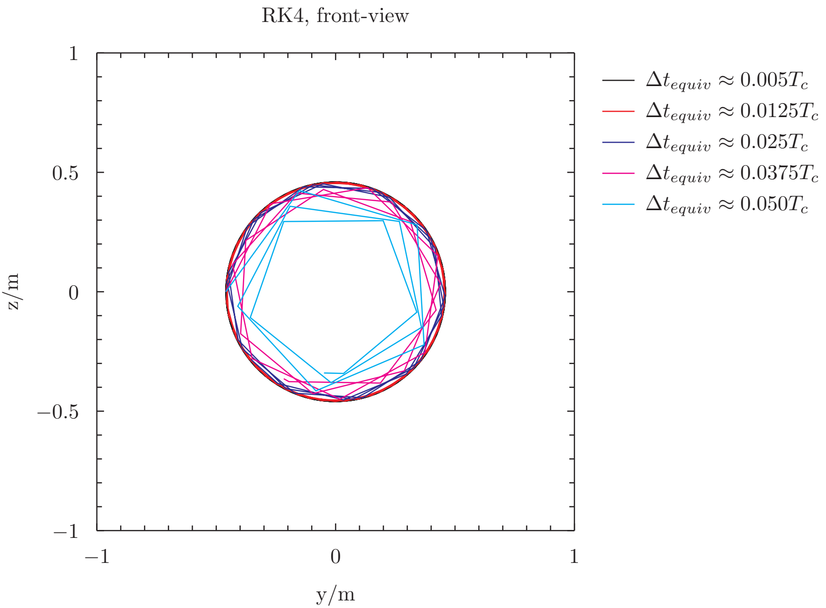


31 steps.

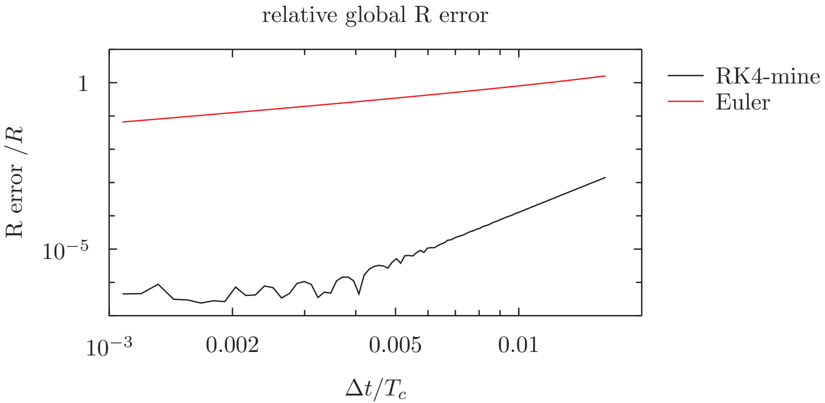
At a glance, front view, no border.



At a glance, front view, no border.



Error as function of Δt_{equiv}.



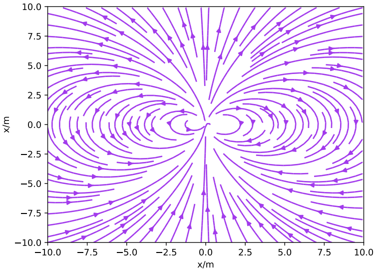
What is not to like?

What is not to like?

- Usually don't know error.
- Hard to pick Δt , and may change:

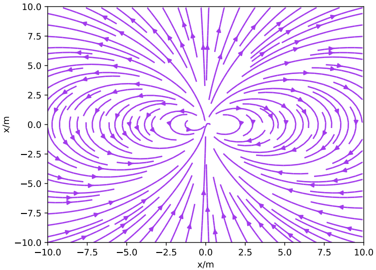
What is not to like?

- Usually don't know error.
- Hard to pick Δt , and may change:
- Inhomogeneous or time dependent fields (here, a true dipole).



What is not to like?

- Usually don't know error.
- Hard to pick Δt , and may change:
- Inhomogeneous or time dependent fields (here, a true dipole).
- Let the computer pick Δt for error.



Embedded Runge-Kutta algorithms.

- Need estimate for error.

Embedded Runge-Kutta algorithms.

- Need estimate for error.
- Runge-Kutta step $\mathbf{X}^{(p)}(t_{i+1})$ will converge to $\mathbf{X}(t_{i+1})$.

Embedded Runge-Kutta algorithms.

- Need estimate for error.
- Runge-Kutta step $\mathbf{X}^{(p)}(t_{i+1})$ will converge to $\mathbf{X}(t_{i+1})$.
- Use 2 different order methods, $\mathbf{X}^{(p-1)}(t_{i+1})$, $\mathbf{X}^{(p)}(t_{i+1})$.

Embedded Runge-Kutta algorithms.

- Need estimate for error.
- Runge-Kutta step $\mathbf{X}^{(p)}(t_{i+1})$ will converge to $\mathbf{X}(t_{i+1})$.
- Use 2 different order methods, $\mathbf{X}^{(p-1)}(t_{i+1})$, $\mathbf{X}^{(p)}(t_{i+1})$.
- Approximate "Error" as $\mathbf{E}_j = |\mathbf{X}_j^{(p-1)}(t_{i+1}) - \mathbf{X}_j^{(p)}(t_{i+1})|$.

Embedded Runge-Kutta algorithms.

- Need estimate for error.
- Runge-Kutta step $\mathbf{X}^{(p)}(t_{i+1})$ will converge to $\mathbf{X}(t_{i+1})$.
- Use 2 different order methods, $\mathbf{X}^{(p-1)}(t_{i+1})$, $\mathbf{X}^{(p)}(t_{i+1})$.
- Approximate "Error" as $\mathbf{E}_j = |\mathbf{X}_j^{(p-1)}(t_{i+1}) - \mathbf{X}_j^{(p)}(t_{i+1})|$.
- Adjust step size to keep the error(s) small (Implementations differ!).

  Dormand, J. R.; Prince, P. J. (1980), "A family of embedded Runge-Kutta formulae", Journal of Computational and AppliMathematics.

Runge-Kutta Dormand Prince 5 (4).

- `ode45` in Matlab, `scipy.solve_ivp` in Python , `RungeKutta_dopri5` in `boost::odeint`.

$$\mathbf{X}^{(5)}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j^{(5)} \mathbf{K}_j$$

$$\mathbf{X}^{(4)}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j^{(4)} \mathbf{K}_j.$$

$$\mathbf{K}_j = \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t \sum_{k}^{j-1} a_{kj} \mathbf{K}_j, t_i + c_3 \Delta t).$$

Runge-Kutta Dormand Prince 5 (4).

- `ode45` in Matlab, `scipy.solve_ivp` in Python , `RungeKutta_dopri5` in `boost::odeint`.

$$\mathbf{X}^{(5)}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j^{(5)} \mathbf{K}_j$$

$$\mathbf{X}^{(4)}(t_{i+1}) - \mathbf{X}(t_i) = \Delta t \sum_{j=1}^{m} b_j^{(4)} \mathbf{K}_j.$$

$$\mathbf{K}_j = \mathbf{f}_{ode}(\mathbf{X}(t_i) + \Delta t \sum_{k}^{j-1} a_{kj} \mathbf{K}_j, t_i + c_3 \Delta t).$$

- 7 $\mathbf{K}_j$'s (actually 6 by clever re-usage) (First same as last principle).
  Dormand, J. R.; Prince, P. J. (1980), "A family of embedded Runge-Kutta formulae", Journal of Computational and Applied Mathematics

Butcher Tableu of Dormand Prince (5) 4.

$$
\begin{array}{c|ccccccc}
c_i & a_{ij} & \dots & & & & & \\
\hline
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{40}{45} & -\frac{56}{15} & -\frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
/b^{(5)} & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
/b^{(4)} & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array}
$$

Runge-Kutta Dormand Prince 5 (4).

- Step size correction (Dormand Prince):

$$\Delta t_{new} = 0.9 \Delta t_{old} \left[\frac{\delta}{||\mathrm{E}||} \right]^{\frac{1}{p+1}}$$

Runge-Kutta Dormand Prince 5 (4).

- Step size correction (Dormand Prince):

$$\Delta t_{new} = 0.9 \Delta t_{old} \left[\frac{\delta}{||\mathrm{E}||} \right]^{\frac{1}{p+1}}$$

- Step size correction (Me):

$$\Delta t_{new} = \min_j \left(0.9 \Delta t_{old} \left[\frac{\delta_j}{\mathrm{E}_j} \right]^{\frac{1}{p}} \right), \quad \mathrm{E}_j > \delta_j : \text{reject}$$

$$\delta_j = \min(\delta_{abs}, |\mathbf{X}_j(t_i)| \delta_{rel}) \sqrt{\frac{\Delta t_{old}}{T}}.$$

Runge-Kutta Dormand Prince 5 (4).

- Step size correction (Dormand Prince):

$$\Delta t_{new} = 0.9 \Delta t_{old} \left[\frac{\delta}{||\mathrm{E}||} \right]^{\frac{1}{p+1}}$$

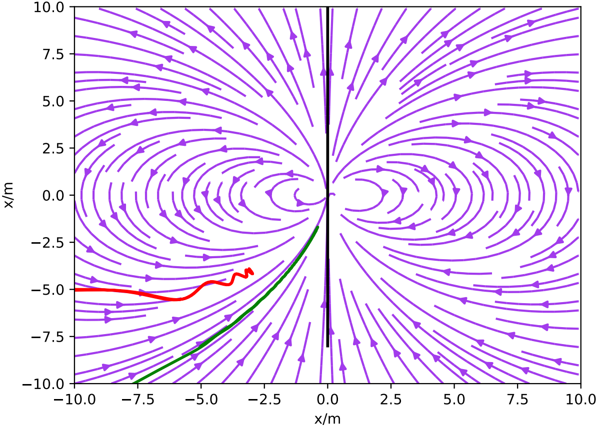
- Step size correction (Me):

$$\Delta t_{new} = \min_j \left(0.9 \Delta t_{old} \left[\frac{\delta_j}{\mathrm{E}_j} \right]^{\frac{1}{p}} \right), \quad \mathrm{E}_j > \delta_j : \text{reject}$$

$$\delta_j = \min(\delta_{abs}, |\mathbf{X}_j(t_i)| \delta_{rel}) \sqrt{\frac{\Delta t_{old}}{T}}.$$

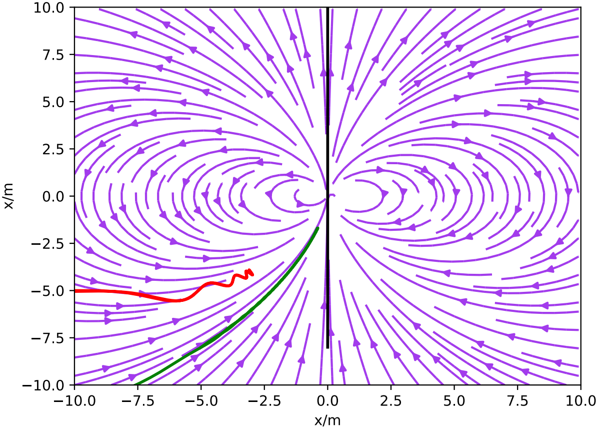
- Scale by step size $\frac{\Delta t_{old}}{T}$, "Fail safe" $\sqrt{\ldots}$.
  Dormand, J. R.; Prince, P. J. (1980), "A family of embedded Runge-Kutta formulae", Journal of Computational and Applied Mathematics.

Does it work?



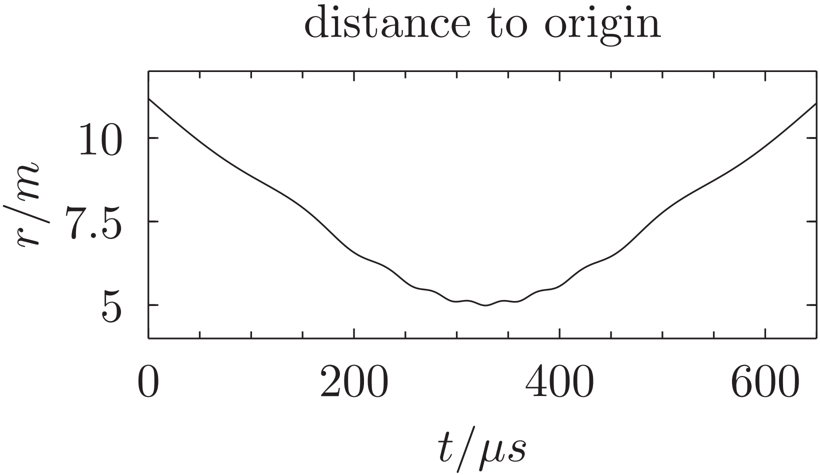
My version relative and absolute error 10^{-6}. 313 steps (+ 25 rejected).

Does it work?

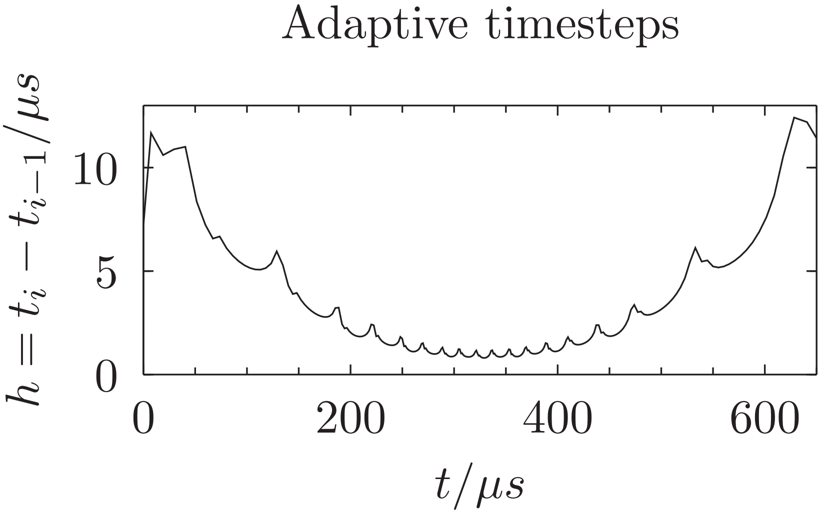


Odeint library relative and absolute error 10^{-7}. 249 steps.

Does it work?



Does it work?



My version, adaptive step size.

Conclusion.

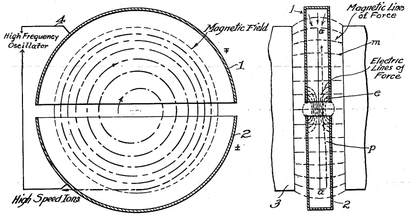
- Simulating particles in electric and magnetic fields.

Conclusion.

- Simulating particles in electric and magnetic fields.
- Numerically solving ordinary differential equations.
- Can easily be generalized to other systems.

Example, cyclotron.

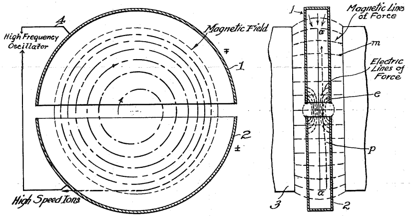
- Electric field accelerates, magnetic contains.



Ernest O. Lawrence, 1934, U.S. Patent 1,948,384; image in Public Domain.

Example, cyclotron.

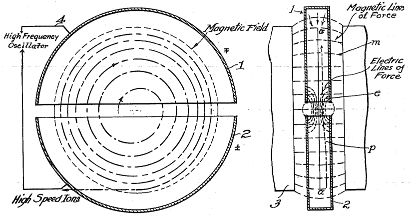
- Electric field accelerates, magnetic contains.
- Single gab, oscillating field.



Ernest O. Lawrence, 1934, U.S. Patent 1,948,384; image in Public Domain.

Example, cyclotron.

- Electric field accelerates, magnetic contains.
- Single gab, oscillating field.
- Uses classical Cyclotron frequency.

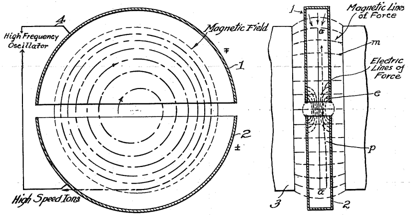


Ernest O. Lawrence, 1934, U.S. Patent 1,948,384; image in Public Domain.

Example, cyclotron.

- Electric field accelerates, magnetic contains.
- Single gab, oscillating field.
- Uses classical Cyclotron frequency.
- Analytical final speed, in principle path.

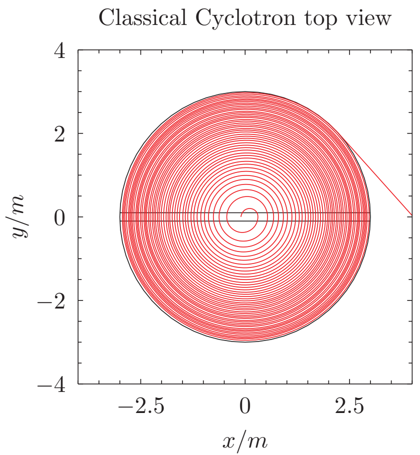
$$\frac{R|q|B}{m} = v_{\perp}.$$



Ernest O. Lawrence, 1934, U.S. Patent 1,948,384; image in Public Domain.

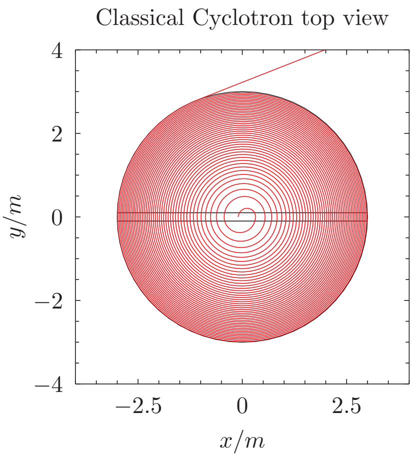
can it be simulated.

- with fixed step size, looks bad 4999 points.



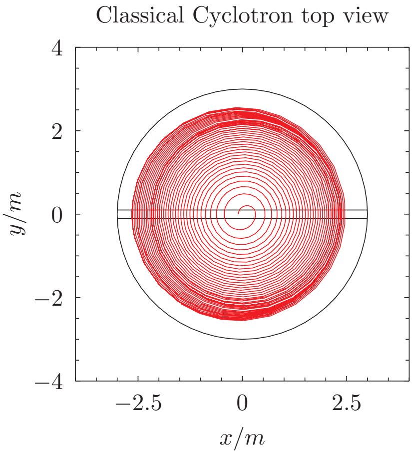
can it be simulated.

- with fixed step size, looks bad 4999 points.
- my adabtive method, error: 10^{-6}.
- Looks great but 2 million points.



can it be simulated.

- with fixed step size, looks bad 4999 points.
- my adabtive method, error: 10^{-6}.
- Looks great but 2 million points.
- odeint library.



can it be simulated.

- with fixed step size, looks bad 4999 points.
- my adabtive method, error: 10^{-6}.
- Looks great but 2 million points.
- odeint library.
- non-continuous ode are bad.