Projeto 3 - Cargas em movimento

Código e Resultados

Bruno Nicolau, Caio B. Naves
Henrique Felix, Ian G. Pauli

07/10/2019

IFSC-USP
Instituto de Física de São Carlos

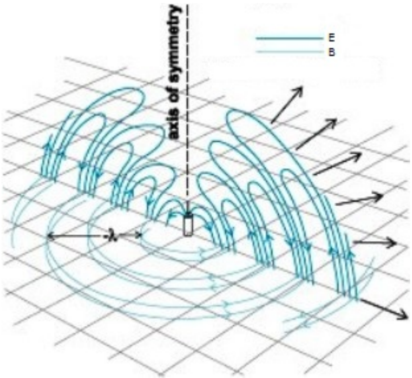
Problema

Carga q com movimento:
$x(t) = A\cos(\omega t + \phi)$

Efeito relativístico $\omega A \sim c$

Calcular **E**, **B** considerando
$t_{ret} \rightarrow \text{Tempo Retardado}$

Potenciais de Liénard-Wiechert



Resultado Esperado

Solução

$$\mathbf{r}(x, y, z, t) \to \text{Posição da Carga}$$
$$\mathbf{R}(x, y, z) \to \text{Posição no Espaço}$$

$$\text{Malha} \to (\mathbf{R} \cdot \hat{\mathbf{x}}, \mathbf{R} \cdot \hat{\mathbf{y}})$$

$$t_{ret} = t - |\mathbf{R} - \mathbf{r}(t_{ret})|/c$$
$$f(t_{ret}) = t - t_{ret} - |\mathbf{R} - \mathbf{r}(t_{ret})|/c$$

Método de Newton

Raiz de $f(t_{ret})$

$$\mathbf{E}(\mathbf{R}, t) = -\boldsymbol{\nabla} V - \frac{\partial \mathbf{A}}{\partial t}$$

$$\mathbf{B} = \nabla \times \mathbf{A}$$

Calibre de Lorenz

$$\nabla \cdot \mathbf{A} + \mu_0 \epsilon_0 \frac{\partial V}{\partial t} = 0$$

---

d'Alembertiano

$$\Box^2 \equiv \nabla^2 + \mu_0 \epsilon_0 \frac{\partial^2}{\partial t^2}$$

Potenciais de Liénard-Wiechert

$$\Box^2 V = -\frac{\rho}{\epsilon_0},\ \Box^2 \mathbf{A} = -\mu_0 \mathbf{J}$$

$$V = \frac{q}{r_{ret}(1 - \hat{\mathbf{r}}_{ret} \cdot \hat{\mathbf{v}}_{ret}/c)}$$

$$\mathbf{A} = \frac{q(\hat{\mathbf{v}}_{ret}/c)}{r_{ret}(1 - \hat{\mathbf{r}}_{ret} \cdot \hat{\mathbf{v}}_{ret}/c)}$$

$$\mathbf{E}(\mathbf{R}, t) = -\boldsymbol{\nabla} V - \frac{\partial \mathbf{A}}{\partial t}$$

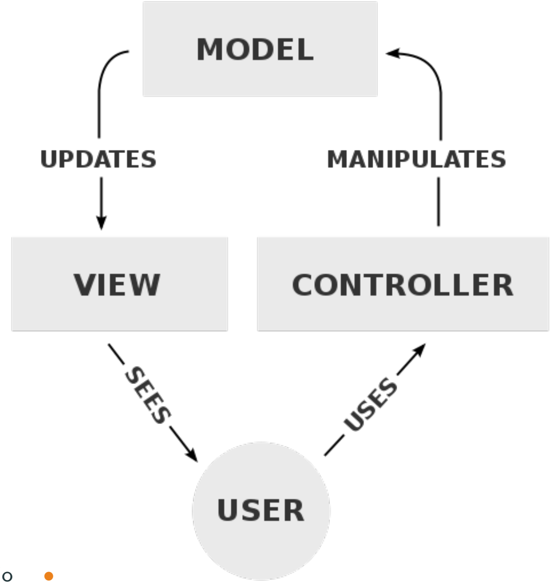
$$\mathbf{B} = \nabla \times \mathbf{A}$$

$$\mathbf{u}_{ret} \equiv c\hat{\mathbf{r}}_{ret} - \hat{\mathbf{v}}_{ret}$$

$$\mathbf{E}(\mathbf{R}, t) = \frac{q r_{ret}}{(\mathbf{r}_{ret} \cdot \mathbf{u}_{ret})^3} \left[\mathbf{u}_{ret}(c^2 - v_{ret}^2) + \mathbf{r}_{ret} \times (\mathbf{u}_{ret} \times \mathbf{a}_{ret})\right]$$

$$\mathbf{B}(\mathbf{R}, t) = \hat{\mathbf{r}}_{ret} \times \mathbf{E}(\mathbf{R}, t)$$

$$\|\mathbf{E}(\mathbf{R}, t)\| \stackrel{N}{=} \|\mathbf{B}(\mathbf{R}, t)\|$$


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

Modelo

```python
class LienardWiechertModel:
    def __init__(self, settings):
        (...)
        self.frames = []
        self.time = 0
        self.time_step = settings.time_step
        (...)
        self.charge = MovingParticle(settings)

    def calculate(self):
        X, Y = meshgrid(self.x_axis, self.y_axis, indexing='xy')
        E = vectorize(self.electric_field, excluded=['self'])
        self.frames.append(E(X,Y))

    def retarded_time(self, R, r):
        f = lambda t_ret: self.time - t_ret - norm(R - r(t_ret))/c
        t_ret = optimize.newton(f, self.time)
        return t_ret
```

Modelo

```python
def electric_field(self, x, y):
    dx, dy, dz = self.translation_x, self.translation_y, \
                                            self.translation_z
    R = array([x+dx,y+dy,dz])

    r = self.charge.position
    v = self.charge.velocity
    a = self.charge.aceleration

    t_ret = self.retarded_time(R, r)
    r_ret = R - r(t_ret)
    u_ret = c * r_ret / norm(r_ret) - v(t_ret)
    a_ret = a(t_ret)
    v_ret = v(t_ret)

    E = norm(r_ret)/(dot(r_ret,u_ret))**3*(u_ret*(c**2-norm(v_ret)**2)
                                          + cross(r_ret,cross(u_ret,a_ret)))
    return norm(E)
```

Visualização

```
class LienardWiechertView:
    def __init__(self, root, model, settings):
        self.model = model
        self.settings = settings
        (...)

    def play(self, event):
        (...)

    def calculate(self, event):
        (...)
        while self.model.time <= time_total:
            self.model.calculate()
            self.model.step()

        (...)

    def save(self, event):
        (...)
```

Controle

```python
class LienardWiechertApp:
     def __init__(self):
        # Recebe Modelo
        self.model = LienardWiechertModel()

        # Recebe Visualização
        self.view  = LienardWiechertView(self.model)

    def run(self):
        # Executa simulação
```

Vamos agora mostrar o programa rodando

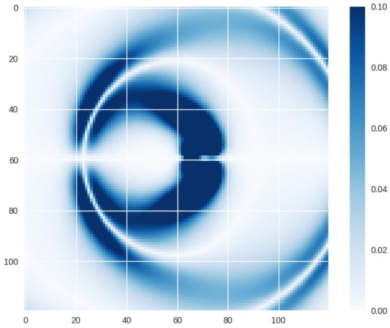
Alguns resultados obtidos

Parametros do movimento da carga

- Amplitude: 1 m
- Frequência Angular: 2×10^{8} rad/s^{-1}
- tais que $\omega A \sim c$

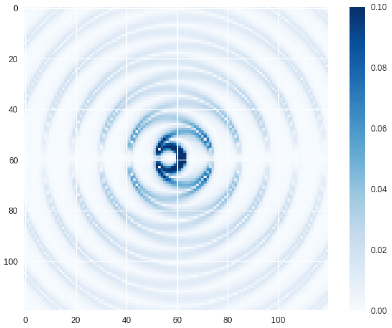
Perto da carga

- Tamanho:
  20x20 m
- Divisões na malha:
  120x120
- Tempo de simulação:
  30 ns
- Intervalo de tempo:
  1 ns



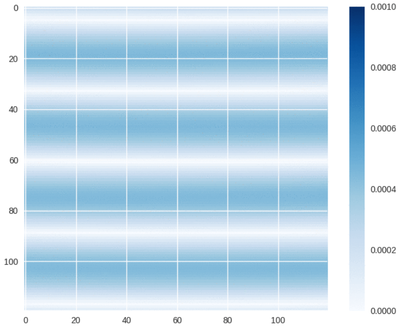
Longe da carga

- Tamanho:
  100x100 m
- Divisões na malha:
  120x120
- Tempo de simulação:
  30 ns
- Intervalo de tempo:
  1 ns



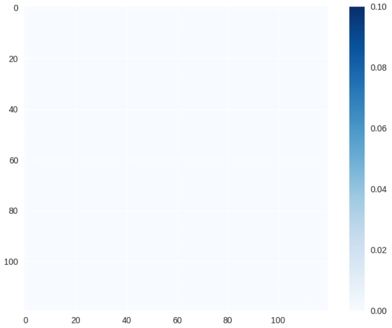
Distante do eixo de oscilação - 1000 u.c.

- Tamanho:
  20x20 m
- Divisões na malha:
  120x120
- Tempo de simulação:
  30 ns
- Intervalo de tempo:
  1 ns



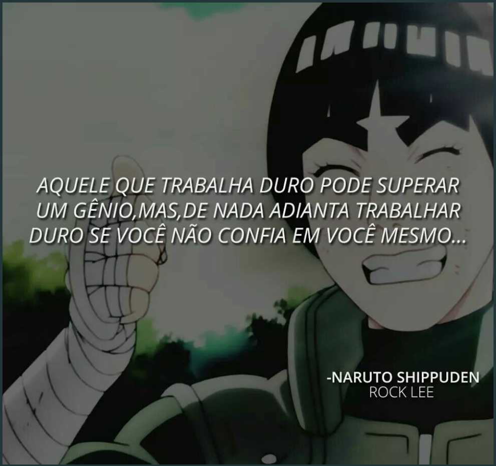
Pulso

- Tamanho:
  20x20 m
- Divisões na malha:
  120x120
- Tempo de simulação:
  50 ns
- Intervalo de tempo:
  1 ns



Conclusões

- Organização do fluxo de trabalho usando Git e MVC.
- Conseguimos visualizar o comportamento temporal e espacial do campo $\mathbf{E}(x, y)$ usando uma interface gráfica simples.
- Campo próximo e campo distante da carga.



Muito obrigado e bom dia!