Introdução a Métodos Computacionais em Física

Leonardo Cabral

9 de agosto de 2019

Roteiro da aula

- Sistemas físicos são frequentemente regidos por equações diferenciais:

- Sistemas físicos são frequentemente regidos por equações diferenciais:
  - $m\frac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{r}, \mathbf{v}, t),$ 2^{a} lei de Newton

- Sistemas físicos são frequentemente regidos por equações diferenciais:
  - $m\dfrac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{r},\, \mathbf{v},\, t),$ 2^{a} lei de Newton
  - $-\nabla^2\,\Phi(\mathbf{r}) = \dfrac{\rho(\mathbf{r})}{\epsilon_0},$ Eq. Poisson

- Sistemas físicos são frequentemente regidos por equações diferenciais:
  - $m\dfrac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{r},\, \mathbf{v},\, t),$ 2^{a} lei de Newton
  - $-\nabla^2\,\Phi(\mathbf{r}) = \dfrac{\rho(\mathbf{r})}{\epsilon_0},$ Eq. Poisson
  - $i\hbar\dfrac{\partial}{\partial t}\Psi(\mathbf{r},\, t) = \left[-\dfrac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r},\, t)\right]\Psi(\mathbf{r},\, t),$ Eq. Schroedinger

UFPE

DF-UFPE

- Soluções analíticas são conhecidas para problemas relativamente simples.

- Soluções analíticas são conhecidas para problemas relativamente simples.
- Entretanto, considere:

UFPE

DF-UFPE

- Soluções analíticas são conhecidas para problemas relativamente simples.
- Entretanto, considere:
    - $m\frac{d\mathbf{v}}{dt} = -\frac{GMm\mathbf{r}}{r^3} - kv^n\mathbf{v}$

- Soluções analíticas são conhecidas para problemas relativamente simples.
- Entretanto, considere:
    - $m\frac{d\mathbf{v}}{dt} = -\frac{GMm\mathbf{r}}{r^3} - kv^n\mathbf{v}$
- Neste caso, soluções numéricas podem ser mais viáveis.

- Algoritmo ou método numérico apropriado para o problema.

- Algoritmo ou método numérico apropriado para o problema.
- Linguagem de programação para implementação do método/algoritmo em computadores.

- Algoritmo ou método numérico apropriado para o problema.
- Linguagem de programação para implementação do método/algoritmo em computadores.
- Depurar (*debugar*) o programa, i.e., identificar eventuais erros no código.

- Algoritmo ou método numérico apropriado para o problema.
- Linguagem de programação para implementação do método/algoritmo em computadores.
- Depurar (*debugar*) o programa, i.e., identificar eventuais erros no código.
- Otimizar, analisar e tratar os resultados obtidos.

- Depende do problema em questão. Diversos fatores devem ser considerados, dentre eles:

- Depende do problema em questão. Diversos fatores devem ser considerados, dentre eles:
  - Facilidade em aprender a linguagem de programação.

- Depende do problema em questão. Diversos fatores devem ser considerados, dentre eles:
  - Facilidade em aprender a linguagem de programação.
  - Tempo de execução do programa.
    $\uparrow$ C/C++, Fortran $\downarrow$ Python, Matlab, Octave, etc.

- Depende do problema em questão. Diversos fatores devem ser considerados, dentre eles:
  - Facilidade em aprender a linguagem de programação.
  - Tempo de execução do programa.
    $\uparrow$ C/C++, Fortran $\downarrow$ Python, Matlab, Octave, etc.
  - Tempo elaboração do programa. Em geral,
    $\downarrow$ C/C++, Fortran $\uparrow$ Python, Matlab, Octave, etc.

- Depende do problema em questão. Diversos fatores devem ser considerados, dentre eles:
  - Facilidade em aprender a linguagem de programação.
  - Tempo de execução do programa.
    $\uparrow$ C/C++, Fortran $\qquad$ $\downarrow$ Python, Matlab, Octave, etc.
  - Tempo elaboração do programa. Em geral,
    $\downarrow$ C/C++, Fortran $\qquad$ $\uparrow$ Python, Matlab, Octave, etc.
  - Reusabilidade do código.
    $\uparrow$ Linguagens orientadas a objetos (eg. C++)
    $\downarrow$ Linguagens não orientadas a objeto (C, por exemplo).

- Depende do problema em questão. Diversos fatores devem ser considerados, dentre eles:
  - Facilidade em aprender a linguagem de programação.
  - Tempo de execução do programa.
    $\uparrow$ C/C++, Fortran $\quad\quad$ $\downarrow$ Python, Matlab, Octave, etc.
  - Tempo elaboração do programa. Em geral,
    $\downarrow$ C/C++, Fortran $\quad\quad$ $\uparrow$ Python, Matlab, Octave, etc.
  - Reusabilidade do código.
    $\uparrow$ Linguagens orientadas a objetos (eg. C++)
    $\downarrow$ Linguagens não orientadas a objeto (C, por exemplo).
  - Portabilidade do código.

- Imprimir na tela.
  Checar resultados rapidamente em trechos específicos $(\ddot\smile)$
  Checar resultados de forma geral $(\ddot\frown)$
  Coletar dados $(\ddot\frown)$

UFPE

DF-UFPE

- Imprimir na tela.
  Checar resultados rapidamente em trechos específicos $(\ddot\smile)$
  Checar resultados de forma geral $(\ddot\frown)$
  Coletar dados $(\ddot\frown)$
- Imprimir/salvar em arquivos.
  Checar resultados rapidamente em trechos específicos $(\ddot\frown)$
  Checar resultados de forma geral $(\ddot-)$
  Coletar dados $(\ddot\smile)$

- Imprimir na tela.
  Checar resultados rapidamente em trechos específicos $(\ddot{\smile})$
  Checar resultados de forma geral $(\ddot{\frown})$
  Coletar dados $(\ddot{\frown})$
- Imprimir/salvar em arquivos.
  Checar resultados rapidamente em trechos específicos $(\ddot{\frown})$
  Checar resultados de forma geral $(\ddot{-})$
  Coletar dados $(\ddot{\smile})$
- Visualização em tempo real.
  Checar resultados rapidamente $(\ddot{\smile})$
  Checar resultados de forma geral $(\ddot{\smile})$
  Coletar dados $(\ddot{\frown})$

UFPE

DF-UFPE

- Se exisitirem soluções analítica, compará-las com os resultados numéricos.

- Se exisitirem soluções analítica, compará-las com os resultados numéricos.
- Se soluções analíticas não forem conhecidas, comparar os resultados encontrados com valores conhecidos (ou esperados) em limites específicos.

- Se exisitirem soluções analítica, compará-las com os resultados numéricos.
- Se soluções analíticas não forem conhecidas, comparar os resultados encontrados com valores conhecidos (ou esperados) em limites específicos.
- Comparar com simulações já realizadas feitas por outros pesquisadores e publicadas em revistas de renome.

- Se exisitirem soluções analítica, compará-las com os resultados numéricos.
- Se soluções analíticas não forem conhecidas, comparar os resultados encontrados com valores conhecidos (ou esperados) em limites específicos.
- Comparar com simulações já realizadas feitas por outros pesquisadores e publicadas em revistas de renome.
- De fato, nem sempre é possível ter absoluta certeza dos resultados. Porém normalmente um código livre de erros e com implementação correta de um método numérico sugere que os resultados obtidos estão corretos dentro de uma margem de erro. Neste caso, o ideal é que ao menos duas pessoas escrevam independentemente códigos e comparem os resultados encontrados.

Roteiro da aula

Declaração de bibliotecas

Comunica que bibliotecas serão utilizadas.

Definições de funções/subrotinas

Definições de funções à parte que podem ser acessadas um número arbitrário de vezes.

Declaração de variáveis e entrada de dados

Em cada função (incluindo a principal):
Informa o tipo (e tamanho, se array) de cada variável que será utilizada.
Fornece os valores iniciais para cada variável (`scanf`, `fscanf`, `cin`, etc).

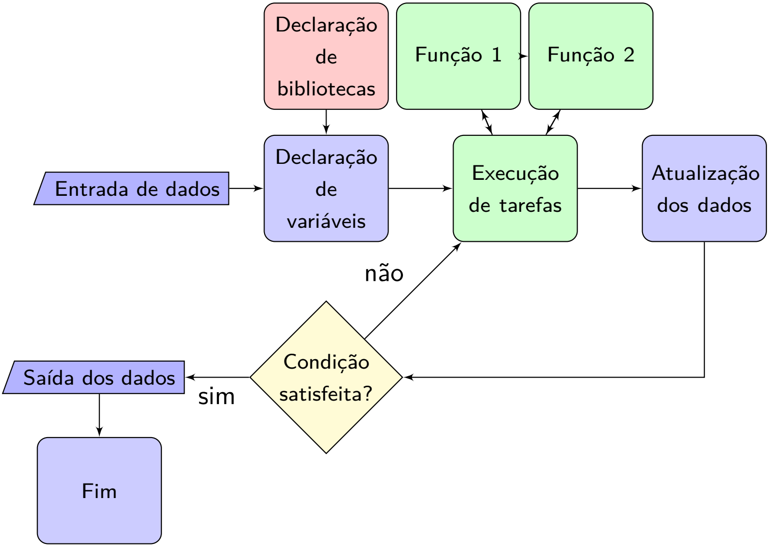
Execução de tarefas

Loops (comandos `for`, `while`, etc),
Comandos de controle (`if-else`, `switch-case`)

Saída de dados

Impressão na tela (`printf`, `cout`)
Imprimir em arquivo (`fprintf`).
Visualização gráfica em janelas (OpenGL).

Diagrama mínimo de uma rotina computacional



Tipos de variáveis (*data types*)

- Numéricas, caracteres, *strings*, lógicas, etc.

Variáveis numéricas

Tipos de variáveis (*data types*)

- Numéricas, caracteres, *strings*, lógicas, etc.
- Cada tipo de variável ocupa um espaço na memória, i.e., um dado número de *bits* ou *bytes* (8 *bits*).

Variáveis numéricas

Tipos de variáveis (*data types*)

- Numéricas, caracteres, *strings*, lógicas, etc.
- Cada tipo de variável ocupa um espaço na memória, i.e., um dado número de *bits* ou *bytes* (8 *bits*).

Variáveis numéricas

- de ponto fixo (*fixed-point*) – que são inteiros (em C/C++, `int`, `short` e `long`). A representação de inteiros é exata, desde que sejam realizadas operações que resultem somente em inteiros (por exemplo, divisão com resto) e que os valores não ultrapassem os valores máximo e mínimo permitidos pelo computador.

Tipos de variáveis (*data types*)

- Numéricas, caracteres, *strings*, lógicas, etc.
- Cada tipo de variável ocupa um espaço na memória, i.e., um dado número de *bits* ou *bytes* (8 *bits*).

Variáveis numéricas

- de ponto fixo (*fixed-point*) – que são inteiros (em C/C++, `int`, `short` e `long`). A representação de inteiros é exata, desde que sejam realizadas operações que resultem somente em inteiros (por exemplo, divisão com resto) e que os valores não ultrapassem os valores máximo e mínimo permitidos pelo computador.
- de ponto flutuante (*floating-point*), usualmente de precisão simples (float em C/C++) ou dupla (double em C/C++).

Roteiro da aula

UFPE

DF-UFPE

Erro de arredondamento (*roundoff error*)

- presente em valores do tipo ponto flutuante devido ao fato de não poderem ser representados com precisão absoluta pela máquina.

Erro de arredondamento (*roundoff error*)

- presente em valores do tipo ponto flutuante devido ao fato de não poderem ser representados com precisão absoluta pela máquina.
- Ex: $\pi = 3.14159265358979$ (decimal) ou

  $\pi = 0001100010110100001000100010101011011111100001001001000000000010$,

  (representação binária do Octave).

Erro de arredondamento (*roundoff error*)

- presente em valores do tipo ponto flutuante devido ao fato de não poderem ser representados com precisão absoluta pela máquina.
- Ex: $\pi = 3.14159265358979$ (decimal) ou

  $\pi = 0001100010110100001000100010101011011111100001001001000000000010,$

  (representação binária do Octave).
- variável ponto flutuante é representada como $s \times M \times B^e$, onde s é um *bit* (0 ou 1) utilizado para representar o sinal ($+$ ou $-$), M é a mantissa (um inteiro positivo representado por um certo número de *bits*), e é um expoente inteiro, também representado por um certo número de *bits*, e B é a base (pode ser 2 ou múltiplo de 2).

UFPE

DF-UFPE

Erro de arredondamento (*roundoff error*)

- presente em valores do tipo ponto flutuante devido ao fato de não poderem ser representados com precisão absoluta pela máquina.
- Ex: $\pi = 3.14159265358979$ (decimal) ou

  $$\pi = 0001100010110100001000100010101011011111100001001001000000000010,$$

  (representação binária do Octave).
- variável ponto flutuante é representada como $s \times M \times B^e$, onde s é um *bit* (0 ou 1) utilizado para representar o sinal ($+$ ou $-$), M é a mantissa (um inteiro positivo representado por um certo número de *bits*), e é um expoente inteiro, também representado por um certo número de *bits*, e B é a base (pode ser 2 ou múltiplo de 2).
- Um número pode, portanto, ser representado de diversas maneiras. Por exemplo, podemos ter o número dois com $s = 0$ (sinal positivo), $M = 10$ (base-2) e $e = 0$ (base-2) ou $M = 1$ (base-2) e $e = 1$ (base-2). Números decimais diferentes são representados com diferentes expoentes a fim de se obter maior precisão.

Erro de arredondamento

- Problema: soma de dois números requer expoentes iguais. Logo, quando dois números de ordens de magnitude muito diferentes são somados, é possível que o resultado seja o mesmo valor do maior número somado. Erros deste tipo são também chamados de erros de arredondamento e são inerentes à máquina utilizada.

Erro de arredondamento

- Problema: soma de dois números requer expoentes iguais. Logo, quando dois números de ordens de magnitude muito diferentes são somados, é possível que o resultado seja o mesmo valor do maior número somado. Erros deste tipo são também chamados de erros de arredondamento e são inerentes à máquina utilizada.
- Há muitas operações algébricas susceptíveis a erros deste tipo, como por ex., $\sqrt{1+x^2}-1$ para $x \ll 1$.

Erro de arredondamento

- Problema: soma de dois números requer expoentes iguais. Logo, quando dois números de ordens de magnitude muito diferentes são somados, é possível que o resultado seja o mesmo valor do maior número somado. Erros deste tipo são também chamados de erros de arredondamento e são inerentes à máquina utilizada.
- Há muitas operações algébricas susceptíveis a erros deste tipo, como por ex., $\sqrt{1+x^2}-1$ para $x \ll 1$.
- Precisão (exatidão) da máquina (machine accuracy, ε_m):

$$\min|\varepsilon_m| \text{ tal que } 1.0+\varepsilon_m \neq 1.0$$

UFPE

DF-UFPE

Erro de arredondamento

- Problema: soma de dois números requer expoentes iguais. Logo, quando dois números de ordens de magnitude muito diferentes são somados, é possível que o resultado seja o mesmo valor do maior número somado. Erros deste tipo são também chamados de erros de arredondamento e são inerentes à máquina utilizada.
- Há muitas operações algébricas susceptíveis a erros deste tipo, como por ex., $\sqrt{1+x^2}-1$ para $x \ll 1$.
- Precisão (exatidão) da máquina (machine accuracy, ε_m):

$$\min|\varepsilon_m| \text{ tal que } 1.0+\varepsilon_m \neq 1.0$$

- Após N operações aritméticas erros de arredondamento podem se acumular, tal que

$$\text{Erro total} \sim \epsilon_m \begin{cases} \sqrt{N}, & \text{erros aleatórios e sem } \textit{bias} \\ N, & \text{erros sistemáticos} \end{cases}$$

UFPE

DF-UFPE

Erro de truncamento

- Surge do algoritmo não possuir precisão absoluta. É dado pela diferença entre o valor encontrado pelo algoritmo e o valor exato. Pode ser controlado.

Estabilidade de um algoritmo

Erro de truncamento

- Surge do algoritmo não possuir precisão absoluta. É dado pela diferença entre o valor encontrado pelo algoritmo e o valor exato. Pode ser controlado.

Estabilidade de um algoritmo

- Algoritmos estáveis são aqueles cujos resultados são próximos ao valor exato.

Erro de truncamento

- Surge do algoritmo não possuir precisão absoluta. É dado pela diferença entre o valor encontrado pelo algoritmo e o valor exato. Pode ser controlado.

Estabilidade de um algoritmo

- Algoritmos estáveis são aqueles cujos resultados são próximos ao valor exato.
- Algoritmos instáveis são aqueles em que erros de arredondamento levam a resultados completamente diferentes do valor exato. Ou seja, o erro resultante a partir de um dada etapa de execução do algoritmo cresce indefinidamente.

Roteiro da aula

Equação de movimento Newtoniana

$$m\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F}\left(\mathbf{r},\ \mathbf{v},\ t\right), \qquad \mathbf{r}(t_0) = \mathbf{r}_0, \qquad \left.\frac{d\mathbf{r}}{dt}\right|_{t_0} = \mathbf{v}_0$$

Forma conveniente para implementação numérica para a EDO de 2^a ordem

$$\frac{d\mathbf{v}}{dt} = \frac{\mathbf{F}\left(\mathbf{r},\ \mathbf{v},\ t\right)}{m}, \quad \rightarrow \quad \mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \int_t^{t+\Delta t} \frac{\mathbf{F}\left(\mathbf{r},\ \mathbf{v},\ t'\right)}{m}\, dt'$$

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}(t) \quad \rightarrow \quad \mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \int_t^{t+\Delta t} \mathbf{v}(t')\, dt'$$

Equivalente a resolver sistema com EDOs de 1^a ordem

$$\frac{dy(x)}{dx} = f(x) \quad \rightarrow \quad y(x+\Delta x) = y(x) + \int_x^{x+\Delta x} f(x')dx'$$

Discretização de variáveis

$$x_j = x_0 + j\Delta x \to \{x_0,\, x_0 + \Delta x,\, x_0 + 2\Delta x, \ldots, x_0 + N\Delta x\}$$

$$y_{j+1} = y_j + \text{estimativa de } \left(\int_{x_j}^{x_{j+1}} f(x')dx' \right) + \mathcal{O}(\Delta x^n)$$

Métodos de integração

- Definem de que maneira as integrais são aproximadas.
- Cada passo de um método possui precisão de ordem Δx^n (erro de truncamento, onde o valor de n depende do método em questão). Como há N iterações, o método tem precisão de ordem $N\Delta x^n = \dfrac{L}{\Delta x}\Delta x^n \propto \Delta x^{n-1}$ (aqui L representa o intervalo no qual o método é iterado).
- Aproximação mais simples considera valor de $f(x)$ em algum $x_j \leq x \leq x_{j+1}$, de forma que
$$y_{j+1} = y_j + f(x)\Delta x + \mathcal{O}(\Delta x^2)$$
- Métodos estão descritos nas notas de aula e no capítulo 3 da 3^a Ed. do Tobochnik.

Roteiro da aula

Queda de corpos

A queda de um corpo de massa m próximo à superfície da terra na presença de força de arrasto pode ser descrita pela equação $\frac{d^2y}{dt^2} = -g - C_d|v|^{b-1}v$, onde $g \approx 9.8$ m/s^2, C_d é o coeficiente de arrasto e $b = 1$ (regime lamelar, baixas velocidades) ou $b = 2$ (regime turbulento, baixas velocidades). Em termos da velocidade terminal (quando a aceleração é zero), $v_t = (g/C_d)^{1/b}$, temos

$$\frac{dv}{dt} = -g\left[1 + \left(\frac{|v|}{v_t}\right)^{b-1}\frac{v}{v_t}\right], \qquad \frac{dy}{dt} = v.$$

Elabore um relatório conciso (em PDF) respondendo às perguntas abaixo:

1. Resolva numericamente este sistema de equações (i.e., encontre $v(t)$ e $y(t)$) utilizando pelo menos dois métodos de integração de ordens diferentes.

   Dica: Talvez seja interessante tornar as grandezas adimensionais. Por exemplo, escreva $\tilde{v} = v/v_t$, $\tilde{t} = gt/v_t$ e $\tilde{y} = gy/v_t^2$, de modo que as equações se tornam $\frac{d\tilde{v}}{d\tilde{t}} = -(1 + |\tilde{v}|^{b-1}\tilde{v})$ e $\frac{d\tilde{y}}{d\tilde{t}} = \tilde{v}$. Por que pode ser vantajoso resolver o problema desta forma?

2. Esboce gráficos de posição x e velocidade v calculados numericamente em função do tempo t.

   Obs 1: Para os gráficos há softwares apropriados, tais como `gnuplot` ou `grace`, bem como executar *scripts* utilizando `python/matplotlib` ou `octave`.

   Obs 2: Não é necessário ter pontos para cada passo de tempo. É suficiente obter da ordem de ~ 1000 pontos para visualiação adequada dos dados.

   Obs 3: se seu código estiver em C, C++ ou FORTRAN, talvez seja necessário salvar os resultados em arquivos para depois plotá-los.

3. Calcule a energia mecânica do sistema e esboce gráficos da mesma em função do tempo. O que você pode concluir dos resultados obtidos?

4. Varie o passo de integração, Δt (por exemplo, escolha $\Delta t = 0.1$, $\Delta t = 0.01$, $\Delta t = 0.001$, etc). Examine a dependência das soluções com Δt. Compare as soluções analítica, v_{exata}, e numérica, v_{num}, fazendo um gráfico de $\log |v_{\text{num}} - v_{\text{exata}}|$ versus t.