Curso de Física Computacional

1. Introdução

Este é um curso introdutório de Métodos Computacionais em Física. Ele é dirigido ao estudante interessados em aprender as principais técnicas de modelagem e simulação em física. Ao longo do curso serão discutidos algoritmos e técnicas de uso geral. Algumas demonstrações, quando estritamente necessárias, serão feitas, no entanto, o estudante deve ter em mente que este não é um curso de física estatística ou de análise numérica. Existem muitos bons textos nestes assuntos e discuti-los em detalhes iria somente complicar desnecessariamente o curso tornando-o enfadonho. Assim, este pretende ser um curso do tipo "mãos na massa". As referências ao fim de cada unidade ajudarão a preencher esta lacuna. Ao estudante que deseja continuar neste ofício encorajo fortemente a consulta àquelas referências, em particular a um livro escrito na década de 1970 por David Potter [**Potter**]

O uso de técnicas de física computacional, principalmente de simulação, é essencial para o desenvolvimento de qualquer pesquisa científica, completando o ciclo Teoria-Experimento-Simulação. Sob o ponto de vista científico a física computacional permite explorar modelos para os quais é impossível se obter soluções analíticas ou realizar experimentos muito custosos em laboratório, ajudando a comprovar ou refutar novos modelos. Existem exemplos nas mais diversas áreas do conhecimento: desenvolvimento de novos reatores nucleares, novos materiais, previsão do tempo e assim por diante.

Nossa pretensão é que este seja um curso autocontido, com os tópicos tendo continuidade ao longo de seu desenvolvimento. Discutiremos as técnicas de integração numérica mais usuais, introduzindo problemas de contorno e integrações no tempo. Na sequência veremos como aplicar estas técnicas a soluções de problemas que aparecem com frequência nos cursos de física. Nos capítulos mais adiantados veremos como tratar um problema de muitos corpos com condições iniciais e de contorno adequados. O cálculo de quantidades médias como temperatura, volume, pressão, magnetização são introduzidas de forma que os resultados obtidos pelo experimento computacional possam ser comparados com previsões teóricas e a experimentos de laboratório. Um ponto essencial, a determinação de erros nas medidas do experimento computacional será tratado ressaltando sua importância. Assim como num experimento em laboratório, um experimento computacional só faz sentido quando acompanhado das respectivas barras de erros.

Os algoritmos e técnicas apresentados no curso podem ser codificadas em qualquer linguagem, no entanto, algum cuidado deve ser observado. Existem linguagens mais apropriadas para o desenvolvimento de códigos que chamamos de "mastigadores de

números" (*number crunching!*). Estas linguagens, em geral, são mais dedicadas e, portanto, apresentam uma melhor performance na solução de longos problemas numéricos. Isto não significa que o estudante não possa desenvolver os códigos em linguagens mais populares atualmente, como é o caso de Python ou Visual Basic. Cada problema pode requerer uma solução diferente ou mais adequada ao momento. Se o único objetivo for calcular numericamente o valor de uma integral, ou obter a solução de um conjunto de equações diferenciais, estas linguagens podem ser mais que adequadas. Contudo, se queremos obter o comportamento de muitas partículas interagindo via potencial de muitos corpos a situação se torna dramática. Nestes casos não só a escolha adequada da linguagem é um fator determinante, mas o uso de técnicas de vetorização e paralelização são imprescindíveis. Tem se tornado bastante comum o uso de processadores de vídeo, no entanto isto deve ser visto com certo cuidado, uma vez que nem sempre seu código será mais eficiente. Existem vários códigos públicos (caixas pretas) para resolver um número muito grande de problemas, mas, deve-se ter em mente que por serem de uso geral, nem sempre são os códigos mais eficientes para resolver um problema particular. O estudante mais interessado deve procurar se inteirar das linguagens e das técnicas de programação adequadas ao seu problema particular. As linguagens para solução de problemas numéricos que fornecem uma melhor performance atualmente são FORTRAN[**FORTRAN**] e C (C++)[**C++**], que caminham par e passo em termos de performace. Uma tentativa de desenvolver uma nova linguagem, JULIA[**Julia**], está sendo feita, mas ainda é de pouco uso, e ainda não há certeza da continuidade de seu desenvolvimento, como ocorre com FORTRAN e C++. A escolha de uma linguagem deve obedecer a alguns critérios fundamentais. Sua empatia com a gramática da linguagem e quão rápida será a execução do código são, sob meu ponto de vista, determinantes. Outro ponto essencial é a portabilidade do código. De nada adianta escrever um código muito eficiente se ele tem que ser readequado ao se mudar o fabricante do compilador ou do processador, por isso, evite escrever em ASSEMBLER!

Ao final do curso espero que o estudante tenha compreendido as técnicas para solução de problemas de média complexidade, estando pronto para voos mais altos.