MAC0499 – Trabalho de Formatura Supervisionado.

Proposta de Trabalho `1.0`

Giuliano Augusto Faulin Belinassi - 8517272

1 Introdução

É de extrema importância para a Engenharia Civil, se não para a Sociedade como um todo, o estudo de vibrações em estruturas para evitar possíveis catástrofes provindas de terremotos, ou talvez incômodos de outras fontes, tais como máquinas operatrizes e linhas ferroviárias. Como na grande maioria dos casos as vibrações chegam às construções através do solo, a principal parte da Engenharia que trata problemas desta espécie é a Dinâmica dos Solos.

Com as evolução dos computadores, é de se esperar que heurísticas e algoritmos fossem projetados para simular o efeitos de tais vibrações em estruturas. Dentre estes, deve-se ressaltar o Método dos Elementos de Contorno (MEC), que utiliza pontos na superfície do volume a ser analisado, e o Método dos Elementos Finitos (MEF), que insere pontos internos ao volume do objeto de estudo. Existem vantagens em escolher o MEC ao MEF, entre elas [1]:

1. Menor tempo para preparação dos dados.
2. Maior precisão dos pontos de estresse.
3. Menor uso de recursos computacionais.
4. Menos informações desnecessárias.

e isto motiva o estudo de tal método para solução de problemas deste tipo. Embora toda a análise teórica do BEM seja interessante no ponto de vista da Engenharia, este trabalho tem enfoque na implementação do algoritmo de forma a explorar recursos computacionais providenciados por processadores de vários núcleos e Unidades de Processamento Gráfico do Propósito Geral, portanto a parte computacional do problema.

2 Objetivos

Este trabalho tem como objetivo implementar o MEC usando Unidades de Processamento Gráfico de Propósito Geral(GPGPU) partindo da implementação fornecida por [2]. A necessidade de paralelização se dá devido a possibilidade de analisar estruturas com superfícies maiores e com mais pontos, aumentando assim a precisão dos resultados obtidos.

Tabela 1: *Proffiling* do código sequêncial

Subrotina	Tempo (%)
solfund	34.11
nonsingd	18.24
nonsinge	14.71
solfune	9.13
gauleg	8.23

Tabela 2: Tempo gasto em um Core(TM) i7 CPU 920 @ 2.6GHz

Processadores	Tempo
1	1m20s
2	42s
4	22s

3 Estado do Trabalho

Este trabalho teve inicio em Setembro de 2016, com a primeira etapa tendo como objetivo estudar a linguagem de programação usada na implementação entregue por [2] (no caso, Fortran). Algumas dificuldades foram encontradas aí, pois o código foi escrito em uma versão mais antiga desta linguagem, além de não compilar no compilador GFortran. Adaptações e manutenções foram necessárias.

Em seguida, algumas técnicas de paralelização de algoritmos foi estudada para que fosse possível prosseguir com o trabalho, além de testes automatizados.

Um *proffiling* foi efetuado no código, logo em seguida, para que os gargalos tornassem evidentes, conforme a tabela 1.

Analisando mais rebuscadamente o código, descobriu-se que paralelizar o procedimento *Ghmatecd*, responsável por montar as matrizes H e G conforme [2], implicaria em chamadas paralelas das subrotinas *Nonsingd* e *Solfund*, sendo assim passou-se a priorizar a paralelização dessa subrotina.

Como alterar um código pode comprometer funcionalidades que deste dependem, a criação de testes automatizados foi fundamental para assegurar que o resultado obtido condiz com o original. Embora existam frameworks para a criação de testes em Fortran, por este projeto tratar com código legado, preferiu-se criar as proprias funções independentes e programas externos que verificam se o resultado é, de fato, o esperado. Portanto, para verificar os erros nas matrizes geradas, preferiu-se utilizar ora a norma infinita, ora a norma 1 da soma da diferença das matrizes, conforme o Teorema 2.1.29 [3].

A partir do resultado obtido, a etapa de paralelização usando OpenMP teve início, em conjunto com algumas otimizações sequênciais. Com uma entrada de ????2160????, o tempo gasto na implementação original era superior a 4m17s. Após paralelizar todo o programa e realizar algumas otimizações sequências este tempo caiu em função da quantidade de processadores alocados, conforme a tabela 2.

Em seguida, com todo o programa paralelizado na CPU, partiu-se para experimentos com GPGPU na tentativa de obter um resultado melhor. Como a plataforma escolhida

foi CUDA e como o compilador de CUDA para Fortran é pago, optou-se por construir uma interface CUDA C $\longleftrightarrow$ Fortran.

Prosseguindo com a paralelização em GPGPU, uma implementação de GHMATECD foi codificada em CUDA C, porém os resultados não foram satisfatórios:

COLETAR OS RESULTADOS NA GPU E INSERIR AQUI.

Referências

[1] Heinz Antes. A short course on bondary elements methods. *Technische Universität Braunschweig*, 2010.

[2] Ronaldo Carrion. *Uma Implementação do Método dos Elementos de Contorno para Problemas Viscoelastodinâmicos Estacionários Tridimensionais de Domínios Abertos e Fechados*. PhD thesis, Universidade Estadual de Campinas, 2002.

[3] David S. Watkins. *Fundamentals of Matrix Computations*, volume 64. John Wiley & Sons, 2 edition, 2004.