APLICAÇÃO DE TÉCNICAS DA COMPUTAÇÃO DE ALTO DESEMPENHO EM ALGORITMOS DE FLUXO DE POTÊNCIA

JOÃO VICTOR DAHER DAIBES*, RAINER ZANGHI†, JULIO CESAR STACCHINI DE SOUZA*, MARCIO ANDRE RIBEIRO GUIMARAENS†

**Instituto de Computação*
Universidade Federal Fluminense - UFF
Niterói, Rio de Janeiro, Brasil

†*Departamento de Engenharia Elétrica*
Universidade Federal Fluminense - UFF
Niterói, Rio de Janeiro, Brasil

Emails: joaovdd@id.uff.br, rzanghi@id.uff.br, julio@ic.uff.br, mguimaraens@id.uff.br

Abstract— Power flow calculation is the most common analysis procedure for power electrical systems. However, this process requires the solution of a large number of nonlinear algebraic equations. Therefore, the usual algorithm makes use of iterative numerical methods and is implemented sequentially, taking advantage of the high sparseness of the problem arising from the characteristic network connectivity to gain speed and conserve storage. In this article we will describe some of the fundamentals of the power flow problem and the solution algorithms that make use of the Newton-Raphson numerical method, as well as the relevance of its optimization to some of the functions of the Smart Grid. Parallel implementation will also be proposed, making use of Graphics Processing Units as accelerators, on which benchmarking tests will be carried out. The suggested implementation presented acceleration greater than 30 times when confronted with a sequential equivalent.

Keywords— High Performance Computing, Parallel Processing, GPUs, Load Flow, Smart Grid.

Resumo— O cálculo do fluxo de potência é o procedimento de análise mais comum no que tange sistemas elétricos de potência. Contudo, esse processo requer a solução de grande número de equações algébricas não lineares. Por isso, o algoritmo usual faz uso de métodos numéricos iterativos e é implementado de forma sequencial, tirando proveito da elevada esparsidade do problema advinda da característica conectividade de rede para ganhar velocidade e conservar armazenamento. Nesse artigo serão descritos alguns dos fundamentos do problema do fluxo de potência e dos algoritmos de solução que fazem uso do método numérico de Newton-Raphson, assim como a relevância de sua otimização para a realização de algumas das funções da Smart Grid. Também será proposta implementação paralelizada fazendo uso de Unidades Gráficas de Processamento como aceleradores, sobre a qual serão realizados testes de benchmarking. A implementação sugerida apresentou aceleração superior a 30 vezes quando confrontada com uma equivalente sequencial.

Palavras-chave— Computação de Alto Desempenho, Paralelismo, GPUs, Fluxo de Potência, Smart Grid.

1 Introdução

O *Sistema Elétrico de Potência* é uma rede composta por usinas geradoras de eletricidade e entes consumidores de energia interligados por diversos componentes encarregados da transmissão de potência. É necessário que se tenha profundo conhecimento de seu estado de operação para que se possa garantir seu funcionamento ou perceber a necessidade de expansões futuras. Contudo, nem sempre existe disponibilidade de informações de forma direta. Estudos de cenários futuros previstos ou hipotéticos e situação de dificuldade de aquisição de dados do sistema real são exemplos de contextos práticos comuns (Huang et al., 2012).

Por esse motivo, a *Análise de Sistemas de Potência* vem se mostrando uma popular solução para esse problema no planejamento, projeto e operação de sistemas elétricos de potência. Esse campo de estudo pode ser subdividido em *Análise de Estado Estacionário* e *Análise de Estabilidade Transitória*. O primeiro é ainda categorizado em *Análise de Fluxo de Potência* e *Análise de Segurança de Estado Estático (SSSA)*. A análise do

fluxo de potência considera conjuntura em que os efeitos transitórios são considerados interrompidos e envolve a determinação das magnitudes e ângulos de fase das tensões, em cada uma das barras de um sistema estável (Monticelli, 1986).

Devido à complexidade e alto custo computacional dos algoritmos de fluxo de potência, técnicas da computação de alto desempenho como computação heterogênea, paralelismo e tratamento especial de algumas das matrizes envolvidas considerando seu elevado grau de esparsidade (Sambarapu and Halpin, 2007) são utilizadas. As contribuições destas técnicas merecem mais estudo e detalhes de suas implementações encontram-se ainda em desenvolvimento.

Além disso, o crescimento da presença, no sistema de distribuição, de *fontes geradoras distribuídas* já pode ser observado e se mostra tendência futura. Segundo Antonova et al. (2012), à medida que a geração dispersa aumenta, os alimentadores de distribuição passam a apresentar características mais próximas às de redes de transmissão, onde a geração e os nós de carga são misturados. Isso exige um sistema de proteção mais complexo

do que o projetado da maneira usual, passando a ser necessários estudos como os de controle de potência ativa, frequência e tensão. Essa inclinação aponta crescente necessidade de se reduzir o tempo computacional gasto pelas funções de análise do sistema de potência, de forma que o processo decisório ocorra em tempo hábil para a execução das ações de controle que se fizerem necessárias. O conceito de redes inteligentes engloba a questão da acomodação desse tipo de geração de energia. Além disso, outros pontos pesquisados como funções da *smart grid* incluem a elevação da confiabilidade da rede e da qualidade da energia, a otimização do uso dos ativos ou a minimização do número e custo computacional das análises. Para que essas funções sejam realizadas, faz-se necessário que seja calculado o fluxo de potência do sistema.

Contudo, sistemas de grande porte podem possuir milhares de barras, o que exige simplificações para que seja alcançada redução do tempo de latência do cálculo ao utilizar métodos de fluxo de potência da forma tradicional (i.e., com processamento serial). Para que o processo decisório seja o melhor possível, é desejável que o tempo de cálculo computacional seja reduzido para tornar factível análise de sistemas de grande porte, considerando os cenários sem reduções e simplificações de modelagem.

Este estudo objetiva a explorar a possibilidade de redução do tempo necessário para a solução computacional do problema do fluxo de potência ao implementá-lo de forma paralelizada, fazendo uso de *unidades gráficas de processamento* (GPUs). Primeiramente, na Seção 2, serão mencionados alguns aspectos básicos do problema sob estudo. Na Seção 3 será exposto porque se mostra relevante estudar o uso de *GPUs* para a solução desse problema. Já na Seção 4 será detalhada a forma como foram construídas as implementações sequencial e paralelizada a serem comparadas. Por fim, na Seção 5 serão descritos os testes, assim como os resultados obtidos.

2 O Problema do Fluxo de Potência

Define-se o cálculo do fluxo de carga (ou de potência) em uma rede de energia elétrica como a determinação do estado dessa rede (e.g., valores de suas tensões nodais complexas), da distribuição desses fluxos, assim como de algumas outras grandezas de interesse (Monticelli, 1986). Para tal, modela-se o sistema de forma estática como um conjunto de equações e inequações algébricas.

Esse trabalho se concentrará na análise de sistemas de transmissão de alta tensão. Essas redes são usualmente balanceadas – i.e., correntes e tensões nas três fases diferem apenas por defasagem de 120°. Por esse motivo, torna-se possível a utilização de circuito equivalente monofásico do sis-

tema elétrico de potência sob análise para a solução do problema.

Para que o problema do cálculo do fluxo de potência possa ser formulado e calculado computacionalmente, faz-se necessário que o sistema seja modelado. Uma forma comum de representar algebricamente o modelo é via a matriz de admitâncias nodais (Monticelli, 1986).

As características da formulação de um problema de fluxo de carga são descritas em Monticelli (1986) que também enfatiza que os numéricos de *Newton-Raphson* e *Newton-Raphson Desacoplado Rápido* são os mais robustos e computacionalmente eficientes para a solução do sistema de equações não lineares inerente do procedimento em questão.

Programas comerciais que efetuam o cálculo do fluxo de carga em sistemas elétricos de potência usualmente possuem implementações de diferentes métodos numéricos para a solução do sistema citado anteriormente. Contudo, pode-se dizer que o *método de Newton-Raphson* se tornou um padrão *de-facto*, presente na maioria dos *softwares* desse tipo. A principal razão para isso é relativa ao fato das *propriedades de convergência* desse método serem robustas quando os valores iniciais escolhidos são suficientemente próximos da solução do sistema.

3 Computação de Alto Desempenho

A computação de alto desempenho é definida por Sravanthi et al. (2014) como a prática de agregar poder computacional de forma tal que seja possível atingir performance superior à alcançada por um *desktop* ou uma estação de trabalho típicos ao solucionar problemas científicos, de engenharia ou comerciais. Esse campo de estudo se concentra na procura de meios, desde a escolha do *hardware* ideal até realização de modificações na forma de implementação de algoritmos, visando atingir o máximo desempenho possível.

A solução do problema mencionado na Seção 2 tem sido amplamente usada de forma serializada ao longo das últimas décadas. Por isso, métodos de fluxo de potência já estão muito otimizados e adequados ao modo de funcionamento e limitações de unidades centrais de processamento (CPUs). Com o passar do tempo, a performance de CPUs foi melhorada com o processo de miniaturização dos transistores que as compõem. Como pôde ser observado nas últimas décadas, a *lei de Moore* (Mack, 2015) continua a se mostrar válida ao passo que o tamanho dos transistores passíveis de construção é reduzido de forma, aproximadamente, linear com o passar do tempo. Isso propicia a construção de processadores com maior capacidade de processamento. Contudo, as complexas unidades de processamento passaram a apresentar um fator limitante à sua velocidade de clock:

a quantidade de energia que elas precisam dissipar. Segundo McKeown et al. (2018), a potência consumida por unidades de processamento se tornou o principal ponto de design para processadores modernos. As GPUs têm contornado esse problema ao assumirem arquitetura mais simples com maior número de núcleos (Luebke and Humphreys, 2007).

Esse fenômeno, em conjunto com bibliotecas de álgebra linear estado da arte, vêm desconstruindo a antiga ideia da ineficácia da aplicação de implementações paralelas em problemas dos sistemas de potência (Zanghi, 2016). O entendimento passado era sustentado pelo fato de problemas dessa área usualmente não possuírem paralelismos óbvios inerentes às suas estruturas matemáticas (Tylavsky et al., 1992). Trabalhos como a simulação de estabilidade transitória em sistemas de potência de grande porte feita por Jalili-Marandi et al. (2012) e outros citados na revisão bibliográfica de Green et al. (2011) são exemplos do potencial desse campo de estudo no que tange o setor elétrico.

4 Implementação

Nesse trabalho, implementou-se o algoritmo do cálculo do fluxo de potência pelo método de Newton-Raphson, tanto de forma serial quanto de forma paralela, para que os respectivos desempenhos fossem analisados e comparados. Os passos executados por cada uma das implementações estão ilustrados nas Figuras 1 e 2.

É importante mencionar que todos os vetores e matrizes foram armazenados de *forma densa*, onde uma posição de memória é alocada para cada elemento, sem que os valores nulos, lidos ou calculados, sejam omitidos e, dessa forma, seja reduzida a quantidade de memória necessária para armazenamento.

5 Avaliação do Desempenho

5.1 Sistemas Teste

Para que os programas desenvolvidos nesse trabalho, conforme descrição da Seção 4, fossem testados e, posteriormente, comparados, utilizou-se como instância de teste os sistemas padrão do IEEE de 14, 30, 57 e 118 barras. Esses sistemas representam partes do Sistema de Energia Elétrica Americano – localizadas no centro-oeste dos Estados Unidos – do início da década de 1960 (Christie, 1963).

Além disso, foram construídos sistemas maiores, com mais ramos e barras, baseados no sistema padrão do IEEE de 118 barras (ieee118). Esses sistemas foram denominados *ieee118xn*, onde *n* é qualquer número natural. Para sua formação, as barras de número 66 de um sistema *ieee118* são

Figura 1: Fluxograma da implementação sequencial do algoritmo de solução do problema do fluxo de potência. Fonte: Autor.

Nome	Número de Barras	Número de Ramos
ieee14	14	20
ieee30	30	41
ieee57	57	80
ieee118	118	186
ieee118x2	236	347
ieee118x4	472	748
ieee118x8	944	1496

Tabela 1: Sistemas teste utilizados na comparação de desempenho entre as implementações sequencial e paralela. Fonte: Autor.

sempre ligadas às barras 66 de outro por meio de uma linha de transmissão idêntica à que liga a barra 49 à 66 em cada um dos sistemas originais. Analogamente, são sempre ligadas barras 89 a barras 89 por meio de linhas de transmissão idênticas às que ligam as barra 88 às 89 em cada sistema *ieee118*. Esses arranjos estão ilustrados na Figura 3. Quanto aos tipos das barras, a *swing* do primeiro sistema de padrão de 118 barras é mantida (barra 69), as barras *swing* dos demais sistemas *ieee118* ligados em anel são convertidas em barras de tipo *PV*.

Todos os sistemas teste utilizados para os estudos, assim como seus números de barras e ramos, estão organizados na Tabela 1.

5.2 Configuração da máquina em que foram realizados os testes

Uma estação de trabalho Dell Precision T3610 com processador Intel Xeon E5-1620 v2 com clock de 3.70GHz, quatro núcleos de processamento e

Figura 2: Fluxograma da implementação paralelizada do algoritmo de solução do problema do fluxo de potência. Fonte: Autor.

Figura 3: Construção dos sistemas teste ieee118x2 e ieee118x4. Um sistema nomeado da forma ieee118xn e construído interligando-se n sistemas padrões do IEEE com 118 barras, pelas barras 66 e 89, formando um anel. Fonte: Autor.

oito threads; memória de 8GB com 1600MHz; placa gráfica Nvidia Quadro K2000 com 2GB de memória, capacidade cuda 3.0, dois multiprocessadores, 384 núcleos CUDA e clock de 954MHz.

Enquanto o código fonte do programa sequencial que faz uso da biblioteca Eigen (versão 3.3) foi compilado com o compilador G++(GCC) 7.3.0, o código fonte da versão paralelizada foi compilado empregando o ambiente de desenvolvimento integrado (IDE) Nvidia Nsight Eclipse Edition 9.2.0, parte integrante do CUDA Toolkit. A biblioteca cuSolver utilizada também é parte do CUDA Toolkit.

5.3 Medição do tempo de execução e trecho do programa sob análise

A função `clock()` e a definição de pré-processador `CLOCKS_PER_SEC`, ambas componentes da biblioteca `time.h` – uma das bibliotecas padrão da linguagem de programação C –, são utilizadas para a mensuração do tempo empregado para ambas as

implementações. Essas etapas são evidenciadas nas Figuras 1 e 2.

5.4 Resultados

Para que o desempenho computacional das implementações seja de fato confrontado, os dois programas serão executados tendo como entrada arquivos no formato de dados comum (de extensão .cdf) contendo informações dos sistemas citados na Subseção 5.1.

Visando a minimização de interferências de execuções de outros comandos pelo sistema operacional, os programas serão executados cinquenta vezes para cada sistema teste. Serão, desse modo, obtidas 50 medidas de tempo de execução da fração da implementação que está sob estudo.

Para que a uniformidade dos valores de tempo medidos seja avaliada, será calculado o desvio padrão das medidas, conforme a fórmula (1).

$$\sigma = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^N (x_i - \mu)^2} \quad (1)$$

Onde N é o número de valores de tempo medidos e μ é a esperança para os valores de x_i , tal que $\mu = \frac{1}{N} \cdot \sum_{i=1}^N x_i$.

Os valores dos tempos médios de execução e os valores dos desvios padrão em milissegundos e percentuais estão organizados na Tabela 2. Comparando os tempos médios de execução dos dois programas para cada um dos sistemas, as possíveis incertezas atreladas aos valores de nenhum deles se aproximam da diferença entre essas quantidades. Por esse motivo, podem ser feitas comparações sem grandes prejuízos. Nesse sentido, organizou-se os valores dos tempos médios de execução no gráfico da Figura 4. Ao observá-lo torna-se mais clara a aceleração promovida pela implementação paralelizada.

	Implementação Sequencial			Implementação Paralela		
Sistema Teste	ieee14	ieee30	ieee57	ieee14	ieee30	ieee57
Tempo médio de execução (ms)	3,16	21,34	104,77	164,49	176,70	206,17
Desvio padrão (ms)	0,63	2,07	2,28	1,68	1,82	1,67
Desvio padrão (%)	20,05%	9,71%	2,17%	1,02%	1,03%	0,81%
Sistema Teste	ieee118			ieee118		
Tempo médio de execução (ms)	613,97			342,90		
Desvio padrão (ms)	5,67			2,04		
Desvio padrão (%)	0,92%			0,81%		
Sistema Teste	ieee118x2	ieee118x4	ieee118x8	ieee118x2	ieee118x4	ieee118x8
Tempo médio de execução (ms)	4531,38	34933,68	274888,73	768,11	2360,40	9082,40
Desvio padrão (ms)	9,58	41,91	315,35	1,82	2,93	3,66
Desvio padrão (%)	0,21%	0,12%	0,11%	0,24%	0,12%	0,04%

Tabela 2: Resultados dos testes com ambas as implementações. Fonte: Autor.

Figura 4: Tempo médio de execução dos programas sequencial e paralelo para cada um dos sistemas teste apresentados na Subseção 5.1. Escala de tempo logarítmica. Fonte: Autor.

Sistema Teste	ieee14	ieee30	ieee57
Aceleração	0,01918	0,1208	0,5082
Sistema Teste	ieee118		
Aceleração	1,791		
Sistema Teste	ieee118x2	ieee118x4	ieee118x8
Aceleração	5,899	14,80	30,26

Tabela 3: Aceleração promovida pelo programa paralelizado para cada um dos sistemas teste. Fonte: Autor.

5.4.1 Aceleração

Para promover melhor visualização dos resultados, será utilizado o conceito de *aceleração* semelhante ao definido por JáJá (1992): seja P um dado problema computacional e n a dimensão de seus dados de entrada. Denote a complexidade sequencial de P por $T(n)$ (i.e., tempo demandado para a execução de P). Seja A um algoritmo paralelo que resolva P no tempo $T_p(n)$ em um computador paralelo com processadores p . Então, a aceleração alcançada por A é definida como:

$$S_p(n) = \frac{T(n)}{T_p(n)} \quad (2)$$

Desse modo, podem ser calculadas acelerações, com relação às implementações construídas, para cada um dos sistemas teste. Elas foram organizadas na Tabela 3. Note que acelerações inferiores à unidade simbolizam, na verdade, que a execução do programa que faz uso de paralelismos foi mais lenta se comparada à sequencial.

Para utilizar a GPU para efetuar cálculos, primeiramente, faz-se necessário que seja alocado es-

paço na sua memória para a transferência de todos os dados a serem manipulados. Esses dados, inicialmente, se encontram armazenados na memória do *host*. Além disso, deve ser criado um *context*, análogo a um processo para CPUs, e o código do *kernel* a ser executado deve ser enviado para a GPU. Após o término da execução, o resultado deve ser enviado para o *hospedeiro* (Nvidia Corporation, 2018). Todo esse procedimento paralelo à execução, de fato, do *kernel* demanda tempo, em especial a transferência dos dados para e da a GPU. Por esse motivo, era esperado que, para sistemas teste de tamanho reduzido, o programa sequencial fosse mais rápido do que o paralelizado. Entretanto, com a elevação do tamanho do sistema teste, era esperado um ponto em que o maior número de *threads* e *núcleos de processamento* que a GPU possui compensem esse maior gasto de tempo inicial e final e o programa paralelizado se tornasse mais rápido do que o sequencial. Isso ocorreu com os sistemas teste com 118 ou mais barras.

6 CONCLUSÃO

A comparação dos tempos médios de execução dos programas construídos via programação sequencial e paralela, ambos com implementações afins – que fazem uso de forma de armazenamento *densa* –, para a solução do problema do fluxo de potência demonstraram o potencial que a aplicação de técnicas da área da *computação paralela* tem para melhorar o desempenho de aplicações. Para o sistema de 994 barras (ieee118x8) a aceleração promovida foi superior a 30 vezes.

Aplicações essenciais de sistemas de gerenciamento de energia, como estimação de estado e análise de segurança, utilizam o algoritmo de fluxo de potência e podem ser beneficiadas pelos resultados apresentados neste trabalho.

Contudo, além de realizar comparações fazendo uso de outras bibliotecas consagradas para a solução do sistema no programa sequencial, como por exemplo a biblioteca LAPACK, ainda é necessário que sejam feitos estudos que considerem os efeitos da utilização de forma de armazenamento esparsa para as implementações paralelizadas e seriais para que sejam revistas as afirmações de Tylavsky et al. (1992) quanto à ineficiência de implementações paralelizadas do problema do fluxo de potência frente à implementação sequencial fazendo uso de matrizes esparsas.

Agradecimentos

Agradecemos à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio à pesquisa.

Referências

Antonova, G., Nardi, M., Scott, A. and Pesin, M. (2012). Distributed generation and its impact on power grids and microgrids protection, *2012 65th Annual Conference for Protective Relay Engineers*, pp. 152–161.

Christie, R. D. (1963). UW Power Systems Test Case Archive, **URL:** <https://egriddata.org/group/uw-power-systems-test-case-archive>.

Green, R. C., Wang, L. and Alam, M. (2011). High performance computing for electric power systems: Applications and trends, *2011 IEEE Power and Energy Society General Meeting*, pp. 1–8.

Huang, Y., Werner, S., Huang, J., Kashyap, N. and Gupta, V. (2012). State Estimation in Electric Power Grids: Meeting New Challenges Presented by the Requirements of the Future Grid, *IEEE Signal Processing Magazine* **29**(5): 33–43.

Jalili-Marandi, V., Zhou, Z. and Dinavahi, V. (2012). Large-Scale Transient Stability Simulation of Electrical Power Systems on Parallel GPUs, *IEEE Transactions on Parallel and Distributed Systems* **23**(7): 1255–1266.

JáJá, J. (1992). *An introduction to parallel algorithms*, Addison-Wesley Publishing Co., Reading, Massachusetts, USA.

Luebke, D. and Humphreys, G. (2007). How GPUs Work, *IEEE Computer Society Computer Magazine* **40**(2): 96–100.

Mack, C. (2015). The Multiple Lives of Moore's Law, *IEEE Spectrum* **52**(4): 31–31.

McKeown, M., Lavrov, A., Shahradd, M., Jackson, P. J., Fu, Y., Balkind, J., Nguyen, T. M., Lim, K., Zhou, Y. and Wentzlaff, D. (2018). Power and Energy Characterization of an Open Source 25-Core Manycore Processor, *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 762–775.

Monticelli, A. J. (1986). *Fluxo de Carga em Redes de Energia Elétrica*, Edgar Blücher, São Paulo.

Nvidia Corporation (2018). Cuda c programming guide, **URL:** https://docs.nvidia.com/pdf/CUDA_C_Programming_Guide.pdf.

Sambarapu, K. M. and Halpin, S. M. (2007). Sparse Matrix Techniques in Power Systems, *2007 Thirty-Ninth Southeastern Symposium on System Theory*, pp. 194–198.

Sravanthi, G., Grace, B. and kamakshamma, V. (2014). A review of High Performance Computing, *IOSR Journal of Computer Engineering* **16**(1): 36–43.

Tylavsky, D., Bose, A., Alvarado, F., Betancourt, R., Clements, K., Heydt, G., Huang, G., Ilic, M., La Scala, M., Pai, M. A., Pottle, C., Talukdar, S., Van Ness, J. and Wu, F. (1992). Parallel processing in power systems computation, *IEEE Transactions on Power Systems* **7**(2): 629–638.

Zanghi, R. (2016). *Meta-heurísticas aplicadas ao Agendamento de Intervenções em Redes Elétricas*, Tese (Doutorado em Computação), Instituto de Computação, Universidade Federal Fluminense, Niterói.