

Desenvolvimento de uma Ferramenta Intervalar para a Análise de Circuitos Elétricos

José Ubirajara N. de Nunez², Pablo S. Grigoletti¹,
Graçaliz P. Dimuro¹, Luciano V. Barboza², Renata H. S. Reiser¹

¹Escola de Informática – Universidade Católica de Pelotas (UCPel)
Rua Félix da Cunha, 412 – 96010-000 – Pelotas– RS – Brasil

²Escola de Arquitetura e Engenharia – Universidade Católica de Pelotas (UCPel)
Rua Félix da Cunha, 412 – 96010-000 – Pelotas– RS – Brasil
{junnunez, pablogri, liz, lvb, reiser}@ucpel.tche.br

***Abstract.** This paper describes the work that has been done in the project whose objective is to develop a system for the electric circuit analysis, based on the use of interval techniques to control and to estimate numerical errors. Such system is being developed according the free software conception, implemented in Python. A module for the resolution of algebraic linear equation system was developed, including the LU decomposition method. It was also implemented a module for the electric circuit analysis, based on the nodal analysis. On going work is related to the development of an interval version of the LU decomposition method. Future work concerns about the development of a graphic interface and a web based version system.*

***Resumo.** Este artigo descreve o andamento do Projeto cujo objetivo é desenvolver um software para a análise de circuitos elétricos, utilizando técnicas intervalares para controle e estimativa dos erros numéricos. Este programa servirá à concepção de software livre, sendo que Python foi a linguagem escolhida para sua implementação. Foi desenvolvido um módulo para a resolução de sistemas de equações lineares algébricas (SELA's) utilizando o método "decomposição LU". Implementou-se também um módulo para a análise de circuitos elétricos. Atualmente, está em desenvolvimento uma versão intervalar da decomposição LU. Como trabalhos futuros, serão desenvolvidos uma interface gráfica e uma versão web para o sistema.*

1. Introdução

Um dos grandes problemas enfrentados pelos alunos do curso de Engenharia Elétrica é o alto custo dos softwares utilizados. É importante salientar também que existem poucos programas livres (baseados na filosofia de software livre) e de qualidade desenvolvidos para a área de análise de circuitos elétricos.

Um outro problema enfrentado na área de computação numérica são os erros gerados pela incerteza dos dados de entrada, bem como erros oriundos de arredondamentos e truncamentos. São esses processos que causam a perda da exatidão dos resultados teoricamente esperados.

Por estes motivos, o objetivo deste projeto é desenvolver uma ferramenta computacional para auxiliar na *análise de circuitos elétricos*, baseada na concepção de software livre e utilizando técnicas intervalares no controle automático e rigoroso dos erros de resultados de computações numéricas [Moore, 1966; 1979]. Isto permite uma análise da influência das incertezas dos dados de entrada nos resultados obtidos. Este projeto recebe apoio financeiro do programa CTINFO/CNPq e FAPERGS.

2. Análise de Circuitos Elétricos

A análise de circuitos elétricos utiliza-se fundamentalmente de métodos da Álgebra Linear que podem exigir um grande esforço computacional. As técnicas para análise de circuitos mais conhecidas são: *análise de malhas* e *análise nodal*, as quais baseiam-se nas *Leis de Kirchhoff* [Hilburn, Johnson e Johnson, 1994; Irwin, 2000].

Estes métodos de análise geram sistemas de equações lineares de n equações e n incógnitas, cuja solução estima os valores das incógnitas de um circuito elétrico, que podem ser de dois tipos: tensão ou corrente. As matrizes geradas são geralmente esparsas, mas é usual a utilização de métodos diretos com técnicas de esparsidade para a solução desses sistemas.

Ao percorrer as malhas de um circuito elétrico, obtém-se, como incógnitas, as tensões sobre os elementos que compõe as malhas (*Lei das Tensões de Kirchhoff*). Porém, se for realizada uma análise das correntes que entram ou saem de um nó, têm-se, como incógnitas, as correntes que percorrem os ramos do circuito (*Lei das Correntes de Kirchhoff*).

Optou-se por utilizar a *técnica de análise nodal*. Observa-se que é exigido muito mais esforço computacional para identificar as malhas de um circuito do que seus nós (pois eles já fazem parte dos dados de entrada), o que justifica a nossa escolha.

Inicialmente, os circuitos analisados podem conter apenas resistores e fontes independentes de corrente e de tensão.

2.1. Análise Nodal

A análise nodal é um método de análise circuitos baseado na Lei das Correntes de Kirchhoff. Nela deve-se escolher um ponto como referência (ponto 0) e após arbitrar, aos demais nós, as respectivas tensões do circuito a serem calculadas. No lado esquerdo das equações nodais, tem-se o somatório dos produtos "condutância x tensão". Do lado direito dessas equações tem-se uma corrente positiva (proveniente de uma fonte de corrente) se a corrente estiver chegando ao nó, e uma corrente negativa, se a corrente estiver saindo do nó.

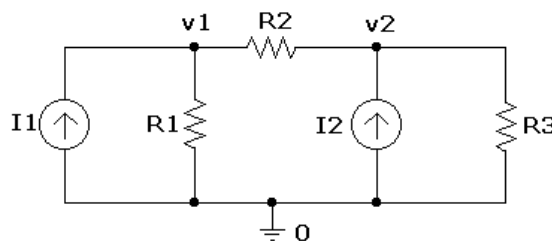


Figura 1. Circuito elétrico sem fontes de tensão

A análise nodal, em muitos casos, torna-se mais fácil quando aplicada a circuitos que contenham fontes de tensão. A existência destas fontes acarreta o surgimento de *supernós* e de *nós aparentes*. Associadas a eles, tem-se as equações vinculares.

Um *supernó* é aquele em que dois nós terminais a uma fonte de tensão estão conectados em dois pontos quaisquer do circuito; e um *nó aparente* é um *supernó* com um de seus nós terminais conectado à referência (ponto 0).

Na análise nodal, as variáveis (incógnitas) são as tensões nodais, portanto, em um *nó aparente*, já se conhece o valor da incógnita.

Para um *supernó*, primeiramente escreve-se a sua equação vincular que é a diferença entre as tensões existentes nos terminais da fonte de tensão. A seguir, trata-se o *supernó* como se fosse um único nó e escreve-se a sua equação nodal. Esta se caracteriza pelo produto "condutância x tensão", de forma análoga a explicada na seção anterior.

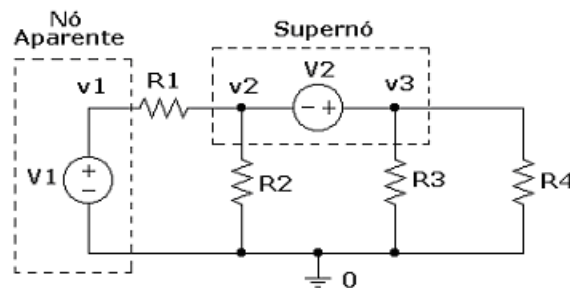


Figura 2. Circuito elétrico com fontes de tensão

3. Resolução de SELA's

A *análise nodal* [Hilburn, Johnson e Johnson, 1994; Irwin, 2000] é uma técnica de análise de circuitos que gera um número de equações igual ao número de nós (incógnitas) menos um (nó de referência).

Sabe-se que no caso de sistemas $n \times n$, a solução de um sistema do tipo $Ax=b$ é dado por $x=A^{-1}b$, onde os valores encontrados para x , correspondem à solução do sistema. Porém o cálculo da matriz inversa A^{-1} requer um grande esforço computacional, pelo número de operações envolvidas e pela complexidade dessas operações.

Portanto, para a resolução de SELA's no aplicativo em desenvolvimento, foram estudados alguns métodos, como, por exemplo, eliminação de Gauss e decomposição LU.

3.1. Eliminação de Gauss

Eliminação Gaussiana [Claudio e Marins, 2000] é a técnica mais conhecida e mais usada para a resolução de SELA's densos de pequeno a médio porte (sistemas de dimensão máxima 30). Este método consiste na aplicação de operações elementares sobre as linhas de uma matriz aumentada.

O Método de Gauss transforma o sistema linear original num sistema linear equivalente, cuja matriz dos coeficientes é triangular superior. A resolução deste sistema equivalente é imediata.

O método dividi-se em duas etapas, o que torna a implementação mais fácil.

A primeira etapa é chamada de *triangularização* e consiste em transformar a matriz dos coeficientes A numa matriz triangular superior, mediante permutações e combinações lineares. Observa-se que as operações realizadas sobre a matriz A devem também ser aplicadas sobre o vetor independente b .

A segunda etapa do método é conhecida como substituição inversa e tem por objetivo o cálculo dos componentes do vetor x , solução de $Ax=b$, a partir da solução imediata do último componente de x , e a substituição regressiva nas equações anteriores.

3.2. Decomposição LU

A *Decomposição LU* [Anton, 2001], consiste na decomposição da matriz A em um produto de duas matrizes triangulares e, em seguida, na solução de dois sistemas triangulares que fornecem a solução do sistema original.

Considerando um sistema do tipo $Ax=b$, e sendo A uma matriz quadrada, então pode-se escrever $A=LU$, onde L é uma matriz triangular inferior unitária e U uma matriz triangular superior.

Se $Ax=b$ e $A=LU$ então $LUx=b$ e considerando que $Ux=y$, obtém-se dois novos sistemas lineares:

$$Ly=b \quad (1)$$

$$Ux=y \quad (2)$$

A partir da solução da Eq. (1), são obtidos os componentes de y , e portanto, com a substituição do vetor y na Eq. (2), obtém-se a solução desta, encontrando os valores dos componentes do vetor x .

A Eq. (1) é solucionada por *substituição direta* e a Eq. (2) por *substituição inversa*.

Na prática, na busca de soluções para minimizar a instabilidade numérica, deve-se optar por uma modificação deste método, denominada *Decomposição LUP* [Cormen et al., 2002]. O objetivo desta decomposição é encontrar três matrizes L , U e P , com dimensão $n \times n$, de modo que $PA = LU$, onde L é uma matriz triangular inferior unitária, U é uma matriz triangular superior e P é uma matriz de permutação (inicialmente uma matriz identidade).

Considerando um sistema do tipo $Ax=b$ e multiplicando ambos os membros da equação por uma matriz P , obtém-se $P Ax=P b$ e se $PA=LU$, então $LUx=Pb$. Considerando $Ux=y$, obtém-se:

$$Ly=Pb \quad (3)$$

$$Ux=y \quad (4)$$

Na Eq. (3), tem-se um sistema triangular inferior e obtém-se a solução para o vetor y através de uma substituição direta. De posse da solução para y , resolve-se o sistema da Eq. (4). Este sistema é triangular superior e dele obtém-se a solução para o vetor x através de uma *substituição inversa*.

Na prática, observa-se que com o pivotamento, a matriz A (matriz original) será alterada em função das trocas de linhas, portanto o vetor b também deveria ser modificado para que a solução do sistema não se altere. É para resolver este problema que existe a matriz P , para poder realizar no vetor b todas as trocas de linha que ocorreram no processo de decomposição da matriz A .

Portanto, a decomposição LUP oferece uma maior estabilidade e robustez numérica na resolução dos SELA's do que os métodos anteriores (método de Gauss e decomposição LU), pois esta permite o pivotamento parcial de linhas e possui um esforço computacional equivalente. Justifica-se, assim, a escolha deste método para a implementação do módulo para resolução das SELA's resultantes da aplicação da análise nodal.

4. Por Que Usar Python?

A linguagem *Python* [Catunda, 2001; Chun, 2001; Matthew e Stones, 2002] foi escolhida por ser poderosa e de fácil e rápido aprendizado. Ela possui estruturas de dados de alto-nível eficientes, bem como adota uma abordagem simples e efetiva para a programação orientada a objetos. Sua sintaxe elegante e tipagem dinâmica, em adição a sua natureza interpretada, tornam Python ideal para *scripting* e para o desenvolvimento rápido de aplicações (RAD - Rapid Application Development) em diversas áreas e na maioria das plataformas. O interpretador de Python e sua extensa biblioteca padrão estão disponíveis na forma de código fonte ou binário para a maioria das plataformas, e podem ser distribuídos livremente.

Além disso, este interpretador é facilmente extensível incorporando novas funções e tipos de dados implementados em C ou C++ (ou qualquer outra linguagem acessível a partir de C), seja para desempenhar operações críticas em máxima velocidade, ou para vincular programas Python a bibliotecas que só estejam disponíveis em formato binário.

Esta linguagem permite organizar programas em módulos que podem ser reutilizados em outros programas escritos em Python. A linguagem provê uma vasta coleção de módulos que podem ser utilizados como base para as aplicações.

Python é uma linguagem interpretada, que pode fazer com que se economize um tempo considerável durante a etapa de desenvolvimento, uma vez que não há necessidade de compilação e vinculação.

É possível construir programas compactos e legíveis, tipicamente mais curtos do que seus equivalentes em C ou C++, por diversas razões:

- os tipos de alto nível permitem que expressar operações complexas em um único comando;
- a definição de bloco é feita por indentação ao invés de marcadores de início e fim de bloco;

- não há necessidade de declaração de variáveis ou parâmetros formais.

Além disso, a linguagem Python possui vários módulos desenvolvidos para a área de computação científica. No desenvolvimento do *analisador de circuitos elétricos* foi utilizada sua biblioteca numérica.

4.1. Biblioteca Numérica

Essa extensão numérica para o Python acrescenta poderosos objetos de array multi-dimensionais à linguagem. Esses novos objetos possibilitam a programação de estruturas matriciais semelhante às disponíveis nas linguagens Matlab e IDL, além de manter todas as vantagens já mencionadas anteriormente.

Esta extensão torna eficiente a manipulação de matrizes de diversos tipos de números de máquina homogêneos (flutuantes, longos, duplos, complexos etc.), possibilitando um número arbitrário de dimensões e também operações estruturais sofisticadas.

5. Técnicas Intervalares

Observa-se que a computação numérica implementada no sistema desenvolvido foi baseada em algoritmos pontuais. Entretanto estes algoritmos geram uma estimativa para a resposta.

Freqüentemente, nem sempre é possível garantir a exatidão da resposta estimada sem o auxílio de uma análise de erro, que é extensa, dispendiosa e muitas vezes inviável. Até mesmo quando uma análise de erro é executada, o número resultante é somente uma estimativa de erro que pode estar presente.

Por outro lado, as *técnicas intervalares* [Moore, 1966; 1979] computam um intervalo, com garantia de que o resultado pertença a este intervalo. Portanto, resultados intervalares carregam consigo a segurança de sua qualidade.

Entretanto, obter uma resposta intervalar não garante que ela contenha algo de interesse. Atingir uma inclusão significativa requer uma fundamentação matemática cuidadosa de todos os estágios do desenvolvimento do algoritmo e a sua implementação. Os algoritmos a serem desenvolvidos devem ser *algoritmos intervalares* e não *versões intervalares de algoritmos pontuais*.

Assim, espera-se que as técnicas intervalares forneçam garantias, e que possam ser aplicadas quase que automaticamente na resolução de SELA's. Pretende-se implementar uma nova versão da *Decomposição LUP*, utilizando técnicas intervalares, com a finalidade de alcançar limites garantidos para os resultados, através do controle rigoroso dos erros do resultado.

6. Considerações Finais

Os estudos desenvolvidos propiciaram a implementação de uma versão inicial do sistema para a análise de circuitos elétricos. Salienta-se que este software poderá ser livremente distribuído.

Para facilitar a utilização do mesmo, será desenvolvido uma interface gráfica amigável para o software, semelhante as dos programas comerciais utilizados na análise

de circuitos elétricos. Desta forma, qualquer usuário com pouca experiência em informática poderá utiliza-lo com facilidade.

Essa interface se faz necessária pois atualmente toda a entrada de dados é realizada com a utilização de um arquivo texto. Usufruindo as potencialidades disponíveis em uma biblioteca gráfica para Python, denominada wxPython, pretende-se criar programas robustos, com interfaces altamente funcionais, de forma rápida e simples. Da mesma forma que o Python, a wxPython tem código-fonte aberto e é multi-plataforma, com isso o software poderá rodar em múltiplas plataformas, sem precisar de nenhuma modificação.

Ressalta-se que, mesmo sendo o Python uma linguagem interpretada, com a utilização de uma biblioteca especial, é possível compilar seus programas. Com o programa compilado, não existe a necessidade de ter a linguagem Python e nem as bibliotecas utilizadas pelo aplicativo instaladas no computador. Neste contexto, outro trabalho a ser desenvolvido futuramente é disponibilizar o sistema em uma versão compilada.

Por fim, propõe-se o desenvolvimento de uma versão para web do software, de tal forma que este possa ser utilizado principalmente em cursos on-line. A versão web será uma segunda alternativa para quem não possui a linguagem Python instalada.

Referências Bibliográficas

- Anton H. Álgebra Linear com Aplicações. 8.ed. São Paulo: Bookman, 2001.
- Brown M. C. Python. s.l: McGraw-Hill Companies, 2001.
- Catunda M. Python: Guia de Consulta Rápida. 1.ed. São Paulo: Novatec, 2001.
- Chun W. J. Core Python Programming. s.l: Prentice Hall PTR, 2001.
- Claudio. D. M.; Diverio. T. A.; Oliveira P. W. Fundamentos da matemática intervalar. Porto Alegre: Sagra-Luzzatto, 1997.
- Claudio D. M.; Marins J. M. Cálculo Numérico Computacional: Teoria e Prática. 3.ed. São Paulo: Atlas, 2000.
- Cormen T. H. et al. Algoritmos: Teoria e Prática. 1.ed. Rio de Janeiro: Campus, 2002.
- Dimuro G. P. Domínios Intervalares da Matemática Computacional. Porto Alegre: CPGCC da UFRGS, 1990.
- Hilburn J. L.; Johnson D. E.; Johnson J. R. Fundamentos de Análise de Circuitos Elétricos. 4.ed. Rio de Janeiro: LTC, 1994.
- Irwin J. D. Análise de Circuitos em Engenharia. 4.ed. São Paulo: Makron Books, 2000.
- Matthew N.; Stones R. Professional Linux Programando. 1ed. São Paulo: Makron Books, 2002.
- Moore, R. E. Interval Analysis. Englewood: Prentice-Hall, 1966.
- Moore, R. E. Methods and Applications of Interval Analysis. Philadelphia: SIAM, 1979.