



**XXII SNPTTE
SEMINÁRIO NACIONAL
DE PRODUÇÃO E
TRANSMISSÃO DE
ENERGIA ELÉTRICA**

BR/GDS/15
13 a 16 de Outubro de 2013
Brasília - DF

GRUPO - X

GRUPO DE ESTUDO DE DESEMPENHO DE SISTEMAS ELÉTRICOS – GDS

EMPREGO DE CONTROLES PROPRIETÁRIOS EM ESTUDOS DE TRANSITÓRIOS ELETROMAGNÉTICOS

Venilton R. Oliveira(*) JORDÃO ENGENHARIA	Antonio C. S. Lima UFRJ	Rodrigo Godim JORDÃO ENGENHARIA	Victor A. Teixeira JORDÃO ENGENHARIA	Braulio Chuco JORDÃO ENGENHARIA
--	--	--	---	--

RESUMO

O artigo apresenta o procedimento para incluir dentro do ATP modelos onde se emprega modelos desenvolvidos em linguagens de programação e conectados ao ATP. Esse procedimento é realizado via chamadas especiais da MODELS. Todo o sistema foi desenvolvido usando o ATPDraw para a elaboração do circuito. As funções externadas foram compiladas com a versão em biblioteca do ATP a partir do pacote MinGW (*Minimalist GNU for Windows*). Foram realizados testes das funções externas empregando-se três linguagens de programação distintas: C, C++ e Fortran.

PALAVRAS-CHAVE

ATP, C++, *Foreign Function*, Fortran, Transitórios Eletromagnéticos

1.0 - INTRODUÇÃO

A renovação do parque gerador brasileiro com a inclusão de fontes renováveis, a entrada das novas estações conversoras para a transmissão de energia da região Amazônica bem como as possíveis interligações transnacionais indicam um crescente uso de conversores de eletrônica de potência. Tipicamente estes dispositivos demandam complexos sistemas de controle que necessitam de ser modelados detalhadamente quando estudos de transitórios eletromagnéticos são considerados. Muitas das vezes, este tipo de abordagem envolveria o uso de tecnologias proprietárias ou mesmo protegidas por patentes. Tipicamente, esses modelos são desenvolvidos em linguagens de programação avançada, como por exemplo, FORTRAN, C ou C++. Esse procedimento permite a geração de *objetos* (componentes compilados) que permitem tanto a proteção do código fonte do controle, como da exportação do mesmo para diferentes plataformas, como PSCAD/EMTDC, ETMP-RV, MATLAB.

Este trabalho tem por objetivo apresentar a possibilidade de implementação de funções em diferentes linguagens de programação dentro do programa ATP e comparar o comportamento das mesmas entre si e com a linguagem própria do ATP (models). Para a utilização das novas funções compiladas no ATP, as mesmas devem ser chamadas através do recurso *foreign function*, ou seja, uma função externa, sendo usada de uma forma semelhante a uma função existente dentro do código do ATP (1)(2). Um fato que se mostra interessante nesta recompilação, é que o código fonte do controle permanece inalterado e interno ao programa ATP, fazendo com que não haja mais acesso ao código fonte original. Além de manter as mesmas funções de controle desenvolvidas pelos fabricantes dos equipamentos.

Para ilustrar esse procedimento, optou-se por apresentar um procedimento bastante simples e que poderia ser facilmente implementado via MODELS ou mesmo empregando a TACS. Dentro das possibilidades existentes foi

(*) Av. Rio Branco, n° 120 – Grupo 830 – CEP 99.999-999 Rio de Janeiro, RJ, – Brasil
Tel: (+55 21) 2413-5743 – Fax: (+55 21) 3553-4902 – Email: venilton.rodrigues@jordaoengenharia.com.br

decidido o uso do emprego de um modelo capaz de calcular o valor eficaz agregado, i.e., cálculo do valor médio quadrático da tensão. Para os testes considera-se que há um curto-circuito e o modelo deve apresentar o cálculo de forma eficiente e precisa.

2.0 - INTRODUÇÃO À LINGUAGEM DE PROGRAMAÇÃO MODELS DO ALTERNATIVE TRANSIENT PROGRAM

O ATP possui uma extensa biblioteca de componentes elétricos e controles disponíveis ao usuário para simulações eletromagnéticas. Além dessa gama de componentes ainda permite ao usuário programar dentro do ambiente ATP construindo suas próprias funções de medição e controle. A linguagem, denominada MODELS, é uma linguagem própria do ATP, estruturada e com sintaxe similar ao Fortran ou Pascal. Cada MODELS conta com uma estrutura básica apresentada na Figura 1

```
MODEL DEFAULT
DATA d1,d2
INPUT i1,i2
OUTPUT o1,o2
VAR o1,o2
INIT
  o1:=0
  o2:=1
ENDINIT
EXEC
  o1:=(i1-i2)*t
  o2:=sin(o1)
ENDEXEC
ENDMODEL
```

Figura 1 – Modelo Básico de uma MODELS

Nessa estrutura a primeira linha consiste na definição do nome da MODELS. A segunda estabelece as variáveis que estão disponíveis na interface com o usuário, isto é, são os parâmetros definidos pelo usuário. É opcional visto que nem todos os modelos precisam de dados fornecidos pelo usuário.

As variáveis classificadas como INPUT consistem em dados de entrada do modelo. Os tipos de dados suportados como entradas (INPUT) totalizam nove ao todo e estão descritos na Tabela 1. Posteriormente temos as variáveis denominadas OUTPUT, que são, como próprio nome indica, variáveis de saída. Essas variáveis podem por sua vez ser conectadas a TACS, redes elétricas e inclusive outras MODELS. Por fim, o último tipo de variáveis é chamado de VAR. Uma observação que merece destaque é que todos os tipos de variáveis declarados como OUTPUT devem ser declarados também como VAR, assim como as demais variáveis auxiliares e intermediárias. O tipo VAR é aquele que se encontra disponível no arquivo de plotagem.

Tabela 1 – Tipos de Entrada da MODELS

ENTRADA DA MODEL	
Input Currente	Nó da corrente
Input Voltage	Nó da tensão
Input Switch	Status da chave
Input Machine	Variável da máquina
Input TACS	Variável da TACS
Input Im(SSU)	Parte imaginária da tensão em regime permanente
Input Im(SSI)	Parte imaginária da corrente em regime permanente
Input Model	Saída de outra MODEL
Input ATP	Saída de outro modelo

O bloco de instrução que segue de INIT até ENDINIT representa a inicialização do modelo. Isto é, são os valores que as variáveis assumem no instante t_0 -. Nesse bloco deve ser realizados os cálculos preparatórios e demais inicializações.

O bloco de instrução que segue do EXEC até o ENDEXEC é onde acontece o fluxo principal da MODELS. O ATP utiliza o método de integração trapezoidal e cada EXEC o tempo é incrementado de um passo de integração.

3.0 - USO DE "FOREIGN FUNCTION"

Apesar da grande flexibilidade do uso da linguagem MODELS, essa não se mostra vantajosa nas seguintes situações:

- Necessidade de modularização do código;
- Manutenção de segredo tecnológico;
- Uso direto do código do equipamento/fabricante;
- Uso de bibliotecas personalizadas.

Por modularização do código entende-se o fracionamento das funções utilizadas. Essa metodologia torna o processo de manutenção e desenvolvimento bem menos laborioso além de ser um facilitador para codificar em equipe. A manutenção do segredo tecnológico é um requisito muito importante para o fabricante que não deseja ter seu código exposto. Assim é possível implementar a tecnologia sem receios de espionagem ou plágios de qualquer natureza. Complementar a isso vem ao uso direto do código do fabricante. A possibilidade de utilizar o mesmo código do fabricante é uma possibilidade muito vantajosa, uma vez que é possível simular com condições bem semelhantes aquelas encontradas no ambiente real. Tal procedimento evita o surgimento de erros devido a importação/exportação do código bem como problemas com a "tradução" do mesmo. Por fim, o uso de bibliotecas personalizadas permite que o desenvolvedor possa utilizar biblioteca de funções importadas ou criadas por sua equipe. É equivalente a uma capacidade de expansão do ATP. Entretanto o uso dessas funções desenvolvidas em outras linguagens no ATP não é uma tarefa trivial. Consiste em trabalhar na edição de diversos arquivos de forma a compatibilizar o ATP e através de um processo de recompilação e *linkagem* que permite gerar um novo executável para o ATP contendo as novas funções implementadas.

4.0 - LINGUAGEM DE PROGRAMAÇÃO UTILIZADAS

Para o desenvolvimento desse trabalho foi considerado o uso de algumas linguagens de programação, a saber: FORTRAN, C e C++. O motivo da escolha dessas três consiste no fato que todas são voltadas para a computação científica e cálculo numérico. As principais vantagens e características de cada uma delas são brevemente descritas a seguir.

4.1 FORTRAN

A linguagem de programação FORTRAN (Formula Translation) é uma linguagem de alto nível, criada com a partir da dificuldade de se trabalhar com Assembly. Esta linguagem prima por programas com velocidade de execução. É devido a esse fator que é extensivamente utilizado em aplicações científicas. Programas do setor elétrico tem em comum o fato de possuir o código-fonte em FORTRAN. O ATP, por exemplo, é desenvolvido em FORTRAN77 e FORTRAN90, EMTP-RV em FORTRAN 95, ANAREDE e ANATEM foram implementados originalmente também em FORTRAN77.

4.2 C

A linguagem de programação C é uma das linguagens de programação utilizadas por diversos tipos de desenvolvedores. Esta linguagem foi base para criação de planilhas eletrônicas a sistemas operacionais como UNIX e Linux. Trata-se de uma linguagem muito simples com funções matemáticas, mas requer a inclusão de bibliotecas padrão da própria linguagem. Permite trabalhar com tipos de dados simples e possui acesso direto à memória. As variáveis são definidas e declaradas de forma simples. Entre algumas das desvantagens dessa linguagem é que a mesma não possui o chamado *multithreading*, isto é não permite que se executem várias tarefas simultaneamente, não faz uso de classes ou objetos, nem possui coleta automática de lixo (capacidade de eliminação de variável não utilizada ocupando espaço na memória).

4.3 C++

A linguagem de programação C++ é uma linguagem bastante flexível, multiparadigma, permite a orientação a objeto bem como a programação procedimental. Trata-se de uma linguagem altamente expressiva e lógica. Com o uso da orientação a objetos é possível o reuso do código de forma mais lógica e produtiva. É possível empregar códigos em C como parte de um código C++. Apresenta uma compilação bem eficiente devido a sua dualidade de linguagem alto e baixo nível.

5.0 - COMPILAÇÃO DAS FUNÇÕES EXTERNAS

Um modelo externo ou função externa deve ser compilado e *linkado* com o ATP para gerar um novo ATP com funções específicas de maneira que as mesmas se tornem internas ao próprio programa. Inicialmente se deve modificar os arquivos *newmods.fedimdef.f* com os valores de dimensão necessários. Esses arquivos devem, a princípio, estar no mesmo diretório onde será compilado o código da função externa. Após essa etapa esses

arquivos são *linkados* com a biblioteca já compilada do ATP e com a biblioteca gráfica *dislin*. Ao final desse processo é gerado um novo arquivo executável do ATP com as funções em códigos C, C++ e Fortran internas ao código original. As rotinas externas devem ser declaradas dentro do arquivo *fgnmod.f* para que o ATP possa chama-las como *foreignfunctions*, esse arquivo é um dos mais importantes desse processo. Todo o procedimento pode ser resumido na estrutura apresentada na Figura 2 abaixo. A partir do uso de *makefiles* e do utilitário *make* é possível automatizar praticamente todo esse procedimento. O pacote MinGW (*Minimalistic GNU for Windows*) que também é usado para gerar a versão GNU do ATP pode ser usado para esse processo de automatização (3)(4).

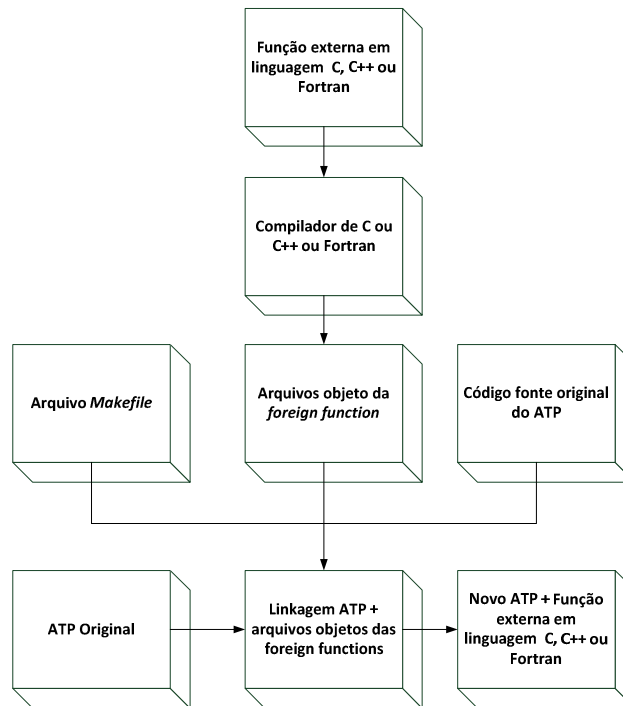


Figura 2 – Processo de compilação e linkagem

6.0 - MODELO PROPOSTO PARA VALIDAÇÃO

Conforme mencionado na introdução do presente informe, optou-se por implementar um medidor de valor eficaz agregado para o procedimento de uso de função externa. A definição do valor eficaz agregado para um sistema trifásico é dada por

$$V_{RMS} = \sqrt{(v_a(t)^2 + v_b(t)^2 + v_c(t)^2)}(1)$$

Na Figura 3 está apresentado o modelo desenvolvido no ATPDraw utilizado para validação. Consiste em um circuito RL onde uma impedância de curto circuito é inserida. Foram criadas quatro modelos externos, i.e. *foreignfunction* da MODELS, cada qual contendo uma implementação do cálculo em um linguagem distinta, incluindo-se a própria MODELS. Todas as funções externas implementadas sofreram o mesmo processo de codificação e apresentam a mesma quantidade de parâmetros. A única diferença sobre o ponto de vista da interface do ATP consiste no nome das funções utilizadas.

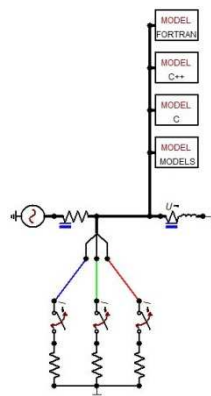


Figura 3 – Modelo no ATPDraw

Na Tabela 2 estão apresentados os nomes das funções implementadas e sua respectiva linguagem de programação utilizada. Na Figura 5 está destacado a função externa implementada na linguagem C. Na chamada da função externa não importa a linguagem de programação e que foi desenvolvida. Isto é, o quadro poderia se referir à função em Fortran ou em C++. A primeira parte do quadro mostra a MODELS como um todo. A segunda parte destaca a declaração da função e definição do número de argumentos. Nota-se que os argumentos são a composição entre as variáveis de entrada, de estado e de saída. Por fim, na terceira parte o enfoque está no bloco EXEC com a passagem de parâmetros para a função, bem como sua chamada.

Tabela 2

FUNÇÕES EXTERNAS	
CRMS	C
CPRMS	C++
FRMS	Fortran

MODEL MODELS

```

INPUT Vin[1..3]
VAR Vrms
INIT
Vrms:= 0
ENDINIT
EXEC
Vrms:=SQRT(Vin[1]*Vin[1]+Vin[2]*Vin[2]+Vin[3]*Vin[3])
ENDEXEC
ENDMODEL

```

Figura 4 – Código interno da MODELS

<pre> MODEL C INPUT Vin[1..3] VAR IN[1..4], OU[1..4] FUNCTION CRMS FOREIGN CRMS {ixarg: 4} INIT IN[1..4] := 0 ENDINIT EXEC --Entradas IN[1]:=Vin[1] IN[2]:=Vin[2] IN[3]:=Vin[3] OU[1..4] := CRMS(IN[1..4]) ENDEXEC ENDMODEL </pre>	<pre> MODEL C INPUT Vin[1..3] VAR IN[1..4], OU[1..4] FUNCTION CRMS FOREIGN CRMS {ixarg: 4} INIT IN[1..4] := 0 ENDINIT EXEC --Entradas IN[1]:=Vin[1] IN[2]:=Vin[2] IN[3]:=Vin[3] OU[1..4] := CRMS(IN[1..4]) ENDEXEC ENDMODEL </pre>	<pre> MODEL C INPUT Vin[1..3] VAR IN[1..4], OU[1..4] FUNCTION CRMS FOREIGN CRMS {ixarg: 4} INIT IN[1..4] := 0 ENDINIT EXEC --Entradas IN[1]:=Vin[1] IN[2]:=Vin[2] IN[3]:=Vin[3] OU[1..4] := CRMS(IN[1..4]) ENDEXEC ENDMODEL </pre>
--	--	--

Figura 5 – Demonstração do uso da “foreignfunction” dentro da MODEL

Todas as linguagens tiveram o mesmo cálculo implementado. Na Figura 6 está apresentado o código implementado na linguagem C, por entender que se trata de uma linguagem de programação de uso bem popular. Os demais códigos, em C++ e Fortran, tem o mesmo processo. Isso motiva o fato que basta saber programar em qualquer uma das linguagens de programação apresentadas para ser capaz de gerar suas próprias funções externas. Este fato é bastante motivador, pois torna o programa de simulação eletromagnética bastante flexível.

Com posse das quatro funções foi gerada uma simulação com 6s de simulação, com a ocorrência de curto circuito trifásico de duração de 100ms ocorrendo no instante de 0,2s. Esse resultado encontra-se na Figura 7, onde todas as curvas estão sobrepostas. A sobreposição dos resultados indica sucesso na validação. O resultado pode ser obtido de quatro formas distintas com a mesma precisão.

```

#include <stdlib.h>
#include <math.h>
#include "func.h"

double *crms_(double arg[])
{
double va2,vb2,vc2;
double vrms;
va2=pow(arg[0],2);
vb2=pow(arg[1],2);
vc2=pow(arg[2],2);
vrms = sqrt(va2+vb2+vc2);
return;
}

```

Figura 6 – Código do arquivo *crms.c*

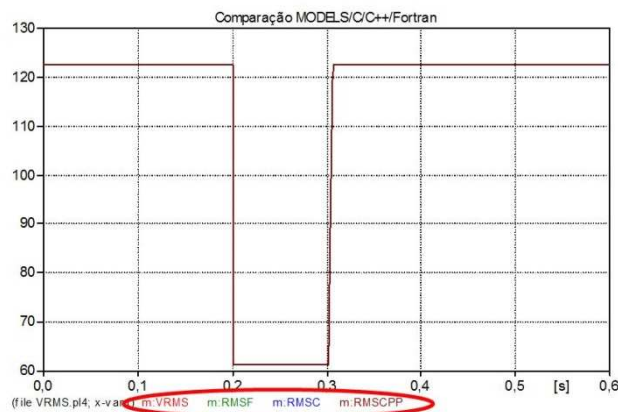


Figura 7 – Validação de todas as “ForeignFunctions”

Um ponto importante é a verificação se a inclusão do modelo externo penaliza o desempenho do arquivo, o que poderia gerar problemas no caso de simulações de redes mais extensas. Para verificar isso foi realizado um teste onde apenas um dos módulos externos é avaliado. Para cada caso foram realizadas três simulações e avaliado o tempo total de execução do arquivo, a Tabela 3 resume esses resultados. Nota-se que mesmo para esse caso simples não houve uma penalização, sendo que todos os casos apresentaram basicamente a mesma performance numérica com uma pequena vantagem para o desempenho da MODELS. Embora não apresentado em detalhes nesse informe foi verificado que no caso de entradas envolvendo relações matriciais haveria uma inversão do desempenho. As versões com os módulos externos passam a apresentar um sensível ganho computacional, maiores detalhes sobre a implementação de “ForeignFunctions” envolvendo funções matriciais podem ser encontrados em (5).

Tabela 3 – Tempo para simulação

FUNÇÕES				
Linguagem	Tempo (s)			Média (s)
MODELS	3,073	3,011	3,089	3,0577
C	3,463	3,510	3,635	3,5360
C++	3,354	3,276	3,307	3,3123
Fortran	3,151	3,120	3,1147	3,1147

7.0 - CONCLUSÃO

Este trabalho apresentou a implementação de modelos de controle externos para serem usados com o ATP, suas diferenças e comparações de diferentes maneiras. Foram utilizadas a linguagem nativa do programa e mais três linguagens de programação voltadas para o uso científico e numérico. As soluções apresentadas permitem um uso mais fácil e flexível para novas modelagens e simulações de circuitos elétricos, podendo ser adicionadas novas ferramentas. A metodologia mostrou-se capaz ainda de criar um código modularizado, possibilitando a personalização de funções e o aproveitamento de rotinas do próprio fabricante, protegendo o segredo industrial.

8.0 - REFERÊNCIAS BIBLIOGRÁFICAS

- (1) F.Janieck, "Distance Digital Relay ModelModel Developed in ATP "Foreign Model" and C++", Journal of ELECTRICAL ENGINEERING , VOL. 57, NO. 5,2006, 268-275
- (2)O.Hevia, "Compilacion del ATP al alcance del usuário," Revista Iberoamericana del ATP, vol.2, 2002, CAUE
- (3) R. G. F. Espinoza, "Implementação e Simulação de Relé de Distância no EMTP/ATP utilizando MODELS Externos Programados em ANSI C", XVIII Congresso Brasileiro de Automática/12 a 16 – setembro-2010, Bonito-MS
- (4) M. Kezunovic, "Interactive Protection System Simulation Using ATP MODELS and C++"
- (5) G. F. Santos Junior, "Metodologia para Análise de Linhas de Transmissão Incluindo Modelos de Arco Secundário, " Tese de Doutorado, Universidade Federal do Rio de Janeiro, Out. 2009, disponível em: <http://www.pee.ufrj.br/teses/?Resumo=2009101602>

9.0 - DADOS BIOGRÁFICOS

Venilton Rodrigues de Oliveira
 Natural do Rio de Janeiro. Ano de nascimento: 1968
 Graduação: Universidade Federal Fluminense: 1992
 Mestrado: Universidade Federal Fluminense: 1996

Antonio Carlos Siqueira de Lima
 Natural do Rio de Janeiro. Ano de nascimento: 1971
 Graduação: Universidade Federal do Rio de Janeiro: 1995
 Mestrado: Universidade Federal do Rio de Janeiro: 1997
 Doutorado: Universidade Federal do Rio de Janeiro: 1999

Rodrigo Godim
 Natural do Rio de Janeiro. Ano de nascimento: 1986
 Graduação: Universidade Federal Fluminense: 2011

Victor Teixeira
 Natural do Rio de Janeiro. Ano de nascimento: 1983
 Bolsista na Jordão Engenharia: 2011