Faster Than Real-Time (FTRT) Dynamics Simulation

Type 3 Wind Turbine Model

Prof. Alexander Flueck, Ming Chen & Xinyi Yu

Department: ECE Theme: Energy

IIT Armour College of Engineering
ILLINOIS INSTITUTE OF TECHNOLOGY

Armour R&D

Background

The goal of “faster than real-time power system dynamics simulation” is to provide actionable information to operators during power system disturbances and cascading outages. The project is developing high performance computer modeling and simulation software for predicting the effects of disturbances faster than real-time. Based on the predictive capabilities of this research, operators will be able to respond before the full effects of a cascading outage are realized, thereby avoiding wide spread blackouts.

Objectives

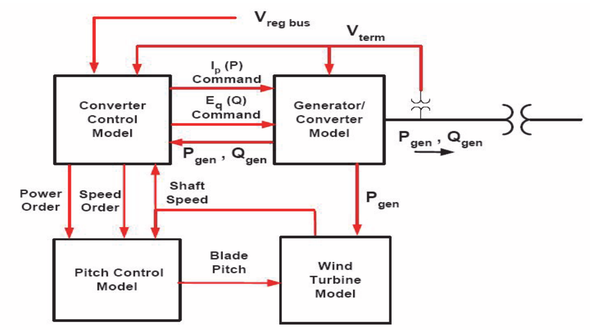
- ❖ The dynamic modeling of Type 3 Generic Wind Turbine in Simulink was done in the last summer. Research in the semester continues.
- ❖ Provide documentation of implementing model into high level programming code.
- ❖ Build the C code for the model and connect them into the main TS3ph Simulator code base.

Tasks

- ❖ Translating the block diagram into equations and write pseudo-code to develop programmer’s guide to help future researchers translate model into C code.
- ❖ Build ResidualFunction for all the models in WT3 in C code.
- ❖ Edit and add necessary files in the secure sever to compile and run the new model.

Introduction to Generic Wind Turbine Type 3

The WT3 PSS®E wind turbine stability model was developed to simulate performance of a wind turbine employing a doubly fed induction generator (DFIG) with the active power controlled by a power converter connected to the rotor terminals.



Methods

- Develop Residual Function based on Simulink models

Sum Function:

$$V_1 = K_{2z} + V_y * K_{pll}/omega0$$

Saturation3:

$$V_{1z} = \begin{cases} Pllmax & if\ V_1 > Pllmax \\ V_1\ if\ V_1 \le Pllmax\ and\ V_1 \ge -Pllmax \\ -Pllmax & if\ V_1 < -Pllmax \end{cases}$$

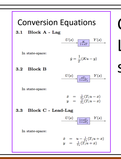
Write equations for limiters and logic operators.

Transfer Fcn (state-space1): $\frac{1}{1+0.02z}$

$$\dot{I_{Xinj}} = \frac{-1}{0.02} I_{Xinj} + \frac{1}{0.02} I_{pcmd2}$$

Transfer Fcn2 (state-space2): $\frac{1}{1+0.02z}$

$$\dot{E_q} = \frac{-1}{0.02} E_q + \frac{1}{0.02} E_{qcmd}$$



Convert control block from Laplace domain to state-space equations.

- Build pseudo-code to get a whole picture of each model

```
#include <complex.h>
#include <math.h>

Ipcmd2=Ipcmd*1.5/2;
VT_radius=VT_angle*2*pi/360;

f[0]=dIXinj/dt=(-1)/0.02*IXinj+1/0.02*Ipcmd2;
f[1]=dEq/dt=(-1)/0.02*Eq+1/0.02*Eqcmd;
IYinj=Eq*(-1/Xeq);

double complex Vterm_Complex=Eterm*cexp(VT_radius*I);   //not sure if this works for polar
//complex number representatiomn
double complex Iinj=IXinj+I*IYinj;
double complex Isorc=Iinj*cexp(I*Delta);
double complex S_com=Vterm_Complex*conj((Vterm_Complex*I/0.1+Isorc)*2*10/100)
Pelec=creal(S_com);
Qelec=cimag(S_com);
```

- Build Residual Function for each model in C code.

Take WT3G1 model for an example:

```
PetscErrorCode ResidualFunctionWT3G1(TS3phGenCtx* Gen,
{
```

Function name of WT3G1

Retrieve the State Variables from the array after Petsc evaluated it in last iteration

```
/*State Variables indices*/
Ip_idx = WT3G1->Ip_idx[idx_wt];
Eq_idx = WT3G1->Eq_idx[idx_wt];
Vpll1_idx = WT3G1->Vpll1_idx[idx_wt];
Vpll2_idx = WT3G1->Vpll2_idx[idx_wt];

/*State Variables*/
Ip = xgen_arr[Ip_idx];
Eq = xgen_arr[Eq_idx];
Vpll1 = xgen_arr[Vpll1_idx];
Vpll2 = xgen_arr[Vpll2_idx];
```

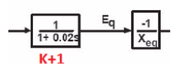
Initialization of Dynamic States

```
if(Gen->net_mode==1 && initdone == 0){
   // Pelec and Qelec must be initialized!
  Pelec = WT3G1->PG[idx_wt]; // Pgen in per unit on SBase
  Qelec = WT3G1->QG[idx_wt]; // Qgen in per unit on SBase

  Isorc_real=(cos(Vterm_angle)*Pelec+sin(Vterm_angle)*Qel
      Isorc_imag=(sin(Vterm_angle)*Pelec-cos(Vterm_angle)

      /*Isorc_real=Isorc_real*SBase/(MBase*NumWt)+Eterm*s
      Isorc_imag=Isorc_imag*SBase/(MBase*NumWt)-Eterm*cos

      Ip=Pelec/(MBonSB*NumWt*Eterm); // Ip in per unit on
      xgen_arr[Ip_idx] = Ip;

      Iyinj=-Qelec/(MBonSB*NumWt*Eterm)-Eterm/Xeq;
      Eq=-Iyinj*Xeq;
      xgen_arr[Eq_idx] = Eq;
```

```
/* Ip_cmd */
Ip_cmd = ugen_arr[2*i+1];

/*1. Residual Function for Ip derivative*/
  BlockInfo blockIp;

  blockIp.input       = &Ip_cmd;
  blockIp.state       = &xgen_arr[Ip_idx];
  blockIp.state_dot   = &xdot_arr[Ip_idx];
  blockIp.K           = PETSC_NULL;
  blockIp.Tden        = &TIp;
  blockIp.nmode       = &Gen->net_mode;

  fgen_arr[Ip_idx] = fvalBlockA(&blockIp);
```

Evaluate control blocks, fgen_arr stores the state variebles and give it to Petsc solver.

Code on the left is implementing the block below:




```
/* Pelec */
//  fgen_arr[WT3G1->PG_idx[idx_wt]]= (Isorc_real*cos(Vt
//  (-Isorc_imag)*sin(Vterm_angle)*Eterm)-Pelec;
WT3G1->PG[idx_wt] = Isorc_real*cos(Vterm_angle)*Eterm -
  (-Isorc_imag)*sin(Vterm_angle)*Eterm;

/* Qelec */
//  fgen_arr[WT3G1->QG_idx[idx_wt]]= (Isorc_real*sin(Vt
//  (-Isorc_imag)*cos(Vterm_angle)*Eterm)-Qelec;
WT3G1->QG[idx_wt] = Isorc_real*sin(Vterm_angle)*Eterm +
  (-Isorc_imag)*cos(Vterm_angle)*Eterm;
```

Finally it reaches to the end of the model, in this case, the output is Pelec and Qelec. Put the updated results in the particular arrays, done with one iteration.

- Connect to the TS3ph Simulator in ECE server

Build data structure for each model

```
typedef struct{
  PetscInt Nwt3g1;
  PetscInt nwt3g1;
  PetscInt * bus;
  char * ID;
  PetscInt    * NumWt;
  PetscScalar * Xeq;
  PetscScalar * Kpll;
  PetscScalar * Kipll;
  PetscScalar * Pllmax;
  PetscScalar * Prated;
  PetscScalar * PG;
  PetscScalar * QG;
  PetscScalar * MBase;

  PetscInt    * Ip_idx;
  PetscInt    * Eq_idx;
  PetscInt    * Vpll1_idx;
  PetscInt    * Vpll2_idx;

  PetscInt    *Vpll1_lflag;

  PetscInt    *iternum;
}WT3G1_MODEL;
```

Build the read dynamic data file

```
PetscErrorCode ReadDyrWT(char * line, char *line2){

  PetscErrorCode ierr;
  char data_format[512];

  PetscFunctionBegin;


  /* WT3G1 MODEL */
  if(strstr(line,"WT3G1")!=NULL){
    ierr = PetscMalloc(sizeof(WT3G1_node),&(WT3G1_cur));CHKERRQ(ierr);
    sprintf(data_format,"%%d %%*s %%%dc %%d %%lf %%lf %%lf %%lf %%lf",MAX_ID_LENGTH);
    sscanf(line,data_format,&WT3G1_cur->bus,(char*)&WT3G1_cur->ID,&WT3G1_cur->NumWt,&
rated);
    if(WT3G1_pre != NULL) WT3G1_pre->next=WT3G1_cur;
    WT3G1_cur->next=NULL;
    WT3G1_cur=WT3G1_cur;
    if(WT3G1_head == NULL) WT3G1_head=WT3G1_cur;
  }/*end if*/
```

```
1,'GENROU',1, 6.1000, 0.0500, 1.0000, 0.1500,  3.:

1,'ESDC1A',1, 0.0, 25.0, 0.2, 0.0, 0.0, 100.0, -1(

2 'WT3G1' 1    10
  0.10000 30.0000 0.00000 0.10000 1.5000 /

2 'WT3E1' 1    0 1 1 0 0 '1 '
  0.15000 18.000 5.0000 0.0000 0.50000E-01 3.0000
```

The code in the Read Dyr data file is able to read the input parameters by user (Dyr file on the left).

Put modified files in the compile sequence

```
 driver/TS3ph.o src/utils/ut
ontrol/blocks.o \
ontrol/jacobian.o \
ata_read/TS3phReadData.o \
ata_read/ReadPSSERaw.o \
ata_read/ReadPSSESeq.o \
ata_read/ReadPSSEDyr.o \
ata_read/ReadDyrGen.o \
ata_read/ReadDyrWT.o \
ata_read/ReadDyrExc.o \
ata_read/ReadDyrGov.o \
ata_read/ReadDyrStab.o \
ata_read/ReadDyrLoad.o \
ata_read/ReadDyrExclim.o \
ata_read/ReadDyrCom.o \
ata_read/StoreGENDATA.o \
ata_read/FreeDyrList.o \
```

```
function/ResidualFunctionEXST1.o \
function/ResidualFunctionEXST3.o \
function/ResidualFunctionCIMSBL.o \
function/ResidualFunctionWT3G1.o \
function/ResidualFunctionWT3E1.o \
acobian/TS3phJacobian.o \
```

Results

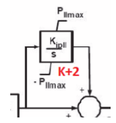
Residual function document for WT3G1, WT3E1, WT3T1, WT3P1 has been written for programming guideline and future research reference.

Residual function code for WT3G1, WT3E1, WT3T1, WT3P1 is almost ready for final testing.

Read Dyr file, data structure and other necessary files to run the simulation of WT3G1 and WT3E1 has been prepared for running partial model testing.

Non-windup limiter as shown on the left was difficult to integrated in the c code.

Original Matlab code as shown below:




```
if ((K2>=Pllmax && K2>0) || (V1>=Pllmax && K2>0))
    K2in=0;
else if (((K2<=-Pllmax)&& K2<0) || (V1<= -Pllmax && K2<0))
    K2in=0;
else
    K2in=Vyg;
```

We discovered a new way to implement none windup limiter in the C code.

```
/* 3. Residual function for Vpll1 derivative */

    /* Vpll1 non-windup limitor*/

    WT3G1->Vpll1_lflag[idx_wt] = 0;
    if(Vpll1>=Pllmax && Vpll1_dot>0){
            fgen_arr[Vpll1_idx] = Vpll1 - Pllmax;
            WT3G1->Vpll1_lflag[idx_wt] = 1;
    }
    else if(Vpll1<=-Pllmax && Vpll1_dot<0){
            fgen_arr[Vpll1_idx] = Vpll1 - (-Pllmax);
            WT3G1->Vpll1_lflag[idx_wt] = -1;
    }
    else{
            BlockInfo blockVpll1;

            blockVpll1.input       = &Vyk;
            blockVpll1.state       = &xgen_arr[Vpll1_idx];
            blockVpll1.state_dot   = &xdot_arr[Vpll1_idx];
            blockVpll1.Tden        = &Constant_one;
            blockVpll1.nmode       = &Gen->net_mode;

            fgen_arr[Vpll1_idx]    = fvalBlockD(&blockVpll1);
    }
```

Conclusions

- ✓ Residual Function document for type 3 wind turbine model has been written for future use. Providing resources for future researchers and guideline for programming.
- ✓ Residual Function for all for models are written in C code.
- ✓ Debugging and testing are needed for future development to make sure the simulation result matches PSSE.
- ✓ The project will add an important feature to the TS3ph simulator and thus contribute to the understanding of how Type 3 wind turbines affect grid stability.

[1] I. A. Hiskens, *Dynamics of Type-3 Wind Turbine Generator Models*, 2012. ▪ [2] EPRI, *Converter Model Grid Interface*. ▪ [3] EPRI, *WECC Type 3 Wind Turbine Generator Model – Phase II*, Knoxville, Tennessee, 2014. ▪ [4] EPRI, *Proposed Changes to the WECC WT4 Generic Model for Type 4 Wind Turbine Generators*, Knoxville, Tennessee, 2011. ▪ [5] *Typical Machine Data for Power System Simulation Modeling*, 2014. ▪ [6] National Renewable Energy Laboratory, *WECC WIND GENERATOR DEVELOPMENT*, 2010. ▪ [7] Q. Xu and Z. Xu, *A Survey on General Models of Wind Turbine Generators in PSS/E*, Hangzhou, Zhejiang, 2010. ▪ [8] Siemens Energy, Inc., *Memorandum*, 2011.