Synchronous Generator Model Based on LabVIEW Software

GENTIAN DUME
Department of Control
Polytechnic University of Tirana
“Mother Teresa” Square, No. 4
ALBANIA
gentian.dume@lme.upt.al http://www.fie.upt.al

Abstract: - On the power system stability analyses, the synchronous generator model is important. LabVIEW, a graphical programming language, mostly used in data acquisition, is becoming a powerful tool due to several modules associated to it. One of them is Control Design & Simulation Module. Even more applications created in LabVIEW can be deployed easily to real-time targets. This paper presents a procedure on mathematical modeling of a synchronous generator in the LabVIEW environment. Simulations of the same synchronous generator in Matlab/Simulink and LabVIEW verify the accuracy of the model in LabVIEW.

Key-Words: - Synchronous generator, LabVIEW, mathematical modeling, simulations, Matlab/Simulink

1 Introduction

Synchronous generator is the main part of the power system. It has a complex dynamic behavior. This behavior influences on the entire power system. Hence, to analyze different problems of the power system, one must build the mathematical model of the synchronous generator [1].

A lot of books include the theory and performance of the synchronous machines. Their modeling and analysis has always been not easy. In the 1920s and 1930s the problem was surveyed in details. In addition to, it has also been part of many recent investigations [2].

In the literature [3] several software solutions for simulations, control and scientific visualization have been reported. Some of them are Director, Authorware, Hypertext, Labtech, Visual C, Visual Basic, LabVIEW and Matlab/Simulink. We will use LabVIEW, as a powerful software for simulations and visualizations [4].

LabVIEW is released by National Instruments, for the first time in 1986 for Apple Macintosh. It was conceived as a programming environment for hardware control. The introduction of an interface between a PC and the instrument controlled by it was the main aim. This graphical interface simulates the controlled instrument on the computer monitor. But LabVIEW is also a graphical programming language, sometimes called “G code”. Applications created in LabVIEW are called “Virtual Instruments” [5]. Nowadays it is also compatible with Windows, Linux and other operating systems.

One of the LabVIEW ability is to get data from the real world, process them in the block diagram and send the results easily outside in the real world. This is very useful for engineers who want to interact and control events in the real world. Programs created with LabVIEW take less time to be completed than with other software, due to its graphical nature. It is becoming an industrial standard for creating new devices prototype [6].

2 Synchronous machine mathematical model

Synchronous machines are widely used as generators for power generation. In their modeling and parameters determination there is plenty of literature.

Referring to [7], it is assumed that the winding currents produce fields distributed in a sinusoidal form around the air gap, and that the stator slots cause no evident changes of any of the rotor winding inductances with rotor angle. In addition to, the saturation is not taken into account. For modeling, the couple-circuit approach is used.

Usually the synchronous machine has only one field winding. To represent the effect of the eddy currents and damper windings, often are added other windings. Simulations on power system through the years has shown that a large number of synchronous generators, can be represented by a model, that has besides the field winding one or two sets of damper windings.

Figure 1 shows a circuit representation of an idealized machine model, usually used in the analysis. Figure 2 shows the equivalent qd0 circuits of the idealized synchronous machine.

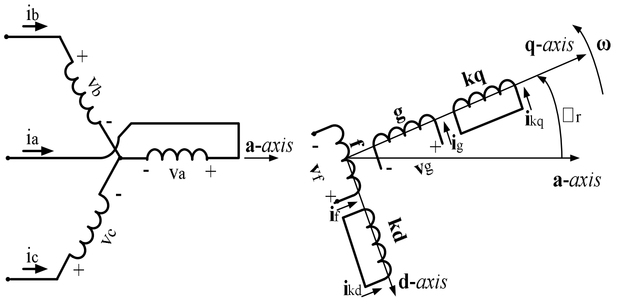


Fig. 1 Idealized synchronous machine circuit representation

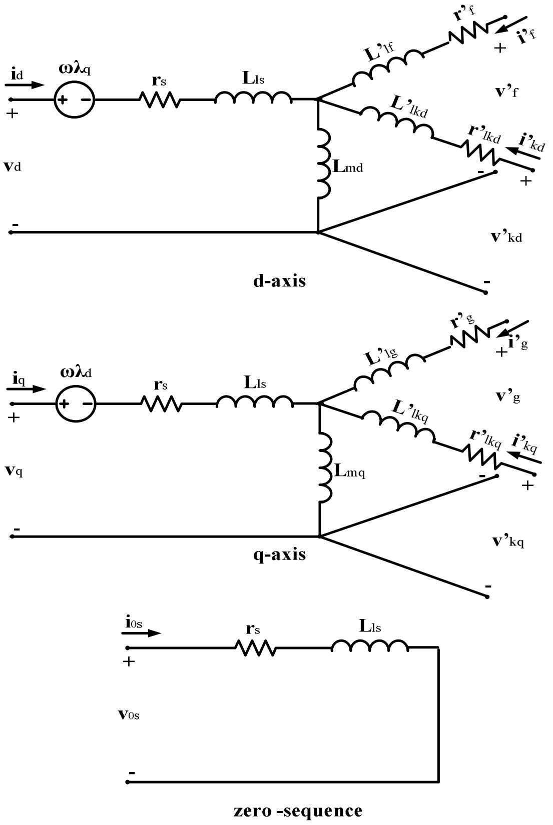


Fig. 2 Equivalent qd0 circuits of the synchronous machine

Rely on [7] for the equations of the synchronous generator mathematical model, since the LabVIEW model will be compared with a Simulink/Matlab model based on those equations.

The transformation from abc to qd rotor is:

$$v_q^s = \tfrac{2}{3}v_a - \tfrac{1}{3}v_b - \tfrac{1}{3}v_c \qquad \text{V}$$

$$v_d^s = \frac{1}{\sqrt{3}}(v_c - v_b) \qquad (1)$$

$$v_0 = \tfrac{1}{3}(v_a + v_b + v_c)$$

$$v_q = v_q^s \cos\theta_r(t) - v_d^s \sin\theta_r(t)$$

$$v_d = v_q^s \sin\theta_r(t) + v_d^s \cos\theta_r(t) \qquad (2)$$

where:

$$\theta_r(t) = \int_0^t \omega_r(t)dt + \theta_r(0) \qquad (3)$$

The equations of qd0 voltage can be written as integral equations of the windings flux linkages, which for a machine with one field winding in the d-axis and a pair of damper windings in the d- and q-axes (in motoring convention) are:

$$\psi_q = \omega_b \int \left\{ v_q - \frac{\omega_r}{\omega_b}\psi_d + \frac{r_s}{x_{ls}}\left(\psi_{mq} - \psi_q\right)\right\} dt$$

$$\psi_d = \omega_b \int \left\{ v_d - \frac{\omega_r}{\omega_b}\psi_q + \frac{r_s}{x_{ls}}\left(\psi_{md} - \psi_d\right)\right\} dt$$

$$\psi_0 = \omega_b \int \left(v_0 - \frac{r_s}{x_{ls}}\psi_0 \right) dt$$

$$\psi'_{kq} = \frac{\omega_b r'_{kq}}{x'_{lkq}} \int \left(\psi_{mq} - \psi'_{kq}\right) dt$$

$$\psi'_{kd} = \frac{\omega_b r'_{kd}}{x'_{lkd}} \int \left(\psi_{md} - \psi'_{kd}\right) dt \qquad (4)$$

$$\psi'_f = \frac{\omega_b r'_f}{x'_{md}} \int \left\{ E_f + \frac{x_{md}}{x'_{lf}}\left(\psi_{md} - \psi'_f\right)\right\} dt$$

where:

$$\psi_{mq} = \omega_b L_{mq}\left(i_q + i'_{kq}\right)$$

$$\psi_{md} = \omega_b L_{md}\left(i_d + i'_{kd} + i'_f\right) \qquad (5)$$

$$E_f = x_{md}\frac{v'_f}{r'_f}$$

$$\psi_q = x_{ls}i_q + \psi_{mq}$$

$$\psi_d = x_{ls}i_d + \psi_{md}$$

$$\psi_0 = x_{ls}i_0$$

$$\psi'_f = x'_{lf}i'_f + \psi_{md} \qquad (6)$$

$$\psi'_{kd} = x'_{lkd}i'_{kd} + \psi_{md}$$

$$\psi'_{kq} = x'_{lkq}i'_{kq} + \psi_{mq}$$

The mutual flux linkages in terms of the total flux linkages of the windings are:

$$\psi_{mq} = x_{MQ}\left(\frac{\psi_q}{x_{ls}} + \frac{\psi'_{kq}}{x'_{lkq}}\right)$$
$$\psi_{md} = x_{MD}\left(\frac{\psi_d}{x_{ls}} + \frac{\psi'_{kd}}{x'_{lkd}} + \frac{\psi'_f}{x'_{lf}}\right) \qquad (7)$$

where:

$$\frac{1}{x_{MQ}} = \frac{1}{x_{mq}} + \frac{1}{x_{ls}} + \frac{1}{x'_{lkq}}$$
$$\frac{1}{x_{MD}} = \frac{1}{x_{md}} + \frac{1}{x_{ls}} + \frac{1}{x'_{lkd}} + \frac{1}{x'_{lf}} \qquad (8)$$

So, finally the winding currents can be determined as:

$$i_q = \frac{\psi_q - \psi_{mq}}{x_{ls}}$$
$$i_d = \frac{\psi_d - \psi_{md}}{x_{ls}}$$
$$i'_{kd} = \frac{\psi'_{kd} - \psi_{md}}{x'_{lkd}} \qquad (9)$$
$$i'_{kq} = \frac{\psi'_{kq} - \psi_{mq}}{x'_{lkq}}$$
$$i'_f = \frac{\psi'_f - \psi_{md}}{x'_{lf}}$$

Using the rotor to stationary qd and stationary qd0 to abc transformations, the stator windings qd currents can be transformed back to abc winding currents, that is:

$$i_q^s = i_q \cos\theta_r(t) + i_d \sin\theta_r(t)$$
$$i_d^s = -i_q \sin\theta_r(t) + i_d \cos\theta_r(t) \qquad (10)$$

$$i_a = i_q^s + i_0$$
$$i_b = -\frac{1}{2} i_q^s - \frac{1}{\sqrt{3}} i_d^s + i_0 \qquad (11)$$
$$i_c = -\frac{1}{2} i_q^s + \frac{1}{\sqrt{3}} i_d^s + i_0$$

A synchronous machine with P-poles develops an electromechanical torque equal to:

$$T_{em} = \frac{3}{2}\frac{P}{2\omega_b}\left(\psi_d i_q - \psi_q i_d\right) \qquad (12)$$

The net acceleration torque (in motoring convention) $T_{em} + T_{mech} - T_{damp}$, is in the direction of the rotor's rotation.

Equating the inertial torque to the net acceleration torque, we have:

$$J\frac{d\omega_{rm}(t)}{dt} = \frac{2J}{P}\frac{d\omega_r(t)}{dt} = T_{em} + T_{mech} - T_{damp} \qquad (13)$$

The angle of the q_r-axis of the rotor with respect to the q_e-axis of the synchronously rotating reference frame is defined as the rotor angle, that is:

$$\delta(t) = \theta_r(t) - \theta_e(t)$$
$$= \int_0^t \{\omega_r(t) - \omega_e(t)\}dt + \theta_r(t) - \theta_e(t) \qquad (14)$$

but $\omega_e(t)$ is constant, so:

$$\frac{d\{\omega_r(t) - \omega_e(t)\}}{dt} = \frac{d\omega_r(t)}{dt} \qquad (15)$$

Replacing $d\omega_r(t)/dt$ from Eq. 15 to Eq. 13, and integrating:

$$\omega_r(t) - \omega_e(t) = \frac{P}{2J}\int_0^t (T_{em} + T_{mech} - T_{damp})dt \qquad (16)$$

Choosing:

S_b - the base power (rated kVA of the machine)
V_b - the peak value of the phase voltage
I_b - the peak value of the phase current

we have:

- base impedance: $z_b = \dfrac{V_b}{I_b}$
- base torque: $T_b = \dfrac{S_b}{\omega_{bm}}$ (17)

where:

ω_{bm} - the base mechanical angular frequency

The electromagnetic torque in per unit is:

$$T_{em(pu)} = \frac{T_{em}}{T_b} = \frac{\frac{3}{2}\frac{P}{2\omega_b}\left(\psi_d i_q - \psi_q i_d\right)}{\frac{3}{2}\left(\frac{\frac{V_b I_b}{2}}{\frac{2}{P}\omega_b}\right)} \qquad (18)$$

Reducing the expression (18) in:

$$T_{em(pu)} = \psi_{d(pu)} i_{q(pu)} - \psi_{q(pu)} i_{d(pu)} \quad (19)$$

Because the base V_b of the voltage is equal as the base for the flux linkages ψ_q and ψ_d.

Expressed in per unit, Eq. 13, becomes:

$$T_{em(pu)} + T_{mech(pu)} - T_{damp(pu)} = \left(\frac{2J}{T_b P}\right)\frac{d\omega_r}{dt} \quad (20)$$

Using the inertia constant H, we have:

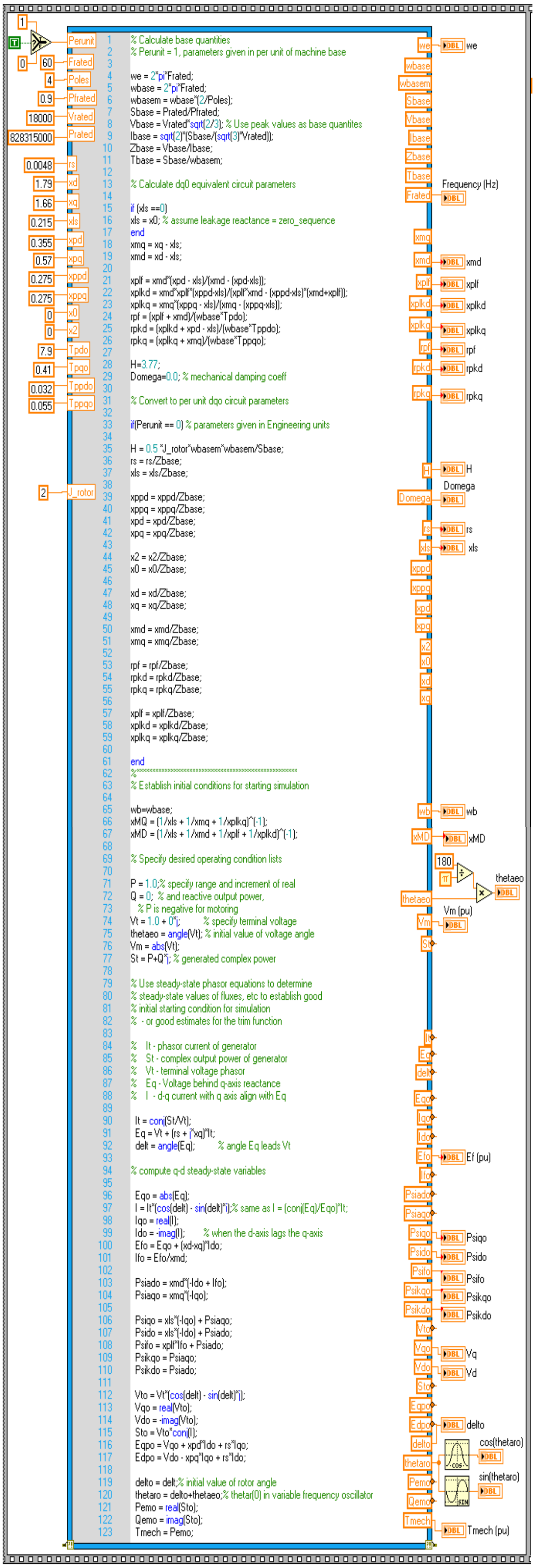
$$T_{em(pu)} + T_{mech(pu)} - T_{damp(pu)} = 2H\frac{d\left(\omega_r/\omega_b\right)}{dt}$$
$$= 2H\frac{d\left\{\left(\omega_r - \omega_e\right)/\omega_b\right\}}{dt} \quad (21)$$

3 Building the model of synchronous generator in LabVIEW

In this section it is shown how to build the model of synchronous generator (SG) in the LabVIEW environment. To do this, we first need to have installed the LabVIEW base package and Control Design & Simulation module (CDSM). The CDSM simplifies the process of building the model, since it has built-in ODE Solvers for solving integral equations. LabVIEW is a programming language, thus the model can be created without the help of CDSM. But if we want to add in the model some other elements in the future, it will be very difficult, because of the graphical nature of the code in LabVIEW.

The code in LabVIEW is written on the Block Diagram (BD) and the inputs (controls), and the outputs (indicators) are placed on the graphic user interface (GUI), called Front Panel (FP). They communicate with the block diagram through terminals [8]. Since LabVIEW is based on the dataflow, to give priority to a piece of code, one way is to use the ***Flat sequence structure***. We can use two sequential frames, one for initialization and for the initial conditions of the simulation (usually the steady-state values), and the second frame for the calculation of variable values.

Figure 3 shows the left frame of the ***flat sequence structure*** (LabVIEW reads the flat sequence from left to the right). Inside, we have placed a ***MathScript node***. In the left border are added the inputs, and in the right border the outputs. Inside the node we can place any text-based scripts (for example a m-file imported from Matlab).



```
% Calculate base quantities
% Perunit = 1, parameters given in per unit of machine base

we = 2*pi*Frated;
wbase = 2*pi*Frated;
wbasem = wbase*(2/Poles);
Sbase = Prated/Pfrated;
Vbase = Vrated*sqrt(2/3); % Use peak values as base quantites
Ibase = sqrt(2)*(Sbase/(sqrt(3)*Vrated));
Zbase = Vbase/Ibase;
Tbase = Sbase/wbasem;

% Calculate dq0 equivalent circuit parameters

if (xls ==0)
xls = x0; % assume leakage reactance = zero_sequence
end
xmq = xq - xls;
xmd = xd - xls;

xplf = xmd*(xpd - xls)/(xmd - (xpd-xls));
xplkd = xmd*xplf*(xppd-xls)/(xplf*xmd - (xppd-xls)*(xmd+xplf));
xplkq = xmq*(xppq - xls)/(xmq - (xppq-xls));
rpf = (xplf + xmd)/(wbase*Tpdo);
rpkd = (xplkd + xpd - xls)/(wbase*Tppdo);
rpkq = (xplkq + xmq)/(wbase*Tppqo);

H=3.77;
Domega=0.0; % mechanical damping coeff

% Convert to per unit dqo circuit parameters

if(Perunit == 0) % parameters given in Engineering units

H = 0.5 *J_rotor*wbasem*wbasem/Sbase;
rs = rs/Zbase;
xls = xls/Zbase;

xppd = xppd/Zbase;
xppq = xppq/Zbase;
xpd = xpd/Zbase;
xpq = xpq/Zbase;

x2 = x2/Zbase;
x0 = x0/Zbase;

xd = xd/Zbase;
xq = xq/Zbase;

xmd = xmd/Zbase;
xmq = xmq/Zbase;

rpf = rpf/Zbase;
rpkd = rpkd/Zbase;
rpkq = rpkq/Zbase;

xplf = xplf/Zbase;
xplkd = xplkd/Zbase;
xplkq = xplkq/Zbase;

end
%**************************************************
% Establish initial conditions for starting simulation

wb=wbase;
xMQ = (1/xls + 1/xmq + 1/xplkq)^(-1);
xMD = (1/xls + 1/xmd + 1/xplf + 1/xplkd)^(-1);

% Specify desired operating condition lists

P = 1.0;% specify range and increment of real
Q = 0;  % and reactive output power,
  % P is negative for motoring
Vt = 1.0 + 0*j;         % specify terminal voltage
thetaeo = angle(Vt); % initial value of voltage angle
Vm = abs(Vt);
St = P+Q*j; % generated complex power

% Use steady-state phasor equations to determine
% steady-state values of fluxes, etc to establish good
% initial starting condition for simulation
%  - or good estimates for the trim function

%    It - phasor current of generator
%    St - complex output power of generator
%    Vt - terminal voltage phasor
%    Eq - Voltage behind q-axis reactance
%    I  - d-q current with q axis align with Eq

 It = conj(St/Vt);
 Eq = Vt + (rs + j*xq)*It;
 delt = angle(Eq);         % angle Eq leads Vt

% compute q-d steady-state variables

 Eqo = abs(Eq);
 I = It*(cos(delt) - sin(delt)*i);% same as I = (conj(Eq)/Eqo)*It;
 Iqo = real(I);
 Ido = -imag(I);        % when the d-axis lags the q-axis
 Efo = Eqo + (xd-xq)*Ido;
 Ifo = Efo/xmd;

 Psiado = xmd*(-Ido + Ifo);
 Psiaqo = xmq*(-Iqo);

 Psiqo = xls*(-Iqo) + Psiaqo;
 Psido = xls*(-Ido) + Psiado;
 Psifo = xplf*Ifo + Psiado;
 Psikqo = Psiaqo;
 Psikdo = Psiado;

 Vto = Vt*(cos(delt) - sin(delt)*j);
 Vqo = real(Vto);
 Vdo = -imag(Vto);
 Sto = Vto*conj(I);
 Eqpo = Vqo + xpd*Ido + rs*Iqo;
 Edpo = Vdo - xpq*Iqo + rs*Ido;

 delto = delt;% initial value of rotor angle
 thetaro = delto+thetaeo;% thetar(0) in variable frequency oscillator
 Pemo = real(Sto);
 Qemo = imag(Sto);
 Tmech = Pemo;
```

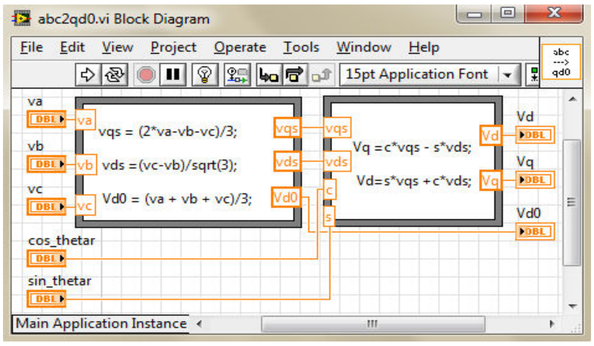
Fig.3 Calculation of initial values for simulation

Table 1 in the appendix shows the ratings and the parameters of the synchronous generator for simulation.

In this frame is placed the same script we will use to compare LabVIEW model with a similar verified model [10] of the synchronous generator, built in Matlab/Simulink [7].

Using ***Select function*** we can calculate for per unit (1) or engineering units (0). Outputs are connected through wires to numeric indicators. Some of the outputs are processed to give the desired value to the respective indicators (for example thetae0 from radians to degrees). It can be seen that some outputs aren't connected to any indicator. This is because they are not used in the simulations we will perform later. But they can be used in future simulations, depending on our target.

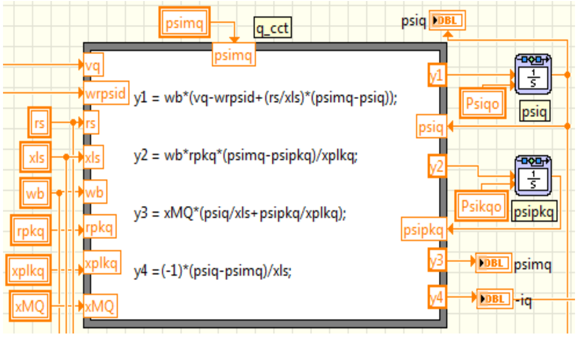
Let's pass now to the right frame. LabVIEW will process this frame after the left one. We will place a ***Control & Simulation loop.*** This loop isn't available in the LabVIEW base package. So, we must have installed CDSM. The dynamic model of the generator was built based on the equations above. Figures 4 to 10 show all the blocks in detail. Since LabVIEW is modular, some subVIs have been created, but not for all blocks, because they have dataflow on integrator nodes, and in CDSM is not allowed to have a ***Control & Simulation loop*** inside another same loop. So, we must place those blocks on the top level application. Figure 11 shows the overall diagram of the synchronous generator model.



```
vqs = (2*va-vb-vc)/3;
vds =(vc-vb)/sqrt(3);
Vd0 = (va + vb + vc)/3;
```

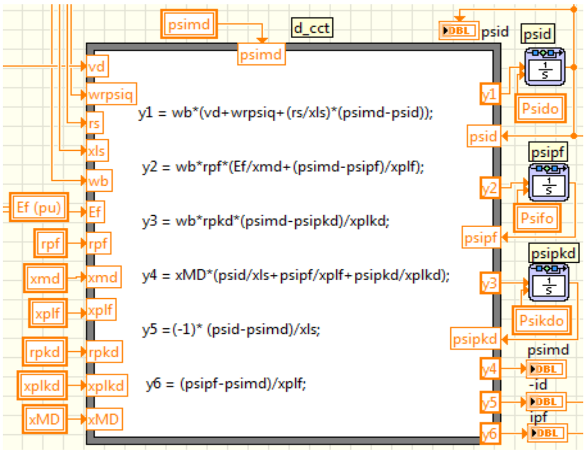
```
Vq =c*vqs - s*vds;
Vd=s*vqs +c*vds;
```

Fig. 4 abc2qd0 transformation subVI



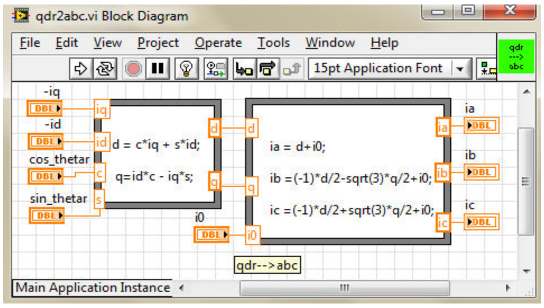
```
y1 = wb*(vq-wrpsid+(rs/xls)*(psimq-psiq));
y2 = wb*rpkq*(psimq-psipkq)/xplkq;
y3 = xMQ*(psiq/xls+psipkq/xplkq);
y4 =(-1)*(psiq-psimq)/xls;
```

Fig. 5 q-axis calculations



```
y1 = wb*(vd+wrpsiq+(rs/xls)*(psimd-psid));
y2 = wb*rpf*(Ef/xmd+(psimd-psipf)/xplf);
y3 = wb*rpkd*(psimd-psipkd)/xplkd;
y4 = xMD*(psid/xls+psipf/xplf+psipkd/xplkd);
y5 =(-1)* (psid-psimd)/xls;
y6 = (psipf-psimd)/xplf;
```

Fig. 6 d-axis calculations



```
d = c*iq + s*id;
q=id*c - iq*s;
```

```
ia = d+i0;
ib =(-1)*d/2-sqrt(3)*q/2+i0;
ic =(-1)*d/2+sqrt(3)*q/2+i0;
```

Fig. 7 qdr2abc transformation subVI

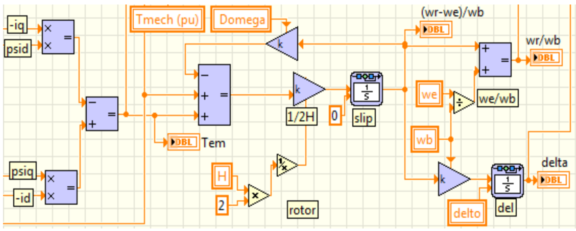


Fig. 8 Rotor block

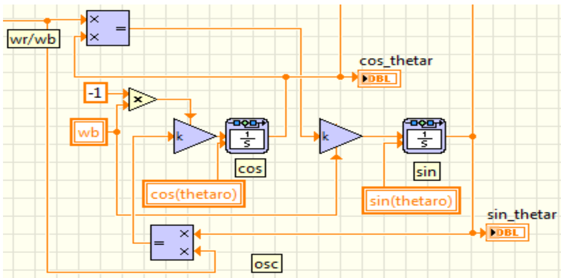
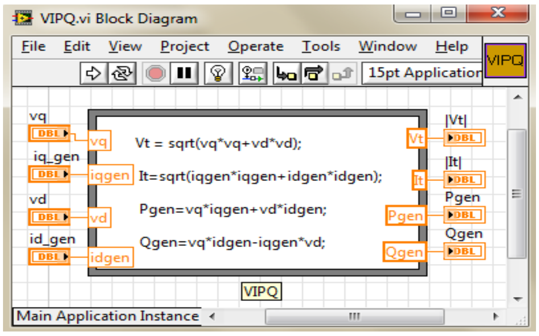
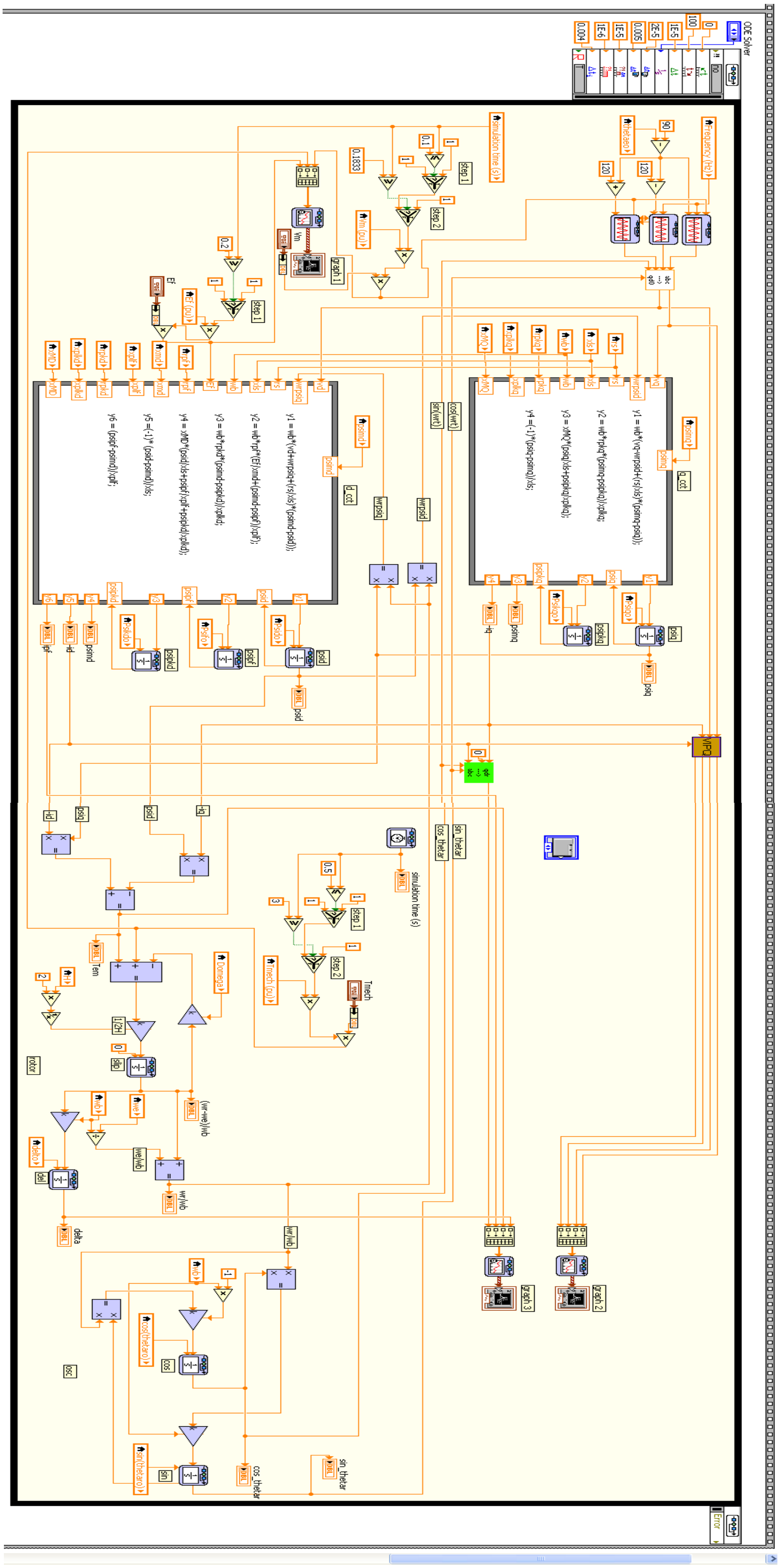


Fig. 9 Oscillator block



```
Vt = sqrt(vq*vq+vd*vd);
It=sqrt(iqgen*iqgen+idgen*idgen);
Pgen=vq*iqgen+vd*idgen;
Qgen=vq*idgen-iqgen*vd;
```

Fig. 10 VIPQ generation subVI



```
y1 = wb*(vd+wrpsiq+(rs/xls)*(psimd-psid));
y2 = wb*rpf*(Ef/xmd+(psimd-psipf)/xplf);
y3 = wb*rpkd*(psimd-psipkd)/xplkd;
y4 = xMD*(psid/xls+psipf/xplf+psipkd/xplkd);
y5 =(-1)*(psid-psimd)/xls;
y6 = (psipf-psimd)/xplf;
```

```
y1 = wb*(vq-wrpsid+(rs/xls)*(psimq-psiq));
y2 = wb*rpkq*(psimq-psipkq)/xplkq;
y3 = xMQ*(psiq/xls+psipkq/xplkq);
y4 =(-1)*(psiq-psimq)/xls;
```

Fig. 11 Overall diagram of synchronous generator in LabVIEW

For calculation of variables in the *q*- and *d*-axes ***formula node*** is used. Also same node is used for transformation abc2qd0, dqr2abc and VIPQ subVIs. To pass data from the left frame to the simulation frame ***local variables*** are used for every indicator on the initialization frame. In spite of, local variables are used for *psimq* and *psimd* variables, since in LabVIEW a node executes only if all its input terminals have data available [9] (psimq and psimd are outputs).

4 Verification of the SG model built in LabVIEW

In this section a comparison of the proposed model with a similar Matlab/Simulink model is done. Figure 12 shows the model of the synchronous generator [7]. This model is verified in [10]. So, we will use the same parameters, initial conditions and disturbances for both models.

As it can be seen in figure 12, the inputs of the model are the stator abc voltages, the excitation voltage E_f and the mechanical torque T_{mech}. The outputs are: P_{gen}- the real power generated, Q_{gen}- the reactive power generated, $|I_t|$ - the stator current magnitude, $|V_t|$- the stator voltage magnitude, *delta* - the power angle, T_{em} - the instantaneous electrical torque, i_f - the field current, i_a - the instantaneous phase-*a* current.

The same inputs and outputs are chosen in the model created in LabVIEW environment. The outputs are displayed on the front panel in two ***waveform charts***. To display all outputs on the screen we have used a ***Tab control***.

In Matlab/Simulink [7] it is recommended to use Dormand-Prince 45 as ode solver, and a variable step size. Thus, to have similar responses from both simulations, we have chosen Runge-Kutta 45 (variable step), as an ode solver in the LabVIEW model, because this solver uses Dormand-Prince coefficients.

Let's us now pass to the verification of the model we built in LabVIEW.

First, we will hold constant both magnitudes of the bus voltage and the mechanical torque, while excitation voltage E_f takes a step increment of 10% at t = 0.2 sec. Figure 13 shows the piece of code added to the block diagram to simulate this change. Figure 14 shows the simulations of synchronous generator in Matlab/Simulink and LabVIEW for this change.

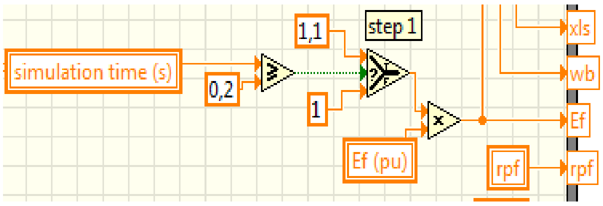


Fig. 13 10% increment of E_f at t = 0.2 sec

As it can be seen, the responses for every output are identical, thus the accuracy of the synchronous generator model built in LabVIEW for changes in the excitation voltage is verified.

Second, we will hold constant both magnitudes of the bus voltage and excitation voltage, while mechanical torque T_{mech} takes a step decrement from 1 pu to 0 pu (machine is generating rated power at unity power factor into the bus) at t = 0.5 sec., and then takes a step increment from 0 pu to -1 pu at t = 3 sec.

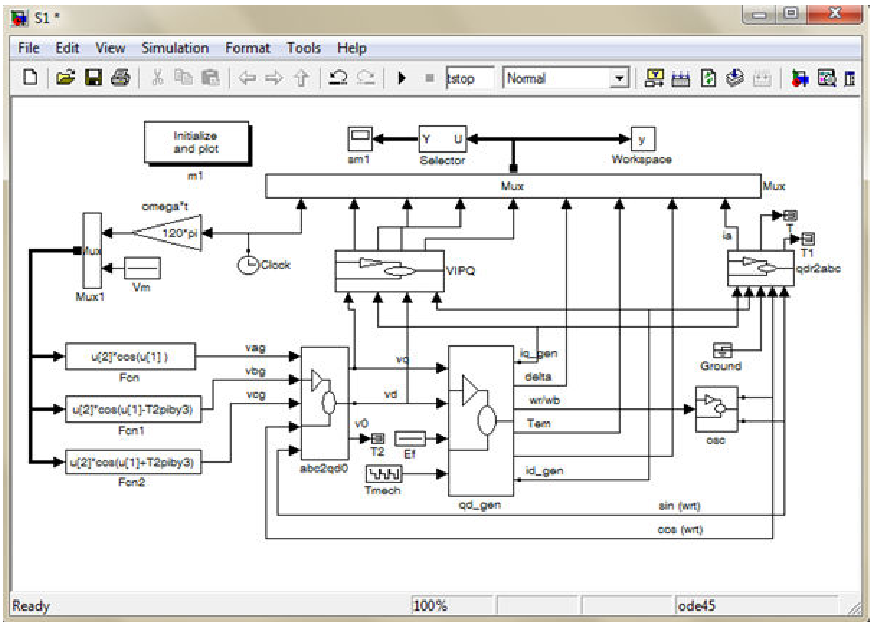


Fig. 12 Overall diagram of synchronous generator in Matlab/Simulink; source [7]

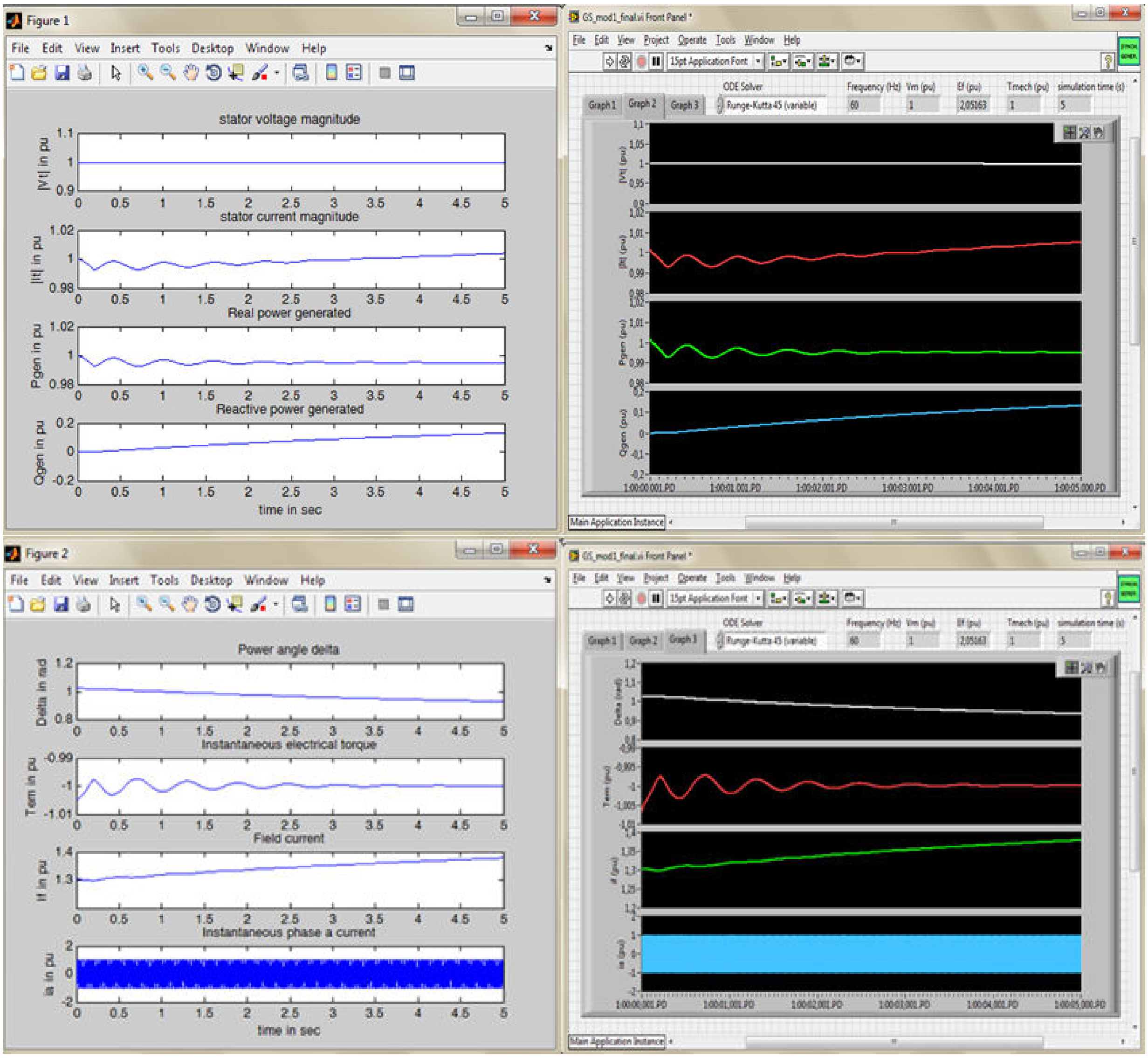


Fig. 14 Comparison of the synchronous generator Matlab/Simulink model with LabVIEW model for a 10% increment of E_f at t = 0.2 sec.

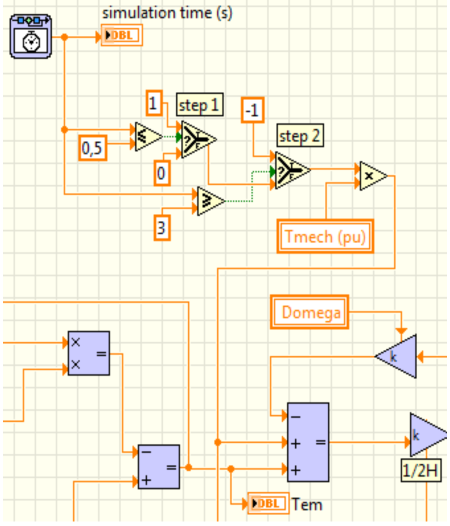


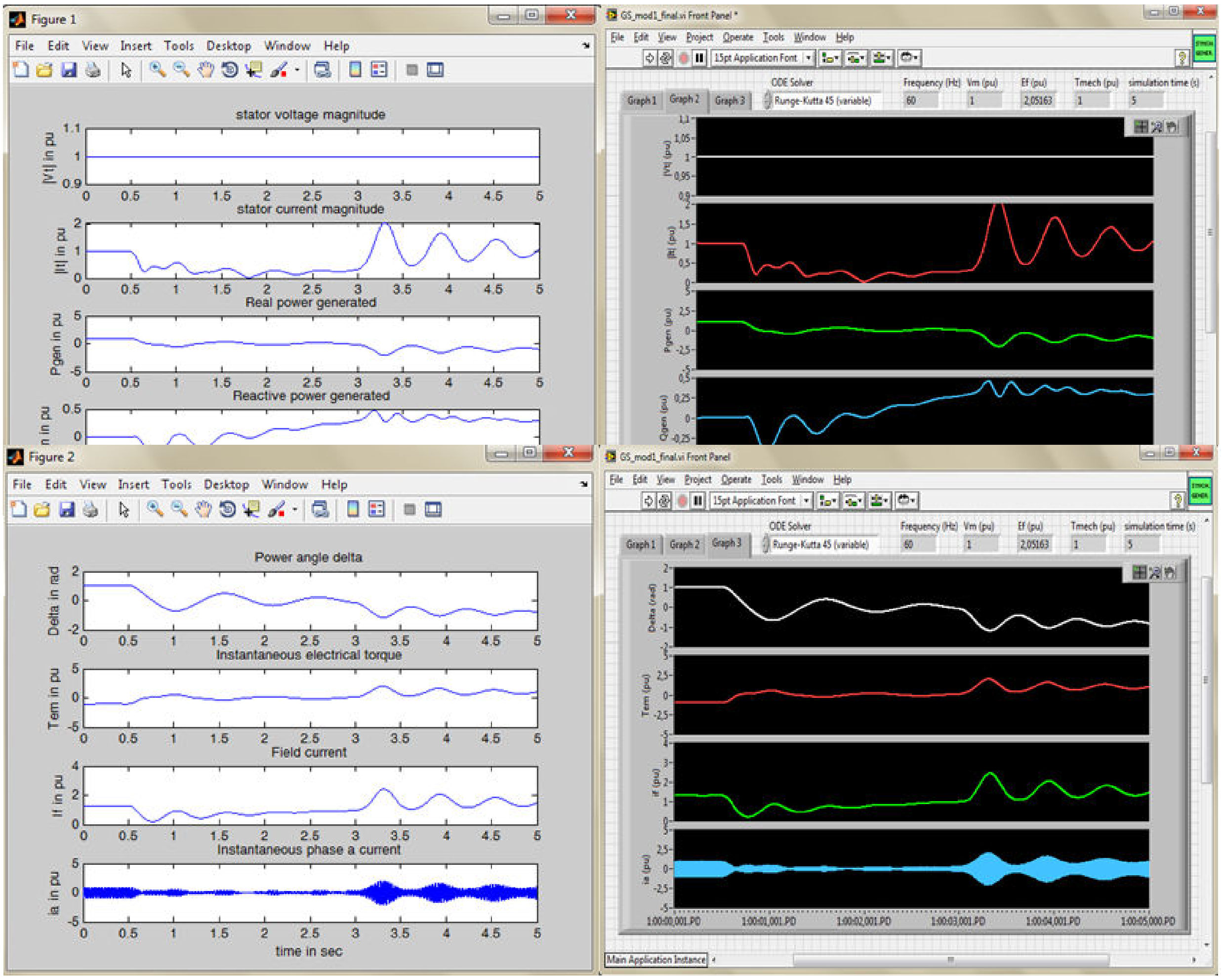
Fig. 15 Decrement of T_{mech} to 0 *at t* = 0.5 sec., and then to the negative of its starting value at *t* = 3 sec.

Figure 15 shows the piece of code added to the block diagram to simulate this change. Figure 16 shows the simulations of synchronous generator in Matlab/Simulink and LabVIEW for this change.

As it can be seen, the responses for every output are identical, thus the accuracy of the synchronous generator model built in LabVIEW for changes in the mechanical torque, is also verified.

Last, we will hold constant both magnitudes of the externally applied mechanical torque and excitation voltage, while the bus voltage V_m from 1 pu to 0 pu (machine is generating rated power at unity power factor into the bus) at t = 0.1 sec. (three phase terminal short circuit fault), and then after 5 cycles into the initial value:

$$t = t_{fstart} + t_{fault} = t_{fstart} + \frac{n_{cycles}}{F_{rated}} = 0.1833\; sec.$$



ODE Solver: Runge-Kutta 45 (variable)
Frequency (Hz): 60
Vm (pu): 1
Ef (pu): 2,05163
Tmech (pu): 1
simulation time (s): 5

Fig. 16 Comparison of the synchronous generator Matlab/Simulink model with LabVIEW model for a 100% decrement of T_{mech} at $t = 0.5$ sec., and then to the negative of its starting value at $t = 3$ sec.

Figure 17 shows the piece of code added to the block diagram to simulate this change. Figure 18 shows the simulations of synchronous generator in Matlab/Simulink and LabVIEW for this fault.

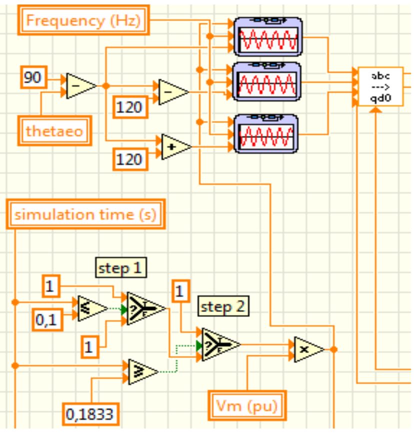
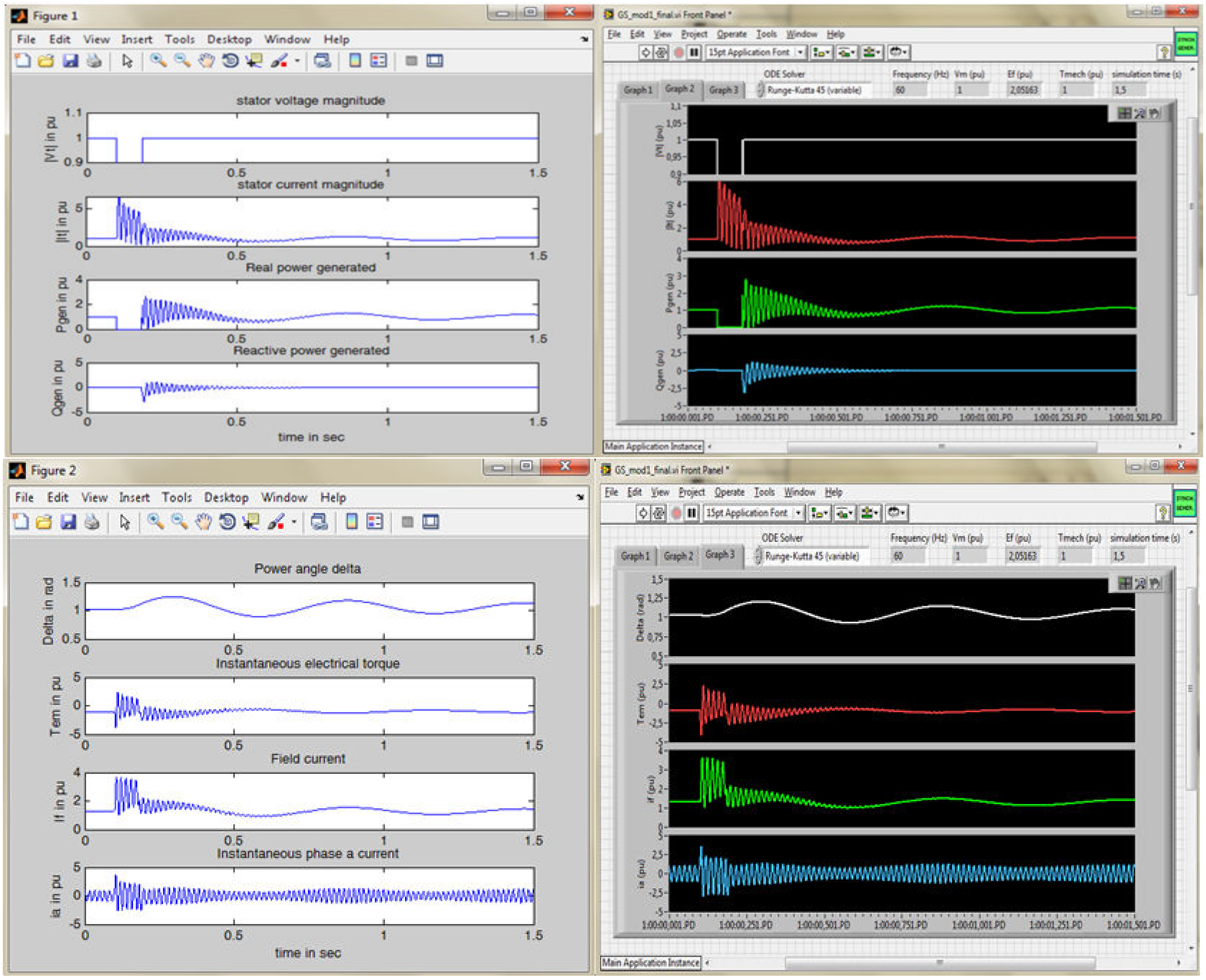


Fig. 17 100% decrement of V_m at t = 0.1 sec., and then to its starting value at t = 0.1833 sec.

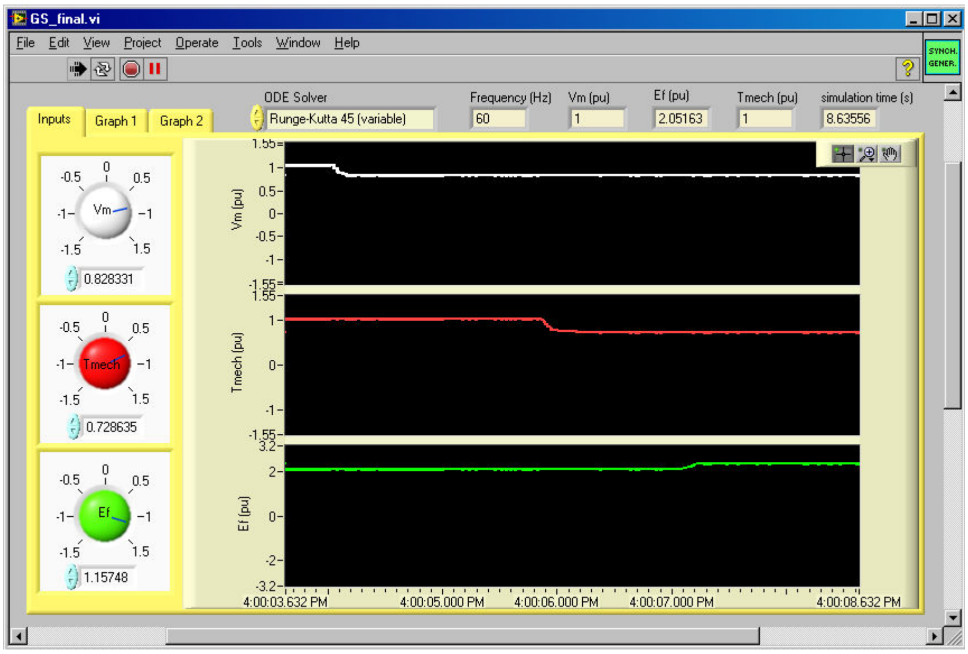
As it can be seen, also in these simulations, the responses for every output are identical, thus the accuracy of the synchronous generator model built in LabVIEW for changes in the stator abc voltages, is also verified.

Now that the model in LabVIEW is verified, we can improve the user interface layout. The way how the objects are arranged on the front panel helps the users who work with the application. Layout can help also the developers who access the subVI panels [11]. Figure 19 shows the final version of the synchronous generator model created in LabVIEW. It has three knob controls to simulate the changes in each input. It is possible to change the value of the inputs when the simulation is running. So, if we set a simulation time, let say 120 sec., we can change the values of the knobs, thus for example we can simulate a fault circuit in every moment, and LabVIEW will calculate all the outputs for this situation. It is also possible to force any knob control to change its value according to any function we want to simulate for. In addition to, a subVI has been created. Therefore, this model can be inserted in another application in LabVIEW, for future studies.



ODE Solver: Runge-Kutta 45 (variable)
Frequency (Hz): 60
Vm (pu): 1
Ef (pu): 2.05163
Tmech (pu): 1
simulation time (s): 1.5

Fig. 18 Comparison of the synchronous generator Matlab/Simulink model with LabVIEW model for V_m from its starting value to zero at $t = 0.1$ sec. and then in to the initial value at $t = 0.1833$ sec.



ODE Solver: Runge-Kutta 45 (variable)
Frequency (Hz): 60
Vm (pu): 1
Ef (pu): 2.05163
Tmech (pu): 1
simulation time (s): 8.63556

Vm: 0.828331
Tmech: 0.728635
Ef: 1.15748

Fig. 19 Front panel of the proposed synchronous generator model in LabVIEW

5 Conclusion

Modeling the synchronous generator in LabVIEW, gives the possibility, to use the capabilities of this powerful graphical programming language, in studying various problems, related to this component of the power system.

Since LabVIEW interfaces easily to a large number of hardware by multifunctional data acquisition cards, synchronous generator model can be used to test real controllers to specific synchronous generators, without disconnecting them from the network.

Synchronous generator model in LabVIEW can be used to test also digital controllers, and consequently controlling online real synchronous generator.

In the education point of view, students at the Electrical Engineering Faculty of Tirana who deal with this model, quickly familiarize, because of the user friendly GUI of LabVIEW (front panel) and the capabilities of this software to change the input values during simulation time.

In future works, other power system components will be modeled in the LabVIEW environment, so as to create the *power system module* in LabVIEW, in order to use all the facilities that this software gives to engineers and researchers.

6 Acknowledgement

The author would like to thank Professor Gëzim Karapici for his support and very useful advice.

References:

[1] Zh. Li, Y. Yang, X. Bao, Simulation and Analysis of the Third-order Model of Synchronous Generator Based on MFC, *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation*, August 9 - 12, Changchun, China, pp 4252

[2] P. Kundur, *Power System Stability and Control*, McGraw-Hill Inc., 1993

[3] Z. J. Pudlowski, *Computers in Electrical Engineering Education-Research, Development and Application*, Monash Engineering Education series, Monash University, Melbourne, Australia, 1995.

[4] G. Dume, Th. Koblara, G. Karapici, Advantages of integrating LabVIEW software in education, *National Conference – 60 Years PUT,* Polytechnic University of Tirana, 2011, API-A028.

[5] R. De Asmundis, *LabVIEW - Modeling, Programming and Simulations,* InTech, 2011.

[6] R.W. Larsen, *LabVIEW for Engineers,* Pearson Educations Inc., 2011.

[7] C. Ong, *Dynamic Simulation of Electric Machinery using Matlab/Simulink,* Prentice Hall, 1998.

[8] R. Bitter, T. Mohiuddin, M. Nawrocki, *LabVIEW Advanced programming techniques,* Taylor & Francis Group LLC., 2007.

[9] J. Travis, J. Kring, *LabVIEW for Everyone*, Prentice Hall, 2006.

[10] R. Sadikovic, *Single-machine infinite bus system*, Internal report, Zurich, 2003.

[11] P. A. Blume, *The LabVIEW style book,* Pearson Education Inc., 2007

Appendix

Table 1 Synchronous generator ratings and parameters; source [7]

F_{rated} = 60 Hz	$Poles$ = 4
P_{rated} = 828315 kW	V_{rated} = 18 kV
Pf_{rated} = 0.9	
x_{ls} = 0.215 pu	r_s = 0.0048 pu
x_d = 1.790 pu	x_q = 1.660 pu
x'_d = 0.355 pu	x'_q = 0.570 pu
x''_d = 0.275 pu	x''_q = 0.275 pu
T'_{do} = 7.9 s	T'_{qo} = 0.410 s
T''_{do} = 0.032 s	T''_{qo} = 0.055 s
H = 3.77 s	$D\omega$ = 0 pu