Wide Area Analysis and

Application in Power System

By

Zhongyu Wu

A dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master Degree

in

Electrical Engineering

Approved by:

Yilu Liu (Co-chair)

Jaime De La Ree (Co-chair)

Richard W. Conners

Dec 7, 2009

Blacksburg, Virginia U.S.A.

Keywords: frequency monitoring network (FNET), dynamic link library, event trigger, triangulation, power system small-signal stability, oscillatory instability, damping control, Power system stabilizer(PSS), evolutionary strategy

Wide Area Analysis and

Application in Power System

Zhongyu Wu

ABSTRACT

Frequency monitoring network (FNET) is an Internet based GPS synchronized wide-area frequency monitoring network deployed at distribution level. It is a very important tool to realize wide area analysis in power system. At first part of this thesis, FNET structure and characteristics are introduced.

After analysis and smoothing FDR signals, thesis presents the algorithm of event trigger and triangulation algorithm estimation of disturbance. Event trigger is based on the DF/DT between points of FDR. And the threshold of DF/DT is also presented. And illustrate the multiply method to calculate event time, which is important when deal with pre-disturbance frequency in time delay of arriving or TDOA part. Estimation of disturbance location method is discussed based on the TDOA in the second part of this work and prove that Time Delay of Arrival is a good algorithm for estimation event location proved by real cases.

Next this thesis shows the detailed programming of these two DLL applications. The purpose of developing DLL modules is for convenience of use. The code of event trigger application as well as its interface is presented. And the interface of DLL module and the key word to import and export DLL variables and function is described. It also presents the work of triangulation module in Visual C++ and its main functions. There are two demonstrations of applying DLL applications. The demonstrations shown how to load and use DLL, as well as the requirements of applications.

In the last part, PSS compensation optimization with a set of nonlinear differential algebraic equations (DAE) is introduced in detail. With the optimization theory, the

optimal control of small-signal stability of power electric systems is solved. The main contents of this thesis include:

（1） Models of power systems and test power electric systems. The dynamic and static models of the elements of power systems, such as generatorbbs, AVRs, PSSs and loads are presented. Method of power system linearization modeling is introduced. Use two system, WSCC 9-bus system, and New England 39-bus system, to test the method of analysis.

（2） The optimization models, voltage control-damping control, are presented. Pareto combined with evolutionary strategy (ES) are used to solve optimizations. Based on traditional PSS parameters optimizations, it can be formulated as a two steps problem, in which, two objective functions should be taken into account. The minimum damping torque should be identified.

Keywords: frequency monitoring network (FNET), dynamic link library, event trigger, triangulation, power system small-signal stability, oscillatory instability, damping control, Power system stabilizer(PSS), evolutionary strategy

Acknowledgement

First of all, I would like to give my sincere appreciation to Prof. Yilu Liu, my advisor. Without her consistent encouragement and strong support I cannot finish this thesis. I started my research activities at Virginia Tech in Prof. Liu's group. She used her selfless care to help me overcome the homesick during the early days when I just entered a new country and a new college. How lucky I can enter Prof. Liu's power IT lab! Here she lavishly shares her knowledge and creativity to me. I really learned a lot from her both in the since and life.

I want to thank Dr. Jaime De La Ree, who is very kind to be the co-chair of my committee. When I have questions, he always helps me a lot to solve it. I also thank Dr. Richard W. Conners for his important suggestions during my research and serving as my committee member. The research work would not have been finished without their guidance.

I would like to express my gratitude to all the professors, friends and students in Power IT Lab and dept. of ECE at Virginia Tech, including Tao Xia, Lang Chen, Yingcheng Zhang, Penn Markham and Yanzhu Ye, for their academic inspiration, support and friendship. I am glad to have so many good colleagues and friends; you are the treasure of my life. All of you make Blacksburg becomes more wonderful and warm.

I would like to express gratitude to my husband, Yaodong Yang, for his love, support and companionship during the stressful time pursuing my degree.

To Yaodong Yang

and our Parents

Table of Content

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Introduce of Frequency Monitoring Network (FNET)

Usually, power system operating conditions is classified into five states: normal, alert, emergency, in extremis and restorative. Figure 1.1 describes the transitions between these states.

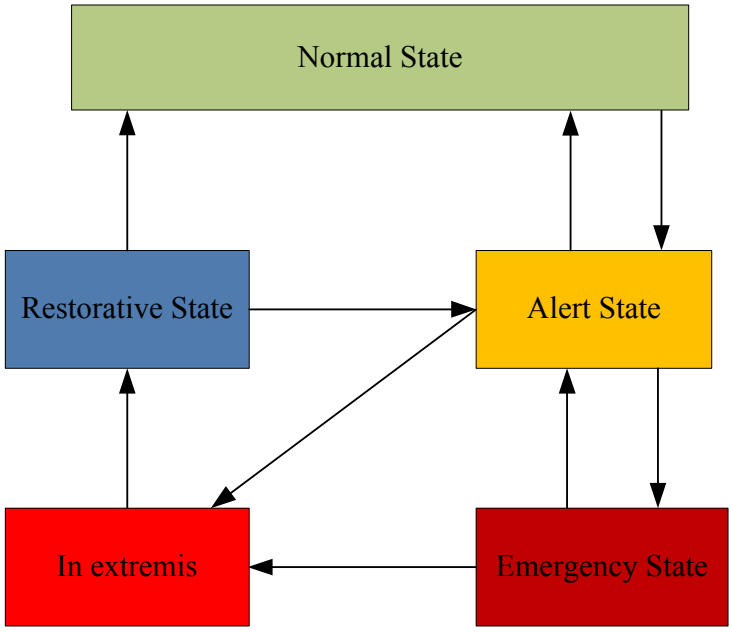

Figure 1.1 Power system operation states

To monitor system, Frequency monitoring network (FNET) collects signals in every tenth second, so it is can used for electromagnetic phenomena. Now FNET monitors the entire North American electric grid with frequency, single phase voltage and angle sampled by frequency distribution recorder (FDR). It is much cheaper and convenient to install comparing with phasor measure units (PMU), which is first popular in 1980s as wide-area measurement. PMU can collect three-phase voltage, current, power flower and is installed in substation with special communication channels. It costs more than 10,000$. This high price slows down its universal around the world [1-3]. Distribution system level measurement let FDR work without high isolation requirement

as well as current or voltage transfers, such substation standard devices. Therefore, the price of single FDR is around 1,000, tenth of PMU's and it can be installed anywhere with 120v outlets even in office and home. Figure1.2 is the structure of FNET.

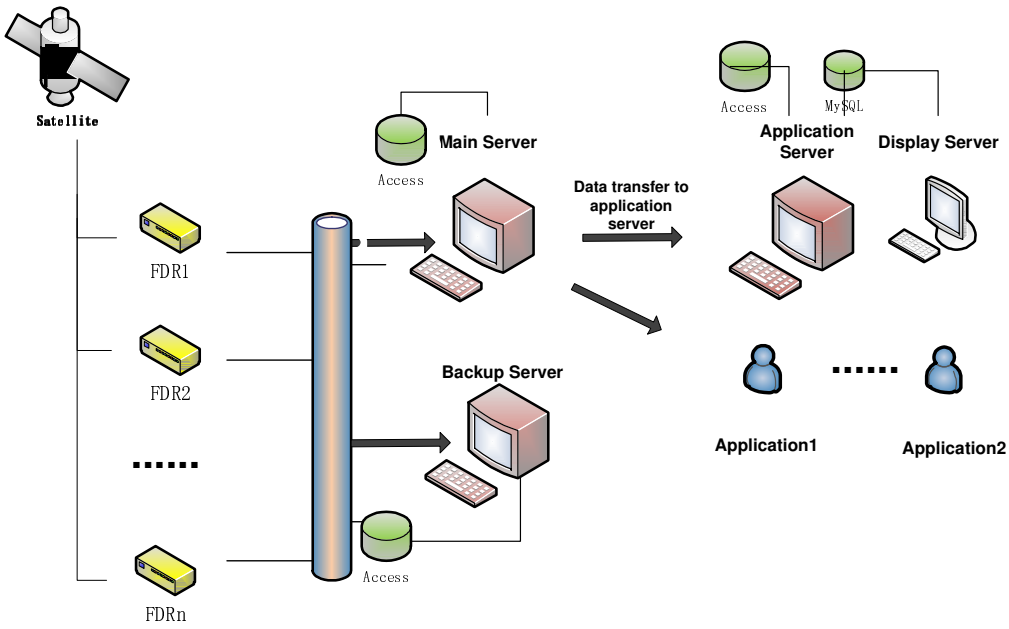

Figure 1.2 structure of FNET

Four components in FNET are FDRs, GPS satellite system, Internet, and FNET server. FDRs collect electric grid signals, frequency, voltage and angle, in distribution system, and send to internet. GPS systems with 120v outlet synchronize system clock for sampling and real time monitor on power system operation. That is, GPS make all the FNET data have three dimensions significance: electric value, timing scale and location scale. And FNET data are more useful and precise for power system overall view. All the data is received and sent on Internet. After receiving data with socket communication, the FNET sever deals with disturbance monitor, event estimate location, oscillation detection and so on. FDRs detect useful data from outlets and send these data with GPS tag (time, latitude and longitude) to internet. Once main server receives data, it will store in access database named as 'ddmmyy.mdb' and transfers data to application server and other application user for further development [4]. As its low-cost merits, FDRs have

been installed throughout whole America and Canada, China, including the Eastern Interconnection (EI), the Western Electricity Coordinating Council system (WECC), the Electric Reliability Council of Texas system (ERCOT), and the Quebec Interconnection. Figure 1.2 show the FDRs deployed and plan to be deployed in US map [5].

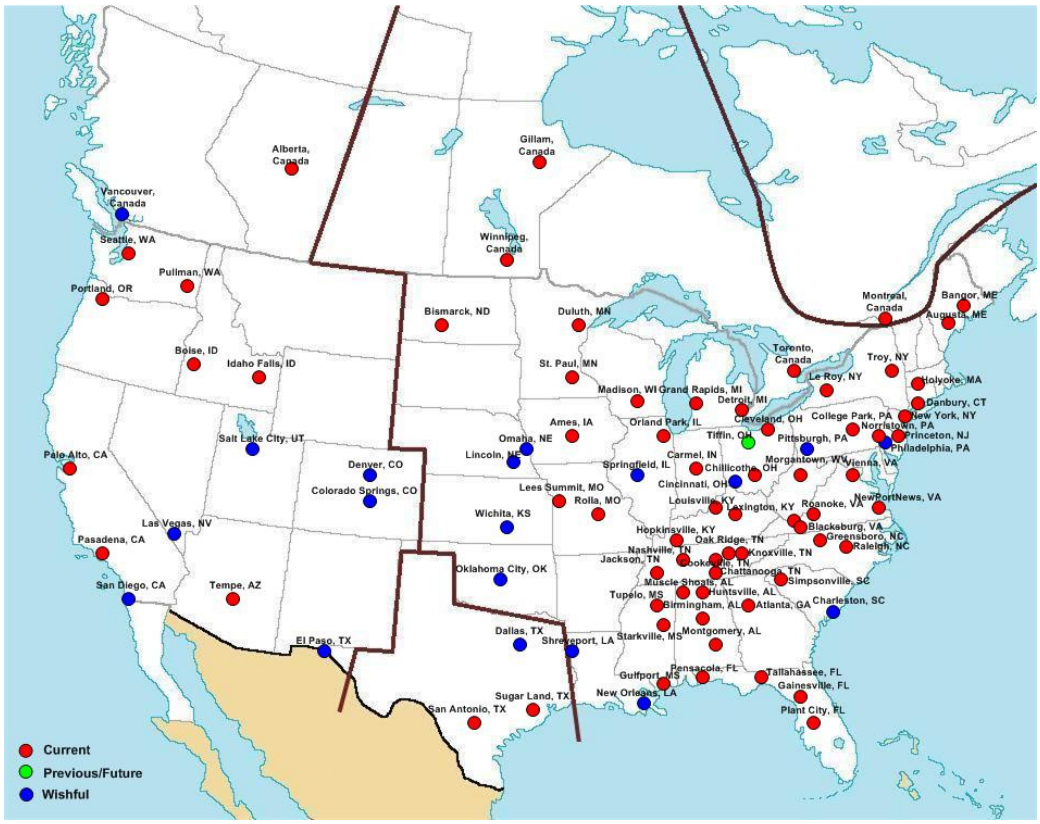

Figure 1.3 FDRs over the USA

FNET can detect frequency and voltage in real time. Once disturbance happens in US, frequency and voltage change. FNET is sensitive to any significant change in frequency with event trigger application. It can differentiate event classifications as load shedding or generator trip and estimate the mismatched power between generators and loads. It can estimate the event location with triangulation application and send event information emails to its clients in real time.

With so many merits, FNET also faces new challenge. The first one is more guaranteed communication. Communication on internet has many insecurity problems,

such as traffic congestion, missing data and missing synchronized signals of GPS need to be dealt with. Since FDRs are deployed in distribution system, a little change in load will bring high noise to FDR measure. So how to filter noises is another big issue on FDR improvement.

1.2 Organization of Study

This dissertation is organized as following:

At the beginning, Chapter 1 illustrated the role and features of Frequency Monitoring Network (FNET), which applies wide-area measurement and collects frequencies and voltage angles information of the whole America, even many other countries. After analysis the valuable data collected by FNET, this thesis explores the programming and application of event analysis and oscillation analysis.

Chapter 2 introduces two DLL modules in FNET Server and analysis dealing with FNET data. Since FNET data maybe have some small errors when delivered through Internet, so noise and missing data need to be handled before further analysis. Section 2.3 was discussing how to filter noise and complete probing. Current event analysis methods were covered and discussed in the remaining sections. Frequency data is applied to determine exact time when event happened at, time delay of arriving of frequency disturbance recorders (FDRs) and other oscillation information. Then, Chapter 2 presented the method of location of event algorithm. Use Time of Arrival Delay to analysis event and estimate its location and mismatch power amount.

Chapter 3 shows the detailed programming of these two DLL applications. Section 3.1 discusses the code of event trigger application as well as its interface. Following, the interface of DLL module and the key word to import and export DLL variables and function is described. Section 3.2 presents the work of triangulation module in Visual C++ and its main functions. Section 3.3 illustrates the multiply method to calculate event time, which is important when deal with pre-disturbance frequency in TDOA part. In section 3.4, there are two demonstrations of applying DLL applications.

The demonstrations shown how to load and use DLL. The requirement of running environment where DLL can be used and the interface exported by DLL are presented in Section 3.4.

Next, Chapter 4 focused on oscillation analysis: oscillation mode analysis, inter-area oscillation statistics, inter-area oscillation visualization and damping oscillation with Power System Stabilizer (PSS).Fundament of oscillation analysis was presented in Section 4.1 and 4.2. Since oscillations between interconnected synchronous generators are inherent to power system stability, statistical analysis of inter-area oscillation and its relationship of disturbance is evaluated in Section 4.3. General oscillation controls were discussed in Section 4.4, and PSS design case was presented in this Chapter.

The last part, Chapter 5 concluded the body of this work with a summary of the DLL applications and PSS small signal analysis.

Chapter 2 DLL Application Modules of FNET Server Program

2.1 DLL Introduction

When called functions in C++ libraries of standard code modules, people added the header file and use its language in .cpp body. For instance, when use format() function, which is store in <stdio.h> , the linker of executable program will find format() extension .lib and copy it into the .exe file[5]. So when one program use hundreds time of such function, such as output and input function, or more than ten programs in computer use format() function, there will need hundreds even thousands copies of format codes in .lib for every executable programs. This will occupy much disk space. To deal with this consider, dynamic link library is been used. Now almost all the application programming interfaces of Microsoft Visual are stored as DLL. The idea of a dynamic library is to store all the code for the functions in lib and dll, but do the linking only when the program is run comparing with static library linking at debugging time, and this is why it's call 'dynamic'. However, dynamic link library is vey like a normal executable program. It has a lot of classes, variables, and functions exported to other programs calling. Fig 2.1 presents how DLL works.

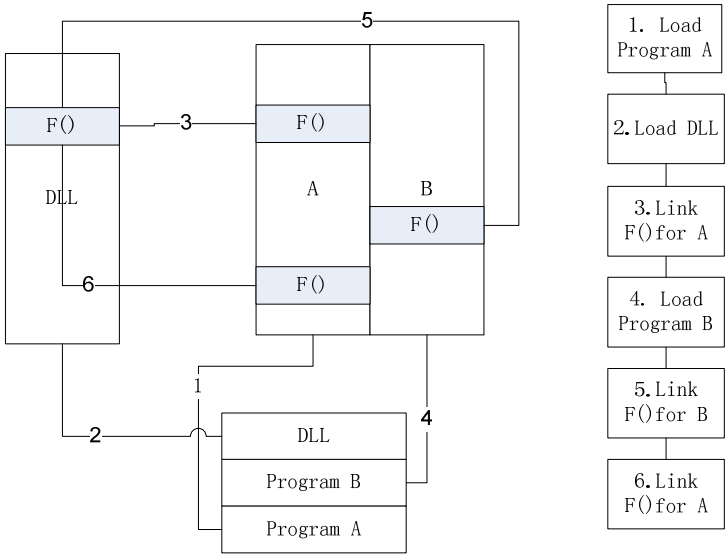

Figure 2.1 DLL work

There are several reasons to use DLL. First, since there is only one copy of the DLL on any computer used by all the applications that need that library code in memory, it can save disk space and make execute program smaller. Second, simplify the project management. When different people did different part in one project, the best method combining all the parts is to use DLL. Besides, when one DLL need to be modified, people don't need do any change in project but only in DLL itself. For example, if there is a bug in the DLL, a new DLL can be created and the bug will be fixed in all the programs that use the DLL just by replacing the old DLL file. DLLs can also be loaded dynamically by the program itself, allowing the program to install extra functions without being recompiled. Introduce DLL modules into FNET server can increase the usability and convenient.

2.2 Build DLL Program

There are three kinds of DLL: No-MFC DLL, MFC regular DLL and extension DLL. The first one is used in win32 program, so it can't use MFC class and is easy to use. MFC regular DLL support MFC class in DLL program, however, class member or function can't be used as interface with executable program. Extension MFC DLL can use all the MFC class and its derived class, but the environment where DLL is called must be MFC program. In this project, setevent() of CEvent class is exported to FNET server program where DLLs are used. And FNET server program is MFC based. So the extension DLL is chosen instead of the others kind [6].

To build an extension DLL file, start from Visual C++ Application Wizard. First, create a new project you can press Ctrl+Shift+N as keyboards shortcut, and choosing MFC project and the template is MFC DLL. Then input the name you want for DLL. Here named as "triaDLLtry.dll", shown in Figure 2.2.

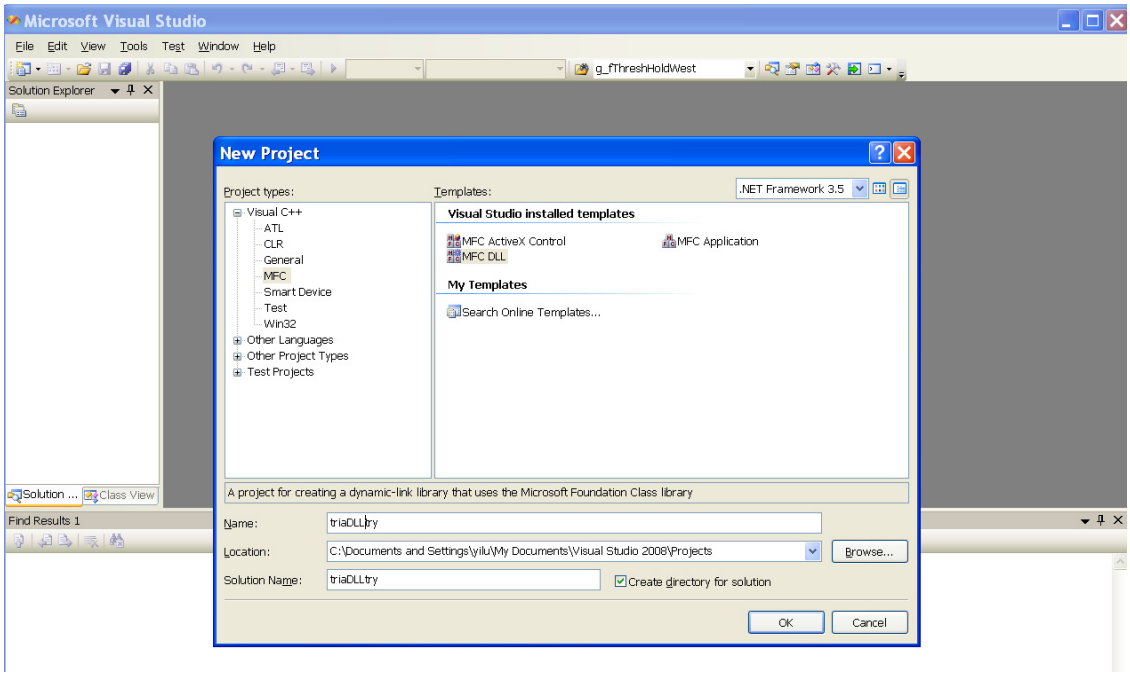

Figure 2.2 Step 1 of building DLL

Click the OK button. In the next window, choose MFC Extension DLL. Now there are two checkboxes “automation” and “socket” can be chosen. Automation function brings you the potential for hosting objects created and managed by one application inside another [5]. Sockets function provides classes to communicate over a network. Here don’t need either of them, left blank, and click finish. Shown as Fig 2.3.

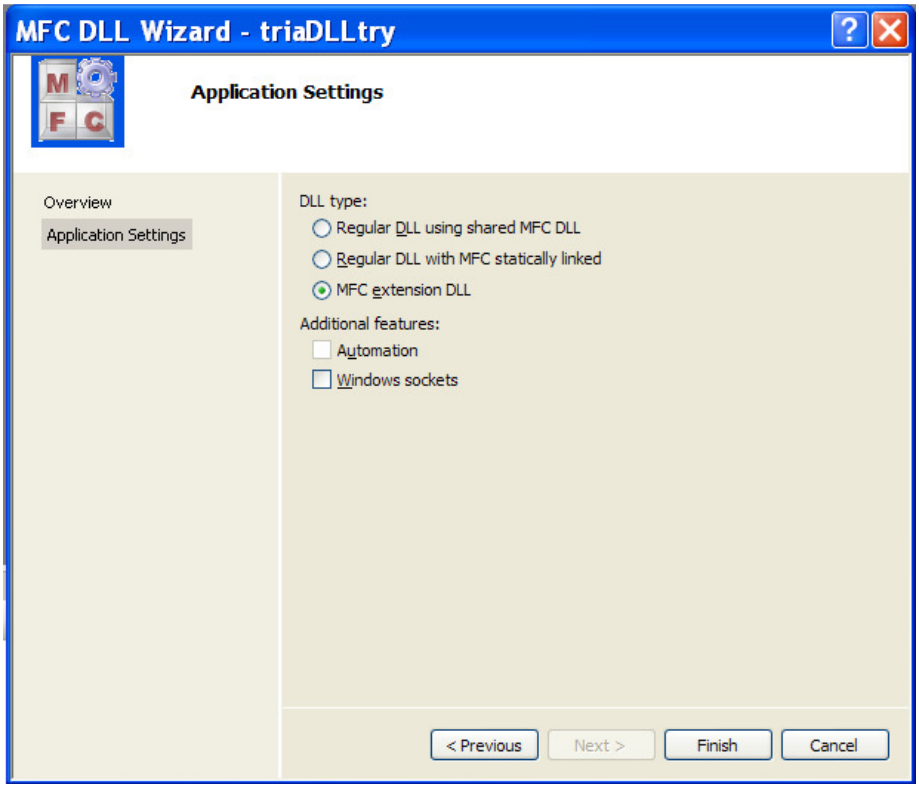

Figure 2.3 Step 2 of building DLL

Now DLL is built with DLLMain() function including in "dllmain.cpp". Next, determine which functions, variables or class need to be exported to the environment where DLL is used. There two main methods to export elements in DLL. First method is use "__declspec(dllexport)" key word. "__declspec(dllexport)" generate the export names automatically and place them in a .lib file. This .lib file can then be used like a static .lib to link with a DLL[7]. As __declspec(dllexport) adds the export directive to the object file so .def file is not required. Another method is using module-definition (.def) file. DEF file is a text file containing one or more module statements that describe various attributes of a DLL. Its format is as following Figure 2.4:

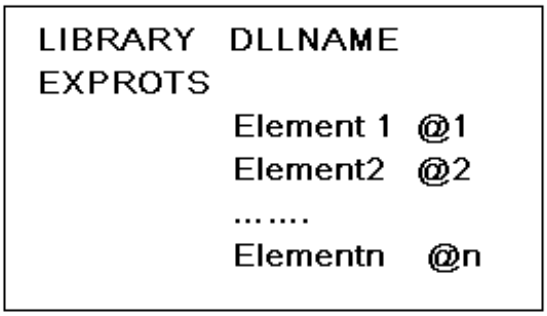

Figure 2.4 DEF file format

Getting elements exported by DLL in calling program has two different methods. One is static with key words "__declspec(dllimport)" , another is real-time calling by getting address of elements. The easiest and dynamic export with three functions: LoadLibrary(), GetProcAddress(), FreeLibrary(). "LoadLibrary" maps DLL file into the address space of the process where DLL is called and return the handle of DLL module (indicating success). After returning the handle of DLL module, use "GetProcAddress" to get the exported elements address, especially functions. With the address, the function can be called in process. At last, make sure to call function "FreeLibrary", decrementing the reference count of the loaded DLL module.

2.3FNET Data Analysis and Frequency Monitor Principle

As discussed in Chapter 1, communication over internet is not always stable as well as the GPS signals. Through practice and experience, interpolation is an effective method to deal with the missing data. However, give up the FDR's data, when its missing data ratio is larger than 10% in the calculation window. For instance, ith FDR missed frequency signals between time p and time k in m-point window. Now its frequency data can be rewritten as:

$$f_i = \{f_{i,1}, f_{i,2}, \ldots, f_{i,p-1}, f_{i,p}, f_{i,k}, f_{i,k+1}, \ldots, f_{i,m}\} \text{ where } p < k-1 \qquad (2.1)$$

To interpolate data lost between p and k, use the formula:

$$f_{i,p+l} = f_{i,p} + \frac{l(f_{i,p} - f_{i,k})}{k - j}, \quad (l \in [1, k - p - 1]) \tag{2.2}$$

Furthermore, FDRs collect signals in distribution level and it is sensitive to the change in distribution level. Therefore, the disturbance or load change in distribution system will interface the value sampled. Although system can detect the disturbance or oscillation in distribution level, which can reflect the power system, still need to filter these minute signals as noise and outliers. . At present, to exclude distribution noise, FNET group use 10 moving median window to smooth data. The calculation of n moving median window is that: given a sequence $\{f_i\}_{i=1}^{N}$, an n-moving mean is another sequence defined from by taking the median as:

$$\hat{f}_i = \begin{cases} f_{\frac{2i+n-1}{2}}, \ n \ is \ odd \\ (f_{\frac{2i+n}{2}-1} + f_{\frac{2i+n}{2}})/2, \ n \ is \ even \end{cases} \tag{2.3}$$

The smoothing result of 10 moving median window is shown in Figure 2.5

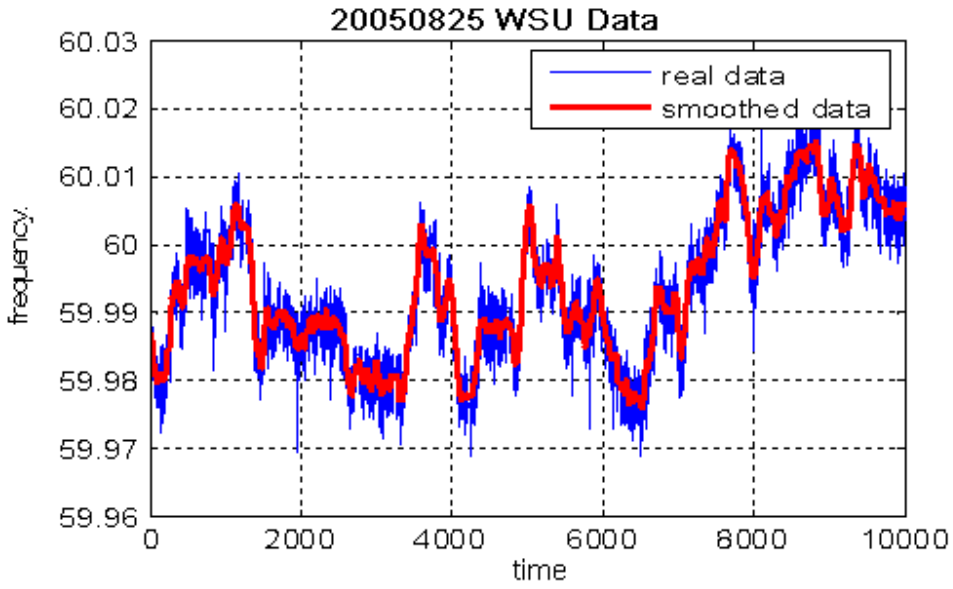

Figure 2.5 10 moving median window filter result

After dealing with data problems, FNET data can reflect power system accurately.

Generator swing equation [8]

$$\frac{2H}{\omega_0}\frac{d^2\delta}{dt^2} = T_m - T_e - \frac{K_D}{\omega_0}\frac{d\delta}{dt} \qquad (2.4)$$

Here:

T_m: mechanical torque; K_D: damping factor;

T_e: electrical torque; H: inertia constant;

ω_0: rated electrical angular velocity; δ: angular position of the rotor;

As $\frac{d^2\delta}{dt^2} = \frac{d\omega_r}{dt} = \omega_0 \frac{d\omega_r}{dt}$ (2.5)

Neglecting damping factor, 2.4 can be rewritten as

$$\frac{2H}{\omega_0}\frac{d\omega}{dt} = 2H\frac{df}{dt} = P_m - P_e - K_D f \approx P_m - P_e \quad (2.6)$$

That is $2H\frac{df}{dt} = P_m - P_e$ (2.7)

From equation 2.7, when a disturbance or event (generation trip, line trip, load shedding) takes place in power system, the imbalance of the generation and load will cause the system frequency to suddenly change. So frequency is an important electrical quantity reflecting power system disturbance. With this principle, FNET detects disturbance based on frequency data collected by FDRs over North America.

2.4Event Trigger Module of FNET Server

Events in power system will either bring frequency increasing or decreasing. As discussed in last section, the rate of change of system frequency can be used as an indicator of the disturbance. So if system can detect the change of frequency in real time that means then can monitor power system in real time. Furthermore, FDRs are installed all over the North America. More FDRs installed, more real-time information can be got for whole USA's grid, including the Eastern Interconnection (EI), the Western Electricity Coordinating Council system (WECC), the Electric Reliability Council of Texas system (ERCOT), and the Quebec Interconnection.

FNET server uses socket communication to build connection with FDRs. Server save data automatically into access database and event buffer, once socket listened data. Event buffer is a 20 size vector. It is used to save 20 seconds simplied FDR data. Fig 2.6 shows its structure and function.

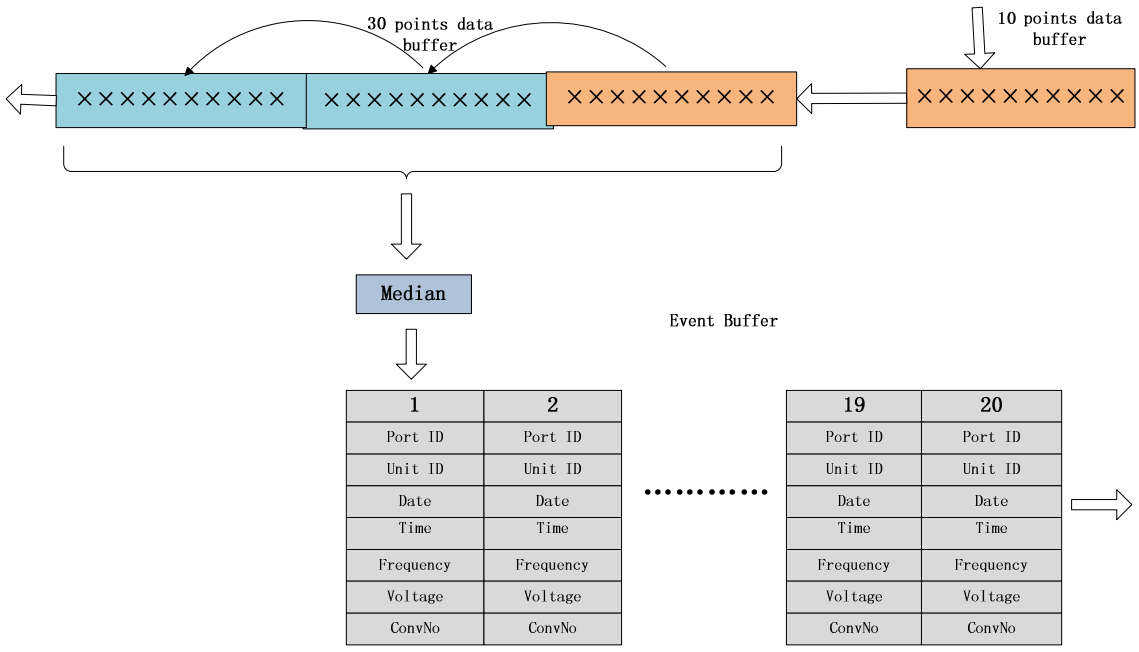

Figure 2.6 Event buffer structure

As seen from Fig 2.6, FNET receives ten points data in every second. First, save these 10 points data into data buffer. When the 10 points data buffer is full, the data in 30 points data buffer are moved forward by 10 points and append the 10 point data at the end. Since FNET signals have noise and outlier, so the data need to use 30 points window size median as equation 2.3, and append it to the end of event buffer. At last need check event with event. After getting ten new points from socket, the server calculates the median of that data and appends it to the end of event buffer, at the same time, deletes the first data to keep the buffer at constant length. Every second, after getting new data, an event check will be performed.

Event trigger can classify event as load tripping or generator tripping, which will lead system frequency to increase or decrease. Other type events, e.g. line trip or

oscillation, will be detected by another trigger. Figure 2.7 and figure 2.8 are the typical frequency responses of generator trip and load shedding.

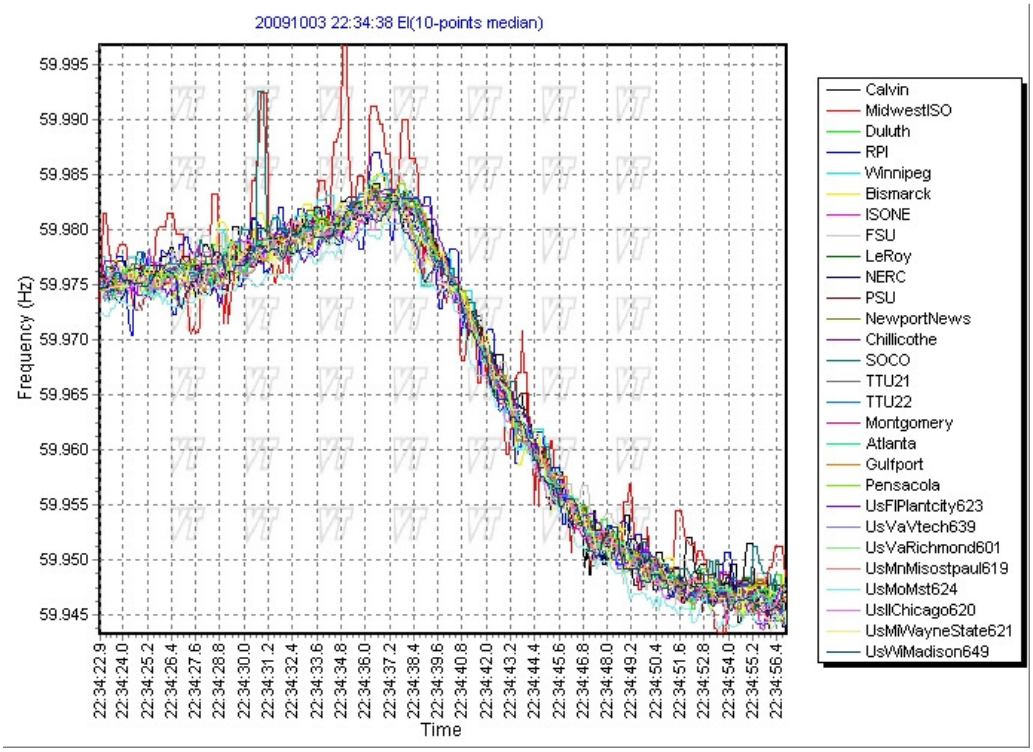

Figure 2.7 frequency response of generator trip 20091003

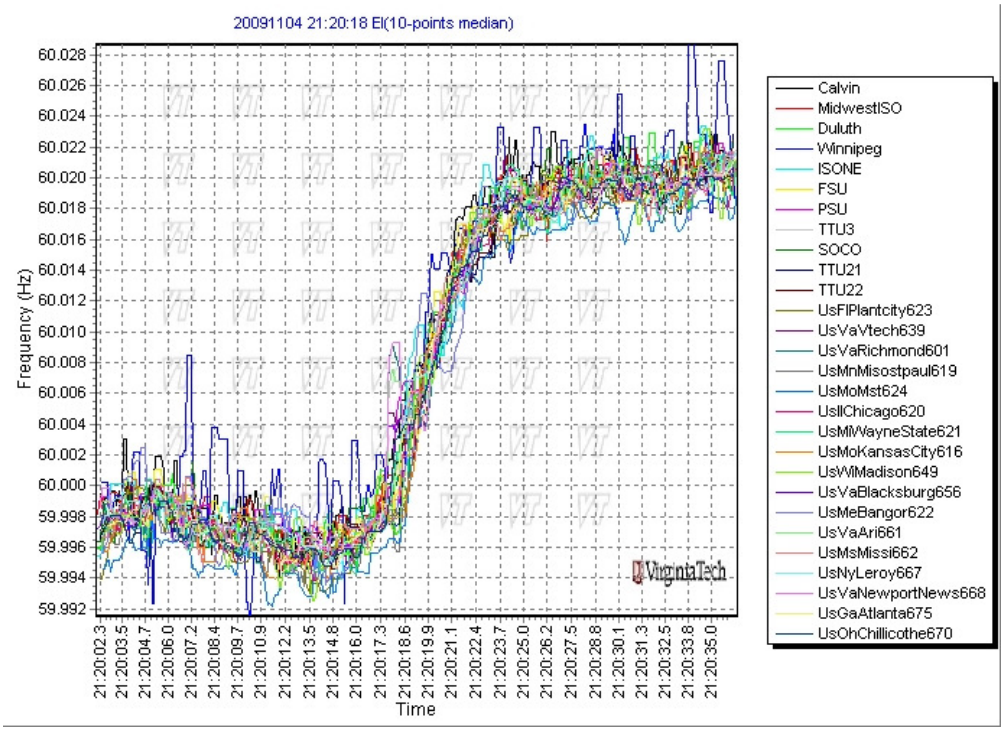

Figure 2.8 frequency response of load shedding 20091104

Now the event trigger algorithm is present as following.

At first, as discussed all above, considering data conditioning with moving window filter, interpolation the missing data and save the data in the buffer. To get the change of frequency, calculate df/dt in ten data window:

$$\begin{cases} \dfrac{df}{dt}_1 = \dfrac{(f_1 - f_5)}{(t_1 - t_5)} \\ \dfrac{df}{dt}_2 = \dfrac{(f_2 - f_6)}{(t_2 - t_6)} \end{cases} \tag{2.8}$$

Here, f_i is the frequency of i-th data point in ten data window. Through the swing equation, the change of frequency indicates mismatch of generation and load, which is called by event. So set a threshold of df/dt, when its absolute values of both df/dts at first data point and second data point are larger than the threshold, event happens. Fig 2.9 shows the process of picking up event.

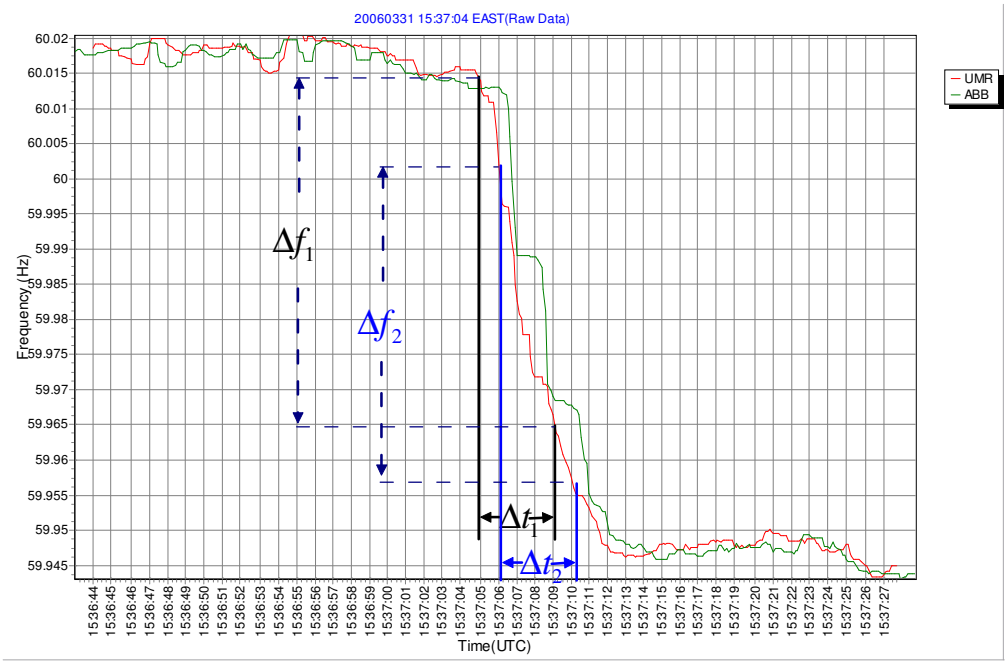

Figure 2.9 *change of frequency*

Now the only question is how to determine the threshold. The thresholds for each interconnection are different since the inertias of different systems are different. The same amount of generation or load loss will cause the system frequency to change at different rates. Based on two years' observation and statistical analysis of every day's df/dt in different areas, FNET set the thresholds for three areas as following table.

Table 2.1 thresholds for interconnection

Interconnection	Threshold(Hz/s)
EI	0.0036
ERCOT	0.0095
WECC	0.007

So the equation of event trigger can expressed as equation 2.9 below.

$$\begin{cases} \dfrac{df}{dt}_1 = \dfrac{(f_1 - f_5)}{(t_1 - t_5)} \\ \dfrac{df}{dt}_2 = \dfrac{(f_2 - f_6)}{(t_2 - t_6)} \end{cases} \geq threshold \qquad \begin{array}{l} EI \ \ 0.0039 \\ ERCOT \ \ 0.0095 \\ \text{WECC } 0.07 \end{array} \qquad (2.9)$$

The event time is the time of first data with df/dt larger than threshold. To get the event time more accurate is important for the triangulation module. Here, a new event time method is present.

The DLL module, all the calculation and data conditioning are doing inside of DLL. The calling process, FNET server, only need know two variables: *"m_cEventtime"* indicating event time and "g_event" representing event trigger state, with code:

```
__declspec (dllimport) CString m_cEventtime;
__declspec (dllimport) CEvent g_event;
```

2.5 Triangulation Module of FNET Server

The main purpose of the triangulation module is to find the event location by using triangulation method. Besides that, this module executes many important works as shown in Figure 2.10.

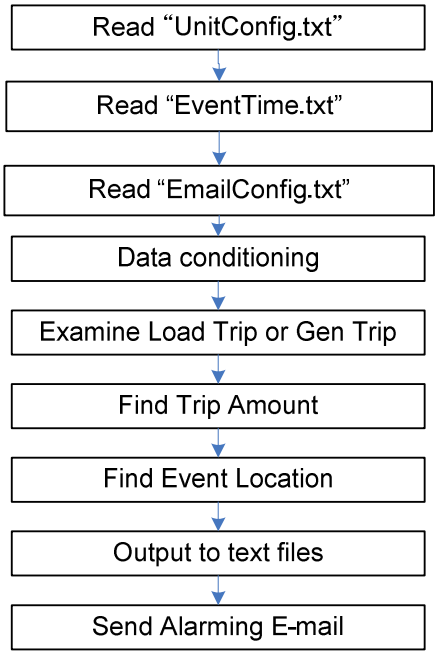

Figure 2.10 basic work process of Triangulation Module

This module is an independent application from FNET Server module. This module can be started by executing “Triangulation.exe”. Normally, this module is executed by the FNET Server. It starts its own application and read data from text files that contains FDR information, E-mail information, and frequency data stored by the FNET Server. The frequency data are measured by many FDR units in various sites and transferred to the FNET Server through the Internet. They might have disturbances according to the measurement environments and some missing data. Therefore, before starting the triangulation module, the frequency data should be conditioned by checking missing data and using moving median.

Frequency perturbations travel throughout the EUS and in any power system in general with finite speed [8]. In the EUS, frequency perturbations tend to travel with velocity on the order of 500 mi/s and in the WECC, frequency perturbations have been observed to travel with velocity on the order of 1000 mi/s [9]. The event location

algorithms herein capitalize on the fact that frequency perturbations after events like generation trips will be measured by each FDR at different times. This time delay of arrival (TDOA) then gives a parameter by which to order the response of each unit. Figure 4.1-1 shows the frequency signature of a typical generation tripping event. Notice the very discernible delay in the frequency drop as measured by several FDRs. Indeed it is discernnable that the frequency plotted in red drops first at time, and the frequency plotted in green drops last at time. This frequency drop is expected as the result of a generation/load mismatch in the few seconds following a significant loss of generation.

The triangulation module examines the cause of the event like either generator trip or load trip and then estimates the trip amount and the event location. It stores the output data to text files and sends alarming message to the operator by e-mail. The flow chart is shown in figure 2.11

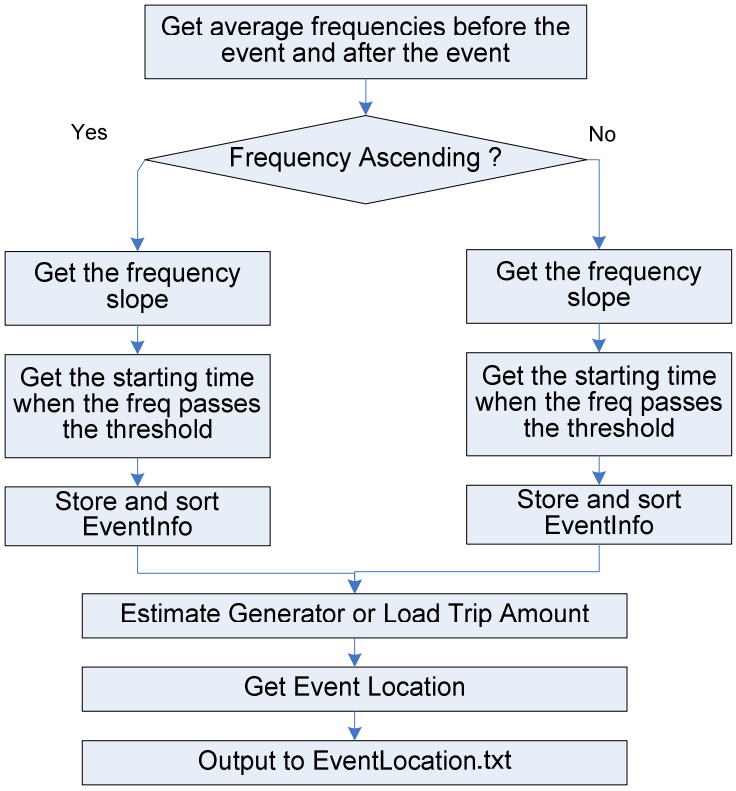

Figure 2.11 progarm decision flow

As known, the power system only has one frequency value even across different parts of an interconnection system during steady state. However, in disturbance analysis, such as generator trip or load rejection, which makes operation equilibrium lost, the system frequency and the change rate of system frequency are not the same through the system in dynamic analysis [10]. At this period of time, system is dominated by the swing equation discussed in chapter 2, which describes the generator rotor speed change and the active power imbalance. FNET claims the frequency deviation propagates as "Frequency Wave" or electromechanical wave [11]. With observation of visualization analysis, frequency perturbations travel throughout the EI with finite speed. So with the distance between FDRs and event location, the time of FDR detected event time will be different [11, 12]. This is time delay of arrival (TDOA). Here distance is electrical

distance which is the inverse proportion to the inductance between the event location and FDR measurement location. To improve the speed of program, geographic distance is used instead of electrical distance. Figure 2.12 shows the event time arrival difference of every FDRs clearly.

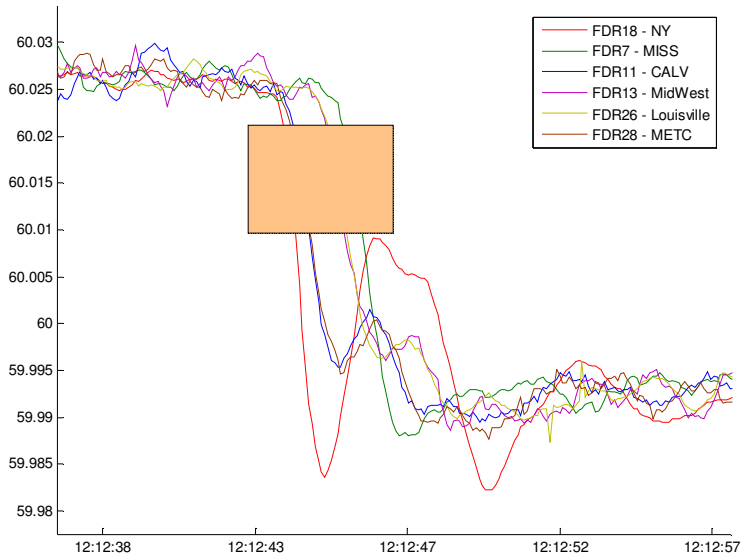

Figure 2.12 TDOA of different FDRs

The algorithm of TDOA is discussed as following. With date saved by event trigger module, and event time calculation, then get the pre-disturbance frequency (f_{pre}) as average of 3 seconds data calculated by n FDRs before disturbance:

$$f_{pre} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{30}\sum_{p=1}^{30} f_{i,p}\right) \qquad (2.10)$$

Here, n: the number FDRs involving this event;

f_{pre}: pre-disturbance frequency;

$f_{i,p}$: frequency at p data point before event time of i-th FDR

With this pre-disturbance, get f_{TDOA} by add or minus 0.008 Hz [13] with different event type (load shedding or generation trip).

$$\begin{aligned} &\textit{Generation trip:} \quad f_{\mathrm{TDOA}} = f_{pre} - 0.008Hz \\ &\textit{Load shedding}: \quad f_{\mathrm{TDOA}} = f_{pre} + 0.008Hz \end{aligned} \tag{2.11}$$

The related time of f_{TDOA} is the TDOA for each FDR. Triangulation module saves f_{TDOA} in vector X along with the FDR's latitude, longitude, and frequency wave propagation speed. The process is shown clearly in Fig. 2.13 and Fig 2.14.

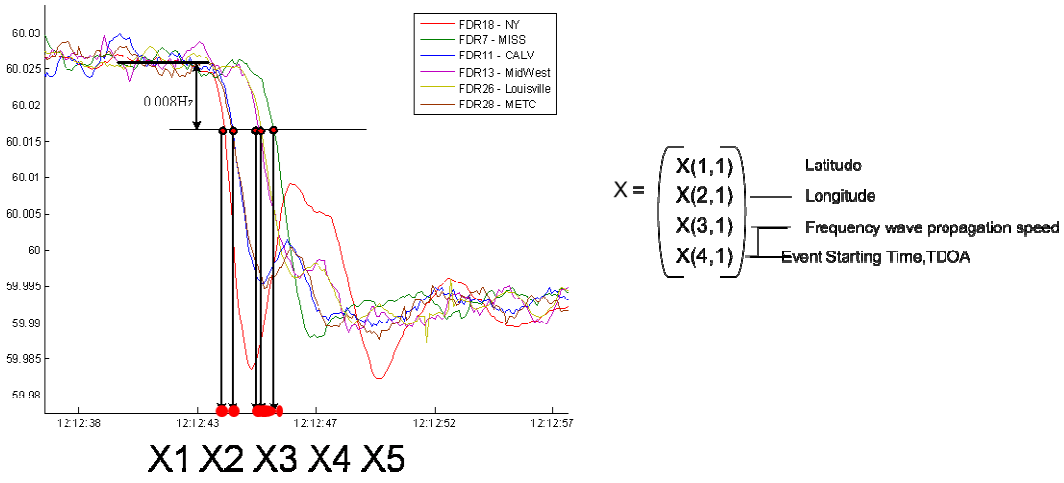

Figure 2.13 TDOA calculation process

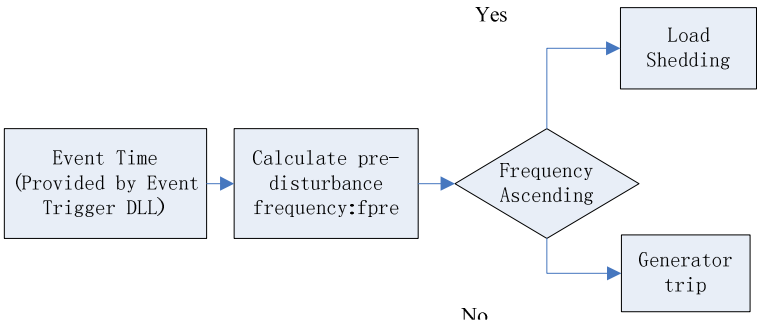

Figure 2.14 programming flow chart of calculating TDOA

As shown in section 2.3, the relationship between change of frequency and imbalance of generation and load as equation 3.3

$$2H\frac{df}{dt} = P_m - P_e \tag{2.12}$$

Instead of using the instantaneous derivative of frequency, which is the frequency deviation over a short period of time, is used to estimate the power imbalance [14]:

$$\begin{aligned} \Delta P &= P_m - P_e = M \cdot \Delta f \\ \Delta f &= f_i - f_{i+1} \end{aligned} \tag{2.13}$$

Here ,

f_i: frequency at i-th point;

f_{i+1}: frequency after 0.1 second of i-th point;

$M(2H)$: inertia constant in seconds

Although $M(2H)$ cannot obtain easily as the information of system load during each event is not available, the system inertia can be estimated by first-order curve fitting

over these data points. With observation of FNET database, if assume a linear relationship between ΔP and Δf in long period as

$$\Delta P = P_m - P_e = a\Delta f + b$$
$$\Delta f = f_{pre} - f_{aft} \qquad (2.14)$$

Here ,

f_{pre}: pre-disturbance frequency;

f_{post}: post-disturbance freuqncy;

a, b: the linear coefficient

Figure 2.15 presents the first-order curve fitting of estimating a and b value. This figure also proves the relationship between the generation/load mismatch and the frequency change rate described in equation 3.3.

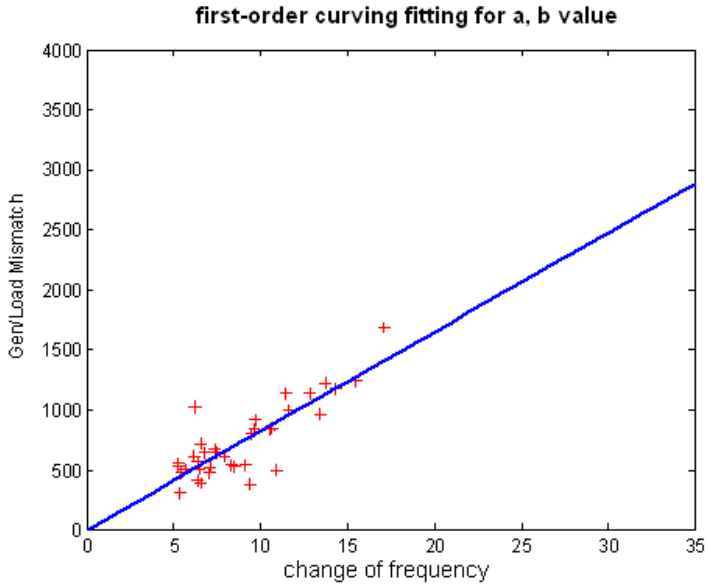

Figure 2.15 first-order process

With statistical analysis, get the experience value of a, b. The

$$\Delta P = 98.031\Delta f - 131.99 \qquad (2.15)$$

Figure 2.16 shows program flow chart of mismatch amount estimation.

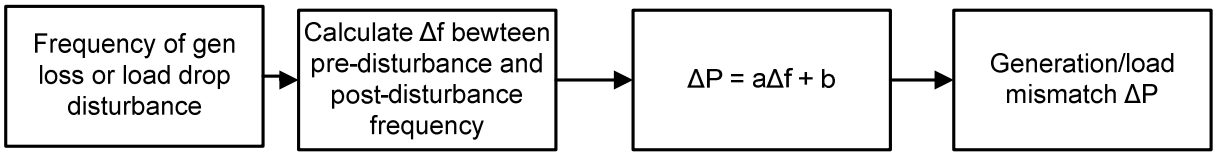

Figure 2.16 flow chart of mismatch estimation

Therefore, when event happens, estimate of the amount of tripped generation or load rejection in fault analysis is based on the relationship between frequency change and active power imbalance in the system according to equation 2.15.

2.6 Algorithm of event location estimation

Implicit in the use of of TDOA to estimate the location of an event is the assumption that the time of arrival (and by consequence the time delay between units) of the electromechanical wave disturbance can be accurately determined. proposed another disturbance location estimation method proposed based on the electromechanical propagation study of FNET. Generally the frequency perturbation caused by the disturbance will propagate throughout the system. Therefore, theoretically the closer the measurement point to the event location, the earlier it will detect the frequency perturbation. The frequency perturbation arrival time or the frequency propagation sequence is the critical information used to estimate the event location.

From basic physics we obtain the key concept in least-squares event location.

Below we have a basic equation that relates mean velocity, time of travel, and distance traveled:

$$Distance = Velocity \times Time \tag{2.16}$$

Applying this concept to electromechanical waves with the assumption that such waves travel throughout the electric grid as a continuum, (2.16) can be developed using the Pythagorean relation written in terms of our data as:

Based on the relation between the distance and the time of wave travel, [15] gives the following equations:

$$(x_i - x_h)^2 + (y_i - y_h)^2 = V^2 (t_i - t_h)^2$$

For each responding FDR, we can write:

$$\begin{aligned}
&(x_1 - x_h)^2 + (y_1 - y_h)^2 = V^2 (t_1 - t_h)^2 \\
&(x_2 - x_h)^2 + (y_2 - y_h)^2 = V^2 (t_2 - t_h)^2 \\
&\qquad\qquad \vdots \\
&(x_n - x_h)^2 + (y_n - y_h)^2 = V^2 (t_n - t_h)^2
\end{aligned}$$

Here (x_n, y_n) is the (x, y) coordinates of n^{th} FDR

(x_h, y_h) is the (x, y) coordinates of hypocenter (origin of the disturbance)

t_n is the frequency perturbation arrival time measured by n^{th} FDR

t_h is the time the disturbance happened

V is the mean value of the electromechanical wave propagation.

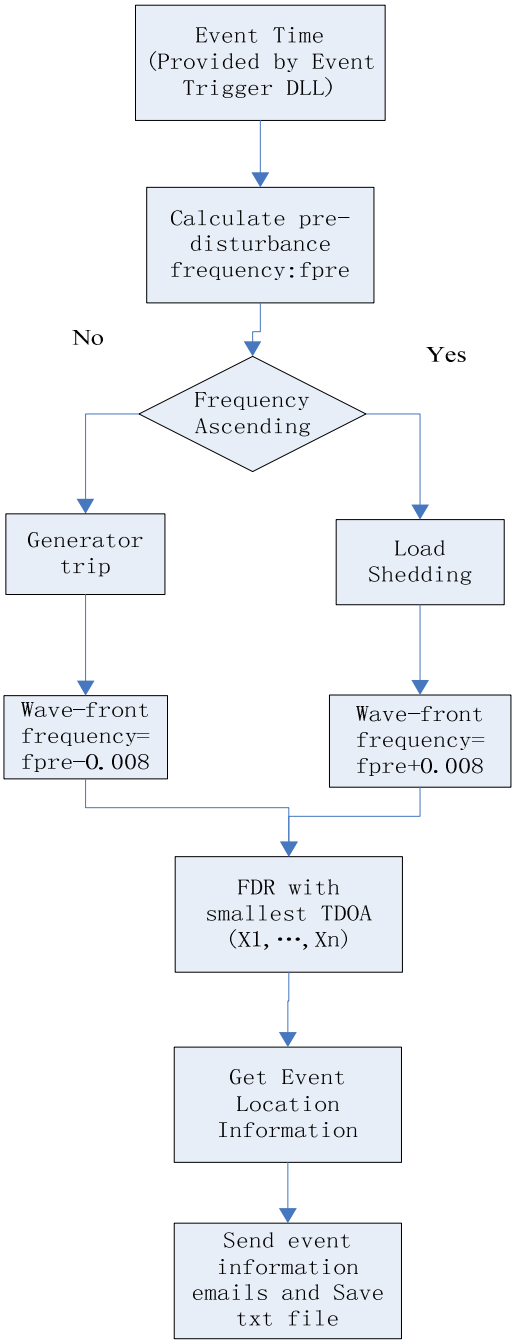

Figure 2.17 Decision flow

Solve the equation group we can get the estimate location. Here, the speed in the Eastern Interconnection (EI) is arranged in 200 miles/sec to 660 miles/sec, and about 1000 miles/sec in WECC[16]. Figure 2.17 shows the whole triangulation module flow chart.

First calculate pre-disturbance frequency. The triangulation module examines the cause of the event like either generator trip or load trip and then estimates the trip amount and the event location. Then, it stores the output data to text files and sends alarming message to the operator by e-mail.

Chapter 3 Programming of DLL Application

3.1 Event trigger programming

In last chapter, the main algorithms of event trigger and triangulation module are presented. The programming of these two applications will be discussed below based on the algorithms and Visual C++.

The function interface of this module is “void Trigger()”, and this is the entering function for event trigger. In this function, it connects FDR raw data database with name of “FNET” and checks the connection result.

```
extern "C" __declspec(dllexport) void Trigger()

{
        AFX_MANAGE_STATE(AfxGetStaticModuleState());

        if (!OpenAccess())
                AfxMessageBox(_T("Unable to Open Connection to Access Database!"));

    m_rs = new CFRURawData11(NULL);

        g_bEventTriging = false;
        g_fThreshHoldWest = 0.003;
        for (int i=0; i<10; i++)  //for 60 Hz system
                m_fTenData[i] = 60.0;
        m_iTenPointBufferIndex = 0;
        m_fSum = 60.0;
        DataIndex=0;

        ReadDataFromAccess();
}
```

If fail to construct connection with FNET access database, it will call the message box of “Unable to Open Connection to Access Database”. Once the connection is established, generate new Recordset objective “m_rs” to deal with the database data.

Here “CFRURawData” is a derived class from Recordset class. It save the information of index, unit ID, date and time, frequency and time index from FNET access database.

After connect to the database and receive the Recordset, the programming initials the variables. The state of event trigger, g_bEventTriging, is false, the index for ten point buffer is zero, sum of frequency is 60Hz, and index of data is zero. Read into the threshold value for interconnection systems, and initials the ten size data buffer, m_fTenData with zero.

Next step is to receive data from database and deal with these data to trigger event. Use the state of database end, function IsEOF(), and the state of event trigger, g_bEventTriging to control the loop times. Every time, read one row information, unit ID, Data and time, frequency and index time, of FNET database with function GetFieldValue(). The programming is as follows:

```
while ((!m_rs ->IsEOF())&&(g_bEventTriging == false))
{
        m_rs ->GetFieldValue (1,str);
        m_UnitItem.UnitID=_wtoi(str);
        m_rs ->GetFieldValue (2,str);
        m_UnitItem.DataTime = str;
    m_rs ->GetFieldValue (3,str);
        m_UnitItem.FinalFreq = _wtof(str);
        m_rs ->GetFieldValue (4,str);
        m_UnitItem.TimeIndex = _wtoi(str);
    m_vDBData.push_back(m_UnitItem);
        EventTrigger(m_UnitItem);
        m_rs->MoveNext();
        }
```

When get the variables value, save them into a structure variable m_UnitItem, then push this struture into m_vDBData, a temple vector with structure of UnitItem. “UnitItem” is a structure to save the data received by socket with elements “UnitID”, saving the ID number i of FDRi, “DataTime”, saving the time of data received by socket,

"FinalFrequ", saving the frequency value detected by FDRi at "DataTime" point, and "TimeIndex", the transferred integer time data with format here:

Hour*36000+Minute*600+Second*10+Demi-second 3.1

With m_UnitItem from FNET database, call the EventTrigger function. This function is programmed to check event data every 1 second, the process is shown in Figure 3.1.

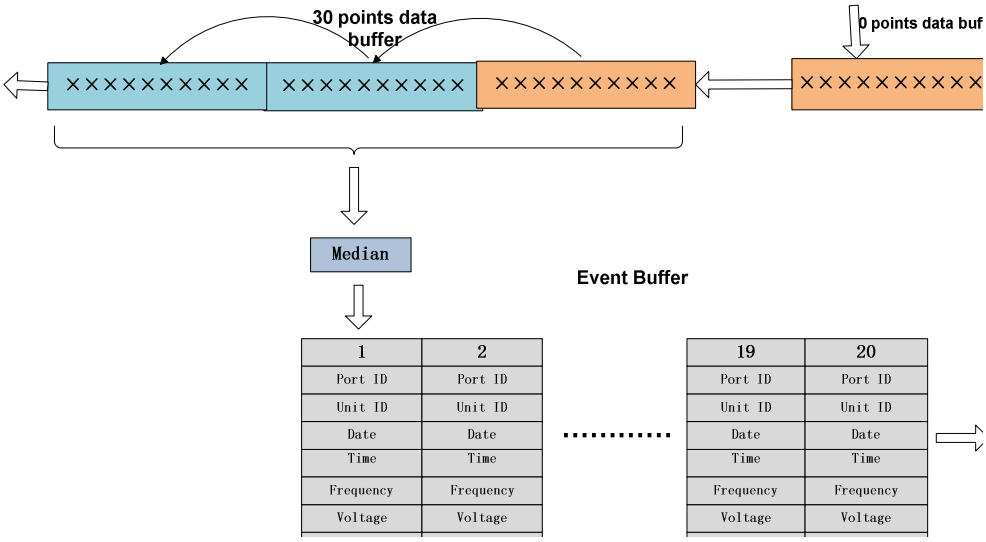

Figure 3.1 Event Trigger Plot

Every second 10 points data are coming into m_fTenData, firstly so we use a 10 points data buffer to save data. Once the 10 points data buffer is full, copy this buffer into a temple buffer to condition the data and save the fifth data into m_strData, 30 points data buffer. Then the new coming 10 points data will be appended to the end of the 30 points data buffer. Data conditioning programming is following:

```
void combsort(float *a, int amount)
{
        int gap=amount;
        bool swapped=false;
        while(gap>1 || swapped)
        {
                gap=(gap*10)/13;
```

```
            if(gap==9 || gap==10)
                gap=11;
            if (gap<1)
                gap=1;
            swapped=false;

            for ( int m=0; m<amount-gap; m++)
            {
                int n=m+gap;
                if (a[m]>a[n])
                {
                    a[m]+=a[n];
                    a[n]=a[m]-a[n];
                    a[m]-=a[n];
                    swapped=true;
                }
            }
        }
```

Then we get a filtered data using 30 points window size median, and append it to the end of event buffer, m_vEventBuffer, a twenty-point buffer. The buffer "m_vEventBuffer" is constructed with this structure for conditioned data of event trigger. At last we check event using the data in event buffer. When this buffer is full, calculate DF/DT to check event. There are two DF/DT value, one is between first point and fifth point, another one is between second point and sixth point. If the absolutions of two DF/DTs are both larger than the threshold, event happens.

```
bool CheckEvent(double dfdt1, double dfdt2,const double threshold)//without location
{
    if( (dfdt1>threshold && dfdt2>threshold) || (dfdt1<-threshold && dfdt2<-threshold) )
    {
        m_cEventData = m_vEventBuffer[0].DataTime;

        if ( !g_bEventTriging )
        {
            g_bEventTriging = true;
            CString str = _T("there is an event at ")+m_cEventData;
            AfxMessageBox(str);
```

```
                }

                return true;
        }
        else
        {
                return false;
        }
}
```

After detected event, clear values in m_vEventBuffer, make the b_EventTriging true.

3.2 Triangulation programming

Triangulation module is to estimate event location. It reads data from text files of FDR information, E-mail information, and frequency data stored by Event Trigger Module and FNET servers. As discussed above, some missing data and noise need to be considered. So before starting the triangulation module, the frequency data should be conditioned by checking missing data and using moving median. The main work of triangulation module is shown in Fig 3.1.

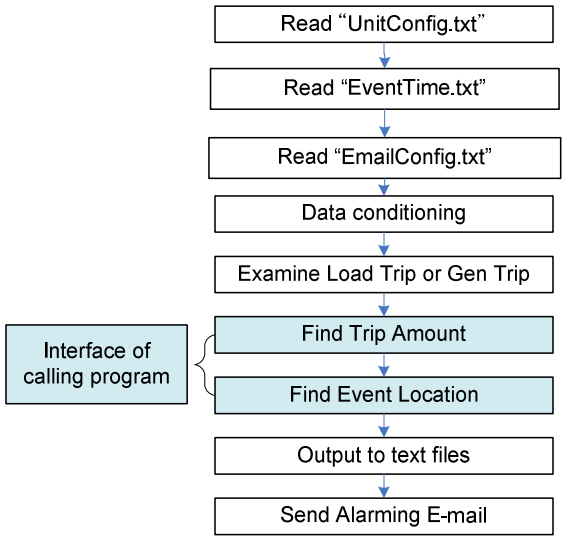

Figure 3.2 work process of Triangulation Module including calling

To use Triangulation module, start with function “void TiangProc()”, the entering function of Triangulation module. Use “extern “C” _declspec (dllexport)” to export function to outside. And use “extern “C” _declspec (dllimport)” to declare this function when used in execute program. In TriangProc() function, it calls read data function to load information of FDR, event and FDC with functions:

```
ReadUnitConfig();          //Read unit config information
    ReadFDCInfo();         //Read FDC information
    ReadEventInfor();      //Read event time information exported by FNETServer
```

Flow chart is shown in Figure 3.3

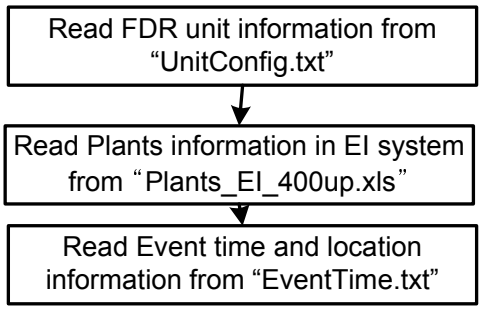

Figure 3.3 Reading data in Triangulation module

Use vector “EastFDR”, “WeccFDR”, and “TxFDR” to save the FDR information with FDR name, FDR port, longitude, latitude, coordinary X value, coordinary Y value and location of interconnection system. Take EastFDR vector for example:

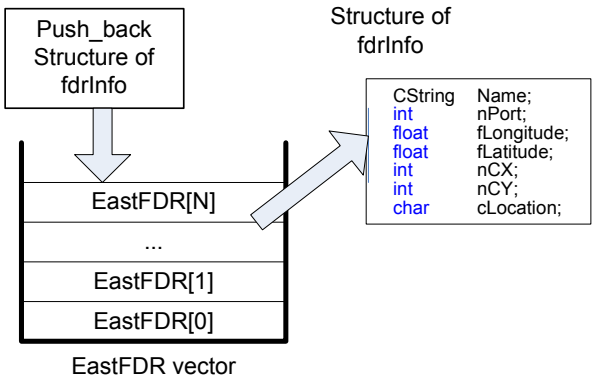

Figure 3.4 read FDR information

Next is to read FDC information into vector FDCInfo with FDCInfo structure: FDC name, region, city, state, zip, latitude, longitude.

```
struct FDCinfomation
{
        UINT Code;
        CString Name;
        CString Region;
        CString City;
        CString State;
        UINT Zip;
        double Latitude;
        double Longitude;
        UINT PSSEBus;
        double Nameplate;
        double Summer;
        double Winter;
};
```

Then read the event time information from EventTime.txt shown as Figure 3.5

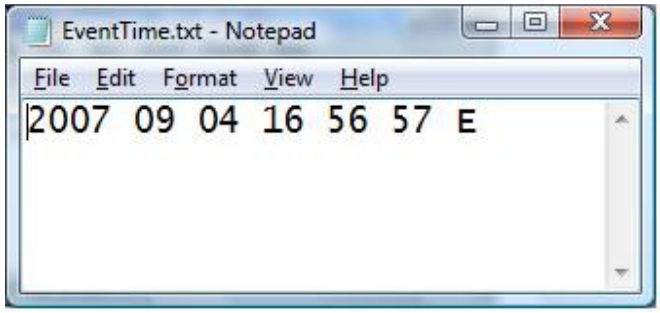

Figure 3.5 Event time file

```
void ReadEventInfor()
{
	FILE* hFile;
	char fname[100];
	COleDateTime datatime;

	sprintf(fname, "EventTime.txt");
	if ( access(fname, 0) != 0 )
	{
		printf("EventLocation.txt not exists.");
		return;
	}

	hFile = fopen(fname, "r");
	if (hFile == NULL)
	{
		printf("EventLoation.txt not exists.");
		fclose(hFile);
		return;
	}

	fscanf(hFile, "%d %d %d %d %d %d %c\n", &m_Year, &m_Mon, &m_Day, &m_Hour, &m_Min,
&m_Sec, &m_cLoc);
	datatime = COleDateTime(m_Year, m_Mon, m_Day, m_Hour, m_Min, m_Sec) +
COleDateTimeSpan(0,0,0,11);
	if ( datatime.GetMonth()<10 )
		cMon.Format(_T("%d%d"),0,datatime.GetMonth());
	else
		cMon.Format(_T("%d"),datatime.GetMonth());
```

```
	if ( datatime.GetDay()<10 )
		cDay.Format(_T("%d%d"),0,datatime.GetDay());
	else
		cDay.Format(_T("%d"),datatime.GetDay());

	if ( datatime.GetHour()<10 )
		cHour.Format(_T("%d%d"),0,datatime.GetHour());
	else
		cHour.Format(_T("%d"),datatime.GetHour());

	if ( datatime.GetMinute()<10 )
		cMin.Format(_T("%d%d"),0,datatime.GetMinute());
	else
		cMin.Format(_T("%d"),datatime.GetMinute());

	if ( datatime.GetSecond()<10 )
		cSec.Format(_T("%d%d"),0,datatime.GetSecond());
	else
		cSec.Format(_T("%d%d"),0,datatime.GetSecond());

	fclose(hFile);
}
```

After get these information, condition the data for triangulation algorithm. CheckDelay( ) is called.

```
void CheckDelay( TriangData &data, vector <CString> &DelayUnit  )
{
	UINT i;
	double sum;
	double aver;

	//if data is empty, then return
	if ( data.Data.size() <= 0 )
		return;

	sum = 0;
	for (i=0; i<data.Data.size(); i++)
```

```
        {
                sum += data.Data[i].TimeDelay;
        }

        //calculate the average of this unit's time delay
        aver = sum/data.Data.size();
        aver = int(aver+0.5)-11;

        for (i=0; i<data.Data.size(); i++)
        {
                data.Data[i].Dt += aver;
        }

        if ( aver > 9.5 && aver < 10.5 )
        {
                DelayUnit.push_back( data.FdrInformation.Name );
        }
};
```

This function checks the delayed units to fix the one-second problem. The one-second problem is that the data time of the abnormal FDR units is one second faster than the normal units due to error code in those FDR units. Normally, the time delay between the normal data time and the data receiving time of the FNET server is about 10.1 ~ 10.2 seconds. However, the time delay of abnormal units is about 11.1 ~ 11.2 seconds. Therefore, this function finds the delayed units by finding units whose average time delays are between 10.5 to 11.5 seconds. After finding the delayed units, this function fixes the problem by adding one second to the data time. The names of the delayed units are recorded to “DelayingUnits.txt.” TreatEventData( ) calls DataCondition( ) function.

```
void DataCondition(vector<TriangData> &data)
{
        vector<TriangData> tempData;
        TriangData fdrData;
        double minv, maxv;
        UINT i, j;
```

```
	tempData.clear();

	if ( data.size() == 1)
	{
		MovingMedian( data );
		return;
	}

	//calculate time scope
	minv = m_Mon*10000000.0 + m_Day*100000.0 + m_Hour*3600.0 + m_Min*60.0 + m_Sec - 18.0;
	maxv = m_Mon*10000000.0 + m_Day*100000.0 + m_Hour*3600.0 + m_Min*60.0 + m_Sec + 19.0;

	//get data in the time scope
	for( i=0; i<data.size(); i++)
	{
		fdrData.Data.clear();

		for ( j=0; j<data[i].Data.size(); j++ )
		{
			if ( data[i].Data[j].Dt > minv && data[i].Data[j].Dt < maxv )
				fdrData.Data.push_back(data[i].Data[j]);
		}

		if ( fdrData.Data.size() > 100) //if data points less than 100, then discard
		{
			fdrData.FdrInformation = data[i].FdrInformation;
			tempData.push_back( fdrData );
		}
	}

	data = tempData;

	MovingMedian( data );
}
```

This function makes the data of EventData in the same time scope such as from 15 seconds before the event to 18 seconds after the event. If there is no missing data, all the data have the same number of data. However, since it is possible that there are some

missing data, this function checks the number of data. If the number of data of a certain FDR is smaller than 100, the corresponding data is discarded. After that the data are filtered by 15-point moving median by MovingMedian( ).

```
void MovingMedian( vector<TriangData> &data )
{
	float val[5];
	UINT k, i, j;

	for ( k=0; k<data.size(); k++)
	{
		for ( i=0; i<data[k].Data.size()-5; i++)
		{

			for (j = 0; j < 5; j++)
			{
				val[j] = data[k].Data[i+j].FinalFreq;
			}
			combsort(val, 5);

			data[k].Data[i].FinalFreq = val[3];
		}
	}
}
```

To find the 15-point moving median, First, grab 15 data from the EventData vector (i.e. EventData[i].Data[j].FinalFrea where i and j are iteration indices) using a sliding window, and sort the data by combsort(), and then take the middle of the data. The code of combsort() function is as following:

```
void combsort(float *a, int amount)
{
	int gap=amount;
	bool swapped=false;

	while(gap>1 || swapped)
	{
		gap=(gap*10)/13;
```

```
                if(gap==9 || gap==10)
                        gap=11;
                if (gap<1)
                        gap=1;
                swapped=false;

                for ( int m=0; m<amount-gap; m++)
                {
                        int n=m+gap;
                        if (a[m]>a[n])
                        {
                                a[m]+=a[n];
                                a[n]=a[m]-a[n];
                                a[m]-=a[n];
                                swapped=true;
                        }
                }
        }
}
```

In triangulation part, first is to calculate mismatch power amount. As discussed above, use $trip\ amount = iMWPerHz \times \left| frequency\ difference\ before\ and\ after\ the\ event \right|$ to get the amount. When the event happens in WECC or East or TX, the *iMWPerHz* is 7866, 25375, and 4140 respectively.

```
 int  GetTripAmount(char cLoc, const double beforeFreq, const double &afterFreq )
{
        double iMWPerHz;

        switch( cLoc )
        {
        case 'W':
        case 'w':
                iMWPerHz = 7866;
                break;
        case 'E':
        case 'e':
                iMWPerHz = 25375;
```

```
                break;
        case 'T':
        case 't':
                iMWPerHz = 4140;
                break;
        }

        return int( fabs(beforeFreq-afterFreq)*iMWPerHz );

}
```

To deal with event location, as discussed in chapter 2, apply MATCOM least square error to minimize objective function J.

$$J = \sum_{i=1}^{4} [(x - x_i)^2 + (y - y_i)^2 - (s \cdot (t - t_i))^2]$$

here, x and y are the latitude and longitude of the event location,

xi and yi are the latitude and longitude of ith FDR location,

t is the starting time of the event,

ti is the starting time of the ith FDR,

s is the speed of frequency wave propagation

According to many test results, the optimal value X of J can satisfy the global minimization. However, to make it clearer, 10,000 points of the whole US are applied to the J, and check if there are other points that make J lower than the X. Then, output event information to “TriangRecord.txt” and “EventLocation.txt”.

3.3 Event Time Method

In event trigger module, event time is calculated at the first point whose df/dt exclude threshold. This method is easy but a little inaccurate. Here, a new accurate event time calculation is discussed. Through recorded data in FDRs, the time event occurred can be observed easily in visual [17]. But using program to find this accurate turning point is hard. I tried four methods to get event time, and found the easiest and accurate method [18].

1) Data method (recommended)

The most important characteristics, discriminating dropping section from pre-event and after-event sections, are more than four continuously declining points and the declining range is much larger. Due to these, I use data analysis method in 12-point window, whose first point is event time in the window reach my thresholds (the largest number of series of continuous declining points, at the same time, range larger than maximum-3mHz). Fig 3.6 below expresses the method.

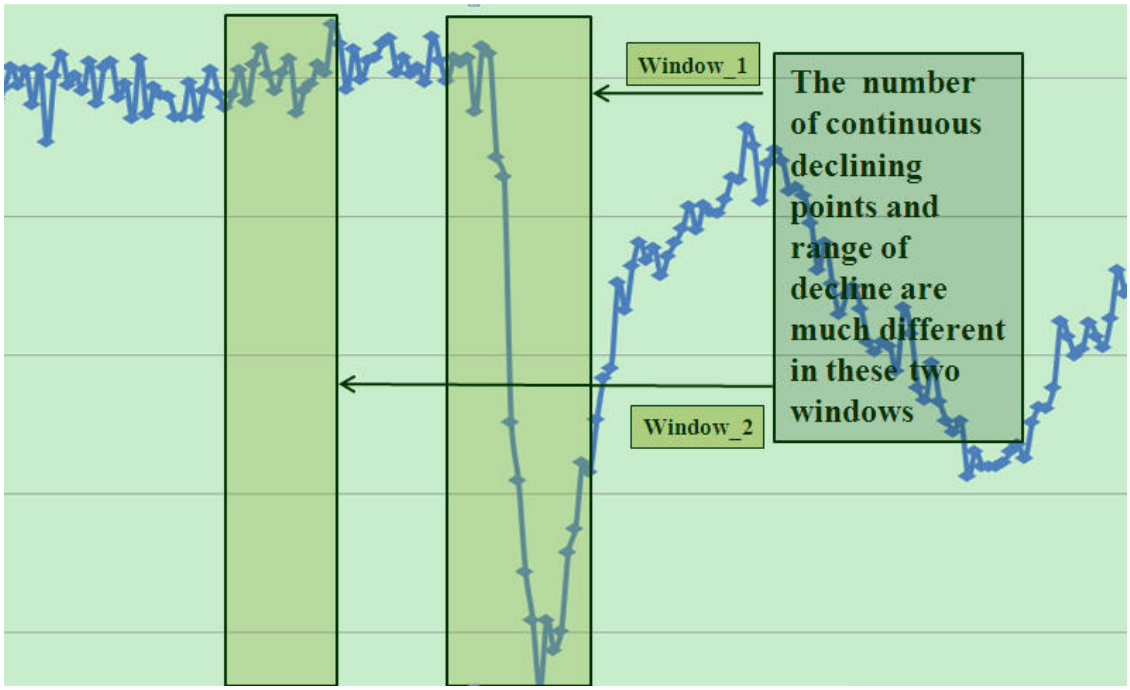

Figure 3.6 Data analysis method of finding event time

Using this easy method, author find the accurate event time of every case. And find there are some event times online are not accurate. Author list event time of all cases in table below and mark the inaccurate recorded cases in yellow.

Table 3.1 data analysis event time results table

All cases	Event time	Event time(recorded)
08.08.25	18;51;00.0	18;51;00
08.08.24	19;58;47.8	19;58;48
08.08.14	07;24;26.4	07;24;25
08.05.22	17;59;47.8	17;59;47

08.05.13	14;03;32.8	14;03;30
08.04.29	19;20;50.1	19;20;49
08.02.07	17;44;08.5	17;44;09
08.02.01	09;44;42.8	09;44;41
07.08.28	17;53;12.8	17;53;13
07.08.16	03;26;59.1	03;26;58
07.07.07	20;37;53.7	20;37;54
07.07.01	23;34;16.5	23;34;16
08.01.10	12;36;29.3	12;36;29
08.01.12	15;12;10.8	15;12;11
08.01.17	00;25;59.9	00;25;57
08.01.20	20;21;41.4	20;21;40
08.01.21	12;58;19.2	12;58;19
08.01.30	19;47;13.2	19;47;11
08.02.03	23;56;29.8	23;56;29
08.02.06	16;34;30.4	16;34;31
08.02.09	09;22;29.9	09;22;29
08.04.07	19;12;37.8	19;12;37
08.05.27	13;12;11.6	13;12;12

2) DF/DT method

DF/DT in dropping section and other sections are different. Due to this factor, use DF/DT to identify event time. However, there are some noises whose slopes are as steep as or even steeper than dropping section, which is the fate shortcoming of this method. In this method[19]:

First, do 30-point median moving, get DF/DT of this smoothed plot; second, find the decline point in DF/DT plot. Fig 3.7 below shows the principle of this method.

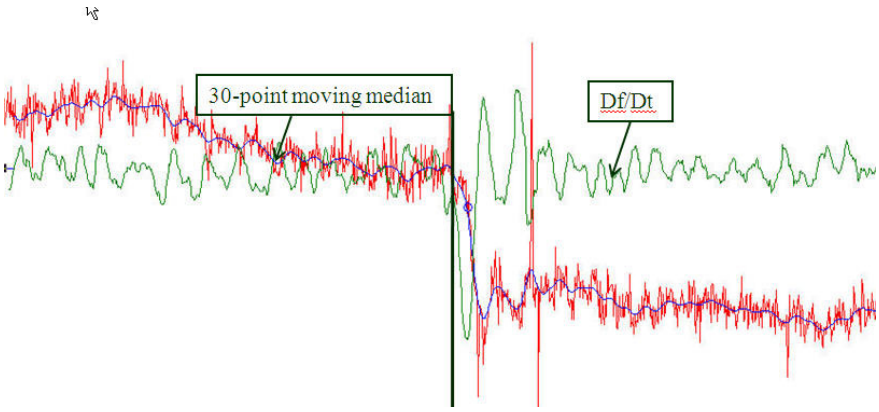

Figure 3.7 DF/DT method of finding event time

3.4 Demonstration

On-line Trigger DLL encapsulates the FNET on-line event detection, so that the algorithm can be used by other data source through calling the DLL file. The function of the On-line Trigger DLL includes missing date handling, completeness probing, data alignment and event trigger application.

C++ is used for developing the DLL and Lib files named 'ETrigNDiatry1.dll' and 'ETrigNDiatry1.dll' respectively. The outputs of the DLL file are two variables 'bEvent' and 'm_cEventData'. The 'bEvent' is a Boolean type to indicate whether there is an event or not. When the value of 'bEvent' is true, it means there is an event. And the

‘m_cEventData’ is a string providing the information of event data and time in ‘YYYY-MM-DD HH:MM:SS’ format.

Because the target application database has not been provided, a testing program is written based on the FNET Access database. The ‘ETrigNDiatry1.dll’ and ‘ETrigNDiatry1.lib’ are included in the folder of testing program, and the lib file path should be added into the additional lib source. By the following variable statements, ‘bEvent’ and ‘m_cEventData’ can be exported from the DLL file.

__declspec (dllimport) CString m_cEventData;

__declspec (dllimport) bool bEvent;

Fig. 3.8 shows the structure and result of testing program. As can been seen, the testing program can obtain the event trigger information by calling the On-line Trigger DLL. Since the DLL file is designed to be used by target application, the data format of target application database is required. Due to the fact that this DLL is developed on the Visual 2008 platform, the Redistributable Package of Visual 2008 is required in order for the files to run successfully [20].

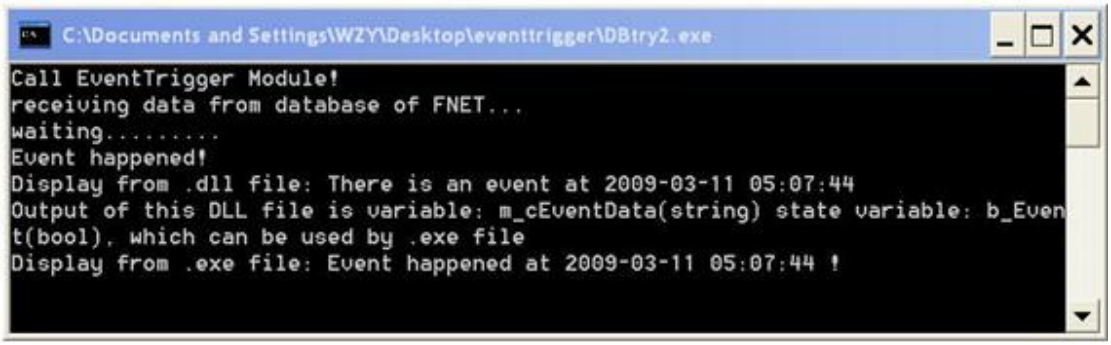

Figure 3.8 On-line Trigger DLL Testing Program

The LOD Application DLL encapsulates the FNET Triangulation Algorithm for estimating event location, so that the algorithm can be used by other data source through calling DLL file.

C++ is used for developing the DLL and Lib files named 'triaDLLtry.dll' and 'triaDLLtry.lib' respectively. The outputs of the DLL file are the LOD information. It includes: unit information (g_csUnits, g_csLocation), estimated event location (g_csFDC, g_csRegion, g_csCity, g_csState, g_csZip, g_fLatitude, g_fLongitude), event information (g_iTripAmount, g_fSpeed) and event time (g_fEvTime).

Because the target application database format has not been provided, a testing program is written based on the FNET text event data file [21]. The 'triaDLLtry.dll' and 'triaDLLtry.lib' are included in the folder of the testing program. The lib file path should be added into the additional lib source. The output variable statements are the same as those of the On-line Trigger DLL. And all the output variables are saved into a text file named 'TriangRecord.txt'.

Fig. 3.8 shows the structure and saved text file of the testing program. As can been seen, the testing program can obtain the LOD information by calling the On-line Trigger DLL. The data format of target application event file is also needed in this part. One of the other options can be converting the target application event files into FDR event data text file. And the Redistributable Package of Visual 2008 is also required.

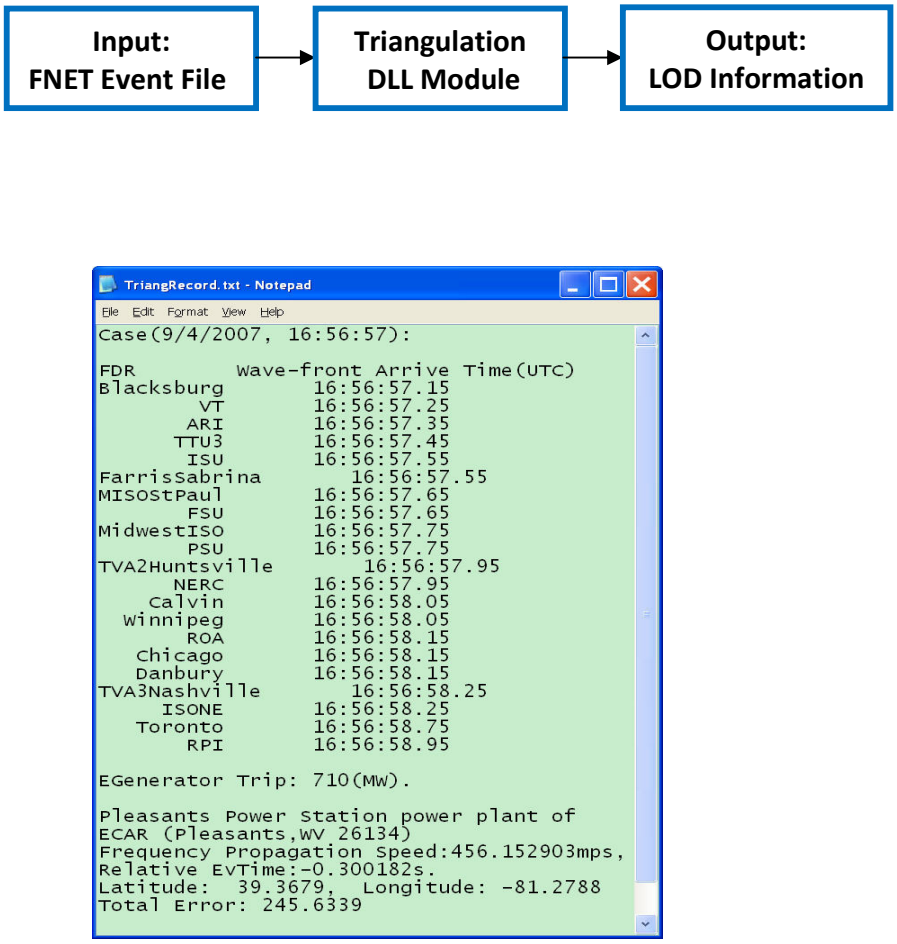

Figure 3.9 Testing Program Structure and Output Text File

Chapter 4 Power system model and test systems

4.1 Introduction

There four methods of analysis the power system: static flow analysis, quasi-static analysis, small signal analysis and time domain simulation [22-24]. Static flow analysis of the system simplified the synchronous motor, load, and other devices to simple nodes (PQ, PV and balance of nodes), the calculation involves only the solution of algebraic equations, is the basis for power system analysis. Quasi Steady-State or QSS analysis is an integrity model of system simulation method which considered a variety of dynamic components and the static model, but ignored the transient process. Small-Signal Stability Analysis or SSSA, is the linearization power system differential-algebraic equations (DAE) in the current operating, by calculating the system state matrix eigenvalues and eigenvectors to analyze the stability of the current state of the system [24]. Time-domain simulation analysis was used to study the dynamic problems, such as first swing power angle stability. It usually uses common equipment or model of the system, including the synchronous motor, turbine, dynamic load, HVDC, FACTS devices and all other fast-action devices. It takes from the tens of microseconds to ten seconds or even longer to solve these models with the corresponding algorithm [25].

This part will focus on the bifurcation phenomenon of power system, including saddle-node bifurcation and Hopf bifurcation modes, because they are in very important in a variety of stability of a system. After nearly 20 years of research, these two bifurcations modes have been accepted by power electrical filed. Study saddle-node bifurcation in system analysis, commonly starts from the quasi-static (QSS) method; whereas the dynamic bifurcation needs small-signal stability analysis [26]. This part will discuss the stability analysis for the dynamic model of power system components, and

formation of the whole network on a linear model for small signal stability analysis base on bifurcation analysis result.

4.2 The mathematical model of power system components

Before the analysis, here gives out the static and dynamic models of the power system components including: generators, AVR, PSS, SVC.

4.2.1 Generator Model

Two-axis generator model:

$$\begin{cases} \dot{\delta}_i = \omega_0(\omega_i - 1) \\ M_i\dot{\omega}_i = P_{mi} - (E'_{qi} - x'_{di}I_{di})I_{qi} - (E'_{di} + x'_{qi}I_{qi})I_{di} - D_i(\omega_i - 1) \\ T'_{d0i}\dot{E}'_{qi} = E_{fdi} - [E'_{qi} + (x_{di} - x'_{di})I_{di}] \\ T'_{q0i}\dot{E}'_{di} = -E'_{di} + (x_{qi} - x'_{qi})I_{qi} \end{cases} \tag{4.1}$$

Generator power equation:

$$\begin{cases} P_{gi} - V_iI_{di}\sin(\delta_i - \theta_i) - V_iI_{qi}\cos(\delta_i - \theta_i) = 0 \\ Q_{gi} - V_iI_{di}\cos(\delta_i - \theta_i) + V_iI_{qi}\sin(\delta_i - \theta_i) = 0 \end{cases} \tag{4.2}$$

Stator equation:

$$\begin{cases} E'_{qi} - V_i\cos(\delta_i - \theta_i) - r_{si}I_{qi} - x'_{di}I_{di} = 0 \\ E'_{di} - V_i\sin(\delta_i - \theta_i) - r_{si}I_{di} + x'_{qi}I_{qi} = 0 \end{cases} \tag{4.3}$$

4.2.2 Excitation System Model

Third-order excitation model:

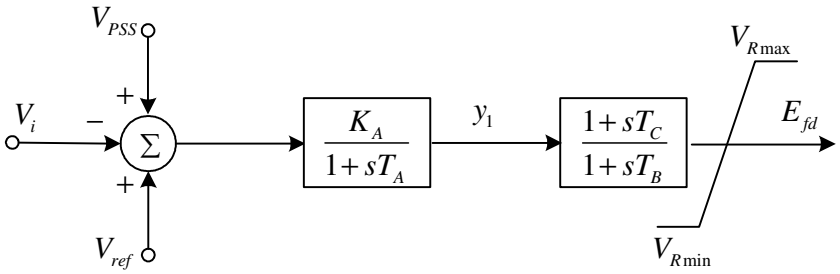

Figure 4.1Third-order model excitation system

$$\begin{cases} T_A \dot{y}_1 = -y_1 + K_A(V_{ref} - V_t + V_{PSS}) \\ T_B \dot{E}_{fd} = -E_{fd} + \dfrac{T_A - T_C}{T_A} y_1 + \dfrac{K_A T_C}{T_A}(V_{ref} - V_t + V_{PSS}) \end{cases} \tag{4.4}$$

In general PSS not installed case; we have $T_B = T_C = 0$ to simplifier the three order model as a first-order model.

$$T_A \dot{E}_{fd} = -E_{fd} + K_A(V_{ref} - V_i) \tag{4.5}$$

4.2.3 Model of Power System Stabilizer (PSS)

Here PSS uses speed deviation as input, the model shown in Figure 4.2:

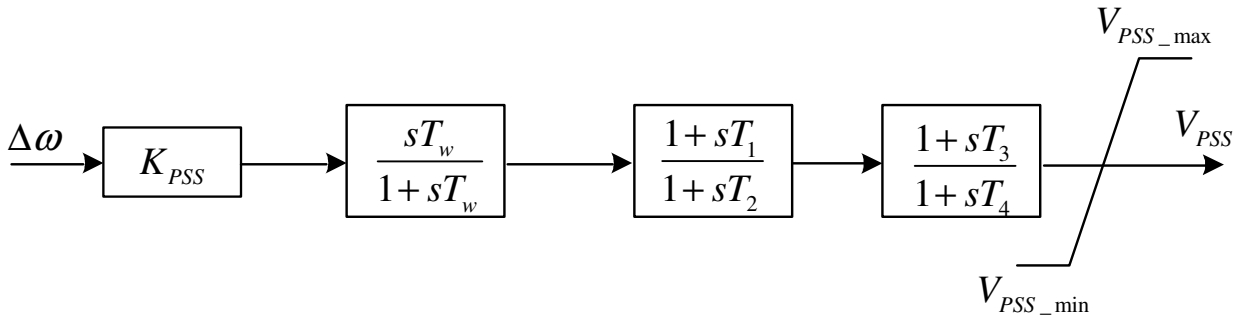

Figure 4.2 PSS model

For the convenient, we can use the PSS transfer function diagram instead the model diagram (Fig.4.2). The equivalence diagram is shown in Figure 4.3:

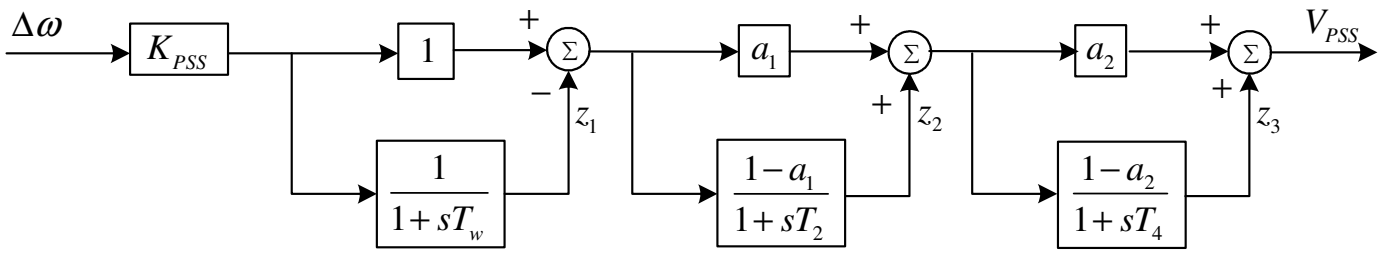

Figure 4.3 *the equivalent transfer function diagram of PPS*

Equations of state 4.6:

$$\begin{cases} T_w \dot{z}_1 = -z_1 + K_{PSS}\Delta\omega \\ T_2 \dot{z}_2 = -(1-a_1)z_1 - z_2 + K_{PSS}(1-a_1)\Delta\omega \\ T_4 \dot{z}_3 = -a_1(1-a_2)z_1 + (1-a_2)z_2 - z_3 + K_{PSS}a_1(1-a_2)\Delta\omega \end{cases} \tag{4.6}$$

Here, $a_1 = T_1 / T_2, a_2 = T_3 / T_4$, we have output of PSS V_{PSS} can be expressed as (Note: V_{PSS} is variable, no longer serviced as a state of quantity; it is the bridge connecting AVR to the PSS):

$$V_{PSS} = -a_1 a_2 z_1 + a_2 z_2 + z_3 + K_{PSS} a_1 a_2 \Delta\omega \tag{4.7}$$

4.2.4 Load Models

To build a good load model is a very difficult topic. In general, the load model can be divided into two kinds: static and dynamic models. Since our research field is in the small signal stability, which does not involve dynamic voltage stability, so we use a static load model here.

Index load model:

$$\begin{cases} P = P_0 (\dfrac{V}{V_0})^a \\ Q = Q_0 (\dfrac{V}{V_0})^b \end{cases} \tag{4.8}$$

Here P_0, Q_0, reference the active and reactive power of load voltage.

When $a = b = 0$， express 3.8 is constant power model； when $a = b = 1$， express is constant current model； when $a = b = 2$ it is impedance model. In this thesis we use the simplest constant power model.

4.2.5 Network Equations

The network equations are the various components in a network and the grid interface, constraint equations as follows:

$$\begin{cases} P_{gi} - P_{li} - V_i \sum_{j=1}^{nbus} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) = 0 \\ Q_{gi} - Q_{li} - V_i \sum_{j=1}^{nbus} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) = 0 \end{cases} \tag{4.9}$$

4.3 The theory of nonlinear systems

4.3.1 Whole system model

Based on the above models of power system components, the whole power system can be described as a set of differential algebraic equation (DAE):

$$\begin{cases} \dot{x} = f(x, y, \mu, p) \\ 0 = g(x, y, \mu, p) \end{cases} \tag{4.10}$$

Where $x \in R^n$ is the state variables related generators, load, and other controllers systems; $y \in R^m$ is overlooked the fast dynamics algebraic variables; μ is a not controllable variables, such as of the load factor; $p \in R^k$ is the controllable variables, such as reference voltages of AVR

In general, the power system was running under equilibrium conditions, or at a balance. According to differential dynamical system, power system equilibrium points (x_0, y_0, μ_0, p_0) can be found by：

$$\begin{cases} 0 = f(x, y, \mu, p) \\ 0 = g(x, y, \mu, p) \end{cases} \tag{4.11}$$

In other words can be decided by algebraic equations group (4.1）～（4.11） (differential equations with a zero at the left can be transfer into algebraic equations). When use eigenvalue analysis method to study the small signal stability, linearizae the equation (3.11) in the equilibrium point, available:

$$\begin{bmatrix} \Delta\dot{x} \\ 0 \end{bmatrix} = \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{4.12}$$

Here $A_{tot} = \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix}$ we got the whole system Jacobian matrix.

Inside, $f_x = \partial f / \partial x\big|_0$, $f_y = \partial f / \partial y\big|_0$, $g_x = \partial g / \partial x\big|_0$, $g_y = \partial g / \partial y\big|_0$.

If g_y is not singular, we can eliminate algebraic variables Δy, then (4.12) describes the DAE system can be simplified to the following ODE system：

$$\Delta\dot{x} = (f_x - f_y g_y^{-1} g_x)\Delta x = A_{sys}\Delta x \tag{4.13}$$

Here, $A_{sys} = f_x - f_y g_y^{-1} g_x$ is the state matrix of a stability model of power system with small perturbation. The following describes the detail of how to linerarzie entire power sytem.

4.3.2 Power System Linearziation Model

The general steps to analysis power system with a small signal perturbation are:

1. Obtain the system steady-state equilibrium point;

2. Linearize the power system model in the equilibrium point;
3. Exam the stability of the system with a small disturbance through the state matrix eigenvalue analysis.

We can see that the power system linearzation is the basis of power system small signal disturbance stability analysis. There are lots of methods can linearzate models (state matrix), in this thesis, we use whole system Jacobian matrix to obtain the state matrix [27]. Because this method has many advantages [28]:

1. Clear. Use differential-algebraic equations to calculate out A_{tot}, donot need consider adopting the coordinate transformation and the network interface.
2. Scalable. It is easy to add new components, the model changes and updates are very convenient, so it is easy to prepare large-scale analysis program.

We already described the power system components in the differential algebraic equations; here we formed the system Jacobian matrix A_{tot}, including four-order model generator, AVR three-order model.

$$
\begin{bmatrix}
\Delta\dot{\delta}_i \\ \Delta\dot{\omega}_i \\ \Delta\dot{E}'_{qi} \\ \Delta\dot{E}'_{di} \\ \cdots \\ \Delta\dot{y}_{1i} \\ \Delta\dot{E}_{fdi} \\ \cdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}
=
\begin{bmatrix}
0 & \frac{\partial\dot{\delta}_i}{\partial\omega_i} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\partial\dot{\omega}_i}{\partial\omega_i} & \frac{\partial\dot{\omega}_i}{\partial E'_{qi}} & \frac{\partial\dot{\omega}_i}{\partial E'_{di}} & \cdots & 0 & 0 & \cdots & \frac{\partial\dot{\omega}_i}{\partial I_{di}} & \frac{\partial\dot{\omega}_i}{\partial I_{qi}} & 0 & 0 & 0 & 0 & \frac{\partial\dot{\omega}_i}{\partial P_{m\_swing}} \\
0 & 0 & \frac{\partial\dot{E}'_{qi}}{\partial E'_{qi}} & 0 & \cdots & & \frac{\partial\dot{E}'_{qi}}{\partial E_{fdi}} & \cdots & \frac{\partial\dot{E}'_{qi}}{\partial I_{di}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial\dot{E}'_{di}}{\partial E'_{di}} & \cdots & 0 & 0 & \cdots & 0 & \frac{\partial\dot{E}'_{di}}{\partial I_{qi}} & 0 & 0 & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & \frac{\partial\dot{y}_{1i}}{\partial\omega_i} & 0 & 0 & \cdots & \frac{\partial\dot{y}_{1i}}{\partial y_{1i}} & 0 & \cdots & 0 & 0 & 0 & 0 & \frac{\partial\dot{y}_{1i}}{\partial V_i} & 0 & 0 \\
0 & \frac{\partial\dot{E}_{fdi}}{\partial\omega_i} & 0 & 0 & \cdots & \frac{\partial\dot{E}_{fdi}}{\partial y_{1i}} & \frac{\partial\dot{E}_{fdi}}{\partial E_{fdi}} & \cdots & 0 & 0 & 0 & 0 & \frac{\partial\dot{E}_{fdi}}{\partial V_i} & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{\partial}{\partial\delta_i} & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & \frac{\partial}{\partial P_{gi}} & 0 & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & 0 & \frac{\partial}{\partial Q_{gi}} & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & \frac{\partial}{\partial E'_{qi}} & 0 & \cdots & 0 & 0 & \cdots & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & 0 & 0 & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & 0 & \frac{\partial}{\partial E'_{di}} & \cdots & 0 & 0 & \cdots & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & 0 & 0 & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \frac{\partial}{\partial P_{gi}} & 0 & \frac{\partial}{\partial V_{i/j}} & \frac{\partial}{\partial\theta_{i/j}} & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \frac{\partial}{\partial Q_{gi}} & \frac{\partial}{\partial V_{i/j}} & \frac{\partial}{\partial\theta_{i/j}} & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\Delta\delta_i \\ \Delta\omega_i \\ \Delta E'_{qi} \\ \Delta E'_{di} \\ \cdots \\ \Delta y_{1i} \\ \Delta E_{fdi} \\ \cdots \\ \Delta I_{di} \\ \Delta I_{qi} \\ \Delta P_{gi} \\ \Delta Q_{gi} \\ \Delta V_i \\ \Delta\theta_i \\ \Delta P_{m\_swing}
\end{bmatrix}
$$

Recalculate Jacobian matrix with power system stabilizer model considered, we get a new A_{tot} as shown in equation 4.14.

$$
\begin{bmatrix}
\Delta\dot{\delta}_i \\ \Delta\dot{\omega}_i \\ \Delta\dot{E}'_{qi} \\ \Delta\dot{E}'_{di} \\ \cdots \\ \Delta\dot{y}_{1i} \\ \Delta\dot{E}_{fdi} \\ \cdots \\ \Delta\dot{z}_{1i} \\ \Delta\dot{z}_{2i} \\ \Delta\dot{z}_{3i} \\ \cdots \\ \Delta\dot{B}_{SVCi} \\ \Delta\dot{x}_{1i} \\ \Delta\dot{x}_{2i} \\ \cdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}
=
\begin{bmatrix}
0 & \frac{\partial\dot{\delta}_i}{\partial\omega_i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\partial\dot{\omega}_i}{\partial\omega_i} & \frac{\partial\dot{\omega}_i}{\partial E'_{qi}} & \frac{\partial\dot{\omega}_i}{\partial E'_{di}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{\omega}_i}{\partial I_{di}} & \frac{\partial\dot{\omega}_i}{\partial I_{qi}} & 0 & 0 & 0 & 0 & \frac{\partial\dot{\omega}_i}{\partial P_{m\_swing}} \\
0 & 0 & \frac{\partial\dot{E}'_{qi}}{\partial E'_{qi}} & 0 & 0 & \frac{\partial\dot{E}'_{qi}}{\partial E_{fdi}} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{E}'_{qi}}{\partial I_{di}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial\dot{E}'_{di}}{\partial E'_{di}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{E}'_{di}}{\partial I_{qi}} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\partial\dot{y}_{1i}}{\partial\omega_i} & 0 & 0 & \frac{\partial\dot{y}_{1i}}{\partial y_{1i}} & 0 & \frac{\partial\dot{y}_{1i}}{\partial z_{1i}} & \frac{\partial\dot{y}_{1i}}{\partial z_{2i}} & \frac{\partial\dot{y}_{1i}}{\partial z_{3i}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{y}_{1i}}{\partial V_i} & 0 & 0 \\
0 & \frac{\partial\dot{E}_{fdi}}{\partial\omega_i} & 0 & 0 & \frac{\partial\dot{E}_{fdi}}{\partial y_{1i}} & \frac{\partial\dot{E}_{fdi}}{\partial E_{fdi}} & \frac{\partial\dot{E}_{fdi}}{\partial z_{1i}} & \frac{\partial\dot{E}_{fdi}}{\partial z_{2i}} & \frac{\partial\dot{E}_{fdi}}{\partial z_{3i}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{E}_{fdi}}{\partial V_i} & 0 & 0 \\
0 & \frac{\partial\dot{z}_{1i}}{\partial\omega_i} & 0 & 0 & 0 & 0 & \frac{\partial\dot{z}_{1i}}{\partial z_{1i}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\partial\dot{z}_{2i}}{\partial\omega_i} & 0 & 0 & 0 & 0 & \frac{\partial\dot{z}_{2i}}{\partial z_{1i}} & \frac{\partial\dot{z}_{2i}}{\partial z_{2i}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\partial\dot{z}_{3i}}{\partial\omega_i} & 0 & 0 & 0 & 0 & \frac{\partial\dot{z}_{3i}}{\partial z_{1i}} & \frac{\partial\dot{z}_{3i}}{\partial z_{2i}} & \frac{\partial\dot{z}_{3i}}{\partial z_{3i}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{B}_{SVCi}}{\partial B_{SVCi}} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{B}_{SVCi}}{\partial V_{i/j}} & \frac{\partial\dot{B}_{SVCi}}{\partial\theta_{i/j}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{x}_{1i}}{\partial x_{1i}} & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{x}_{1i}}{\partial V_{i/j}} & \frac{\partial\dot{x}_{1i}}{\partial\theta_{i/j}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\dot{x}_{2i}}{\partial x_{1i}} & \frac{\partial\dot{x}_{2i}}{\partial x_{2i}} & 0 & 0 & 0 & 0 & \frac{\partial\dot{x}_{2i}}{\partial V_{i/j}} & \frac{\partial\dot{x}_{2i}}{\partial\theta_{i/j}} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & \frac{\partial}{\partial P_{gi}} & 0 & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & 0 & \frac{\partial}{\partial Q_{gi}} & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & \frac{\partial}{\partial E'_{qi}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & 0 & 0 & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
\frac{\partial}{\partial\delta_i} & 0 & 0 & \frac{\partial}{\partial E'_{di}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial}{\partial I_{di}} & \frac{\partial}{\partial I_{qi}} & 0 & 0 & \frac{\partial}{\partial V_i} & \frac{\partial}{\partial\theta_i} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial}{\partial P_{gi}} & 0 & \frac{\partial}{\partial V_{i/j}} & \frac{\partial}{\partial\theta_{i/j}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial}{\partial Q_{gi}} & \frac{\partial}{\partial V_{i/j}} & \frac{\partial}{\partial\theta_{i/j}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\Delta\delta_i \\ \Delta\omega_i \\ \Delta E'_{qi} \\ \Delta E'_{di} \\ \cdots \\ \Delta y_{1i} \\ \Delta E_{fdi} \\ \cdots \\ \Delta z_{1i} \\ \Delta z_{2i} \\ \Delta z_{3i} \\ \cdots \\ \Delta B_{SVCi} \\ \Delta x_{1i} \\ \Delta x_{2i} \\ \cdots \\ \Delta I_{di} \\ \Delta I_{qi} \\ \Delta P_{gi} \\ \Delta Q_{gi} \\ \Delta V_i \\ \Delta\theta_i \\ \Delta P_{m\_swing}
\end{bmatrix}
\tag{4.14}
$$

Variables are listed here:

- Generator part：

$$\Delta\delta_i = [\Delta\delta_1 \Delta\delta_2 \cdots \Delta\delta_{nmac}]^T, \Delta\omega_i = [\Delta\omega_1 \Delta\omega_2 \cdots \Delta\omega_{nmac}]^T,$$
$$\Delta E'_{qi} = [\Delta E'_{q1} \Delta E'_{q2} \cdots \Delta E'_{qnmac}]^T, \Delta E'_{di} = [\Delta E'_{d1} \Delta E'_{d2} \cdots \Delta E'_{dnmac}]^T$$

- AVR part： $\Delta y_{1i} = [\Delta y_{11} \Delta y_{12} \cdots \Delta y_{1nmac}]^T, \Delta E_{fdi} = [\Delta E_{fd1} \Delta E_{fd2} \cdots \Delta E_{fdnmac}]^T$
- PSS part:

$$\Delta z_{1i} = [\Delta z_{11} \Delta z_{12} \cdots \Delta z_{1nPSS}]^T, \Delta z_{2i} = [\Delta z_{21} \Delta z_{22} \cdots \Delta z_{2nPSS}]^T, \Delta z_{3i} = [\Delta z_{31} \Delta z_{32} \cdots \Delta z_{3nPSS}]^T$$

Other algebraic variables：

$$\Delta I_{di} = [\Delta I_{d1} \Delta I_{d2} \cdots \Delta I_{dnmac}]^T, \Delta I_{qi} = [\Delta I_{q1} \Delta I_{q2} \cdots \Delta I_{qnmac}]^T,$$
$$\Delta P_{gi} = [\Delta P_{g1} \Delta P_{g2} \cdots \Delta P_{gnmac}]^T, \Delta Q_{gi} = [\Delta Q_{g1} \Delta Q_{g2} \cdots \Delta Q_{gnmac}]^T,$$
$$\Delta V_i = [\Delta V_1 \Delta V_2 \cdots \Delta V_{nbus}]^T, \Delta \theta_i = [\Delta \theta_1 \Delta \theta_2 \cdots \Delta \theta_{nbus}]^T, \Delta P_{m\_swing} = [\Delta P_{m\_swing}]^T.$$

According to equation (4.1) to (4.13), we can give out the expression of each element in a system-wide Jacobian matrix A_{tot}:

- State variable δ

$$\frac{\partial \dot{\delta}_i}{\partial \omega_i} = \omega_0 = 100\pi$$

- State variable ω

$$\frac{\partial \dot{\omega}_i}{\partial \omega_i} = \frac{1}{M_i}[-D_i], \frac{\partial \dot{\omega}_i}{\partial E'_{qi}} = \frac{1}{M_i}[-I_{qi}], \frac{\partial \dot{\omega}_i}{\partial E'_{di}} = \frac{1}{M_i}[-I_{di}],$$
$$\frac{\partial \dot{\omega}_i}{\partial I_{di}} = \frac{1}{M_i}[-E'_{di} + (x'_{di} - x'_{qi}) I_{qi}], \frac{\partial \dot{\omega}_i}{\partial I_{qi}} = \frac{1}{M_i}[-E'_{qi} + (x'_{di} - x'_{qi}) I_{di}]$$

If I is a generator in a balance node， we need increase one element：

$$\frac{\partial \dot{\omega}_{swing}}{\partial P_{m\_swing}} = \frac{1}{M_{swing}}$$

- State variable E'_q

$$\frac{\partial \dot{E}'_{qi}}{\partial E'_{qi}} = -\frac{1}{T'_{d0i}}, \frac{\partial \dot{E}'_{qi}}{\partial E_{fdi}} = \frac{1}{T'_{d0i}}, \frac{\partial \dot{E}'_{qi}}{\partial I_{di}} = \frac{1}{T'_{d0i}}[-(x_{di} - x'_{di})]$$

- State variable E'_d

$$\frac{\partial \dot{E}'_{di}}{\partial E'_{di}} = -\frac{1}{T'_{q0i}}, \frac{\partial \dot{E}'_{di}}{\partial I_{qi}} = \frac{1}{T'_{q0i}}[x_{qi} - x'_{qi}]$$

- State variable y_1 (When the AVR is the third-order model)

$$\frac{\partial \dot{y}_{1i}}{\partial \omega_i} = \frac{K_{Ai}}{T_{Ai}}[a_{1i}a_{2i}K_{PSSi}], \frac{\partial \dot{y}_{1i}}{\partial y_{1i}} = -\frac{1}{T_{Ai}},$$
$$\frac{\partial \dot{y}_{1i}}{\partial z_{1i}} = -\frac{K_{Ai}}{T_{Ai}}[a_{1i}a_{2i}], \frac{\partial \dot{y}_{1i}}{\partial z_{2i}} = \frac{K_{Ai}}{T_{Ai}}[a_{2i}], \frac{\partial \dot{y}_{1i}}{\partial z_{3i}} = \frac{K_{Ai}}{T_{Ai}}, \frac{\partial \dot{y}_{1i}}{\partial V_i} = -\frac{K_{Ai}}{T_{Ai}}$$

Here $a_1 = T_1 / T_2, a_2 = T_3 / T_4$ are time constants of PSS.

- State variable E_{fd}

Without PSS, AVR uses first-order model：

$$\frac{\partial \dot{E}_{fdi}}{\partial E_{fdi}} = -\frac{1}{T_{Ai}}, \frac{\partial \dot{E}_{fdi}}{\partial V_i} = -\frac{K_{Ai}}{T_{Ai}}$$

With PSS, AVR uses three-order model：

$$\frac{\partial \dot{E}_{fdi}}{\partial \omega_i} = \frac{K_{Ai}T_{Ci}}{T_{Ai}T_{Bi}}[a_{1i}a_{2i}K_{PSSi}], \frac{\partial \dot{E}_{fdi}}{\partial y_{1i}} = \frac{T_{Ai} - T_{Ci}}{T_{Ai}T_{Bi}}, \frac{\partial \dot{E}_{fdi}}{\partial E_{fdi}} = -\frac{1}{T_{Bi}},$$
$$\frac{\partial \dot{E}_{fdi}}{\partial z_{1i}} = -\frac{K_{Ai}T_{Ci}}{T_{Ai}T_{Bi}}[a_{1i}a_{2i}], \frac{\partial \dot{y}_{1i}}{\partial z_{2i}} = \frac{K_{Ai}T_{Ci}}{T_{Ai}T_{Bi}}[a_{2i}], \frac{\partial \dot{E}_{fdi}}{\partial z_{3i}} = \frac{K_{Ai}T_{Ci}}{T_{Ai}T_{Bi}}, \frac{\partial \dot{E}_{fdi}}{\partial V_i} = -\frac{K_{Ai}T_{Ci}}{T_{Ai}T_{Bi}}$$

Here $a_1 = T_1 / T_2, a_2 = T_3 / T_4$, are time constants of PSS.

- State variable z_1

$$\frac{\partial \dot{z}_{1i}}{\partial \omega_i} = \frac{K_{PSSi}}{T_{wi}}, \frac{\partial \dot{z}_{1i}}{\partial z_{1i}} = -\frac{1}{T_{wi}}$$

- State variable z_2

$$\frac{\partial \dot{z}_{2i}}{\partial \omega_i} = \frac{K_{PSSi}}{T_{2i}}[1-a_{1i}], \frac{\partial \dot{z}_{2i}}{\partial z_{1i}} = -\frac{1}{T_{2i}}[1-a_{1i}], \frac{\partial \dot{z}_{2i}}{\partial z_{2i}} = -\frac{1}{T_{2i}}$$

- State variable z_3

$$\frac{\partial \dot{z}_{3i}}{\partial \omega_i} = \frac{K_{PSSi}}{T_{4i}}[a_{1i}(1-a_{2i})], \frac{\partial \dot{z}_{3i}}{\partial z_{1i}} = -\frac{1}{T_{4i}}[a_{1i}(1-a_{2i})], \frac{\partial \dot{z}_{3i}}{\partial z_{2i}} = \frac{1}{T_{4i}}[1-a_{2i}], \frac{\partial \dot{z}_{3i}}{\partial z_{3i}} = -\frac{1}{T_{4i}}$$

here， $a_1 = T_1 / T_2, a_2 = T_3 / T_4$， are time constants of PSS.

Active power equation of generator

$$P_{gi} - V_i I_{di} \sin(\delta_i - \theta_i) - V_i I_{qi} \cos(\delta_i - \theta_i) = 0$$

The partial differential of generator active power equation

$$\frac{\partial}{\partial \delta_i} = -V_i I_{di} \cos(\delta_i - \theta_i) + V_i I_{qi} \sin(\delta_i - \theta_i),$$

$$\frac{\partial}{\partial I_{di}} = -V_i \sin(\delta_i - \theta_i), \frac{\partial}{\partial I_{qi}} = -V_i \cos(\delta_i - \theta_i), \frac{\partial}{\partial P_{gi}} = 1,$$

$$\frac{\partial}{\partial V_i} = -I_{di} \sin(\delta_i - \theta_i) - I_{qi} \cos(\delta_i - \theta_i), \frac{\partial}{\partial \theta_i} = V_i I_{di} \cos(\delta_i - \theta_i) - V_i I_{qi} \sin(\delta_i - \theta_i)$$

Generator reactive power equation

$$Q_{gi} - V_i I_{di} \cos(\delta_i - \theta_i) + V_i I_{qi} \sin(\delta_i - \theta_i) = 0$$

The partial differential of generator reactive power equation

$$\frac{\partial}{\partial \delta_i} = V_i I_{di} \sin(\delta_i - \theta_i) + V_i I_{qi} \cos(\delta_i - \theta_i),$$

$$\frac{\partial}{\partial I_{di}} = -V_i \cos(\delta_i - \theta_i), \frac{\partial}{\partial I_{qi}} = V_i \sin(\delta_i - \theta_i), \frac{\partial}{\partial Q_{gi}} = 1,$$

$$\frac{\partial}{\partial V_i} = -I_{di} \cos(\delta_i - \theta_i) + I_{qi} \sin(\delta_i - \theta_i), \frac{\partial}{\partial \theta_i} = -V_i I_{di} \sin(\delta_i - \theta_i) - V_i I_{qi} \cos(\delta_i - \theta_i)$$

Stator voltage equation

$$E'_{qi} - V_i \cos(\delta_i - \theta_i) - r_{si} I_{qi} - x'_{di} I_{di} = 0 \tag{4.15}$$

The partial differential of Stator voltage equation

$$\frac{\partial}{\partial \delta_i} = V_i \sin(\delta_i - \theta_i), \frac{\partial}{\partial E'_{qi}} = 1, \frac{\partial}{\partial I_{di}} = -x'_{di}, \frac{\partial}{\partial I_{qi}} = -r_{si},$$
$$\frac{\partial}{\partial V_i} = -\cos(\delta_i - \theta_i), \frac{\partial}{\partial \theta_i} = -V_i \sin(\delta_i - \theta_i)$$

Another stator voltage equation

$$E'_{di} - V_i \sin(\delta_i - \theta_i) - r_{si} I_{di} + x'_{qi} I_{qi} = 0$$

The partial differential of above equation

$$\frac{\partial}{\partial \delta_i} = -V_i \cos(\delta_i - \theta_i), \frac{\partial}{\partial E'_{di}} = 1, \frac{\partial}{\partial I_{di}} = -r_{si}, \frac{\partial}{\partial I_{qi}} = x'_{qi},$$
$$\frac{\partial}{\partial V_i} = -\sin(\delta_i - \theta_i), \frac{\partial}{\partial \theta_i} = V_i \cos(\delta_i - \theta_i) \tag{4.16}$$

Network equation

$$P_{gi} - P_{li} - V_i \sum_{j=1}^{nbus} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) = 0 \tag{4.17}$$

The partial differential of above equation

$$\frac{\partial}{\partial P_{gi}} = 1,$$
$$\frac{\partial}{\partial V_i} = -\sum_{j \neq i}^{nbus} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) - 2V_i G_{ii}, \frac{\partial}{\partial V_j} = -V_i (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}),$$
$$\frac{\partial}{\partial \theta_i} = V_i \sum_{j=1}^{nbus} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}), \frac{\partial}{\partial \theta_j} = -V_i V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij})$$

Another network equation

$$Q_{gi} - Q_{li} - V_i \sum_{j=1}^{nbus} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) = 0 \tag{4.18}$$

The partial differential of above equation

$$\frac{\partial}{\partial Q_{gi}} = 1,$$

$$\frac{\partial}{\partial V_i} = -\sum_{j \neq i}^{nbus} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) + 2V_i B_{ii}, \frac{\partial}{\partial V_j} = -V_i (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij})$$

$$\frac{\partial}{\partial \theta_i} = -V_i \sum_{j \neq i}^{nbus} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}), \frac{\partial}{\partial \theta_j} = V_i V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij})$$

In this thesis, we believe the mechanical power of a PV generator node is controllable, according to changes in a control parameter μ, while the mechanical power P_m of a generator at the balance of the node is an unknown value to balance the requirement of system active power. When terminal resistor of balancing node generator $R\ s\_swing = 0$, $Pm\_swing = P\ g\_swing$。

As a result, system-wide Jacobian matrix A_{tot} has been solved, which will proide mathematical basis for small signal disturbance stability analysis.

4.4 Test System

We give a 3-machine 9-bus small systems and a 10-machine 39-bus large-scale system to verify the algorithm.

4.4.1 WSCC 3 machine 9 bus system

3-machine 9-bus system is a small control algorithm used to test. There are two local oscillation modes. PSS usually installed in the 2 and 3 nodes.

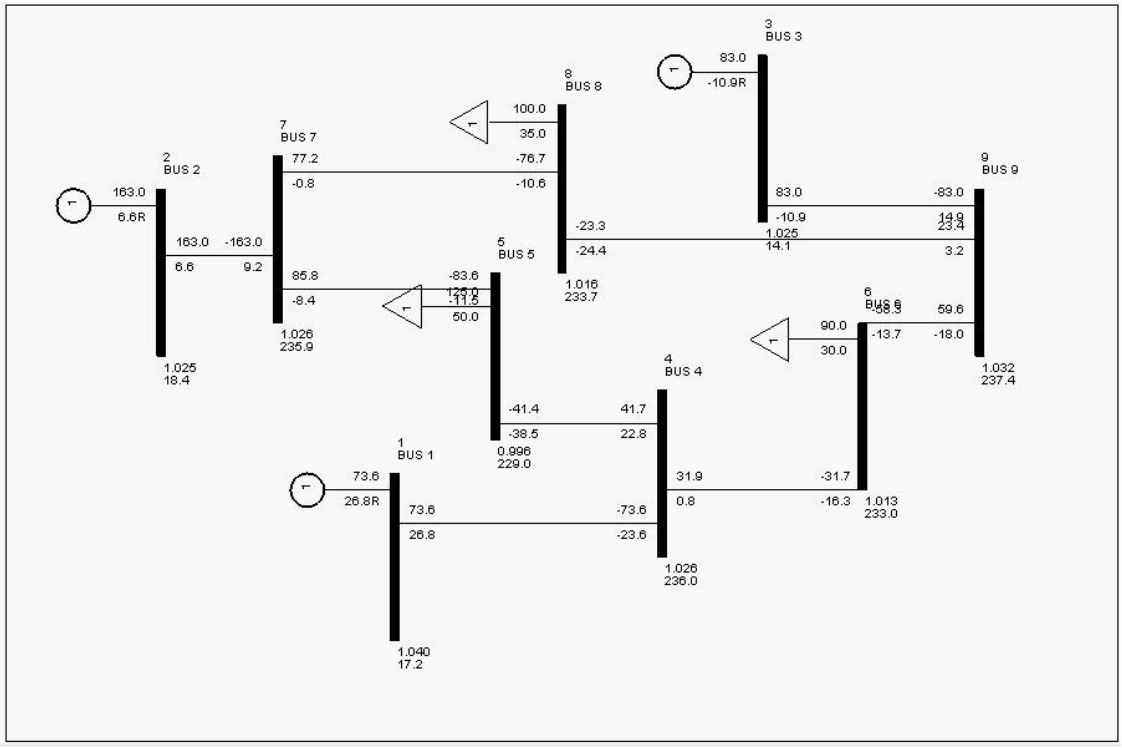

Figure 4.4 a WSCC 3 machine 9 bus system

4.4.2 New England 10 machine 39-bus system

New England 10 machine 39-bus is a relatively large-scale test system [27], which is an important test system in power system stability analysis. Testing the system control algorithm via this system is convincing.

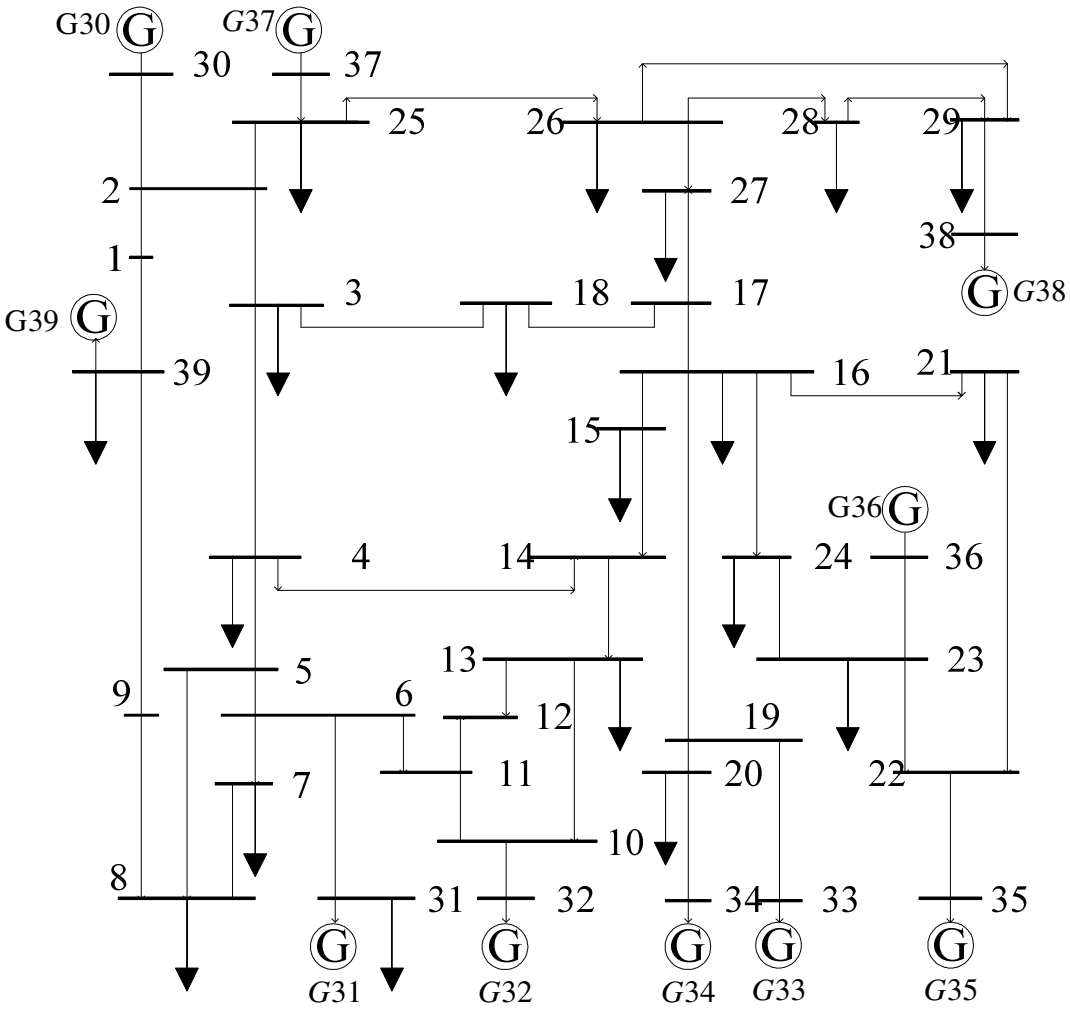

Figure 4.5 New England 10 machine 39-bus system

4.5 Optimization of power system small signal stability control

General stability of the system is divided into two categories: small signal disturbance stability and large disturbance stability. The concern of large disturbance stability is studying the control capable and maintaining stability of a system in a large disturbance, such as system failure, loss of load, loss of generators, etc [29-30]. At present, an effective mean to study of the stability of a large disturbance is time-domain

simulation. Whereas, the good way to study the stability of small perturbations, such as the slow load changes is using QSS analysis.

We can see power system stability control includes two levels: first, ensure the stability of the system from large disturbance. That is to leave a certain margin during normal running. The system can keep stability even meets large disturbance; Second, When system meets small perturbations, such as the slowly increase in load will reduce stability margin, and other issues we need various means to take it back.

There are two methods to maintain the stability of system from a small disturbance: one is to ensure system voltage stability; another is to maintain the system power angle stability [31].

For the small signal stability control has been carried out extensive research, the current study has several bottlenecks:

1. Fully understand the mechanism of instability of small perturbations. How distinguish the voltage instability and the instability of power angle? How distinguish oscillation instability with monotone instability?
2. Definition of stability index. Stability index determines to establish an effective optimization model.
3. The shortage of rapid and accurate optimization algorithms. As mentioned above, to solve such large-scale constrained nonlinear problem, regardless of mathematical optimization methods and modern heuristics methods to solve has its own limitations.

Here we try to establish a effectively optimized model according to different stability indicators. Use control measurement, and optimize the controller parameters in order to achieve a variety of purposes, further to ensure the stability of power system with small disturbance [32]. In this thesis, we use a complex optimization model, whose objective functions are not explicitly expressed, that is why we use a modern heuristic optimization method, evolutionary strategy: to solve multi-objective optimization

problem, in this thesis, we used Pareto optimal sequencing and evolutionary strategy as composite algorithm. Optimize and improve algorithm is not our main goal, Here focuses on establishing an optimal model, and deeply understands the small disturbance stability control.

4.6 Optimization Algorithm

Evolutionary strategy, genetic algorithms, and evolutionary planning are three kinds of algorithms collectively known as evolutionary computation, which belong to modern global optimization method. No matter how genes change, the comprehensive character always follow mean equal zero with a Gaussian distribution deviation [33]. The Different between evolutionary strategies and genetic algorithms is evolutionary strategies directly operates on the individual's performance rather than genes, also omits the encoding and decoding steps. Evolutionary strategy is more suitable for numerical optimization. Each individual in the evolutionary strategy is expressed as a floating-point vector, $v = (x, \sigma)$. here, x is one point in searching space； σ is used to guide the variation of the standard deviation vector.

Here we can give an example to see how the evolution works.

There is a group, the population of this evolutionary group is μ , each individual has *m* parameters need be optimized, and each individual represents an result of optimization. In the evolutionary of individuals, produced by hybridization can generate λ_1 individuals, at the same time the variation produced λ_2 individuals, so the total number of individuals is $\lambda = \lambda_1 + \lambda_2$. In a simulation we can easily use $\mu = 20, \lambda = 90$.

We can make the following deal: when the mutation or the hybrid happened in each generation, these the individual does not meet the constraints will be discarded. We will appropriate adjustment the control parameters if this event has a high probability. Here the control parameters are: $\sigma_i = 0.05/\sqrt{n}$, *n* is the number of optimization parameters.

The detail steps of evolutionary strategy are listed here:

1. Group initialization. For example, random generate μ individuals according to parameter range.
2. Randomly selected $2\lambda_1$ individuals from the population for hybrid operation in random, then produced λ_1 next generation individuals. Following apply a mutation operator on these emerging individuals.
3. After mutation operating on parent individuals, it produces λ_2 next generation individuals.
4. All individuals are evaluated (for multi-objective optimization, Pareto sorting is needed), according the fitness to retain the best μ individuals.
5. Final judgment. When the evolution meets the requirements or the individuals exceed the maximum evolution of algebra, we will stop the calculation. Otherwise, back to step 2 then continue. The Maximum evolution of algebra can be chosen as 50 generations to stop.

4.7 PSS optimization

Traditional goal of PSS global optimization is to minimum damping ratio of a system, usually people use heuristic optimization algorithms to realize it. In this thesis we present new PSS optimization method with wide frequency compensation [34]. Compare with regular method, our optimization has following features: (1) try to use the same phase of damping torque ΔT_e and $\Delta\omega$ generated by the low frequency oscillation of PSS as the optimization object. Select these individuals who can meet the requirement as the initial value for the next optimization step [35]; (2) then select to system-wide minimum damping ratio of PSS as a significant objective optimization parameters in one approach. System simulation results show that this method is effective [36].

Figure 4.10 is the linear one machine reference model, PSS need compensate the phase which is the lag angle of a excitation system (The lag phase between the U_{PSS} and $\Delta E'_q$).

$$\frac{\Delta E'_{qi}}{U_{PSSi}} = \frac{K_{Ai}(1+pT_{Ci})}{(1+pT_{Ai})(1+pT_{Bi})(K_{3i}+pT'_{d0})+K_{Ai}K_{6i}(1+pT_{Ci})} \tag{4.19}$$

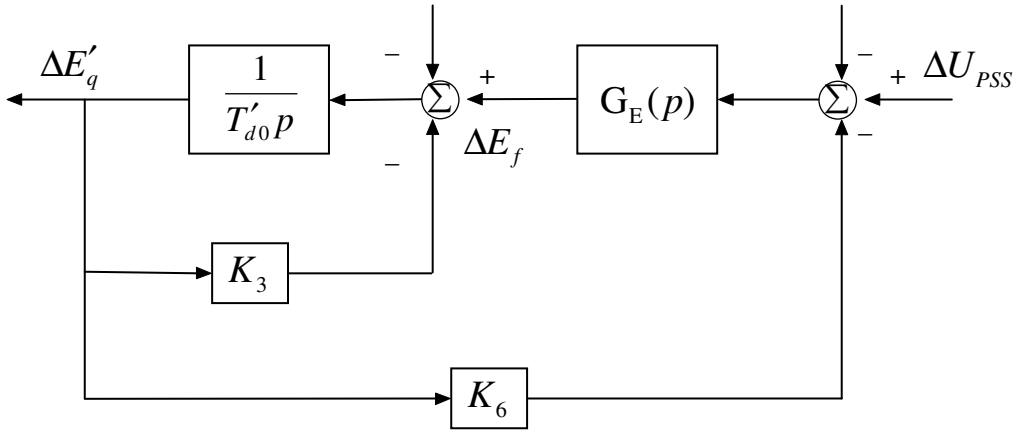

Figure 4.6 The transfer function diagram to show relationship between U_{PSS} and $\Delta E'_q$

The preact-lag time constant of PPS determines the compensation in frequency domain. A rationally optimized time constant is the key of the entire optimization. We want to input signal, $\Delta\omega$, into PSS to generate a damping torque ΔT_e and $\Delta\omega$ has similar phase as close as possible. The preact angle of PSS must cover the lag angle of excitation system. Optimization objective function showed here:

$$\begin{cases} \min J = \sum_{i=1}^{n} \left| \theta_{\Delta T_e - \Delta\omega}(f_i) \right| = \sum_{i=1}^{n} \left| \theta_{PSS}(f_i) + \theta_{Ex}(f_i) \right| \\ s.t. \quad T_1^{\min} \le T_{1i} \le T_1^{\max} \\ \quad T_2^{\min} \le T_{2i} \le T_2^{\max} \\ \quad T_3^{\min} \le T_{3i} \le T_3^{\max} \\ \quad T_4^{\min} \le T_{4i} \le T_4^{\max} \\ \quad -45^{\circ} \le \theta_{\Delta T_e - \Delta\omega}(f_i) \le 10^{\circ} \end{cases} \tag{4.20}$$

Here, $f_1, f_2, \cdots, f_n$ are points on the low-frequency oscillation with band (0.1 ~ 2Hz), e.g.: $f_1 = 0.1Hz$, $f_n = 2Hz$. $\theta_{\Delta T_e - \Delta\omega}(f_i)$ 、 $\theta_{PSS}(f_i)$ and $\theta_{Ex}(f_i)$ stand for in the frequency point f_i, the phase angle difference between PSS damping torque ΔT_e 与 $\Delta\omega$, compensation phase angle of PSS and lag phase angle of excitation system, respectively. The phase after PSS compensating need to meet the range (-45 ° ~ 10 °). When n is large enough, the objective function will be able to ensure that PSS to meet the requirements of the compensation in the entire low-frequency, which means system can operate in a variety of modes. Optimization problem of the preact-lag time in PPS can be described as a problem to find out the minimum of multi-parameter nonlinear function, and then we can use evolutionary strategy [37].

Here we can find several candidates, which will be used as the initial value of next step for global optimization. Damping ratio measures a system dynamic performance, which directly reflects the damping level of a whole system. Therefore, it is better to use the damping ratio as the goal of global optimization. We know the general damping ratio requirement of power system is not less than 5%, now we have the second optimization objective function [38]:

$$\begin{cases} \max \ (\min\{\xi_i\}, i \in \text{oscillation model}) \\ \text{s.t.} \quad k_i^{\min} \le k_i \le k_i^{\max} \\ \quad T_1^{\min} \le T_{1i} \le T_1^{\max} \\ \quad T_2^{\min} \le T_{2i} \le T_2^{\max} \\ \quad T_3^{\min} \le T_{3i} \le T_3^{\max} \\ \quad T_4^{\min} \le T_{4i} \le T_4^{\max} \\ \quad -45^\circ \le \theta_{\Delta T_e - \Delta\omega}(f_i) \le 10^\circ \end{cases} \tag{4.21}$$

The second step will optimize all the parameters of PSS in global, also limit the compensation of the broadband segment. If the individual cannot meet the requirement, it will minus a penalty value to avoid destruct frequency-domain constraints. PSS gains K_{PSS} values range into [0.01,15].

The result of WSCC 3-machine 9-bus result and table are list as following:

Table 4.1 WSCC system PSS optimization results

Location(Bus No)	Kpss	T1	T2	T3	T4
2	9. 502	0. 2192	0. 0632	0. 4157	0. 0109
3	3. 566	0. 1255	0. 0216	0. 4651	0. 0136

Table 4.2 WSCC system quantity with PSS and without PSS

Without PSS		With PSSs	
Eigenvalues	Damping Ratio (%)	Eigenvalues	Damping Ratio (%)
-0.753±j12.819	5.87	-2.733±j12.631	21.1
-0.128±j8.327	1.33	-2.112±j7.95	25.8

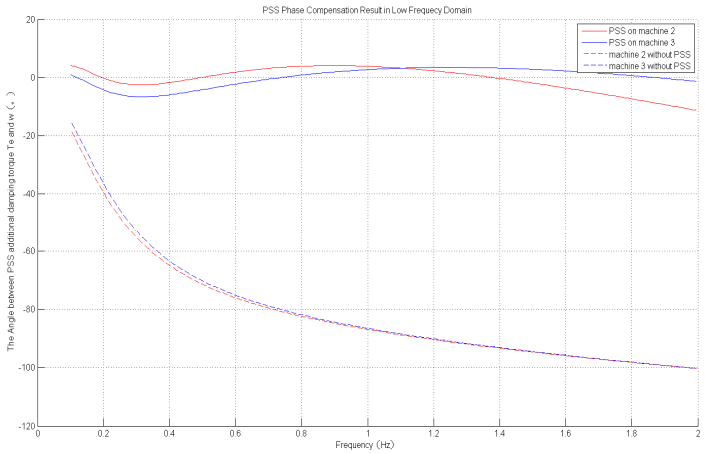

Figure 4.7 WSCC PSS low frequency compensation results

To prove the reliability of this method, we use a larger system, New England 10-machine 39-bus, as an experiment to demonstrate our PSS parameters optimization is very effective. Excitation system parameters are $K_A = 200, T_R = 0.03, T_A = 0.02, T_B = 10, T_C = 1$. The basic working mode is named as mode 1, operation 2 is designed for the 14-15 lines outage, operation 3 is the 21-22 lines outage. Before PSS installing in the system, oscillation modes are shown in Table 4.1. The results shown from Table 4.1 told us damping ratios of most oscillation modes are very low, even has a negative damping in some cases. In order to improve the system damping, we installed PSS on all nine generators, after using the optimization program memetioned in this thesis to optimize the PSS. The results are shown in Table 4.2.

Table 4.3 Oscillation modes of New England 10 machine 39-node system without PSS

Operation 1		**Operation 2**		**Operation 3**	
Eigenvalue	**Damping ratio (%)**	**Eigenvalue**	**Damping ratio(%)**	**Eigenvalue**	**Damping ratio(%)**

-0.6855±j10.906	6.2733	-0.686±j10.9026	6.2800	-0.680±j10.8647	6.2470
-0.3069±j8.8874	3.4516	-0.3048±j8.879	3.4310	-0.3032±j8.8601	3.4202
-0.3431±8.6372	3.9690	-0.3456±j8.6319	4.0010	-0.3386±j8.5669	3.9491
-0.2587±8.1905	3.1574	-0.2596±j8.1886	3.1686	-0.2597±j8.1566	3.1827
-0.1316±7.4561	1.7650	-0.1239±j7.4309	1.6669	-0.1295±j7.4652	1.7346
-0.2136±7.3417	2.9081	-0.2194±j7.3392	2.9877	-0.1234±j6.9699	1.7701
-0.0782±6.4954	1.2032	0.1565±j6.0547	-2.5844	0.0399±j6.2040	-0.6428
0.1565±6.0557	-2.5835	-0.02±j5.86	0.0037	0.0998±j5.5952	-1.7828
0.0458±3.8751	-1.1830	0.0508±j3.7365	-1.3597	0.1578±j3.5583	-4.4311

Table 4.4 New England 10 machine 39-node system PSS optimization results

Location	**Kpss**	**T1**	**T2**	**T3**	**T4**
1	14.8019	0.4770	0.0857	0.4943	0.0261
2	7.6569	0.2658	0.0279	0.7478	0.0614
3	14.1087	0.4779	0.0879	0.3766	0.0207
4	8.4443	0.1145	0.0160	0.9461	0.0425
5	10.1302	0.9466	0.0408	0.1606	0.0319
6	9.5867	0.2498	0.0649	0.6192	0.0239
7	8.8845	0.1642	0.0121	0.7742	0.0655

8	4.9758	0.6123	0.0121	0.2743	0.0772
9	11.9999	0.7246	0.0683	0.1687	0.0167

Table 4.5 Oscillation modes of New England 10 machine 39-nodes system after installing PSS

Operation 1		**Operation 2**		**Operation 3**	
Eigenvalue	**Damping ratio (%)**	**Eigenvalue**	**Damping ratio(%)**	**Eigenvalue**	**Damping ratio(%)**
-2.350±j10.6441	21.5588	-2.3469±j10.638	21.5428	-2.3279±j10.596	21.4591
-1.6152±j7.4979	21.0584	-1.6119±j7.4954	21.0246	-1.6139±j7.4531	21.1641
-2.9053±j7.4738	36.2314	-2.9021±j7.4675	36.2236	-2.8434±j7.4788	35.5382
-2.5216±j7.5175	31.8019	-2.5190±j7.5158	31.7792	-2.4133±j7.5262	30.5345
-2.0533±j6.2087	31.3986	-1.7675±j6.2112	27.3702	-1.6892±j6.2164	26.2230
-1.6101±j6.0895	25.5621	-2.0749±j6.2284	31.6052	-2.1595±j5.8166	34.8049
-1.9360±j5.7370	31.9748	-1.4905±j5.0083	28.5251	-1.6154±j5.1627	29.8623
-1.7975±j4.9643	34.0464	-1.7968±j4.9870	33.8969	-1.4100±j4.8453	27.9409
-0.5636±j3.2331	17.1717	-0.5709±j3.0775	18.2380	-0.3532±j3.1224	11.2399

Set the parameters of PSS as optimization results, the system oscillation modes are shown in Table 4.5. Here we can see, all the mechanical oscillation modes reached satisfactory damping ratios. The compensation performance of each PSS in the low frequency can be seen in Figure 4.8.

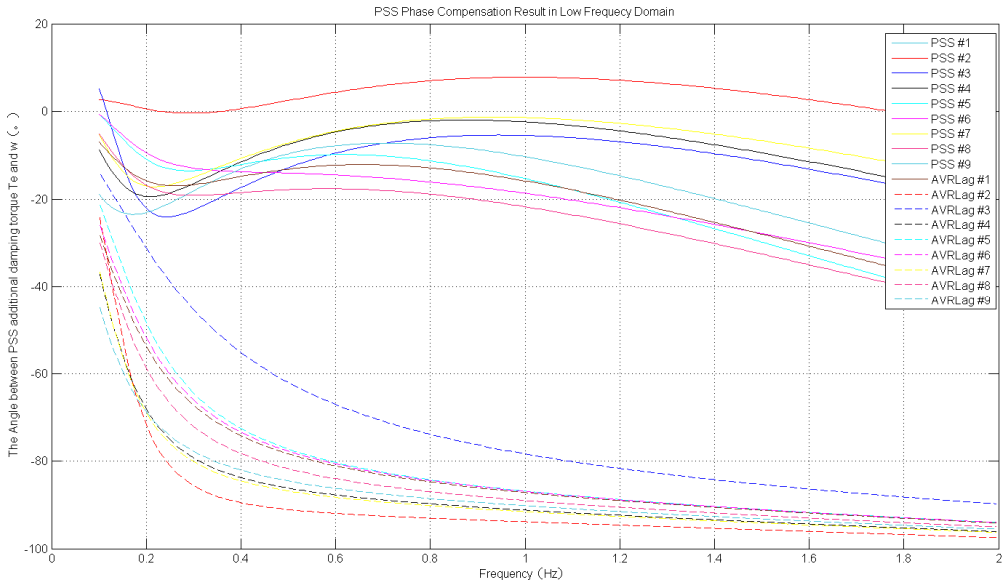

Figure 4.8 New England 10 machine 39-node system PSS low frequency compensation results

Simulate the New England system in PSS/E. At first, we use the dynamic models without PSS. Line fault between bus 26 and bus 29 at 1 second. Clear this disturbance after 0.2 second, and run simulation to 5 second. We get the machine speed plot as shown in fig. 4.9

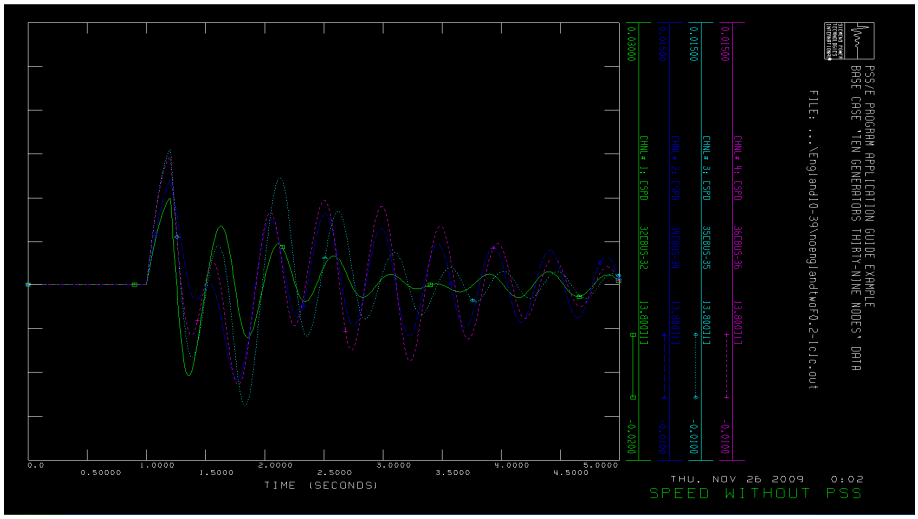

Figure 4.9 PSS/E simulation plot without PSS

Next, add the PSS dynamic models at the locations as discussed above. Run the same simulation process as without PSS trial, the result is show in Fig. 4.10.

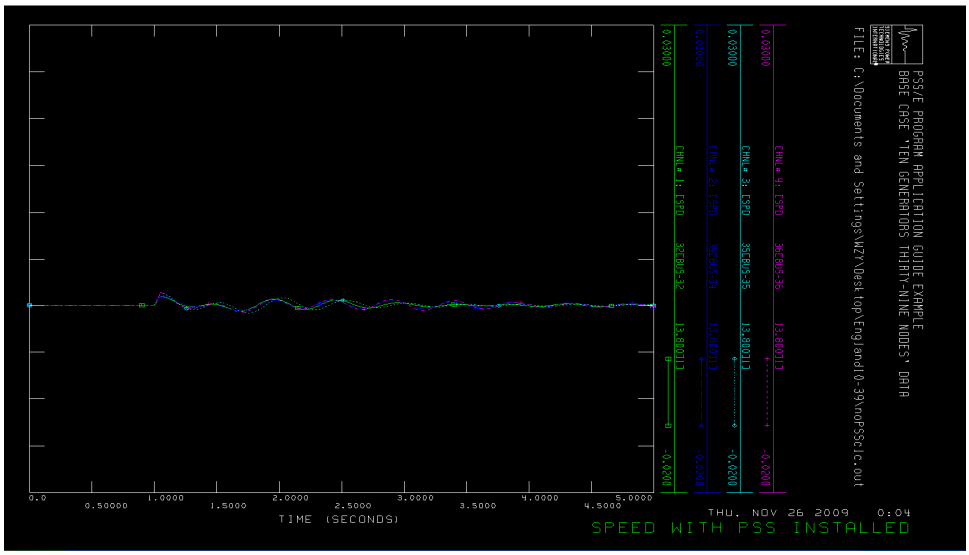

Figure 4.10 PSS/E simulation plot with PSS

The oscillation damped clearly from the simulation of PSS/E. This experiment proves again the feasibility of power system stability compensation on broad frequency range based on evolutionary strategy.

Chapter 5 Conclusion

5.1 DLL Modules demo program and summary

On-line Trigger DLL encapsulates the FNET on-line event detection, so that the algorithm can be used by other data source through calling the DLL file. The function of the On-line Trigger DLL includes missing date handling, completeness probing, data alignment and event trigger application.

C++ is used for developing the DLL and Lib files named 'ETrigNDiatry1.dll' and 'ETrigNDiatry1.dll' respectively. The outputs of the DLL file are two variables 'bEvent' and 'm_cEventData'. The 'bEvent' is a Boolean type to indicate whether there is an event or not. When the value of 'bEvent' is true, it means there is an event. And the 'm_cEventData' is a string providing the information of event data and time in 'YYYY-MM-DD HH:MM:SS' format.

Because the target application database has not been provided, a testing program is written based on the FNET Access database. The 'ETrigNDiatry1.dll' and 'ETrigNDiatry1.lib' are included in the folder of testing program, and the lib file path should be added into the additional lib source. By the following variable statements, 'bEvent' and 'm_cEventData' can be exported from the DLL file.

```
__declspec (dllimport) CString m_cEventtime;
__declspec (dllimport) CEvent  g_Event;
```

Fig.5. 1 shows the structure and result of testing program. As can been seen, the testing program can obtain the event trigger information by calling the On-line Trigger DLL. Due to the fact that this DLL is developed on the Visual 2008 platform, the Redistributable Package of Visual 2008 is required in order for the files to run successfully. Fig. 5.2 is the demo program calling event trigger DLL, and display event time value from DLL.

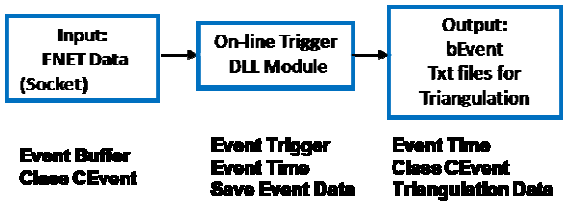

Figure 5.1 Event Trigger DLL structure

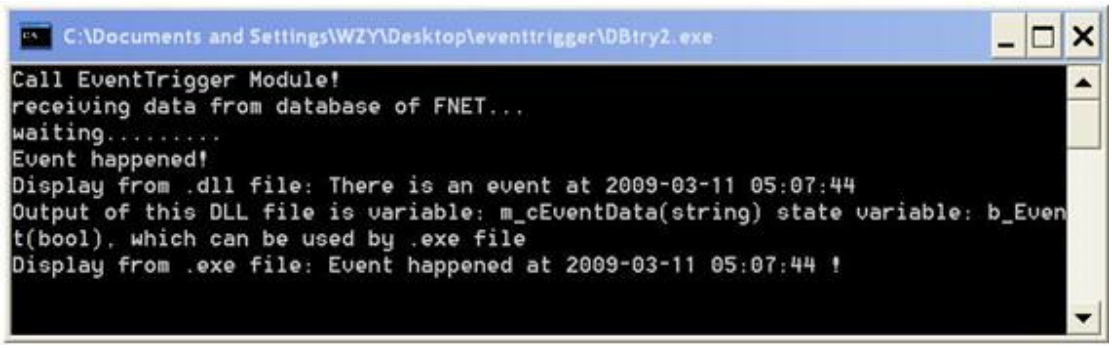

Figure 5.2 On-line Trigger DLL Testing Program

The Triangulation Application DLL encapsulates the FNET Triangulation Algorithm for estimating event location, so that the algorithm can be used by other data source through calling DLL file. C++ is used for developing the DLL and Lib files named 'triaDLLtry.dll' and 'triaDLLtry.lib' respectively. The outputs of the DLL file are the LOD information. It includes: unit information (g_csUnits, g_csLocation), estimated event location (g_csFDC, g_csRegion, g_csCity, g_csState, g_csZip, g_fLatitude, g_fLongitude), event information (g_iTripAmount, g_fSpeed) and event time (g_fEvTime).

Put both the 'triaDLLtry.dll' and 'triaDLLtry.lib' in the folder of the calling executable program. The lib file path should be added into the additional lib source. The output variable statements are the same as those of the On-line Trigger DLL. And all the output variables are saved into a text file named 'TriangRecord.txt'.

Fig.5.3 shows the module structure. Fig. 5.4 is the saved text file and alerm email sent by demo program call DLL. As can been seen, the testing program can obtain the estimating location of disturbance information by calling the On-line Trigger DLL.

Figure 5.3 struction of dll

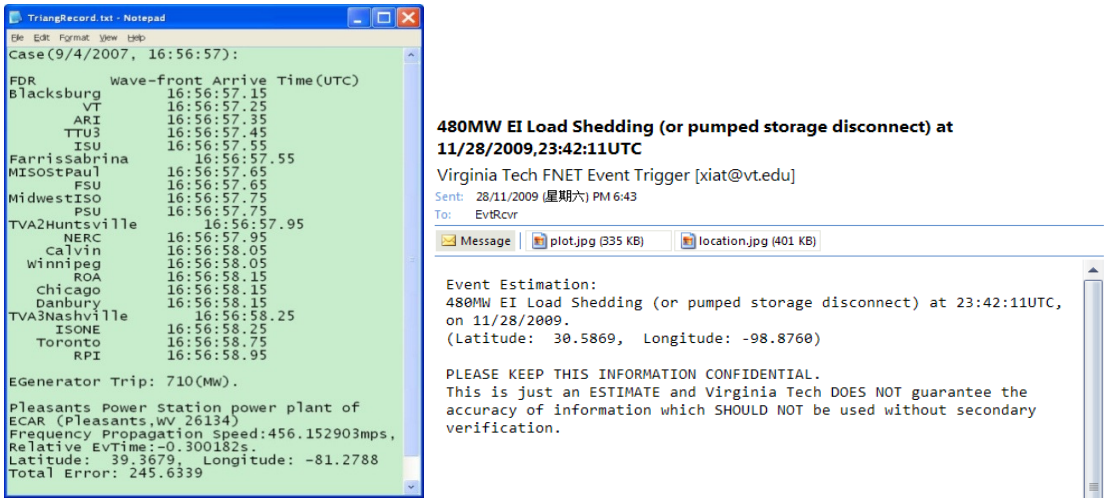

Figure 5.4 Testing Program Structure and Output Text File

5.2 PSS phase compensation conclusion

With the development of the power system, grid size is increasing and become more vulnerable. As the nature of the non-linear, high dimension, multiple steady-state solution and the lack accurate description, the optimize coordination and control is much more difficult than control system. In Power system stabilizer analysis, this paper analyzes the influence PSS brought to power system small signal stability, and optimizes PSS controllability in damping oscillation. The main steps are as following. First, small signal stability analysis models, including four-order generation model, AVR three-order model and PSS model. And introduces the method of generating system linear equation,

derives Jacobian matrix for the small signal stability analysis. Then optimize PSS dynamic parameters to compensation AVR's lagging phase in wide frequency range. Evolutionary strategy is the main optimization method used in analysis. Fig 5.5 shows the flow chart and step of PSS optimization. All the analysis can calculation is programmed in Matlab.

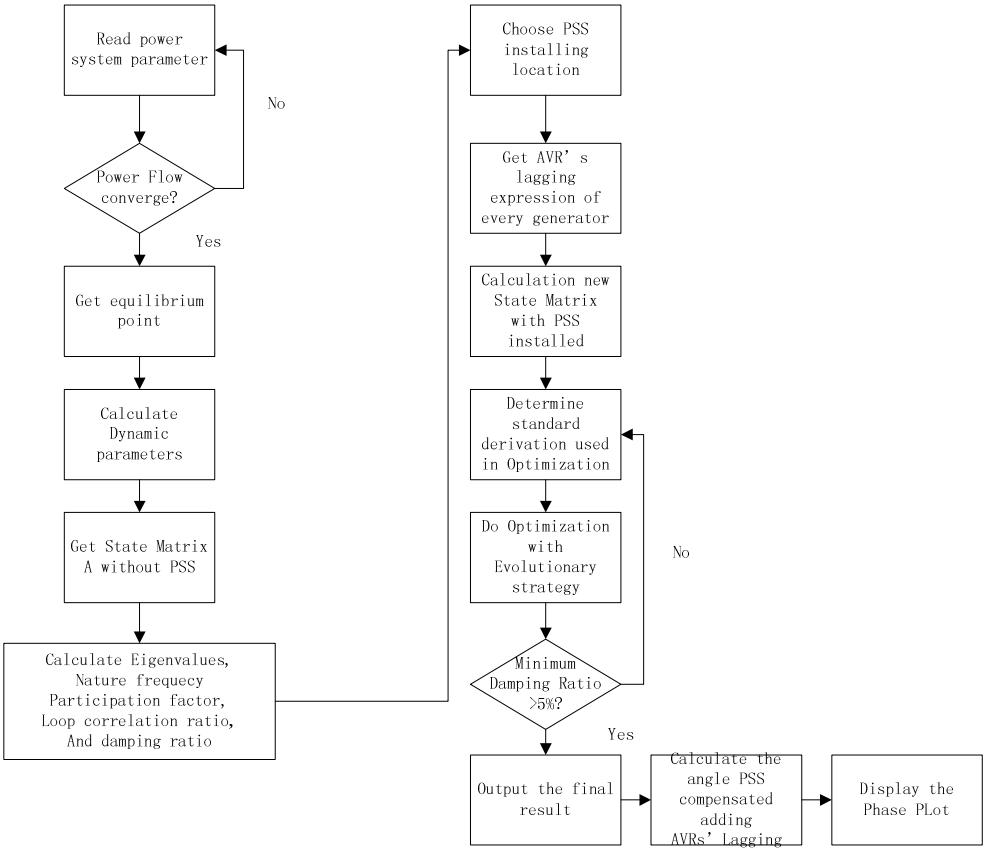

Figure 5.5 Optimization flow chart in Matlab

Reference

[1] A. G. Phake, "Synchronized Phasor Measurements-- A historical overview," Transmission and Distribution Conference and Exhibition 2002: Asia Pacific. IEEE/PES, vol. 1, pp. 476-479, Oct.2002.

[2] "IEEE Std 1344-1995 IEEE Standard for Synchrophasor for Power System."

[3] A. G. Phadke, J. S. Thorp, and K. Karimi, "State Estimation with Phasor Measurements," Trans. of IEEE on Power System, Feburary 1986.

[4] B.Qiu, L.Chen, V. A. Centeno, X. Dong, and Y. Liu, "Internet based frequency monitoring network (FNET)," presented at IEEE Power Eng. Soc. Winter Meeting, 2001.

[5] Beginning of Visual C++ 2008: McGraw-Hill, 1994.

[6] F. Chunju, L. Shengfang, Y. Weiyong, and L. K.K., "Study on adaptive relay protection scheme based on phase measurement unit (PMU)," presented at Developments in Power System Protection ,2004. Eighth IEE International Conference on, April 2004.

[7] J. S. Thorp, A. G. Phadke, S. H. Horowitz, and M. M. Begovic, "Some Applications of Phasor Measurements to Adaptive Protection," presented at PICA 87, Montreal Canada, May 1988.

[8] V.Centeno, J. D. L. Ree, A. G. Phadke, G. Michel, and J. Murphy, "Adaptive Out-of-step Relaying Using Phasor Measurement Techniques," Computer Applications in Power, IEEE, vol. 6, pp. 12-17, Oct 1993.

[9] Z. Zhong, C. C. Xu, B. J. Billian, L. Zhang, S.-J. S. Tsai, R. W. Conners, V. A. Centeno, A. G. Phade, and Y. Liu, "Power system frequency monitoring network (FNET) implementation," IEEE Transactions on Power System, vol. 20, pp. 1914-1921, Nov. 2005.

[10] J. Machowski, J. W. Bialek, and J. R. Bumby, Power System Dynamics and Stability: John Wiley & Sons, 1998.

[11] P. Kundur, Power System Stability and Control: McGraw-Hill, 1994.

[12] J. Chen, "Accurate frequency estimation with phasor angles," in ECE, vol. Master. Blacksburg: Virginia Tech, 1994.

[13] A. G. Phadke, "Synchronized Phasor Measurement in Power System," IEEE Computer Applications in Power, vol. 6, pp. 10-15, April 1993.

[14] J. Zuo, Z. Zhong, R. Gardner, H. Zhang, and Y. Liu, "Off-line Event Filter for the Wide Area Frequency Measurements," presented at IEEE PES General Meeting, 2006.

[15] N. Mithulananthan, C. A. Canizares, J. Reeve, and G. J. Rogers, "Comparison of PSS, SVC, and STATCOM controllers for damping power system oscillations," Power Systems, IEEE Transactions on, vol. 18, pp. 786 - 792, May 2003.

[16] A. Elices, L. Rouco, H. Bourles, and T. Margotin, "Physical interpretation of state feedback controllers to damp power system oscillations," Power Systems, IEEE Transactions on, vol. 19, pp. 436 - 443 Feb. 2004

[17] A. J. Arana, J. N. Bank, R. M. Gardner, and Y. Liu, "Estimating Speed of Frequency Disturbance Propagation Through Transmission and Distribution Systems," presented at Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES Oct. 29 2006-Nov. 1 2006.

[18] R. J. Murphy, "Power system disturbance monitoring," presented at Western Protective Relay Conference, Spokane,WA, Oct 17-20, 1994.

[19] L. Huang, M. Parashar, A. G. Phake, and J. S. Thorp, "Impact of electromechanical wave propagation on power-system reliability," presented at Proc. 39th CIGRE Conf., Paris, France, Aug. 25-30,2002.

[20] S. S. Tsai, Z. Zhong, J. Zuo, and Y. Liu, "Analysis of wide-area frequency measurement of bulk power systems," presented at IEEE Power Engineering Society General Meeting, 2006.

[21] J. Zuo, Z. Zhong, R. M. Gardner, H. Zhang, and Y. Liu, "Off-line event filter for the wide area frequency measurements," presented at IEEE Power Engineering Society General Meeting, 2006.

[22] T. Van Cutsem, and C.D. Vournas. "Voltage stability analysis in transient and mid-term time scales," IEEE Transactions on Power Systems, 1996, 11(1): 146-154.

[23] V. Ajjarapu, and B. Lee. "Bifurcation theory and its application to nonlinear dynamical phenomena in an electrical power system," IEEE Transactions on Power Systems, 1992, 7(1): 424-431.

[24] B. Lee, and V. Ajjarapu. "Invariant subspace parametric sensitivity (ISPS) of structure-preserving power system models," IEEE Transactions on Power Systems, 1996, 11(2): 845.

[25] W. Ji, and V. Venkatasubramanian. "Dynamics of a minimal power system: invariant tori and quasi-periodic motions," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 1995, 42(12): 981-1000.

[26] C.S. Chang, and J.S. Huang. "Optimal SVC placement for voltage stability reinforcement," Electric Power Systems Research, 1997, 42(3): 165.

[27] N. Mithulananthan, C.A. Canizares, and J. Reeve. "Comparison of PSS, SVC, and STATCOM controllers for damping power system oscillations," IEEE Transactions on Power Systems, 2003, 18(2): 786.

[28] C.A. Canizares. "Power Flow and Transient Stability Models of FACTS Controllers for Voltage and Angle Stability Studies," in Proceedings of the 2000 IEEE-PES Winter Meeting. 2000. Singapore.

[29] C.A. Canizares, N. Mithulananthan, and A. Berizzi. "On the linear profile of indices for the prediction of saddle-node and limit-induced bifurcation points in power systems," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2003, 50(12): 1588-1595.

[30] S. Ayasun, C.O. Nwankpa, and H.G. Kwatny. "Computation of singular and singularity induced bifurcation points of differential-algebraic power system model," IEEE Transactions on Circuits and Systems I: Regular Papers, 2004, 51(8): 1525-1538.

[31] C.A. Canizares, F.L. Alvarado, and C.L. DeMarco. "Point of collapse methods applied to AC/DC power systems," IEEE Transactions on Power Systems, 1992, 7(2): 673-683.

[32] C.J. Parker, I.F. Morrison, and D. Sutanto. "Application of an optimisation method for determining the reactive margin from voltage collapse in reactive power planning," IEEE Transactions on Power Systems, 1996, 11(3): 1473-1481.

[33] A.A.P. Lerm, and C.A. Canizares. "Multiparameter bifurcation analysis of power systems," in Proc. North Amer. Power Symp. 1998. Cleveland, OH.

[34] D. Chattopadhyay, and B.B. Chakrabarti. "Reactive power planning incorporating voltage stability," IEEE Transactions on Power Systems, 2002, 24(3): 185-200.

[35] Q. Wu, D.H. Popovic, and D.J. Hill. "Avoiding sustained oscillations in power systems with tap changing transformers," International Journal of Electrical Power and Energy System, 2000, 22(8): 597-605.

[36] Y. Zhou, and V. Ajjarapu. "A fast algorithm for identification and tracing of voltage and oscillatory stability margin boundaries," 2005, 93(5): 934-946.

[37] G. Rogers, "Power system oscillations," Kluwer Academic Publishers, 2000.

[38] P.B.d. Araujo, and L.C.X. Jr. "Pole placement method using the system matrix transfer function and sparsity," Electrical Power and Energy Systems, 2001, 23(3): 173-178.