FINAL PROJECT: Estimating security of modern electrical networks in adverse weather conditions

MS2, Winter 2016
Due: 13.1.2017

1 Overview

Over the first half of the semester we've been developing from the ground up a model of an electrical network with distributed generation and monitoring by a supervisory controller. The schematic is shown in Figure 1. The model includes:

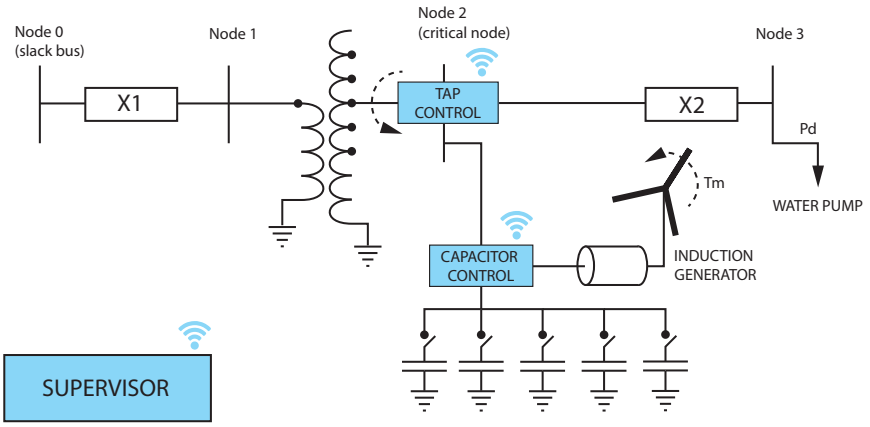

Figure 1: Network system schematic.

Component 1: *Network* - the network backbone includes three nodes. Node 0 is a slack bus where the voltage is held constant voltage. Node 2 is the critical bus to which everything else connects including a wind turbine, a capacitor bank, a voltage sensor, and a reactive power sensor. Node 3 is the location of the network load, a water pump. *MATLAB files: dNetwork, jNetwork, eNetwork; UMDES files: T_switch, T_sensor*

Component 2: *Tap controller* - the tap controller is connected to the voltage sensor and a tap changer. It includes an integrated stopwatch by which it commands the tap switch in response to voltage changes. The controller also accepts inputs from the remote Supervisor. *MATLAB files: dTimerT, jTimerT, eTimerT; UMDES files: T_ 6sup, T_ timer*

Component 3: *Capacitor bank controller* - the capacitor bank controller is connected to the power sensor and the capacitor bank switch valves. It includes an integrated stopwatch by which it commands the valves in response to power changes. The controller also accepts inputs from the remote Supervisor. *MATLAB files: dTimerC, jTimerC, eTimerC; UMDES files: C_ 6sup, C_ timer*

Component 4: *Load* - the load includes a water pump that maintains local reservoir water levels. The control of the water pump does not consider network states and hence is here omitted. *MATLAB files: dLoad, jLoad*

Component 5: *Capacitor bank* - the capacitor bank includes five shunt capacitors with switchable connections a a reactive power sensor. The capacitor bank connections are controlled by the capacitor bank controller. *MATLAB files: dCapacitor, jCapacitor; UMDES files: C_ switch, C_ sensor*

Component 6: *Turbine* - the wind turbine includes the physical blades as well as an induction generator. It includes no further control systems. It's reactive power demand is supplied by the adjacent capacitor bank. *MATLAB files: dTurbine, jTurbine*

For a given set of parameters p characterising the above components, we also have a working executive model in Matlab that computes a single deterministic run, i.e., the time t, state x and the state derivative d trajectories as well as the event times te, the dynamic parameter values z and the fired events e. The time interval is $[0, tF]$. *MATLAB function: NetworkSimRun.* The function NetworkSimRun.m accepts two inputs, the simulation time tF and the parameter values for parameters $X2, Ps, Tm$ in a single vector $Dat = (X2, Ps, Tm)$. The function outputs are the following arrays:

- t: time trajectory
- x: continuous state trajectory
- te: event time trajectory

- z: dynamic parameter trajectory
- e: event trajectory, where the even names match the names listed in FSA schematics.

Hence, in order to execute a run of the system enter the following at the command prompt:[t,x,d,te,z,e] = NetworkSimRun(tF,Dat). Note that every time you call the function a series of plots is also generated. You will find that in your Monte Carlo simulation it is beneficial to comment out the plot generation.

2 Problem

The system in Figure 1 is analysed under severe weather conditions when the system parameters may be perturbed away from their nominal values. There are three highly sensitive parameters:

Component 1: *X2* - the impedance of the local line may change due to interference and temperature changes.

Component 2: Ps - the steady state load will change if the abnormal weather includes flooding, which increases the pump activity.

Component 3: Tm - the torque placed on the wind turbine will increase if the abnormal weather includes wind conditions.

Variation of these parameters was measured for 365 days of abnormal weather. These data points are shown in Figure 2 together with marginal distributions. You can access the your assigned data set at `http://ccy.zcu.cz/index.php/Courses`. In addition to the data points, you also have access to a frequency function defined on a $5 \times 5 \times 5$ grid. The function is defined by

$$f(X2, Ps, Tm) = \#\text{ of points in the grid block centred at } (X2, Ps, Tm). \tag{1}$$

The values of the frequency function and the grid points are given by the $6 \times 6 \times 6$ arrays $f, X2, Ps, Tm$. *MATLAB file: NetworkData.mat*

The assignment is to construct a Monte Carlo simulator of the above system to estimate $\mathbb{E}(T_{ab}(X2, Ps, Tm))$, where $T_{ab}(X2, Ps, Tm)$ is the total time the network voltage at node was abnormal if the parameter values

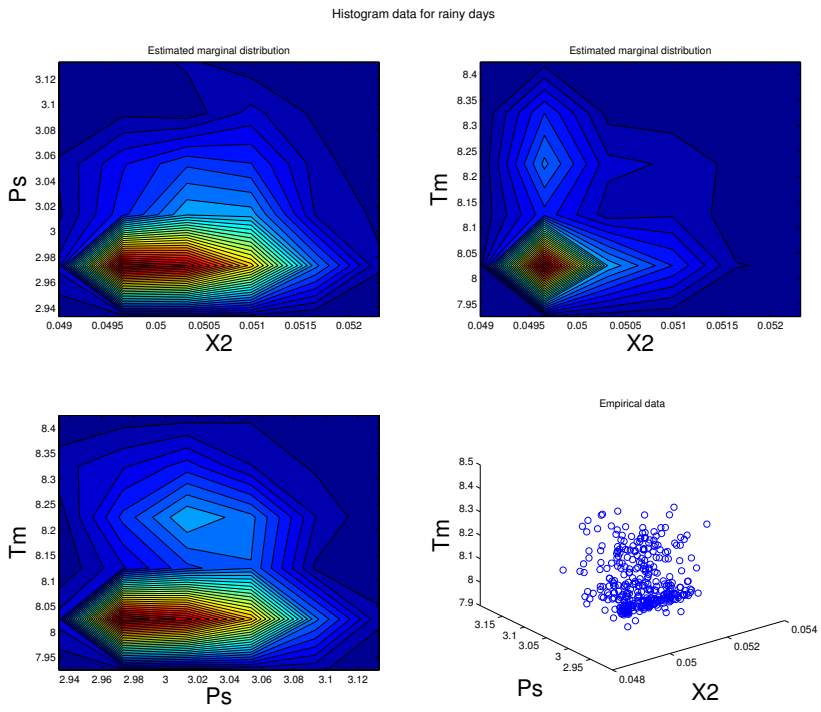

Figure 2: Measured data. First three plots illustrate estimated marginal distributions. The last plot shows the measured values

are perturbed from their nominal values to the values $(X2, Ps, Tm)$ whose estimated distribution is described above. Formally,

$$T_{ab} = \int_0^{tF} I\left(|V2| < Vm \vee |V2| > Vp \vee Q2 < Qm \vee Q2 > Qp\right) dt. \qquad (2)$$

The precision of the estimate should be $\pm 0.05 sec$ with 95% confidence level.

While executing your individual runs, make the following assumptions:

Assumption 1: the individual runs are independent, hence you can always start from the same initial state.

Assumption 2: the run final time (input tF to NetworkSimRun) is an exponential random variable with a mean of 6sec. It is assumed that tap changer and capacitor bank adjustments are only a first measure in ensuring network security and after the run is over the network is always returned back to the safe limits, i.e., $T_{ab} \leq tF$.

The following are some additional practical remarks.

Remark 1: The simulation should converge for all the data points you're given. However, it may be that the simulation does not converge for the randomly generated samples within your Monte Carlo simulation (you can assume that if a run does not complete in 10seconds it will not converge). You are allowed to simply ignore these cases.

Remark 2: While you may just simulate the system using the provided measured data points, this will not give you the accuracy you need. Hence, you will need to select a method for generating random parameter values in the Monte Carlos simulation, e.g., using the Metropolis-Hastings Markov Chain Monte Carlo method discussed in lecture.