Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Search Methods Based Control of the Simplified Model of the ABB Power Network

Samuel Shapero, Alexandre Donzé

Report n^o TR-2005-6

March 1, 2005

Search Methods Based Control of the Simplified Model of the ABB Power Network

Samuel Shapero, Alexandre Donzé

March 1, 2005

Abstract

This paper describes the modelling and control optimisation of a power transmission system. This system was adapted from a simplified version of the ABB Test Case model (Larsson 2002), provided within Computation and Control (CC) project, with the addition of a variable capacitance, the control input parameter (μ). A search methods based control scheme taking advantage of the dynamic properties of each input variable was implemented and successfully tested. The same methodology can be applied to more complicated power systems.

Keywords: Search based optimal control methods, Power Systems, Model Predictive Control, Hybrid Systems

Reviewers:

Notes:

How to cite this report:

@techreport { ,
title = { Search Methods Based Control of the Simplified Model of the ABB Power Network},
authors = { Samuel Shapero, Alexandre Donzé},
institution = { Verimag Technical Report },
number = {TR-2005-6},
year = { },
note = { }
}

1 System description

The system under study is a simple load voltage bus include a line transmission, a transformer and a capacitor bank (see fig 1). Its dynamics can be described by a differential algebraic equation:

$$\dot{x} = f(x, y, u) \quad (1)$$
$$0 = g(x, y, u) \quad (2)$$

where $x = [x_p\ x_q]^T$ is the dynamic state vector, $y = [v\ \delta]^T$ is the algebraic state vector representing the voltage amplitude and phase at the load bus, and $u = [n\ \mu\ k\ X]^T$ is the input parameter vector.

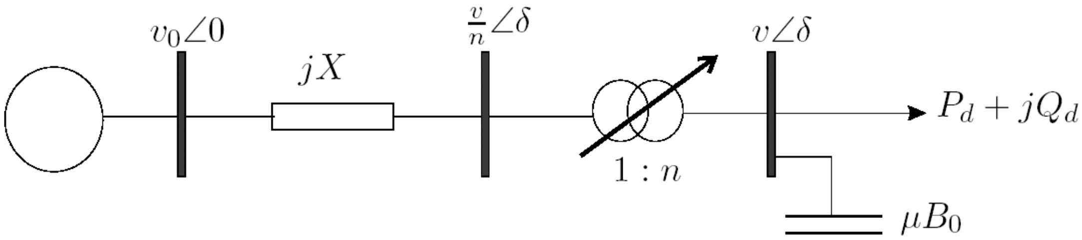


Figure 1: Power system under study

Of the four inputs, the first three are control variables:

- n - the tap ratio. The system includes a transformer, and n represents the ratio between the input and output voltage. It has range $[.8...1.2]$ and can be incremented .02 every 30 seconds. This is the least expensive control variable to change, but also produces the smallest change in the output voltage.
- μ - the compensating capacitance. The system includes a capacitor to compensate for faults in the line impedance, and its capacitance μ has range [.25 .75 1 1.25 1.5]. It can be changed every thirty seconds, at a greater cost than n, but with generally more affect on the output voltage.
- k - load shedding. As an emergency measure, the system can reduce the load by proportion k. The load shedding has range [0 .05 .1 .15] and is adjustable every thirty seconds. While changing k is free, having a high k is extremely expensive. Minimizing k is a system constraint and k should only be raised to avoid system collapse.

There is one disturbance input:

- X - the line impedance. By default, it starts at .25 p.u., but increases to represent a fault. If the fault is sufficiently large, the system will collapse without adequate control.

We are only concerned with one output variable:

- v - the output voltage. Our first priority is to avoid system collapse at all costs, generally by keeping v above .9 p.u. A secondary goal is to have v stabilize around 1.0 p.u.

Analytical description of the DAE can be derived :

$$\dot{x}_p = -\frac{x_p}{T_p} + P_0(v^2 - 1) \quad (3)$$
$$\dot{x}_q = -\frac{x_q}{T_q} + Q_0(v^2 - 1) \quad (4)$$
$$0 = \frac{v_0 v}{nX}\sin\delta + (1-k)(\frac{x_p}{T_p} + P_0 v^2) \quad (5)$$
$$0 = \frac{v_0 v}{nX}\cos\delta + \frac{v^2}{n^2 X} - \mu B_0 v^2 + (1-k)(\frac{x_q}{T_q} + Q_0 v^2) \quad (6)$$

2 Control

Control of the system was designed to minimize the cost function $C(x,y,u)$ over the future runtime of the system, where:

$$C(x,y,u) \quad = \quad \int_{t^*}^{\infty} A_1(y-y_{ref}) + A_2\big(u(t^+),u(t^-)\big) + V_c(x,y,u)\mathrm{d}t \tag{7}$$

The three costs reflect the demands and realities placed upon the system:

- $A_1(y-y_{ref})$ is the secondary cost of deviation from the reference voltage of 1 p.u., representing the cost of delivering an unstable or underpowered voltage supply to consumers.
- $A_2\big(u(t^+),u(t^-)\big)$ is the cost of switching input parameters, in this case μ and n.
- $V_c(x,y,u)$ is the cost of constraint violations: raising k above 0, which represents a forced system brownout, or v descending below .9 p.u., a condition likely leading to system collapse. This cost is much higher than the others.

The system is simple enough that for given $x(t)$ and $u(t^+)$, $y(t^+)$ can be determined explicitly except for one first order zero-search algorithm. Further, this allows an explicit calculation of $\dot{x}(t^+)$, and with only one more search, a future value of y. These two values of y are used for a linear predictive model to give the cost A_1 over a thirty second horizon before u is changed again. Over this horizon, there is an error of less than 1.5%.

An exhaustive optimization method would examine the cost of every possible combination of input parameters over a long horizon. Given, however, that there are 60 possible input combinations every thirty seconds, the number of computations required for an intelligent forecast quickly rises into the thousands. Given the goal of applying control methods to more complex systems, this method is far too computationally expensive.

Instead, because of large differences in cost (and in effect on v), each input variable can be optimized independently. k can be indepentently constrained to the conditions that v does not descend below .9 p.u. within the next thirty seconds and that it is ultimately stabilizible above .9 p.u. This second condition is verifiable by comparing $\dot{x}_p$ to x_p, and extrapolating a corresponding minimum v.

μ can also be calculated independently, because it consistently causes much larger changes in v than does n (which can only be changed incrementally). The cost of a change in μ and of voltage deviation can be optimized simply by comparing the cost from all five choices for μ. For this system, μ was assigned a value that would allow it to change if it decreased the average deviation by at least .02 p.u.

Being the only variable left, n can then be directly optimized with the voltage deviation. Since, at every thirty second mark, there are only three choices (k and μ generally remaining constant after an intial shift), n can be optimized for a rather large horizon. 120 seconds requires only several hundred computations at most. By using a search optimization tree that halts queries that have already incurred a cost greater than the minimum, this number is further reduced. Once the system has stabilized, there are usually no more than a dozen calculations. For this system, n was assigned a value that would allow it to change if it decreased the average deviation by at least .0067 p.u.

There is one potentially fatal problem with this control method: an accurate prediction of the deviation cost requires knowledge of the next value of every input. By placing the constraint violation optimization last, we can ensure that this optimization will be correct. Fortunately, μ and n are being optimized by comparative cost, rather than by absolute deviation. As they both have monotonic effects on v, so it is only necessary to know the exact value of v around 1 p.u., a range k will generally not be changing anyway.

3 Simulation and Results

For this project, the system was simulated in a Matlab environment, using M-files to optimize the controls and as a frame for the C++ code that did the runtime simulation between the control changes every thirty seconds.

In order to test the capabilities of the segregated control method, we tested the limits of the systems ability to stabilize after jumps in the line impedance. In every case, the system began at an impedance of $X = .25$ p.u., and was raised to some crisis value.

Figure 2 shows the voltage and input parameters over time, after a jump at 10 seconds.

max k	max X(p.u.)
0.15	0.6387
0.10	0.6194
0.05	0.5760
0.00	0.5372

Table 1: Max stabilizable Impedance by Load Shedding Cost

X	Segregated Cost	Exhaustive Cost
.45	11.47	11.47
.50	273.4	274.2
.55	1676	1679
.60	3160	3195

Table 2: Cost of Segregated vs. Exhaustive Search

As shown in table 1, repeated simulation also allowed us to compare the limits of the system for given cost limits - determined by the maximum final value of k required for stabilization.

A comparison with an exhaustive search method (analyzing every option and choosing the one with the lowest cost, but only over a 30 second horizon), demonstrates the superiority of the search algorithm we used. Not only is it much faster once the system has stabilized, but it generally produces a lower cost, as shown in table 2.

4 Conclusion and Future work

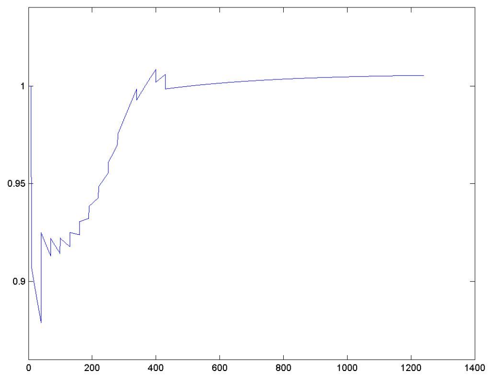
The segregated search algorithm presented in this paper allowed to provide a controller that exhibits both high performances and low computation costs. The size of the search tree of the optimal control sequence was indeed heavily reduced by considering the nature of each input variable along with the use of classical search heuristics. Nothing [1] prevents a priori the same methodology to be applied to more complex power systems like the medium scale ABB testcase. In order to develop less problem specific methods, we are also currently investigating a way to compute off-line, using dynamic programming like algorithms, a decision function that would help guiding the search and thus reducing further the size of the search tree.

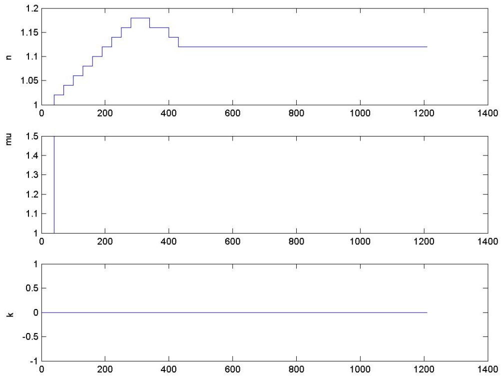
5 References

Larsson, Mats. (2002). A Simple Test System Ilustrating Load-Voltage Dynamics in Power Systems. Corporate Research, ABB Schweiz AG.

[1] but implementation complexity considerations

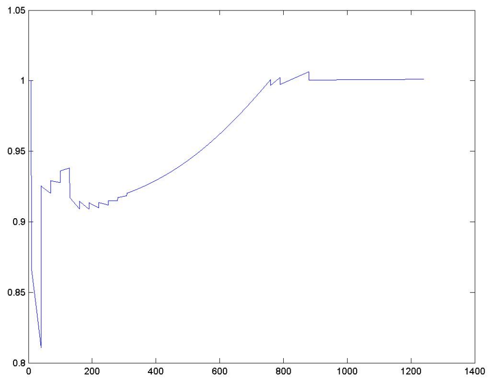
X jumps to .50 p.u. : Voltage is stabilised without resort to load shedding (k remains 0)

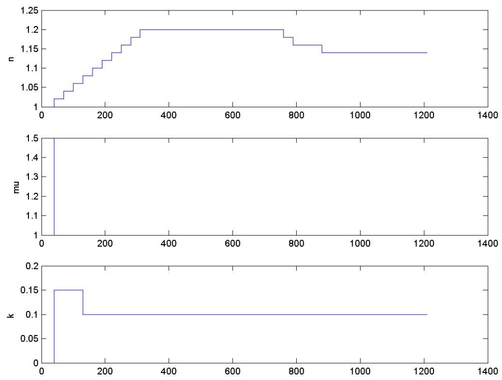




Voltage vs Time Inputs vs. Time

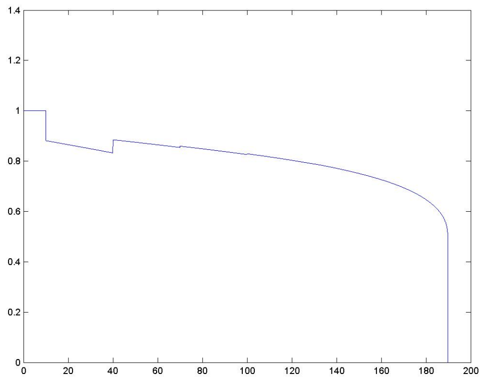
X jumps to .60 p.u. : load shedding has to be used to stabilise the system

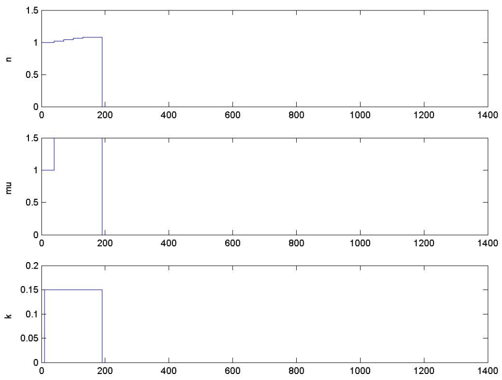




Voltage vs Time Inputs vs. Time

X jumps to .70 p.u. : nothing can be done to prevent collapse





Voltage vs Time Inputs vs. Time

Figure 2: Simulation plots : recall that the objective is to stabilise voltage around 1

A Control Code

Below is the code that selects the control variables every thirty seconds, using a segregated control scheme:

```
%change controls
    %capacitor controls
    for cap_con = 1:5,
        if (MuVals(cap_con) > mu) && (Y(s,3) + 3000*(Y(s,3)-Y(s-1,3)) < .98)
            [v(1),delta(1),cap_cost(cap_con)] = cost_calc(Y(s,1),Y(s,2),n(1),MuVals(c
        elseif (MuVals(cap_con) < mu) && (Y(s,3) + 3000*(Y(s,3)-Y(s-1,3)) > 1.02)
            [v(1),delta(1),cap_cost(cap_con)] = cost_calc(Y(s,1),Y(s,2),n(1),MuVals(c
        elseif MuVals(cap_con) == mu
            [v(1),delta(1),cap_cost(cap_con)] = cost_calc(Y(s,1),Y(s,2),n(1),mu,k,X,3
        else
            cap_cost(cap_con) = 3000;
        end
    end
    mu = MuVals(cap_cost == min(cap_cost));

    % tap ratio controls
    nn = [n(1) n(1)+TapStep n(1)-TapStep];

    [v(1),delta(1),cost(1)] = cost_calc(Y(s,1),Y(s,2),nn(1),mu,k,X,Horizon,0,3000);
    if(n(1)<1.2)
        [v(2),delta(2),cost(2)] = cost_calc(Y(s,1),Y(s,2),nn(2),mu,k,X,Horizon,TapCos
    else
        cost(2) = cost(1);
    end
    if(n(1)>.8)
        [v(3),delta(3),cost(3)] = cost_calc(Y(s,1),Y(s,2),nn(3),mu,k,X,Horizon,TapCos
    else
        cost(3) = cost(2);
    end
    n = nn(cost == min(cost));

    %load shedding controls
    k = constraint_viol(Y(s,1),Y(s,2),n(1),mu,X,v_limits);
```

B Constraint Violation Code

This segment of code is used to determine k, incrementing it until a stable solution is reached:

```
while ((v < .9) | (v_next < .9) | unstable) && (k < .15)
    k = k+.05;
    step = step + 1;
    [v delta] = ABB_solver(xp,xq,n,mu,k,X,.1);
    dxp = -xp/Tp + P0*(1-v^2);
    dxq = -xq/Tq + Q0*(1-v^2);
    xp_next = xp + dxp/10;
    xq_next = xq + dxq/10;
    [v_next delta_next] = ABB_solver(xp_next,xq_next,n,mu,k,X,.1);
    dv = v_next - v;
```

```
    dxp_next = -xp_next/Tp + P0*(1-v_next^2);
    v_next = v + 300*dv;
    unstable = (v < v_limits(step));
end
```