AN ABSTRACT OF THE THESIS OF

Michael P. Antonishen for the degree of Master of Science in Electrical and Computer Engineering presented on June 8, 2012.

Title: Use of a Model Predictive Control Framework for Optimal Control of Grid Scale Electrical Energy Storage in Conjunction with a Wind Farm.

Abstract approved: _____________________________________________________

                          Ted K.A. Brekken

Over the last decade, wind penetration in the Pacific Northwest has increased rapidly. The variable nature of this massive new resource has increased stress on the hydropower resource to the point where system limits are currently being reached. In order to cultivate continued growth of the wind energy industry both in the Pacific Northwest and the rest of the world, something must be added to help mitigate the effects of the variability of wind power. This research aims to show what can be done by adding energy storage to a wind farm. A novel model predictive control structure has been created with the focus of increasing the dispatchability and reliability of wind farm power output along with allowing participation in frequency regulation. First, the effectiveness of the addition of energy storage with simple control is explored. This is followed by a study on the performance of the system when predictive control is added. Finally, a cost analysis is performed to assess the level of savings and potential profitability of the simulated system. Conclusions support the use of an energy storage resource for more reliable wind farm performance. However, storage technologies are still approaching the price point needed to ensure profitability.
© Copyright by Michael P. Antonishen
June 8, 2012
All Rights Reserved
Use of a Model Predictive Control Framework for Optimal Control of Grid Scale Electrical Energy Storage in Conjunction with a Wind Farm

by
Michael P. Antonishen

A THESIS

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Master of Science

Presented June 8, 2012
Commencement June 2012
Master of Science thesis of Michael P. Antonishen presented on June 8, 2012

APPROVED:

Major Professor, representing Electrical and Computer Engineering

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

__________________________
Michael P. Antonishen, Author
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ted Brekken, for all of the time, support, and teaching he bestowed upon me over the course of my education. Additionally, I would like to thank Dr. von Jouanne for her guidance and Dr. Yokochi for the tough questions and insights.

Thank you also to Doug Halamay, not only has it been a joy to work with you but I have learned so much from your experience. To my fellow graduate students, I could not ask for a better work environment or better people to be around. I appreciate all of the collaboration and ideas.

Thank you to Bonneville Power Administration for funding my research and to my advisors for securing grants and enabling us graduate students to do our best research.

Finally thank you to my friends and family here in Corvallis, in Michigan, and abroad. Your support is priceless.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction................................................................. 1</td>
</tr>
<tr>
<td>1.1 Background ........................................................................ 1</td>
</tr>
<tr>
<td>1.2 Scope of Thesis .............................................................. 6</td>
</tr>
<tr>
<td>2 Proposed Control Structure.................................................. 7</td>
</tr>
<tr>
<td>2.1 Model Predictive Control Formulation .................................. 8</td>
</tr>
<tr>
<td>2.2 Model Predictive Control Implementation ............................. 10</td>
</tr>
<tr>
<td>3 Simulation .......................................................................... 12</td>
</tr>
<tr>
<td>3.1 Simulation Inputs and Test Program ..................................... 12</td>
</tr>
<tr>
<td>3.2 Simulation Results ................................................................. 16</td>
</tr>
<tr>
<td>3.2.1 Participation in Frequency Response ................................. 17</td>
</tr>
<tr>
<td>3.2.2 No Energy Storage ............................................................ 19</td>
</tr>
<tr>
<td>3.2.3 Energy Storage with No Predictive Control ....................... 24</td>
</tr>
<tr>
<td>3.2.4 Energy Storage with Predictive Control ............................. 26</td>
</tr>
<tr>
<td>3.2.5 Finding Real Time Damage Incurrence Cost R that Minimizes</td>
</tr>
<tr>
<td>Overall Cost of Operation J ...................................................... 31</td>
</tr>
<tr>
<td>3.3 Synthesis of Results ............................................................ 36</td>
</tr>
<tr>
<td>4 Conclusion .......................................................................... 40</td>
</tr>
<tr>
<td>4.1 Future Work ...................................................................... 41</td>
</tr>
<tr>
<td>5 Bibliography ..................................................................... 42</td>
</tr>
<tr>
<td>Appendices .............................................................................. 45</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX 1</td>
<td>Main Simulation File Code</td>
<td>46</td>
</tr>
<tr>
<td>APPENDIX 2</td>
<td>Constraint Making Function Code</td>
<td>62</td>
</tr>
<tr>
<td>APPENDIX 3</td>
<td>Make E_k Vector Code</td>
<td>63</td>
</tr>
<tr>
<td>APPENDIX 4</td>
<td>Make P_agc Code</td>
<td>64</td>
</tr>
<tr>
<td>APPENDIX 5</td>
<td>Make P_loss Code</td>
<td>65</td>
</tr>
<tr>
<td>APPENDIX 6</td>
<td>Make Pwind Code</td>
<td>66</td>
</tr>
<tr>
<td>APPENDIX 7</td>
<td>Make Q Cost Matrix Code</td>
<td>75</td>
</tr>
<tr>
<td>APPENDIX 8</td>
<td>Make R Cost Matrix Code</td>
<td>76</td>
</tr>
<tr>
<td>APPENDIX 9</td>
<td>Make T_k Tracking Vector Code</td>
<td>77</td>
</tr>
<tr>
<td>APPENDIX 10</td>
<td>Make V_k Vector Code</td>
<td>78</td>
</tr>
<tr>
<td>APPENDIX 11</td>
<td>State Space Model Code</td>
<td>79</td>
</tr>
<tr>
<td>APPENDIX 12</td>
<td>Plotting Code</td>
<td>80</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 1.1. Installed wind capacity in the Pacific Northwest from 1998 to 2012, plot taken from (“BPA Wind Generation Capacity,” BPA).</td>
<td>2</td>
</tr>
<tr>
<td>Fig. 1.2. Historical energy production in the Pacific Northwest, from 2002 to 2010 (Dragoon).</td>
<td>3</td>
</tr>
<tr>
<td>Fig. 1.3. Currently available energy storage technologies with time scale and power information. Figure reproduced using information from (“Energy storage systems for transport and grid applications,” Vazquez).</td>
<td>5</td>
</tr>
<tr>
<td>Fig. 2.1. MPC controller organization shown with a block diagram.</td>
<td>8</td>
</tr>
<tr>
<td>Fig. 3.1. Simulation with energy storage off, persistence forecasting, 60 minute scheduling, and AGC on. Corresponds to case 1 in Table 3.1 and Table 3.3.</td>
<td>19</td>
</tr>
<tr>
<td>Fig. 3.2. Simulation with energy storage off, meteorological forecasting, 60 minute scheduling, and AGC on. Corresponds to case 2 in Table 3.1 and Table 3.3.</td>
<td>20</td>
</tr>
<tr>
<td>Fig. 3.3. Simulation with energy storage off, meteorological forecasting, 30 minute scheduling, and AGC on. Corresponds to case 3 in Table 3.1 and Table 3.3.</td>
<td>21</td>
</tr>
<tr>
<td>Fig. 3.4. Simulation with energy storage off, perfect forecasting, 60 minute scheduling, and AGC on. Corresponds to case 4 in Table 3.1 and Table 3.3.</td>
<td>22</td>
</tr>
<tr>
<td>Fig. 3.5. Comparison of error in cases 1-4 over one day of the testing period. Case numbers correspond to those listed in Table 3.1.</td>
<td>24</td>
</tr>
<tr>
<td>Fig. 3.6. Simulation with energy storage on, meteorological forecasting, 60 minute scheduling, and AGC on. Corresponds to case 5 in Table 3.1.</td>
<td>25</td>
</tr>
<tr>
<td>Fig. 3.7. Bar plot showing MAE for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.</td>
<td>28</td>
</tr>
<tr>
<td>Fig. 3.8. Bar plot showing $J_Q$ cost for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.</td>
<td>29</td>
</tr>
<tr>
<td>Fig. 3.9. Bar plot showing following reserves as a percent of base case (case 2) for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.</td>
<td>30</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>Fig. 3.10. Bar plot showing imbalance reserves as a percent of base case (case 2) for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.</td>
<td>31</td>
</tr>
<tr>
<td>Fig. 3.11. Plot of costs with respect to the R input cost for a week long simulation. In this simulation meteorological forecasting with 60 minute scheduling was used. Frequency response is turned on, and predictive control with an HP of 6 was used.</td>
<td>34</td>
</tr>
<tr>
<td>Fig. 3.12. Total savings from base case for each point in the R sweep seen in Fig. 3.11.</td>
<td>35</td>
</tr>
<tr>
<td>Fig. 3.13. Visual comparison of mean absolute error (MAE) and J cost for the best case versus base case. This illustrates maximum error reduction and cost savings.</td>
<td>37</td>
</tr>
<tr>
<td>Fig. 3.14. Plot of important variables for case 8.5. $P_{plant} = P_{farm} + P_{es}$. The difference between $P_{sched}$ and $P_{plant}$ illustrates the difference between total plant power output and scheduled power output.</td>
<td>38</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1. Program of all simulations run.</td>
<td>15</td>
</tr>
<tr>
<td>Table 3.2. Comparison of the base case (case 0) and case 2 with everything else held constant and participation in frequency response turned on.</td>
<td>18</td>
</tr>
<tr>
<td>Table 3.3. Comparison of error, costs, and reserve requirements for each of the cases shown. Numbers correspond to those shown in Table 3.1.</td>
<td>23</td>
</tr>
<tr>
<td>Table 3.4. Error, costs, and reserve requirements for the case with energy storage activated but no predictive control. Case number corresponds to Table 3.1.</td>
<td>26</td>
</tr>
<tr>
<td>Table 3.5. Error, costs, and reserve requirements for the case with energy storage and predictive control sweeping $Hp$. Case numbers correspond to Table 3.1.</td>
<td>27</td>
</tr>
<tr>
<td>Table 3.6. Cost information for simulations run to sweep $R$ input with all else held constant. Case numbers refer to the key in Table 3.1.</td>
<td>33</td>
</tr>
<tr>
<td>Table 3.7. Performance metric comparison for best case (case 8.5) vs. base case (case 2) based on total $JQ$ cost.</td>
<td>36</td>
</tr>
</tbody>
</table>
Use of a Model Predictive Control Framework for Optimal Control of Grid Scale Electrical Energy Storage in Conjunction with a Wind Farm

1 Introduction

1.1 Background

Over the last forty years the world has come to a new realization on the finite nature of fossil fuel resources. In response to this, a number of promising renewable energy technologies including wind, solar, and wave energy have been developed. While both solar and wave power are still in development and approaching profitable levels for large scale deployment, worldwide wind energy capacity is growing rapidly. However, the variable nature of wind has limited grid penetration due to increased operating costs from amplified load following by other resources.

In the Pacific Northwest, installed wind capacity has been on the rise exponentially since 1998 (“BPA Wind Generation Capacity,” BPA). Fig. 1.1 illustrates the feverish pace of wind installation in the Pacific Northwest in recent years.
As wind penetration in the Pacific Northwest has grown, the region has been introduced to many novel problems due to the fact that power from the wind cannot be commanded; it is a non-dispatchable resource. Since power is both generated and consumed in the same instant, any balanced power system must be able to match power consumption with power production. Additionally, the current power market has almost no demand-side participation. It is up to the regional balancing authorities to ensure that generation matches load at all times. Therefore, since wind is a non-dispatchable resource, some other form of generation must be performing load following on its behalf. In the Pacific Northwest this other form of generation is hydropower. The fast load following that hydropower provides in the Pacific
Northwest is commonly done with natural gas in other parts of the world. Other forms of generation like nuclear, geothermal, and coal are not as capable of providing fast response in an economical manner.

![Energy Production Chart](image)

Fig. 1.2. Historical energy production in the Pacific Northwest, from 2002 to 2010 (Dragoon).

Fig. 1.2 shows the historical breakdown of energy production in the northwest from 2002-2010. According to the Northwest Power and Conservation Council, hydropower makes up 56.07% of online generating capacity as of February 2012 with wind accounting for 11.81%. While the amount of wind power capacity online is small compared to hydropower, it is still causing problems due to the numerous limits, such as irrigation and biological concerns, that affect hydropower generation. (Dragoon)
Most of the installed wind in the Pacific Northwest is in the Columbia River Gorge, a relatively small geographical area (“Current and proposed wind project interconnections to BPA transmission facilities,” BPA). This means that when the wind suddenly picks up in this area, power generation from wind can go from nearly 0 MW up to 4421 MW in a matter of hours, meaning that up to 4421 MW of generation must be reduced somewhere else on the system in a short period of time. Additionally, like all other forms of generation, wind farms are required to declare the amount of power they will sell before it is generated so the grid operator can schedule the power production to meet demand. This is currently a source of large error. Finally, wind farms do not currently participate in any form of frequency regulation. All of these factors along with numerous environmental and irrigation concerns that hydropower operators must address have led the Bonneville Power Administration (BPA) to state that “the hydropower system’s limits are being reached” (“Connecting variable generating resources to the federal Columbia river transmission system (FCRTS),” BPA).

One of the most promising solutions to these problems is the emerging technology of grid-scale electrical energy storage. As of 2010, US generating capacity was 1,088 GW and storage capacity only 22 GW (Johnson). If substantial amounts of energy storage were to be developed, especially in areas with large amounts of non-dispatchable generation, the stress on the rest of the system would be reduced. A good analogy is that energy storage is essentially a warehouse in a supply chain that currently has none (Johnson). The addition of energy storage would allow excess
energy to be stored to meet future demand, reducing the strain this load following normally puts on dispatchable generators.

Fig. 1.3. Currently available energy storage technologies with time scale and power information. (Figure reproduced using information from “Energy storage systems for transport and grid applications,” Vazquez).

Fig. 1.3 shows power capacity and time of storage capability for a number of available energy storage technologies. While it may look like a diverse set of options, few technologies have been able to match the scale of pumped hydropower and compressed air energy storage. The technologies of particular interest in this research are those that exist or could potentially exist in the red box, the utility scale applications which are capable of high power capacity, high energy storage volume and long storage time. Currently, many technologies are capable of high power storage but lack the volume needed to be effective at grid level. Additionally, high
capital cost or low cycle life currently prevents these technologies from being economically feasible. However, even pumped hydropower is limited geographically.

1.2 Scope of Thesis

This paper will focus on simulations that combine a wind farm with a grid level energy storage source in the form of a Zinc Bromine Battery (ZBB) system. The addition of the energy storage source is meant to increase the dispatchability and reliability of the wind farm while also helping it to participate in frequency regulation just like normal dispatchable generation sources. A novel model predictive control system has been developed for use with this combined system.

Previous work has shown that energy storage can be used to increase wind power predictability and reduce variability (Han), (Thatte). Additionally, many other control structures and theories have been tested and the results suggest that the combination of energy storage and wind power is technically and financially feasible (Han), (Baone), (Mendis), (Banham-Hall), (Fazeli).

Explorations of the cost of participation in frequency response, the effectiveness of the addition of energy storage with and without predictive control, the length of prediction horizon, and different types of power scheduling and forecasting will be discussed. Findings from an experiment meant to find a breakpoint cost for where energy storage will be economical will be presented. Results will be quantified in terms of mean absolute error (MAE), dollars saved based on the current charges wind farms face, and reserve requirements.
2 Proposed Control Structure

In these experiments, a novel approach has been taken by implementing a predictive controller. This control structure is generally called model predictive control (MPC) and can be implemented in many different ways. In this formulation, mathematical models have been created to predict future values of disturbances to the plant being controlled, which is also modeled. These predictions are then fed into a controller that optimizes the actions taken by the plant according to weights specified in an objective function. After the control actions for a certain prediction horizon have been optimized, the first control action of the optimized sequence is implemented in the real plant. This process is then repeated iteratively for each additional step.

One of the benefits of having a predictive controller, as opposed to a reactive controller like the one implemented in (Antonishen) is that the system can take preparatory actions to mitigate the negative effects of events such as wind ramps before they happen. Additionally, the model predictive control structure contains information on system limits and the cost of control actions. With all of this information, the model predictive controller can mathematically optimize control actions over a future control horizon according to the real cost of any specific control action or error caused by lack of control action.

Fig. 2.1 shows a block diagram of the MPC controller including inputs, system control, and outputs. The controller takes inputs in the form of wind data, wind predictions, frequency data, and frequency predictions and simply outputs an energy storage power command. There are additional inputs built into the controller that
remain constant, such as system limits and costs associated with certain control commands or lack of control commands. Inputs will be discussed more in section 3.1.

Fig. 2.1. MPC controller organization shown with a block diagram.

### 2.1 Model Predictive Control Formulation

Before presenting the MPC formulation and specific state space matrices used for these experiments, the expression of the plant, shown in state space form, incremented generally over a prediction horizon ($H_p$) must be discussed. In discrete time, the state update equation can be seen in (2.1) and the output equation in (2.2).

\[
x(k + 1) = Ax(k) + Bu(k) + B_vv(k)
\]

(2.1)

\[
y(k) = Cx(k) + Du(k) + D_vv(k)
\]

(2.2)

In (2.1), $x(k + 1)$ gives the system state one step in the future, $A$ is the state matrix, $x(k)$ is the current system state, $Bu$ is the input control matrix, $u(k)$ is the input control vector, $B_v$ is the input disturbance matrix, and $v(k)$ is the input disturbance vector. In (2.2), $y(k)$ is the output vector, $C$ is the output matrix, $Du$ is the feed through input control matrix, and $D_v$ is the feed through input disturbance.
matrix. Incrementing both of these by $H_p$ steps and substituting each $x(k)$ into the $y(k)$ equation yields (2.3) which with the underlined variable names gives (2.4). The derivation of (2.3) and (2.4) comes from (Rossiter).

\[
\begin{pmatrix}
  y(k) \\
  y(k + 1) \\
  y(k + 2) \\
  \vdots \\
  y(k + H_p)
\end{pmatrix}
\begin{pmatrix}
  C \\
  CA \\
  CA^2 \\
  CA^3 \\
  \vdots \\
  CA^{H_p}
\end{pmatrix}
\begin{pmatrix}
  x(k) \\
  \tilde{y}(k)
\end{pmatrix}
\]

\[
\begin{pmatrix}
  Du & 0 & 0 & 0 & \cdots & 0 \\
  CB_u & Du & 0 & 0 & \cdots & 0 \\
  CAB_u & CB_u & Du & 0 & \cdots & 0 \\
  CA^2B_u & CAB_u & CB_u & Du & \cdots & 0 \\
  \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
  CA^{H_p-1}B_u & CA^{H_p-2}B_u & \cdots & CB_u & Du
\end{pmatrix}
\begin{pmatrix}
  u(k) \\
  u(k + 1) \\
  u(k + 2) \\
  u(k + 3) \\
  \vdots \\
  u(k + H_p)
\end{pmatrix}
\]

\[
\begin{pmatrix}
  v(k) \\
  v(k + 1) \\
  v(k + 2) \\
  v(k + 3) \\
  \vdots \\
  v(k + H_p)
\end{pmatrix}
\]

\[
\tilde{y}(k) = S_x x(k) + S_u \tilde{u}(k) + H_v \tilde{v}(k)
\]  

The MPC controller used in these experiments was formulated using a traditional linear quadratic regulator (LQR) objective function with tracking as seen in (2.5) where $t(k)$ gives the tracking information for the output variables, $y(k)$ is the
system output, \( Q \) is the output cost matrix, \( u(k) \) is the input vector, and \( R \) is the input cost matrix.

\[
J(k) = (\vec{u}(k) - \vec{Y}(k))^T Q (\vec{u}(k) - \vec{Y}(k)) + \vec{u}(k)^T R \vec{u}(k)
\]  

(2.5)

In order to manipulate (2.5) and put it in a form that the MATLAB quadprog solver can handle, a new tracking error variable \( e(k) \) is defined, shown in (2.6), which yields the final result shown in (2.7). The specific matrices used in this formulation are covered in the next section.

\[
\vec{e}(k) = \vec{t}(k) - (\vec{Y}(k) - S_u \vec{u}(k)) = \vec{t}(k) - (S_x \vec{x}(k) + H_u \vec{v}(k))
\]

(2.6)

\[
\vec{J}(k) = \frac{1}{2} \vec{u}(k)^T (S_u^T Q S_u + R) \vec{u}(k) - \vec{e}(k)^T Q S_u \vec{u}(k)
\]

(2.7)

2.2 Model Predictive Control Implementation

In this specific MPC implementation, the discrete time state space matrices that are applied to the system can be seen in (2.8) and (2.9).

\[
\begin{bmatrix}
    SOC \\
    P_{ES,-1}
\end{bmatrix}_{k+1} = \begin{bmatrix}
    1 & 0 \\
    0 & 0
\end{bmatrix} \begin{bmatrix}
    SOC \\
    P_{ES,-1}
\end{bmatrix}_{k} + \begin{bmatrix}
    -m_D \\
    1
\end{bmatrix} \begin{bmatrix}
    P_{ES}
\end{bmatrix}_{k} + \begin{bmatrix}
    0 & 0 & 0 & 0 & -m_D
    0 & 0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
    P_{AGC} \\
    P_{sched} \\
    P_{farm} \\
    \Delta P_{farm} \\
    P_{loss}
\end{bmatrix}_{k}
\]  

(2.8)

\[
\begin{bmatrix}
P_{error} \\
\Delta P_{ES} \\
\Delta P_{plant} \\
SOC \\
SOC_{next}
\end{bmatrix}_{k} = \begin{bmatrix}
0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
SOC \\
P_{ES,-1}
\end{bmatrix}_{k} + \begin{bmatrix}
-1 \\
1 \\
1 \\
1 \\
-1
\end{bmatrix} \begin{bmatrix}
P_{ES}
\end{bmatrix}_{k} + \begin{bmatrix}
1 & 1 & -1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -m_D
\end{bmatrix} \begin{bmatrix}
P_{AGC} \\
P_{sched} \\
P_{farm} \\
\Delta P_{farm} \\
P_{loss}
\end{bmatrix}_{k}
\]  

(2.9)

In (2.8), \( SOC \) represents the state of charge of the energy storage resource being used, \( SOC_{next} \) represents the next \( SOC \) based on the losses and current energy
storage command, \( P_{ES_{-1}} \) is the energy storage command from the last time step, \( m_\Delta \) is the normalized rate of change of the SOC, \( P_{ES} \) is the current energy storage command, \( P_{AGC} \) is the power required to meet the plant’s automatic generation control (AGC), also referred to as frequency regulation, obligation, \( P_{sched} \) is the scheduled power output of the wind farm, \( P_{farm} \) is the actual output of the wind farm, and \( P_{loss} \) is the amount of leakage loss in the energy storage resource. In (2.9) \( P_{error} = (P_{AGC} + P_{sched}) - P_{plant} \), and \( P_{plant} \) is the combined wind farm and energy storage output.

In these simulations, the time step is ten minutes. Since the transients seen in energy storage hardware are on the order of seconds, the energy storage dynamics state equation seen in (2.8) does not need to include transient information. Both the SOC slope and \( P_{es} \) change instantaneously with commands on the ten minute interval.

This model has been defined so that specific constraints can be applied to each output as well as each input. These constraints give the model limits so that it acts like the real system. Additionally, through \( Q \) and \( R \), costs can be applied to the tracking error in each output variable as well as each control action taken, respectively.
3 Simulation

3.1 Simulation Inputs and Test Program

The specific input data for this model came from a number of sources. Frequency data was taken from a large wind farm in the Pacific Northwest. Wind farm power forecast and actual data came from the BPA wind power website (BPA). At this time, adequate meteorological forecast data correlated to historical power data for a single wind farm is unavailable to the researcher, so aggregate data was used to represent a single wind farm. The energy storage sizing follows what was found in previous research, 25% power and 50% energy capacity (Han). For simulations with a 100 MW wind farm, 25 MW (power) and 50 MWh (energy) of energy storage is appropriate.

In order to execute predictive control, models must also be used to predict the future activity of wind and frequency over the control horizon. In the case of frequency, a simple persistence model was used since frequency plays only a minor role and remains extremely close to a nominal 60 Hz. For wind, both a forecast and a projected schedule were needed. The forecast is defined as the best prediction of what the actual wind farm power output will be over the prediction horizon and the schedule is defined as the power output that the wind farms say they will adhere to over a certain scheduling length. The difference between forecast and schedule, scheduling error, is of paramount importance in this research as it is the main source of error that the controller seeks to correct optimally. Several different combinations
of data, forecast, and schedule combinations were made available for experimentation and can be seen in Table 3.1.

The input costs for $Q$ and $R$, used in the cost function seen in (2.7) and applied to (2.8) and (2.9), were derived as much as possible from reality. For the purpose of these experiments, the only $Q$ cost which was not zero was $P_{error}$. The cost of $P_{error}$ in real time was derived from the BPA variable energy resource balancing service (VERBS) which is currently charged to wind farms monthly based on installed capacity. The VERBS charge is currently $1.23$ per kW/mo of capacity and can be subdivided into regulating reserves for balancing moment to moment ($0.08$ per kW/mo), following reserves for balancing larger mismatches within the hour ($0.37$ per kW/mo), and imbalance reserves for balancing difference between the generators schedule and the actual generation during an hour ($0.78$ per kW/mo) (“2012 BPA final rate Proposal, Generation inputs study,” BPA).

In order to make VERBS into a real time cost, scheduling error was summed over 4 months for a 100 MW wind farm in the Pacific Northwest. This error was then divided by the cost for 4 months of the VERBS service. The end result is a numerical value with units of $\$/PU^2$ that can be used in $Q$ to represent real time cost incurrence of scheduling error. Since this is a study to explore the impacts of different costs of energy storage as well as the effectiveness of a model predictive controller, $R$ was left as a free variable.

Participation in frequency regulation followed standards set forth by the North American Electric Reliability Corporation (NERC) for primary control. Primary
control is also commonly known as frequency response and is an action taken within seconds of a change in system frequency to stabilize and arrest frequency deviations. It was set up this way for the purpose of our simulations. For a deviation of 1 Hz a power command of .33 pu would be given. (NERC)

Losses were included in the state space matrix as a constant power drain on the state of charge (SOC) and modeled after the chemical leakage losses of the Zinc Bromine Battery in the Wallace Energy Systems and Renewables Facility (WESRF) lab. Approximately 1% of the total energy capacity is lost each hour except at SOC levels below 2% where losses become extremely low.

The only constraints used for this study were placed on the input command of $P_{es}$ and the output variable SOC. $P_{es}$ was limited between .25 and -.25 pu based on the limits of energy storage resources available in the Wallace Energy Systems and Renewables Facility, and SOC had realistic limits of 1 and 0 pu representing full and empty state of charge, respectively.

Due to both the use of the BPA aggregate data source and the reality of dealing with a large energy capacity storage system, such as a flow cell battery, all simulation was performed with a 10 minute sample time and each simulation covers a week in real time. This sample time is commonly used for larger size power corrections and balancing and can be supported by current large scale energy storage technologies available in the WESRF laboratory (NERC).
Table 3.1. Program of all simulations run.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Forecast</th>
<th>Schedule</th>
<th>ES</th>
<th>$H_p$</th>
<th>$Q \ Cost$</th>
<th>$R \ Cost$</th>
<th>$FR$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Meteorological</td>
<td>1 Hour</td>
<td>Off</td>
<td>NA</td>
<td>VERBS</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>Persistence</td>
<td>1 Hour</td>
<td>Off</td>
<td>NA</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>2</td>
<td>Meteorological</td>
<td>1 Hour</td>
<td>Off</td>
<td>NA</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>3</td>
<td>Meteorological</td>
<td>30 Min</td>
<td>Off</td>
<td>NA</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>4</td>
<td>Perfect</td>
<td>1 Hour</td>
<td>Off</td>
<td>NA</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>5</td>
<td>Meteorological</td>
<td>1 Hour</td>
<td>On</td>
<td>0</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>6.1-6.5</td>
<td>Persistence</td>
<td>1 Hour</td>
<td>On</td>
<td>6,12,18,24,30</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>7.1-7.5</td>
<td>Meteorological</td>
<td>1 Hour</td>
<td>On</td>
<td>6,12,18,24,30</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>8.1-8.5</td>
<td>Meteorological</td>
<td>30 Min</td>
<td>On</td>
<td>6,12,18,24,30</td>
<td>VERBS</td>
<td>0</td>
<td>On</td>
</tr>
<tr>
<td>9.1-9.28</td>
<td>Meteorological</td>
<td>1 hour</td>
<td>On</td>
<td>6</td>
<td>VERBS</td>
<td>Sweep</td>
<td>On</td>
</tr>
</tbody>
</table>

Table 3.1 shows which tests were chosen to be run for this exploration. $ES$ is energy storage, $FR$ is frequency response. With so many different combinations of variables, it was important to choose only those that answered questions of interest. In testing, three different types of wind forecasts were used. Persistence is the simplest and assumes that whatever the wind is doing at a chosen moment, it will continue to do over the next horizon. The meteorological forecast was obtained as a part of the BPA wind data set and is based on meteorological principles and models run by BPA. Finally, perfect forecasting is done by using the future data as the forecast. The cases using perfect forecasting are demonstrations of both how well the controller can work
and how little energy storage input is needed when wind forecasting is perfect. The two different schedules being explored are 1 hour, which holds steady for 40 minutes and ramps to the next level for 20 minutes, and 30 minutes, which holds steady for 20 minutes and ramps to the next level for 10 minutes.

First, a simple exploration of frequency response was done by comparing case 0 to case 2. Next, there are several base cases established first with energy storage off (cases 1-4). The horizon, $H_p$, is marked as NA (not applicable) in these cases because control actions are not taken as there is nothing to control. For the cases where energy storage is turned on there is an exploration with no predictive control ($H_p = 0$) representing a simple reactive controller (case 5) and then many cases with different forecasts and schedules to explore the use of different prediction horizons and scheduling types with this controller (cases 6.1-8.5). Finally, an exploration of the total cost $J$ is done with many different costs ascribed to $R$ to see how this affects control decisions (cases 9.1-9.28). For all tests, the energy storage state of charge was chosen to start at 50%.

### 3.2 Simulation Results

All of the following simulations were run with the same frequency and wind data sets based on what happened during the week spanning from Monday, March 26th 2012 to Monday April 2nd 2012. A large amount of discussion will be focusing on the $J_Q$ cost, which is directly related to $Q$ cost. With the cost applied to $R$ held at zero for most simulations, successful minimization of this $J_Q$ cost can be directly correlated with control actions that minimize error and reserve requirements. Both error and
reserve requirements are also used to analyze the effectiveness of energy storage and explore the potential cost savings of the implementation of such a system.

There are two different types of reserves measured in these simulations. Following reserves are calculated by taking the difference between hourly average power and 10-minute power while imbalance reserves are the difference between the scheduled hourly power and the hourly average power actually produced. In Table 3.2, the incremental reserve (Inc) represents the maximum of either following or imbalance over the time series while (Dec) represents the minimum over the time series. The Inc and Dec give the amount of additional generation, held in reserve, that must be available to either turn on (Inc) or shut off (Dec) to balance wind fluctuations. It should be noted that according to standard industry practices, the top and bottom 0.25% of outliers are first eliminated from the data set before performing these calculations. (“2012 BPA final rate Proposal, Generation inputs study,” BPA), (“Reserve Capacity Forecast For Wind Generation Within-Hour Balancing Service,” BPA)

### 3.2.1 Participation in Frequency Response

As noted before, participation in frequency response, also known as primary control (NERC), was done according to standards set forth by the NERC. The frequency bias used in this case was .33/pu/Hz. Table 3.2 shows the performance metrics with (case 0) and without (case 2) participation in frequency response. All other factors are constant. It can be clearly observed that the additional participation in frequency response does not affect error, cost, or reserves. This result shows that
according to NERC guidelines, wind farms can easily participate in frequency response with the energy storage sizing used in these tests.

Table 3.2. Comparison of the base case (case 0) and case 2 with everything else held constant and participation in frequency response turned on.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Error [pu]</th>
<th>Cost [$/kW/mo]</th>
<th>Reserves [pu]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>$J_q$</td>
<td>$J_R$</td>
</tr>
<tr>
<td>0</td>
<td>.0455</td>
<td>.3996</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>.0454</td>
<td>.3995</td>
<td>0</td>
</tr>
</tbody>
</table>

In all subsequent experiments, .03/pu/Hz was the frequency bias used due to a decimal error. The simulations were not re-run because frequency does not play a large enough role to change the major implications of the results.
3.2.2 No Energy Storage

Four different simulations were run with the energy storage disabled. These tests are meant to establish baselines for comparison and explore what happens to error and cost while simply varying schedule and forecast. The tests presented here in Fig. 3.1, Fig. 3.2, Fig. 3.3 and Fig. 3.4 correspond to cases 1-4 of Table 3.1, respectively.

Fig. 3.1. Simulation with energy storage off, persistence forecasting, 60 minute scheduling, and AGC on. Corresponds to case 1 in Table 3.1 and Table 3.3.

Fig. 3.1 shows the error generated with a simple persistence forecast and 60 minute scheduling based on that forecast. It can be observed that since the energy storage is turned off, all error between $P_{\text{sched}}$ and $P_{\text{farm}}$ shows up directly in $P_{\text{error}}$. The changes that can be made upon this using a meteorological forecast are shown in Fig. 3.2.
Fig. 3.2. Simulation with energy storage off, meteorological forecasting, 60 minute scheduling, and AGC on. Corresponds to case 2 in Table 3.1 and Table 3.3.

Observing $P_{\text{sched}}$ and $P_{\text{farm}}$ in Fig. 3.2, it is clear that scheduling power based on a meteorological forecast is at times more accurate due to the fact that it partially eliminates the lagging present in Fig. 3.1 by predicting the ramps. However, it is also less accurate at times due to large overshoots in predicting ramps.
Simulate with energy storage off, meteorological forecasting, 30 minute scheduling, and AGC on. Corresponds to case 3 in Table 3.1 and Table 3.3.

Fig. 3.3 and Fig. 3.4 show that with 30 minute scheduling and perfect forecasting, the difference between $P_{\text{sched}}$ and $P_{\text{farm}}$ can be even further reduced. All of the qualitative observations discussed can be quantitatively observed in Table 3.3 and in Fig. 3.5 where the errors are plotted together.
Fig. 3.4. Simulation with energy storage off, perfect forecasting, 60 minute scheduling, and AGC on. Corresponds to case 4 in Table 3.1 and Table 3.3.

In Table 3.3 the quantitative results show that simply changing the forecast and schedule type with all other things held constant has a large effect on cost, error, and imbalance reserves. As expected, the meteorological forecast outperforms simple persistence, 30 minute scheduling is clearly better than 60 minute, and using a perfect forecast is the absolute best case scenario. Following reserves are not affected by changing schedule and forecast because they are simply the difference between the wind farm actual hour average and the actual activity of the plant. An interesting result of this base case exploration is that we have shown 30 minute scheduling to be significantly better than 60 minute in terms of error, cost, and imbalance reserves. Case 4, where the forecast is perfect, is a theoretical minimum case meant for comparison. If this result was achievable, VERBS costs would be heavily reduced as
the imbalance reserves needed for this situation are almost 6 times lower than case 2, which is closest to what is currently implemented now. For future comparisons the measure of effective wind farm size, calculated by taking $J_Q/J_{base}$, will be used. This measure is important as it gives an easy and non-dimensional way to track improvement. The $J_{base}$ used for all calculations, .3995, is from case 2; meteorological forecast with 60 minute schedule and energy storage off. This case was chosen as it is believed to be the closest to what is currently being used in real systems.

Table 3.3. Comparison of error, costs, and reserve requirements for each of the cases shown. Numbers correspond to those shown in Table 3.1.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Error [pu]</th>
<th>Cost [$/kW/mo]</th>
<th>Reserves [pu]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>$J_Q$</td>
<td>$J_R$</td>
</tr>
<tr>
<td>1</td>
<td>.0528</td>
<td>.6155</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>.0454</td>
<td>.3995</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>.0344</td>
<td>.2445</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>.0174</td>
<td>.0719</td>
<td>0</td>
</tr>
</tbody>
</table>
Fig. 3.5. Comparison of error in cases 1-4 over one day of the testing period. Case numbers correspond to those listed in Table 3.1.

3.2.3 **Energy Storage with No Predictive Control**

Fig. 3.6 shows the first simulation with energy storage on. In this case, predictive control was turned off so that a baseline with energy storage can be established. This baseline represents a simple reactive controller which only corrects the error present at any given moment.
Both Fig. 3.6 and Table 3.4 show that even with this simple control structure, a large amount of the error is absorbed due to the use of the energy storage system. Compared to the same case without energy storage, case 2, the overall MAE is reduced by a factor of 5.1 and the $J_Q$ cost is reduced by a factor of 4.8. The effective wind farm size in this situation is 20.53 percent of the base case. Additionally, imbalance reserve requirements are reduced by 24.7 percent of what is seen in case 2 and following reserves by 37.2 percent.
Table 3.4. Error, costs, and reserve requirements for the case with energy storage activated but no predictive control. Case number corresponds to Table 3.1.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Error [pu]</th>
<th>Cost [$/kW/mo]</th>
<th>Reserves [pu]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>$J_Q$</td>
<td>$J_R$</td>
</tr>
<tr>
<td>2</td>
<td>.0454</td>
<td>.3995</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>.0089</td>
<td>.0820</td>
<td>0</td>
</tr>
</tbody>
</table>

### 3.2.4 Energy Storage with Predictive Control

After establishing a baseline for system performance with energy storage in 3.2.3, the controller framework was used with predictive control implemented. Simulations were run to explore simple functionality of the predictive controller with prediction horizons ranging from 6 (1 hour) to 30 (5 hours). This is referred to as an exploration of simple functionality because only one variable, the error, is still being tracked, minimized, and has a cost associated with it. Table 3.5 shows results for the tests run.
Table 3.5. Error, costs, and reserve requirements for the case with energy storage and predictive control sweeping $H_p$. Case numbers correspond to Table 3.1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>$J_Q$</td>
<td>$J_R$</td>
<td>$J$</td>
</tr>
<tr>
<td>6.1</td>
<td>.0112</td>
<td>.0925</td>
<td>0</td>
<td>.0925</td>
</tr>
<tr>
<td>6.2</td>
<td>.0115</td>
<td>.0800</td>
<td>0</td>
<td>.0800</td>
</tr>
<tr>
<td>6.3</td>
<td>.0116</td>
<td>.0709</td>
<td>0</td>
<td>.0709</td>
</tr>
<tr>
<td>6.4</td>
<td>.0116</td>
<td>.0669</td>
<td>0</td>
<td>.0669</td>
</tr>
<tr>
<td>6.5</td>
<td>.0117</td>
<td>.0640</td>
<td>0</td>
<td>.0640</td>
</tr>
<tr>
<td>7.1</td>
<td>.009</td>
<td>.0756</td>
<td>0</td>
<td>.0756</td>
</tr>
<tr>
<td>7.2</td>
<td>.009</td>
<td>.0753</td>
<td>0</td>
<td>.0753</td>
</tr>
<tr>
<td>7.3</td>
<td>.009</td>
<td>.0751</td>
<td>0</td>
<td>.0751</td>
</tr>
<tr>
<td>7.4</td>
<td>.009</td>
<td>.0750</td>
<td>0</td>
<td>.0750</td>
</tr>
<tr>
<td>7.5</td>
<td>.009</td>
<td>.0749</td>
<td>0</td>
<td>.0749</td>
</tr>
<tr>
<td>8.1</td>
<td>.0022</td>
<td>.0149</td>
<td>0</td>
<td>.0149</td>
</tr>
<tr>
<td>8.2</td>
<td>.0022</td>
<td>.0149</td>
<td>0</td>
<td>.0149</td>
</tr>
<tr>
<td>8.3</td>
<td>.0022</td>
<td>.0148</td>
<td>0</td>
<td>.0148</td>
</tr>
<tr>
<td>8.4</td>
<td>.0022</td>
<td>.0146</td>
<td>0</td>
<td>.0146</td>
</tr>
<tr>
<td>8.5</td>
<td>.0022</td>
<td>.0145</td>
<td>0</td>
<td>.0145</td>
</tr>
</tbody>
</table>

Fig. 3.7 and Fig. 3.8 show the performance of the predictive controller for each different control horizon and forecast/schedule choice. This experiment shows that
the length of prediction horizon used does not have a large effect on overall results. While noticeable changes did occur with persistence forecasting, effectively no change was seen using meteorological forecasts.

![Bar plot showing MAE for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.](image)

While large changes were not seen with varying $H_p$, forecast and schedule type clearly have a large effect on error and cost. With meteorological forecasting and 30 minute scheduling, cases 8.1-8.5 in Table 3.5, the cost per kW per month was brought down to as low as 0.0145 $/kW/mo. The effective wind farm size for this cost is 3.63% of the base case and 5.93% when compared with case 3 which has the same 30 minutes scheduling. Additionally, there was an improvement from 20.53% to 18.77% effective wind farm size, $0.082$ to $0.075$, just from the implementation of the
predictive element of the controller. This can be observed when comparing case 5, seen in Table 3.4, to cases 7.1-7.5 in Table 3.5.

Fig. 3.8. Bar plot showing $J_Q$ cost for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.

Another interesting comparison to make is that with both energy storage and predictive control, the minimal cost for 60 minute scheduling length, $0.064$, is driven below the level seen with scheduling based on a perfect forecast with no energy storage, $0.072$. 

Fig. 3.9. Bar plot showing following reserves as a percent of base case (case 2) for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.

Fig. 3.9 and Fig. 3.10 show the effect of changing prediction horizon on reserves. While it is clear that the addition of energy storage and predictive control reduces reserve requirements, only small changes are seen when sweeping the prediction horizon of the controller. The results suggest that once the forecast and scheduling is accurate enough, changing prediction horizon only minimally affects the reserve requirements. As expected, the 30 minute scheduling case with meteorological forecast has the lowest reserve requirements for both following and imbalance. However, it is surprising to see that once predictive control is added both the following and imbalance reserves of the persistence forecast case are lower than meteorological forecast for the same scheduling length. This is most likely attributed
to the error in the meteorological forecast, while the MAE seen in Fig. 3.7 is lower the maximum deviations must be consistently higher. Interestingly enough, the best overall factor of performance, cost, seems to side with the persistence forecast and the lower reserve requirements. Of course, the best case of all of these is still the 30-minute forecasting case where both MAE and reserves are lowest.

![Bar plot showing imbalance reserves as a percent of base case (case 2) for simulation results seen in Table 3.5. Persistence, meteorological 60 and meteorological 30 refer to the type and length of schedule that was used.](image)

3.2.5 Finding Real Time Damage Incurrence Cost $R$ that Minimizes Overall Cost of Operation $J$

Up to this point in the study the input $R$ cost, which is a measure of how costly each energy storage command is, has been kept at zero so that changes in $J_Q$ cost can be observed and documented. The studies using $J_Q$ cost as a performance measure were
helpful to assess the effectiveness of the controller. However, it is also important to see how varying the $R$ input cost changes the control actions. In these tests, all other variables are held constant and $R$ is swept to observe the correlation between real time damage incurrence cost and overall cost savings from the base case.

The $R$ input cost, ranging from zero to one million $$/pu^2$ per ten minute interval, is a number that represents the real time damage incurrence (RDI) of the energy storage resource. It is important to note that this has nothing to do with capital costs, it is simply the cost of using an energy storage resource at a certain power level over a ten minute period. For the purposes of these simulations, it is assumed that operations and maintenance costs are included in the real time damage incurrence figure $R$. Using this metric, an end result can be obtained that gives both the cost savings and the real time damage incurrence that is associated with that level of cost savings over a simulated week of real time. This cost point is important because it can be used to extrapolate the week long savings to the approximate lifetime of an energy storage source, chosen as 20 years, and see how economical current technologies are. Since there is no long term data on energy storage plants of this size, the 20 year life figure is a rough approximation that comes from the fact that the life span of a wind turbine is typically 20 years (Vestas). Much more advanced engineering economics cost analysis should be done on this topic, but at this point there is not enough information on the real time damage incurrence and capital costs of an energy storage system of this size. Many of the emerging energy storage technologies are still in the
research stage and have only produced small scale prototypes (Johnson). Table 3.6 shows results from the $R$ sweep.

Table 3.6. Cost information for simulations run to sweep $R$ input with all else held constant. Case numbers refer to the key in Table 3.1.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Input</th>
<th>Cost [$/kW/mo]</th>
<th></th>
<th>Case #</th>
<th>Input</th>
<th>Cost [$/kW/mo]</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>1000000</td>
<td>.3975</td>
<td>.0011</td>
<td>.3986</td>
<td>9.15</td>
<td>2000</td>
</tr>
<tr>
<td>9.2</td>
<td>500000</td>
<td>.3955</td>
<td>.0021</td>
<td>.3976</td>
<td>9.16</td>
<td>1000</td>
</tr>
<tr>
<td>9.3</td>
<td>250000</td>
<td>.3918</td>
<td>.0039</td>
<td>.3957</td>
<td>9.17</td>
<td>900</td>
</tr>
<tr>
<td>9.4</td>
<td>150000</td>
<td>.3873</td>
<td>.0061</td>
<td>.3933</td>
<td>9.18</td>
<td>800</td>
</tr>
<tr>
<td>9.5</td>
<td>125000</td>
<td>.3850</td>
<td>.0071</td>
<td>.3922</td>
<td>9.19</td>
<td>700</td>
</tr>
<tr>
<td>9.6</td>
<td>100000</td>
<td>.3818</td>
<td>.0087</td>
<td>.3905</td>
<td>9.20</td>
<td>600</td>
</tr>
<tr>
<td>9.7</td>
<td>75000</td>
<td>.3766</td>
<td>.0111</td>
<td>.3877</td>
<td>9.21</td>
<td>500</td>
</tr>
<tr>
<td>9.8</td>
<td>50000</td>
<td>.3672</td>
<td>.0153</td>
<td>.3825</td>
<td>9.22</td>
<td>400</td>
</tr>
<tr>
<td>9.9</td>
<td>25000</td>
<td>.3396</td>
<td>.0276</td>
<td>.3673</td>
<td>9.23</td>
<td>300</td>
</tr>
<tr>
<td>9.10</td>
<td>12500</td>
<td>.29</td>
<td>.0492</td>
<td>.3392</td>
<td>9.24</td>
<td>200</td>
</tr>
<tr>
<td>9.11</td>
<td>10000</td>
<td>.2702</td>
<td>.0568</td>
<td>.3270</td>
<td>9.25</td>
<td>100</td>
</tr>
<tr>
<td>9.12</td>
<td>50000</td>
<td>.1883</td>
<td>.0834</td>
<td>.2717</td>
<td>9.26</td>
<td>75</td>
</tr>
<tr>
<td>9.13</td>
<td>4000</td>
<td>.1612</td>
<td>.0898</td>
<td>.2510</td>
<td>9.27</td>
<td>50</td>
</tr>
<tr>
<td>9.14</td>
<td>3000</td>
<td>.1288</td>
<td>.0945</td>
<td>.2233</td>
<td>9.28</td>
<td>0</td>
</tr>
</tbody>
</table>
Fig. 3.11. Plot of costs with respect to the R input cost for a week long simulation. In this simulation meteorological forecasting with 60 minute scheduling was used. Frequency response is turned on, and predictive control with an $H_p$ of 6 was used.

Fig. 3.11 shows the results of the sweep of $R$ input cost, tabulated results can be seen in Table 3.6. As expected, as $R$ approaches infinity, the optimal control chooses to not use energy storage at all as the cost of error, $J_Q$, is much cheaper. An interesting observation is that no matter how expensive energy storage is to use, the presence of optimal control finds a way to use it infinitesimally to reduce cost by a small amount. As $R$ trends towards zero, the energy storage controller begins to use it much more and significant cost savings over the base case is seen. In order to calculate the cost savings at each level, each point on the red line is subtracted from $0.3996$, which is the total $J$ cost for the base case (meteorological forecast, 60 minute...
schedule, $H_p = 6$). The results of this subtraction and extrapolation over a twenty year life can be seen in Fig. 3.12.

![Graph showing cost savings over R input cost](image)

Fig. 3.12. Total savings from base case for each point in the $R$ sweep seen in Fig. 3.11.

Fig. 3.11 shows a maximal cost savings of $0.3372$ per kW/mo with an $R$ input of $500/pu^2$ per decaminute. It is important to note that large savings are seen all the way up to an $R$ of $1000/pu^2$ per decaminute before the savings starts to drop drastically. This break point could serve as real time damage incurrence target for energy storage technologies given the same input $Q$ cost.

Extrapolating the cost savings figure of $0.3372$ per kW/mo over 20 years gives a cost savings of $7,208,200$ as seen in Fig. 3.12. This means that in order for energy storage to be cost effective over twenty years, the total capital cost of
installation, including all substation power electronics, minus the salvage value or remaining value after twenty years must be less than $7,208,200, or $144/kWh. However, this does not consider other possible profitable uses of the energy storage system. As stated before, it is assumed that operations and maintenance costs are included in the real time damage incurrence.

3.3 Synthesis of Results

Of the many metrics that have been presented to analyze the data in this manuscript, total $J$ cost is the most important of all as it is the final result of direct optimization by the model predictive controller. Results presented so far have all included price comparisons to the chosen base case. Fig. 3.13 and Table 3.7 show the absolute best case, chosen from the total $J$ cost, compared to the base case. The numbers from this comparison will serve as a realistic talking point for improvements that can be made by adding energy storage to a wind farm.

Table 3.7. Performance metric comparison for best case (case 8.5) vs. base case (case 2) based on total $J_Q$ cost.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Error [pu]</th>
<th>Cost [$/kW/mo]</th>
<th>Reserves [pu]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>$J_Q$</td>
<td>$J_R$</td>
</tr>
<tr>
<td>2</td>
<td>.0454</td>
<td>.3995</td>
<td>0</td>
</tr>
<tr>
<td>8.5</td>
<td>.0022</td>
<td>.0145</td>
<td>0</td>
</tr>
</tbody>
</table>
Fig. 3.13. Visual comparison of mean absolute error (MAE) and $J$ cost for the best case versus base case. This illustrates maximum error reduction and cost savings.

The absolute best case in terms of cost improvement and mean absolute error is case 8.5. This case number refers to meteorological forecasting with 30 minute scheduling and a prediction horizon of five hours. The results of this case can be seen in Fig. 3.14 where $P_{plant}$ has been plotted against $P_{sched}$ to illustrate how well the combination of $P_{farm} + P_{es}$ matches the scheduled power.
Fig. 3.14. Plot of important variables for case 8.5. $P_{plant} = P_{farm} + P_{es}$. The difference between $P_{sched}$ and $P_{plant}$ illustrates the difference between total plant power output and scheduled power output.

In case 8.5, the price in $/kW/mo was driven down to $0.0145 with the help of 30 minute scheduling and predictive control. Using this number to calculate cost savings over 20 years of use compared to the base case (case 2), as was done in 3.2.5, a savings figure of $8,230,092 is obtained. Again, this means that in order for energy storage to be cost effective over twenty years the capital cost of installation must be below $8,230,092, or $164.60/kWh. While this is a slightly higher value than what was found in 3.2.5 due to longer prediction horizon and shorter scheduling, both are in the same range. It is important to note that $8,230,092 is the cost savings when energy storage is simply helping reduce VERBS related costs. There are other potential uses...
for energy storage, such as shaving peaking power, that are profitable and can be used to justify greater cost.

To put these numbers in perspective, current technologies and their respective price points will be used. According to ZBB Energy Corporation, the cost of building a 25 MW/50 MWh flow cell battery installation, a “Zinc plating plant”, including substation, inverters, and transformers could be as low as $27,500,000, $550/kWh, in 5 years. This includes an assumption that the technology matures enough to reduce the price of the actual energy storage installation to $250/kWh, a figure that has been found by large wind farm developers to be the value of an installation like this (Johnson). The current price of a ZBB energy storage installation alone is still over $500/kWh. Alternatively, pumped hydropower storage, which is already a mature technology, can cost anywhere from $80/kWh to $120/kWh (Johnson). Unfortunately, pumped hydropower storage is extremely geographically limited. Therefore it cannot be widely implemented at this cost and another solution is necessary.
4 Conclusion

Wind penetration in the Pacific Northwest is expected to reach 6000 MW of installed capacity by the end of 2014 and possibly 8000 MW in 2017 (“Growth Forecast,” BPA). With the hydropower system already reaching limits balancing wind at current levels, something must be added to the system to allow for future expansion ("Connecting variable generating resources to the federal Columbia river transmission system (FCRTS)", BPA).

It was shown that the addition of an energy storage resource with a power capacity 25% of the wind farm rated power and energy capacity of 50% can significantly reduce the amount of reserves needed from the hydropower system. While participation in frequency response did not affect overall error, cost, or reserves, changes in scheduling type and length had a large influence. The absolute best case according to overall cost was 30 minute scheduling with meteorological forecast and a prediction horizon of 5 hours. Sweeping the prediction horizon from zero up to five hours showed that the predictive element helped the controller minimize total error but the length of prediction horizon had little effect on cost or reserves, especially with meteorological forecasting. Cost analysis on the real time damage incurrence (RDI) variable $R$ yielded results that suggest 500 $$/pu^2$ as an acceptable RDI target. Finally, total potential savings for the system was assessed. With an assumed system lifetime of 20 years it was found that in order to be economical, an energy storage system for a 100 MW wind farm would need to cost less than $8,230,092, or $164.60/kWh.
While energy storage with predictive control can improve the reliability and dispatchability of a wind firm, cost estimations suggested that given the current price of large energy storage technologies it is not economically feasible to implement. Long term economic factors including increasing energy prices, increasing wind penetration, and decreasing price of energy storage technology all support the future use of a system like the one tested.

4.1 Future Work

With the model predictive control framework in place, many additional explorations are planned. Along with the error minimization experiments in this document, the controller will be used to decrease the frequency of ramp events through minimization of $\Delta P_{plant}$. Furthermore, any combination of the current output variables, seen in (2.8), can have a cost associated with it and can be minimized. As understanding of new problems is developed, the potential for even more explorations is present. Other topics of interest include the placement of energy storage and supplementary uses for energy storage like peak shaving that could help make an installation be more profitable. Finally, hardware testing will be done to confirm the simulation results.
5 Bibliography


BPA, “Current and proposed wind project interconnections to BPA transmission facilities,” BPA,


Appendices
APPENDIX 1: Main Simulation File Code

% Energy storage control using BPA aggregate wind data
% Created by: Ted Brekken, Michael Antonishen, Doug Halamay
% with assistance of: Annette von Jouanne, Alex Yokochi, Jiajia
% Song, Jim Davidson, David Naviaux
% Parallelized by: Douglas A. Halamay
% Created: January 6, 2012
% Last modified: April 4, 2012

clear all; close all;

%% Documentation
% State space formulation is of the form
% \( x(k+1) = A \cdot x(k) + B_u \cdot u(k) + B_v \cdot v(k) \)
% \( y(k) = C \cdot x(k) + D_u \cdot u(k) + D_v \cdot v(k) \)
% where \( x = [\text{SOC} ; \text{P_es_prev}] \) % state vector
% \( u = [\text{P_es}] \) % controlled inputs
% \( v = [\text{P_AGC} ; \text{P_sched} ; \text{P_farm} ; \text{delP_farm} ; \text{P_loss}] \)
% uncontrolled inputs (disturbances)
% \( y = [\text{P_error}; \text{delP_es}; \text{P_plant}; \text{delP_plant}; \text{SOC}; \text{SOC_next}] \) % outputs
% % \( \text{P_error} \) = Error between requested output (\( \text{P_agc} + \text{P_sched} \)) and
% and 
% \( \text{P_es_prev} \) = Previous energy storage output
% \( \text{delP_es} \) = Derivative of \( \text{P_es} \)
% \( \text{P_es} \) = Actual energy storage output
% \( \text{SOC} \) = Energy storage state of charge
% \( \text{SOC_next} \) = Energy storage state of charge at the next sample
% \( \text{P_agc} \) = Forecasted AGC power (from frequency deviation)
% -
% Secondary Component
% \( \text{P_sched} \) = Scheduled wind output
% \( \text{P_farm} \) = Forecasted actual wind output
% \( \text{delP_farm} \) = Derivative of \( \text{P_farm} \) (so we can punish ramps)
% \( \text{P_loss} \) = Power loss in the energy storage due to leakage or
% power electronics
% \( \text{P_plant} \) = \( \text{P_farm} + \text{P_es} \)
% \( \text{delP_plant} \) = Derivative of \( \text{P_plant} \)

% Doug made an executive decision that the energy storage is
% 0.25 pu Rated Power and 0.5 pu-hours Rated Energy

%% MPC Control Initialization
% Set loop to 0 if we want to only run a one-off experiment, 
% otherwise set 
% it to 1.
loop = 0;

if loop == 1
    % ExpInputs.mat contains the ExpInputs matrix and the loadData
    % The ExpInputs matrix is 12xN, where N is the number of
    % experiments we
    % want to run and is composed thusly:
    % Row       Data Input
    % 1         makePwindType
    % 2         EnergyStorageSwitch
    % 3         Hp
    % 4         startTime
    % 5         numSteps
    % 6         SOC_init
    % 7         predict
    % 8         derateFactQ
    % 9         derateFactR
    % 10        Qcost
    % 11        Rcost
    % 12        AGCSwitch
    load Data/ExpInputs.mat;
end

% Here we specify the loopMax depending on the value of loop.
if loop == 0
    loopMax = 1;
else
    loopMax = size(ExpInputs,2);
end

% Turn plots on (1) or off (0)
plots = 1;

% Control line width on plots (wide (1) or thin (0))
lines = 0;

% Display each step's control vector
displaySteps = 0;

% Define bias for AGC participation (per-unit power/Hz) (normally
negative)
bias = -0.33;

% Define State of Charge (SOC) slope of increase/decrease (from -1 to
1)
% (0 to full SOC) in 12 samples (two hours) over the rated power)
md = (2/12)/0.25;

% This one has no meaning for the BPA data
modelNumber = 1;
% Define previous step's P_es (traditionally, always zero for our tests)
P_es prev_init = 0;

% Here we load the single-run parameters (if loop == 0) (otherwise, we
% already have a ExpInputs matrix so we get the data from there).
if loop == 0
    % This decides which makePwind we use
    % 0 -- BPA forecast, persistence scheduling (makePwindBPA)
    % 1 -- BPA forecast, average persistence scheduling 60 min
    (makePwindAVE)
    % 2 -- BPA forecast, average persistence scheduling 30 min
    (makePwind30)
    % 3 -- Perfect forecast, average persistence scheduling 60 min
    % (makePwindAVE_BPAPerfect)
    makePwindType = 1;

    % Energy Storage off or on switch. When the storage is switched
    off, the
    % P_es command will always be 0.
    % 0 = off, 1 = on
    EnergyStorageSwitch = 0;

    % Define prediction horizon Hp (6 samples = 1 hour)
    % No limit on how large Hp can be, Hp>=6
    Hp = 6;

    % Specify startTime for makePwind
    startTime = 1;

    % Define number of steps to take in simulation (1 step = 10
    minutes)
    % Max number of steps is 936 (6.5 days) based on amount of
    continuous wind
    % data available
    numSteps = 888;  % (one day = 144 steps)

    % Define initial SOC
    SOC_init = 0;

    % Prediction capability
    predict = 1;

    % Define derating factor for Q matrix
    % derateFactQ = -log(0.5);
    derateFactQ = 0;

    % Define derating factor for R matrix
    % derateFactR = -log(0.5);
    derateFactR = 0;
\% Define weights for Q matrix
\% 2.5029e+003 ---
\% This number was calculated using error data over a 4 month period with no
\% energy storage source available. Units are (Dollar/PU^2)
Qweight = [2.5029e+003; 0; 0; 0; 0; 0];

\% Define weight for R matrix
\% 640.2 $/PU^2 of energy
Rweight = 0;

\% Turn on (1) or off (0) AGC
AGCSwitch = 1;

\% Build our ExpInputs matrix for the one-off case
ExpInputs = zeros(12,1);
ExpInputs(1,1) = makePwindType;
ExpInputs(2,1) = EnergyStorageSwitch;
ExpInputs(3,1) = Hp;
ExpInputs(4,1) = startTime;
ExpInputs(5,1) = numSteps;
ExpInputs(6,1) = SOC_init;
ExpInputs(7,1) = predict;
ExpInputs(8,1) = derateFactQ;
ExpInputs(9,1) = derateFactR;
ExpInputs(10,1) = Qweight(1);
ExpInputs(11,1) = Rweight;
ExpInputs(12,1) = AGCSwitch;

\% Set our loadData string
loadData = 'Data/BPAWindForecast040212.mat';
end

\% To save the pertinent data from our experiments, we set up the recording
\% variables like so (note that numSteps (ExpInputs(5,:)) must be the same
\% for all experiments since we are using rectangular recording matrices).
P_sched_rec = zeros(ExpInputs(5,1),loopMax);
P_farm_rec = zeros(ExpInputs(5,1),loopMax);
P_es_rec = zeros(ExpInputs(5,1),loopMax);
P_error_rec = zeros(ExpInputs(5,1),loopMax);
P_agc_rec = zeros(ExpInputs(5,1),loopMax);
SOC_rec = zeros(ExpInputs(5,1),loopMax);

\% Initialize error records
P_error_MAE = zeros(loopMax,1);
P_error_RMSE = zeros(loopMax,1);

\% Initialize cost totals
QCostTotal = zeros(loopMax,1);
RCostTotal = zeros(loopMax,1);
JCostTotal = zeros(loopMax,1);

% Initialize reserve requirement records
followMin = zeros(loopMax,1);
followMax = zeros(loopMax,1);
imbalMin = zeros(loopMax,1);
imbalMax = zeros(loopMax,1);

% Clear and then declare globals for the makePwindBPA function
clear global BPAAActual
clear global BPAPForecast

global BPAAActual

global BPAPForecast

% Load the data (BPAAActual, BPAPForecast, datestring)
load(loadData);

% Load the frequency data (10 min averaged from 2 second data from
BPA)
load('Data/freq10minBPAPFarmA.mat');

%% Main Loop
% Open the pool
% But first we check to see if the pool is already open
if matlabpool('size') == 0
    matlabpool open
end

% Now we run the experiments
parfor l = 1:loopMax
%for l = 1:loopMax
    % Parameter set-up
    % Load parameters from the ExpInputs matrix
    makePwindType = ExpInputs(1,l);
    EnergyStorageSwitch = ExpInputs(2,l);
    Hp = ExpInputs(3,l);
    startTime = ExpInputs(4,l);
    numSteps = ExpInputs(5,l);
    SOC_init = ExpInputs(6,l);
    predict = ExpInputs(7,l);
    derateFactQ = ExpInputs(8,l);
    derateFactR = ExpInputs(9,l);
    Qweight = [ExpInputs(10,l); 0; 0; 0; 0; 0];
    Rweight = ExpInputs(11,l);
    AGCSwitch = ExpInputs(12,l);

    % Check that our numSteps will work
    if mod(numSteps,6) ~= 0
        error('Number of steps must be a multiple of 6 for inc/dec
calculations')
```
end

% Define loss size for power electronics (given 0.25 rated power
and a loss
% of 0.5 kW per 25 kW in the ZBB system (equivalent to a drop of
full SOC
% in 100 hours)).
% If the Energy Storage is off, there should be no losses.
if EnergyStorageSwitch == 0;
    loss_size = 0;
else
    loss_size = (0.25*(.5/25));
end

% makeQ
% - function to generate Q matrix
% - inputs: Hp, weighting for Q matrix (a vector), derating
% factor,
% prediction capability on/off (1/0)
% - outputs: Q matrix (size: 6*(Hp+1) x 6*(Hp+1))
Q = makeQ(Hp, Qweight, derateFactQ, predict);

% makeR
% - function to generate R matrix
% - inputs: Hp, weighting for R matrix (a single value), derating
% factor
% - outputs: R matrix (size: Hp+1 x Hp+1)
R = makeR(Hp, Rweight, derateFactR);

% Create initial state vector
x_k = [SOC_init; P_es_prev_init];

% Define output constraints
% (SOC and SOC_next must be between -1 and 1, and the rest do not
matter).
y_const_up = [1e3; 1e3; 1e3; 1e3; 1; 1];
y_const_low = [-1e3; 1e3; 1e3; 1e3; 1; 1];

% Define input constraints (0.25 pu Rated Power of Energy
Storage)
u_const_up = 0.25;
u_const_low = -0.25;

% State space matrix initialization
% Define A, Bu, Bv, C, Du, Dv
A = [1 0
     0 0];
Bu = [-md;1];
Bv = [0 0 0 0 -md
     0 0 0 0 0];
C = 
\begin{bmatrix}
0 & 0 \\
0 & -1 \\
0 & 0 \\
0 & -1 \\
1 & 0 \\
1 & 0
\end{bmatrix};

Du = [-1; 1; 1; 1; 0; -md];

Dv = [1 1 -1 0 0 \\
0 0 0 0 0 \\
0 0 1 0 0 \\
0 0 0 1 0 \\
0 0 0 0 0 \\
0 0 0 0]

% Control display of matrices in formulateMPC function
dispMat = 0;

% Formulate MPC matrices
[Sx, Su, Hv] = formulateMPC(A, Bu, Bv, C, Du, Dv, Hp, dispMat);

%% QuadProg Loop (w/Generative Function Calls)
% It's a trap!

% Initiate P_locked vector for makePwind functions
P_locked = [];

% Set up temporary recording arrays (workaround for parfor)
P_es_rec_temp = zeros(numSteps,1);
P_farm_rec_temp = zeros(numSteps,1);
P_sched_rec_temp = zeros(numSteps,1);
P_error_rec_temp = zeros(numSteps,1);
P_agc_rec_temp = zeros(numSteps,1);
SOC_rec_temp = zeros(numSteps,1);

% Begin das loop!
tic
for n = startTime:startTime+numSteps-1;

    % Get frequency at the current sample
    if AGCSwitch == 1
        cur_freq = freq10(n);
    else
        cur_freq = 60;
    end

    % makeP_agc
    % - function to generate P_agc vector using a simple persistence model
    % - inputs: Hp, bias, cur_freq (frequency at current sample)
% - outputs: P_agc vector of length (Hp+1) (positive P_agc means more
generation)
P_agc = makeP_agc(Hp,bias,cur_freq);

% global_makePwind
% - function to generate P_sched, P_farm, delP_farm vectors
% - This is actually a bridge function to introduce the
globals for
% the parfor loop to work correctly.  global_makePwind
performs the
% actual function calls
[P_sched, P_farm, delP_farm, P_locked] =
global_makePwind(BPAActual,BPAForecast,makePwindType,n,Hp,modelNumber
,P_locked);

% Check for NaNs in case makePwind did something strange
if(find(isnan(P_sched),1) ~= 0 | find(isnan(P_farm),1) ~= 0)
 disp(['num2str(l),',',num2str(n),' NaN Error'])
 break;
 end

% makeP_loss
% - function to generate P_loss vector
% - inputs: Hp, loss_size
% - outputs: P_loss vector of length (Hp+1)
P_loss = makeP_loss(Hp, loss_size, x_k);

% makeV_k
% - function
% - inputs: Hp, P_agc, P_sched, P_farm, delP_farm, P_loss
% - output: V_k vector of length (Hp+1)*5
V_k = makeV_k(Hp, P_agc, P_sched, P_farm, delP_farm, P_loss);

% makeT_k
% - function
% - inputs: Hp
% - outputs: T_k vector of length (Hp+1)*6 containing
 requested
% trajectories of P_error, delP_ES, P_plant,
delP_plant, SOC,
% and SOC_next (includes current and future samples)
T_k = makeT_k(Hp);

% makeE_k
% - function
% - inputs: Hp, Sx, Hv, T_k, V_k, x_k
% - outputs: E_k vector of length (Hp+1)*6 containing free-
evolution error
% trajectories (current and future)
E_k = makeE_k(Hp, Sx, Hv, T_k, V_k, x_k);
% makeConstraints
% - function to generate constraint matrices for use in quadprog
%   - inputs: Hp, Sx, Su, Hv, V_k, x_k, y_const_up,
y_const_low, u_const_up, u_const_low
%   - outputs: Aconstrain (left side of constraints) and Bconstrain (right side of constraints)
[Aconstrain, Bconstrain] = makeConstraints(Hp, Sx, Su, Hv,
V_k, x_k, y_const_up, y_const_low, u_const_up, u_const_low);

% Set up variables for call to quadprog
H = Su'*Q*Su + R;
f = -(Su'*Q*E_k);

% Call quadprog with appropriate options depending on which version of
% Matlab we are using (LargeScale off for pre-2011 versions or
% interior-point-convex algorithm for subsequent versions)
if verLessThan('optim','6.0')
    opts = optimset('LargeScale','off');
else
    opts = optimset('Algorithm','interior-point-convex');
end
opts = optimset(opts,'Display','off','MaxIter',1000);
[U_k,fval,exitflag,output1,lambda] =
quadprog(H,f,Aconstrain,Bconstrain,[],[],[],[],[],opts);

% Catch infeasibility (could add a more descriptive error here)
if(exitflag ~= 1)
    disp(['ERROR WITH QUADPROG']);
end

% If we have an error with quadprog, we need to substitute in U_k,
% so we create U_k of the appropriate length
U_k = zeros(Hp+1,1);

% Then, we populate the first entry of U_k (i.e., P_es) (really,
% the only one we care about) with the value of % P_error (P_agc + P_sched - P_farm)
U_k(1) = P_agc(1) + P_sched(1) - P_farm(1);

% We then need to check that P_es is within its constraints. If we've violated a limit, set P_es to that limit.
if(U_k(1) > u_const_up)
    U_k(1) = u_const_up;
    disp(['Infeasible, Invalid P_es (upper)']);
end
elseif(U_k(1) < u_const_low)
    U_k(1) = u_const_low;
    disp(['num2str(l),'','num2str(n),'' Infeasible, Invalid P_es (lower)']);
end

% Calculate some results for debugging
Y_k = Sx*x_k + Su*U_k + Hv*V_k;
SOC_cur = (x_k(1)+1)*0.5;

% If energy storage is off (0), set U_k(1) (i.e., P_es) to zero
if EnergyStorageSwitch == 0
    U_k(1) = 0;
end

% Select first entry from U_k and apply to model to generate new x_k
[x_k_new_temp, y_k_temp] = runModel(A, Bu, Bv, C, Du, Dv, x_k, U_k(1), V_k(1:5), u_const_up, u_const_low);

% Display Uk and SOC for checking
% disp(['num2str(U_k(1)), ', 'num2str(y_k_temp(5))'])

% Check to see if the control action has caused an infeasible SOC
% If so, set the P_es to the closest possible value that will not cause
% an infeasible SOC and rerun the model. Else, store the temporary
% model outputs (x_k_new and y_k) into the appropriate variables and
% continue.
% We need to do this because we may have not had a successful quadprog
% run earlier and thus we may not maintain our SOC limits appropriately

% when we run the model
if(x_k_new_temp(1) < -1)
    if((x_k_new_temp(1) + 1) < 1e-6)
        disp(['num2str(l),'','num2str(n),'' Invalid SOC (neg.) - probable false positive']);
    else
        disp(['num2str(l),'','num2str(n),'' Invalid SOC (neg.)']);
    end
    U_k(1) = (-1 - x_k(1) + md*V_k(5))/(-md);
else
    disp(['num2str(l),'','num2str(n),'' Invalid SOC (neg.)']);
end

elseif(x_k_new_temp(1) > 1)
if((x_k_new_temp(1) - 1) < 1e-6)
disp([num2str(l),',',num2str(n),' Invalid SOC (pos.) - probable false positive']);
else
disp([num2str(l),',',num2str(n),' Invalid SOC (pos.)']);
end
U_k(1) = (1 - x_k(1) + md*V_k(5))/-md;
[x_k_new, y_k] = runMode1(A, Bu, Bv, C, Du, Dv, x_k, U_k(1), V_k(1:5), u_const_up, u_const_low);
if dispSteps == 1
disp(['x_k: ' num2str((x_k(1)+1)*.5), ' x_k_new: ' num2str((x_k_new(1)+1)*.5)]);
end
% Set new x_k
x_k = x_k_new;

% calculateCost
% - function to determine individual and total costs
% - inputs: Hp, y_k, T_k, U_k, Q, R
% - outputs: QCost, RCost, JCost numbers
[QCost, RCost, JCost] = calculateCost(Hp, y_k, T_k, U_k, Q, R);

% Sum the costs for each step
QCostTotal(l) = QCostTotal(l) + QCost;
RCostTotal(l) = RCostTotal(l) + RCost;
JCostTotal(l) = JCostTotal(l) + JCost;

% Populate temporary vectors each loop
P_es_rec_temp(n) = x_k(2);
SOC_rec_temp(n) = (x_k(1)+1)*.5;
P_farm_rec_temp(n) = P_farm(l);
P_sched_rec_temp(n) = P_sched(l);
P_error_rec_temp(n) = y_k(l);
P_agc_rec_temp(n) = P_agc(l);
end

% Store the time series data outside of the inner loop (parfor workaround for slicing variables)
P_es_rec(:,l) = P_es_rec_temp;
SOC_rec(:,l) = SOC_rec_temp;
P_farm_rec(:,l) = P_farm_rec_temp;
P_sched_rec(:,l) = P_sched_rec_temp;
P_error_rec(:,l) = P_error_rec_temp;
P_agc_rec(:,l) = P_agc_rec_temp;
%% Plotting & Data analysis Code
for l = 1:loopMax
  %% Quantification & display of error
  P_error_MAE(l) = sum(abs(P_error_rec(:,l)))/length(P_error_rec(:,l));
  P_error_RMSE(l) = sqrt(sum(P_error_rec(:,l).^2)/length(P_error_rec(:,l)));
  cprintf('blue',
          ['P_error_mae:   ',num2str(P_error_MAE(l)),'\n']);
  cprintf('blue',
          ['P_error_rmse: ',num2str(P_error_RMSE(l)),'\n']);
  cprintf('blue',
          ['Q_cost_total: ',num2str(QCostTotal(l)),'\n']);
  cprintf('blue',
          ['R_cost_total: ',num2str(RCostTotal(l)),'\n']);
  cprintf('blue',
          ['J_cost_total: ',num2str(JCostTotal(l)),'\n']);
  if plots == 1
    % Put together string for energy storage on/off
    if ExpInputs(2,l) == 1
      ES_string = 'on';
    else
      ES_string = 'off';
    end
    % Put together string for AGC on/off
    if ExpInputs(12,l) == 1
      AGC_string = 'on';
    else
      AGC_string = 'off';
    end
    % String for prediction on/off
    if ExpInputs(7,l) == 1
      Predict_string = 'on';
    else
      Predict_string = 'off';
    end
    % Plot the figure depending on whether we want thick or thin lines
    if lines == 1
      figure;
      subplot(1,1,1,'FontSize',14);
      plot(P_sched_rec(:,l),'r','LineWidth',1.3);
      hold on;
      plot(P_farm_rec(:,l),'b','LineWidth',1.3);
      plot(P_es_rec(:,l),'k','LineWidth',1.3);
      plot(P_error_rec(:,l),'m','LineWidth',1.3);
      plot(P_agc_rec(:,l),'g','LineWidth',1.3);
      toc
    end
  end
plot(SOC_rec(:,l), 'c', 'LineWidth', 1.3);
axis([0 ExpInputs(5,l)+12 -0.25 1.1]);
titleString = ['Hp:', num2str(ExpInputs(3,l)), ' ES:', ES_string, ' F/S:', num2str(ExpInputs(1,l)), ' Predict:', Predict_string, ' ', datestring];
hold on;
legend('
P_{sched}', 'P_{farm}', 'P_{es}', 'P_{error}', 'P_{agc}', 'SOC', 'Location', 'SouthEast')
set(gca, 'xtick', (0:24*6:24*6*6), 'XTickLabel', {'0', '1', '2', '3', '4', '5', '6'});
xlabel('Time (days)');
ylabel('PU');
printStr = strcat('.\images\Fig', 'Hp', num2str(ExpInputs(3,l)), 'ES', ES_string, 'Model', num2str(ExpInputs(1,l)), 'Predict', Predict_string, 'Date', datestring, 'Thin');
print('-dpng', '-r600', printStr);
else
    figure;
    subplot(1,1,1, 'FontSize', 14);
    plot(P_sched_rec(:,l), 'r');
    hold on;
    plot(P_farm_rec(:,l), 'b');
    plot(P_es_rec(:,l), 'k');
    plot(P_error_rec(:,l), 'm');
    plot(P_agc_rec(:,l), 'g');
    plot(SOC_rec(:,l), 'c');
    axis([0 ExpInputs(5,l)+12 -0.25 1.1]);
titleString = ['Hp:', num2str(ExpInputs(3,l)), ' ES:', ES_string, ' F/S:', num2str(ExpInputs(1,l)), ' Predict:', Predict_string, ' ', datestring];
%     title(titleString,'color','b');
legend('
P_{sched}', 'P_{farm}', 'P_{es}', 'P_{error}', 'P_{agc}', 'SOC', 'Location', 'NorthWest')
set(gca, 'xtick', (0:24*6:24*6*6), 'XTickLabel', {'0', '1', '2', '3', '4', '5', '6'});
xlabel('Time (days)');
ylabel('PU');
printStr = strcat('.\images\Fig', 'Hp', num2str(ExpInputs(3,l)), 'ES', ES_string, 'Model', num2str(ExpInputs(1,l)), 'Predict', Predict_string, 'Date', datestring, 'AGC', AGC_string, 'Thin');
print('-dpng', '-r600', printStr);
end
end

%% Reserve requirement calculations
if (ExpInputs(1,l) ~= 2)
    % Calculate wind hour averages
    % These are calculated at the 0:50 point, including the previous six data
    % points (including the 0:50 point).
In order to save having to do a for loop through the data to calculate the hour averages, we first calculate a simple 6-point moving average using the filter function:

\[
\text{plantMoveAve} = \text{filter}\left(\frac{\text{ones}(1,6)}{6},1,(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))'\right);
\]

% Then, we take every 6th point from our moving average (i.e., the hour's average) and multiply by a 6x1 matrix of ones to get a column for each hour with six repeated points. Then, we reshape to get a single vector.

\[
\text{plantHourAve} = \text{reshape}\left(\frac{\text{ones}(6,1)\times\text{plantMoveAve}(6:6:\text{length}(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))),1,[1]}{\text{follow} = \text{plantHourAve} - (P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))'}\right);
\]

% Now we have to deal with the between-hour transitions (at the :00 point). First, we deal with the very first point in the dataset and set it equal to the original first point.

\[
\text{plantHourAve}(1) = P_{\text{es\_rec}}(1,l)+P_{\text{farm\_rec}}(1,l);
\]

% Then we make the :00 point for each subsequent hour an average of the :50 point and the 1:10 point.

\[
\text{plantHourAve}(7:6:\text{length}(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))) = (\text{plantMoveAve}(6:6:(\text{length}(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l)))-1)) + \text{plantMoveAve}(12:6:\text{length}(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))))/2;
\]

follow = plantHourAve - (P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))';
imbalance = P_{\text{sched\_rec}}(:,l)' - plantHourAve;

else
% Calculate 30 min averages
% In order to save having to do a for loop through the data to calculate the half-hour averages, we first calculate a simple 3-point moving average using the filter function:

\[
\text{plantMoveAve} = \text{filter}\left(\frac{\text{ones}(1,3)}{3},1,(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))'\right);
\]

% Then, we take every 3rd point from our moving average (i.e., the half-hour's average) and multiply by a 3x1 matrix of ones to get a column for each half-hour with three repeated points. Then, we reshape to get a single vector.

\[
\text{plantHourAve} = \text{reshape}\left(\frac{\text{ones}(3,1)\times\text{plantMoveAve}(3:3:\text{length}(P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))),1,[1]}{\text{follow} = \text{plantHourAve} - (P_{\text{es\_rec}}(:,l)+P_{\text{farm\_rec}}(:,l))'}\right);
\]

% The function that calculates the imbalance is not included in the provided text.
follow = plantHourAve - (P_es_rec(:,1)+P_farm_rec(:,1))';
imbalance = P_sched_rec(:,1)' - plantHourAve;
end

% Outlier removal
% % Throw out the top and bottom 0.25% of the following and imbalance
% matrices by using a temp matrix to sort into and then trimming

% Sort the data
followTempSort = sort(follow);
imbalanceTempSort = sort(imbalance);

% Calculate new data length
dataLength = length(follow);
newDataLength = ceil(dataLength*.995);

% Cut off the bottom and top 0.25% of the data and save in the new matrices
followSort = followTempSort(ceil((dataLength - newDataLength)/2):floor((dataLength + newDataLength)/2));
imbalanceSort = imbalanceTempSort(ceil((dataLength - newDataLength)/2):floor((dataLength + newDataLength)/2));

% Follow and imbalance data analysis (min/max calculations)
% % Initialize min and max vectors (two columns - follow and imbal)
plantMin = zeros(1,2);
plantMax = zeros(1,2);

% Find the min and max for follow and imbalance
plantMin(1,1) = min(followSort);
plantMax(1,1) = max(followSort);
plantMin(1,2) = min(imbalanceSort);
plantMax(1,2) = max(imbalanceSort);

% Store the variables for future use
followMin(l) = plantMin(1,1);
followMax(l) = plantMax(1,1);
imbalMin(l) = plantMin(1,2);
imbalMax(l) = plantMax(1,2);

% Calculate the total reserve requirement (inc plus dec)
followTot = abs(plantMin(1,1)) + plantMax(1,1);
imbalTot = abs(plantMin(1,2)) + plantMax(1,2);

cprintf('blue',['Following (Inc):
',num2str(plantMax(1,1)),'
']);
cprintf('blue',['Following (Dec):
',num2str(plantMin(1,1)),'
']);
cprintf('blue', ['Imbalance (Inc):
', num2str(plantMax(1,2)), '
']);
cprintf('blue', ['Imbalance (Dec):
', num2str(plantMin(1,2)), '
']);

% Display the title string to the command window for ease of data collection
if plots == 1
    cprintf('red', [titleString, '
']);
end

end

%% Close the pool
matlabpool close
APPENDIX 2: Constraint Making Function Code

function [Aconstrain, Bconstrain] = makeConstraints(Hp, Sx, Su, Hv, V_k, x_k, y_const_up, y_const_low, u_const_up, u_const_low)
% Function to generate constraint matrices for use in quadprog
% Takes Hp, Sx, Su, Hv, V_k, x_k, y_const_up, y_const_low, u_const_up, u_const_low as inputs
% Generates Aconstrain (left side of constraints) and Bconstrain (right side of constraints) as outputs, where Aconstrain*x <= Bconstrain applies to quadprog
% Pre-allocate for speed
Aconstrain = zeros(2*(Hp+1)+2*size(Su,1),Hp+1);
Bconstrain = zeros(2*(Hp+1)+2*size(Su,1),1);

% First, create the Aconstrain side
% This will look like [I;-I;Su;-Su]
Aconstrain(1:Hp+1,:) = eye(Hp+1);
Aconstrain(Hp+2:2*(Hp+1),:) = -eye(Hp+1);
Aconstrain(2*(Hp+1)+1:2*(Hp+1)+size(Su,1),:) = Su;
Aconstrain(2*(Hp+1)+size(Su,1)+1:end,:) = -Su;

% Next, create the Bconstrain side
% This will look like
% [U_const_up_k;
% -U_const_low_k;
% Y_const_up_k-Sx*x_k-Hv*V_k;
% -Y_const_low_k+Sx*x_k+Hv*V_k]

% Create the Y_const_up and Y_const_low matrices (note the capital Y)
% by repeating the y_const_up and y_const_low vectors (Hp+1) x 1 times.
Y_const_up = repmat(y_const_up,Hp+1,1);
Y_const_low = repmat(y_const_low,Hp+1,1);

% Assign the appropriate values
Bconstrain(1:Hp+1,:) = u_const_up;
Bconstrain(Hp+2:2*(Hp+1),:) = -u_const_low;
Bconstrain(2*(Hp+1)+1:2*(Hp+1)+size(Su,1),:) = Y_const_up - Sx*x_k - Hv*V_k;
Bconstrain(2*(Hp+1)+size(Su,1)+1:end,:) = -Y_const_low + Sx*x_k + Hv*V_k;
end
APPENDIX 3: Make E_k Vector Code

function [E_k] = makeE_k(Hp, Sx, Hv, T_k, V_k, x_k)
% Function to generate E_k vector for MPC formulation
% Takes Hp, Sx, Hv, T_k, V_k, x_k as inputs
% Generates the E_k vector as an output

E_k = T_k - (Sx*x_k + Hv*V_k);
end
APPENDIX 4: Make P_agc Code

function [P_agc] = makeP_agc(Hp, bias, cur_freq)
% Function to generate P_agc
% Takes Hp, bias, and cur_freq (frequency at the current sample) as inputs
% Generates P_agc as an output

P_agc = zeros(Hp+1,1);

P_agc(:) = bias*(cur_freq-60);

end
APPENDIX 5: Make P_loss Code

function [P_loss] = makeP_loss(Hp, loss_size, x_k)
% Function to generate P_loss
% Takes Hp and loss_size as inputs
% Generates P_loss as an output

P_loss = zeros(Hp+1,1);

if x_k(1) <= -.98
    P_loss(:) = loss_size*((x_k(1)+1)/2);
else
    P_loss(:) = loss_size;
end

end
function [P_sched, P_farm, delP_farm] = makePwind_BPA(currentData,Hp,~)

%makePwind_BPA: Function to generate P_sched, P_farm, and delP_farm
from BPA
% wind forecast data
%
% This function utilizes BPA wind forecast data to generate the
% necessary
disturbance vectors for the MPC controller
%
% Created by: Douglas A. Halamay
% Created: March 12, 2012
% Last Modified: March 12, 2012 by MPA

%% Housekeeping and initialization

% Input error-checking code
% Check the size of Hp to ensure it is not too large
if Hp > (72*6)
    error('Time horizon is too long (must be no more than 72 hours');
end

% Check the currentData pointer to be sure it is within the data limits
if currentData > 936
    error('Requested data position is too large (must be no more than 936');
end

% As of 2/29/12, 4131 MW installed
% As of 3/11/12, 4421 MW installed
windBase = 4131;

% Declare global variables. These probably do not need to be globals, but
% we will declare them anyway in order to limit data traffic.

% BPAActual contains the actual wind for each point in time
% (BPAActual.data10) and the timestamps in UTC (BPAActual.datetime10)
global BPAActual;

% BPAPredict contains one week of hourly wind forecasts for the next 72
% hours (with the most current forecast first) (BPAPredict.data) and the
% timestamps in UTC (BPAPredict.datetime60)
global BPAPredict;
%% Archived data extraction code
% Code necessary to convert to UTC and handle data ordering
%
% % Remove PST/PDT from the datetime strings
% for i = 1:length(BPAActual.datetime(:,1));BPAActual.datetime2{i,1} = BPAActual.datetime{i,1}(1:end-4);end
% % Convert to datevectors
% BPAActual.datetime3 = datevec(BPAActual.datetime)
% % Convert PST to UTC
% BPAActual.datetime3(1:1464,4) = BPAActual.datetime3(1:1464,4)+8;
% for i = 1:1464;if(BPAActual.datetime3(i,4) > 23);BPAActual.datetime3(i,3) = BPAActual.datetime3(i,3) + 1;BPAActual.datetime3(i,4) = BPAActual.datetime3(i,4) - 24;end;end;
% % Convert PDT to UTC
% BPAActual.datetime3(1:end,4) = BPAActual.datetime3(1:end,4)+7;
% for i = 1:1897;if(BPAActual.datetime3(i,4) > 23);BPAActual.datetime3(i,3) = BPAActual.datetime3(i,3) + 1;BPAActual.datetime3(i,4) = BPAActual.datetime3(i,4) - 24;end;end;
% % Convert datevectors to character arrays in the specified format
% BPAActual.datetime4 = datestr(BPAActual.datetime3,'mm/dd/yyyy HH:MM AM');
% % Convert character arrays to cell arrays
% BPAActual.datetime10 = cellstr(BPAActual.datetime4)
% % % Grab every other row (we want ten-minute data, not five-minute)
% BPAActual.data10 = BPAActual.data(1:2:end);
% BPAActual.datetime10 = BPAActual.datetime10(1:2:end);
% % % Reverse the forecast data order
% BPAForecast.data = BPAForecast.data(end:-1:1,:);
% BPAForecast.datetime60 = BPAForecast.datetime60(end:-1:1,:);

%% Generate P_farm and delP_farm
% Because the BPA data is generated only on the hour and only includes
% hourly forecasts, we must interpolate (at least once and often twice) in
% order to generate the necessary forecasts

% First, set up interpolation array for the forecast
forecast_interp = zeros(1,Hp*6);

% Now, we can build P_farm and delP_farm
if ~mod((currentData-1),6)

% If the requested forecast time is on the hour (00:00), we can interpolate
% directly without first performing an initial interpolation

% Set up time array - we need one more than the requested time horizon
% in order to determine delP_farm later
timearray = 0:10:(Hp+1)*10;
% Now, interpolate to get the forecast.
% We create a function that uses both the current data point as well
% as however many hours of forecasted data we need in order to generate
% forecasts through the time horizon
forecast_interp = interp1(0:60:(Hp+6)*10,[BPAActual.data10(currentData) BPAForecast.data(((currentData-1)/6)+1,1:length(0:60:(Hp+6)*10)-1)],timearray);
else
% If the requested forecast time is NOT on the hour, we must first
% interpolate to get the forecast array of hourly predictions and then
% interpolate again to get the ten-minute predictions
% First, determine the ten-minute sample we want for the hourly
% forecast data
timeremainder = mod((currentData-1),6);

% Now, interpolate to get the hourly forecast data
forecast_interp_60 = interp1(0:60:60,BPAForecast.data(((currentData-1-timeremainder)/6)+1:((currentData-1-timeremainder)/6)+2,:),timeremainder*10);

% Set up the time array - we need one more than the requested time
% horizon in order to determine delP_farm later
timearray = 0:10:(Hp+1)*10;

% And then interpolate to get the forecast - same as last time except
% we have a custom-created forecast array to use
forecast_interp = interp1(0:60:(Hp+6)*10,[BPAActual.data10(currentData) forecast_interp_60(1:length(0:60:(Hp+6)*10)-1)],timearray);
end

% Extract P_farm and calculate delP_farm
P_farm = forecast_interp(1:end-1);
delP_farm = diff(forecast_interp);

% Generate P_sched

% Preallocate for P_sched - this is longer than we eventually return
% because it makes the looping easier
P_sched = zeros(1,Hp+7);
% Now we need to build P_sched
%
% We have six separate cases depending on the starting position of our
% currentData pointer. These are roughly laid out in order of decreasing
% code difficulty:
% 1) 00:40
% 2) 00:50
% 3) 00:00
% 4) 00:10
% 5) 00:20
% 6) 00:30
%
% We step through each of the possible cases
if mod(currentData,6) == 5
    % 1) 00:40
    % First, we must ensure that we have sufficient forecast data to
    % generate P_sched directly
    if currentData > 5
        % If we have enough forecast data, we split up the generation
        % of P_sched into two parts: a 'locked' schedule that is
determined at
        % 00:40 of the most current hour and an 'unlocked' schedule
        (at
        % least one hour out) that varies as the forecast is improved
        until
        % it must be locked.
        for i = 1:6:Hp
            if i == 1
                % The 'locked' schedule that is based on the current
                wind
                % production
                % First, we deal with remnants of the previous hour
                % (based
                % on the previous schedule)
                P_sched(i:i+1) = BPAActual.data10(currentData-6);
            end % P_farm forecast (which, for i = 1 is actually equal
            % to
            % the current wind production).
            P_sched(i+3:i+7) = BPAActual.data10(currentData);
            % Then, we deal with the transition point between the
            hours
            P_sched(i+2) = (P_sched(i+1) + P_sched(i+3))/2;
        else
        end
    end
end
The 'unlocked' schedule that is based on the forecasted wind production at the 00:40 point of the next hour.

The next hour's schedule is based on the forecast at the 00:40 point.

\[
P_{\text{sched}}(i+3:i+7) = P_{\text{farm}}(i);
\]

% Deal with the transition point
\[
P_{\text{sched}}(i+2) = \frac{(P_{\text{sched}}(i+1) + P_{\text{sched}}(i+3))}{2};
\]

end
else
% If we don't have enough forecast data, we must fill in the first partial hour as best we can. In this case, we use the first actual wind production number
\[
P_{\text{sched}}(1:2) = \text{BPAActual.data10}(1);
\]

% Now, we proceed as before
for i = 1:6:Hp
    if i == 1
        % First, the 'locked' schedule
        \[
P_{\text{sched}}(i+3:i+7) = \text{BPAActual.data10}(\text{currentData});
        \]
        % Transition point
        \[
P_{\text{sched}}(i+2) = \frac{(P_{\text{sched}}(i+1) + P_{\text{sched}}(i+3))}{2};
        \]
    else
        % Now, the 'unlocked' schedule
        \[
P_{\text{sched}}(i+3:i+7) = P_{\text{farm}}(i);
        \]
        % And the transition point
        \[
P_{\text{sched}}(i+2) = \frac{(P_{\text{sched}}(i+1) + P_{\text{sched}}(i+3))}{2};
        \]
    end
end
end
elseif mod(currentData,6) == 0;
% 2) 00:50
% As before
if currentData > 6
    for i = 1:6:Hp
        if i == 1
            % 'Locked' schedule
            % Note that we must go backwards in time to get the previously scheduled wind production
            \[
P_{\text{sched}}(i) = \text{BPAActual.data10}(\text{currentData}-7);
            \]
            \[
P_{\text{sched}}(i+2:i+6) = \text{BPAActual.data10}(\text{currentData}-1);
            \]
            \[
P_{\text{sched}}(i+1) = \frac{(P_{\text{sched}}(i) + P_{\text{sched}}(i+2))}{2};
            \]
        else
            \[
P_{\text{sched}}(i+3:i+7) = P_{\text{farm}}(i);
            \]
            % Transition point
            \[
P_{\text{sched}}(i+2) = \frac{(P_{\text{sched}}(i+1) + P_{\text{sched}}(i+3))}{2};
            \]
        end
    end
end
else

% 'Unlocked' schedule

% Since i > 1, we are free to go backwards in the
P_farm
% vector to grab the new forecast
P_sched(i+2:i+6) = P_farm(i-1);
P_sched(i+1) = (P_sched(i) + P_sched(i+2))/2;
end
end
else
  % Fill in the partial hour
  P_sched(1) = BPAActual.data10(1);
  for i = 1:6:Hp
    if i == 1
      % 'Locked' schedule
      P_sched(i+1:i+5) = BPAActual.data10(currentData-1);
      P_sched(i) = (BPAActual.data10(currentData-8) +
                    BPAActual.data10(currentData-2))/2;
    else
      % 'Unlocked' schedule
      P_sched(i+1:i+5) = P_farm(i-2);
      % Since i > 1, we can go backwards in time
      P_sched(i) = (P_sched(i-1) + P_sched(i+1))/2;
    end
  end
elseif mod(currentData,6) == 1;
  % 3) 00:00
  % As before, except we must be sure to extend our loop to get the
  % last
  % scheduled point in (since we schedule ahead) in cases where Hp
  % is one
  % sample less than another full hour.
  % And we have to split this into two parts because we don't
  % always have
  % a full hour of previous schedule data
  if currentData > 7
    for i = 1:6:Hp+1
      if i == 1
        % 'Locked' schedule
        P_sched(i+1:i+5) = BPAActual.data10(currentData-2);
        % For the transition point here, we need to go back
        % previous schedule to do the averaging
        P_sched(i) = (BPAActual.data10(currentData-8) +
                      BPAActual.data10(currentData-2))/2;
      else
        % 'Unlocked' schedule
        P_sched(i+1:i+5) = P_farm(i-2);
      end
    end
  end
else
end
end
elseif currentData > 1
  for i = 1:6:Hp+1
    if i == 1
      % ' Locked' schedule
      P_sched(i+1:i+5) = BPAActual.data10(currentData-2);
    end
    if i > 1
      if i == 1
        % ' Locked' schedule
        P_sched(i+1:i+5) = BPAActual.data10(currentData-2);
        % Here, we must get slightly creative because there
        % previous hour's schedule, so we do the best we can
        % grab the first data point to do the averaging
        P_sched(i) = (BPAActual.data10(currentData-6) + BPAActual.data10(currentData-2))/2;
      else
        % ' Unlocked' schedule
        P_sched(i+1:i+5) = P_farm(i-2);
        % Since i > 1, we can go backwards in time
        P_sched(i) = (P_sched(i-1) + P_sched(i+1))/2;
      end
    end
  end
else
  % Fill in the partial hour - there is no 'locked' schedule
  % per se, % in this case because there is no 00:40 data point from the
  % previous hour. We follow convention and set it to the
  % first data
  % point
  P_sched(1:6) = BPAActual.data10(1);
  % Now, we can fill in the 'unlocked' schedule
  for i = 7:6:Hp+1
    P_sched(i+1:i+5) = P_farm(i-2);
    P_sched(i) = (P_sched(i-1) + P_sched(i+1))/2;
  end
end
elseif mod(currentData,6) == 2;
  % 4) 00:10
  % As before, making sure we extend the loop far enough to cover
  % entire horizon
  if currentData > 2
    for i = 1:6:Hp+2
      if i == 1
        % ' Locked' schedule
        P_sched(i+4) = BPAActual.data10(currentData-3);
      else
        % ' Unlocked' schedule
        P_sched(i+4) = P_farm(i-3);
      end
    end
  end
end
else
  % Fill in the partial hour
  P_sched(1:5) = BPAActual.data10(1);
  for i = 7:6:Hp+2
    % And the rest of the 'unlocked' schedule
    P_sched(i:i+4) = P_farm(i-3);
  end
end

% We deal with the transition points afterwards in the final three
% cases because we must have both the locked and unlocked
% schedules in place in order to do the averaging
for i = 6:6:Hp+2
  P_sched(i) = (P_sched(i-1) + P_sched(i+1))/2;
end

elseif mod(currentData,6) == 3;
% 5) 00:20

  % As before
  if currentData > 3
    for i = 1:6:Hp+3
      if i == 1
        % 'Locked' schedule
        P_sched(i:i+3) = BPAActual.data10(currentData-4);
      else
        % 'Unlocked' schedule
        P_sched(i-1:i+3) = P_farm(i-4);
      end
    end
  else
    % Fill in the partial hour
    P_sched(1:4) = BPAActual.data10(1);
    for i = 7:6:Hp+3
      % And the rest of the 'unlocked' schedule
      P_sched(i-1:i+3) = P_farm(i-4);
    end
  end

  % And the transition points
  for i = 5:6:Hp+3
    P_sched(i) = (P_sched(i-1) + P_sched(i+1))/2;
  end

elseif mod(currentData,6) == 4;
% 6) 00:30

  % As before
  if currentData > 4
    for i = 1:6:Hp+4
      if i == 1
        % 'Locked' schedule


P_sched(i:i+2) = BPAActual.data10(currentData-5);
else
    % 'Unlocked' schedule
    P_sched(i-2:i+2) = P_farm(i-5);
end
end
else
    % Fill in the partial hour
    P_sched(1:3) = BPAActual.data10(1);
    for i = 7:6:Hp+4
        % And the rest of the 'unlocked' schedule
        P_sched(i-2:i+2) = P_farm(i-5);
    end
end

% And the transition points
    for i = 4:6:Hp+4
        P_sched(i) = (P_sched(i-1) + P_sched(i+1))/2;
    end
end

% Finally, we capture the segment of P_sched that we want
P_sched = P_sched(1:Hp+1);

% Normalize outputs

% Convert to per-unit for use in the MPC controller
P_sched = P_sched/windBase;
P_farm = P_farm/windBase;
delP_farm = delP_farm/windBase;

% And transpose
P_sched = P_sched';
P_farm = P_farm';
delP_farm = delP_farm';

end
APPENDIX 7: Make Q Cost Matrix Code

function [Q] = makeQ(Hp, Qweight, derateFactQ, predict)
% Function to generate the Q matrix
% Takes Hp, Q weights (a vector), and derateFactQ (derating factor)
% as inputs
% If derateFactQ is not specified, it is assumed to be 0.
% Outputs the Q matrix

% if nargin < 3, derateFactQ = 0; end

if predict == 1
    Q = [];
    for i = 1:(Hp+1)
        temp = diag(Qweight)*exp((-derateFactQ)*(i-1));
        Q = blkdiag(Q,temp);
    end
else
    Q = zeros(6*(Hp+1));
    Q(1) = Qweight(1);
end
end
APPENDIX 8: Make R Cost Matrix Code

```matlab
function [R] = makeR(Hp, Rweight, derateFactR)
% Function to generate the R matrix
% Takes Hp, R weight (a single value), and derateFactR (derating factor) as inputs
% Outputs the R matrix

temp = zeros(Hp+1,1);

for i = 1:Hp+1
    temp(i) = Rweight*exp((-derateFactR)*(i-1));
end

R = diag(temp);
end
```
APPENDIX 9: Make $T_k$ Tracking Vector Code

```matlab
function [T_k] = makeT_k(Hp)
% Function to generate $T_k$ vector for MPC formulation
% Takes Hp as an input
% Generates the $T_k$ vector as an output

P_error = zeros(Hp+1,1);
delP_ES = zeros(Hp+1,1);
P_plant = zeros(Hp+1,1);
delP_plant = zeros(Hp+1,1);
SOC = zeros(Hp+1,1);
SOC_new = zeros(Hp+1,1);

T_k = zeros(6*(Hp+1),1);

for i = 1:(Hp+1)
    T_k((i-1)*6+1:i*6) = [P_error(i); delP_ES(i); P_plant(i);
                          delP_plant(i); SOC(i); SOC_new(i)];
end
end
```
APPENDIX 10: Make V_k Vector Code

```matlab
function [V_k] = makeV_k(Hp, P_agc, P_sched, P_farm, delP_farm, P_loss)
% Function to generate V_k vector for MPC formulation
% Takes vectors of length Hp+1 as input (P_agc, P_sched, P_farm, delP_farm, P_loss) and generates the V_k vector as output

%% Size checks
if (size(P_agc,1) ~= (Hp+1))
    error('P_agc is the wrong size');
end
if (size(P_sched,1) ~= (Hp+1))
    error('P_sched is the wrong size');
end
if (size(P_farm,1) ~= (Hp+1))
    error('P_farm is the wrong size');
end
if (size(delP_farm,1) ~= (Hp+1))
    error('delP_farm is the wrong size');
end
if (size(P_loss,1) ~= (Hp+1))
    error('P_loss is the wrong size');
end

%% Generate V_k
V_k = zeros(5*(Hp+1),1);

for i = 1:Hp+1
    V_k(((i-1)*5+1:i+5) = [P_agc(i); P_sched(i); P_farm(i); delP_farm(i); P_loss(i)];
end
end
```
APPENDIX 11: State Space Model Code

```matlab
function [x_k_new, y_k] = runModel(A, Bu, Bv, C, Du, Dv, x_k, u_k, v_k, u_const_up, u_const_low)
% Function to generate the new state using the state-space model
% x(k+1) = A*x(k) + B_u*u(k) + B_v*v(k)
% y(k) = C*x(k) + D_u*u(k) + D_v*v(k)
% Takes A, Bu, Bv, C, Du, Dv, x_k, u_k, v_k, u_const_up, u_const_low
% as inputs
% Generates x_k_new and y_k as outputs

% Check to see if P_es violates the limits.
% If so, set P_es to the violated limit.
if (u_k(1) > u_const_up)
    u_k(1) = u_const_up;
    disp('Invalid P_es (upper)');
elseif (u_k(1) < u_const_low)
    u_k(1) = u_const_low;
    disp('Invalid P_es (lower)');
end

% Run the model
x_k_new = A*x_k + Bu*u_k + Bv*v_k;
y_k = C*x_k + Du*u_k + Dv*v_k;
end
```
APPENDIX 12: Plotting Code

% MPA Thesis plotting code
% Created 4/18/2012 by Michael P. Antonishen

close all
clc

load 'ExpOutputs041712.mat'
load 'ExpInputs2.mat'

%%% Switches
%
% Switches barplots on and off
barplots = 0;
costplots = 0;
errorplots = 1;

%%% Initialization and analysis
%
% Vector initialization
MAE = zeros(3,5);
J = zeros(3,5);
follow = zeros(27,1);
imbal = zeros(27,1);
Imbalance = zeros(3,5);
Following = zeros(3,5);
FinResult = zeros(2,2);

% Adds incs and decs together into one reserves vector
for i = 1:21;
    follow(i) = followMax(i)+abs(followMin(i));
    imbal(i) = imbalMax(i)+abs(imbalMin(i));
end

% Calculate P_plant
P_plant = P_es_rec + P_farm_rec;

% Calculates how much reserves are reduced in terms of % compared to base
% case (case 2) -- meteor 60 minutes, no ES
followreduced = (follow./follow(2))*100;
imbalreduced = (imbal./imbal(2))*100;

% change J cost into $/kW/Mo
J_kWMo = (JCostTotal./888)*(4320/100000);  % 888 decaminutes tested, 4320 per month, 100,000 kW farm.
R_kWMo = (RCostTotal./888)*(4320/100000);
Q_kWMo = (QCostTotal./888)*(4320/100000);

% Populating vectors for barplot
Imbalance(1,1:5) = imbalreduced(7:11);
Imbalance(2,1:5) = imbalreduced(12:16);
Imbalance(3,1:5) = imbalreduced(17:21);

Following(1,1:5) = followreduced(7:11);
Following(2,1:5) = followreduced(12:16);
Following(3,1:5) = followreduced(17:21);

MAE(1,1:5) = P_error_MAE(7:11);
MAE(2,1:5) = P_error_MAE(12:16);
MAE(3,1:5) = P_error_MAE(17:21);

J(1,1:5) = J_kWMo(7:11);
J(2,1:5) = J_kWMo(12:16);
J(3,1:5) = J_kWMo(17:21);

% [cost, error]
FinResult(1,1:2) = [.3996 .0454];
FinResult(2,1:2) = [.0145 .002];

% Plotting
if errorplots == 1
    figure;
    subplot(1,1,1,'FontSize',14);
    plot(P_error_rec(:,2),'k');
    hold on
    plot(P_error_rec(:,3),'r');
    plot(P_error_rec(:,4),'b');
    plot(P_error_rec(:,5),'g');
    axis([288 432 -0.4 .4]);
    legend('Case 1','Case 2','Case 3','Case 4', 'Location', 'NorthWest');
    ylabel('Error [pu]');
    xlabel('Time (days)');
    printStr = strcat('.\images\Fig','ErrorPlot');
    print('-dpng','-r600',printStr);
figure;
subplot(1,1,1,'FontSize',14);
plot(P_sched_rec(:,21),'r');
hold on
plot(P_plant(:,21),'b');
plot(P_error_rec(:,21),'m');
plot(SOC_rec(:,21),'c');
axis([0 900 -0.25 1.1]);
legend('P_{sched}','P_{plant}', 'P_{error}', 'SOC', 'Location', 'NorthWest')
ylabel('PU');
set(gca,'xtick', (0:24*6:24*6*6), 'XTickLabel',{'0','1','2','3','4','5','6'});
xlabel('Time (days)');
printStr = strcat('.\images\Fig','PplantPlot');
print('-dpng','-r600',printStr);
end

if costplots == 1
% Log plot of R sweep
figure;
subplot(1,1,1,'FontSize',14);
semilogx(ExpInputs(11,22:49),J_kWMo(22:49),'-k*'
ExpInputs(11,22:49),Q_kWMo(22:49),'-r*
ExpInputs(11,22:49),R_kWMo(22:49),'-b*'
legend('J Cost','J_Q Cost','J_R Cost', 'Location', 'Best')
ylabel('Cost [$/kW/Mo]');
xlabel('R Input cost [$/pu^2 per 10m]');
printStr = strcat('.\images\Fig','RCostSweepLog');
print('-dpng','-r600',printStr);

% Zoomed plot of R sweep
figure;
subplot(1,1,1,'FontSize',14);
plot(ExpInputs(11,25:49),JCostTotal(25:49),'-k*'
ExpInputs(11,25:49),QCostTotal(25:49),'-r*
ExpInputs(11,25:49),RCostTotal(25:49),'-b*'
legend('J Cost','J_Q Cost','J_R Cost', 'Location', 'Best')
ylabel('Cost [$]');
xlabel('R Input cost [$/pu^2 per 10m]');
printStr = strcat('.\images\Fig','RCostSweepZoomed');
print('-dpng','-r600',printStr);

% Plot of Money to spend capitally (20 year lifespan) vs. damage cost/10 minutes
figure;
subplot(1,1,1,'FontSize',14);
semilogx(ExpInputs(11,22:49),((8212.1789 - QCostTotal(22:49))*52*20),'-k*');
plt.plot(ExpInputs(11,22:49),((8212.1789 - QCostTotal(22:49))*52*20),'-k*');

%legend('J Cost','J Q Cost','J R Cost', 'Location', 'Best')
xlabel('R input cost [$/pu^2 per 10m]');
ylabel('Money saved [$/20 year life]');

printStr = strcat('.\images\Fig','RCostsTotalSweepLog');
print('-dpng','-r600',printStr);
end

if barplots == 1
    figure;
    subplot(1,1,1,'FontSize',14);
    MAEBar = bar(MAE);
    set(get(MAEBar(1),'BaseLine'),'LineWidth',2,'LineStyle',':');
    colormap summer % Change the color scheme
    set(gca,'XTickLabel',{'';'Persistence';'Meteorological 60';'Meteorological 30'});
    legend('H_p = 6', 'H_p = 12', 'H_p = 18', 'H_p = 24', 'H_p = 30', 'Location', 'Best')
    ylabel('Mean Absolute Error [PU]');
    printStr = strcat('.\images\Fig','MAEBarPlot');
    print('-dpng','-r600',printStr);

    figure;
    subplot(1,1,1,'FontSize',14);
    JBar = bar(J);
    set(get(JBar(1),'BaseLine'),'LineWidth',2,'LineStyle',':');
    colormap summer % Change the color scheme
    set(gca,'XTickLabel',{'';'Persistence';'Meteorological 60';'Meteorological 30'});
    legend('H_p = 6', 'H_p = 12', 'H_p = 18', 'H_p = 24', 'H_p = 30', 'Location', 'Best')
    ylabel('Cost [$/kW/mo]');
    printStr = strcat('.\images\Fig','JCostBarPlot');
    print('-dpng','-r600',printStr);

    figure;
    subplot(1,1,1,'FontSize',14);
    ImbalanceBar = bar(Imbalance);
    set(get(ImbalanceBar(1),'BaseLine'),'LineWidth',2,'LineStyle',':');
    colormap summer % Change the color scheme
set(gca,'XTickLabel',{"Persistence","Meteorological 60","Meteorological 30"})
legend('H_p = 6', 'H_p = 12', 'H_p = 18', 'H_p = 24', 'H_p = 30', 'Location', 'NorthEast')
ylabel('Imbalance Reserves as a percent of base case [%]');
printStr = strcat('.\images\Fig','ImbalanceBarPlot');
print('-dpng','-r600',printStr);

figure;
subplot(1,1,1,'FontSize',14);
FollowingBar = bar(Following);
set(get(FollowingBar(1),'BaseLine'),'LineWidth',2,'LineStyle',':')
colormap summer % Change the color scheme
set(gca,'XTickLabel',{"Persistence","Meteorological 60","Meteorological 30"})
legend('H_p = 6', 'H_p = 12', 'H_p = 18', 'H_p = 24', 'H_p = 30', 'Location', 'NorthEast')
ylabel('Following Reserves as a percent of base case [%]');
printStr = strcat('.\images\Fig','FollowingBarPlot');
print('-dpng','-r600',printStr);

figure;
subplot(1,1,1,'FontSize',14);
FinResultBar = bar(FinResult);
set(get(FinResultBar(1),'BaseLine'),'LineWidth',2,'LineStyle',':')
colormap summer % Change the color scheme
set(gca,'XTickLabel',{"Base Case","Best Case"})
legend('Total J Cost [$/kW/mo]', 'Mean Absolute Error [pu]', 'Location', 'Best')
%ylabel('Mean Absolute Error [PU]');
printStr = strcat('.\images\Fig','MAEBarPlot');
print('-dpng','-r600',printStr);

end