NASA/CR—2008-215240

Intelligent Engine Systems

Adaptive Control

Nathan Gibson
General Electric Aircraft Engines, Cincinnati, Ohio

June 2008

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 301–621–0134

- Telephone the NASA STI Help Desk at 301–621–0390

- Write to:
  NASA Center for AeroSpace Information (CASI)
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/CR—2008-215240

Intelligent Engine Systems

Adaptive Control

Nathan Gibson
General Electric Aircraft Engines, Cincinnati, Ohio

Prepared under Contract NAS3–01135, Work element 2.1, Task order 37

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

June 2008

This work was sponsored by the Fundamental Aeronautics Program at the NASA Glenn Research Center.

Level of Review: This material has been technically reviewed by NASA technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Available electronically at http://gltrs.grc.nasa.gov

Table of Contents

Intelligent Engine Systems
Adaptive Control

Nathan Gibson
General Electric Aircraft Engines
Cincinnati, Ohio 45215

1. Introduction

The primary purpose of this study is to determine the variations in the computational effort as the update rate, control horizon, and prediction horizon are varied for a Model Predictive Control (MPC) algorithm applied to a high-bypass turbofan commercial engine application. In addition, we have examined the variations that occur in the performance of the MPC algorithm as these variations are introduced, in terms of the algorithm's ability to track the two tracked output variables – thrust and T41 – and the frequency and magnitude of violations of the "soft" constraints on stall margins and other engine variables.

Our results demonstrate that significant reductions in the computational effort required for MPC can be obtained by reducing the rate at which MPC updates are computed, by using control horizons that are as short as possible, and, if possible by limiting the consideration of constraints to only those that are likely to be active at the flight point of interest. However, these alterations to the baseline MPC strategy generally incur penalties in terms of performance (thrust tracking error, maximum T41 value, violation of constraints). This document quantifies some of these performance penalties.

Near the end of this document, we discuss some possible areas for future study on how MPC might be implemented in a more efficient fashion.

1.1 Background and Overview of MPC

In general, the concept behind Model Predictive Control (MPC) is that a reasonably accurate model can be used to predict what the inputs to the system will cause the outputs to do in response. If such a model is available in real time to a controller, then the controller can use this predictive capability to optimize the current inputs, where any constraints that may be present on the system outputs can also be considered. For any particular application, the user of MPC must specify the constraints on the outputs that must be considered, any constraints on how the input variables can change, the cost function to be minimized by the inputs, and algorithm to be used for accomplishing the minimization. Most MPC implementations use a quadratic cost function of some sort, and then use available quadratic programming techniques to do the optimization.

For MPC the primary goal is always to generate the commanded thrust without violating constraints on several of the engine variables, while keeping the turbine inlet temperature (T41) as low as possible. In a previous study of MPC for a similar engine model [1], the feasibility of MPC was demonstrated. That study developed a cost function for MPC that produced valid

results at several flight points for several transient cases, and showed that MPC could work across a wide range of deterioration conditions. The previous study produced the baseline MPC implementation that is used, after some modifications, in this investigation.

Based on the previous study, the objective of MPC for the at any flight point is to track reference trajectories for thrust and T41 subject to hard constraints on the MPC-determined input variables and "soft" constraints on several other engine variables (stall margins, pressure limits, rotor speeds, etc.). "Soft" constraints in this context are constraints that can be violated, but for which large penalties are included in the MPC cost function to be minimized, thereby making large violations unlikely. The reference trajectories for both thrust and T41 are generated by running a simulation of a "new" engine at the appropriate flight point using previously stored FADEC input histories, and then storing the resulting thrust and T41 histories. The MPC algorithm includes a first-order lag pre-filter on the reference trajectories with a time constant of 1.4 seconds that was selected during the previous MPC study. This pre-filter delays the thrust and T41 reference trajectories slightly relative to what a FADEC-controlled new engine would produce, as we shall see in the results below.

The MPC-determined inputs for all cases are the following three engine inputs:

1) Main fuel flow rate (WF36)
2) Variable bleed valve position (AE24)
3) Variable stator vane position (STP25)

The MPC algorithm determines the optimal values of these inputs using an embedded model of the engine, which is a component-level model. At each MPC update time, the MPC algorithm uses the embedded model to calculate a linearized, simplified approximation of the system dynamics. The algorithm then uses this linearized model to minimize a cost function that weights errors in tracking the tracked outputs and the violations of any "soft" constraints over the prediction horizon, and also weights the changes in the MPC-determined inputs (listed above) over the control horizon. The input changes are also subject to hard limits, as will be discussed momentarily. We left the relative weights on the tracking errors, the violations of the "soft" constraints, and the changes to the inputs at the baseline values from the previous MPC study for all runs in the work reported here. The particular deteriorated engine used for all simulation runs here was also the same as the prior MPC study. This makes all of the results reported here directly comparable in terms of performance values and computational effort for MPC.

The input variables are limited in the MPC algorithm to lie between specified limits for their maximum and minimum values, and are also limited in the amount by which they can change from one MPC update step to the next. This latter constraint avoids violation of the rate limits on the inputs. The MPC algorithm treats these input constraints as "hard" constraints, and therefore will not call for input changes that violate them. As noted in a later section, however, we had to make a modification to the baseline MPC implementation to avoid violating the input rate limits with the MPC inputs when MPC updates are computed less frequently than every minor simulation time frame.

The seven engine variables for which constraints are specified are:

1) Fan rotor speed (XN2) – maximum
2) Core rotor speed (XN25) – maximum
3) Burner static pressure (PS3) – maximum
4) Fan stall margin (SMfan) – minimum
5) Booster stall margin (SMboost) – minimum
6) Compressor stall margin (SMcomp) – minimum
7) Turbine stage 1 clearance (CLRst1) – minimum

In the MPC algorithm, these seven constraints are treated as "soft" constraints. In other words, these constraints can be violated, but the MPC cost function for optimization puts a weight on predicted violations of these constraints, thereby discouraging violations.

It is worth noting at this point that the only "soft" constraint violations observed in the MPC results across all five of the flight points considered were for the fan, booster, and compressor stall margins and the Stage 1 turbine clearance. This motivates a later study of the computational effort and performance of MPC with a reduced number of constraints considered in the MPC implementation.

1.2 Alterations to MPC operation

1.2.1 Addition of monitoring variables (CPU times for linearization and optimization)

In order to measure the computational effort required for MPC, we need some measure of the computation time required to execute the steps of the MPC algorithm. Matlab provides a built-in function that can gives us some indication of the computational effort involved. This function is 'cputime'. Upon any call to cputime, the accumulated CPU time in seconds since the start of the current Matlab session is returned. By storing this value just before a step in a Matlab script or function, and then observing it again just after this step or function, the CPU time for this step can be measured as the difference between the two observed cputime values.

We used this cputime function to measure the computation time required for two separate steps in the MPC optimization implementation: The linearization of the embedded CLM to get the approximate linear model used by MPC for predictions of the values of the outputs and constrained variables, and the actual optimization itself. Because the MPC optimization and linearization steps occur in the MPC block of a Simulink model that implements the simulation of an MPC-controlled engine, we had to alter this Simulink model to compute and produce as outputs the linearization and optimization CPU times. In particular, the S-function implementing the MPC algorithm was modified to add three more output variables – a binary index indicating whether or not MPC updates were calculated at the current simulation time step, the CPU time required for linearization, and the CPU time required for optimization. The latter two values are both zero if no MPC update is executed at the current time step of the simulation.

We then compute the two totals of the CPU time increments over a transient simulation run to arrive at a total CPU time for all MPC optimization steps and for all linearization steps during the run. These totals can then be compared among different simulation runs to examine the effects of update rate, control horizon, and prediction horizon on the computational effort required. Although we have computed CPU effort for linearization throughout our study, we do not present results for linearization computation effort here because the time required for linearization varies little with control horizon or prediction horizon, and essentially is just a function of how many MPC updates are performed.

The Matlab cputime function has a resolution of 1/64 sec (.015625 sec or 15.625 msec). Therefore, the CPU time increments are always the nearest integer multiple of this resolution. This means that many of the CPU increments for a particular MPC simulation run may be zero because the CPU time used for the MPC optimization step is less than half of .015625 sec, hence an increment of the CPU time does not occur. The sum of all CPU increments over a simulation run will then be biased toward low values for these cases. The MPC runs considered here are generally for time periods ranging from 25 sec to 45 sec. With a minor time frame length of 15 msec for all cases considered here, this means the simulations of the transient cases examined here involve anywhere from 1670 to 3000 minor frames. With MPC updates occurring anywhere from every minor frame to every 10 minor frames, the number of MPC updates in any particular run can then range from 167 to 3000.

To make the total CPU times comparable for the various combinations of flight points, MPC update rates, and control and prediction horizon lengths considered here, all cases were run on the same PC with the same implementation of Matlab/Simulink and with essentially no other processes running on the PC at the same time. The PC used for these studies is a 3GHz Intel Pentium 4 machine with 0.5 GB RAM running Microsoft Windows 2000 version 5.0 (Service Pack 4). The Matlab/Simulink versions are Matlab 7.0 and Simulink 6.1 (Revision 14, Service Pack 1).

For short control and prediction horizons, the average CPU time per MPC update on this machine with this software implementation is typically around 5 msec. This is only about one-third of the CPU time resolution for Matlab's cputime function noted above. Hence, there are typically a large number of MPC updates that produce Matlab-indicated CPU increments of zero during each MPC simulation run. The reader is therefore cautioned about drawing firm conclusions regarding absolute CPU times from the results presented here. However, by normalizing the total CPU time used during MPC optimization over each run to a baseline case for that run, valid comparisons of computational effort for MPC under various conditions can be made, and these comparative results are presented in what follows. This baseline case for each flight point is usually the case with MPC updates every minor frame, with the control and prediction horizons both equal to 1 MPC update step, and with all 7 constraints from the original MPC baseline study case considered. For a few flight points, this baseline case varies due to problems with MPC convergence for that particular flight point and transient.

1.2.2 Correction to discretization of pre-filter model

The baseline implementation of MPC from the prior MPC study included a slight flaw in the implementation of the pre-filter used on the reference values of thrust and T41. The reference pre-filter is specified as a continuous-time state space system, which is then converted within the MPOC algorithm to a discrete-time form using an Euler approximation. In the baseline implementation, the Euler approximation used a user-specified time step length between MPC updates for this approximation. However, there was no restriction on this user-specified time period to guarantee that it was an integer multiple of the minor time frame for the simulation. As a result, the pre-filter could be converted to a discrete-time form that was incorrect for the actual time interval between MPC updates, which must *always* be an integer multiple of the minor time frame used for simulation. This flaw could cause the pre-filtered references to differ from the desired trajectories based on the continuous time pre-filter dynamics.

We have corrected this flaw in our modified MPC implementation. There is now a built-in check to make sure the time interval used for pre-filter conversion to discrete-time form is an integer multiple of the minor time frame. If the user specifies something other than this for the MPC update cycle length, the modified algorithm converts this request to the nearest integer multiple of the minor frame.

1.2.3 First-order hold for slower MPC updates

The baseline implementation of MPC from the previous study allowed for the possibility of performing MPC updates less frequently than every minor simulation frame. However, this implementation was flawed. If the MPC update time increment was specified to be larger than the minor frame length, then the set-up for MPC optimization increased the allowable input increments by a factor equal to the ratio of the MPC update interval length to the simulation minor frame length. Ostensibly, this modification allows the input to vary a larger amount over the longer MPC update interval while still maintaining the appropriate limit on the rate of change of the input variables.

However, the baseline implementation of MPC in the previous study then held the inputs at the previously computed MPC input values between the MPC update times. Therefore, the baseline MPC implementation introduced the full input increments due to the newly computed MPC optimal inputs at the simulation time step immediately following each MPC update time. Because the inputs were still being held at the previous MPC update values until just one minor frame before this increment was introduced, the input increments often violated the specified limits on the rates of change of the inputs. This forced the rate limits to be applied by the physical actuator limits implemented in the simulated engine model, and the baseline MPC implementation had no such physical limits implemented in its model for the engine.

To avoid this problem, we introduced a modification to the way in which the MPC inputs are applied when MPC updates are computed less frequently than every minor frame. In particular, we introduced a first-order hold into the MPC input implementation. A first-order hold

generates an input value that varies linearly over the time interval between two consecutive specified input values. In our modified MPC implementation, we use the last updated MPC input values and the new updated values as the two endpoints for the linear input variation over the time interval between MPC updates. Thus, the MPC-controlled inputs are incremented at each minor frame by a fraction of the difference between the old MPC inputs and the new. This fraction is the inverse of the number of minor frames per MPC update cycle. Thus, if MPC updates are computed every 5 minor frames, once the newly optimized MPC input values are computed, the inputs, which are currently at the MPC input values from the last optimization step, are incremented by 1/5 of the difference between the old values and the new values. This is repeated for each minor frame until the next MPC update time. The new MPC input values are therefore reached one minor frame prior to the next MPC update time, and the input rate constraints are not violated.
To implement this change in the MPC algorithm, we edited the S-function implementing the MPC block. In particular, we defined a global variable in this S-function that holds the increment of the MPC inputs from the last optimization to the current one. The S-function then uses this increment with the value of the number of minor frames between MPC updates to properly increment the MPC inputs at each minor frame between MPC updates.

We tested this modified MPC implementation by simulating a transient with the actuator rate limits set in the simulation model of the "true" engine, and comparing these results to a case without actuator limits but with the modified MPC implementation including the first-order hold in place. The results were nearly identical. (The results are never exactly identical because there are situations where the MPC updates are large enough to violate the limits on the rates of change of the inputs, but not so large that they cause the actuator rate limits to be saturated for the entire time between MPC updates.)

1.2.4 Stall margin limit modification

For some of the flight points examined in this study, the quadratic programming optimization routine in the MPC algorithm would not converge within the allowable number of iterations at the steady state initial conditions for many (in some cases, all) combinations of the MPC update rate, the MPC control horizon and the MPC prediction horizon. Although the diagnostics produced by Matlab for the MPC optimization algorithm did not make clear the exact reasons for this non-convergence, it was apparent that at least part of the convergence problem was with one or more of the stall margin constraints on the fan, booster and compressor. In particular, it appeared that the iterative optimization algorithm could not "find" a solution for the inputs that simultaneously avoided violation of the input constraints, and avoided large violations of the "soft" constraints on these three stall margins without incurring large thrust tracking errors.

In the baseline MPC algorithm from the previous study, the three stall margin limits are set at 95% of the minimum stall margins observed during a transient run of a new engine with open-loop FADEC inputs at the appropriate flight point. However, all simulated MPC test runs here are for a deteriorated engine, with the stall margin limits treated as soft constraints. Therefore, any stall margin violation is heavily weighted in the cost function to be minimized, although

violations are not completely ruled out. Avoiding significant violation of the limits shown above, set by a new engine, with a deteriorated engine is quite difficult.

For the SLS, idle-takeoff case, the fan and booster stall margin limits are both violated using the open-loop FADEC inputs, and the compressor stall margin misses violation of the limit by just 0.5%. For the single case at altitude, the booster stall margin limit is substantially in violation of the limit using the FADEC inputs.

Based on this observation, we chose to modify the three stall margin limits to give the MPC algorithm a chance to find optimal inputs for a deteriorated engine without such strict limits on the minimum stall margins. In particular, we modified each stall margin limit to be the larger of: 50% of the minimum stall margin observed for FADEC control of a new engine, or the stall margin limit that applied to the original SLS, idle-takeoff case that we considered as the baseline for our MPC investigation.

The major impact of the stall margin limit modification is on the fan and booster stall margin limits for the altitude cases. But the relaxed booster stall margin limit is still above the observed minimum stall margin for the (20k, Mach 0.5) reverse Bode transient case for a deteriorated engine using FADEC inputs. And while the modified fan stall margin limit for the altitude cases is now substantially lower than 95% of the minimum fan stall margin observed with FADEC inputs for either a new or deteriorated engine, it is still substantially above the fan stall margin that is permitted at sea-level conditions.

This relaxation of the stall margin limits allowed successful simulation of the MPC control algorithm for all of the flight points of interest with only a few combinations of MPC update rate, control horizon, and prediction horizon excepted. For the purposes of showing performance relative to stall margin herein, stall margins have been normalized to unity, so a violation of a particular stall margin is less than one.

2. MPC Effort and Performance for Baseline MPC Case

We began the study of MPC performance with a baseline transient case provided with the MPC code from the previous MPC study. This baseline case is a 30-second, idle-to-takeoff transient at sea level, static (SLS) standard conditions. This transient includes a brief initial power increase from about 4 seconds to about 6 seconds, followed by a larger acceleration to full takeoff power from about 14 seconds to about 16 seconds. The reference thrust and T41 histories are similar to the thrust and T41 histories shown in Figures 2.1.1 and 2.1.2 below.

All runs were executed on a 3 GHz Pentium 4 PC with about 0.5 GB of installed RAM running the Windows 2000 operating system with no other applications (other than a file explorer window) running. Thus, while the absolute numbers for CPU time required may vary considerably from machine to machine, the comparisons between the CPU times required should be meaningful here with all runs executed on the same machine.

2.1 Baseline MPC Case with MPC Updates Every Minor Frame

For the initial study of the effect of control horizon and prediction horizon, we ran this transient with MPC updates at every minor time frame of the simulation, i.e., with MPC updates every 15 msec. Because the transients and MPC times are identical for all cases, the time histories of the outputs to be tracked (pre-filtered thrust and T41 histories) are identical for all cases.

The mean squared output errors and the mean squared input deviations are formed by summing the squared errors (or deviations) at the time points where new MPC optimizations are performed (for this particular case, that is at every simulation time sample, as noted above) and dividing the resulting sum by the number of time points at which MPC optimization is performed (in this case, all 2001 points).

The format for the identifying the MPC cases in the results shown below (and throughout this report) is (x,y) where x is the control horizon (in MPC cycles) and y is the prediction horizon (also in MPC cycles).

The CPU times shown below are totaled over the entire simulation run for this baseline case (2001 calls to the MPC linearization and optimization steps). As discussed above, Matlab returns CPU times rounded to the nearest increment of 1/64 second (or 15.625 msec). Therefore, there are often linearization and/or optimization calls that produce an indicated CPU time of 0 (if the actual CPU time is 7.8 msec or less, it will be rounded to zero increments of 15.6 msec). Therefore, the total CPU times shown must be considered with care, especially when the total time is such that the average CPU time per call is on the order of 10 msec or less, i.e. when the total time for these 2001-point runs is 20 sec or less.

Table 2.1.1 shows the CPU times (total for the run, in seconds), mean square and peak errors in the tracked outputs (lb for thrust, deg R for T41), and mean square deviations in the MPC-controlled inputs (pph for WF36, angles for AE24 and STP25) for several combinations of prediction horizon and control horizon.

Table 2.1.1. Normalized MPC computation and performance results for baseline MPC case.

Case	Lineariz. CPU time (s)	Optimiz. CPU time (s)	MS Thrust Error	MS T41 Error	Peak Thrust Error	Peak T41 Error	MS WF36 Deviation	MS AE24 Deviation	MS STP25 Deviation
(1,1)	7.4531	17.578	3.38E-05	0.00314	0.0325	0.210	0.066	0.0120	3.23E-04
(1,2)	12.25	16.438	2.86E-05	0.00317	0.0307	0.218	0.082	0.0136	4.76E-04
(1,3)	12.281	17.266	2.47E-05	0.00320	0.0290	0.218	0.101	0.0144	5.68E-04
(2,2)	12.125	21.609	1.77E-05	0.00316	0.0253	0.208	0.063	0.0127	4.40E-04
(2,3)	12.391	22	1.48E-05	0.00318	0.0237	0.211	0.069	0.0132	4.95E-04
(2,6)	12.313	22.281	9.72E-06	0.00322	0.0203	0.223	0.074	0.0133	6.22E-04
(2,10)	12.516	23.906	6.13E-06	0.00325	0.0166	0.226	0.083	0.0104	6.49E-04
(3,3)	12.266	27.141	1.14E-05	0.00320	0.0213	0.211	0.066	0.0130	4.86E-04
(3,6)	12.484	28.688	6.36E-06	0.00322	0.0167	0.215	0.077	0.0130	5.65E-04
(3,10)	12.875	30.328	3.62E-06	0.00324	0.0131	0.224	0.069	0.0124	6.19E-04
(5,6)	12.109	42.984	4.68E-06	0.00328	0.0147	0.224	0.080	0.0129	5.43E-04
(5,10)	12.609	44.094	2.11E-06	0.00330	0.0099	0.224	0.096	0.0128	5.69E-04

We can make some brief and preliminary conclusions based on these results:

1) The CPU time required for linearization at each MPC step is relatively short, and it is not generally affected by the choice of prediction or control horizon (as should be expected). For this reason, we will not consider the computation time for linearization in the rest of this report.

2) The CPU time required for optimization is affected by the length of each of the horizons. But is primarily affected by the control horizon length, rising in this baseline case by about 30% when the control horizon is increased from 1 MPC update cycle to 2, and just under another 30% with an increase from 2 MPC cycles to 3, and then about 50% in increasing from 3 MPC cycles to 5.

The latter of the two preceding conclusions suggests that the computational burden of MPC (in terms of CPU time alone – memory and input/output requirements are not examined here) increases between 25% and 30% for each single cycle increase in the MPC control horizon, with increases in the prediction horizon making only a small difference to computational effort. This approximate relationship will be true for essentially all of the cases considered in this report.

The performance results for these baseline MPC cases are shown in Figures 2.1.1 through 2.1.12 below. They clearly indicate that the MPC optimization is set up primarily to track the thrust reference trajectory with substantially smaller penalties on T41 errors, input deviations, and violations of the limits on the "soft" constraints.

For this baseline case, the thrust tracking error drops substantially, both in terms of mean squared error and peak error, as the control and prediction horizons are increased, with the control

horizon having a slightly larger impact than the prediction horizon. The root mean square and peak thrust tracking errors are both reduced by nearly a factor of 4 in going from the shortest control and prediction horizons considered (both one MPC cycle) and the longest (5 samples for the control horizon and 10 for the prediction horizon).

On the other hand, the T41 errors are nearly insensitive to both horizon lengths in this case, and, if anything, show a slight trend toward increasing as the horizons become larger.

The input deviations are also not affected much by the control and prediction horizons either. However, there is a clear trend that the WF36 input deviations are larger for longer prediction horizons. There is no discernible trend to the AE24 input deviations. The STP25 deviations generally become smaller for larger control horizons, but trend in the opposite direction with larger prediction horizons, and appear to depend more on the prediction horizon than the control horizon. In any case, the changes in the AE24 and STP25 deviations are small as the horizons are varied. We will see later that the frequency of MPC updates plays a larger role in the input deviations than the control or prediction horizons do.

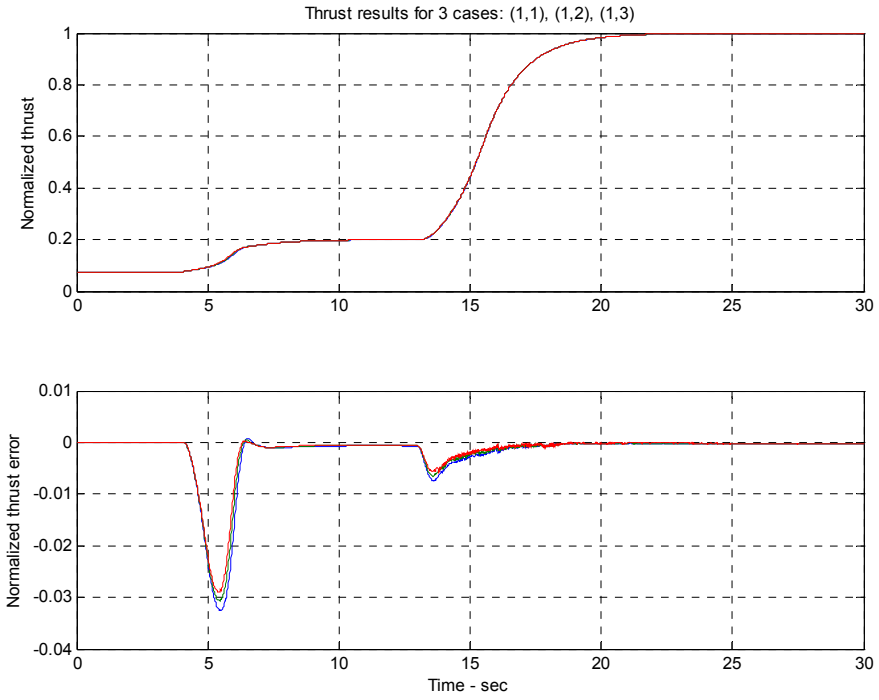

Figure 2.1.1. Thrust results for MPC cases with control horizon =1, various prediction horizons.

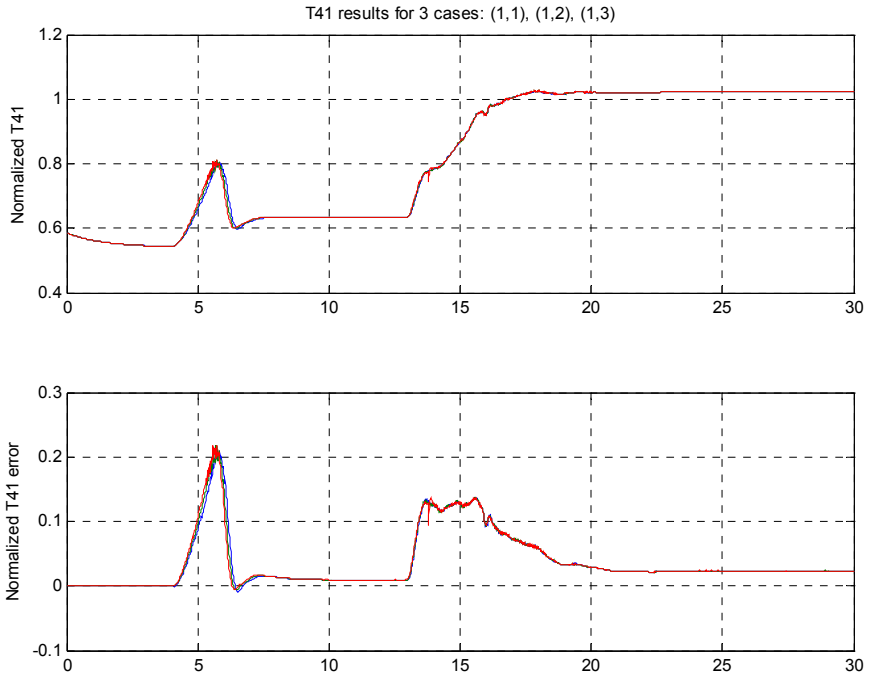

Figure 2.1.2. T41 results for cases with control horizon =1, various prediction horizons.

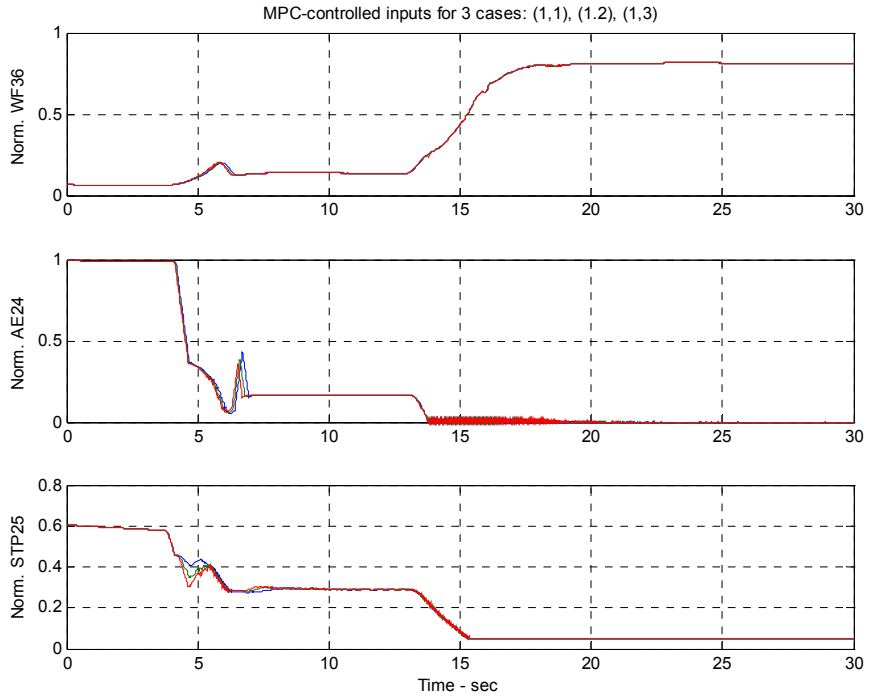

Figure 2.1.3. MPC inputs for cases with control horizon =1, various prediction horizons.

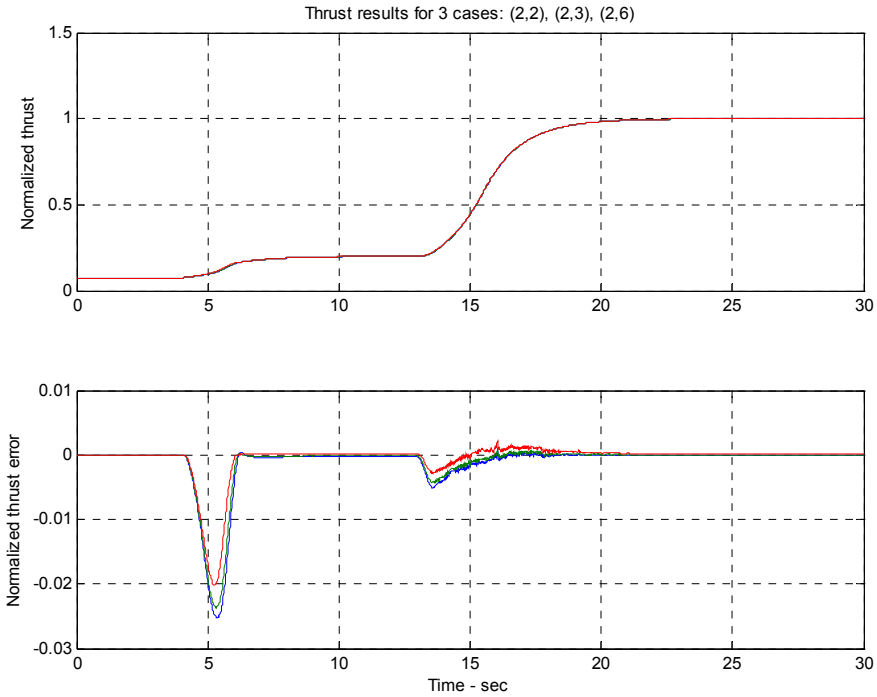

Figure 2.1.4. Thrust results for cases with control horizon =2, various prediction horizons.

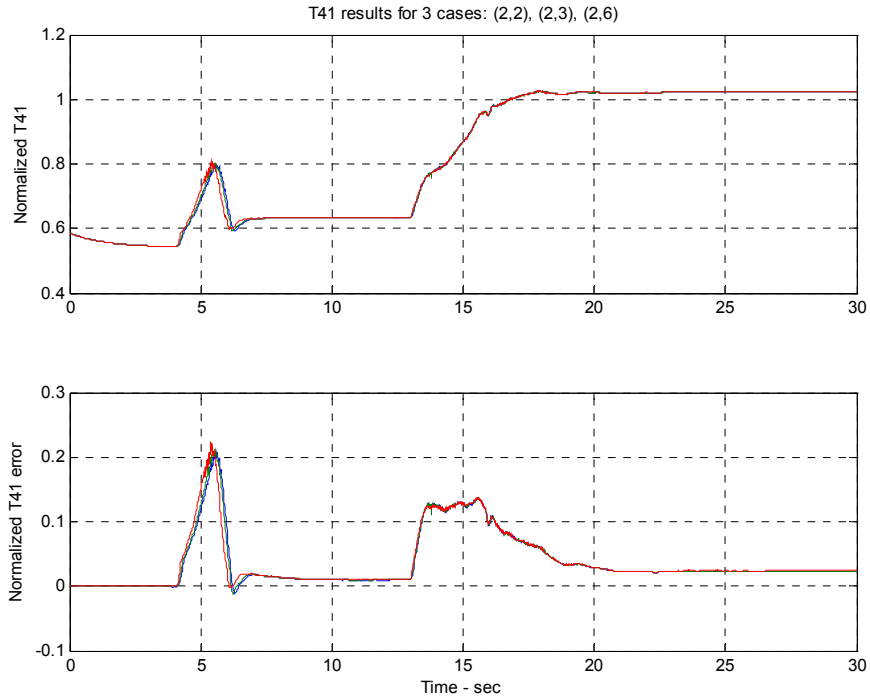

Figure 2.1.5. T41 results for cases with control horizon =2, various prediction horizons.

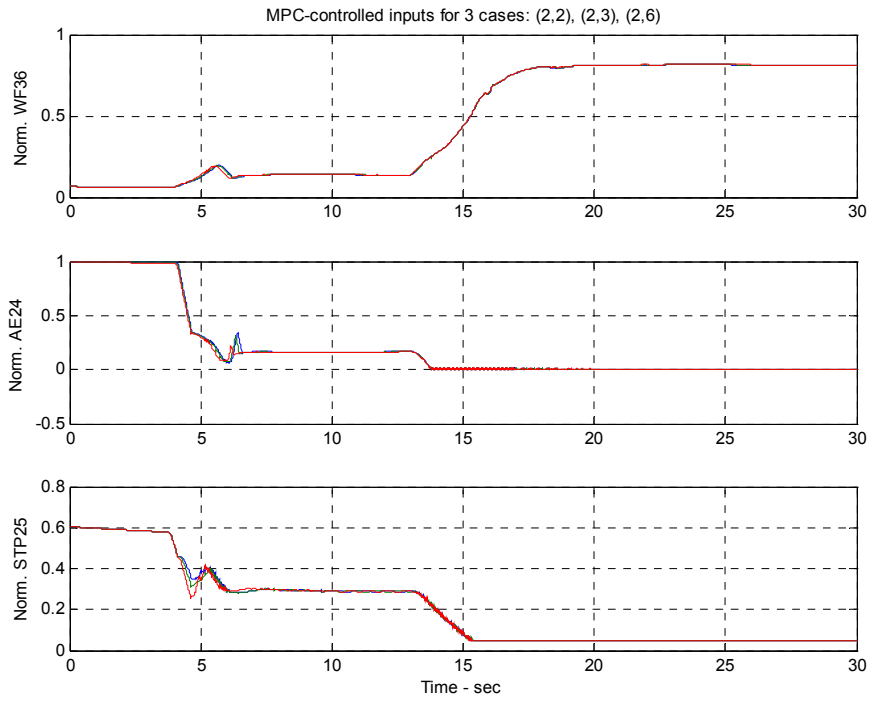

Figure 2.1.6. MPC inputs for cases with control horizon =2, various prediction horizons.

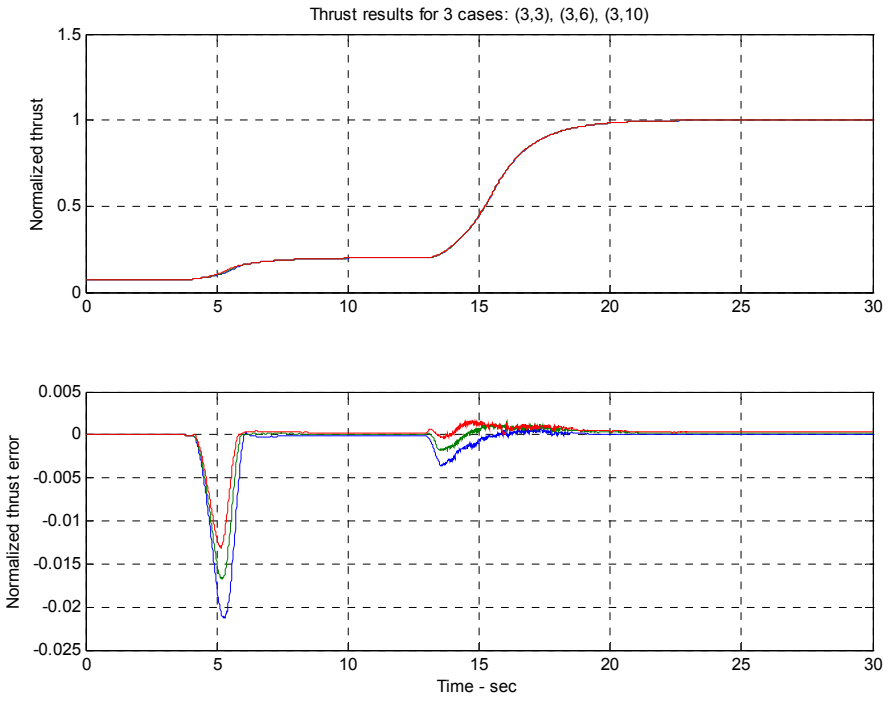

Figure 2.1.7. Thrust results for cases with control horizon =3, various prediction horizons.

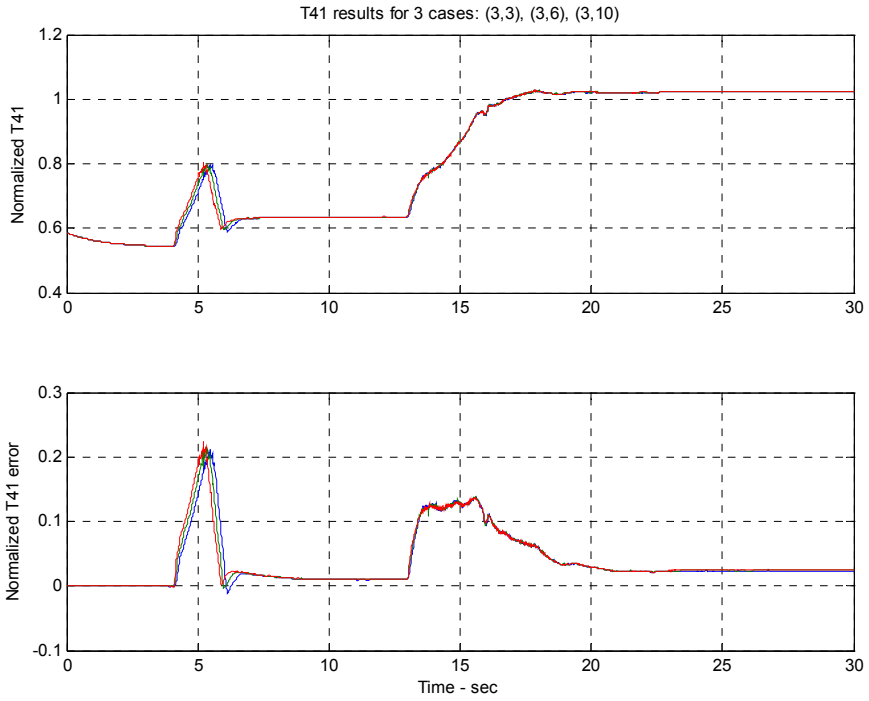

Figure 2.1.8. T41 results for cases with control horizon =3, various prediction horizons.

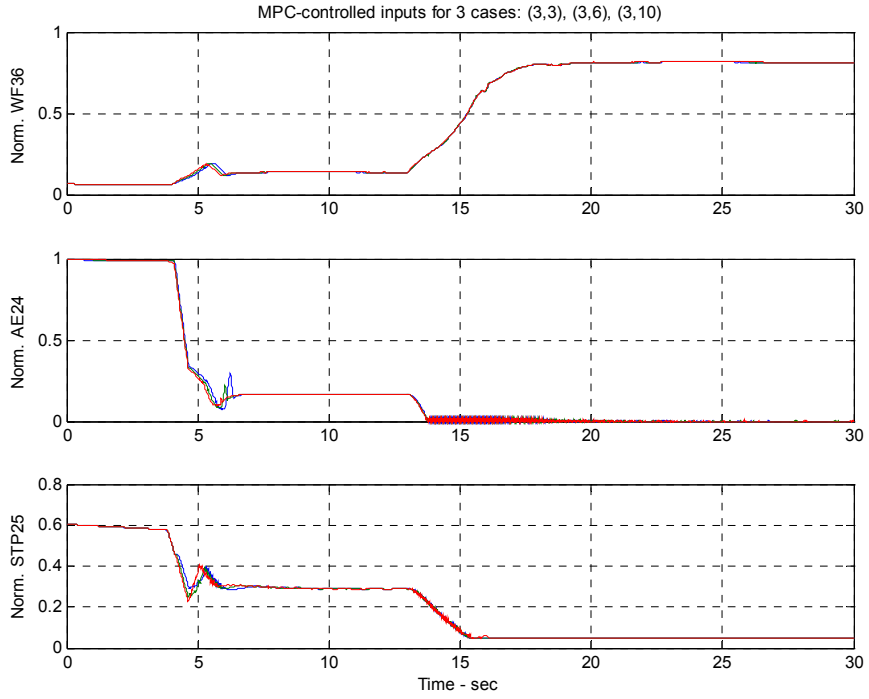

Figure 2.1.9. MPC inputs for cases with control horizon =3, various prediction horizons.

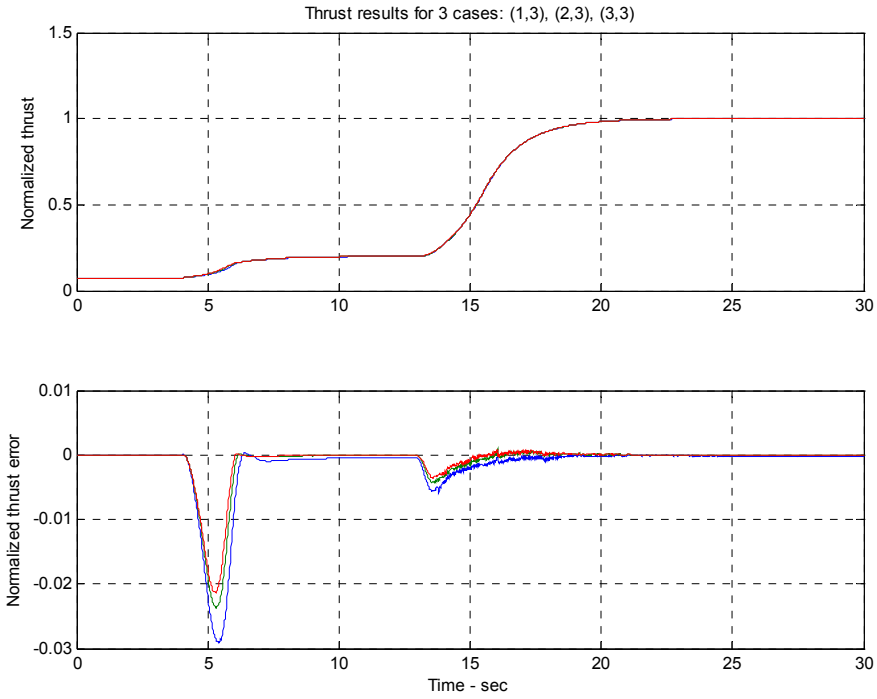

Figure 2.1.10. Thrust results for cases with prediction horizon =3, various control horizons.

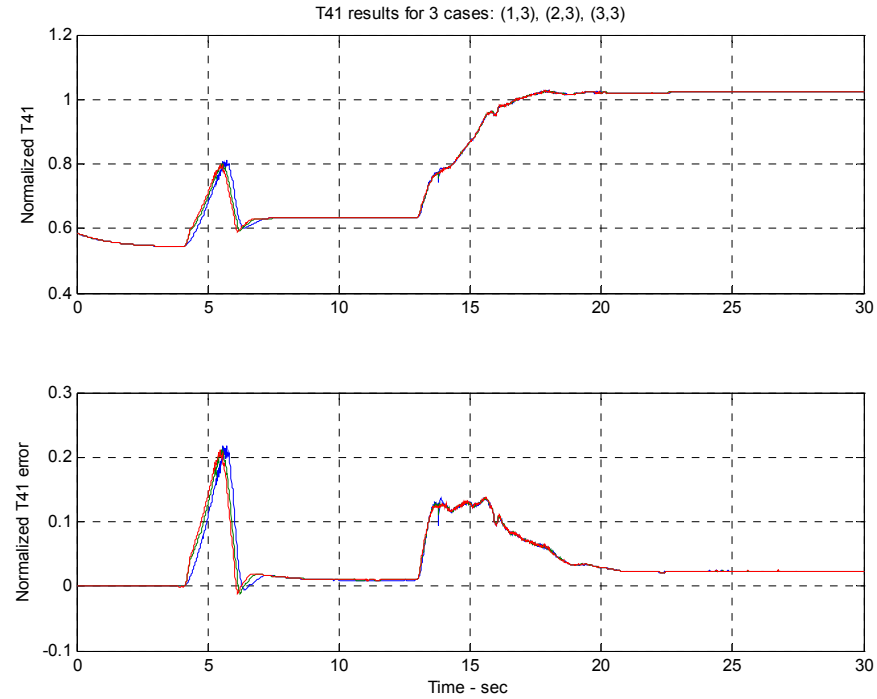

Figure 2.1.11. T41 results for cases with prediction horizon =3, various control horizons.

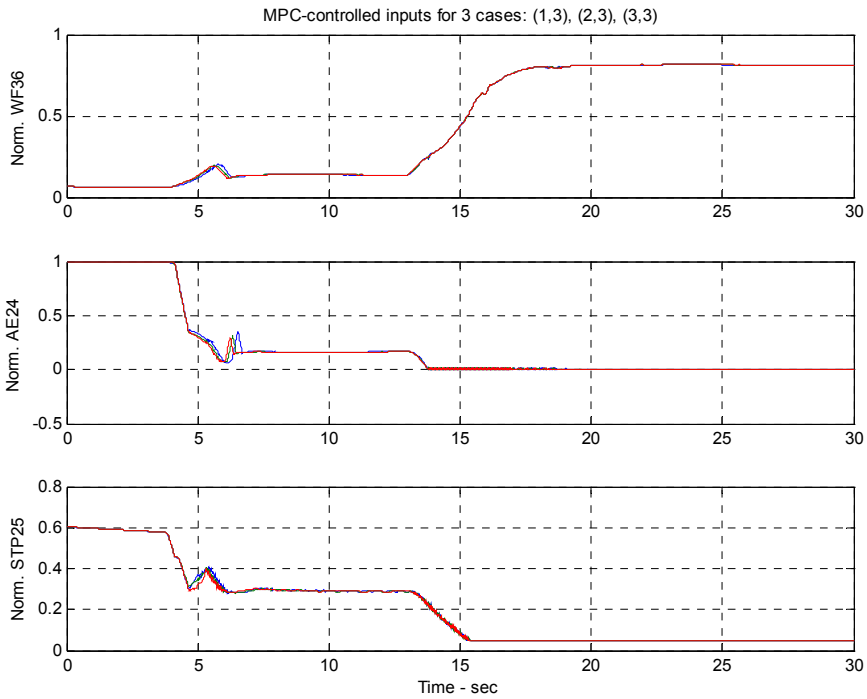

Figure 2.1.12. MPC inputs for cases with prediction horizon =3, various control horizons.

2.2. Effect of MPC Update Rate on Computation and Performance

MPC updates need not be re-calculated at every minor time frame of the controller implementation. One approach to saving on computation time for MPC is to re-optimize the control only every few minor frames, say every n minor frames where n is a small integer equal to at least 2. By doing the optimization less often than every minor frame, the computation required for the optimization is reduced.

When the approach of doing less frequent MPC updates is taken, a strategy must be in place for implementing the control during the minor frames intermediate to the MPC update times. For the simulation results shown below, we have implemented the first-order hold strategy described in Section 1.2.3 for the three MPC-optimized engine inputs. Under this strategy, when a new MPC update to the control is calculated, it is implemented in n equal increments over the next n minor frames, starting with the current frame, where each increment is 1/n times the total requested change in the control over the MPC update frame. Thus, each control is "ramped up" from its previous value to the newly requested optimal value over the MPC update frame. This allows the MPC optimization to call for a maximum total control update of n times the maximum input rate (subject to the overall magnitude limits on the input), and the "ramping up" of the input over n minor frames then avoids violation of the input rate limits. (Note: An alternative strategy is to allow MPC to call for whatever input changes it wants based on optimization, and then to let the physical actuator limits prevent violation of the input rates. While this might theoretically result in slightly faster responses to transient commands, it also defeats much of the

optimization effort because the MPC-calculated input changes are far different from the actual input changes during large transients. Therefore, this alternative was not pursued.)

The results below summarize the cases of n=1 (MPC update every minor frame), 2, 3, 5, 7, and 10. These values were chosen as a representative sample, and the results will show that n beyond 10 is probably not realistic due to performance issues that arise.

2.2.1 Computation time results

The first table below shows the Matlab-generated QP optimization computation CPU times (total in sec for identical 30-sec idle to takeoff power transient at SLS conditions) for various update rates and control horizons and various values of n (number of minor frames per MPC update cycle). As noted above, for MPC updates at a lower rate that every minor frame, the first-order hold implementation of the MPC inputs is used. All runs use a "true" engine model with physical actuator limits (magnitude and rate) enforced.

The minor frame for this simulation is 15 msec, so a 30-sec simulation run has 2000 minor frames. Thus, there are 2000 MPC updates for MPC updates every minor frame, 1000 for updating every 2 frames, 666 for MPC updates every 3 frames, 400 every 5 frames, 285 every 7 frames, and 200 every 10 frames.

The prediction horizons vary here (usually 1, 2, 3, 5, and 8 MPC update time frames), hence the computation time results are stated as ranges of values. However, the prediction horizon variations have only a small effect on the computation time, and the effect is not always monotonic, i.e. longer prediction horizons can sometimes produce shorter computation times.

Table 2.2.1.1. Effect of MPC update rate on CPU time for MPC optimization.

Control horizon	MPC updt every frame	MPC updt every 2 frames	MPC updt every 3 frames	MPC updt every 5 frames	MPC updt every 7 frames	MPC updt every 10 frames
1	16.09 – 18.38	8.42 – 9.39	5.703 – 6.672	3.594 – 4.125	2.563 – 2.828	1.813 – 2.109
2	20.92 – 22.63	10.69 – 11.84	7.094 – 8.156	4.5 – 5.016	3.281 – 3.688	2.172 – 2.875
3	27.41 – 30.91	13.45 – 14.56	8.859 – 9.953	5.469 – 6.031	4.016 – 4.359	2.859 – 3.203
5	42.19 – 44.67	20.06 – 21.06	13.70 – 14.25	8.609 – 8.719	6.203 – 6.234	4.219 – 4.266
8	-	34.36	22.5	14.53	10.91	7.781*

* - MPC algorithm did not converge on one MPC update (out of 200 updates)

We can alternatively express the results from the table above on an "optimization computation time per update" basis (in msec). The results stated in this fashion are in the next table:

Table 2.2.1.2. Effect of MPC update rate on CPU time per update for MPC optimization.

Control horizon	MPC updt every frame	MPC updt every 2 frames	MPC updt every 3 frames	MPC updt every 5 frames	MPC updt every 7 frames	MPC updt every 10 frames
1	8.0 – 9.2	8.4 – 9.4	8.6 – 10.0	9.0 – 10.3	9.0 – 9.9	9.1 – 10.5
2	10.5 – 12.0	10.7 – 11.8	10.6 – 12.2	11.3 – 12.5	11.5 – 12.9	10.9 – 14.4
3	13.7 – 15.5	13.5 – 14.6	13.3 – 14.9	13.7 – 15.1	14.1 – 15.3	14.3 – 16.0
5	21.1 – 22.3	20.1 – 21.1	20.5 – 21.4	21.5 – 21.8	21.8 – 21.9	21.1 – 21.3
8	-	34.4	33.7	36.3	38.3	38.9*

* - One MPC update (out of 200) did not converge.

From the preceding tables, we see that the computation time per MPC update is primarily affected by the control horizon. Each increment in the control horizon adds about 3 to 4 msec to the time required for the MPC optimization, and this increment grows slightly as the control horizon gets bigger. Hence, from a computational effort point of view, there is strong incentive to make the control horizon as short as possible, but only a small incentive for short prediction horizons.

2.2.2 Engine performance results

Essentially all of the cases produce reasonably good tracking of the reference thrust trajectory for this baseline case (idle-to-takeoff at sea level static conditions). The maximum thrust tracking errors for frequent MPC updating tend to occur when the TRA is first increased. For less frequent MPC updates (larger n values), the largest thrust tracking errors tend to occur during the second run-up in TRA.

As a representative set of results, we will examine here the case of control horizon = 1 and prediction horizon = 3 for the 6 MPC update rates considered (n = 1, 2, 3, 5, 7 and 10). Figure 2.2.2.1 shows the thrust trajectories for these 6 update rates. The figure shows both the thrust trajectories (top plot), and the thrust tracking error histories (bottom plot). As noted above, the smaller n values tend to produce the most thrust error during the initial run-up, while larger n values produce the largest errors during the long run-up to takeoff power.

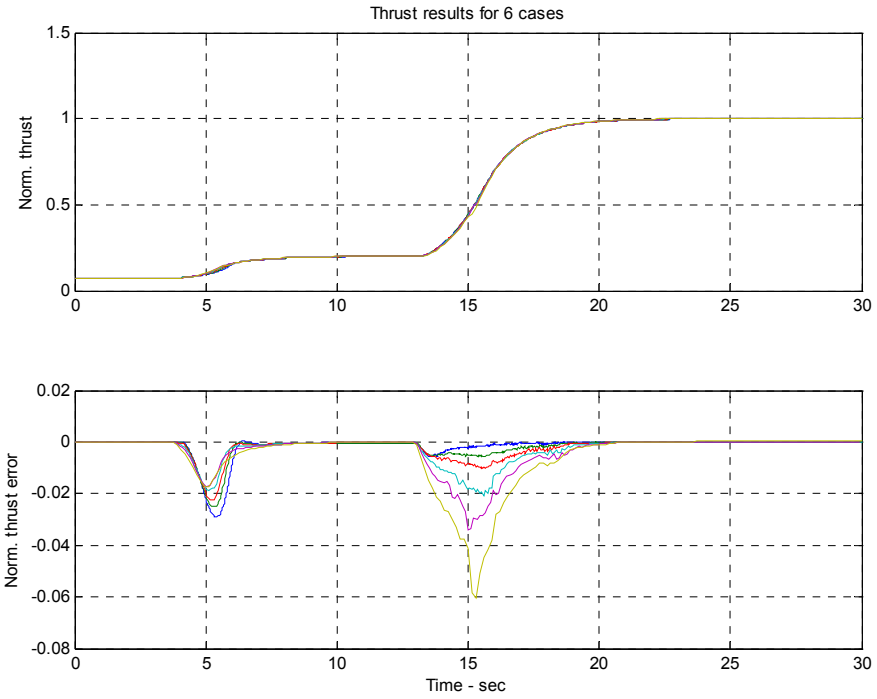

Figure 2.2.2.1. Thrust and thrust tracking error histories for (1,3) case, various MPC update rates

Figure 2.2.2.2 shows the T41 trajectories (on top, errors relative to reference on the bottom) for the same cases. Figure 2.2.2.3 shows the histories of the fan (top) and booster (bottom) stall margins. The normalized fan stall margin limit (1.00) is never approached. The normalized booster stall margin limit (1.00) is occasionally breached, and this will be examined in more detail below. Figure 2.2.2.4 shows the histories of the compressor stall margin (top) and the Stage 1 turbine blade tip clearance (bottom).

Violations of the booster stall margin occasionally occur at the beginning of both run-ups. This is also true also of violations of the compressor stall margin. We examine the results for these two stall margins in more detail below. No other limited variables display violations of their constraints for this transient at this flight condition.

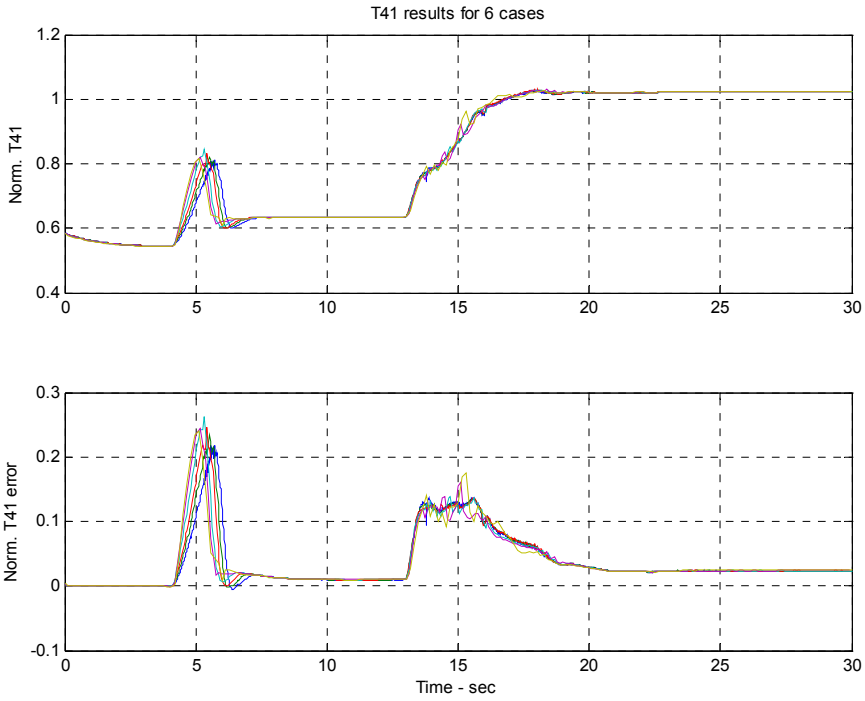

Figure 2.2.2.2. T41 and T41 error trajectories for (1,3) case, various MPC update rates

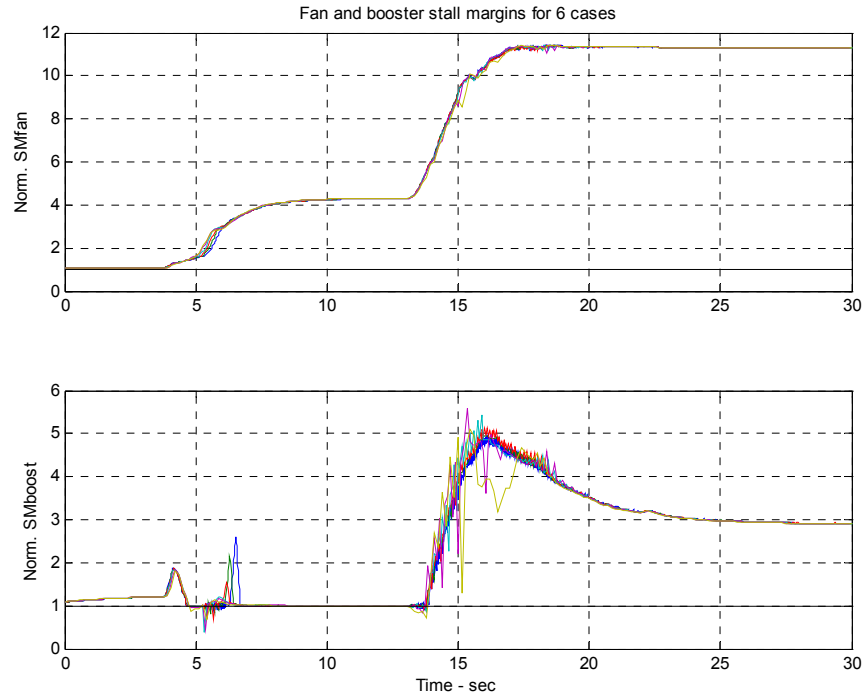

Figure 2.2.2.3. Fan and booster stall margins for (1,3) case, various MPC update rates.

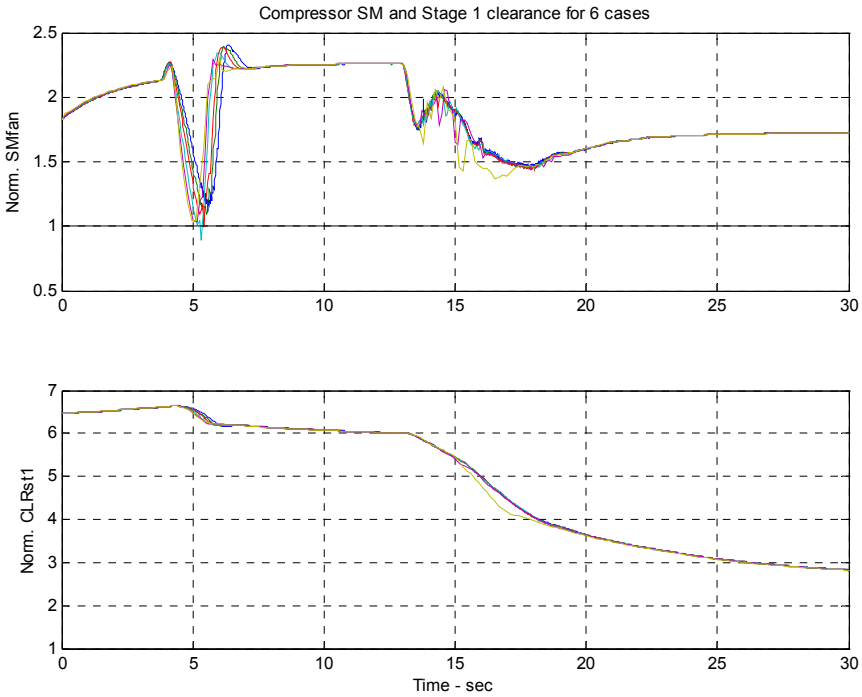

Figure 2.2.2.4. Compressor stall margin and Stage 1 clearance for (1,3) case, various MPC update rates.

The primary effect of the prediction horizon, and to some degree the control horizon, is on the booster stall margin violations. Longer prediction horizons (to a limit) help to avoid booster stall margin violations. The number of booster stall margin violations (out of 2000 minor frames in the simulation run) and the largest magnitude violation for each case are shown in the following Table 2.2.2.3.

Table 2.2.2.3. Number and normalized size of booster stall margin violations.

CH, PH	1 frame updt		2 frame updt		3 frame updt		5 frame updt		7 frame updt		10 frame updt	
	# Viol.	Size	# Viol.	Size	# Viol.	Size	# Viol.	Size	# Viol.	Size	# Viol.	Size
1,1	328	0.098	491	0.249	515	0.409	530	0.562	485	0.768	370	0.753
1,2	277	0.112	375	0.250	496	0.392	511	0.670	461	0.438	535	0.654
1,3	262	0.116	365	0.238	407	0.394	462	0.690	438	0.622	440	0.365
1,5			366	0.243	408	0.385	477	0.563	361	0.392	486	0.668
1,8			357	0.256	413	0.383	475	0.619	283	0.681	466	1.112
2,2	318	0.097	412	0.234	421	0.254	496	0.252	500	0.193	69	0.293
2,3	335	0.100	390	0.225	449	0.361	509	0.059	51	0.190	38	0.208
2,5*	309	0.133	453	0.238	479	0.341	34	0.067	6	0.087	29	0.180
2,8*	313	0.125	290	0.239	54	0.374	16	0.034	none	none	37	0.407
3,3	303	0.103	425	0.238	433	0.379	145	0.068	55	0.113	37	0.208
3,5*	284	0.106	436	0.215	165	0.150	26	0.071	2	0.013	28	0.182
3,8*	298	0.128	228	0.197	30	0.023	1	0.021	none	none	36	0.183
5,5*	313	0.122	409	0.208	460	0.058	27	0.050	20	0.086	33	1.053
5,8*	290	0.094	263	0.111	32	0.035	1	0.024	8	0.092	28	0.193
8,8	---	---	333	0.219	31	0.038	1	0.024	none	none	28	0.190

* - For MPC updates every frame, simulations were run with PH = 6 and 10, respectively, in place of 5 and 8.

Interestingly, it appears from the above results that booster stall margin violations are best avoided when the prediction horizon is a certain number of minor simulation frames. Consider that when MPC updates are calculated every n minor frames, if the prediction horizon is P update frames, then the prediction horizon is n*P minor frames. Notice in Table 2.2.2.3 that no booster stall margin violations occur for n*P = 56 for control horizons equal to 2, 3 or 5, and that only 1 booster stall margin violation occurs for n*P = 40 for control horizons of 3, 5 or 8, and only 2 or 6, respectively, for n*P = 35 for control horizons of 3 and 2. Relatively small numbers of violations also occur for n*P = 21, 24 and 25 for control horizons at least as large as 2. For n = 10 (last column of table), the "best" prediction horizon for 3 of the 5 control horizons tabulated is 5, corresponding to n*P = 50. Thus, there appears to be something "magical" about n*P in the range of about 30 to 60. This corresponds to a prediction time horizon of 450 to 900 msec (or 0.45 to 0.90 sec).

However, there is a price that is paid for less frequent MPC updates and longer prediction horizons. This price is violation of the compressor stall margin. Table 2.2.2.4 below has the same format as Table 2.2.2.3 above, but this table is for compressor stall margin violations:

Table 2.2.2.4. Number and normalized size of compressor stall margin violations.

CH, PH	1 frame updt		2 frame updt		3 frame updt		5 frame updt		7 frame updt		10 frame updt	
	# Viol.	Size	# Viol.	Size	# Viol.	Size	# Viol.	Size	# Viol.	Size	# Viol.	Size
1,1	none	none	none	0.0125	1	0.0015	5	0.1338	6	0.1770	11	0.1709
1,2	none	none	none	0.0126	none	none	8	0.1404	8	0.0691	5	0.0101
1,3	none	none	none	0.0120	2	0.0557	6	0.1331	none	none	none	none
1,5	---	---	none	0.0122	none	none	none	none	none	none	3	0.0357
1,8	---	---	none	0.0129	none	none	none	none	64	0.0109	39	0.0437
2,2	none	none	none	0.0118	none	none	none	none	none	none	29	0.0219
2,3	none	none	none	0.0113	none	none	none	none	1	0.0014	29	0.0220
2,5*	none	none	none	0.0120	none	none	none	none	8	0.0141	30	0.0221
2,8*	none	none	none	0.0120	none	none	none	none	14	0.0139	31	0.0221
3,3	none	none	none	0.0120	none	none	none	none	10	0.0132	29	0.0219
3,5*	none	none	none	0.0108	none	none	none	none	none	none	29	0.0220
3,8*	none	none	none	0.0099	none	none	none	none	7	0.0162	30	0.0220
5,5*	none	none	none	0.0105	none	none	none	none	none	none	29	0.0219
5,8*	none	none	none	0.0056	none	none	none	none	6	0.0143	29	0.0219
8,8	---	---	none	0.011	none	none	none	none	6	0.0145	29	0.0219

* - For MPC updates every frame, simulations were run with PH = 6 and 10, respectively, in place of 5 and 8.

We see from Table 2.2.2.4 that keeping n at 5 or less is best. Furthermore, longer prediction horizons tend to produce more compressor stall margin violations, and by larger maximum amounts until a limit is reached, as shown by the results for n=10. The variation with control horizon length varies. Interestingly, the n*P = 35 or 40 cases with control horizon 1, 2, 3 or 5 produce no compressor stall margin violations except one case (n=7, P=5, C=2). We saw above that for these cases, with the control horizon at 2 or more, produced only a few booster stall margin violations for this transient.

Based on the stall margin violation results then, the "optimal" MPC strategy for this baseline case seems to be to generate MPC updates no less frequently than every 5 to 8 minor frames, with the prediction horizon P set such that n*P is somewhere near 40, and with the control horizon set at a value of at least 2, but not more than a few frames less than P. Since the per-update computational load rises quickly with control horizon, smaller control horizons are preferred to longer ones when the choice is available.

3. Other Flight Points with All Constraints Active

We also examined the computational effort and performance characteristics for MPC for five other combinations of flight condition and transient thrust and T41 demands. There five combinations are:

1) Sea-level static (SLS), idle-to-takeoff (without the step-up of the baseline cases)
2) Sea-level static (SLS), takeoff-to-idle
3) 20,000 ft altitude, Mach 0.5, Bode transient (thrust up, then down)
4) 20,000 ft altitude, Mach 0.5, reverse Bode transient (thrust down, then up)
5) 35,000 ft altitude, Mach 0.84, reverse Bode transient (thrust down, then up)

As we have seen above for the baseline MPC case, one approach to reducing the computational effort for MPC is to calculate the optimal MPC control input updates less frequently than every simulation time step (or "minor frame"). To study the effects of less frequent MPC updates on the computational effort and performance of MPC for these five flight points, we considered the following six MPC update rates for each flight point:

MPC updates every minor frame (every 15 msec) – baseline case
MPC updates every 2 minor frames (every 30 msec)
MPC updates every 3 minor frames (every 45 msec)
MPC updates every 5 minor frames (every 75 msec)
MPC updates every 7 minor frames (every 105 msec)
MPC updates every 10 minor frames (every 150 msec)

For each MPC update rate, we considered 15 different combinations of the MPC control horizon and prediction horizon. In the format (control horizon, prediction horizon), the cases we examined are:

(1,1),	(1,2),	(1.3),	(1,5),	(1,8)
	(2,2),	(2,3),	(2,5),	(2,8)
		(3,3),	(3,5),	(3,8)
			(5,5),	(5,8)
				(8,8)

No cases where the control horizon is larger than the prediction horizon were considered. For MPC, it is unreasonable to optimize the inputs over a control horizon longer than that for which the output behavior is predicted by the approximate system model. Furthermore, as we have seen in the baseline cases above, and as we shall see in the results that follow, the length of the control horizon has considerably more effect on the computational effort required for MPC than does the prediction horizon length.

All cases use the same MPC cost function as the baseline MPC cases above. One aspect of the problem that does vary slightly from case to case is the set of minimum constraint values for the stall margins for the fan, booster, and compressor. See the earlier section on modified constraints for the constraint values used for each flight point.

Furthermore, all cases are run on the same deteriorated engine used for the baseline MPC cases above. Therefore, the performance values and computational effort are directly comparable.

3.1. Flight point 1: SLS, idle-to-takeoff

Figure 3.1.1 shows the thrust and T41 transients generated by a new engine using FADEC inputs for this flight point. These transients, filtered by the first-order low-pass filter with a time constant of 1.4 seconds that is used as the reference trajectory pre-filter, then become the reference trajectories for thrust and T41 for the MPC-controlled deteriorated engine to track. The filtered trajectories are shown in Figure 3.1.2, along with the unfiltered versions for comparison. The effect of the 1.4 sec delay in the reference trajectory pre-filter is apparent. As noted above, the MPC algorithm attempts to track these reference trajectories with minimal error for a deteriorated engine while also avoiding violation of the seven constraints listed above, including minimum values for the fan, booster, and compressor stall margins.

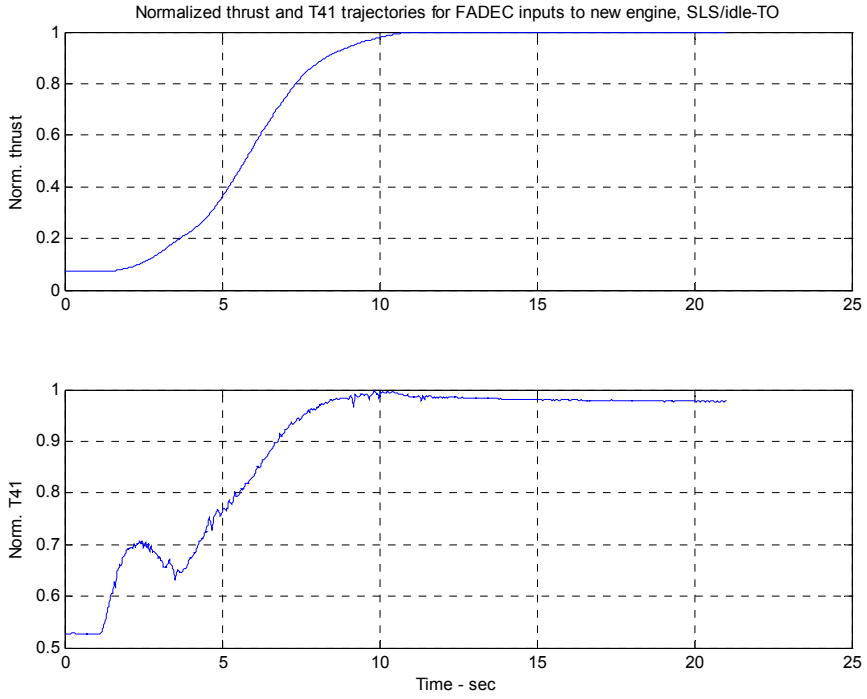

Figure 3.1.1. Thrust and T41 trajectories generated by FADEC inputs to new engine.

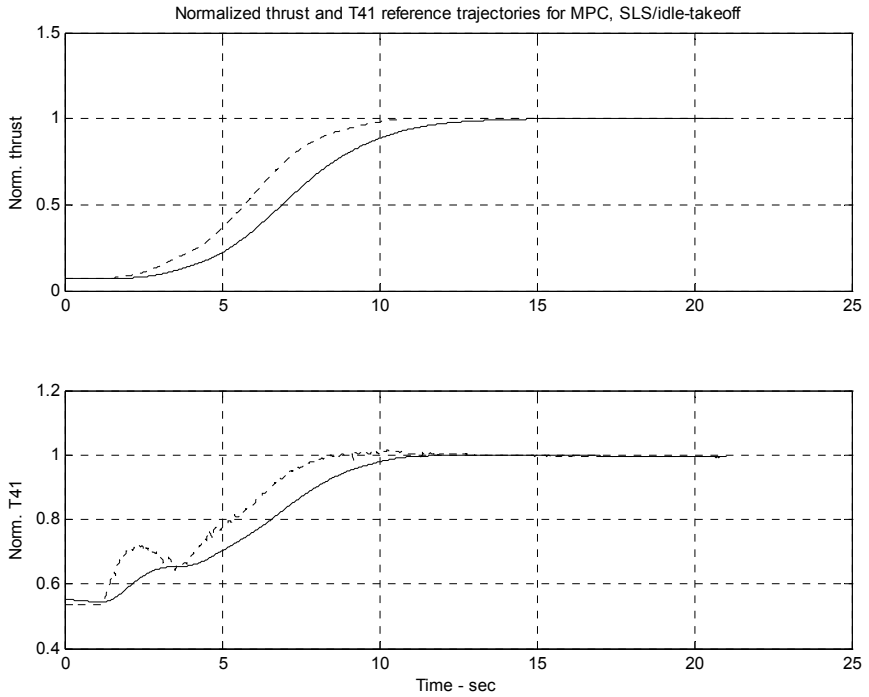

Figure 3.1.2. Reference trajectories for MPC (solid) compared to trajectories from FADEC control of a new engine (dotted) – SLS idle-to-takeoff case.

Table 3.1.1 below shows the normalized computational effort required for MPC optimization for a 21-second run of this SLS, idle-to-takeoff flight point with MPC updates computed at every minor frame. All CPU times are normalized by the CPU time required for the baseline case where MPC updates are performed every minor frame with the control horizon and prediction horizon both equal to 1. Hence, the normalized computational effort for this case in the table is exactly 1.000.

Table 3.1.1. Normalized computational effort for SLS, idle-to takeoff case, MPC updates every minor frame, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	1.000	1.157	1.174	1.123	1.200
2		1.369	1.398	1.400	1.437
3			1.650	1.736	1.768
5				2.461	2.610
8					4.286

We see from Table 3.1.1 that the control horizon has the largest effect on the MPC computational effort. The prediction horizon has only a small impact on computational effort. In particular for this case, each single time step increase in the control horizon adds about 20% to

25% to the computational effort required, while each single time step increase in the prediction horizon adds only a few percent to the computational effort.

Different combinations of the control horizon and prediction horizon lengths for MPC produce different results for the time histories of the engine input and output variables. To illustrate this, Figure 3.1.3 shows the MPC-controlled responses of thrust for 5 cases with the control horizon equal to 1 time step and the prediction horizon = 1, 2, 3, 5, and 8 time steps. The top plot shows the thrust results for all five cases vs. the reference trajectory for thrust. This shows that the thrust tracking is very good for all cases. To better illustrate the differences in thrust tracking, the bottom plot shows the thrust tracking error for all five prediction horizons. Now, slight differences among the cases become apparent. Generally speaking, the thrust tracking error is smallest for the longest prediction horizon. This pattern will become a trend as more results are presented below.

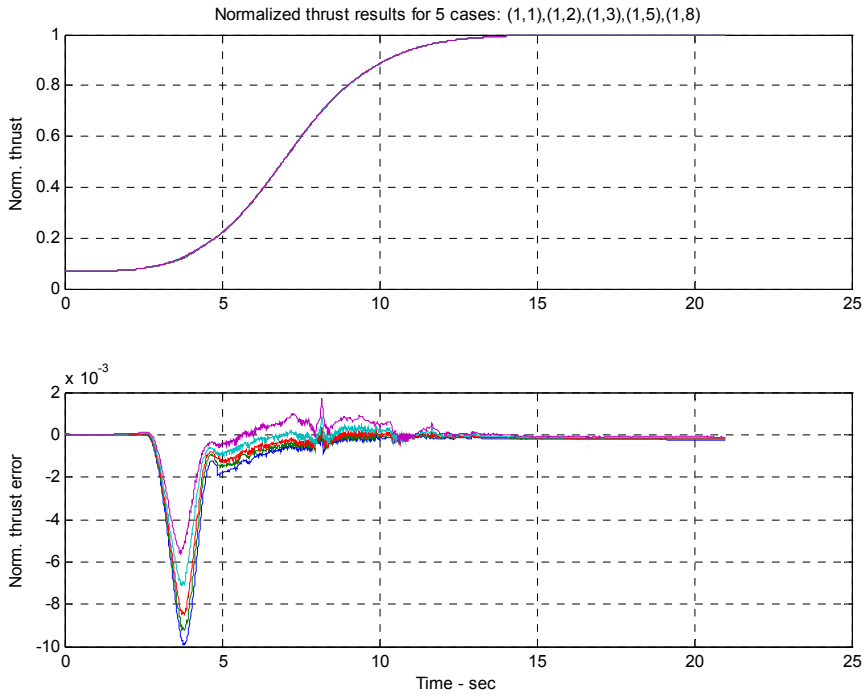

Figure 3.1.3. Thrust tracking results for MPC updates every minor frame, control horizon = 1, prediction horizon = 1, 2, 3, 5, and 8.

Figure 3.1.4 is the analogous figure for T41 tracking for these cases. We see that T41 for this simulated deteriorated engine is always higher than the reference trajectory generated by a new engine, as would be expected. The T41 tracking error relative to the reference trajectory is plotted in the bottom of the two plots shown in Figure 3.1.4. We can see that the prediction horizon has little impact on the size of this error.

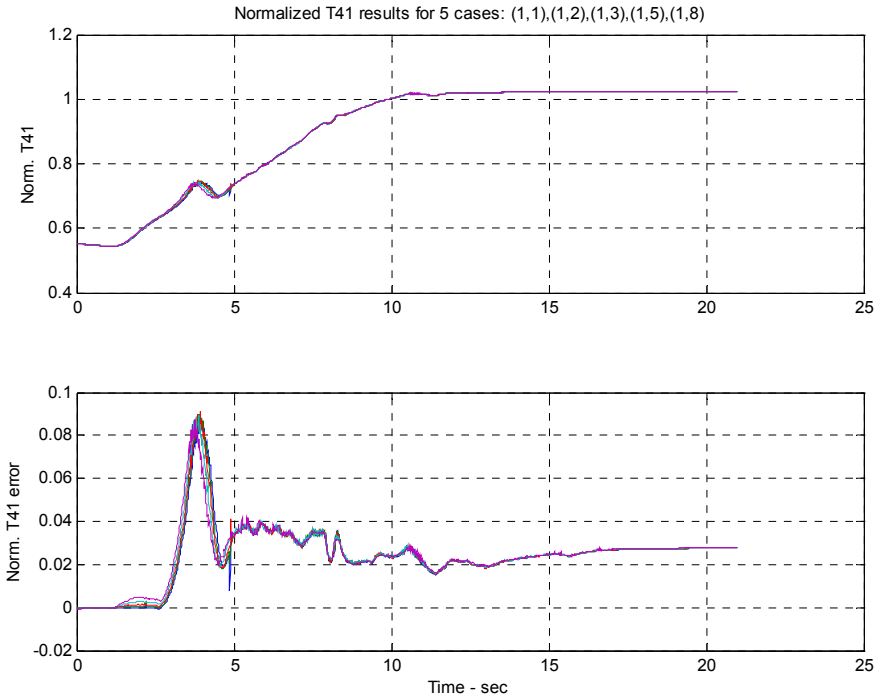

Figure 3.1.4. T41 tracking results for MPC updates every minor frame, control horizon = 1, prediction horizon = 1, 2, 3, 5, and 8. Reference T41 is also shown on the top plot.

The control provided by MPC to the deteriorated engine cannot avoid T41 values that are higher than the T41 reference trajectory. One of the reasons for this is that in computing MPC, a penalty is placed on violations of any "soft" constraints. For this flight point and this transient, the critical constraint is the booster stall margin. The five histories of the booster stall margin, and its constraint for this flight condition, are plotted in Figure 3.1.5. The top plot is the entire time history of the booster stall margin, while the bottom plot is an expanded view in the vicinity of time = 4 sec. We see that the booster stall margin "bounces around" its limit under MPC during the time period when the T41 tracking error is the largest. From the expanded view, we see that the limit is violated several times between time = 3.3 sec and time = 4.9 sec. Thus, MPC cannot control the deteriorated engine to exactly track both the thrust and T41 reference trajectories generated by a new engine without some constraint violations occurring. We also see that the prediction horizon length has little impact on these violations. For this particular flight point and transient, no other constraint violations occur under MPC.

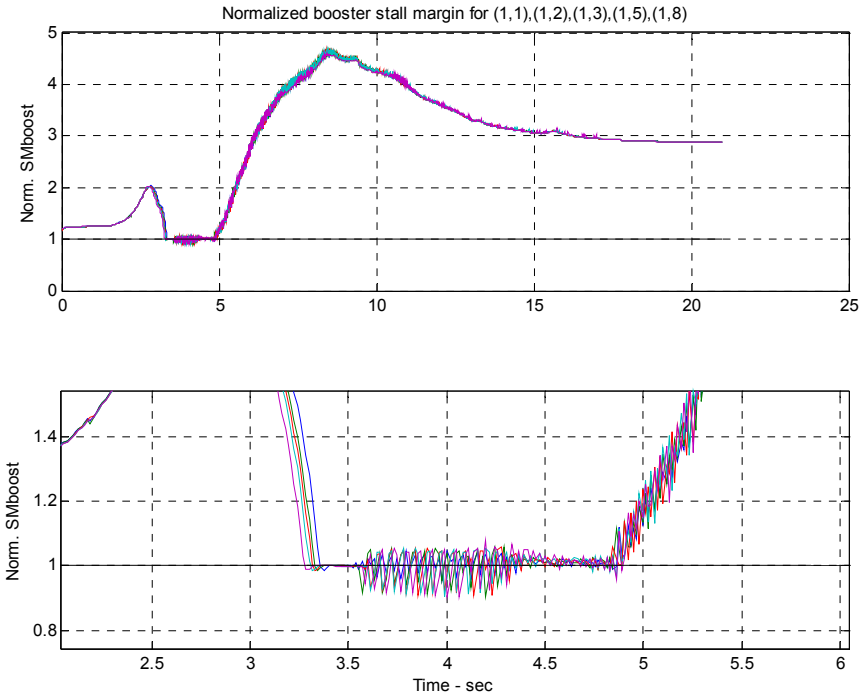

Figure 3.1.5. Booster stall margin vs. limit, MPC updates every minor frame, control horizon = 1 time step, prediction horizon = 1, 2, 3, 5, and 8 time steps.

Figures 3.1.6 through 3.1.8 show the results of varying the control horizon with fixed prediction horizon for this case. The thrust outputs for all five values of the control horizon track the thrust reference well, with the largest thrust tracking error for the shortest control horizon. The T41 tracking is also similar to the cases above for varying the prediction horizon. As above, the booster stall margin is violated during the period when the T41 tracking error is the largest as MPC struggles to track the reference trajectories without violating this constraint. No other constraints are violated during this transient.

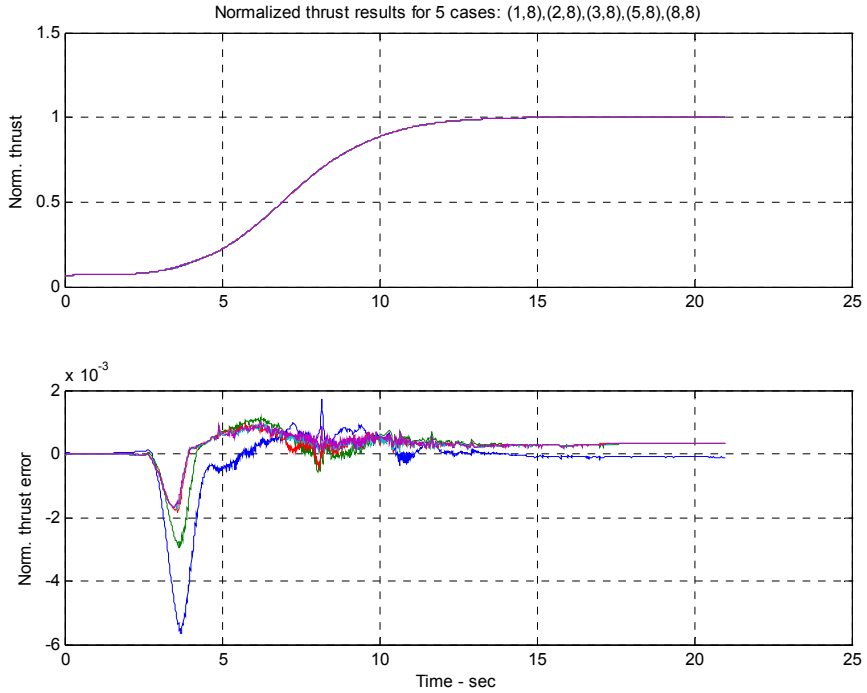

Figure 3.1.6. Thrust tracking results for varying control horizon (= 1, 2, 3, 5, 8) with fixed prediction horizon (= 8 time steps).

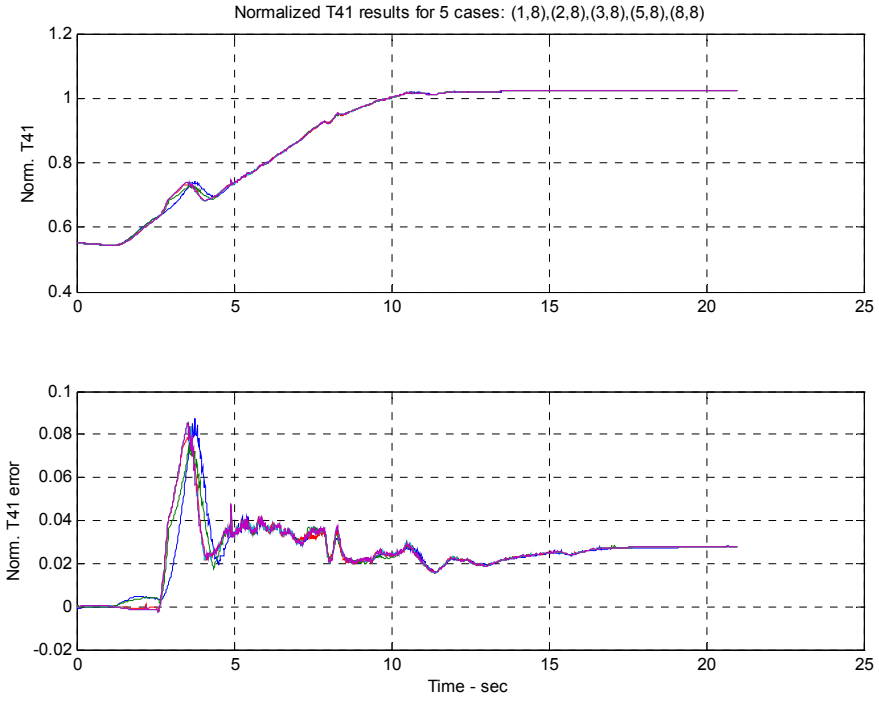

Figure 3.1.7. T41 tracking results for varying control horizon (= 1, 2, 3, 5, 8) with fixed prediction horizon (= 8 time steps).

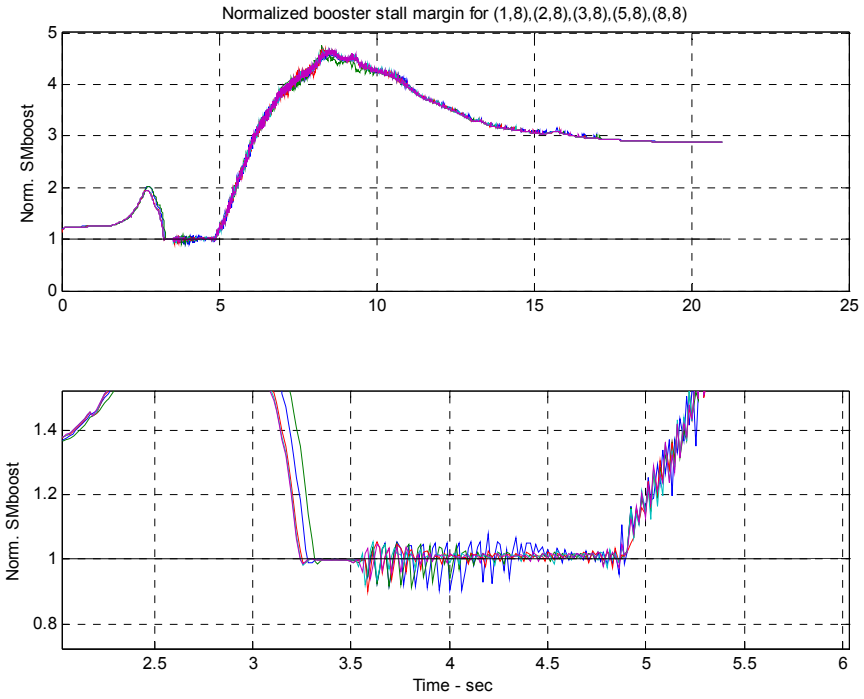

Figure 3.1.8. Booster stall margin results for MPC updates every minor frame, control horizon = 1,2,3,5,8, prediction horizon = 8.

Less frequent updates of MPC for Flight Point 1

One way to substantially reduce the computational burden of MPC is to calculate MPC updates less frequently than every minor frame. As noted in the earlier section on how the MPC algorithm was modified for this study, reducing the MPC update rate to less than every minor frame requires some alterations to the way in which the MPC inputs are optimized and applied. Because the MPC inputs are subject to limits on both their magnitudes and their rates of change, these limits must be taken into account properly in computing and implementing the MPC input updates. The baseline MPC algorithm properly accounted for these input constraints in computing the optimal MPC input updates by limiting the allowable change in each input to its allowable change per minor frame times the number of minor frames between MPC updates. However, the original implementation of the MPC updates was to introduce the entire computed change in input in a single minor simulation time frame. We have altered the MPC implementation, as discussed above, so that the optimal MPC input changes are now divided by the number of minor frames per MPC update, and the MPC input change is then "ramped in" by this amount per minor frame. This first-order hold implementation guarantees that the MPC update changes do not exceed the input rate limits.

If the MPC update rate is reduced to every 2 minor frames for the SLS, idle-to-takeoff transient, the total CPU times shown in Table 3.1.2 result, where each CPU time total is normalized by the

total CPU time required for the baseline run for this transient with MPC updates every minor frame and control and prediction horizons both equal to one MPC update time step.

Table 3.1.2. Normalized computational effort for SLS, idle-to takeoff case, MPC updates every 2 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.5523	0.5523	0.5617	0.5711	0.5737
2		0.6716	0.6877	0.7172	0.7480
3			0.7762	0.8378	0.8915
5				1.2145	1.3043
8					2.0349

Relative to making MPC updates every minor frame, we see a 40% to 60% reduction in the computational effort required, which is consistent with the 50% reduction in the number of times the MPC optimization must be performed. As before, we see an increase of only a few percent for each single step increase in the prediction horizon, but a more substantial 12% to 20% increase in effort for each single step increase in the control horizon.

Figures 3.1.7 and 3.1.8 illustrate the thrust and T41 tracking performance when MPC updates are computed and implemented every 2 minor frames for various prediction horizons with the control horizon fixed at 1 MPC update step.

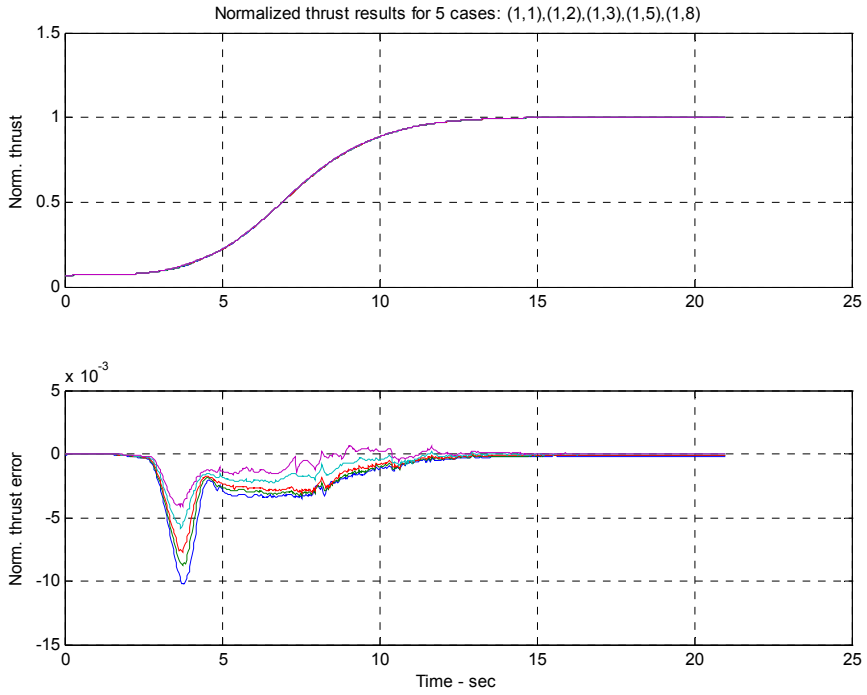

Figure 3.1.9. Thrust tracking results with MPC updates every 2 minor frames, control horizon = 1, prediction horizon varying (= 1, 2, 3, 5, 8). Thrust reference is also pictured in top plot.

We see that the thrust and T41 tracking errors are similar to those for the case with MPC updates every minor frame for this transient when we execute MPC updates only every 2 minor frames. In light of the significant reduction in computational effort, this suggests that less frequent MPC updating is reasonable for this flight point and transient.

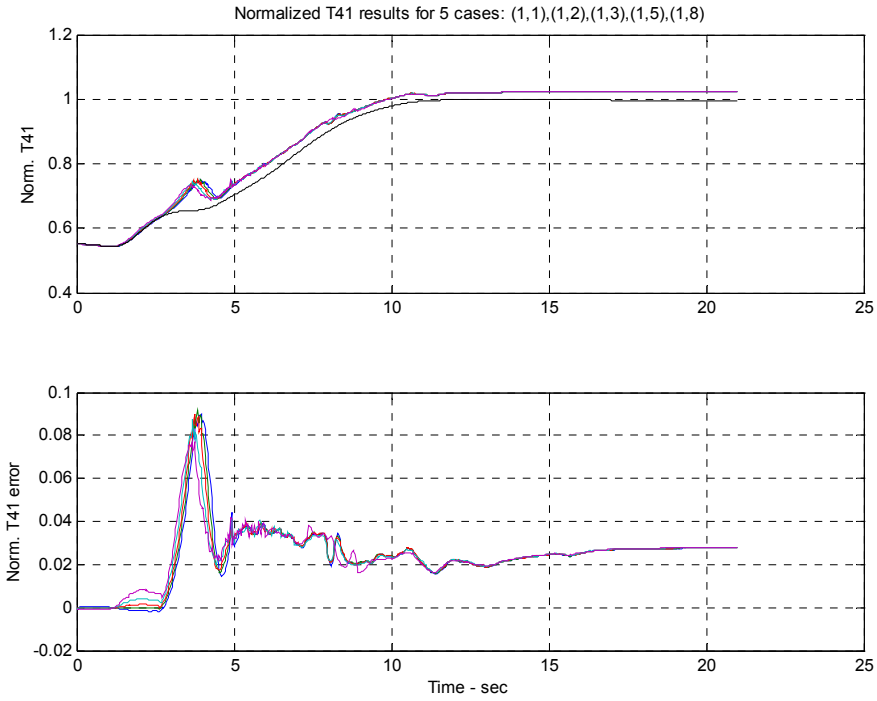

Figure 3.1.10. T41 tracking results with MPC updates every 2 minor frames, control horizon = 1, prediction horizon varying (= 1, 2, 3, 5, 8). T41 reference is also pictured in top plot.

Figure 3.1.9 shows the booster stall margin results for varying prediction horizon with the control horizon fixed at 1 MPC update time, along with the booster stall margin limit for this flight point. Much like the case when MPC updates are calculated every minor frame, the booster stall margin violations tend to occur when the T41 tracking error is largest during the acceleration. Compared to the results when doing MPC updates every cycle, we see that these violations are a bit larger in magnitude. In fact, the maximum booster stall margin violation is about twice as large for MPC updates every 2 minor frames as for MPC updates every minor frame.

Figure 3.1.10 and 3.1.11 are for MPC updates every 2 minor frames, now with fixed prediction horizon (8 time MPC update times) and varying control horizon (1, 2, 3, 5, and 8 MPC update times). Figure 3.1.10 shows that the thrust tracking is again very good for all control horizons. The smallest maximum thrust tracking error is for the longest prediction horizon, but only by a narrow margin. Figure 3.1.11 shows that the T41 tracking is similar for varying control horizons, and resembles the results for various prediction horizons. The T41 tracking results are also similar to those obtained with MPC updates every minor frame.

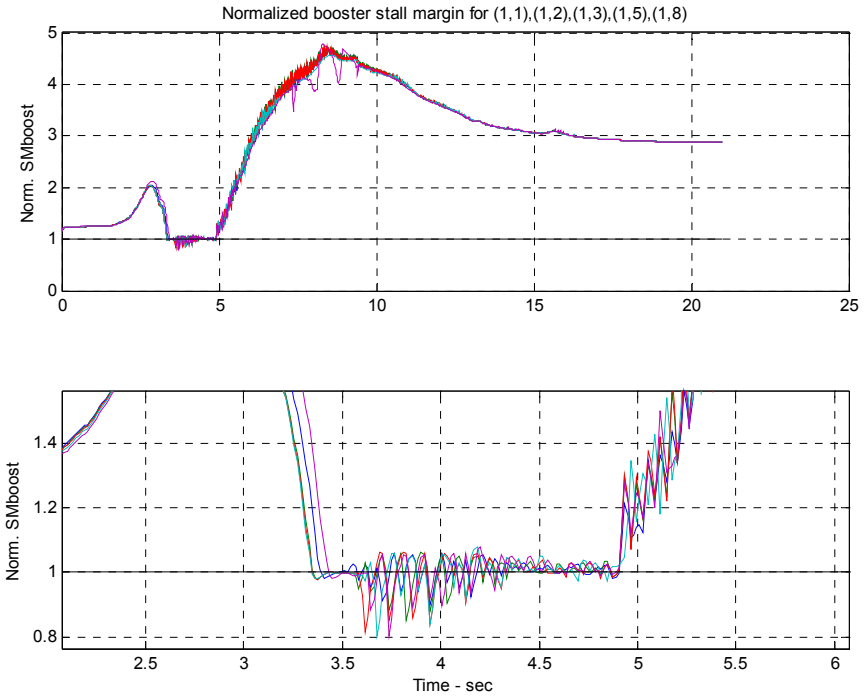

Figure 3.1.11. Booster stall margin vs. limit, MPC updates every 2 minor frames, control horizon = 1 MPC update step, prediction horizon = 1, 2, 3, 5, and 8 MPC update steps.

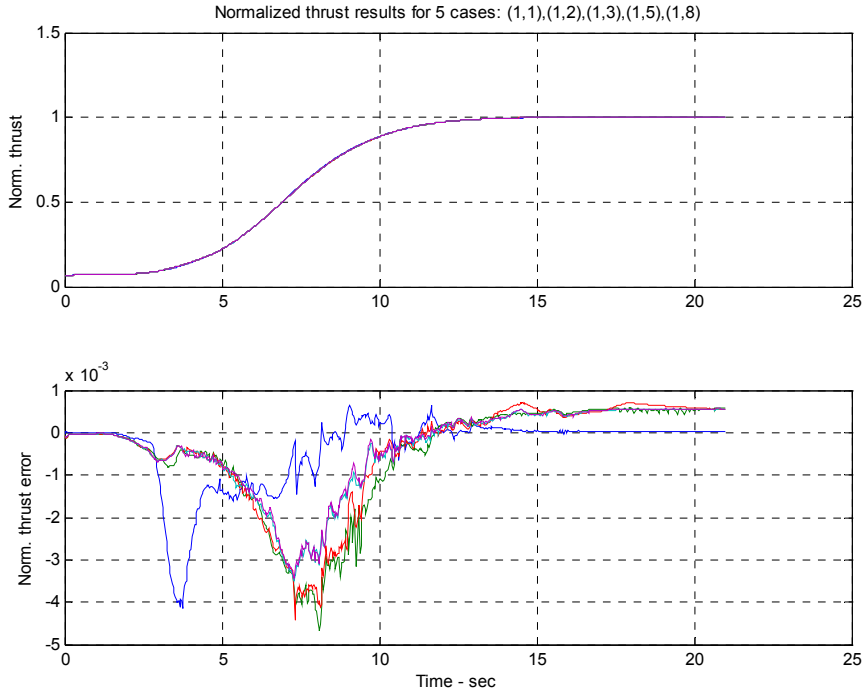

Figure 3.1.12. Thrust tracking results with MPC updates every 2 minor frames, prediction horizon = 8 MPC update steps, control horizon = 1, 2, 3, 5, and 8 MPC update steps. Thrust reference omitted (all cases track reference closely).

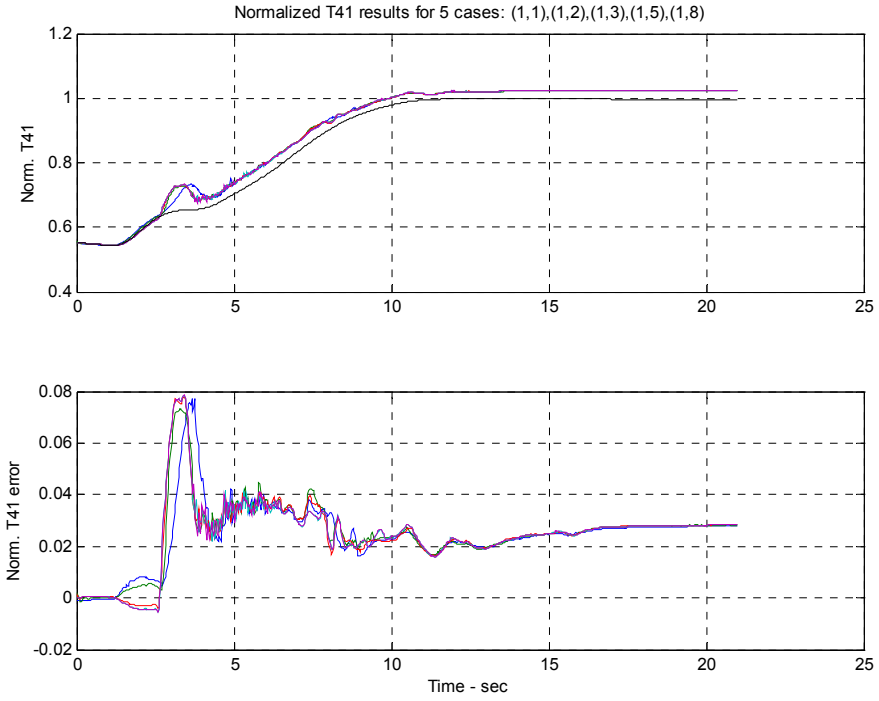

Figure 3.1.13. T41 tracking results with MPC updates every 2 minor frames, prediction horizon = 8, control horizon varying (= 1, 2, 3, 5, 8). T41 reference is also pictured in top plot.

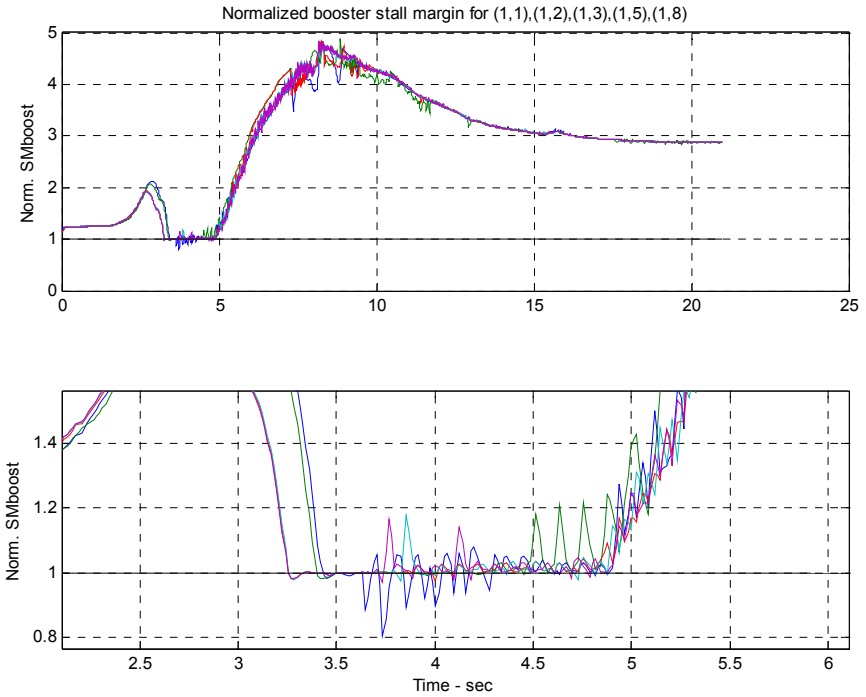

Figure 3.1.14. Booster stall margin vs. limit, MPC updates every 2 frames, control horizon = 1, 2, 3, 5, and 8 MPC update steps, prediction horizon = 8 MPC update steps.

The booster stall margin results for various control horizons are illustrated in Figure 3.1.12. This figure shows that the shortest control horizon produces slightly larger booster stall margin limit violations than the longer control horizons. However, the period over which booster stall margin violations occur is still just after the period when the T41 tracking errors are the largest.

We also examined implementing MPC with MPC updates every 3, 5, 7, and 10 minor frames. Tables 3.1.3 through 3.1.6 show the computational effort results for MPC optimization for each of these respective MPC update rates. Each table shows all the combinations of control and prediction horizon lengths considered above.

Table 3.1.3. Normalized computational effort for SLS, idle-to-takeoff case, MPC updates every 3 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.3727	0.3767	0.3834	0.3874	0.4035
2		0.4276	0.4585	0.4585	0.5000
3			0.5389	0.5617	0.6166
5				0.8405	0.8727
8					1.3674

Table 3.1.4. Normalized computational effort for SLS, idle-to-takeoff case, MPC updates every 5 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.2225	0.2265	0.2306	0.2319	0.2386
2		0.2829	0.3056	0.3190	0.3083
3			0.3418	0.3740	0.3740
5				0.5322	0.5429
8					0.8700

Table 3.1.5. Normalized computational effort for SLS, idle-to-takeoff case, MPC updates every 7 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.1649	0.1676	0.1649	0.1662	0.1796
2		0.2038	0.2145	0.2131	0.2145
3			0.2467	0.2333	0.2493
5				0.3646	0.3472
8					0.5871

Table 3.1.6. Normalized computational effort for SLS, idle-to-takeoff case, MPC updates every 10 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.1059	0.1019	0.1072	0.1005	0.1341
2		0.1622	0.1729	0.1676	0.1609
3			0.1850	0.1716	0.1783
5				0.2627	0.2547
8					0.4008

We see from the six tables (Tables 3.1.1 through 3.1.6) that the computational effort for MPC optimization is very nearly linear in the number of MPC updates per minor frame for this flight point and transient. We also see that the control horizon length has a substantial impact on the computational effort required, but that the prediction horizon length has almost no impact. In fact, for some MPC update rates, the longer prediction horizons actually require slightly less time for MPC optimization than do shorter prediction horizons. We will see later that these observations will hold for the other flight points and transients as well.

With the substantial reduction in computational effort that we obtain by reducing the rate at which the MPC updates are computed, the question of what rate to use then hinges on the performance sacrifices that we make for doing MPC updates less frequently. Essentially, for this flight condition and this transient, the performance sacrifices are in the accuracy of the thrust tracking, the maximum T41 value, and the number and size of the booster stall margin limit violations.

To illustrate these performance sacrifices, Figure 3.1.13 shows the comparison of the time histories of the thrust tracking error, T41, and the booster stall margin for the (1,8) MPC case for all 6 of the MPC update rates that we studied. The maximum thrust tracking grows larger as the update rate decreases. The maximum T41 also rises as the MPC updates become less frequent. In fact, the (1,8) case with MPC updates every 10 minor frames produced the largest peak T41 observed among the MPC simulations of this transient at this flight point. Finally, the booster stall margin limit violations become more numerous and larger as the MPC update rate decreases. Although not shown here, the maximum core speed is largest for the least frequent MPC updating, and the minimum compressor stall margin is smallest for the slowest MPC updating. Both remain within their specified limits for this flight condition, however.

Figure 3.1.14 shows the same comparison among the MPC update rates for MPC case (2,8). We observe performance properties that are similar to those in Figure 1.13 for the (1,8) case. However, the maximum thrust tracking errors are larger, while the booster stall margin violations are less severe for the slowest MPC update rates. The (2,8) case with MPC updates every 7 minor frames is the only case considered for this flight point that produced no booster stall margin violations during this transient. However, it also produced one of the larger maximum thrust tracking errors (as can be seen in Figure 3.1.14). The (2,8) case with MPC updates every 10 minor frames produced the largest maximum thrust tracking error (see Figure 3.1.14) and one of the largest peak T41 values.

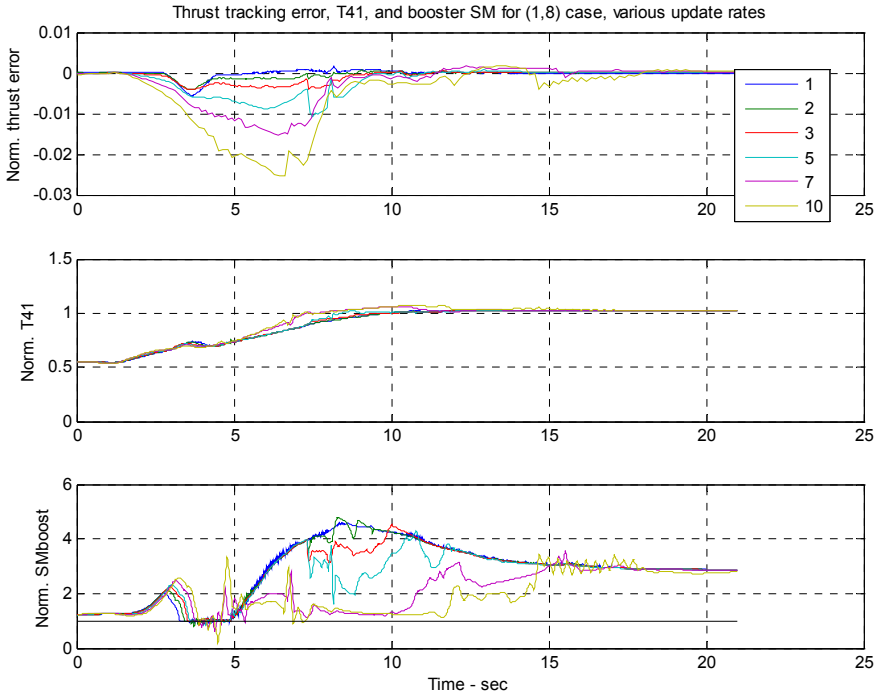

Figure 3.1.15. Thrust tracking error, T41, and booster stall margin for (1,8) MPC case for various update rates.

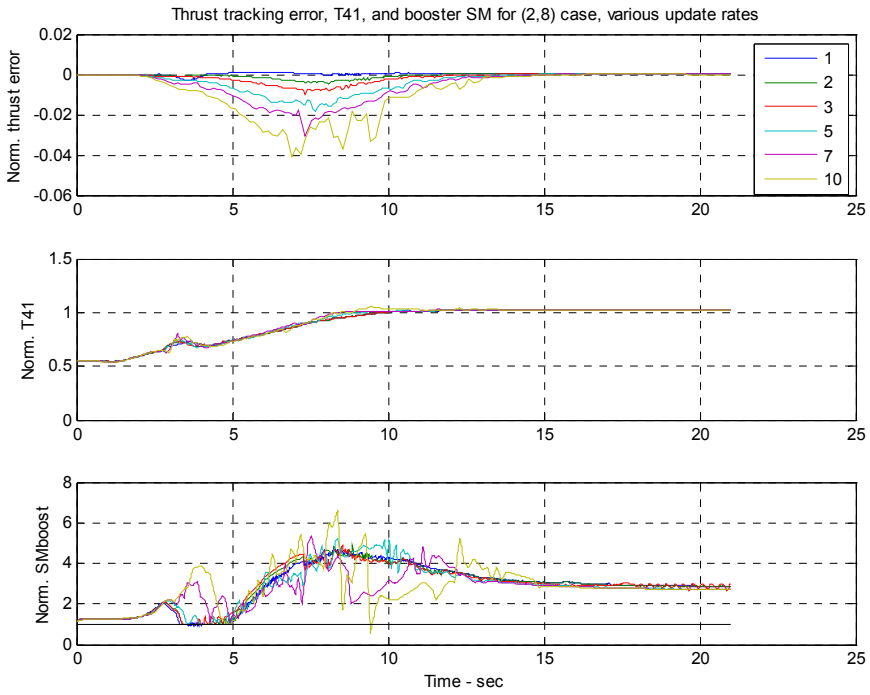

Figure 3.1.16. Thrust tracking error, T41, and booster stall margin for (1,8) MPC case for various MPC update rates.

To get a sense of the performance penalties associated with slower MPC updating, Table 3.1.7 shows the maximum thrust tracking error, peak T41, and largest violation of the booster stall margin limit for all of the combination of the MPC update rate, control horizon, and prediction horizon for this flight point and transient. The table shows that the thrust tracking error tends to be smallest for the fastest MPC update rate, usually with the largest prediction horizon for a given control horizon. The performance does not show a clear trend as a function of control horizon length, however. Sometimes, the shortest control horizon is the best, but sometimes this is not the case.

In light of the substantial computational savings for slower MPC updating and the lack of sensitivity of computational effort to the prediction horizon but strong sensitivity to the control horizon, these results suggest that the "best" MPC approach is to calculate MPC updates at the lowest rate for which the performance is tolerable using a relatively long prediction horizon, but the shortest possible control horizon.

3.2. Flight point 2: SLS, takeoff-to-idle

Figure 3.2.1 shows the thrust and T41 trajectories generated by a new engine under FADEC control, and the pre-filtered MPC reference trajectories, for a takeoff-to-idle power transient at SLS conditions.

For this transient, the MPC optimization algorithm was unable to find a feasible initial solution for the MPC-controlled inputs for MPC update rates more frequent than every 3 minor frames.

Even with MPC updates calculated only every 3 minor frames, most combinations of the control and prediction horizon lengths produce MPC results that do not track the reference thrust trajectory. It is not completely clear from the diagnostic messages produced by Matlab exactly why these difficulties occur for this transient at this flight point. However, we surmise based on the values of the variables before the non-convergence occurs that the MPC optimization algorithm cannot find a solution that simultaneously achieves the initial value of the reference thrust and avoids large violations of the Stage 1 clearance limits and unacceptably high T41 values (large enough that the model become invalid) when it is applied to the deteriorated engine running at takeoff power.

The only three combinations of CH and PH that produce accurate tracking of the thrust reference trajectory without significant early violations of the Stage 1 clearance limit for MPC updates every 3 minor frames are (1,3), (1,5), and (1,8). The histories of the thrust tracking error, T41, the booster stall margin, and the Stage 1 clearance (with its limit of 0.010) are shown in Figure 3.2.2 for these three cases. The Stage 1 clearance value "rides" its constraint before the decrease in power command occurs, after which the Stage 1 clearance is no longer an issue. After about 3.5 seconds of the transient response to the commanded change in power level, the booster stall margin begins to "ride" its constraint, and it stays there for the rest of the power transient.

Table 3.1.7. Normalized maximum thrust tracking error (lb), normalized peak T41 (deg R), and normalized maximum violation of booster stall margin limit for various MPC update rates, control horizons (CH) and prediction horizons (PH)

MPC updates every minor frame															
PH→ ↓CH	1			2			3			5			8		
1	.01	.0017	0.251	0.0093	.0017	0.515	.0085	.0017	0.510	.0072	.0017	0.518	.0057	.0017	0.523
2				.0062	.0017	0.352	.0054	.0017	0.443	.0042	.0017	0.596	.003	.0017	0.482
3							.0046	.0017	0.405	.0031	.0017	0.492	.0019	.0017	0.551
5										.0032	.0017	0.399	.0017	.0017	0.432
8													.0017	.0017	0.327
MPC updates every 2 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.0102	.0017	0.544	.0088	.0017	0.812	.0078	.0017	1.010	.0059	.0017	1.071	.0042	.0017	1.077
2				.0041	.0017	0.336	.0031	.0017	0.435	.0038	.0017	0.376	.0047	.0017	0.097
3							.0026	.0017	0.382	.0032	.0017	0.229	.0045	.0017	0.135
5										.0028	.0017	0.194	.0034	.0017	0.137
8													.0035	.0017	0.170
MPC updates every 3 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.0105	.0017	0.723	.0086	.0017	1.145	.0073	.0017	1.254	.0051	.0017	0.642	.0041	.0017	0.815
2				.0059	.0017	0.181	.0058	.0017	0.166	.0073	.0017	0.114	.0097	.0017	0.209
3							.0057	.0017	0.409	.0069	.0017	0.179	.0084	.0017	0.112
5										.0065	.0017	0.287	.0068	.0017	0.140
8													.0068	.0017	0.323
MPC updates every 5 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.013	.0017	0.986	.012	.0017	1.069	.011	.0017	0.579	.014	.0017	0.319	.01	.0017	0.796
2				.012	.0017	0.353	.013	.0017	0.243	.014	.0017	0.291	.019	.0017	0.197
3							.013	.0017	0.215	.013	.0017	0.291	.013	.0017	0.212
5										.013	.0017	0.238	.013	.0017	0.284
8													.014	.0017	0.287
MPC updates every 7 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.02	.0017	1.813	.02	.0017	0.665	.02	.0017	0.420	.024	.0017	0.977	.015	.0017	2.147
2				.019	.0017	0.468	.02	.0017	0.420	.02	.0017	0.378	.03	.0017	0.
3							.02	.0017	0.425	.02	.0017	0.354	.019	.0017	0.410
5										.02	.0017	0.388	.019	.0017	0.403
8													.02	.0017	0.315
MPC updates every 10 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.03	.0017	1.829	.03	.0017	1.081	.03	.0017	0.655	.03	.0017	1.796	.03	.0017	4.449
2				.03	.0017	0.560	.03	.0017	1.020	.03	.0017	0.597	.04	.0017	2.338
3							.03	.0017	0.951	.03	.0017	0.619	.03	.0017	0.358
5										.03	.0017	0.691	.03	.0017	0.656
8													.03	.0017	1.397

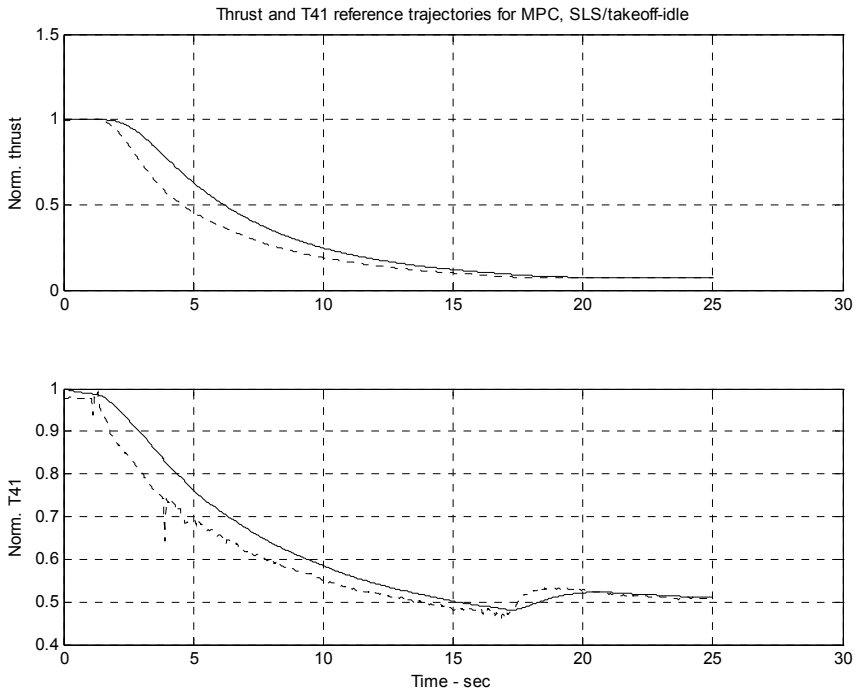

Figure 3.2.1. Pre-filtered thrust and T41 reference trajectories (solid) for MPC for SLS, takeoff-idle transient, compared to thrust and T41 from FADEC-controlled new engine (dotted).

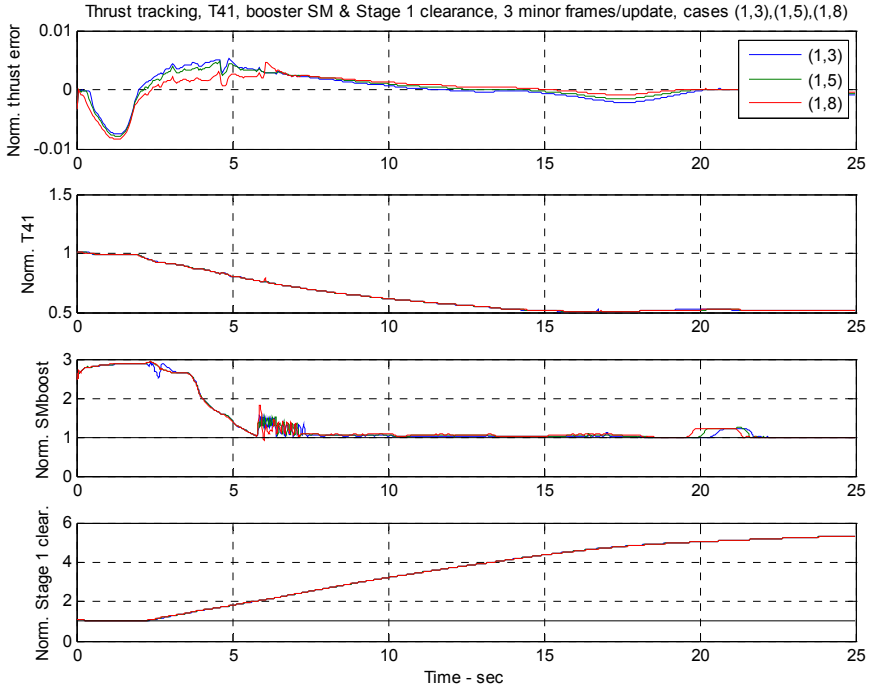

Figure 3.2.2. Thrust tracking, T41, booster stall margin, and Stage 1 clearance results for MPC updates every 3 minor frames, MPC cases (1,3), (1,5), (1,8).

For all of the MPC update rates slower than every 3 minor frames that we considered here, the thrust tracks the reference thrust trajectory well for all control horizon and prediction horizon combinations that we considered. All cases do produce booster stall margin limit violations or early Stage 1 clearance limit violations, or, in most cases, both. However, these violations are relatively small.

Figure 3.2.3 shows the thrust tracking, T41, booster stall margin, and Stage 1 clearance results for MPC case (1,8) for MPC updates every 3, 5, 7 and 10 minor frames. As with the idle-to-takeoff transient, less frequent MPC updating produces larger maximum thrust tracking errors. In this case, the booster stall margin violations tend to be fewer and less severe with less frequent updating, but this is not uniformly the case.

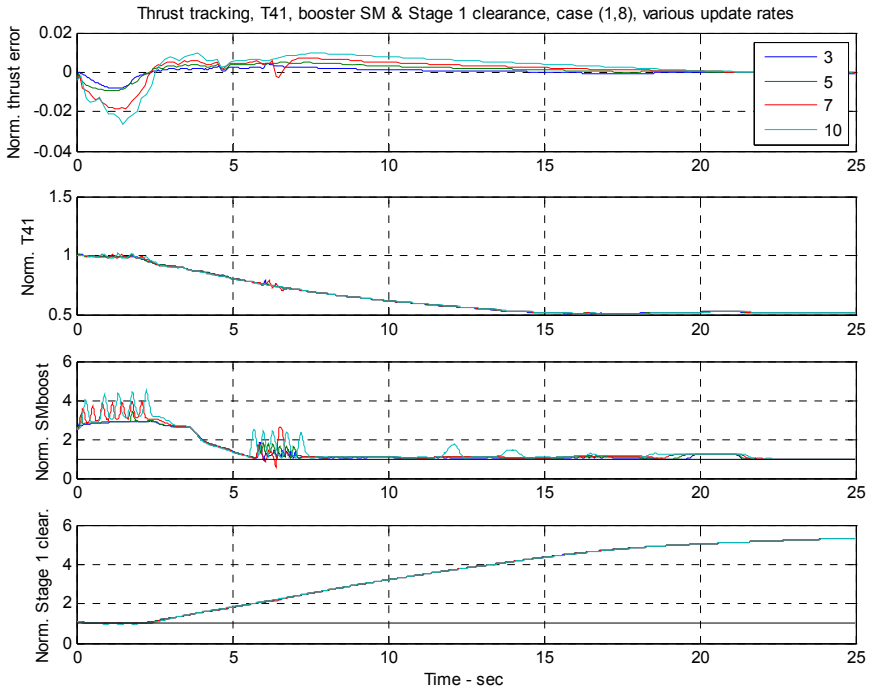

Figure 3.2.3. Thrust tracking, T41, booster stall margin, and Stage 1 clearance for (1,8) case of MPC with updates every 3, 5, 7, and 10 minor frames.

The violation of the Stage 1 clearance limit is not shown in the table because the largest such violation is still extremely small.

The results here mimic in many ways the results in the preceding sections for the SLS idle-takeoff transients, except that the longer prediction horizons do not necessarily produce better performance in this case. However, we see that the best performance occurs for the shortest control horizon in every case.

Tables 3.2.2 through 3.2.5 present the normalized computational effort for MPC optimization for all of the cases considered at this flight point where the reference thrust is adequately tracked. Because the baseline case of MPC updates every minor frame with CH and PH both equal to one MPC time step did not produce a solution for this flight point, we have normalized the computational effort here by the computational time for optimization for the (1,3) case with MPC updates every 3 minor frames. This means that the normalized computational effort here is not comparable to the other flight points. However, the results in Tables 3.2.2 through 3.2.5 display the same trends in computational effort that we observe for the other flight points. Therefore, the results for this flight point are consistent.

Table 3.2.2. Normalized computational effort for SLS, takeoff-to-idle case, MPC updates every 3 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	*	*	1.0000	1.0353	1.1000
2		*	*	1.2265	*
3			*	1.5294	*
5				2.1412	*
8					3.7000

* - Cannot track thrust reference

Table 3.2.3. Normalized computational effort for SLS, takeoff-to-idle case, MPC updates every 5 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.6029	0.6147	0.6294	0.6382	0.6765
2		0.7206	0.7971	0.7882	0.8353
3			0.9294	0.9471	1.0147
5				1.3765	1.4794
8					2.2206

Table 3.2.4. Normalized computational effort for SLS, takeoff-to-idle case, MPC updates every 7 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.4441	0.4235	0.4441	0.4706	0.4735
2		0.5206	0.5294	0.5824	0.6059
3			0.6206	0.6588	0.6647
5				0.9647	1.1206
8					1.4971

Table 3.2.5. Normalized computational effort for SLS, takeoff-to-idle case, MPC updates every 10 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.2588	0.2294	0.2206	0.2677	0.3000
2		0.4000	0.4441	0.4677	0.4235
3			0.4853	0.5500	0.5471
5				0.7088	0.8853
8					1.1559

3.3. Flight point 3: 20,000 ft, Mach 0.5, Bode transient

Figure 3.3.1 shows the thrust and T41 trajectories generated by a new engine under FADEC control, and the pre-filtered reference trajectories for a Bode power transient at a flight condition of 20,000 feet altitude, Mach 0.5.

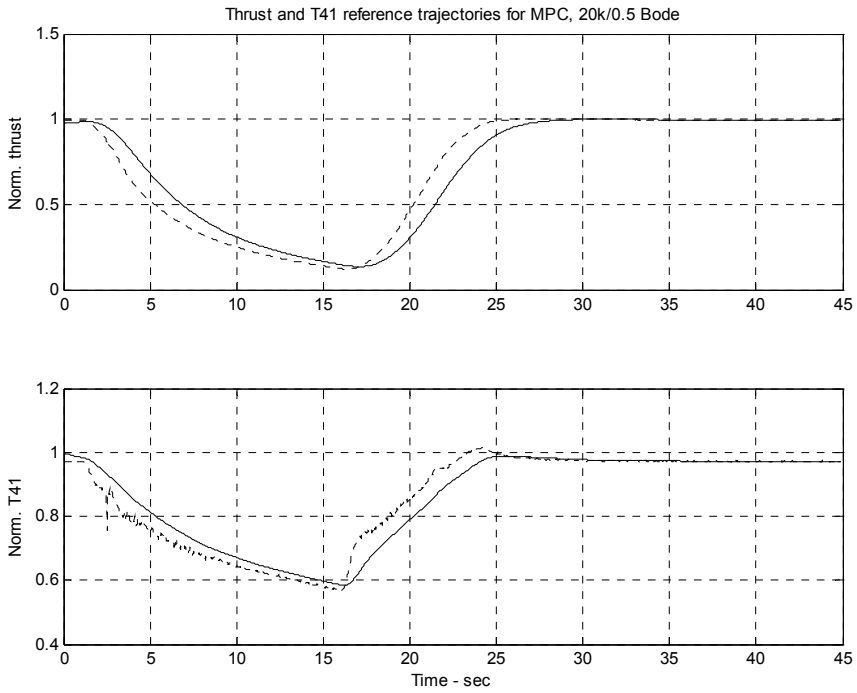

Figure 3.3.1. MPC reference trajectories (solid) for thrust and T41 for 20k/Mach 0.5 Bode transient, compared to results from FADEC at this condition (dotted).

Figure 3.3.1 shows the reference trajectories for thrust and T41 to be tracked by MPC for this condition. These are generated by simulating a FADEC-controlled new engine at this condition, and then pre-filtering the thrust and T41 outputs with a first-order low-pass filter with a time constant of 1.4 sec. The effect of the filtering is apparent on the plots.

For this transient, the MPC optimization algorithm is unable to find a feasible initial solution for the MPC-controlled inputs for update rates more frequent than every 2 minor frames. In this regard, the results here are similar to those for the takeoff-to-idle transient at SLS conditions.

Tables 3.33.1 to 3.3.5 show the normalized computation times required for MPC optimization for this flight point for each of the MPC update rates, respectively, that we considered. Because the usual baseline case of MPC updates every minor frame with control and prediction horizon both equal to one MPC update step could not be run at this flight point, the computation times are normalized by the total CPU time required for the (1,1) case with MPC updates every 2 minor frames. This means that the normalized computation times cannot be directly compared to those at the other flight points. However, the trends in computation effort that we observe here are similar to those we observed at the other flight points. Namely, the computational effort goes down linearly as the MPC update rate decreases, and it increases about 25% for each MPC cycle increase in the control horizon. Prediction horizon again has only a small effect on computational effort.

Table 3.3.1. Normalized computational effort for 20,000 ft/Mach 0.5, Bode case, MPC updates every 2 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	1.0000	1.0139	1.0256	1.0489	1.1211
2		1.2072	1.2491	1.2957	1.3748
3			1.4563	1.5820	1.6635
5				2.3201	2.4435
8					4.0407

Table 3.3.2. Normalized computational effort for 20,000 ft/Mach 0.5, Bode case, MPC updates every 3 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.6705	0.6705	0.6892	0.7136	0.7346
2		0.7777	0.8137	0.8603	0.8417
3			1.0139	1.0302	1.1083
5				1.5762	1.5937
8					2.5716

Table 3.3.3. Normalized computational effort for 20,000 ft/Mach 0.5, Bode case, MPC updates every 5 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.4051	0.4214	0.4191	0.4377	0.4796
2		0.5215	0.5460	0.5332	0.5693
3			0.6170	0.6426	0.7229
5				0.8999	1.0687
8					1.6822

Table 3.3.4. Normalized computational effort for 20,000 ft/Mach 0.5, Bode case, MPC updates every 7 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.2782	0.2643	0.2945	0.3097	0.3201
2		0.3586	0.3690	0.3946	0.4040
3			0.4366	0.4633	0.4563
5				0.6647	0.7159
8					1.1828

Table 3.3.5. Normalized computational effort for 20,000 ft/Mach 0.5, Bode case, MPC updates every 10 minor frames, varying control horizon (CH) and prediction horizon (PH)

PH→ ↓CH	1	2	3	5	8
1	0.1630	0.1420	0.1723	0.2212	0.1909
2		0.2759	0.2619	0.2899	0.2969
3			0.3236	0.3527	0.3376
5				0.5006	0.4971
8					0.8044

Table 3.3.6 shows the performance variations among the various MPC cases for this flight point. The only constraint that is occasionally violated for this flight point is the booster stall margin. As noted on the section that describes the constraint modifications made for this MPC study, the booster stall margin limit was reduced for this flight point. Without this reduction in this limit, the previous booster stall margin limit was high for a deteriorated engine, and MPC could not track the thrust for some cases.

Among the cases considered at this flight point, there is very little variation in the maximum T41. For all cases simulated, the peak T41 occurs 25 to 27 seconds into the transient. This is just at the time when the thrust is recovering to its initial high value after the Bode transient. And the peak T41 observed in each case is only slightly higher than the initial T41 for this flight point.

As is the case for all flight points considered here, there are performance penalties for doing MPC updates less frequently. These are apparent in Table 3.3.6 where the thrust tracking error clearly is larger for slower MPC update rates.

Figure 3.3.1 shows how the MPC update rate affects the thrust tracking, T41, and booster stall margin. The thrust tracking error is considerably larger for the slower MPC update rates, and this error occurs while the thrust is rapidly rising back toward its high initial value.

Table 3.3.6. Normalized maximum thrust tracking error (lb), normalized peak T41 (deg R), and normalized maximum violation of booster stall margin limit (5.431) for various MPC update rates, control horizons (CH) and prediction horizons (PH)

MPC updates every minor frame – MPC cannot track thrust															
MPC updates every 2 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	0.01	1.025	0.332	.012	1.025	0.241	.01	1.025	0.247	.008	1.025	0.341	.006	1.025	0.399
2				.007	1.025	0.173	.02	1.025	0.409	.01	1.025	0.187	.008	1.025	0.196
3							.019	1.025	0.519	.021	1.025	0.552	.011	1.025	0.249
5										.021	1.025	0.491	.021	1.025	0.441
8													.021	1.025	0.434
MPC updates every 3 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.013	1.025	0.227	.01	1.025	0.503	.008	1.025	0.423	.006	1.025	0.445	.006	1.025	0.392
2				.012	1.025	0.388	.013	1.025	0.367	.02	1.025	0.660	.011	1.025	0.367
3							.012	1.025	0.282	.012	1.025	0.445	.011	1.025	0.516
5										.014	1.025	0.298	.018	1.025	0.660
8													.017	1.025	0.474
MPC updates every 5 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.013	1.025	0.576	.012	1.025	0.527	.012	1.025	0.587	.011	1.025	0.310	.011	1.025	0.533
2				.02	1.025	0.573	.014	1.025	0.516	.015	1.025	0.240	.015	1.025	0.364
3							.014	1.025	0.378	.012	1.025	0.890	.012	1.025	0.365
5										.012	1.025	0.863	.012	1.025	2.286
8													.012	1.025	1.685
MPC updates every 7 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.019	1.025	0.579	.018	1.025	0.552	.017	1.025	0.538	.015	1.025	0.419	.019	1.025	0.470
2				.018	1.025	0.503	.02	1.025	0.547	.017	1.025	0.310	.02	1.025	0.232
3							.018	1.025	0.202	.017	1.025	1.132	.019	1.025	0.467
5										.016	1.025	0.268	.017	1.025	4.035
8													.018	1.025	2.593
MPC updates every 10 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.027	1.025	0.400	.027	1.025	0.551	.027	1.025	0.539	.035	1.025	1.254	.037	1.025	1.560
2				.027	1.025	0.435	.027	1.025	1.660	.027	1.025	0.471	.029	1.025	4.360
3							.025	1.025	0.778	.026	1.025	0.551	.033	1.025	0.770
5										.026	1.025	3.708	.025	1.025	3.813
8													.027	1.025	2.625

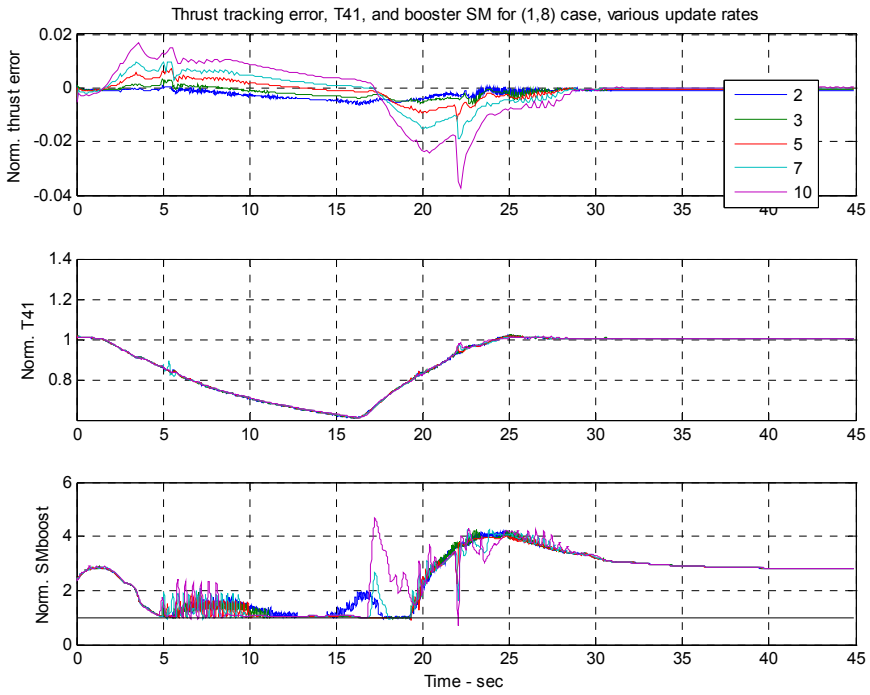

Figure 3.3.2. Thrust tracking error, T41, and booster stall margin for various MPC update rates, MPC case (1,8).

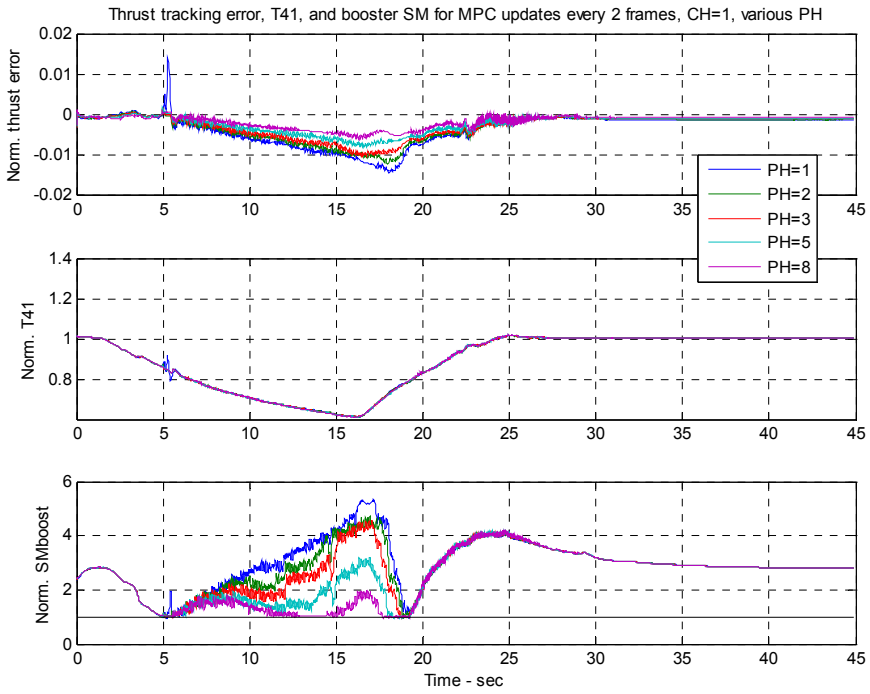

Figure 3.3.2. Thrust tracking, T41, and booster stall margin for control horizon = 1 MPC update step and various prediction horizons with MPC updates every 2 minor frames.

Figure 3.3.2 shows how the performance variables are affected by the MPC prediction horizon for a fixed control horizon (of 1 MPC update step). The thrust tracking error tends to be smallest for the longest prediction horizon, but the booster stall margin also stays near its (modified) limit for the longest prediction horizon.

3.4. Flight Point 4: 20,000 ft/Mach 0.5, reverse Bode

The reverse Bode transient is, as the name suggests, essentially the reverse of the Bode transient, hence the commanded thrust rises from an initially low level to a high level approximating maximum thrust for this condition, and then is cut back to the initial, low level again. The flight conditions for this flight point are identical to the previous one.

For this flight point and transient, MPC applied to the standard deteriorated engine used throughout this study was able to match the desired thrust reference quite well for all of the MPC update rates, control horizons, and prediction horizons that we examined. Therefore, the computational results to be presented for this flight point are normalized by the standard baseline case of MPC updates every minor frame with the control and prediction horizons both at one MPC cycle.

Tables 3.4.1 to 3.4.6 show the normalized computation times required for MPC optimization for this flight point for each of the MPC update rates, respectively, that we considered. The trends in computation effort that we observe here are similar to those we observed at the other flight points. Namely, the computational effort goes down linearly as the MPC update rate decreases, and it increases about 25% for each MPC cycle increase in the control horizon. Prediction horizon again has only a small effect on computational effort.

Table 3.4.1. Normalized computational effort for 20,000 ft/Mach 0.5, reverse Bode case, MPC updates every minor frame, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	1.0000	1.0046	1.0808	1.0647	1.0882
2		1.3265	1.3540	1.3734	1.4685
3			1.6352	1.7016	1.7446
5				2.5086	2.6615
8					4.4416

Table 3.4.2. Normalized computational effort for 20,000 ft/Mach 0.5, reverse Bode case, MPC updates every 2 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.4926	0.5172	0.5023	0.5223	0.5527
2		0.6380	0.6466	0.6575	0.7085
3			0.7640	0.7921	0.8402
5				1.2016	1.2572
8					2.0264

Table 3.4.3. Normalized computational effort for 20,000 ft/Mach 0.5, reverse Bode case, MPC updates every 3 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.3253	0.3345	0.3431	0.3522	0.3700
2		0.4101	0.4032	0.4318	0.4296
3			0.5132	0.5298	0.5435
5				0.7938	0.7623
8					1.2967

Table 3.4.4. Normalized computational effort for 20,000 ft/Mach 0.5, reverse Bode case, MPC updates every 5 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.2005	0.1999	0.2113	0.2119	0.2343
2		0.2583	0.2675	0.2652	0.2675
3			0.3104	0.3099	0.3368
5				0.4330	0.4737
8					0.7360

Table 3.4.5. Normalized computational effort for 20,000 ft/Mach 0.5, reverse Bode case, MPC updates every 7 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.1415	0.1369	0.1472	0.1586	0.1564
2		0.1781	0.1753	0.1879	0.1953
3			0.1890	0.1999	0.2154
5				0.2915	0.3339
8					0.5063

Table 3.4.6. Normalized computational effort for 20,000 ft/Mach 0.5, reverse Bode case, MPC updates every 10 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.0802	0.0710	0.0859	0.1014	0.0945
2		0.1254	0.1231	0.1323	0.1397
3			0.1524	0.1690	0.1724
5				0.2199	0.2486
8					0.3482

Table 3.4.7 shows the performance variations among the various MPC cases for this flight point. The only constraint that is violated for this flight point by any of the simulated cases is the booster stall margin. As noted on the section that describes the constraint modifications made for this MPC study, the booster stall margin limit was modified for this case. For some of the slower MPC update rates, we see large violations of the stall margin limit despite the reduction in its allowable value. For MPC updates every 5 minor frames, large booster stall margin limit violations occur for the longest prediction horizon. For MPC update rates slower than every 5 minor frames, large violations occur for some control horizons for every prediction horizon length that we tried. These violations occur either near the initial acceleration of the reverse Bode transient, or during the steep deceleration back to the initial, relatively low power point. This is probably unacceptable booster stall margin behavior. Some modification of the MPC algorithm would therefore be necessary for implementation for this flight point with reduced MPC update rates.

Figure 3.4.1 shows the thrust tracking error, T41, and booster stall margin time histories when MPC updates are calculated every 2 minor frames. All cases shown use a control horizon of 1 MPC cycle, with various prediction horizons (1, 2, 3, 5, or 8 MPC cycles). We see from this figure that none of the five cases tracks the thrust exactly in the steady state. All 5 cases indicate slight undershoots of the final thrust level at the end of the simulated transient.

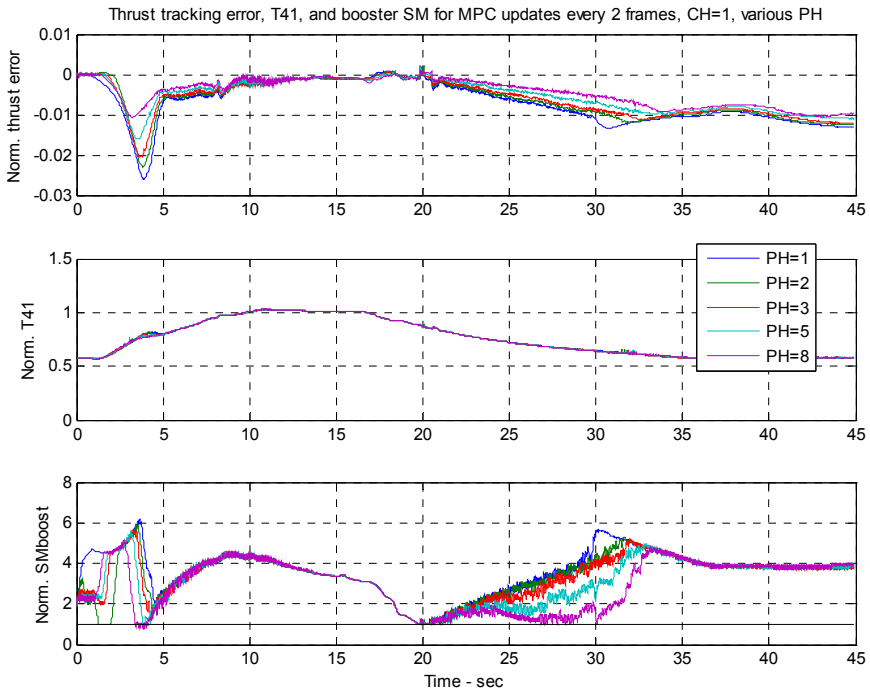

Figure 3.4.1. Thrust tracking error, T41, and booster stall margin for MPC updates every 2 minor frames, CH=1 MPC cycle, PH = 1, 2, 3, 5, 8 MPC cycles.

The thrust undershoot is substantially less when we increase the control horizon to 2 MPC cycles. This is illustrated in Figure 3.4.2. We see in this figure that the smallest thrust tracking errors occur for the longest prediction horizon (8 MPC cycles), but this case also produces a booster stall margin that "rides" its limit.

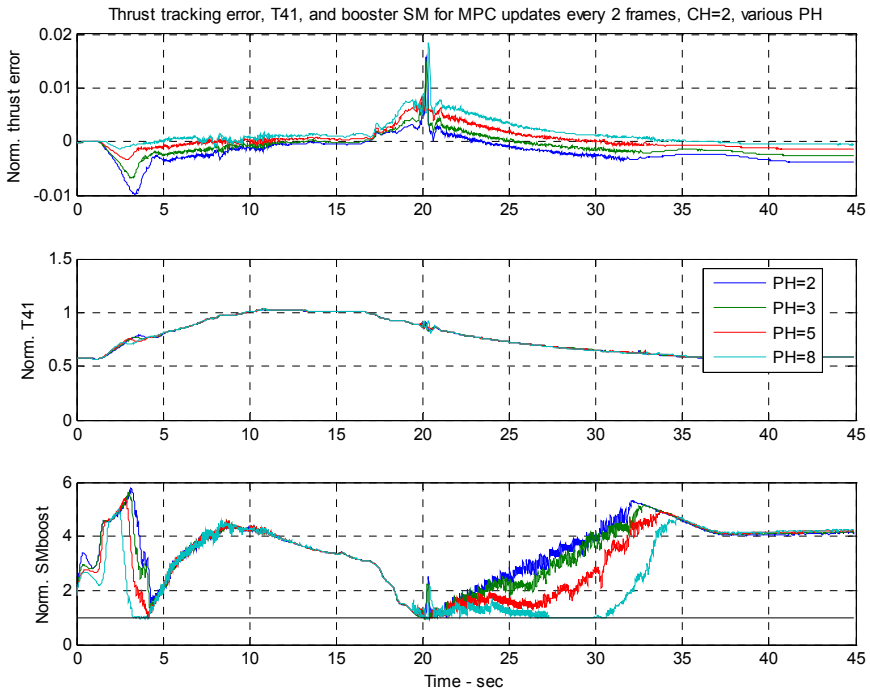

Figure 3.4.2. Thrust tracking error, T41, and booster stall margin for MPC updates every 2 minor frames, CH=2 MPC cycles, PH = 2, 3, 5, 8 MPC cycles.

Figure 3.4.3 shows the histories of the thrust tracking error, T41, and booster stall margin for MPC updates every 2 minor frames with the long prediction horizon (8 MPC cycles) but with various control horizons (1, 2, 3, 5, and 8 MPC cycles). The (1,8) case is the one for which the thrust is a few hundred pounds below the demanded thrust late in the transient. The other control horizons produce better thrust tracking for this transient, but they also produce more booster stall margin limit violations over the time period from 10 to about 32 sec.

The minimum mean square thrust tracking error for MPC for this flight point and this transient occurs for the (2,8) case with MPC updates every minor frame. The (2,8) case for MPC updates every 2 minor frames (one of the cases illustrated in Figure 4.2) gives the sixth-smallest mean square thrust tracking error among the 15 cases of MPC control and prediction horizon tested at this flight point, and the third-smallest of the cases with control horizon less than 3 MPC cycles. For MPC updates every 3 minor frames, the (2,8) case is no longer one that gives one of the smaller mean square thrust tracking errors, but at this slower MPC update rate, we begin to see the large booster stall margin limit violations for the longer prediction horizons. The (2,8) case for various MPC update rates is then a good case for performance results comparisons. Figure 4.x shows the time histories of the thrust tracking error, T41, and the booster stall margin for this flight point and transient.

Table 3.4.7. Normalized maximum thrust tracking error (lb), normalized peak T41 (deg R), and normalized maximum violation of booster stall margin limit (5.431) for various MPC update rates, control horizons (CH) and prediction horizons (PH)

MPC updates every minor frame															
PH→ ↓CH	1			2			3			5			8		
1	.027	1.029	0.645	.025	1.029	0.279	.023	1.029	0.255	.02	1.029	0.050	.016	1.029	0.
2				.015	1.029	0.352	.012	1.029	0.381	.013	1.029	0.281	.013	1.029	0.357
3							.013	1.029	0.229	.013	1.029	0.432	.013	1.029	0.295
5										.013	1.029	0.250	.017	1.029	0.331
8													.013	1.029	0.237
MPC updates every 2 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.026	1.029	0.125	.02	1.029	0.156	.022	1.029	0.049	.016	1.029	0.121	.01	1.029	1.367
2				.015	1.029	0.330	.016	1.029	0.310	.008	1.029	0.158	.018	1.029	0.380
3							.016	1.029	0.375	.008	1.029	0.161	.016	1.029	0.472
5										.019	1.029	0.470	.016	1.029	0.441
8													.012	1.029	0.452
MPC updates every 3 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.025	1.029	0.239	.02	1.029	0.217	.018	1.029	0.176	.012	1.029	1.402	.008	1.029	1.702
2				.011	1.029	0.114	.019	1.029	0.489	.009	1.029	0.797	.010	1.029	0.911
3							.012	1.029	0.109	.011	1.029	0.365	.010	1.029	0.506
5										.01	1.029	0.379	.009	1.029	0.449
8													.009	1.029	0.563
MPC updates every 5 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.023	1.029	0.018	.018	1.029	0.688	.013	1.029	1.050	.011	1.029	0.805	.009	1.029	1.926
2				.018	1.029	0.117	.013	1.029	0.454	.014	1.029	0.460	.013	1.029	0.477
3							.014	1.029	0.517	.013	1.029	0.529	.012	1.029	2.353
5										.012	1.029	0.965	.012	1.029	2.612
8													.012	1.029	3.958
MPC updates every 7 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.024	1.029	0.865	.017	1.029	1.853	.017	1.029	2.165	.015	1.029	1.754	.02	1.029	3.387
2				.018	1.029	0.486	.019	1.029	0.982	.016	1.029	0.438	.02	1.029	0.312
3							.016	1.029	0.438	.016	1.029	2.894	.017	1.029	0.447
5										.015	1.029	0.310	.015	1.029	2.319
8													.016	1.029	3.937
MPC updates every 10 minor frames															
PH→ ↓CH	1			2			3			5			8		
1	.026	1.029	1.143	.026	1.029	2.593	.025	1.029	2.146	.027	1.029	7.852	.024	1.029	2.697
2				.025	1.029	0.667	.023	1.029	2.679	.025	1.029	3.839	.027	1.029	1.241
3							.023	1.029	1.006	.024	1.029	0.825	.027	1.029	0.080
5										.024	1.029	0.293	.024	1.029	5.342
8													.025	1.029	3.063

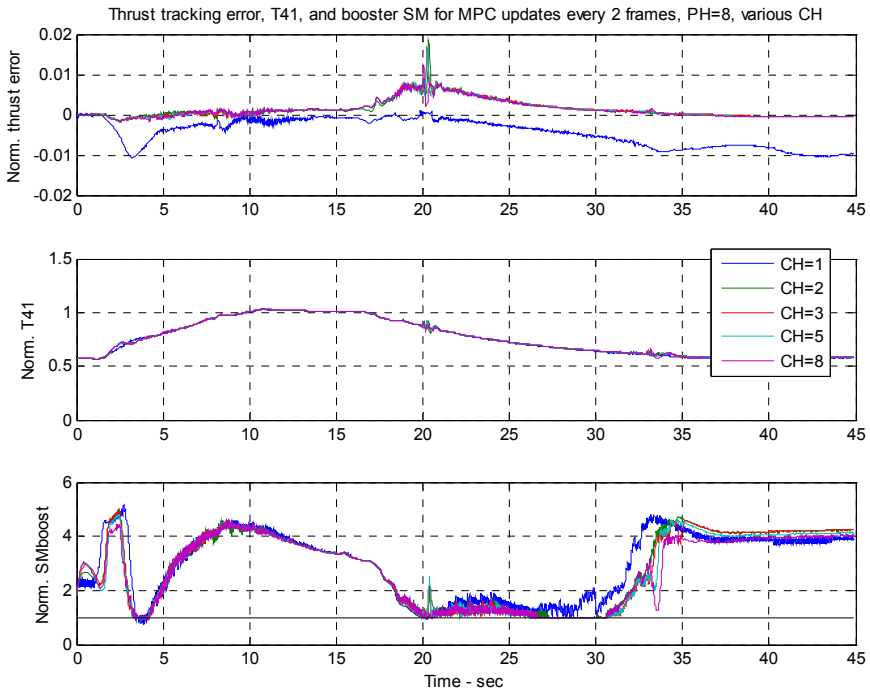

Figure 3.4.3. Thrust tracking error, T41, and booster stall margin for MPC updates every 2 minor frames, CH=1, 2, 3, 5, 8 MPC cycles, PH = fixed at 8 MPC cycles.

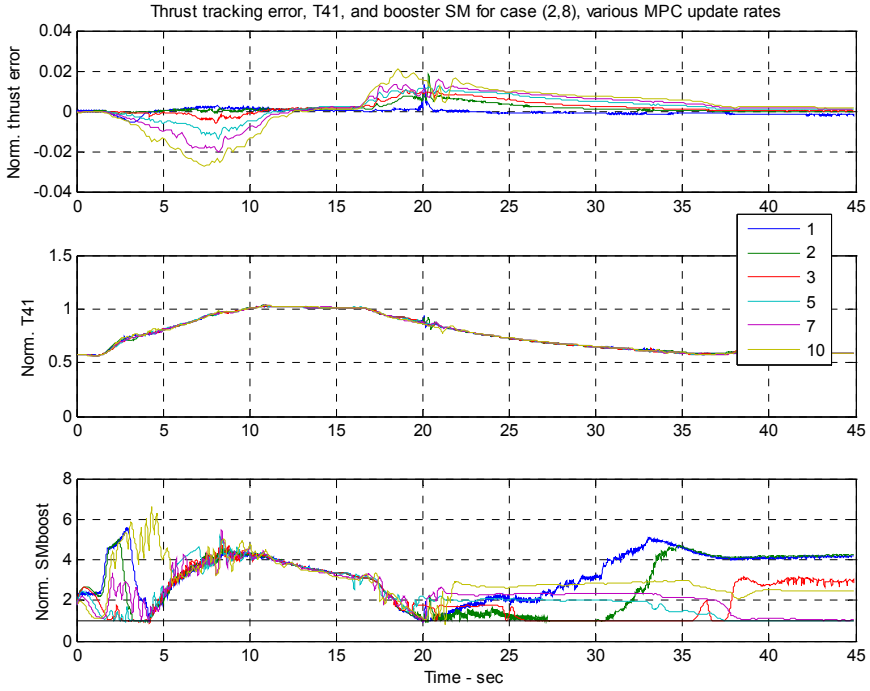

Figure 3.4.4. Thrust tracking error, T41, and booster stall margin for various MPC update rates, CH = 2 MPC cycles, PH = 8 MPC cycles.

Figure 3.4.4 shows the thrust tracking error, T41, and booster stall margin for several different MPC update rates for the (2,8) case. We see that the thrust tracking for this transient tends to be best when MPC updates are made more frequently. However, the booster stall margin issues do not vary uniformly with the MPC update rate.

3.5. Flight Point 5: 35,000 ft/Mach 0.84, reverse Bode

This flight point approximates a cruise condition for this application. As for the previous flight point, the reverse Bode transient involves a rise in commanded thrust from an initially low level to a high level approximating maximum thrust for this condition, and then a cut-back to the initial, low thrust level. For this particular condition, the peak of the commanded thrust is about (normalized) 1.006 lb about 17 sec into the transient.

For this flight point and transient, when we applied MPC to the standard deteriorated engine used throughout this study, there were several combinations of the MPC update, control horizon, and prediction horizon for which MPC could not reach the peak commanded thrust within about (normalized) 0.23 lb. This is a 25% shortfall in thrust, which is unacceptable for this condition. These large thrust-tracking errors occurred for all cases with MPC updates at every minor frame, all cases with a prediction horizon less than 5 MPC cycles for MPC updates every 2 minor frames, all cases with the prediction horizon less than 3 MPC cycles for MPC updates every 3 minor frames, and the only case considered with prediction horizon equal to 1 MPC cycle for MPC updates every 5 minor frames. Apparently, there is something "magical" about predicting the response at least 9 minor frames ahead that allows the baseline MPC algorithm to work for this flight point and transient. All of these cases also produced compressor stall margin limit violations in addition to booster stall margin limit violations. These cases will be omitted from the figures because their thrust-tracking behavior is unacceptable. However, the computation times for these cases are still relevant and will be included in the computational effort results.

Tables 3.5.1 through 3.5.6 present the computational effort required for MPC optimization for each of the MPC update rates, control horizons, and prediction horizons considered in this study. As with most of the other flight points, the computational effort is normalized by the effort required for the baseline case of MPC updates every minor frame, and control and prediction horizons both equal to 1 MPC cycle. The trends in the computational effort are the same for this flight pint as the others, namely a nearly linear decrease in computational effort as the MPC update rate is decreased, a 25% to 35% increase in effort for each additional MPC cycle added to the control horizon, and only a small increment in effort for increasing the prediction horizon.

Table 3.5.7 summarizes the MPC performance results for this flight point. Cases where the peak thrust demand was not tracked within (normalized) 0.23 lb are not shown. There are two cases in the table – (3,5) for MPC updates every 10 minor frames and (1,8) for MPC updates every 7 minor frames – for which a few compressor stall margin limit violations occur.

Table 3.5.1. Normalized computational effort for 35,000 ft/Mach 0.84, reverse Bode case, MPC updates every minor frame, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	1.0000	1.0756	1.1004	1.1201	1.1114
2		1.4451	1.4757	1.4844	1.5288
3			1.7985	1.8123	1.8343
5				2.6657	2.8048
8					4.5983

Table 3.5.2. Normalized computational effort for 35,000 ft/Mach 0.84, reverse Bode case, MPC updates every 2 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.5063	0.5138	0.5225	0.5381	0.5456
2		0.7044	0.7038	0.6721	0.6882
3			0.8585	0.8031	0.8216
5				1.1720	1.2280
8					2.0011

Table 3.5.3. Normalized computational effort for 35,000 ft/Mach 0.84, reverse Bode case, MPC updates every 3 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.3308	0.3430	0.3435	0.3684	0.3655
2		0.4648	0.4128	0.4411	0.4740
3			0.5277	0.5294	0.5797
5				0.8054	0.8464
8					1.3175

Table 3.5.4. Normalized computational effort for 35,000 ft/Mach 0.84, reverse Bode case, MPC updates every 5 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.2038	0.1998	0.2107	0.2252	0.2454
2		0.2639	0.2581	0.2852	0.2898
3			0.3199	0.3401	0.3562
5				0.4694	0.4561
8					0.7205

Table 3.5.5. Normalized computational effort for 35,000 ft/Mach 0.84, reverse Bode case, MPC updates every 7 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.1426	0.1339	0.1518	0.1582	0.1634
2		0.1755	0.1796	0.1882	0.1911
3			0.1871	0.2032	0.2177
5				0.3020	0.3302
8					0.4827

Table 3.5.6. Normalized computational effort for 35,000 ft/Mach 0.84, reverse Bode case, MPC updates every 10 minor frames, varying control (CH) and prediction horizons (PH)

PH→ ↓CH	1	2	3	5	8
1	0.0733	0.0924	0.0982	0.1149	0.1224
2		0.1316	0.1380	0.1357	0.1409
3			0.1576	0.1645	0.1686
5				0.2211	0.2390
8					0.3516

Figure 3.5.1 shows the thrust tracking results for MPC updates every 7 minor frames. Here, the prediction horizon is held at 8 MPC cycles and the control horizon varies (1, 2, 3, 5, and 8 MPC cycles). We see from the top plot that all five cases track the thrust reasonably well. The bottom plot shows the thrust tracking error, and here we see that the longer control horizons produce less thrust undershoot near the end of the transient.

Figure 3.5.2 shows the compressor stall margin and booster stall margin for the same five cases shown in Figure 3.5.1. Here, we see that for all of the cases, the booster stall margin "rides" its limit for some time after 20 sec of the transient. The longer control horizons tend to produce the longest period with the booster stall margin right at its limit. We also see that the compressor stall margin limit violation noted in Table 3.5.7 for the (1,8) case is somewhat of an anomaly. It occurs only very briefly just after 5 sec and occurs simultaneously with a significant booster stall limit violation. But both stall margins quickly return to values above their limits.

Figure 3.5.3 shows the thrust tracking error and booster stall margin behavior for MPC updates every 5 minor frames with the prediction horizon fixed at 8 MPC cycles and various control horizons (1, 2, 3, 5, and 8 MPC cycles). We see that the thrust tracking is near zero towards the end of the transient for all but the shortest control horizon.

Table 3.5.7. Maximum thrust tracking error (lb), peak T41 (deg R), and maximum violation of booster stall margin limit (5.431) for various MPC update rates, control horizons (CH) and prediction horizons (PH)

MPC updates every minor frame – all cases fail to track peak thrust demand																
PH→ ↓CH	1			2			3			5			8			
1	.23			.23			.23			.23			.23			
2				.23			.23			.23			.23			
3							.23			.23			.23			
5										.23			.23			
8													.23			
MPC updates every 2 minor frames – several cases fail to track peak thrust demand																
PH→ ↓CH	1			2			3			5			8			
1	.23			.23			.23			.03	3044	0.078	.034	3047	1.024	
2				.23			.23			.03	3045	0.650	.033	3048	0.133	
3							.23			.016	3044	0.121	.034	3047	0.573	
5										.026	3045	0.352	.032	3047	0.541	
8													.032	3047	0.539	
MPC updates every 3 minor frames – a few cases fail to track peak thrust demand																
PH→ ↓CH	1			2			3			5			8			
1	.23			.23			.033	3043	0.161	.022	3046	0.737	.014	3050	0.936	
2				.23			.02	3045	0.124	.016	3047	0.925	.014	3050	0.941	
3							.029	3045	0.400	.03	3047	0.401	.014	3051	1.228	
5										.04	3046	0.612	.031	3051	0.879	
8													.047	3050	1.679	
MPC updates every 5 minor frames – (1,1) case fails to track peak thrust demand																
PH→ ↓CH	1			2			3			5			8			
1	.24			.033	3045	0.	.022	3047	0.656	.022	3050	1.011	.012	3054	1.291	
2				.022	3046	0.219	.022	3048	0.329	.022	3052	0.262	.019	3058	0.116	
3							.022	3047	0.583	.019	3052	2.608	.015	3061	0.514	
5										.017	3052	3.150	.015	3059	0.705	
8													.017	3059	0.747	
MPC updates every 7 minor frames – (1,8) case also violates compressor stall margin limit																
PH→ ↓CH	1			2			3			5			8			
1	.037	3045	0.521	.026	3046	0.616	.017	3047	2.007	.015	3055	1.755	.015	3100	5.355*	
2				.025	3048	0.548	.019	3051	0.553	.019	3058	0.712	.015	3066	0.227	
3							.024	3051	0.573	.017	3057	2.112	.015	3066	2.490	
5										.016	3057	3.566	.015	3066	0.817	
8													.015	3065	1.253	
MPC updates every 10 minor frames – (3,5) case also violates compressor stall margin limit																
PH→ ↓CH	1			2			3			5			8			
1	.031	3047	1.280	.021	3047	2.431	.021	3050	2.384	.021	3059	7.354	.024	3067	3.699	
2				.03	3052	1.357	.023	3055	1.429	.022	3063	0.261	.021	3071	0.256	
3							.027	3055	2.093	.035	3342	8.549*	.031	3152	5.376	
5										.018	3063	3.151	.019	3069	0.825	
8													.018	3068	5.225	

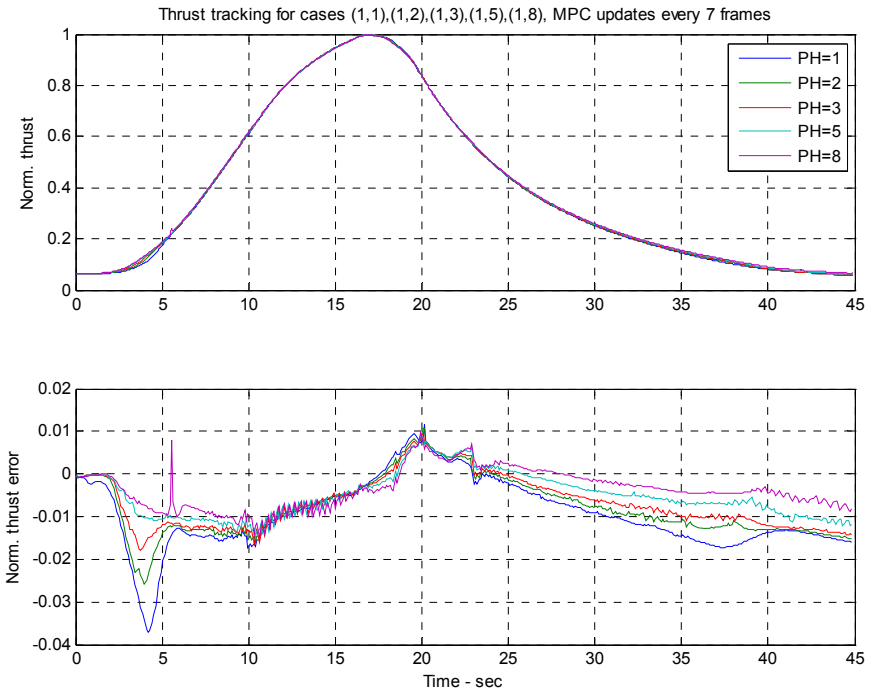

Figure 3.5.1. Thrust tracking results for MPC updates every 7 minor frames, CH=1, PH=1,2,3,5,8.

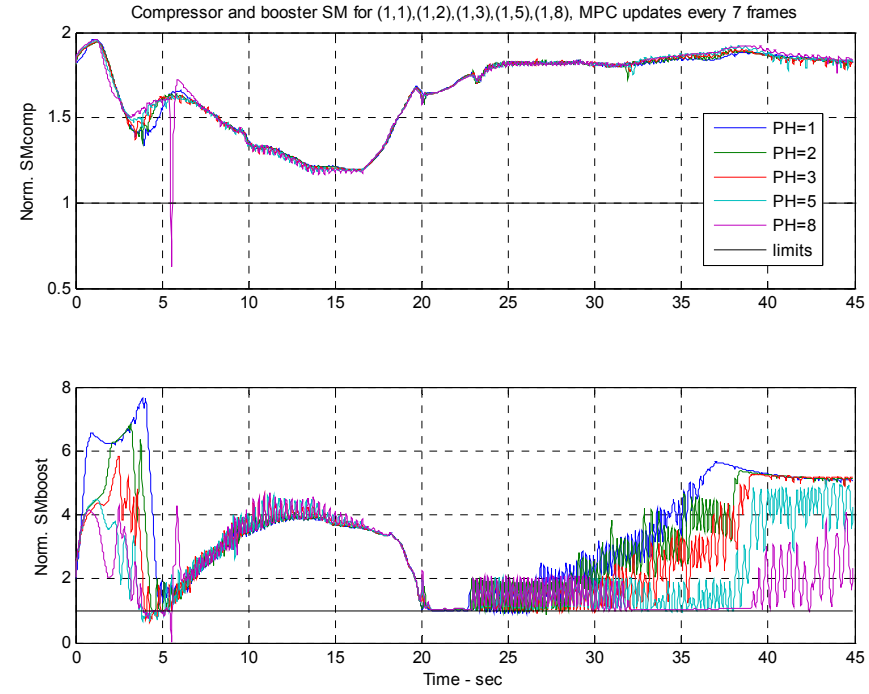

Figure 3.5.2. Compressor and booster stall margins for MPC updates every 7 minor frames, CH = 1, PH = 1, 2, 3, 5, 8.

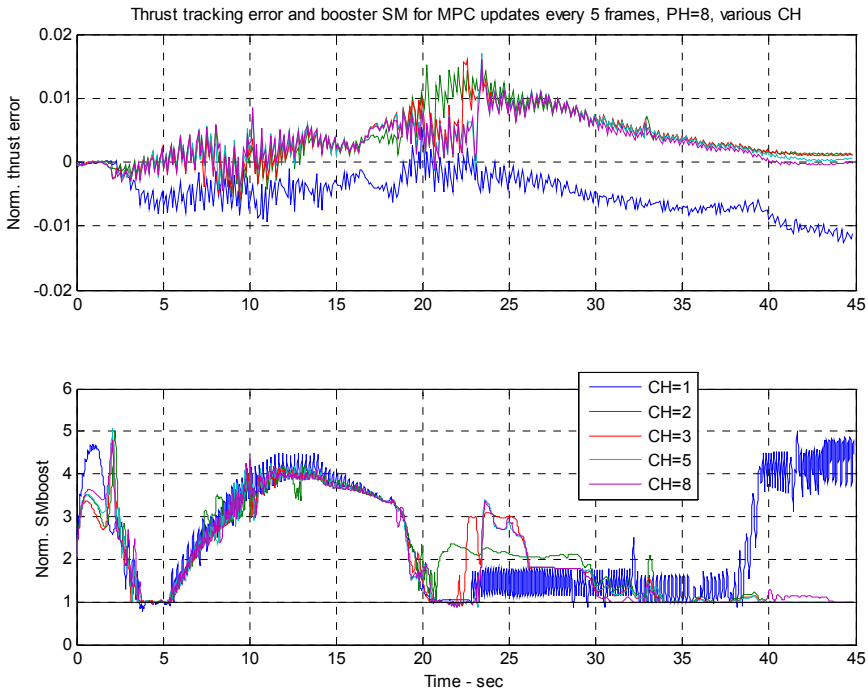

Figure 3.5.3. Thrust tracking and booster stall margin for MPC updates every 5 minor frames, PH = 8, CH = 1, 2, 3, 5, 8.

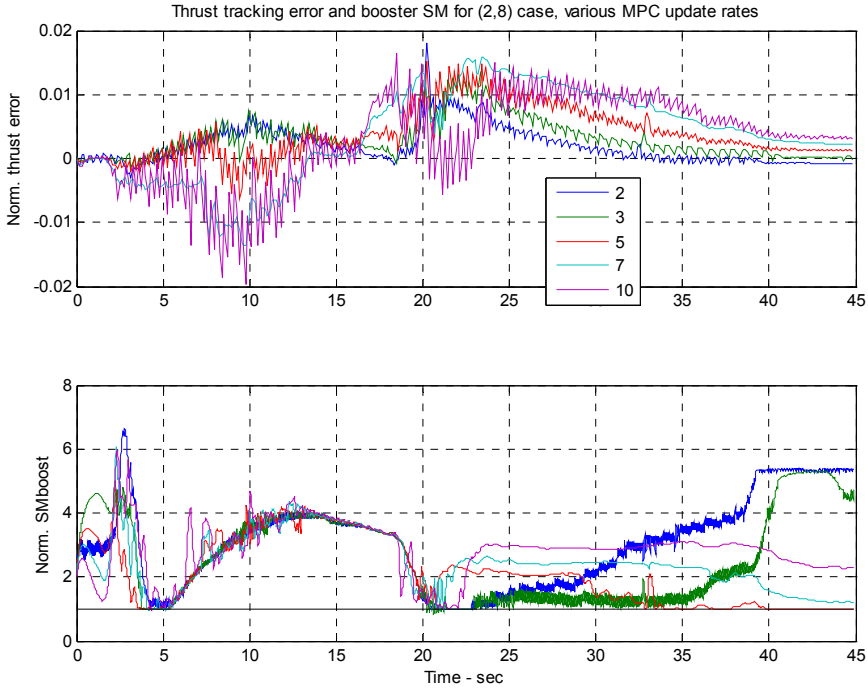

Figure 3.5.4. Thrust tracking error and booster stall margin for various MPC update rates – 2, 3, 5, 7, and 10 minor frames per update, (2,8) case.

The effects of the MPC update rate are shown in Figure 5.4. Here, the thrust tracking error and the booster stall margin are plotted for the (2,8) case for 5 different MPC update rates (every 2, 3, 5, 7 and 10 minor frames). We see that the thrust tracking error tends to be smallest for the fastest MPC update rate. However, there is not a clear pattern to the booster stall margin behavior.

4. Removal of Ineffective Constraints

For the idle-to-takeoff power transient at sea level standard (SLS) conditions studied to this point, several of the constraints that are taken into consideration by the MPC algorithm are not violated at any point during the transient. The MPC optimization is performed while taking into account limits on the maximum core and fan speeds (N1 and N2), the maximum combustor inlet pressure (PS3), the minimum stall margins for fan, booster and compressor, and the minimum Stage 1 clearance in the turbine. For the 30-second idle-to-takeoff transient at the SLS flight condition, only the booster and compressor stall margin limits are effective. If the optimization can be done with only the effective limits taken into account, some savings in computational effort should result.

To study this hypothesis, we altered the MPC set-up to include only the booster and compressor stall margin constraints. Then, we ran the modified MPC simulation for all of the cases considered above in the study of various MPC update rates, control horizons, and prediction horizons. A summary of the results follows.

4.1. Baseline Case

For essentially all of the cases using a single step for the control horizon, the results for the MPC-controlled input histories, and therefore the results for the histories of all outputs, including the tracked variables thrust and T41, were identical at every time point to what was obtained before with all of the constraints considered. The only exceptions among the cases with control horizon = 1 were the (control horizon = 1, prediction horizon = 1) case when MPC updates are done every minor frame, the (1,8) case when MPC updates are done every 7 minor frames, and the (1,1) case when updates are done every 10 minor frames. The maximum thrust differences over a 30-second transient for these three cases relative to the corresponding cases with all constraints considered were, respectively and normalized, 3.76×10^{-4} lb, 7.53×10^{-5} lb, and 5.68×10^{-3} lb, so these differences are negligible. The case-to-case variations in T41 are similarly negligible.

To illustrate how tiny the thrust and T41 performance differences are, Figure 4.1.1 superimposes the thrust (top plot) and T41 (bottom plot) trajectories for the (1,1) case with MPC updates every 10 minor frames with and without all constraints considered. This is the case with the least agreement.

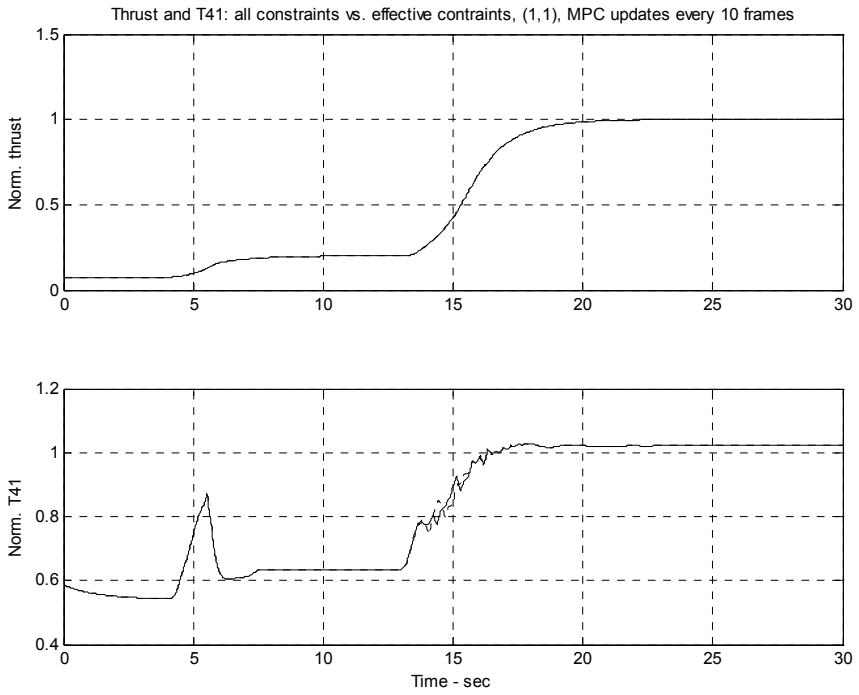

Figure 4.1.1. Thrust and T41 trajectories with and without all constraints considered, (1,1) case with MPC updates every 10 minor frames

For control horizons longer than one MPC update step, most of the cases produced slightly different results than those obtained when all constraints were considered. However, in all but a few cases, the differences are again small compared to the typical case-to-case variations. Out of the 53 combinations of control horizon, prediction horizon, and MPC update rate considered with a control horizon of two or more steps (2, 3, 5 and 8 were considered), only 7 cases produced a normalized maximum thrust difference of more than 2.09×10^{-3} lb or a maximum T41 difference of more than 4.47×10^{-2} deg R. The normalized maximum thrust difference for any case was 1.2×10^{-2} lb for case (3,5) with updates every 10 minor frames. The thrust and T41 trajectories for this case are depicted in Figure 4.1.2. The booster and stall margins are shown in Figure 4.1.3, which shows that they also change very little when the ineffective constraints are no longer considered.

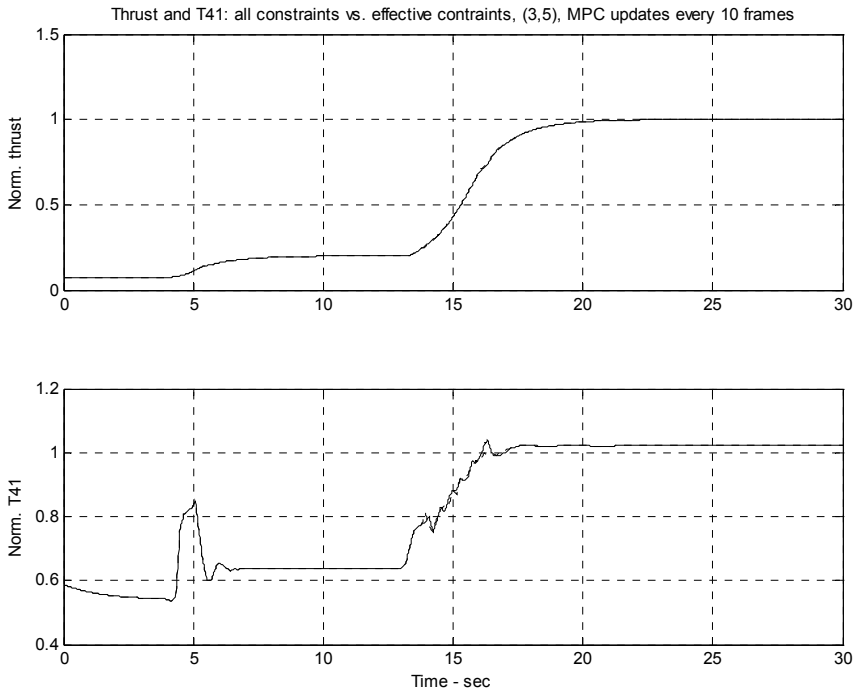

Figure 4.1.2. Thrust and T41 trajectories for (3,5) case with MPC updates every 10 minor frames

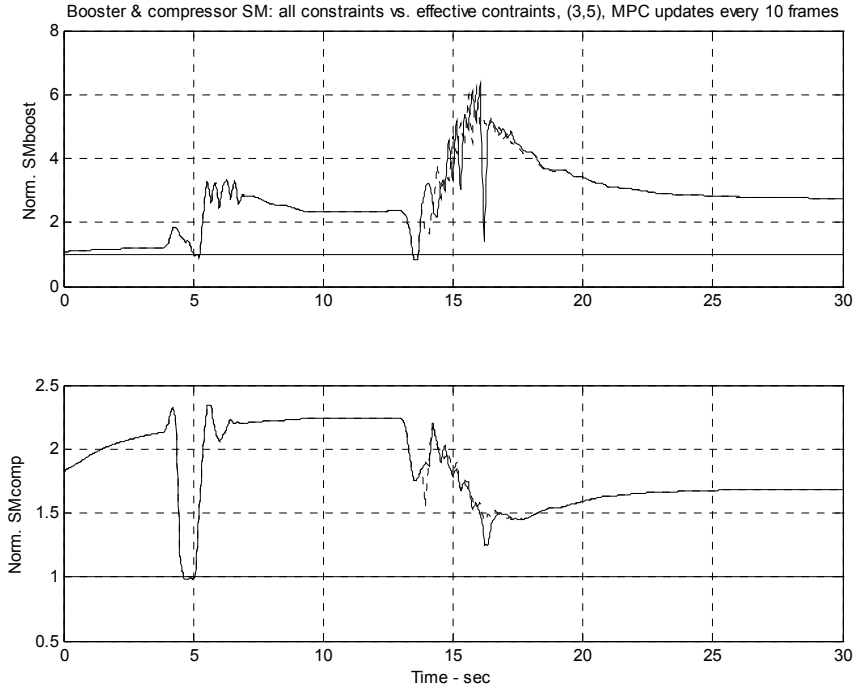

Figure 4.1.3. Stall margin trajectories for (3,5) case with MPC updates every 10 minor frames

The maximum T41 difference relative to the results with all constraints considered for control horizons larger than one occurs for case (5,8) with MPC updates every 10 minor frames. The maximum T41 difference is 5.09×10^{-2} deg R, which is actually smaller than the maximum T41 difference for case (1,1) with updates every 10 frames above. Figures 4.1.4 and 4.1.5 depict, respectively, the thrust and T41 trajectories, and the booster and compressor stall margin trajectories for this case.

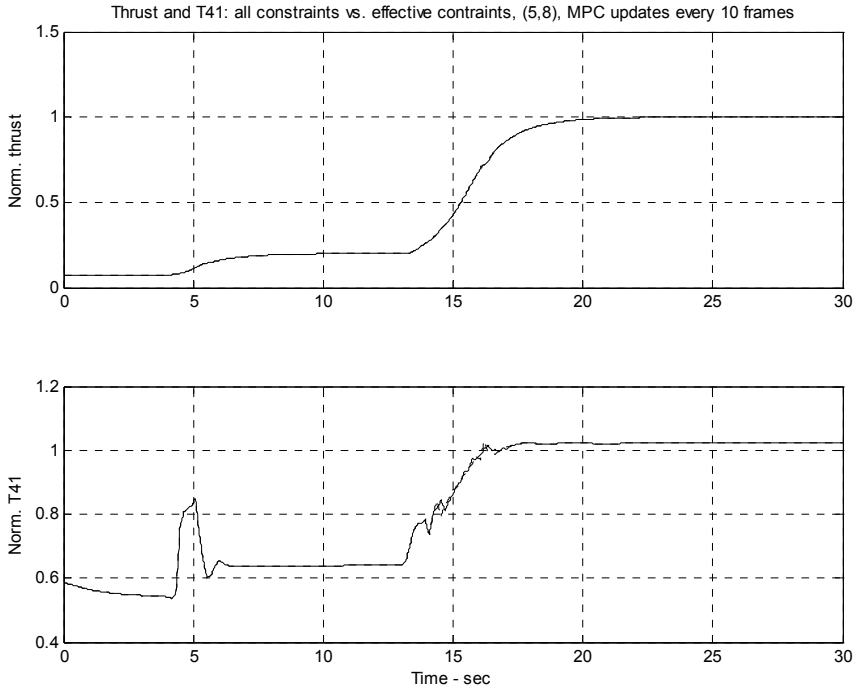

Figure 4.1.4. Thrust and T41 trajectories with and without all constraints considered, case (5,8) with MPC updates every 10 minor frames.

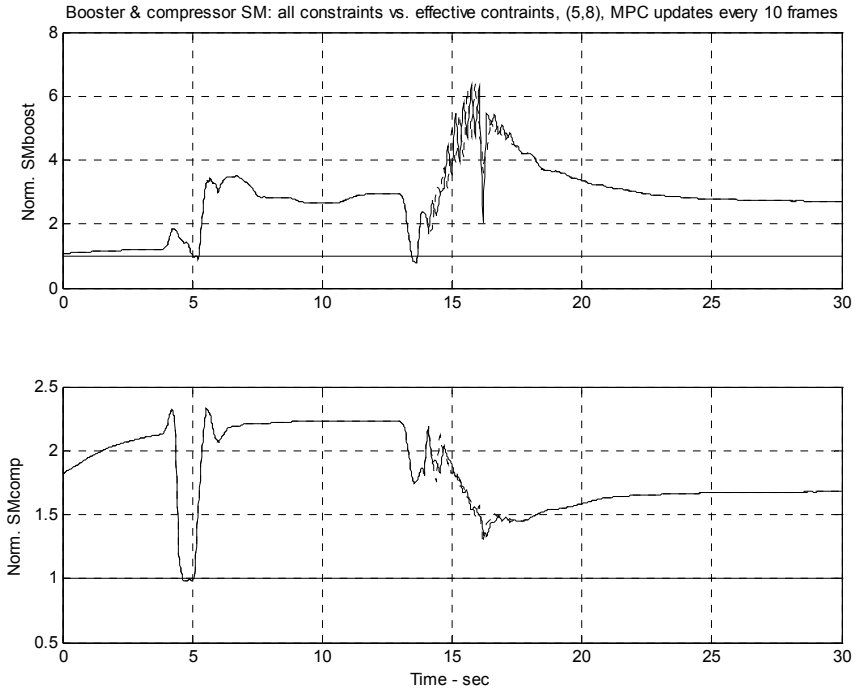

Figure 4.1.5. Booster and compressor stall margin trajectories for case (5,8) with updates every 10 minor frames, with and without all constraints considered

Our conclusion then is that eliminating the consideration of ineffective constraints has very little effect on the MPC-controlled inputs, and thus very little effect on the tracked outputs or on the outputs for which constraints are sometimes effective.

While the performance of the MPC is essentially unaltered for the baseline case with the ineffective constraints not considered, there is a big difference is in the computation effort required. The total computation time for MPC optimization, determined by the same procedure as previously, for each of the cases considered for a 30-second, idle-to-takeoff transient at SLS is shown in Table 4.1.1.

Table 4.1.1. Total MPC optimization computation times with only two constraints considered

Control horizon, pred. hor.	MPC updt every frame	MPC updt every 2 frames	MPC updt every 3 frames	MPC updt every 5 frames	MPC updt every 7 frames	MPC updt every 10 frames
1, PH=1,	8.781	5.172	2.813	2.297	1.703	1.281
2	9.563	6.438	3.609	2.328	1.813	1.422
3	10.047	5.063	3.859	2.25	1.703	1.344
5		5.531	3.516	2.281	1.688	1.391
8		6.141	3.797	2.453	1.891	1.375
2, PH=2,	13.219	6.359	4.594	2.984	2.016	1.578
3	12.688	6.734	4.5	2.781	2.156	1.734
5		7.109	4.703	3.125	2.313	1.656
8		7.078	5.078	3.047	2.226	1.625
3, PH=3,	16.219	8.328	5.672	3.375	2.516	1.875
5		8.797	6.	3.953	2.609	1.969
8		9.078	6.984*	3.75	2.625	1.923
5, PH=5,		12.766	8.578	5.609	4.547*	2.766
8		13.344	9.469	5.484	4.094	2.813
8, PH=8		23.219	15.625	10.266	7.469	4.969

* - Convergence error took place for one MPC update.

These computation times are substantially shorter than those for the same cases when all of the constraints are taken into account. Table 4.1.2 shows the percent reduction in computation time for each case considered.

Table 4.1.2. Percent reduction in MPC computation times with only two constraints considered

Control horizon, pred. hor.	MPC updt every frame	MPC updt every 2 frames	MPC updt every 3 frames	MPC updt every 5 frames	MPC updt every 7 frames	MPC updt every 10 frames
1, PH=1,	52%	43%	51%	36%	38%	32%
2	41%	24%	39%	36%	33%	33%
3	42%	41%	36%	39%	37%	26%
5		38%	43%	42%	34%	26%
8		35%	43%	41%	33%	34%
2, PH=2,	37%	41%	35%	35%	39%	27%
3	42%	39%	39%	38%	34%	26%
5		37%	39%	35%	33%	38%
8		40%	38%	39%	41%	43%
3, PH=3,	41%	38%	36%	38%	37%	41%
5		36%	36%	34%	36%	31%
8		38%	30%	37%	40%	35%
5, PH=5,		36%	37%	36%	27%	35%
8		37%	34%	36%	34%	33%
8, PH=8		32%	31%	29%	32%	36%

There are some variations in computation time for successive runs of the same case, but those variations do not alter the pattern indicated by the table. Namely that the reduction of the constraints considered from seven to two has cut the computation time for MPC optimization by 24% to 52%, with about a 36% reduction for most cases.

The drawback to eliminating constraints, of course, is that at different flight conditions and for different transients, different constraints are effective. Thus, the constraints that can be ignored because they are ineffective for one flight condition or transient might be the effective constraints for a different flight condition or transient. The constraints to be included for any particular condition and transient can only be determined by simulation of that condition and transient. We consider other conditions and transients below.

4.2 Reduced Constraints at Other Flight Points

Throughout the discussion of the results for MPC at the various flight points, the only constraints that are ever violated are the stall margins on the fan, booster, and compressor. There are some Stage 1 clearance violations for the SLS takeoff-to-idle case, but these are a result of the transient itself and not MPC control. In fact, once the decrease in the power command begins, the Stage 1 clearance minimum violations disappear. Thus, it is primarily the stall margin limits (particularly the booster stall margin limit) that are effective for almost all of the cases studied here.

One approach to reducing the effort required for MPC implementation is to consider only those constraints that are occasionally active. In this case, that means considering only the stall margin constraints.

We altered the MPC algorithm to include only the three stall margin limits as constraints. Then, we re-simulated every case that we discussed above for the various flight points and transients. Because these simulations occurred near the end of the study, time does not permit a complete analysis of the results. However, we can note the following.

The computational savings, as measured by the total CPU time required for MPC optimization, is reduced by about 30% when using only the three stall margin constraints instead of the full set of seven constraints. This savings varies on a case-by-case basis, but is rarely less than 20% and only occasionally exceeds 35%. This agrees with Table 4.1.2 above for the baseline MPC case.

The performance of MPC with reduced constraints is briefly analyzed below for each of the five flight points and transients considered in this study. For each flight point and transient and each MPC update rate for that case, we characterize the engine performance variables as matching with "exact agreement," matching with "near agreement," or showing "significant differences." "Exact agreement" means precisely that, namely that the time histories of every variable in the simulation are exactly the same as when MPC is applied to the same transient with all constraints considered. "Nearly matching" means that the time histories of both of the tracked variables – thrust and T41 – differ by no more than 8.3×10^{-4} lb or 2.98×10^{-2} deg R from the time histories using MPC with all constraints included. "Significant difference" means that the time histories

of one or both of the tracked variables differ by more than these values from the time histories using MPC with all constraints included. Note that this disagreement need only happen at a single point in time to classify the case as showing "significant differences."

The MPC cases are indicated in the usual format of (CH,PH), where CH is the control horizon and PH the prediction horizon, for each MPC update rate

Flight point 1: SLS, idle-takeoff:

MPC update rate (minor frames per MPC update): 1
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(2,3),(2,8),(3,3),(3,5),(5,8),(8,8)
Cases nearly agreeing: (2,5),(5,5)
Cases with significant differences: None

MPC update rate (minor frames per MPC update): 2
Cases with exact agreement:
(1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(2,3),(2,5),(2,8),(3,3),(3,5),(5,5),(5,8),(8,8)
Cases nearly agreeing: (3,3)
Cases with significant differences: None

MPC update rate (minor frames per MPC update): 3
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(2,3),(2,8),(3,3),(3,5),(5,5)
Cases nearly agreeing: None
Cases with significant differences: (3,8),(5,8),(8,8)

MPC update rate (minor frames per MPC update): 5
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,3),(3,3),(3,5),(5,5)
Cases nearly agreeing: (2,8),(5,8)
Cases with significant differences: (2,2),(2,5),(3,8),(8,8)

MPC update rate (minor frames per MPC update): 7
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(3,3),(3,5),(3,8)
Cases nearly agreeing: (5,5)
Cases with significant differences: (2,3),(2,5),(2,8),(5,8),(8,8)

MPC update rate (minor frames per MPC update): 10
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(2,2),(2,3),(2,8),(3,3),(3,5),(3,8),(5,5)
Cases nearly agreeing: None
Cases with significant differences: (1,8),(5,8),(8,8)

Flight point 2: SLS, takeoff-idle:

MPC update rate (minor frames per MPC update): 1
No MPC case tracks thrust

MPC update rate (minor frames per MPC update): 2

No MPC case tracks thrust

MPC update rate (minor frames per MPC update): 3
Only 7 MPC cases track thrust: (1,3),(1,5),(1,8),(2,5),(3,5),(5,5),(8,8)
All 7 cases have significant differences

MPC update rate (minor frames per MPC update): 5
All 15 MPC cases have significant differences

MPC update rate (minor frames per MPC update): 7
All 15 MPC cases have significant differences

MPC update rate (minor frames per MPC update): 10
All 15 MPC cases have significant differences

Flight point 3: 20,000 ft/Mach 0.5, Bode:

MPC update rate (minor frames per MPC update): 1
No MPC case tracks thrust

MPC update rate (minor frames per MPC update): 2
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(2,3),(3,3),(3,5),(5,5)
Cases nearly agreeing: (5,8)
Cases with significant differences: (2,5),(2,8),(3,8),(8,8)

MPC update rate (minor frames per MPC update): 3
Cases with exact agreement: (1,1),(1,2),(1,3),(1,8),(2,2),(2,8)
Cases nearly agreeing: (1,5),(2,3),(2,5),(3,3),(3,5),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: None

MPC update rate (minor frames per MPC update): 5
Cases with exact agreement: (1,1),(1,3),(1,5),(1,8),(3,3),(2,8)
Cases nearly agreeing: (1,2),(2,2),(2,3)
Cases with significant differences: (2,5),(2,8),(3,5),(3,8),(5,5),(5,8),(8,8)

MPC update rate (minor frames per MPC update): 7
Cases with exact agreement: (1,3),(1,5),(1,8)
Cases nearly agreeing: (1,1),(1,2),(2,2),(2,3),(2,5),(3,3)
Cases with significant differences: (2,8),(3,5),(3,8),(5,5),(5,8),(8,8)

MPC update rate (minor frames per MPC update): 10
Cases with exact agreement: (1,1),(1,3),(1,8)
Cases nearly agreeing: (1,2),(1,5),(2,2),(2,3),(2,8),(3,3),(3,8),(5,8)
Cases with significant differences: (2,5),(3,5),(5,5),(8,8)

Flight point 4: 20,000 ft/Mach 0.5, reverse Bode:

MPC update rate (minor frames per MPC update): 1
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(2,5),(2,8),(3,3),(3,8),(5,8)
Cases nearly agreeing: (2,3),(3,5),(5,5),(8,8)
Cases with significant differences: None

MPC update rate (minor frames per MPC update): 2
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8)
Cases nearly agreeing: (2,2),(2,5),(3,3),(3,5),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: (2,8)

MPC update rate (minor frames per MPC update): 3
Cases with exact agreement: (1,1),(1,2),(1,3),(3,3)
Cases nearly agreeing: (1,5),(1,8),(2,2),(2,3),(2,5),(2,8),(3,5),(5,5),(5,8),(8,8)
Cases with significant differences: (3,8)

MPC update rate (minor frames per MPC update): 5
Cases with exact agreement: (1,2),(1,5),(1,8),(2,2)
Cases nearly agreeing: (1,1),(1,3),(2,5),(2,8),(3,3),(3,5),(3,8),(5,8)
Cases with significant differences: (2,3),(5,5),(8,8)

MPC update rate (minor frames per MPC update): 7
Cases with exact agreement: (1,1),(1,2)
Cases nearly agreeing: (1,3),(1,5),(1,8),(2,2),(2,3),(2,5),(3,3),(3,5),(5,5),(5,8)
Cases with significant differences: (2,8),(8,8)

MPC update rate (minor frames per MPC update): 10
Cases with exact agreement: (1,1),(1,2),(1,5),(3,3)
Cases nearly agreeing: (1,3),(1,8),(2,2),(2,3),(2,5),(2,8),(3,5),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: None

Flight point 5: 35,000 ft/Mach 0.84, reverse Bode:

MPC update rate (minor frames per MPC update): 1
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(2,5),(2,8),(3,3),(3,5)
Cases nearly agreeing: (2,3),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: None

MPC update rate (minor frames per MPC update): 2
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2),(3,3)
Cases nearly agreeing: (2,5),(2,8),(3,5),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: (5,8)

MPC update rate (minor frames per MPC update): 3
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5),(1,8),(2,2)

Cases nearly agreeing: (2,3),(2,5),(2,8),(3,3),(3,5),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: None

MPC update rate (minor frames per MPC update): 5
Cases with exact agreement: (1,1),(1,2),(1,3),(2,2),(3,3)
Cases nearly agreeing: (1,5),(1,8),(2,3),(2,5),(2,8),(3,5),(3,8),(5,5),(5,8)
Cases with significant differences: (8,8)

MPC update rate (minor frames per MPC update): 7
Cases with exact agreement: (1,1),(1,2),(1,3),(1,5)
Cases nearly agreeing: (1,8),(2,2),(2,3),(2,5),(3,5),(3,8),(5,5),(5,8),(8,8)
Cases with significant differences: (2,8)

MPC update rate (minor frames per MPC update): 10
Cases with exact agreement: (1,1),(1,8),(2,8),(3,3)
Cases nearly agreeing: (1,2),(1,3),(1,5),(1,8),(2,3),(2,8),(3,5),(3,8),(5,5),(5,8)
Cases with significant differences: (2,5),(8,8)

As we can see from these results, the use of only the three minimum stall margin constraints in the MPC optimization produces, in many cases, nearly identical results to MPC that accounts for all of the constraints. But these results are obtained with a significant 30% savings (or thereabouts) in computational effort for MPC. The only exception is the SLS, takeoff-to-idle transient, where significant differences in many of the variables occur when the constraints are reduced to just the three minimum stall margin limits. The results at this flight point should be examined further, but time does not permit this examination s part of this study.

5. Conclusions/Further Work

We have studied the application of the baseline MPC algorithm to the control of main fuel flow rate (WF36), variable bleed valve (AE24) and variable stator vane (STP25) control of a simulated high-bypass turbofan engine. Using reference trajectories for thrust and turbine inlet temperature (T41) generated by a simulated new engine, we have examined MPC for tracking these two reference outputs while controlling a deteriorated engine. We have examined the results of MPC control for six different transients: two idle-to-takeoff transients at sea level static (SLS) conditions, one takeoff-to-idle transient at SLS, a Bode power command and reverse Bode power command at 20,000 ft/Mach 0.5, and a reverse Bode transient at 35,000 ft/Mach 0.84.

For all cases, our primary focus was on the computational effort required by MPC for varying MPC update rates, control horizons, and prediction horizons. We have also considered the effects of these MPC parameters on the performance of the control, with special emphasis on the thrust tracking error, the peak T41, and the sizes of violations of the constraints on the problem, primarily the booster stall margin limit, which for most cases is the lone constraint that is violated with any frequency.

Based on our studies, we have concluded the following:

1) The computational effort decreases nearly linearly with increasing numbers of minor simulation time frames per MPC update. Thus, updating 1/5 as frequently typically yields computational effort of about 20% of that required to update every minor frame. However, slower MPC updating typically results in larger thrust tracking errors or more frequent or larger stall margin limit violations.

2) The MPC control horizon has a significant impact on the computational effort, typically requiring 25% to 35% more computational effort for each single MPC cycle increase in the control horizon. This argues for control horizons that are short, like a single MPC cycle, for the least computational effort. The effect of control horizon on the performance of MPC varies by case. In some cases, the shortest control horizon gave the best performance. In others, it did not.

3) The MPC prediction horizon has little impact on the computational effort, typically requiring just a few percent more computational effort for each single MPC cycle increase in the prediction horizon. Furthermore, for many cases, the MPC controller's ability to track the thrust demands is increased by using longer prediction horizons. In some cases, there appears to be an "optimum" prediction horizon in terms of performance. But this is not the case often enough to conclude that there is always an optimum prediction horizon.

4) The performance of MPC is affected little by reducing the number of constraints considered to just those that are occasionally effective (primarily the stall margins on fan, booster and compressor here). While dropping some constraints from consideration reduced the computational effort for MPC by about 35%, the performance is identical or very nearly so in many cases to the results when all constraints are considered. However, there are a few cases where significant differences occur.

Further work

There are many topics that could still be pursued related to MPC for this application. Among these are the following:

1) We conducted a very brief study of what happens if T41 is not included as a tracked variable. Our preliminary results indicated that substantial thrust tracking errors and frequent violations of constraint limits occur. This should be investigate further, since the T41 histories obtained from the deteriorated engine simulation to which MPC was applied were always higher than the T41 for the new engine that generated the reference trajectories.

2) All of our studies were conducted on the same deteriorated engine. It is relatively easy to generate simulations of other deteriorated engines, and the MPC results we show here should be verified against other deteriorated engines.

3) Our simulation studies used the same deteriorated engine for both the true engine and the model engine that is linearized for the application of MPC. In real applications, the model engine would not agree with the true engine. The sensitivity of the MPC performance to this mismatch should be studied.

4) A more thorough analysis of the computational effort required by MPC would involve better measures of the CPU time, and would include the effects of overhead operations like the memory storage required and how frequently it must be accessed. This was beyond the scope of our study.

5) We have not come up with an approach to "spreading out" the MPC computations over several minor frames in cases where the MPC updates are calculated less frequently than every minor frame.

6) In light of the computational savings from using a reduced set of constraints, we suggest that an "adaptive" approach to treating the constraints be considered. In other words, at any flight point, only those constraints that are occasionally "active" at that flight point should be included for consideration. And the transient behavior of the engine variables themselves can be used to predict when a constraint may become active, and therefore must be considered. This would involve making the MPC algorithm "adapt" to the changing number of constraints that must be considered.

7) We have made no effort in this study to examine the effects of the weights that were specified in the baseline MPC study. The effect of these on the MPC performance should also be considered.

Reference

[1] A. Kumar and D. Viassolo, "Advance Propulsion Systems Technology – Final Report," final report by GE Global Research Center to NASA on Contract NAS3-01135, January, 2004.

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-06-2008	Final Contractor Report	April 2005-April 2006

4. TITLE AND SUBTITLE
Intelligent Engine Systems
Adaptive Control

5a. CONTRACT NUMBER
NAS3-01135

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
2.1

6. AUTHOR(S)
Gibson, Nathan

5d. PROJECT NUMBER

5e. TASK NUMBER
37

5f. WORK UNIT NUMBER
WBS 984754.02.07.03.11.03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
General Electric Aircraft Engines
One Neumann Way
Cincinnati, Ohio 45215

8. PERFORMING ORGANIZATION REPORT NUMBER
E-16499

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORS ACRONYM(S)
NASA

11. SPONSORING/MONITORING REPORT NUMBER
NASA/CR-2008-215240

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Categories: 01 and 06
Available electronically at http://gltrs.grc.nasa.gov
This publication is available from the NASA Center for AeroSpace Information, 301-621-0390

13. SUPPLEMENTARY NOTES

14. ABSTRACT
We have studied the application of the baseline Model Predictive Control (MPC) algorithm to the control of main fuel flow rate (WF36), variable bleed valve (AE24) and variable stator vane (STP25) control of a simulated high-bypass turbofan engine. Using reference trajectories for thrust and turbine inlet temperature (T41) generated by a simulated new engine, we have examined MPC for tracking these two reference outputs while controlling a deteriorated engine. We have examined the results of MPC control for six different transients: two idle-to-takeoff transients at sea level static (SLS) conditions, one takeoff-to-idle transient at SLS, a Bode power command and reverse Bode power command at 20,000 ft/Mach 0.5, and a reverse Bode transient at 35,000 ft/Mach 0.84. For all cases, our primary focus was on the computational effort required by MPC for varying MPC update rates, control horizons, and prediction horizons. We have also considered the effects of these MPC parameters on the performance of the control, with special emphasis on the thrust tracking error, the peak T41, and the sizes of violations of the constraints on the problem, primarily the booster stall margin limit, which for most cases is the lone constraint that is violated with any frequency.

15. SUBJECT TERMS
Update rate; Control horizon; Prediction horizon

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email:help@sti.nasa.gov)
U	U	U	UU	77	19b. TELEPHONE NUMBER *(include area code)* 301-621-0390