Model Based Optimal Sensor Network Design for Condition Monitoring in an IGCC Plant

Final Technical Report

Reporting Period: October 2010 – December 2012

Principal Author: Rajeeva Kumar

Issued: March 2013

DOE Award Number: DE-FE0005712

Submitted by:
GE Global Research
One Research Circle
Niskayuna NY 12309

List of Authors

Aditya Kumar
Dan Dai
Gayathri Seenumani
John H Down
Rajeeva Kumar
Rodrigo Lopez-Negrete de la Fuente

GE Global Research
One Research Circle
Niskayuna NY 12309

“This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.”

Table of Contents

1 ABSTRACT 8

2 EXECUTIVE SUMMARY 9

2.1 TASK 2 – MODEL ENHANCEMENT AND TECHNOLOGY REVIEW 9
2.2 TASK 3 – DEVELOPMENT OF OSP ALGORITHM 9
2.3 TASK 4 – DEMONSTRATION OF OSP ALGORITHM PERFORMANCE 10

3 INTRODUCTION 10

4 TASK 2: MODEL ENHANCEMENT AND TECHNOLOGY REVIEW 14

4.1 ELEMENTS OF A MODEL BASED OSP ALGORITHM 15
4.1.1 Network Requirements 16
4.1.2 Estimation Problem 16
4.1.3 Optimization Problem 18
4.1.3.1 Precision Constraint 18
4.1.3.2 Reliability Constraint 18
4.2 TECHNOLOGY REVIEW: INTEGER PROGRAMING ALGORITHMS FOR OSP PROBLEM 21
4.2.1 Branch-and-Bound 22
4.2.1.1 Branch and Bound – steps 23
4.2.1.2 Integer relaxation - NLP Relaxation 25
4.2.1.3 Lagrangian Relaxation 26
4.2.1.4 SDP Relaxation 29
4.2.1.4.1 SDP and Integer 0/1 (Binary) Programming 30
4.2.2 Generalized Benders Decomposition and Outer Approximation Algorithms 32
4.2.2.1 Primal Problem 33
4.2.2.2 Master Problem 34
4.2.3 Outer Approximation with Feasibility Pump 37
4.2.4 LMI Formulation for OSP Problem 40
4.2.4.1 Linear Matrix Inequality and ARE Problem 40
4.3 MODEL ENHANCEMENT FOR GASIFIER REFRACTORY WEAR 44
4.4 MODEL ENHANCEMENT FOR RSC FOULING 50

5 TASK 3: DEVELOPMENT OF OSP ALGORITHM 62

5.1 SIMPLIFIED GASIFIER REFRACTORY MODEL FOR OSP IMPLEMENTATION 62
5.1.1 Monitoring requirement for the simplified gasifier OSP problem 63
5.2 IMPLEMENTATION CHALLENGES FOR ESTIMATION RELIABILITY USING BRANCH AND BOUND BASED APPROACH 67
5.2.1 Approach 1 using NLPRF 68
5.2.2 Approach 2 using NLPRP 70
5.3 PROPOSED OUTER APPROXIMATION BASED APPROACH FOR INLP 72
5.4 LMI BASED APPROACH 73
5.4.1 LMI solver evaluation 74
5.4.2 Discussion of LMI implementation 74
5.5 RESULTS OF OA-INLP ON 1D GASIFIER MODEL 75
5.6 MODIFICATIONS TO BB AND OA FRAMEWORK FOR RELIABILITY CONSTRAINT 75
5.6.1 Modification to OSP Formulation for Handling Functional Reliability Constraints 77
5.6.2 Modification to OSP Formulation for Handling Estimation Reliability Constraints 78
5.6.2.1 Implementation Considerations 78
5.6.3 Results 79
5.6.4 Analytical Gradient Computation 82
5.7 PRACTICAL CONSIDERATIONS: CONSTRAINT SOFTENING FOR OSP PROBLEM 84

6 TASK 4: DEMONSTRATION OF OSP ALGORITHM PERFORMANCE 87

6.1 SOLUTION TO OSP PROBLEM FOR CONDITION MONITORING OF GASIFIER REFRACTORY WEAR AND RSC FOULING 88
6.1.1 Reduced Size Gasifier Model for the OSP Algorithms 88
6.1.2 Optimal Sensor Placement Results for Reduced Order 3D Model 92
6.1.2.1 Analysis and Remarks 96

6.1.3 OSP Solution to the Detailed 3D Gasifier Model *98*
6.2 OPTIMAL SENSOR PLACEMENT FOR CONDITION MONITORING OF RSC FOULING 99
6.3 ASSESSING THE PERFORMANCE OF THE OSP ALGORITHM 104
6.3.1 Assessing solution to OSP problem for CM of gasifier refectory wear *105*
6.3.2 Assessing solution to OSP problem for CM of RSC fouling profile *109*

7 CONCLUSION **110**

7.1 MODEL ENHANCEMENT AND TECHNOLOGY REVIEW 111
7.2 DEVELOPMENT OF OSP ALGORITHM 113
7.3 DEMONSTRATION OF OSP ALGORITHM PERFORMANCE 114

8 REFERENCES **116**

Appendix A: Modular Tools **118**
A.1 Modular Extended Kalman Filter block in Matlab/Simulink© 118
A.1.1 Input and Output Signals 119
A.1.2 User Supplied Models 120
A.1.3 EKF Estimator Model 124
A.1.4 EKF Block Operation 129

A.2 Modular OSP Tool in Matlab© 131
A.2.1 Input Data 132
A.2.2 Output Data 133
A.2.3 Software Package Needed for the OSP Tool 134

List of Figures

Figure 1: Program overview - team, scope and objectives 11
Figure 2: Model based optimal sensor placement framework 12
Figure 3: Condition monitoring for gasifier refractory wear 14
Figure 4: Condition monitoring of RSC tube fouling 15
Figure 5: Schematic of model based approach for solving an OSP Problem 16
Figure 6: A full enumeration tree in BB algorithm with two integer variables (Chinneck 2010) 23
Figure 7 : Schematic of a Branch and Bound Algorithm using NLP relaxation 25
Figure 9: Schematic of the Integrated Feasibility Pump algorithm. 39
Figure 10: Comparison chart of existing MINLP methods with respect to different relaxations. 40
Figure 11: Typical model based sensing scheme 44
Figure 12: Non-uniform wear on hot inner surface of gasifier refractory lining 45
Figure 13: Nodes for 3-dimensional heat transfer model in gasifier refractory lining. 45
Figure 14: Example non-uniform/local wear profile on inner refractory brick layer. 47
Figure 15: Temperature profiles at brick interfaces and metal shell for local wear profile. 48
Figure 16: Transient temperature profiles across radial nodes at different axial and azimuthal locations. 49
Figure 17: Sparsity pattern of linear model for gasifier refractory. 50
Figure 18: Fitting the RSC model heat fluxes to CFD data by parameterizing the heat transfer coefficients for the un-fouled RSC tubes in case of: NOC, 80% throughput, and 50% throughput. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the percent error in the heat flux fits. The heat flux trends shown here are for a generic model and do not represent any specific site data. 51
Figure 19: Fitting the RSC model heat fluxes to CFD data by parameterizing the heat transfer coefficients for the fully fouled RSC tubes in case of: NOC, 80% throughput, and 50% throughput. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the percent error in the heat flux fits. The heat flux trends shown here are for a generic model and do not represent any specific site data. 52
Figure 20: Effect of fouling on syngas temperature profiles inside the RSC. Fully fouled, half fouled, and no fouling cases have uniform fouling while bottom fouled and top fouled cases have non-uniform fouling along the RSC. 1st node corresponds to the top of the RSC and the 30th node corresponds to the bottom of the RSC. 53
Figure 21: Effect of fouling on the amount of steam flow from the RSC drum. 54
Figure 22: Effect of fouling on the strain on the RSC bolt. 54
Figure 23: The fouling profiles used in simulation. 55
Figure 24: The comparison of the RSC model heat fluxes between linearized model and nonlinear model for un-fouled and small non-uniform fouling (10%) at different zones. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the error in term of deviation for the linear model from the nonlinear model. 56
Figure 25: The comparison of the RSC model heat fluxes between linearized model and nonlinear model for non-uniform half fouled zones. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the error in term of deviation for the linear model from the nonlinear model. 57
Figure 26: The comparison of the effect of fouling on the exit syngas temperature on the RSC bolt between linearized model and nonlinear model for small non-uniform fouling (10%) at different zones. 58
Figure 27: The comparison of the effect of fouling on the exit syngas temperature on the RSC bolt between linearized model and nonlinear model for non-uniform half fouled zones. The bottom right plot shows the error in term of deviation for the linear model from the nonlinear model. 59
Figure 28: The comparison of the effect of fouling on the strain on the RSC bolt between linearized model and nonlinear model for non-uniform half fouled zones. The bottom right plot shows the error of strain in term of deviation for the linear model from the nonlinear model. 60
Figure 29: The comparison of the effect of fouling on the exit syngas temperature on the RSC bolt between linearized model and nonlinear model for small non-uniform fouling (10%) at different zones. 61
Figure 30: Sensor Cost Figure 31: Sensor Failure Rate 64
Figure 32: Approximation to indicator function. 66
Figure 33: Smoother approximation of the Indicator function 69
Figure 34: Tight approximation of the Indicator function 70

Figure 35: BB iterates for approach 2, MIP: (ZRU) Upper bound when reliability and precision are satisfied, MIP_prec: ((ZPU) Upper bound when ONLY precision is satisfied, NLP: Solution of NLPRP problem 71
Figure 36: Schematic of OA-INLP method 72
Figure 37: Branch-and-Bound Iteration result with LMI relaxation 73
Figure 38: BB method functional reliability 79
Figure 39: OA method functional reliability 80
Figure 40: BB implementation of estimation reliability 81
Figure 41: OA implementation for estimation reliability 81
Figure 42: Hot surface of the 3-D refractory model 88
Figure 43: The refractory wear outside the block affects the state derivatives with in the block. That effect can be captured through the process noise model of the block boundary nodes. 89
Figure 44: Maximum temperature derivatives for each boundary nodes of the smaller block *abcd,* for various wear scenarios within complete gasifier *ABCD*, are recorded in simulation. The magnitude of the process noise (3σ) for the boundary nodes are increased by the noise magnitude thus obtained. 90
Figure 45: Comparing precision of the wear estimates obtained from a small wedge block of the gasifier to that obatined from the full gasifier 91
Figure 46: Solve the OSP problem for the block considered by imposing CM requirement constraints only on part of it. 92
Figure 47: Extend the OSP solution obtained using smaller block to obtain OSP solution for complete refractory gasifier by repeating the solution. 92
Figure 48: Hypothetical Sensor Cost 94
Figure 49: Sensor Reliability 94
Figure 50: OSP Results for $s \leq .044 \mathrm{tbrick.r} \geq 0.9$ 94
Figure 51: OA bound generation 95
Figure 52: Wear Estimate Precision of the innermost surface 95
Figure 53: Sensor network for case 1 96
Figure 54: Sensor network for case 2. 96
Figure 55: Comparison of the OSP solution with cases 1 - 5 97
Figure 56: Cost, precision comparison of OSP solution. 97
Figure 57: Node 16 (Interface 1) sensor placement 99
Figure 58: Heat flux sensor location vs. sensor cost 100
Figure 59: Heat Flux sensor reliability vs. sensor location 101
Figure 60: Branch-and-Bound Iteration result with NLP relaxation: precision only 103
Figure 61: Branch-and-Bound Iteration result with NLP relaxation: unequal cost, precision and reliability 104
Figure 62: Randomly distributed refractory wear 105
Figure 63: Refractory wear profile 106
Figure 64: Wear estimate through model based sensing 106
Figure 65: Distribution of the mean wear estimation error 107
Figure 66: Precision constraint obtained from simulation based on 24600 wear estimates 108
Figure 67: Wear estimate when few sensor from the solution to the OSP problem are moved further away 108
Figure 68: Fouling profiles and fouling parameters estimates 109
Figure 69: Distribution of mean estimation error for RSC fouling parameters 110
Figure 70: EKF Simulink - top level block 118
Figure 71: Standard Interface Blocks for User Defined Models 121
Figure 72: User Defined Plant Model as an Embedded Matlab Block 122
Figure 73: User Defined Auxiliary Model as a Simulink Diagram 123
Figure 74: Propagate and Update Functions within the EKF Block 129
Figure 75: EKF Model Linearization Block 130

1 Abstract

This report summarizes the achievements and final results of this program. The objective of this program is to develop a general model-based sensor network design methodology and tools to address key issues in the design of an optimal sensor network configuration: the type, location and number of sensors used in a network, for online condition monitoring. In particular, the focus in this work is to develop software tools for optimal sensor placement (OSP) and use these tools to design optimal sensor network configuration for online condition monitoring of gasifier refractory wear and radiant syngas cooler (RSC) fouling. The methodology developed will be applicable to sensing system design for online condition monitoring for broad range of applications. The overall approach consists of (i) defining condition monitoring requirement in terms of OSP and mapping these requirements in mathematical terms for OSP algorithm, (ii) analyzing trade-off of alternate OSP algorithms, down selecting the most relevant ones and developing them for IGCC applications (iii) enhancing the gasifier and RSC models as required by OSP algorithms, (iv) applying the developed OSP algorithm to design the optimal sensor network required for the condition monitoring of an IGCC gasifier refractory and RSC fouling. Two key requirements for OSP for condition monitoring are desired precision for the monitoring variables (e.g. refractory wear) and reliability of the proposed sensor network in the presence of expected sensor failures. The OSP problem is naturally posed within a Kalman filtering approach as an integer programming problem where the key requirements of precision and reliability are imposed as constraints. The optimization is performed over the overall network cost. Based on extensive literature survey two formulations were identified as being relevant to OSP for condition monitoring; one based on LMI formulation and the other being standard INLP formulation. Various algorithms to solve these two formulations were developed and validated. For a given OSP problem the computation efficiency largely depends on the “size” of the problem. Initially a simplified 1-D gasifier model assuming axial and azimuthal symmetry was used to test out various OSP algorithms. Finally these algorithms were used to design the optimal sensor network for condition monitoring of IGCC gasifier refractory wear and RSC fouling. The sensors type and locations obtained as solution to the OSP problem were validated using model based sensing approach. The OSP algorithm has been developed in a modular form and has been packaged as a software tool for OSP design where a designer can explore various OSP design algorithm is a user friendly way. The OSP software tool is implemented in Matlab/Simulink© in-house. The tool also uses few optimization routines that are freely available on World Wide Web. In addition a modular Extended Kalman Filter (EKF) block has also been developed in Matlab/Simulink© which can be utilized for model based sensing of important process variables that are not directly measured through combining the online sensors with model based estimation once the hardware sensor and their locations has been finalized. The OSP algorithm details and the results of applying these algorithms to obtain optimal sensor location for condition monitoring of gasifier refractory wear and RSC fouling profile are summarized in this final report.

2 Executive Summary

This section summarizes the key accomplishments of the program. The overall two-year program consisted of three main technical tasks. Each key task is described next in more detail.

2.1 Task 2 – Model Enhancement and Technology Review

The objectives in this Task were to (i) define condition monitoring requirement and OSP evaluation metrics, (ii) enhancing the available gasifier and RSC models for use in OSP calculation, and (iii) analyzing trade off of alternate OSP algorithms. First, high-level condition-monitoring requirements relevant to the gasifier refractory degradation and RSC fouling monitoring were identified. These requirements were then transformed into design requirements for the OSP algorithms. There are two key metrics important for the accessing any OSP solution: (i) accuracy/precision of key process variables/parameters being monitored and (ii) the reliability of the sensor network in the presence of probabilistic failure of sensors. Thereafter, available dynamic models for the gasification section were extended to be consistent with the condition monitoring requirements. More specifically, the gasifier model was extended to include a transient 3-D thermal model of the refractory lining to relate the effects of hot surface wear on potential temperature sensors placed in the refractory lining. This will enable a model-based estimation of the unknown wear through the potential temperature sensor measurements. Similarly, the RSC model was extended to include a 1-D fouling variation along the length of the RSC and its effects on potential sensors like heat flux, temperatures, and strain. In parallel, existing literature on sensor placement was reviewed. The solution to OSP essentially requires decision about the number, type and location of the hardware sensors. This naturally leads to integer nonlinear programming (INLP) optimization problems, which are computationally expensive to solve due to the combinatorial explosion of integer search space, especially for larger-scale OSP problems (e.g. the full 3-dimensional gasifier refractory monitoring problem). Available methods for solving such INLP problems and appropriate relaxations were reviewed. This highlighted the need to develop a suite of tools that can be used for small, medium or large-scale OSP problems.

2.2 Task 3 – Development of OSP Algorithm

The objective of this task was to develop and implement all the essential elements of an OSP algorithm: (i) model-based estimation algorithm, and (ii) optimization algorithm for integer nonlinear programming problem. At the core of the optimal sensor network design is a model based estimator that estimates the variables important for condition monitoring and controls using various process measurements from available/potential sensors. This estimator is wrapped within an optimization framework that optimizes the number, type and location of the hardware sensors. Any model-based sensor network design methodology must account for common sources of modeling errors as well as sensing inaccuracies (noise, bias). This motivates the formulation of OSP problem using Kalman filtering techniques for estimation and coupling it with various integer optimization algorithms to compute the optimal sensor network configuration. A steady state Kalman filter based estimation algorithm is used for all OSP algorithms. One of the two key metrics that OSP

algorithms are designed around namely desired precision, can readily be expressed through the steady state error covariance matrix in Kalman filter formulation. The second metric namely the reliability deals with meeting the precision requirements with a desired level of certainty in the presence of probabilistic failures of the sensors used in the network. This requirement leads to combinatorial computation that is prohibitively expensive even for small number of sensors. Practical approximations are made to include the reliability requirement within the OSP problem formulation. Various algorithms were developed to solve the optimal sensor placement problems as integer nonlinear optimization problem treating the reliability and precision requirements as constraints. These algorithms have varying level of computational complexities and are well suited for specific OSP problem based on the “size” of the problem. To facilitate faster design and development of the OSP algorithms, a simplified 1-D gasifier model assuming axial and azimuthal symmetry were used. In addition to the modular OSP algorithm, a modular Extended Kalman Filter (EKF) block was also developed in Matlab/Simulink©. This EKF block can be utilized for model based sensing of important process variables that are not directly measured through combining the online sensors with model based estimation once the hardware sensor and their locations has been finalized. Both, the modular OSP algorithm tool as well as modular EKF block are going to be delivered to DOE at the end of the project.

2.3 Task 4 – Demonstration of OSP Algorithm Performance

The objective of this task was to apply the fully developed OSP algorithm of Task 3 to gasifier and RSC system models to design cost optimal sensor networks in terms of number, type and location of hardware sensors required for condition monitoring of the gasifier section of an IGCC plant. In this task, a cost optimal sensor network is designed to online condition monitor of gasifier refractory wear. Similarly another cost optimal sensor network is designed for online monitoring of radiant syngas cooler fouling. The quality of the performance to the OSP algorithm is tested through Monte Carlo simulation studies. More specifically, in these studies a model based sensing technique using Kalman filter was used to monitor randomly generated multiple cases of gasifier refractory through the sensor set obtained using the OSP algorithm. The ensemble variance of wear estimates obtained in these simulation studies statistically match the precision for which the optimal sensor network was designed. Similar studies were done for estimating the fouling in RSC tubing with randomly generated RSC fouling profiles. Here again the ensemble variance of fouling estimates obtained in these simulation studies statistically match the precision for which the optimal sensor network was designed.

3 Introduction

Integrated Gasification Combined Cycle (IGCC) technology holds the promise for clean and efficient power generation from coal and is currently being commercialized by GE. A key challenge in accelerating its commercialization is to increase IGCC plant reliability and availability. The critical components affecting IGCC plant availability are gasifier and radiant syngas cooler (RSC) in the core gasification section. The core gasification section

of the plant has a particularly harsh environment with high temperatures (1200-1600 ^{o}C) and pressures (exceeding 40atm) and presence of slag and corrosive elements which lead to significant degradation of the health/condition of key process equipment like the gasifier and the radiant syngas cooler (RSC). Currently, in the absence of online monitoring, the plant is operated conservatively and is shut down periodically for manual inspection and maintenance of the gasifier refractory lining. This adversely impacts plant availability and efficiency. Similarly, the heat transfer performance in the RSC degrades over time due to fouling buildup from ash and slag, which also affects the plant efficiency. Owing to the harsh environment, very limited sensing is currently available for online monitoring and controls in this section. An online condition monitoring of these core component is key to improving overall plant efficiency and availability.

This program focused at developing a systematic model-based computational solution for optimal sensor placement (OSP) in a sensor network used for online condition monitoring of these key process equipment in an IGCC plant. The overall program scope, objective and the team is shown in Figure 1

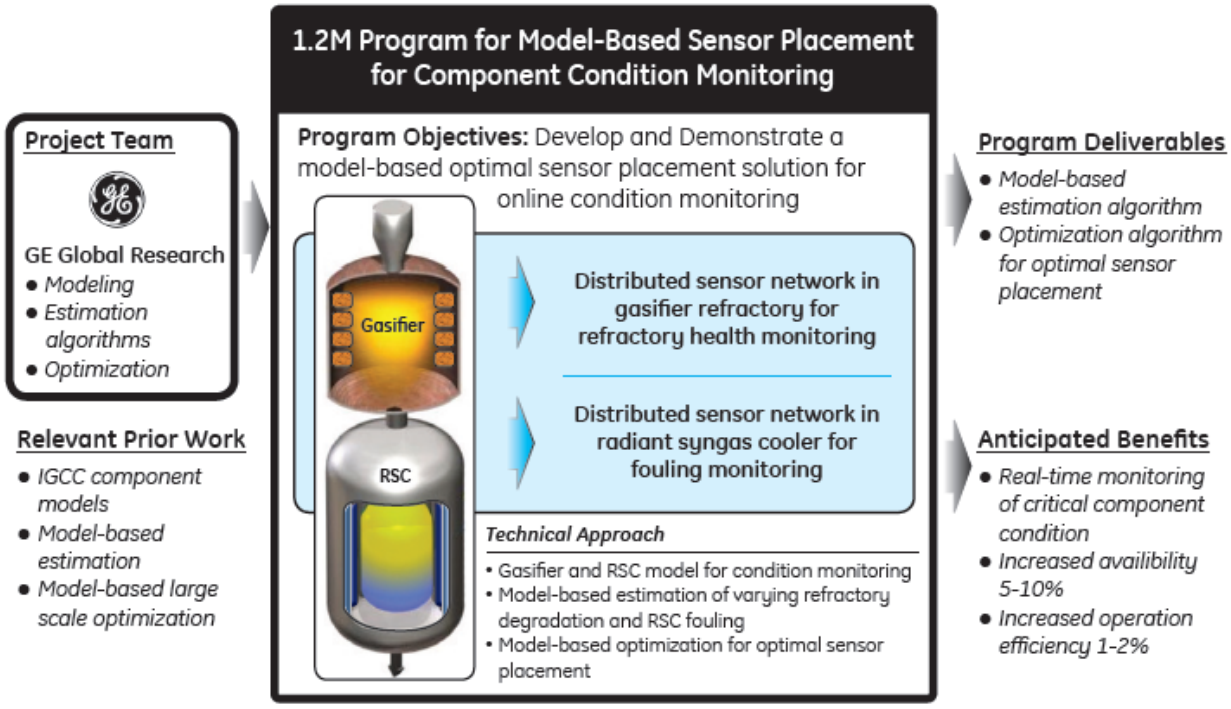


Figure 1: Program overview - team, scope and objectives

In this two-year program, a systematic model-based approach was developed for addressing the problem of optimal sensor placement for condition monitoring of the gasifier refractory lining and RSC fouling, that combined a model based estimator with a model based optimizer. The overall model based optimal sensor design framework is shown in Figure 2. In this regard, initially available first-principles physics-based models were extended. These models describe the effect of refractory wear and RSC fouling on the overall process dynamics and potential sensors that could be used in the sensor network for online condition monitoring. Thereafter, nonlinear model-based estimation algorithm

was coupled with nonlinear integer optimization algorithm to develop a sensor network solution with optimal combination of sensors that allows a robust online monitoring of the equipment health, in the presence of modeling and sensor measurement errors as well as probability of sensor failure in the harsh environment.

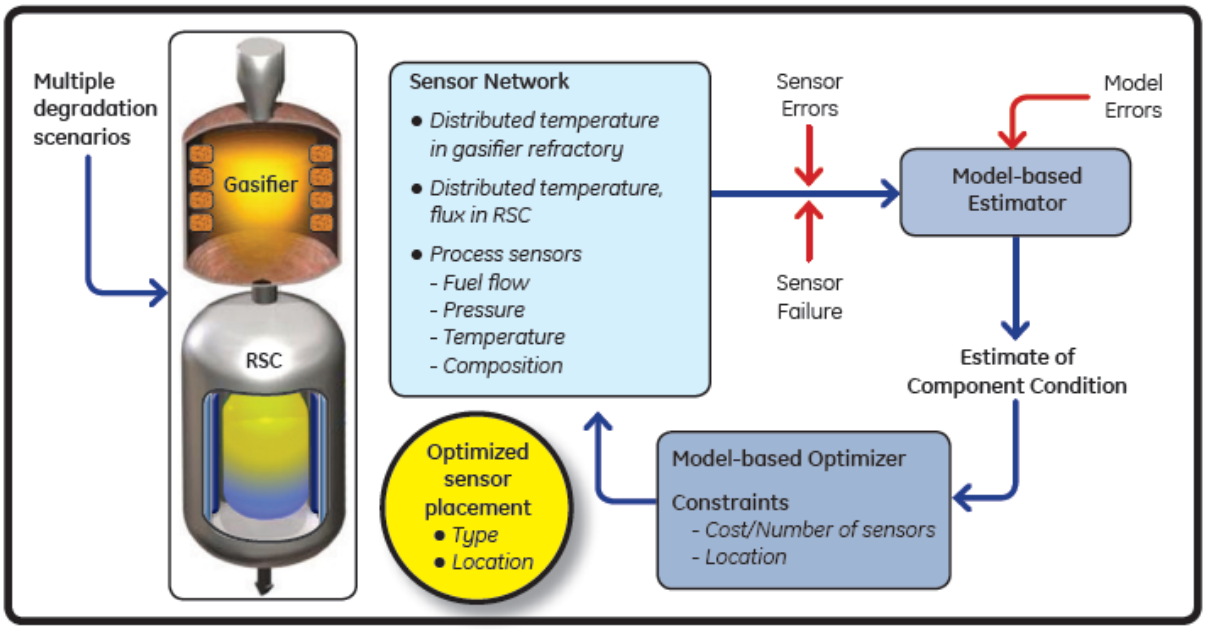


Figure 2: Model based optimal sensor placement framework

The overall two-year program consists of three main technical tasks:

- Task 2: Model enhancement and technology review,
- Task 3: Development of OSP algorithm, and
- Task 4: Demonstration of OSP algorithm performance.

In Task 2, the existing literature for optimal sensor placement was reviewed to identify available solutions and gaps for practical design of optimal sensor network that can systemically account for sensor characteristics and model error. This review is summarized in Section 4. Then high-level requirements for condition monitoring of the gasifier refractory degradation and RSC fouling were identified. Based on these requirements, the OSP problem was formulized. These requirements are described in Section 4. Based on these required the OSP problem are formulized as described in Section 4. At the core of the optimal sensor network design is a model based estimator that estimates the variables important for condition monitoring using various process measurements. This estimator is wrapped within an optimization framework that optimizes the number, type and location of the hardware sensors. One of the subtasks in Task 2 focused on analyzing alternate OSP problem formulations, and associated estimation and optimization algorithms to be implemented for the gasification section as well as be applicable to more general applications. Based on extensive literature survey two formulations were identified as being relevant to OSP for condition monitoring; one based on LMI formulation and the other being standard INLP formulation.

The available techniques for solving LMI and Integer Programming Problem were reviewed to identify relative merits and demerits of various options. The algorithms to solve the integer problems suited for OSP application can be classified in two major subclasses. One class of solution is based on branch and bound techniques and the other on outer approximations techniques. A brief summary of the review is presented in Section 4. Based on the literature review, a few promising options were extended and adapted for pursuing in this program.

The OSP algorithms are finally used to design optimal sensor network for condition monitoring of gasifier refractory wear and RSC fouling. Hence, available dynamic models for the gasification section were extended to be consistent with the condition monitoring requirements. More specifically, the gasifier model was extended to include a transient 3-D thermal model of the refractory lining that relates the effects of hot surface wear on potential temperature sensors placed in the refractory lining. This is a prerequisite for a model-based estimation of unknown wear through the temperature sensor measurements. Similarly, the RSC model was extended to include a 1-D fouling variation along the length of the RSC and its effects on potential sensors like heat flux, temperatures, and strain used for online monitoring. The model enhancements for the gasifier and RSC are described in Section 4.

Section 5 presents the Task 3 related work. In Task 3, the model-based estimation algorithm and nonlinear optimization algorithms were implemented in a modular fashion. These are tested for individual performance, and then integrated to achieve the overall OSP solution within a software tool meant for solving an optimal sensor placement problem. As mentioned earlier the computation efficiency of OSP algorithm depends largely on the size of the problem. Algorithms which are computationally efficient for a small size problem are not at all scalable for large size problems. Within this software OSP tool one can choose various algorithms for solving an OSP problem based on the size of the problem.

Section 6 presents the Task 4 related work. In Task 4, the implemented OSP algorithm are tested against the gasification section model to demonstrate its performance in condition monitoring of the gasifier refractory wear and RSC fouling. More specifically, the OSP algorithm was integrated with the enhanced gasification section model from Task 2, and the performance of the OSP algorithm are tested through simulation studies for multiple cases of random gasifier refractory wear and RSC fouling profiles.

There are two key software deliverables from this program: modular EKF block in Matlab/Simulink© and modular OSP algorithms. These modular tools are described in Section 7.

Finally, conclusion from the overall program and potential directions for application of the developed technology are summarized in Section 8.

4 Task 2: Model Enhancement and Technology Review

The core gasification section of the plant has a particularly harsh environment. It operates in high temperatures, pressures and presence of slag and corrosive elements that lead to significant degradation of the health/condition of key process equipment like the gasifier and the radiant syngas cooler (RSC).

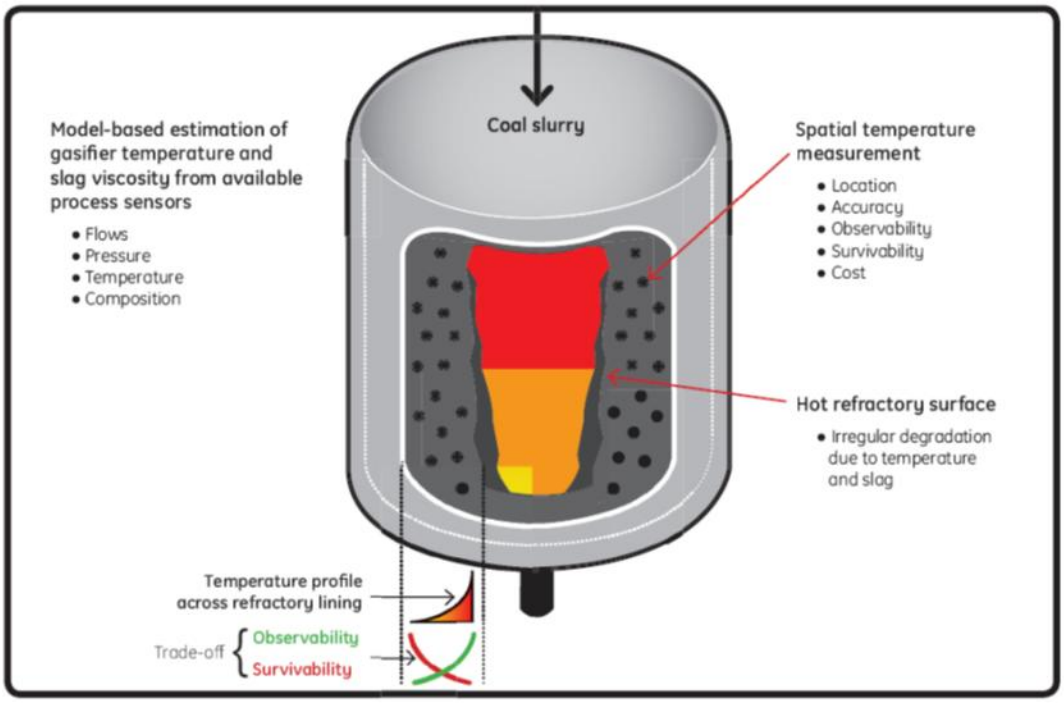


Figure 3: Condition monitoring for gasifier refractory wear

The gasifier has a refractory lining with multiple layers of refractory bricks and an outer metal shell. The refractory lining provides thermal insulation, and thus, protection for the metal shell. Over time, due to severe operating conditions inside the gasifier, the hot inner surface of the refractory lining wears out, i.e., the refractory lining thickness degrades thereby leading to higher temperatures in the lining and metal shell. The wear is distributed non-uniformly over the entire hot inner surface. This program focused on monitoring the wear on the main cylindrical section of the refractory lining, and to track the extent of wear as the inner-most brick worn out as a function of axial and azimuthal location on the cylindrical section. The temperature sensors located spatially in between the brick layers, provide signature which along with the thermal model of the gasifier refractory are used to measure the wear profile in the gasifier inner layer. The objective of the OSP algorithm is to identify the number and location of these temperature sensors at each of the brick layers. In particular, there is a trade-off in “observability” and “survivability” of the sensors (Figure 3). In particular, it is expected that temperature sensors located closer to the inner brick layers will provide highest sensitivity, and thus, highest observability of the hot surface wear profile. In contrast, sensors located at the interface of further away brick

layers will provide lower observability. On the other hand, due to the higher temperatures, the temperature sensors at the first brick interface will have relatively lower survivability.

Figure 4: Condition monitoring of RSC tube fouling

The RSC has heat exchange tubes (Figure 4), get fouled over time due to ash/slag deposition, thereby reducing the heat transfer and steam generation capability. This fouling profile is expected to be non-uniform along the length of the RSC. To monitor this change in axial fouling profile a combination of existing sensors, e.g. exit syngas temperature and steam production rate, heat flux sensors at a few key axial locations and strain sensors on the mechanical structure that support the weight of the heat exchange tubes – the weight will increase due to gradual fouling buildup, can be used. On a side note, such strain sensors were installed and used successfully in the previous program (DE-FC26-07NT43094) in the RSC at the Polk Power Station IGCC plant.

4.1 Elements of a Model Based OSP Algorithm

The solution to the OSP problem determines an optimal sensor network that allows direct sensing or indirect model-based estimation of key process variables for monitoring and control. The overall objective is to design a cost-effective sensor network that allows a robust monitoring of key process variables in the presence of sensor measurement errors as well as modeling errors, in addition to expected sensor failure rates. This problem is addressed within a model-based framework as shown in Figure 5, which combines both estimation and optimization, under which the latter is required to optimize over the sensor set used for estimation of the process variables in order to ensure that the estimation specifications are all satisfied.

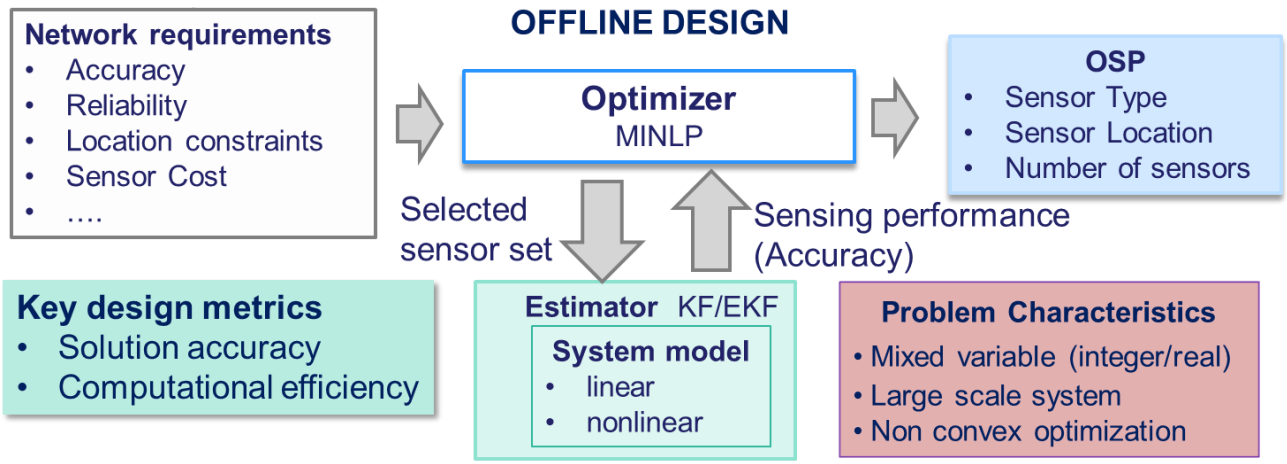


Figure 5: Schematic of model based approach for solving an OSP Problem

These specifications provide indicators to measure the quality of a particular sensor network. As shown in Figure 5, the OSP algorithm consists of three main elements: network requirement, estimator and the optimizer. Next, these parts are discussed in detail.

4.1.1 Network Requirements

The basic purpose of a sensor network is to "measure" certain signals that cannot be measured directly with sufficient accuracy. Hence the "measurement" obtained through sensor network should possess similar characteristics that are expected of a direct sensor measurement namely the accuracy of the measurement and the reliability. The reliability refers to the probability of having the required measurement accuracy in the presence of hardware sensor failures. In case of a sensor network consisting of model based sensing where the "measurement" is actually "estimate" of certain process variables these requirements are mapped appropriately through estimation process.

4.1.2 Estimation Problem

The overall objective of OSP is to design a cost-effective sensor network that allows a robust monitoring of key process variables in the presence of sensor measurement errors as well as modeling errors. This problem is readily addressed within the framework of model-based estimation using available techniques like Kalman filtering for linear systems or extended Kalman filtering (EKF) for nonlinear systems.

Consider a nonlinear dynamic system with the following discrete-time state-space model:

Eq. (1)

$$x_{k+1} = f(x_k) + w_k$$
$$y_k = h(x_k) + v_k$$

Where at a current time-sample k, x_k denotes the vector of states and y_k denotes the vector of measured outputs from the available sensors. The vectors w_k and v_k denote the process model uncertainty and measurement noise, assumed to be white noise with covariance *Q* and *R*, respectively. For the above nonlinear dynamic system, starting at an initial state estimate $\hat{x}_{0|0}$ and initial state covariance $P_{0|0}$, the EKF recursively updates at each sample k, the state estimate $\hat{x}_{k|k}$ and its covariance $P_{k|k}$:

Eq. (2)

$$\begin{aligned}
&\hat{x}_{k|k-1} = f\left(\hat{x}_{k-1|k-1}\right) \\
&\hat{y}_k = h\left(\hat{x}_{k|k-1}\right) \\
&P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q \\
&\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_k) \\
&K_k = P_{k|k-1} C_k^T \left(C_k P_{k|k-1} C_k^T\right)^{-1} \\
&P_{k|k} = (I - K_k C_k) P_{k|k-1} (I - K_k C_k)^T + K_k R K_k^T
\end{aligned}$$

Where the matrices $A_k = \left.\frac{\partial f}{\partial x}\right|_{x_k}$ and $C_k = \left.\frac{\partial y}{\partial x}\right|_{x_k}$ denote the linearized model matrices:

Eq. (3)

$$\begin{aligned}
x_{k+1} &= A_k x_k + w_k \\
y_k &= C_k x_k + v_k
\end{aligned}$$

In particular, the EKF provides a recursive estimate of the expected uncertainty in the state estimates as the corresponding covariance $P_{k|k}$. For an estimated output of interest depending on the states $y_k^e = h^e(x_k)$, or with the corresponding linear equation $y_k^e = C_k^e x_k$, the corresponding covariance or uncertainty in the output estimate is obtained as $C_k^e P_{k|k} C_k^{e^T}$. This provides a natural candidate to assess performance of estimation accuracy in the presence of expected modeling errors and sensor measurement errors, as a function of the chosen sensor set.

The measurement noise covariance *R* is normally obtained from the typical noise characteristics of the candidate sensors. The process noise reflects the expected uncertainty in the process model and its specification is more subtle. For OSP purpose, the signals of interest for condition monitoring (e.g. the wear condition on the hot inner surface of the gasifier refractory or the fouling in the RSC) are modeled as unknown process inputs u_k or parameters p_k that would need to be estimated. These inputs and parameters are included in the overall system model as:

Eq. (4)

$$\begin{aligned}
x_{k+1} &= f(x_k, p_k, u_k) + w_k \\
y_k &= h(x_k, p_k, u_k) + v_k
\end{aligned}$$

To allow estimation of these process inputs and parameters within a Kalman filter framework, they are included in an extended state vector $x^e = [x^T \;\; p^T \;\; u^T]^T$.

Eq. (5)

$$\begin{aligned}
x_{k+1} &= f(x_k, p_k, u_k) + w_k \\
p_{k+1} &= F_p(p_k) + w_{pk} \\
u_{k+1} &= F_u(u_k) + w_{uk} \\
y_k &= h(x_k, p_k, u_k) + v_k
\end{aligned}$$

However, this entails postulating a suitable model for these inputs/parameters to be estimated. Some examples include a random walk model $F_p(p_k) = 0$, a first-order Markov model $F_p(p_k) = \alpha p_k + (1-\alpha)p_0$, or a random ramp model $p_{k+1}^1 = p_k^2,\ p_{k+1}^2 = w_{pk}$. The choice of the postulated model for model parameters and inputs as well as the process noise covariance Q has to be done carefully based on practical knowledge of the variable being estimated/monitored. For instance, for process model parameters, typically their mean values and expected range of variation is known, which motivates the use of a first-order Markov model that yields the mean value p_0, and a desired variance based on the choice of α and the noise covariance Q.

4.1.3 Optimization Problem

The optimization problem deals with finding the minimum set of sensors required to meet network requirement. Let $q = [q_1, q_2, .., q_N]^T$ denote the binary decision variables with values 1/0 corresponding to whether a sensor at a given location $i, i \in \{1, ., N\}$ is being used or not and $c \in \Re^N$ denotes the associated sensor cost. This cost may include physical and installation cost *and/or* weight (in other applications). Then the optimization problem deals with finding the minimum set of sensors required to meet the desired estimation precision and reliability.

4.1.3.1 Precision Constraint

The estimation precision requirement for condition monitoring is defined as (Chmielewski et al 2002)

Eq. (6)
$$s_l(q) = C_l^e P_\infty(q) C_l^{e^T} \leq \bar{s}_l$$

Where, $P_\infty(q)$ is the solution of Algebraic Riccati Equation (*ARE),* given by:

Eq. (7)
$$P_\infty(q) = [(AP_\infty(q)A^T + Q)^{-1} + C^T R^{-1}(q) C]^{-1}$$

It should be mentioned that, depending on the specification of the process and measurement noise covariance Q and R, the above specification on precision based on steady-state Kalman filter covariance can be misleading. For instance, if the process noise covariance Q is specified as too small, then the steady-state covariance and corresponding precision achievable with a sensor set may be very good, but the transient estimation performance may be too slow for any practical use. The need for transient estimation performance is even more critical for control purposes, where the sensing/estimation is integrated with control/optimization. In that case use of the transient matrix $P_{\{k|k\}}(q)$ (Eq. (2) may be more appropriate instead of a steady-state one.

4.1.3.2 Reliability Constraint

The reliability is used to impose a design target probability of meeting the required estimation precision in the presence of expected sensor failures.

There are multiple definitions of reliability that have been considered in literature (e.g., Benqlilou et al. (2004)).

- Hardware reliability, which is a probability measure that at least one of the sensors in a given network will survive in order to estimate the monitored variable. This notion is defined by,

Eq. (8)

$$r_H(q) = 1 - \prod_{i=1}^{N}(1 - q_i p_i) > \bar{r}_l$$

where, p_i is the individual sensor survival rates and $q_i \in \{0,1\}$ indicates if the sensor at location i, is present $(q_i = 1)$ or not $((q_i = 0)$. This notion of reliability is a sufficient measure of accuracy under failures, when the monitored variables are directly sensed. However for the specific case of model based sensing (e.g., refractory wear monitoring, which estimates refractory wear through temperature measurements) such an indicator will not be applicable.

- Functional reliability, which increases the sensor network survivability by allowing a sufficient measure of redundancy to each sensor at any given location and is defined by

Eq. (9)

$$r_F(q) = 1 - \prod_{i=1}^{N}(1 - q_i p_i)^{n_i} > \bar{r}_l$$

Even though, the reliability definitions in Eq. (8) and Eq. (9) look similar, the key difference is that by allowing multiple sensors at each location (Eq. (9)), the survival rates increase thereby leading to robust estimation under failures which is not the case with hardware reliability. For the same reason, this notion will be applicable to refractory wear monitoring.

- Estimation reliability: Here, the key observation is that reliability is fundamentally tied to the precision and includes a measure of redundancy to the sensor network.

While the redundancy in the functional reliability is the number of sensors at each location, the redundancy in this notion is increasing the observability of a given network by considering as many configurations that can achieve a desired estimation precision. This is defined as,

Eq. (10)

$$r_l(q) = \sum_{\omega_k \in \Omega_q} \Pr(\omega_k) I(\omega_k) > \bar{r}_l$$

Where, $\Omega_q$is a set of possible failures associated with a particular sensor network $q \in \{0,1\}^N$. The reliability requirement involves computation of the conditional probability of each failure scenario $\Pr(\omega_k|q)$ based on individual sensor failure rates as well as the binary valued function $I(\omega_k)$ denoting whether with the reduced sensor set the precision requirement is met $(I(\omega_k) = 1)$ or not $(I(\omega_k) = 0)$.

It should be noted that for the OSP problem, the estimation reliability provides the cost-effective measure of capturing the requirements as compared to the multiple-sensor notion. However, implementing this reliability constraint involves a combinatorial enumeration (and summation) of all the possible failure scenarios for a given sensor network which becomes computationally prohibitive even for a small scale systems.

With the above two key requirements on precision and reliability in the presence of modeling and sensor errors, the OSP problem is posed as an integer nonlinear programming (INLP) problem:

Eq. (11)

$$\begin{aligned} &\min_{q} c^T q \\ &s_l(q) \leq \bar{s}_l \\ &r_l(q) \geq \bar{r}_l \\ &\quad q_i = \{0,1\}, \quad for \;\; i = 1, 2, \dots, N \end{aligned}$$

Where, the vector $q = [q_1 \; q_2 \;\; \dots \;\; q_N]^T$ denotes the binary decision variables with values 1 and 0, corresponding to whether the respective sensor is used or not used, respectively, and the vector *c* denotes the cost of each sensor to be used in the network. To be consistent with the binary decision for each q_i, the sensor noise covariance $R_{i,i}$ is scaled with q_i according to the relation:

Eq. (12)

$$\frac{1}{R_{i,i}} = \frac{q_i}{R_{i,i}^0}$$

Where, $R_{i,i}^0$ denotes the baseline measurement noise covariance for the ith sensor. The inverse of the noise covariance, i.e., $R_{i,i}^{-1}$, is scaled linearly with respect to q_i. Thus, for $q_i = 1$, the baseline sensor noise covariance $R_{i,i}$ is same as $R_{i,i}^0$. This amounts to using the ith sensor with its intrinsic noise properties. On the other hand, for $q_i = 0$, the sensor noise covariance, $R_{i,i}$, is scaled to infinity, thereby effectively eliminating sensor from the sensor pool as the corresponding Kalman filter gain with respect to the ith sensor will be zero.

The OSP problem in Eq. (11) has to be solved through an integer programming optimization algorithm. There are multiple approaches for solving the INLP problem. In general, large scale INLP problems are the toughest optimization problems to solve. The key difficulty arises due to the combinatorial nature of the problem. The following section reviews available methods suited for solving such INLP problems in the context of OSP problem and their relative merits and demerits.

4.2 Technology review: Integer Programing Algorithms for OSP Problem

There is a vast literature on general integer and mixed-integer linear and nonlinear programming algorithms, where the latter involves both real and integer variables in the decision space. The OSP problem (see previous section) being confined only to integer decision variables is treated as an integer nonlinear programming problem (INLP). An excellent survey of the available methods and software is found in Bonami et.al. (2005, 2009), and this section provides a brief summary of these available algorithms.

The key difficulty in solving an integer programming problem stems from the fact that the optimal integer solutions cannot be computed by rounding the optimal solutions of the continuous relaxation for which known NLP solvers exist. The true integer optimal solution and the rounded solution are seldom the same or even close. This implies the need for methods that can search in the integer space, which is a combinatorial search space. This combined with the associated nonlinearities motivated a separate methodology development for the MINLP programming. The following ILP example (Chinneck , 2010) illustrates that the method of rounding the optimal solutions for continuous relaxations does not work. The ILP problem given by,

$$max\ _{q} q_1 + 5q_2$$

Eq. (13)

$$q_1 + 10q_2 \leq 20, q_1 \leq 2, q_1, q_2 \geq 0$$

has an integer optimal solution of $q_1 = 0\ , q_2 = 2$. The continuous relaxed LP problem yields the optimal solution $q_1 = 2\ , q_2 = 1.8$. Rounding this continuous relaxed solution yields an integer solution $q_1 = 2\ , q_2 = 2$, which is infeasible, or $q_1 = 2\ , q_2 = 1$, which is sub-optimal.

The basic idea of the existing methodologies is the generation, at each iteration, of a lower and an upper bound on the optimal solution of the MINLP/INLP problem wherein the bounds are progressively tightened at every iteration. The generation of the bounds at every iteration is done by solving a relaxation of the original MINLP/INLP problem.

The bounds are either tightened by an exhaustive enumeration/tree search OR by tightening the relaxation of the feasible region. Based on the approach used, the existing methods in literature can be broadly classified into the following:

1. Branch and bound (BB) (Floudas et.al, 1995) and its variants which requires an enumeration at every iteration
2. Outer approximation (OA) (Duran et.al. 1986) and Generalized Benders Decomposition (GBD) (Geoffrion et.al. 1974)

where for methods 2 and 3, the relaxed feasible region is tightened at every iteration. The branch and bound based methods incur the issue of combinatorial explosion due to the exhaustive enumeration, but is guaranteed to converge to the true integer optimal solution. Since the GBD is very similar to the outer approximation in terms of the algorithm and the

underlying assumptions, the main focus of the discussion will be restricted to BB and OA based methodologies.

In the sequel, both these methods are described in detail along with their applicability to the OSP problem under consideration. To do so, the following general MINLP problem is considered.

Eq. (14)

$$Z^* = \min_q f(q,y)$$
$$g(q,y) \leq 0$$
$$\mathrm{q} \in \{0,1\}^n, \mathrm{y} \in \mathrm{Y} \subseteq \Re^m$$

where, $f(.), g(.)$ are convex functions and q,y are the continuous and integer decision variables. It must be noted that the OSP problem in Eq. (11) is a subset of the class of MINLP problems in Eq. (14) with the decision space being only restricted to integer variable (more specifically, $q \in \{0,1\}^n$) which implies y = []. In a general MINLP problem in Eq. (14) the *q* values may take any integer values, which can be converted into an MINLP with only binary-valued integer variables by introducing new binary variables for non-zero integer value allowed. So in the rest of this report, only binary-valued integer variables will be considered in the MINLP. Moreover, for the OSP problem in Eq. (11) the integer variables for each sensor are binary-valued variables to begin with.

The following notations will be used through the remained of the report. The optimal value, optimal solution and feasible space to the MINLP problem are compactly denoted as $(\boldsymbol{Z}^*, \boldsymbol{q}^*, \boldsymbol{F})$ and $(\overline{\boldsymbol{Z}}, \overline{\boldsymbol{q}}, \overline{\boldsymbol{F}})$ denotes the optimal value, solution and the feasible space to any relaxation of the MINLP problem.

4.2.1 Branch-and-Bound

The branch and bound (BB) method (Nemhauser et.al. 1988) has been used for solving a wide variety of linear/nonlinear integer/mixed-integer programming problems. The BB method is a global optimization method and works on the idea that enumeration of the integer solutions has a tree structure. The following terminology is used in standard branch and bound algorithms in describing the tree structure which is shown in Figure 6.

- Root node which is the root of the tree structure
- The bud node is any node in the tree which represents the partial solution which is the solution of the NLPR problem which is either feasible or infeasible. The bud node represents a node that can be potentially grown.
- Leaf Node is the node that is at the bottom of the tree, where all the variables are known or specified.

The root of the tree structure contains all possible solutions to the MINLP problem in Eq. (14) which is typically computed as the optimal solution of a relaxation of the original problem (e.g. integer relaxation). The leaf nodes of the tree structure are subsequently obtained by enumerating the actual integer solutions. For example, for a problem with two integer variables, the enumeration tree is shown in Figure 6, where the node at the upper right of the tree represents the solution in which $x_1 = 0 \; and \; x_2 = 0$.

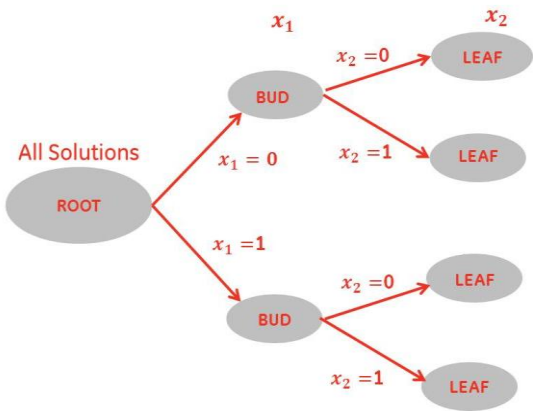


Figure 6: A full enumeration tree in BB algorithm with two integer variables (Chinneck 2010)

Each node in the tree represents a set of all possible solutions that can be obtained by growing the tree. It can be immediately seen that the tree enumeration is exponential in terms of the number of integer variables making this method computationally prohibitive if the whole tree has to be grown. For the same reason the main idea of the BB method is to grow the tree ONLY in stages, i.e., grow the most promising node at any stage. This is achieved based on the fact that the optimal solution to the relaxed problem at the parent node provides a lower bound on the optimal solution to the relaxed problem at the child node, i.e.,

Eq. (15) $$\bar{Z}_{root} \leq \bar{Z}_{leaf}.$$

At any node, if the optimal solution of the relaxed problem satisfies all the constraints of the original MINLP problem, i.e., $\bar{q} \in \{0,1\}^n \; and \; g(\bar{q}, y) \leq 0$, then the objective function value is called as the incumbent, namely the best possible objective function, denoted as (I = f($\bar{q}, y$)); the incumbent provides the upper bound on the optimal solution. The determination of the most promising node is done by estimating the best lower bound across all bud nodes. Given the current best lower bound and upper bound, a specific bud node is selected for further growth of the tree to iteratively arrive at the optimum solution. To this effect, there are three important steps in any standard branch and bound algorithm.

4.2.1.1 Branch and Bound – steps

Branch and bound algorithm has following steps:

1. Branching which happens when a node is selected for further growth, commonly called as the bud node, from a list of all possible nodes. There are many different ways of node-selection (Floudas, 1995), the most common strategies being
    a. *Depth-first***,** where the bud nodes are grown till they can no longer be.
    b. *Breadth-first*, where the bud-nodes are expanded in the same order in which they were created.

   c. *Best-first*, where the bud node with the best value of the objective function is grown

2. Bounding happens when the best value of the objective function provided by the parent node is attained with integer variables, thereby yielding a leaf node and possible incumbent.

3. Fathoming is the step where the nodes are either completely discarded from further growth when it is concluded that all the child nodes for a given node are infeasible or sub-optimal, or retained as an incumbent. More specifically,
   a. A bud node may be fathomed by bounds, which happens when the lower bound at the bud node is higher than the incumbent (for minimization).
   b. Similarly, a bud node may be fathomed by infeasibility, when there are no feasible solutions to the relaxed problem, which would mean that the original problem is also infeasible due to the fact that $F \subseteq \overline{\boldsymbol{F}}$.
   c. Lastly, if the solution at a bud node is integer feasible and the optimum value is lower than the incumbent, then it is fathomed and retained as the new incumbent replacing the previous one.
   Steps a, and b above lead to pruning, i.e., the current bud node is discarded from solution candidates. This is an important step to control the growth of the tree and limit the search space. On the other hand, step c updates the upper bound for the solution.

4. Termination happens when
   a. When there are no more bud nodes to search and the incumbent is identified as an optimal solution.
   b. Alternatively, one may terminate when the upper bound (incumbent) and lower bound (current best solution among all bud nodes) are within a specified tolerance to get an optimal/sub-optimal solution within that tolerance.

While the BB provides a framework for solving the MINLP problem, there are many factors that need to be considered in order to avoid the tree explosion. Some important factors that influence the speed of the algorithm include getting the incumbent as quickly as possible to obtain a good upper bound, and the choice of a good bounding function with a tight relaxation such that the corresponding lower bound is closer to the optimum value.

For nonlinear integer optimization the branch and bound method can be represented as in Figure 7. A key element of the branch and bound algorithm is to use appropriate relaxation of the original problem to generate lower bound of the optimum solution at each node. The relaxed problem is relatively easier to solve. If the solution of the relaxed problems is all integer variables, then the upper bound/incumbent is updated. Subsequently, a tree-search algorithm is used to pursue promising solution candidates until a candidate solution to the original un-relaxed problem is obtained, thereby providing an incumbent with an upper bound on the optimum solution. The tree search is continued until the updated upper and lower bounds converge (or get within a desired tolerance) to identify the optimal solution (or get within desired tolerance to optimum solution). The efficiency of the branch and bound algorithm depends on the search algorithm (e.g. depth-first, bread-first, most-promising-first, etc.) as well as the relaxation to the original problem.

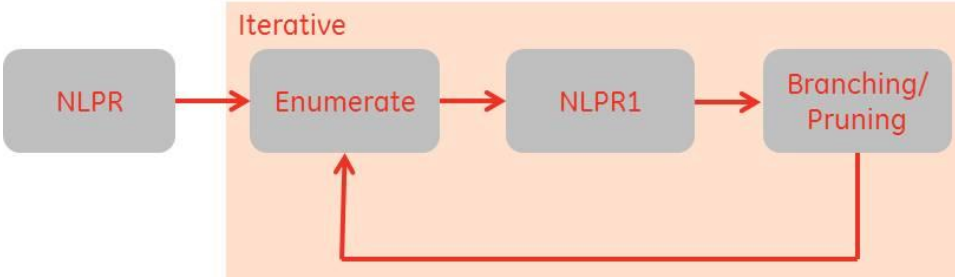


Figure 7 : Schematic of a Branch and Bound Algorithm using NLP relaxation

Many different kinds of relaxations and bounding functions exist in literature and they can be classified into

- Integer relaxation
- Lagrangian relaxation (Fisher 1981, Geoffrion, 1974)
- SDP relaxation (Laurent and Rendl, 2002)

As mentioned earlier, all these relaxation techniques aim to solve relatively easier problem to obtain lower bound at each node. NLP relaxation relaxes the integrality constraints. The NLPR1 (in Figure 7) is solved by relaxing the integer constraint and solving the OSP problem as a convex optimization problem. Various relaxation techniques are discussed next.

4.2.1.2 Integer relaxation - NLP Relaxation

This relaxation is obtained by relaxing the integer constraints on the decision variables and replacing with simple bound constraints to obtain:

Eq. (16) NLPR

$$\bar{Z} = \min_{q} f(q,y)$$
$$g(q,y) \leq 0$$
$$\mathrm{q} \in [0\ 1]^n, \mathrm{y} \in \mathrm{Y} \subseteq \Re^n$$

At the root node of the BB algorithm, the NLPR problem is solved. If all the decision variables of the NLPR take integer values the search is stopped, otherwise a tree search is performed by expanding the bud nodes in the tree. Each node in the enumeration tree represents potential solutions and solves the following NLPR1 sub-problem given by

Eq. (17) NLPR1

$$\min_{q} f(q,y)$$
$$g(q,y) \leq 0$$
$$q \in [0\ 1]^n, y \in Y \subseteq \Re^n$$
$$q_i \geq 1, i \in I^{Upper},$$
$$q_i \leq 0, i \in I^{Lower},$$

Where, at any iteration, I^{Upper} and I^{Lower} are the index subsets of the integer variable specified as 1 or 0, while the remaining variables are relaxed to be real variables in the entire range 0-1. For example, the node 2 in Figure 6, corresponding to $q_1 = 1$ solves the following with an additional constraint and is given by $q_i \geq 1$. It should be noted under assumptions of the convexity of the objective function and constraints in NLPR that $Z_{NLPR} \leq Z_{NLPR1}$, where Z_{NLPR}, Z_{NLPR1} denote the optimal solutions of NLPR and NLPR1, respectively.

Remarks

While the attractiveness of the BB method comes from the wide range of applicability to linear/nonlinear functions with integer only/mixed-integer decision variables, the size of the decision space poses challenges especially for nonlinear problems. The BB method is very well established for MILP and ILP problems given the many commercial linear programming packages such as GLPK, CPLEX etc.

A key requirement in the case of nonlinear problems is the availability of good NLP solvers, since the NLPR problem needs to be solved at each iteration. These solvers should be able to converge to the global optimal solutions for large-scale problems (a few thousand decision variables) in a finite amount of time and should not suffer from many numerical issues. There have been a lot of improvements in the development of computationally efficient NLP solvers such as IPOPT, which makes this requirement more and more feasible. Nevertheless, for large-scale problems, BB with NLP relaxation may end up searching large set of combinations before converging at the solution. In particular, often it may approach close or even at the solution, but has to complete an exhaustive search of the remaining bud nodes until they are all discarded to confirm the optimality of the incumbent.

4.2.1.3 Lagrangian Relaxation

As mentioned earlier, branch-and-bound algorithm is a very commonly used algorithm for solving linear/nonlinear integer programming problems. A key to efficient use of branch-and-bound is to arrive at a “relaxed” problem that is easier to solve and provides an optimistic estimate of the optimal solution, i.e. a lower bound in the case of minimization or an upper bound in the case of maximization. The relaxation involves approximating or ignoring a set of “difficult” constraints in the original problem such that the resulting problem can be solved readily. One such straightforward and common relaxation involves ignoring the integer constraints and treating the integer variables, e.g. the binary integer variables q_i in the sensor placement problem (Eq. (11)), as real variables with lower and upper bounds. This is referred to as LP (for linear) and NLP (for nonlinear) relaxation. While such an LP/NLP relaxation is straightforward and allows using available algorithms for LP & NLP problems within a branch-and-bound framework, the relaxed problem is often very optimistic and leads to increased number of iterations in the branch-and-bound algorithm.

This motivated the use of “tighter” relaxations, where the relaxed problem provides an optimistic, yet closer bound to the original integer programming problem as compared to

the LP/NLP relaxation. One such relaxation technique is the Lagrangian relaxation. The Lagrangian relaxation technique was introduced in the pioneering work of Held and Karp (1970, 1971), where they used this technique to address the solution of the well-known traveling salesman problem. The traveling salesman problem entails solving for the optimal route of a salesman between multiple cities, starting and ending at the same city and visiting every other city once. The relaxation of the original problem with a minimum spanning tree problem, where the constraint of starting and ending cities on the tour being the same is relaxed, yielded sharp lower bound on the original optimum. Geoffrion (1974) developed a formal framework for using Lagrangian relaxation and exploiting special structures for solving linear integer programming problems within a branch-and-bound technique. Additional work in this field in the 70's is summarized in the review paper by Fisher (1981).

Lagrangian relaxation has been used for solving linear mixed integer or integer programming problems. More specifically, for linear binary integer programming problems of the following general form:

Eq. (18)

$$\begin{aligned} &P_{IP} = \min_{x} c^T x \\ &Ax \leq b \\ &Cx \leq d \\ &x_i = \{0,1\} \end{aligned}$$

with binary decision variables x_i and linear objective function and constraints, the Lagrangian relaxation is used to relax a sub-set of the constraints. More specifically, the constraints are classified into two sub-sets (i) easy constraints and (ii) hard constraints. The integer programming problem is considered relatively easier to solve if only the easy sub-set of constraints were present, e.g. simple bound constraints. The inclusion of the hard constraints (e.g. general inequality constraints) makes the problem much harder to solve. The categorization of constrains in an integer programming problem as easy/hard is problem specific with some heuristic guidelines. However, it is a key step in deciding which constraints to relax and the subsequent efficiency of the solution of the original integer programming problem. There is often a trade-off in the complexity involved in solving the relaxed problem and the tightness of the bound obtained from the relaxed problem.

For the problem in Eq. (18), it is considered that the first set of constraints $Ax \leq b$ represents the hard constraints, while $Cx \leq d$ represents the easy constraints. Given this classification of the constraints, the hard constraints are then "relaxed" by discarding these constraints and including them in the objective function, with a penalty weight to obtain the following relaxed problem:

Eq. (19)

$$\begin{aligned} &P(\lambda) = \min_{x} c^T x + \lambda^T (Ax - b) \\ &Cx \leq d \\ &x_i = \{0,1\} \end{aligned}$$

In the above relaxed problem, the penalty weight vector λ denotes the vector of Lagrange multipliers for each of the hard constraints and is all non-negative, i.e. $\lambda_i \geq 0$. Note that for $\lambda_i \geq 0$ and for all feasible points x satisfying $Ax \leq b$, it follows that $P(\lambda) \leq P_{IP}$. Thus, the relaxed problem in Eq. (19) provides an optimistic or lower bound for the original integer

programming problem in Eq. (18), which can then be used in the branch-and-bound algorithm to solve the integer programming problem. In particular, the corresponding dual problem

Eq. (20)
$$D = \max_{\lambda} P(\lambda)$$
$$\lambda \geq 0$$

provides the tightest lower bound for the original problem . For linear programming (LP) problems, there is a well-known strong duality theory that establishes that at the optimum solution, the original problem and the dual problem have the same value. In contrast, for linear integer programming problems, the strong duality doesn't hold. Instead only a weak lower bound relationship holds:

Eq. (21)
$$D \leq P_{IP}$$

The above fact allows using the optimistic lower bound obtained through the Lagrangian relaxation within a branch-and-bound algorithm. A key issue that impacts the efficiency of the branch-and-bound algorithm is to obtain as tight a lower bound as possible to the original problem at each node in the branch-and-bound algorithm. This entails solving the dual problem in Eq. (20) using gradient based search techniques to identify the optimal value of the Lagrange multipliers λ. However, unlike for LP problems, the dual function is only piecewise differentiable. This led to the use of sub-gradient optimization methods, iteratively optimizing for the Lagrange multipliers λ. A vector s is defined as sub-gradient of the function $P(\lambda)$ at a point $\bar{\lambda}$, if it satisfies:

Eq. (22)
$$P(\lambda) \leq P(\bar{\lambda}) + s(\lambda - \bar{\lambda}), \qquad for\ all\ \lambda$$

In particular, the vector $s = Ax - b$ is a sub-gradient at a point x that is a solution of $P(\lambda)$.

The sub-gradient optimization method is summarized as:

- Choose a starting value, i.e. $\lambda_0 \geq 0$ – for instance it can be initialized at the previous converged value from the parent node in a branch-and-bound technique.
- Choose a sub-gradient, i.e., $s_k = Ax_k - b$ at current iteration *k*. If $s_k = 0$ then stop.
- Update the Lagrange multiplier $\lambda_{k+1} = \max(0, \lambda_k + \gamma_k s_k)$, where $\gamma_k$denotes a step size in the sub-gradient direction.
- Repeat for next iteration until convergence.

A common heuristic used to decide the step size is:

Eq. (23)
$$\gamma_k = \mu_k \frac{P^* - P(\lambda_k)}{||Ax_k - b||^2}$$

where P^* is the best value of the original integer programming problem found so far and μ_k is a decreasing step-size adaptation parameter in the range $0 < \mu_k \leq 2$, initialized at 2 and decreased by half after a pre-set number of iterations fail to increase the value of $P(\lambda_k)$.

The Lagrangian relaxation of the original linear integer programming problem provides a tighter lower bound than the straightforward linear programming (LP) relaxation where the binary integer constraints are replaced with simple bounds on the decision variables, i.e. $0 \leq x_i \leq 1$. While there is no general theoretical result on the tightness of the lower bound achieved through the Lagrangian relaxation, in practice, it often provides a very tight bound for a range of linear integer programming problems – see Table 2 in Fisher (1981). This reduces the degree of conservatism and reduced search in the branch-and-bound algorithm.

Remarks

Lagrangian relaxation has been used to solve many practical integer linear programming problems (see Fisher 1985 for some review of practical applications), and its promise lies in the ability to obtain tight bounds on the optimal solution for the original integer programming problem (see Table 2 in Fisher 1981 for a list of problems and tightness of bounds obtained by Lagrangian relaxation). However, there are some key limitations to the approach. First, this approach has been used for linear integer programming problems, where the relaxed problem, which is still a linear integer programming problem, can be solved more readily once the "difficult" constraints have been relaxed. The improved tightness of the bounds compared to standard LP relaxation, often depends on which constraints are relaxed. This leads, to an often unclear option of which sub-set of constraints should be relaxed, and thus it is often left to the user to experiment with alternative options and decide. Furthermore, this approach has not been applied to nonlinear integer programming problems – note that while Lagrangian relaxation can be applied in concept to nonlinear problems, the relaxed problem will still be an integer nonlinear programming problem (INLP), which, in general, is no easier to solve than the original INLP. The OSP problem described in the previous section has linear objective function, but the precision and reliability constraints are nonlinear functions of the integer sensor placement decision variables. So, available Lagrangian relaxation techniques are not readily applicable to the OSP problem.

4.2.1.4 SDP Relaxation

Consider the following linear 0/1 programming problem (Laurent et.al. 2002):

Eq. (24)

$$\begin{aligned} P_{IP} &= \min_x c^T x \\ Ax &\leq b \\ x_i &= \{0,1\}^n \end{aligned}$$

The classical polyhedral approach to solving this problem consists of formulating the above problem in Eq. (24) as a linear programming problem:

Eq. (25)

$$\begin{aligned} P_{LP} &= \min_x c^T x \\ sub&: x \in P \end{aligned}$$

over the polytope

$$P \coloneqq conv(\{x \in \{0,1\}^n | \, Ax \leq b\})$$

and applying linear programming techniques to it. For this, one needs to find the linear description or a good linear relaxation of *P*. A common linear relaxation of *P* is:

$$K \coloneqq \{x \in R_+^n |\ Ax \leq b\}$$

In general $K \neq P$ and one has to find "cutting planes" to strengthen the relaxation *K* by cutting off its fractional vertices

There are few general purpose methods for finding "cutting planes" applied to arbitrary 0/1 problems. The Chv´atal-Gomory cuts based on integer rounding is one such example. Another method of constructing a linear relaxation to polytope *P* is lift and project. In the nineties several authors investigated lift-and-project methods. The basic idea here is to find representation of a 0/1 polytope as the projection of a polytope lying in higher dimension. Depending upon the construct, the lift and project methods also lead to Semidefinite relaxations as in the case of Lov´asz and Schrijver technique (Lov´asz 1991). Further constructions for semidefinite relaxations have been recently investigated, based on algebraic results about representations of nonnegative polynomials as sums of squares of polynomials (e.g., Kojima 2003).

A semidefinite program (SDP) is the analogue of the linear program where the vector variable $x \in R^n$ is replaced with a matrix variable $X \in R^{nxn}$, constrained to be symmetric positive semidefinite:

Eq. (26)

$$\begin{aligned} &P_{SDP} = \min_X C \cdot X \\ &sub \quad A_j \cdot X \ \leq b_j \ (j = 1, .., m) \\ &X \geq 0 \end{aligned}$$

where (·) represents the inner product of matrices (e.g., $A_j \cdot X \coloneqq tr(A_j^T X)$). There are many algorithms to solve these programs in polynomial-time within any specified accuracy, e.g., interior-point algorithms.

4.2.1.4.1 SDP and Integer 0/1 (Binary) Programming

In order to embed the 0/1 linear program of Eq. (24) in SDP formulation (Laurent 2002), the problem is linearized by introducing a new matrix variable *X* for xx^T, i.e., define $X_{ij} := x_i x_j, \forall i, \forall j$, etc . The integer constraint $x_i \in \{0,1\}$ is equivalent to $x_i^2 = x_i,$ and it is embedded by requiring $X_{ii} := x_i, \forall i$ or in other words, $diag(X) = x$. However the constraint $X - diag(X)diag(X)^T = 0$ is not convex. This rank-1 condition is relaxed to $X - diag(X)diag(X)^T \geq 0$, which can be equivalently written as:

Eq. (27)

$$\begin{pmatrix} 1 & diag(X)^T \\ diag(X) & X \end{pmatrix} \geq 0$$

Laurent (Laurent 2002) refers to several possibilities for modeling a linear constraint $a^T x \leq \beta$ from the system $Ax \leq b$. For example:

Diagonal representation:

$$\langle diag(a), X \rangle \leq \beta$$

Eq. (28)

Squared representation:

$$(a^T x)^2 \leq \beta^2 \;\Rightarrow\; \langle aa^T, X \rangle \leq \beta^2$$

Eq. (29)

Extended square representation:

$$(a^T x)^2 \leq \beta (a^T x) \;\Rightarrow\; \langle aa^T - \beta diag(a), X \rangle \leq 0$$

Eq. (30)

Another possibility is to exploit the fact that the variable x_i satisfies $0 \leq x_i \leq 1$ and to multiply $a^T x \leq \beta$ by x_i and $1 - x_i$:

Eq. (31)

$$\sum_{j=1}^{n} a_j X_{ij} \leq \beta X_{ii};\;\; and$$
$$\sum_{j=1}^{n} a_j (X_{jj} - X_{ij}) \leq \beta (1 - X_{ii}) \;\; \forall i$$

The tightness of the relaxation depends upon the representation. For example here, the relaxation in Eq. (31) gives the tightest bound whereas the relaxation in Eq. (28) is the least tight. Putting all together, the SDP relaxation of the linear integer optimization problem in, Eq. (24) using representation in Eq. (31) for the linear constraint, is as follows:

Eq. (32)

$$\min_X \langle diag(c), X \rangle$$
$$\sum_{j=1}^{n} a_j X_{ij} \leq \beta X_{ii};\;\; and$$
$$\sum_{j=1}^{n} a_j (X_{jj} - X_{ij}) \leq \beta (1 - X_{ii}) \;\; \forall i$$
$$\begin{pmatrix} 1 & diag(X)^T \\ diag(X) & X \end{pmatrix} \geq 0$$

4.2.1.4.2 Lift and Project Methods

As mentioned earlier, SDP relaxation for integer program is a lift-and-project method to represent a 0/1 polytope as a projection of a polytope lying in higher dimension. In fact any n-dimensional 0/1 polytope can be equivalently realized as convex positive semidefinite program in $2^n - 1$ variables (Lasserre 2001). Of course, from practical computation purpose, this result is of little value because the number of variables grows exponentially with the size of the problem.

There are several general purpose methods for constructing projection representations for general 0/1 polyhedral. For example, lift and project method by Balas, Ceria and Cornu´ejols (Balas et.al. 1993), Reformulation-Linearization Technique (RLT) by Sherali and Adams (Sherali et.al. 1990), matrix-cuts by Lov´asz and Schrijver (Lov´asz et.al. 1991) and explicit exact SDP relaxation by Lasserre (Lasserre 2001). A common feature of these methods is the construction of a hierarchy of linear or semidefinite relaxation K_i of P which finds the exact convex hull in n steps. For more details about each of these techniques refer to the literature cited earlier in this paragraph. One can also refer to (Laurent,2001) or (Laurent 2002).

Remarks

SDP relaxation has been used to solve many practical integer programming problems. This approach is readily applicable to linear, quadratic and polynomial optimization problems (Laurent 2002, Mevissen et.al., 2010, Kojima et.al., 2000, Kojima 2003). There are efficient polynomial time algorithms, for example interior point algorithm, for solving SDP to any fixed prescribed precision.

For certain problems, e.g., Max-Cut, SDP relaxation has been shown to yield strong relaxation (0:8786 approximation algorithm for max-cut (Goemans & Williamson 1995)). The number of variables and/or constraints in an SDP relaxation is one order of magnitude higher than that of the original problem. Hence, the cost of solving such SDP problems grows quickly as the size of the problems increases. In other words, a key issue here is the scalability of the SDP relaxation approach with respect to the problem size. In this regard, the problem size of the optimal sensor placement problem makes it a rather unattractive option, especially for medium/large-scale OSP problems with hundreds/thousands of integer decision variables.

For certain problems, e.g., Max-Cut and the Max-Bisection, rank-two relaxation (Burer, et. al. 2001) produces better-quality approximate solutions while taking only a fraction of time in comparison to the SDP relaxation approach. However, such approach and its performance is very problem specific.

4.2.2 Generalized Benders Decomposition and Outer Approximation Algorithms

The Generalized Benders Decomposition (GBD) (Geoffrion, 1972) and the Outer Approximation (OA) (Duran et.al.,1984) are non-enumerative methods as opposed to the BB method and have been applied to MINLP problems. Since they avoid the enumeration

of integer combinations, they are potentially faster, especially for large-scale problems. The basic idea in both these methods is the generation of an upper and a lower bound on the optimal solution sought for the MINLP problem, and update these bounds until convergence. Even though at the onset this may seem similar to the BB method, it is not quite true. More specifically, in BB the bounds are obtained through enumeration and solving an NLP problem. In contrast, in OA/GBD based methods, the bounds are obtained by iteratively solving the primal (upper bound) and master problems (lower bound) till convergence to true integer optimal solution is obtained.

The assumptions made in the development of both the algorithms are similar and are listed for the MINLP problem in Eq. (14). It must be noted that the original development also includes the equality constraints and the readers are referred to (Floudas, 1995) for further details.

Assumptions

1. Y is non-empty and the functions f(.) and g(.) are convex for each fixed $q \in \{0,1\}^n$
2. The set $\Omega = \{q \in [0,1]^n | g(q,y) \leq 0 \; for \; some \; y \in Y\}$ is closed
3. For each fixed $q \in (\{0,1\}^n \cup \{q | g(q,y) \leq 0, for \; some \; y \in Y\})$, one of the following holds true
    a. The problem in Eq. (14) has a finite solution and has optimal Lagrange multiplier
    b. The problem in Eq. (14) is unbounded.

4.2.2.1 Primal Problem

The primal problem corresponds to the problem in Eq. (14) with fixed integer variables, where the decision space is only over the continuous variables. The idea is based on the Benders approach (Geoffrion, 1974) that exploits the notion of complicating variables, i.e., variables which, when temporarily fixed, render the remaining optimization problem considerably more tractable. Thus, in the primal problem, by fixing the integer variables, the MINLP problem in Eq. (14) can be reduced to a tractable NLP problem and is given by

Eq. (33) NLP2

$$\begin{aligned} Z^{UB} &= \min_{y} f(q^k, y) \\ g(q^k, y) &\leq 0 \\ y &\in Y \subseteq \Re^m \end{aligned}$$

Based on the assumptions (1-3) that primal problem has a global optimal solution for a fixed integer, an upper bound on the MINLP problem is obtained if the NLP2 problem is feasible. The solution of this problem also provides the Lagrange multipliers for the constraints and the corresponding Lagrangian function given as

Eq. (34)

$$L(q_k, y, \lambda^k) = f(q^k, y) + {\lambda^k}^T g(q^k, y),$$

If the primal problem is infeasible, then a feasibility problem is solved which is given by

Eq. (35) FP

$$\min_{y} \sum_{i=1}^{p} \alpha_i$$
$$g_i(q^k, y) \leq \alpha_i$$
$$\alpha_i \geq 0\,, i = 1,..p$$

The solution of this problem provides the Lagrange multiplies for the constraints and the Lagrangian function is given by

Eq. (36)

$$L\left(\bar{q}_k, y, \bar{\lambda}^k\right) = \bar{\lambda}^{k^T} g(\bar{q}_k, y)$$

It must be noted that no upper bound is generated if the primal problem is infeasible.

4.2.2.2 Master Problem

The master problem is a relaxation and hence it is used to provide the lower bound. The GBD and OA differ in the master problem formulation, even though they are conceptually quite related. The key ideas in both of these problems are first summarized, and then the actual formulation is defined. The MINLP problem Eq. (14) can be written in terms of the primal problem NLP2 and is given by

Eq. (37) MP

$$\min_{q} v(q)$$
$$\mathrm{q} \in (\{0,1\}^{\mathrm{n}} \bigcap \Omega)$$

where $\Omega\ is\ defined\ in\ assumption\ 2$ and *v(q)* is given as

Eq. (38)

$$v(q) = \inf_{v} f(q, y)$$
$$g(q, y) \leq 0$$
$$y \in Y \subseteq \Re^n$$

The MP problem in Eq. (37) is the projection of MINLP on the integer space and is hard to solve given that the set Ω and *v(q)* are known implicitly. The master problem defined in GBD and OA based methods alleviates this problem and characterizes the set Ω and *v(q)* explicitly, where the set Ω is defined by a collection of intersecting feasible regions that contain the set. The two different formulations are given below

<u>GBD Master Problem</u>

The GBD method makes use of the nonlinear duality theory and the main ideas used to characterize *v(q)* and Ω are based on the dual representations of each of them, given as follows: The master problem is a relaxation and hence is used to provide the lower bound. The GBD and OA differ in the master problem formulation, even though these are conceptually quite related. The key ideas in both of these problems are first summarized and then the actual formulation is defined.

1. The set $\Omega = \{q | g(q,y) \leq 0 \; for \; some \; y \in Y\}$ is characterized by the dual and is given as $\inf_{\bar{\lambda}} \bar{L}\left(\bar{\lambda}, \bar{q}, y\right) \leq 0$
2. Then using the *v(q)* can be defined by the dual and is given by

Eq. (39) MP

$$\min_{q} \sup_{\lambda} \inf_{y \in Y} L(y,q,\lambda)$$
$$\inf_{y \in Y} \bar{L}(\bar{\lambda}, \bar{q}, y) \leq 0$$

Based on the above ideas and assumptions, the MP in Eq. (39) for the GBD is given as,

Eq. (40) MGBD

$$\min_{\alpha,q,y} \alpha$$
$$\mathrm{f}\left(\mathrm{q}^{\mathrm{k}}, y^{k}\right) + \nabla \mathrm{f}_{\mathrm{q}}\left(\mathrm{q}^{\mathrm{k}}, y^{k}\right)(q - q^{k}) + \lambda^{k} \nabla \mathrm{g}_{\mathrm{q}}\left(\mathrm{q}^{\mathrm{k}}, y^{k}\right)(q - q^{k}) \leq \alpha, \forall k \in feasible\ primal$$
$$\bar{\lambda}^{k} \nabla \mathrm{g}_{\mathrm{q}}\left(\mathrm{q}^{\mathrm{k}}, y^{k}\right)(q - q^{k}) \leq 0, \forall k \in infeasible\ primal$$
$$q \in \{0,1\}^{n}, y \in Y \subseteq \Re^{m}, \alpha \in \Re \min_{q} v(q)$$
$$\mathrm{q} \in (\{0,1\}^{\mathrm{n}} \bigcap \Omega)$$

Where, $\lambda^{k}, \bar{\lambda}^{k}$ are the associated Lagrange multipliers when the primal is feasible and infeasible, respectively. The master problem MGBD is an MILP problem which is more tractable than an MINLP problem.

<u>OA Master Problem</u>

The earlier developments of this algorithm (Duran and Grossman, 1986) considered MINLP problems with a specific class of functions, namely functions linear in integer variables with the objective function $f(q,y) = c^{T}q + f(y)$ and constraints of the form $\mathrm{g}(q,y) = \mathrm{K}q + g(y) \leq 0$. However, many recent versions of OA as well as its variants do not have this restriction.

Similar to the arguments used in GBD, the OA algorithm defines the master problem by characterizing *v(q)* and the set Ω. Unlike the GBD algorithm that uses duality to do so, the OA algorithm utilizes linear approximations of *v(q)*, namely the supports of *v(q)* to compute it. In addition, the set Ω is not computed explicitly as in GBD which uses the Lagrange multipliers of the NLP2 and FP problems defined in Eq. (33) and Eq. (35) respectively.

The outer approximation technique identifies the optimal integer solution by iteratively solving a primal problem by fixing the integer variables at some value, and a master problem obtained by linearizing the nonlinear objective function and constraints. These two sub-problems respectively yield an upper and lower bound to the optimal solution and ultimately converge over successive iterations. This algorithm exploits the fact that the

primal problem, by fixing the integer variables, becomes an NLP and can be readily solved by NLP solvers like IPOPT. On the other hand, at every major iteration, *k*, the master problem is an integer linear programming problem (ILP) of the form:

Eq. (41) MOA

$$\min_{\alpha,q,y} \ \alpha$$
$$f(q^k, y^k) + \nabla f(q^k, y^k)\begin{pmatrix} q - q^k \\ y - y^k \end{pmatrix} \leq \alpha$$
$$g(q^k, y^k) + \nabla g(q^k, y^k)\begin{pmatrix} q - q^k \\ y - y^k \end{pmatrix} \leq 0$$
$$\sum_{i \in B^k} y_i^k - \sum_{i \in NB^k} y_i^k \leq |B^k| - 1$$
$$q \in \{0,1\}^n, y \in Y \subseteq \Re^n \ , \alpha \in \Re, k = 1, \dots, K$$

Where, y^k is the solution of the primal problem in Eq. (33) if feasible, otherwise it is the solution of the FP problem defined in Eq. (35). Here, the sets $B^k and\ NB^k$ denote the index sets that correspond to integer variables with values 1 and 0, respectively. Note that the first two constraints in MOA, namely the linearized cost and constraints are the supports of *v(q)* and used to characterize it. The third and the fourth constraints in MOA are used to implicitly define the set Ω. Specifically, the fourth constraint is called an *integer cut*, which removes all infeasible integer points from the set $\{0,1\}^n$ thereby capturing the set Ω.
It must be noted that the integer cut is a point wise elimination and can be ineffective for large scale problem, namely the 3D gasifier model. The feasibility pump variant mitigates this issue.

The schematic of the OA/GBD is shown in Figure 8. The algorithm involves iterative solution of the primal (NLP) and the master (MILP) problems to calculate the upper and lower bounds, respectively, until convergence of the upper and lower bounds. Note that the primal problem alleviates the integer constraints, while the master problem relaxes the nonlinear constraints, thereby using a combination of NLP and MILP to obtain the solution to the original MINLP problem using available mature solvers for the sub-problems. There are many commercial software packages such as GLPK and CPLEX for solving MILP problem and IPOPT for solving NLP problems.

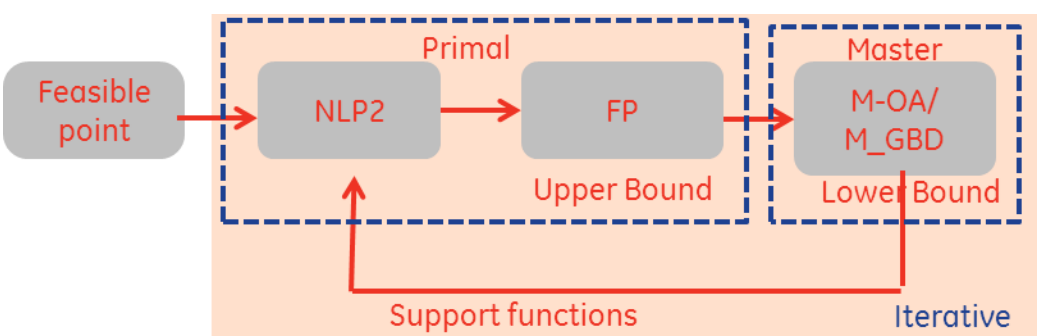


Figure 8: Schematic of the GBD and OA algorithms

Remarks

The important factors that determine the choice of the algorithm for OSP are the ease of use of the algorithm, computational efficiency and most importantly the availability of commercial software to solve each of the sub-problems.

- The OSP problem is a binary valued INLP which means that there are no continuous variables. Three observations can be made here:
    - This implies that the step of solving the primal problem does not apply and hence the upper bounds cannot be computed till an incumbent is obtained.
    - Even if the primal problem cannot be solved, the characterization of the set Ω is possible in OA given that it is done by integer cuts and the M-OA problem can be solved as a relaxation.
    - The GBD method is not suitable for the OSP problem for the simple reason that the Lagrange multipliers used in the definition of the master problem M_GBD are solutions to primal problem.
- Given the absence of primal solutions, the integer cuts are the only way to characterize the decision space, i.e., the set Ω. Each integer cut is a point elimination and at convergence, the intersection of the feasible regions of these integer cuts will be the same as the set Ω. Note that the integer cuts are weak cuts, i.e., they eliminate one integer point per iteration. This means that for problems with large number of integer variables, a large number of iterations may be required to characterize the set Ω accurately. For medium/large scale OSP problems, mechanisms that speed up the characterization of Ω need to be developed, one such mechanism being supporting hyper-planes.

One possible method to use OA for the large scale OSP problem would be to combine OA with a step for regaining feasibility and defining a supporting hyper-plane at this solution. The feasibility pump (FP) (Bonami et. al., 2006) based methods in existing literature have already solved the problem of finding a feasible solution to a MINLP problem and is delineated in the next section.

4.2.3 Outer Approximation with Feasibility Pump

The feasibility pump is a heuristic algorithm developed in order to find feasible solutions to the MILP and MINLP problems. Under assumptions of convexity of the feasible space, this algorithm becomes an exact algorithm. The integrated FP (IFP) method described in Bonami et. al, 2006, integrates the OA and FP algorithms and is an iterative method. Similar to the OA/GBD approaches, at every iteration the generation of an upper and lower bound is achieved by solving different kind of relaxed sub-problems. The potential benefit of the IFP as proposed in the existing literature is the speed of convergence to the integer optimal solution due to the increased ability to find feasible solution.

Two additional sub-problems are defined in IFP on top of the NLP2 (Primal) and M-OA problem.

1. The first one deals with regaining nonlinear feasibility by projecting an infeasible integer point to the feasible space, which is assumed to be convex. This problem relaxes the integral constraint and hence is called as FP-NLP problem. The FP-NLP is defined as

Eq. (42) FPNLP

$$\mathrm{Min}_q \|q - q^k\|$$
$$g(q, y) \leq 0$$
$$q \in [0, 1]^n$$
$$y \in Y \subseteq \Re^m$$

Where, $q^k \in \{0,1\}^n$,i.e, integer feasible.

2. The second one looks for a better solution than the best upper bound/incumbent over the integer space. Here the relaxation is in terms of the non-linearity and is a kind of outer approximation. This problem, called as FP-OA, is an MILP problem and is formulations are given as.

Eq. (43) FPOA

$$\min_q \|q - q^k\|$$
$$\nabla \mathrm{f}(\mathrm{q}^\mathrm{k}, y^k) \begin{pmatrix} q - q^k \\ y - y^k \end{pmatrix} \leq -\mathrm{f}(\mathrm{q}^\mathrm{k}, y^k) + UB^k - tolerance$$
$$g(\mathrm{q}^\mathrm{k}, y^k) + \nabla \mathrm{g}(\mathrm{q}^\mathrm{k}, y^k) \begin{pmatrix} q - q^k \\ y - y^k \end{pmatrix} \leq 0$$
$$q \in \{0,1\}^n,$$
$$\text{-}y \in Y \subseteq \Re^m$$

Where, $q^k \in [0,1]^n$,i.e, the relaxed integer feasible
A schematic of the algorithm is shown in Figure 9. At each iteration, the lower bound is computed using Master problem (MILP) and the upper bound is computed by NLP2 or Feasibility pump in case the NLP2 is infeasible. The NLP2 and the FPNLP requires NLP solvers while the OA and FP-OA use MILP solvers such as GLPK, CPLEX etc. For further details on the algorithm and computational benefits, the readers are referred to Bonami et. al 2006.

- It must be noted that IFP cannot be directly applied to the OSP problem, being an INLP problem with no real variables and thus no primal problem. However, the ability to regain feasibility at every step makes this the most viable choice for the OSP problem.

- The IFP, in addition, does not have solver requirements above and beyond the OA, but requires an additional MILP solver as compared to the BB method. It is expected that the computational effectiveness of this method will outweigh the added software requirement.
- Finally a solver for the OSP problem can be obtained by integrating the OA and the FPNLP sub-problems, where the solutions of the FPNLP can be used to compute the support functions in addition to the integer cuts in OA for faster convergence.

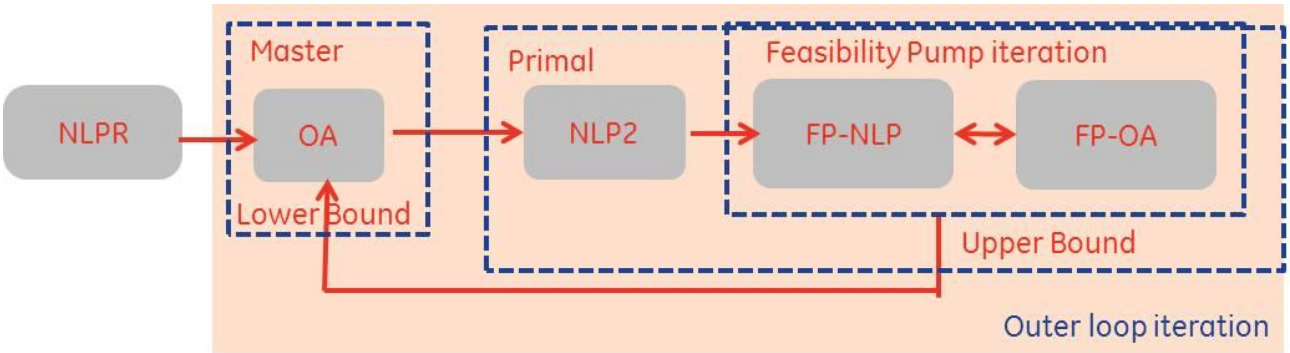


Figure 9: Schematic of the Integrated Feasibility Pump algorithm.

Other Methods

Many other variants of the OA such as the OA-Branch and Cut (OA-BC) and the OA-Hybrid (Bonami et.al. 2005) have been proposed in the existing methodologies for solving MINLP problems. While these methods work on similar assumptions as OA, the intended benefit has been mainly for the sake of improved computational efficiency. However, the choice of the methods depends on the specific problem. It is shown that for problems where the linearization is a good approximation and the integer relaxations are tight, OA-BC is a good choice. The OA-Hybrid combines the benefits of pure OA and pure BB algorithms. Hybrid between OA and pure BB is used for faster convergence by tightening the bounds from solving the NLP-R for certain iterations. Since the underlying concepts for the basic units have already been described, only a mention of these algorithms is done in this section for the sake of completeness.

To conclude this section, a comparison chart of the existing MINLP methods with respect to the different relaxations is shown in Figure 10. The two most commonly used relaxations are integer relaxation where the decision space is relaxed to continuous variables, and nonlinear relaxation which considers the linearization of the objective and constraint function. In the figure, the x axis relaxes the integrality and the y axis relaxes the nonlinearity. The choice of methods will be problem specific and will depend on which relaxation is most suitable.

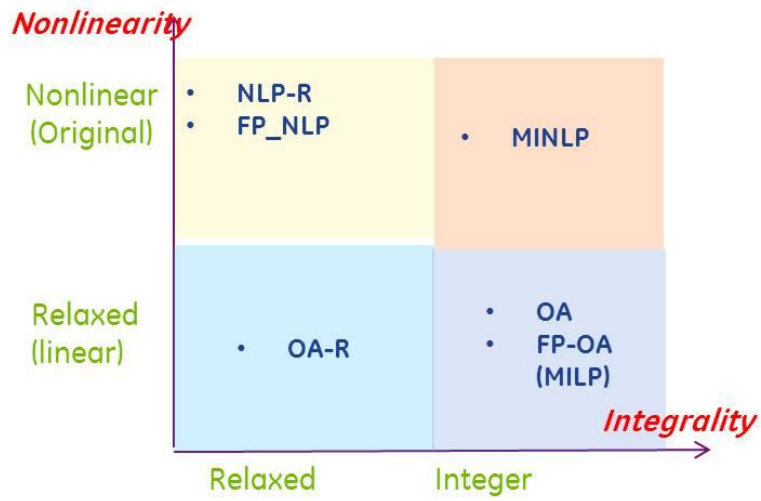


Figure 10: Comparison chart of existing MINLP methods with respect to different relaxations.

4.2.4 LMI Formulation for OSP Problem

Recall that OSP problem was formulated (Eq. (11)) as a linear cost optimization problem with precision and reliability constraints, where the precision constraint was evaluated using the solution to ARE (Eq. (8)). This formulation can be cast in LMI framework as proposed by Chmielewski et al (2002). They reformulated the general NLP of OSP into a convex program through the use of Linear Matrix Inequalities (LMIs). With the application of steady-state Kalman Filter for linear dynamic system, the technique developed by Chmielewski et al (2002) shows that the sensor-placement problem can be solved globally and efficiently using LMIs and branch-and-bound search algorithm.

4.2.4.1 Linear Matrix Inequality and ARE Problem

Consider the dynamic system in Eq. (3). If the system matrix A_k is nonsingular then the following matrix is positive definite

Eq. (44)
$$Y_0 = A_k P_{ss} A_k^T + Q$$

Substituting this into Eq. (7) leads to

Eq. (45)
$$P_{ss} = [Y_0^{-1} + C_k^T Q^{-1} C_k]^{-1}$$

Which, through the matrix inversion lemma, leads to:

Eq. (46)
$$P_{ss} = Y_0 - Y_0 C_k^T (C_k Y_0 C_k^T + Q)^{-1} C_k Y_0$$

Substituting the equality into the definition of Y_0 in Eq. (44) the following equality relation is obtained:

Eq. (47)
$$A_k Y_0 A_k^T - Y_0 - A_k Y_0 C_k^T (C_k Y_0 C_k^T + R)^{-1} C_k Y_0 A_k^T + Q = 0$$

With some appropriate technical conditions (see References Boyd et al. (1994) and Chmielewski et al (2002)), the above ARE is feasible if and only if there exists $Y > Y_0 > 0$ such that the following inequality holds, by using the matrix inversion lemma again.

Eq. (48)
$$-Y + Q + A_k (Y^{-1} + C_k^T R^{-1} C_k)^{-1} A_k^T < 0$$

To convert Eq. (48) into LMI, a positive definite matrix is defined

Eq. (49)
$$X = (A_k^T)^{-1} (Y^{-1} + C_k^T R^{-1} C_k) A_k^{-1}$$

Since $Y^{-1} > 0$ and A_k is nonsingular. From X one can find $Y = (A_k^T X A_k - C_k^T R^{-1} C_k)^{-1}$, and substituting it into inequality Eq. (48) and using standard Schur Complement Theorem, the inequality Eq. (48) is cast into LMI as follows:

Eq. (50)
$$\begin{bmatrix} X - A_k^T X A_k + C_k^T R^{-1} C_k & X \\ X & X + Q^{-1} \end{bmatrix} > 0$$

Therefore, the existence of a steady-state Kalman Filter by solving ARE in Eq. (7) is converted into the feasibility problem in LMI in Eq. (50).

4.2.4.1 Performance criteria and Linear Matrix Inequality

By using the definitions of Eq. (46), Eq. (49) and the relationship $Y > Y_0 > 0$, it is easy to conclude that $P_{ss} \leq (A_k^T X A_k)^{-1}$. Hence, the estimation precision in Eq. (6) can be cast into the form of LMI as follows:

Eq. (51)
$$C_i^e (A_k^T X A_k)^{-1} C_i^{eT} < \bar{s}_i$$
$$\begin{bmatrix} \bar{s}_i & C_i^e \\ C_i^{eT} & A_k^T X A_k \end{bmatrix} > 0$$

With the integer nonlinear programming (INLP) problem defined in Eq. (11) can be converted into convex optimization problem over LMIs with estimation precision constraints:

Eq. (52)
$$\min_{q_i} c_i q_i$$
$$\begin{bmatrix} X - A_k^T X A_k + C_k^T R^{-1}(q_i) C_k & X \\ X & X + Q^{-1} \end{bmatrix} > 0$$
$$\begin{bmatrix} \bar{s}_i & C_i^e \\ C_i^{eT} & A_k{}^T X A_k \end{bmatrix} > 0$$
$$q_i = \{0,1\} \quad for\ i = 1,2,\dots,n$$

Again, note that $R^{-1}(q_i)$ is linear function with respect to decision variable q_i as defined in Eq. (12). Therefore, Eq. (52) is a minimization of a linear objective under LMI constraints.

To include reliability constraint, Chmielewski et al (2002) define a slack variable to relax the nonlinear indicator $I(w_k)$. Recall that the function $I(\omega_k)$ denoting whether the reduced sensor set allows meeting the precision accuracy($I(\omega_k) = 1$), or not($I(\omega_k) = 0$). Thus, for $0 \leq \epsilon_{ik} \leq 1$, the condition

Eq. (53)

$$s_i(\omega_k) < \frac{\overline{s_\iota}}{\epsilon_{ik}} \Leftrightarrow s_i(\omega_k)\epsilon_{ik}^2 < \overline{s_\iota}\epsilon_{ik}$$

can always be met as long as $s_i(\omega_k) < \infty$. This amounts to relaxing the reliability requirement in Eq. (11), so that it can be converted into LMIs:

Eq. (54)

$$\sum_{\omega_k \in \Omega} \Pr(\omega_k)\, \epsilon_{ik} > \overline{r_\iota}$$

$$\begin{bmatrix} \overline{s_0} + \overline{s_\iota}\epsilon_{ik} & \epsilon_{ik} {C_i}^e \\ C_i^{eT}\epsilon_{ik} & {A_k}^T X A_k \end{bmatrix} > 0$$

Where Chmielewski et al (2002) add a small constant, $\overline{s_0}$, to make Eq. (53) well posed in Eq. (54) and avoid singularity. As discussed previously, the set Ω of all sensor failure combinations to be considered can be very large, and the calculation of Eq. (54) becomes very expensive. Therefore, it is changeling to approximate the reliability constraint in the LMI constraint. Instead the function reliability constraint (Eq. (9)) will be used. With that the solution to OSP involves solving the following problem:

Eq. (55)

$$\min_{X, q_i, n_i} c_i q_i n_i$$

$$\begin{bmatrix} X - A_k^T X A_k + C_k^T R^{-1}(q_i) C_k & X \\ X & X + Q^{-1} \end{bmatrix} > 0$$

$$\begin{bmatrix} \overline{s_\iota} & C_i^e \\ C_i^{eT} & {A_k}^T X A_k \end{bmatrix} > 0$$

$$r_l^h = 1 - \prod_{i=1}^{n} [1 - r_i^s q_i]^{n_i} \geq \overline{r_l}$$

$$q_i = \{0,1\}, for\ i = 1,2, \dots, N;\ n_i = 1,2, \dots, k.$$

Where, *N* is the total number of sensor locations n_i is the number of sensors at each location $i = 1, \dots, N$, and r_i^s is the intrinsic reliability of each sensor i used to monitor process *I*; and r_l^h denotes the overall reliability of the sensor network for estimating or monitoring the l^{th}process variable

With LMI formulation above, the original NLP problem was converted into a convex optimization problem in LMIs as in Eq. (52) with binary decision variables q_i. since q_i 's are linear/convex in LMIs, they can be relaxed into real values so that variables $q_i \in [0,1]$ solved from LMIs won't have impact on the LMI feasibility. In other words, because of the convexity, the LMIs are feasible for a certain set of q_i taking values $(0,1]$ if and only if the LMIs are feasible for that set of q_i taking values $q_i = 1$. In the sensor network design procedure, a branch-and-bound algorithm with LMI-constrained minimization problems can be used to solve the sensor placement problem.

The way branch-and-bound algorithm works with LMIs is standard. LMIs solve for real decision variables $q_i \in [0,1]$ at each bud node, and branch-and-bound turns one of the decision variable into 0 or 1 one at a time to arrive at a child node where another LMI is solved for the non-integer components q_i, and so on. To make the searching more efficient, branch-and-bound chooses the largest q_i from the LMI solution as the next searching node. At the bud nodes, LMI gives the lower bound of the cost since q_i takes fractional value between 0 and 1, while a leaf node where all decision variables have binary integer values provides the best upper bound if it is the incumbent. If the lower bound at a bud node is higher than the upper bound from the incumbent, then the bud node is discarded from the search. Progressively, the lower bound and upper bounds converge and the optimal solution is found.

With the branch-and-bound algorithm, the optimal sensor placement problem can be solved by minimizing the cost with guaranteed precision as the convex optimization problem in LMIs in Eq. (52). To impose the reliability constraints a few practical implementations was proposed. The simplest one is to check the reliability at the leaf node when the decision variables are decided as 0 or 1. If the reliability is satisfied, the leaf node is saved and the upper bound of the cost is updated. Otherwise, the node is discarded as infeasible solution and branch-and-bound algorithm is forced to expand more bud nodes. Since there is no reliability constraint before it reaches the leaf node, the searching criteria won't be optimal and will be expensive for large scale problems. Other practical approaches/approximations to reliability will be sought to enable including it in the LMI/Branch-and-bound framework.

Remarks

The LMI formulation for OSP problem is easy to apply, computationally tractable and efficient for small size problem. However the LMI technique has some practical limitations:

- One limitation is the size of the optimization problem. When the size of the system increases, including the size of the system states, the sensor type, and the sensor location, the solvability of the existing commercial solvers and non-commercial solvers for LMIs are subject to further test.
- The other concern is the conservativeness of the LMIs. By converting the precision constraint into convex optimization, the LMI imposes a stricter constraint. The feasible solution certainly guarantees the required precision, but the other way might not be true. It would be necessary to investigate numerical and practical method to make the constraint less conservative.

4.3 Model Enhancement for Gasifier Refractory Wear

As mentioned earlier that the key focused on this program was on optimal sensor placement for condition monitoring of the gasifier refractory wear and RSC fouling profile. The direct measurement of the quantities of interest (wear and fouling) for the condition monitoring of the gasifier refractory wear as well as that of RSC fouling are very difficult due to harsh condition in which they operate. Hence a model based sensing approach is used as shown in Figure 11, that utilizes a physics based model of the process to relate sensed signals (e.g., temperature) to the signal of interest (e.g., refractory wear).

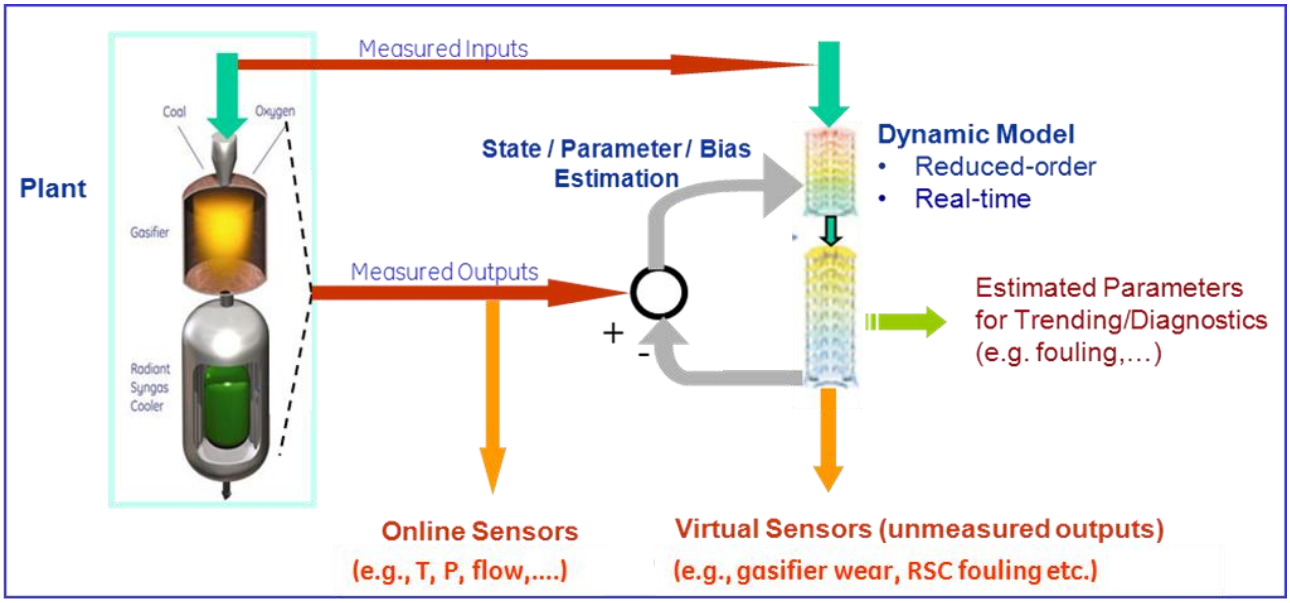


Figure 11: Typical model based sensing scheme

First-principles physics-based models allow capturing the process knowledge through rigorous mass and energy balances, and utilizing it systematically for improved online model-based sensing and controls. In the face of limited online sensors, first-principles models allow relating process variables across the entire system, and can thus be used as the basis for online model-based estimation or “virtual” sensing of key unmeasured process variables. With this motivation, one of the key objective of Task 2 was to obtain a suitable dynamic model of the gasification section in Matlab/Simulink® that could be used for model base optimal sensor placement as well as real-time transient simulations in Tasks 3 and 4.

In a previous program (DE-FC26-07NT43094), a dynamic model of the core gasification section of the IGCC plant was implemented for advanced model-based controls design. For pre-heating of the gasifier during startup, a 1-dimensional heat transfer model of the refractory lining was developed to simulate the transient thermal gradients in the radial direction (assuming symmetry in axial and azimuthal directions) and the resulting thermal strains. On the other hand, for nominal plant operation, the gasification reactions were modeled in detail, while a simple quasi-steady-state thermal model of the refractory lining was used to calculate a small heat transfer loss.

In the current program, the key focus is on monitoring the wear of the gasifier refractory lining on the hot inner surface. Figure 12 shows a schematic representation of the gasifier refractory lining starting with uniform cylindrical profile on the cold outer and hot inner surfaces for a new lining. Over time, the hot inner surface wears non-uniformly, characterized by a growing inner surface radius $\Delta r(z, \theta)$ that varies with the axial location z and azimuthal location θ. This non-uniform wear leads to a non-uniform thermal profile in the refractory lining, which can be monitored through temperature sensors located in the refractory lining to estimate the current wear profile. To enable this model-based estimation, a 3-dimensional heat transfer model is needed that allows asymmetric temperature profiles.

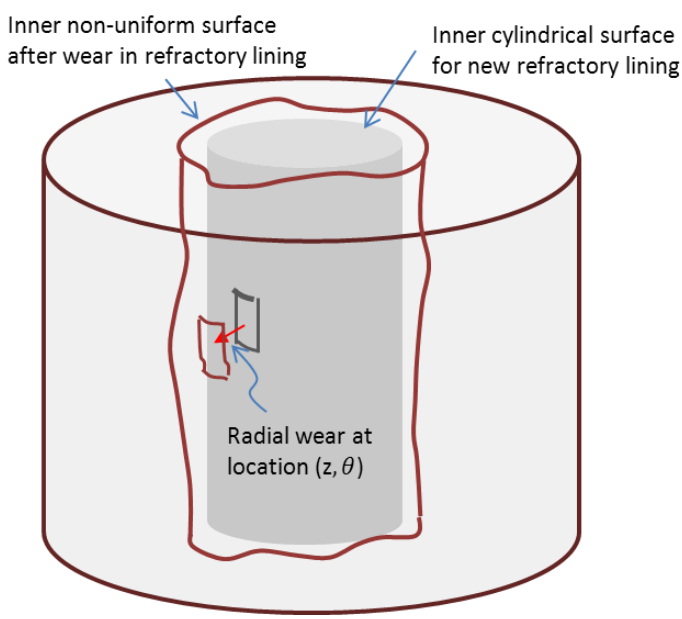


Figure 12: Non-uniform wear on hot inner surface of gasifier refractory lining

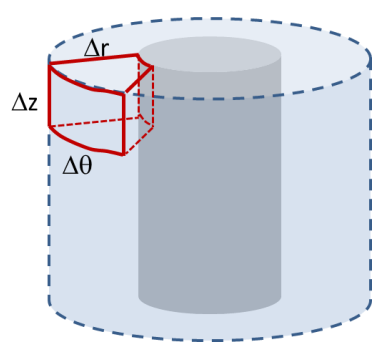


Figure 13: Nodes for 3-dimensional heat transfer model in gasifier refractory lining.

In particular, in the main cylindrical section of gasifier refractory lining, the 3-dimensional heat transfer is described by heat transfer through conduction in the refractory material. Figure 13 shows a node in cylindrical coordinates in the cylindrical section of the gasifier

including the three layers of refractory bricks as well as the metal shell. The dynamic heat balance is described by the second-order partial differential equation (PDE):

Eq. (56)

$$\begin{aligned}\left(\frac{\rho C_p}{k}\right)\frac{\partial T}{\partial t} &= \left\{\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2}+\frac{\partial^2 T}{\partial z^2}\right\}\\ &= \left\{\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2}+\frac{\partial^2 T}{\partial z^2}\right\}\\ &= \nabla^2 T\end{aligned}$$

The above PDE was implemented using finite differences with approximately 30 nodes in the axial direction z, 20 nodes in the azimuthal direction θ and 15 nodes in the radial direction r, yielding a total of 9000 nodes or temperature states.

Eq. (57)

$$\begin{aligned}\rho C_p \Delta V\frac{dT}{dt} = k&\left\{\left(\frac{T_{r-}-T_r}{\Delta r}\right)r\Delta\theta\Delta z-\left(\frac{T_r-T_{r+}}{\Delta r}\right)(r+\Delta r)\Delta\theta\Delta z\right\}\\ &+k\left\{\left(\frac{T_{\theta-}-T_\theta}{r\Delta\theta}\right)-\left(\frac{T_\theta-T_{\theta+}}{r\Delta\theta}\right)\right\}\Delta r\Delta z\\ &+k\left\{\left(\frac{T_{z-}-T_z}{\Delta z}\right)-\left(\frac{T_z-T_{z+}}{\Delta z}\right)\right\}r\Delta r\Delta\theta\end{aligned}$$

The heat transfer in the core of the system is governed by thermal conduction, using temperature dependent thermal conductivity k for each brick/metal material. The heat transfer from the hot gas in the gasifier to the hot inner refractory surface is governed by a combination of radiation and convection, while the heat transfer from the outer surface of the metal shell to the ambient is governed by natural convection. The heat transfer to the top and bottom sections of the gasifier are modeled by conduction to the respective top/bottom sections with the same 15 radial nodes assumed to be uniform in azimuthal and axial directions.

In the discretized model in Eq. (57), the volume of the brick material is denoted by ΔV. For a fresh brick lining, this volume is the full node volume. However, as the inner brick wears, i.e. the hot inner surface moves radially outwards, this volume decreases to a fraction of the grid cell until it goes to zero as the wear progress from the first radial node to the next. This reduction in the brick volume in a cell corresponding to varying wear level, leads to problems with “stiff” dynamics as the brick volume in a cell becomes too small, in turn leading to numerical simulation problems. To overcome this issue, once the brick volume in a cell reduces to a chosen threshold, its dynamics is approximated by a quasi-steady-state condition, thereby allowing easy numerical simulation without sacrificing significant accuracy.

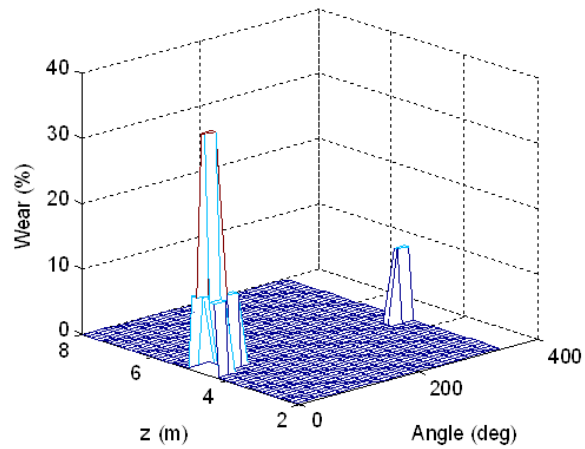


Figure 14: Example non-uniform/local wear profile on inner refractory brick layer.

This model was implemented and simulated for various uniform and non-uniform wear profiles. An example non-uniform wear profile is shown in Figure 14, where a few nodes at mid axial location $z = 5$ and around $\theta = 0$ have a localized wear profile up to ~35% of the inner-most brick layer thickness, while the rest of the bricks have zero wear. The simulation was started with initial condition at a steady-state profile with no brick wear, and the brick wear increased linearly from zero wear at t=5hr to the specified wear profile at t=15hr.

Figure 15 shows the profiles for the temperature difference from initial steady-state values at the brick interfaces and metal shell as a function of the axial and azimuthal locations. The profiles for layers 1-3 denote the temperature profiles on the outer surfaces of each refractory brick layer, while layers 4 and 5 show the metal shell inner and outer surface temperatures respectively. Clearly, the temperature profiles show sharp peaks in the inner layers, which gets diffused to shallow and more spread out peaks in the outer layers, as expected.

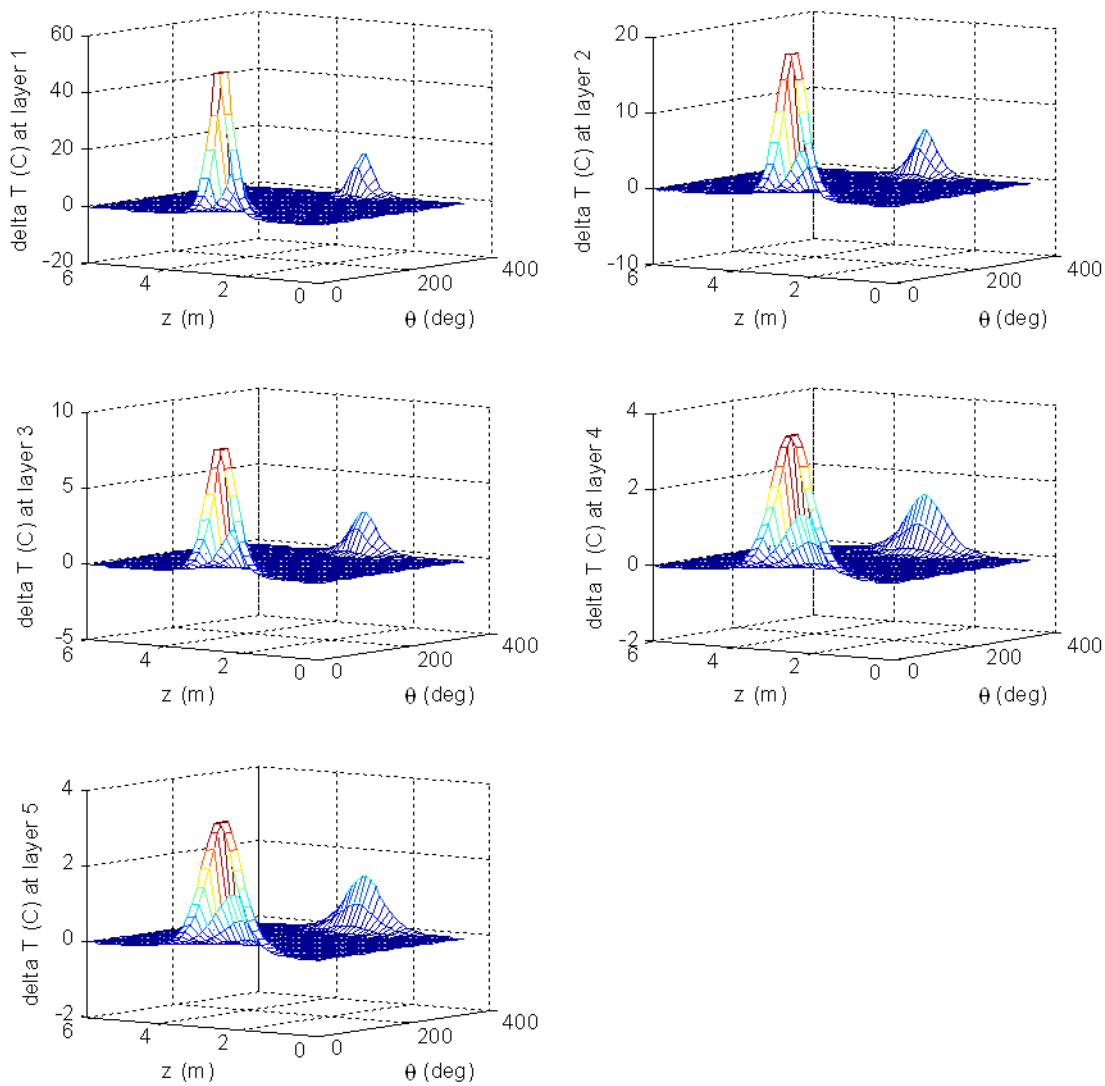


Figure 15: Temperature profiles at brick interfaces and metal shell for local wear profile.

For the same simulation run, the transient temperature profile at all 15 radial node locations (numbered from innermost to outer metal) are shown in Figure 16 for two axial/azimuthal locations. The plot on the left shows the temperature profile for node location $i_z = 22, i_\theta = 1$, where the radial wear increases from zero at t=5hr to ~35% of brick thickness at t=15hr. The plot on the right shows the temperature profile for node location $i_z = 24, i_\theta = 1$, where the radial wear increases from zero at t=5hr to ~11% of brick thickness at t=15hr. Note that for the left plot, as the radial wear progresses from the innermost node (blue) to the second(green) and ultimately to the third (red) node, the temperatures from the first and second radial nodes decrease and ultimately become the same as the 3rd node due to the quasi-steady-state condition, as expected. On the other hand, for the plot on right, the wear at the end of the simulation run is still limited to the first radial node (blue) and all radial nodes show a monotonic increase in temperature.

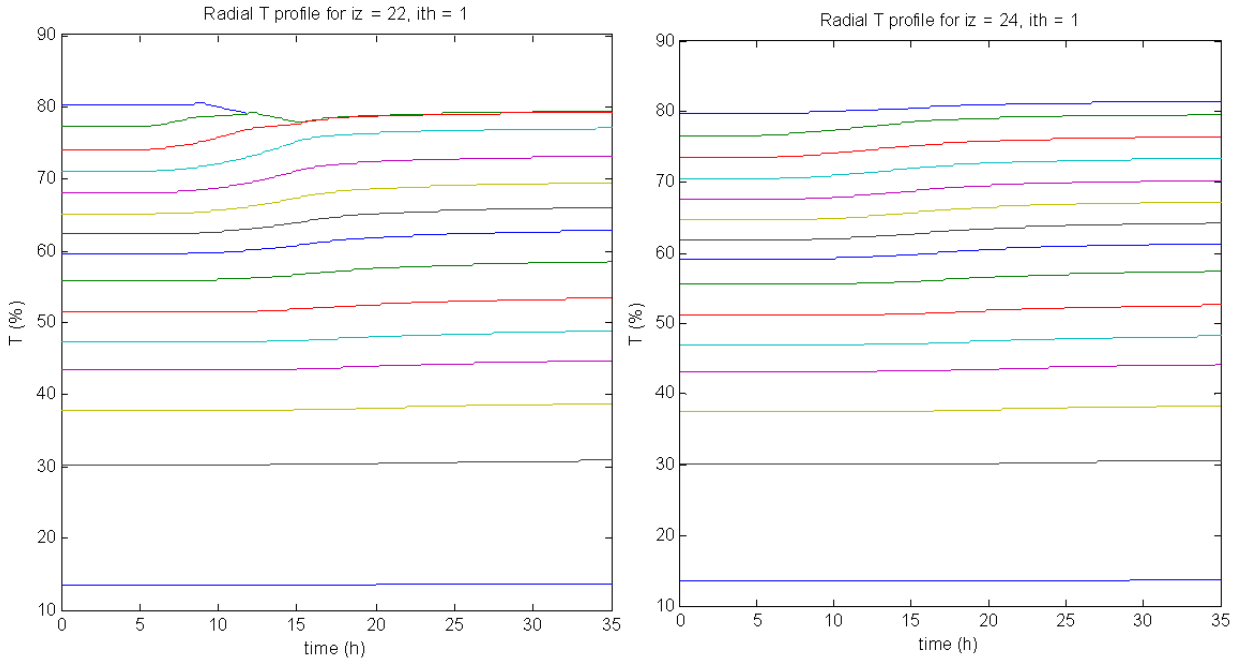


Figure 16: Transient temperature profiles across radial nodes at different axial and azimuthal locations**.**

The above simulation plots show that the model is running well and can be simulated for varying non-uniform wear profiles and used to simulate the plant for OSP studies. However, the model is of very high order (~9000 states) and takes significant time for simulation. Also, the nonlinear model was linearized to deriver linear state-space model for use in OSP, e.g. for linear Kalman filter or EKF implementation. However, owing to the large number of states, standard numerical perturbation techniques will result in large matrices and will take significant time for perturbing each state. To address this, sparse matrices and sparse linearization techniques have been used. For the full nonlinear dynamic model given by nonlinear ODEs:

Eq. (58)

$$\dot{x} = f(x, u)$$
$$y = h(x)$$

Where, x denotes the vector of all node temperature states, u denotes the vector of radial wear profile $\Delta r(z, \theta)$, and y denotes the vector or potential sensor measurements of temperature at each brick layer outer surface, the corresponding linear model has the form:

Eq. (59)

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \quad (D = 0)$$

Owing to the spatial nature of the model, i.e. only neighboring nodes interact through thermal conduction, the above linear model will have a sparsity pattern. Figure 17 shows the sparsity pattern for the combined linear model matrix $[A\ B; C\ D]$ $(where\ D = 0)$, which is clearly very sparse, with less than 0.5% of the entries being non-zero. This allows

significant saving. Also, the numerical perturbation linearization can be done very efficiently using sparse linearization techniques. In particular, exploiting the structural orthogonality of the columns, corresponding orthogonal combinations of (x, u) can be perturbed together rather than one variable at a time. For the sparsity pattern shown here, only 26 perturbations were needed to evaluate the linear system matrices.

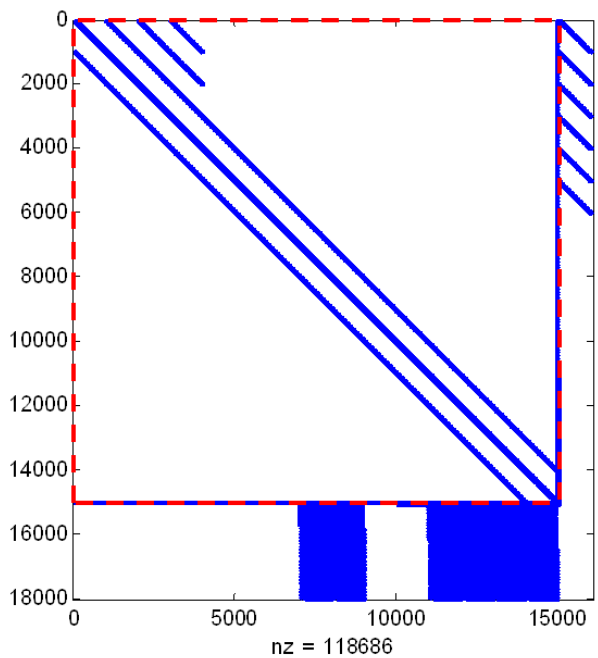


Figure 17: Sparsity pattern of linear model for gasifier refractory.

4.4 Model Enhancement for RSC Fouling

A 1-D dynamic generic model of the Radiant Syngas Cooler (RSC) was developed in the previous program (DE-FC26-07NT43094), where the RSC was modeled to have axial temperature distributions for the syngas and for the water inside the RSC tubes. In that model, fouling on the RSC tubes, which decreases the heat transfer between the hot syngas and the cold water inside the riser tubes thereby decreasing the high-pressure steam production from the RSC drum, was modeled to be uniform along the length of the RSC. In addition, for startup pre-heating operation, the model was modified to include thermal transients in the shell and the tubes and the resulting stress and strain on the support structure due to the thermal gradients and weight of the RSC tubes. In that program, it was shown that this uniform fouling level was “observable” using the existing set of online sensors.

The assumption of the uniform fouling is a simplification of the actual process that takes place in an IGCC plant. In the current program, the assumption of uniform fouling was relaxed to allow for non-uniform fouling profile along the RSC length, i.e., allow different levels of fouling at different axial locations in the RSC. This increases the challenge in monitoring the non-uniform fouling profile, which is in all likelihood not feasible with the existing set of sensors in the gasification section. To address this, inclusion of additional

online sensors was explored. In particular, use of heat flux sensors at few key locations along the RSC length, together with strain sensors on the mechanical support structure that provide information about the cumulative fouling weight was explored. While the heat flux profile is already modeled along the length of the RSC, the model had to be updated to allow for non-uniform fouling and the cumulative effect of the fouling weight on the strain sensors.

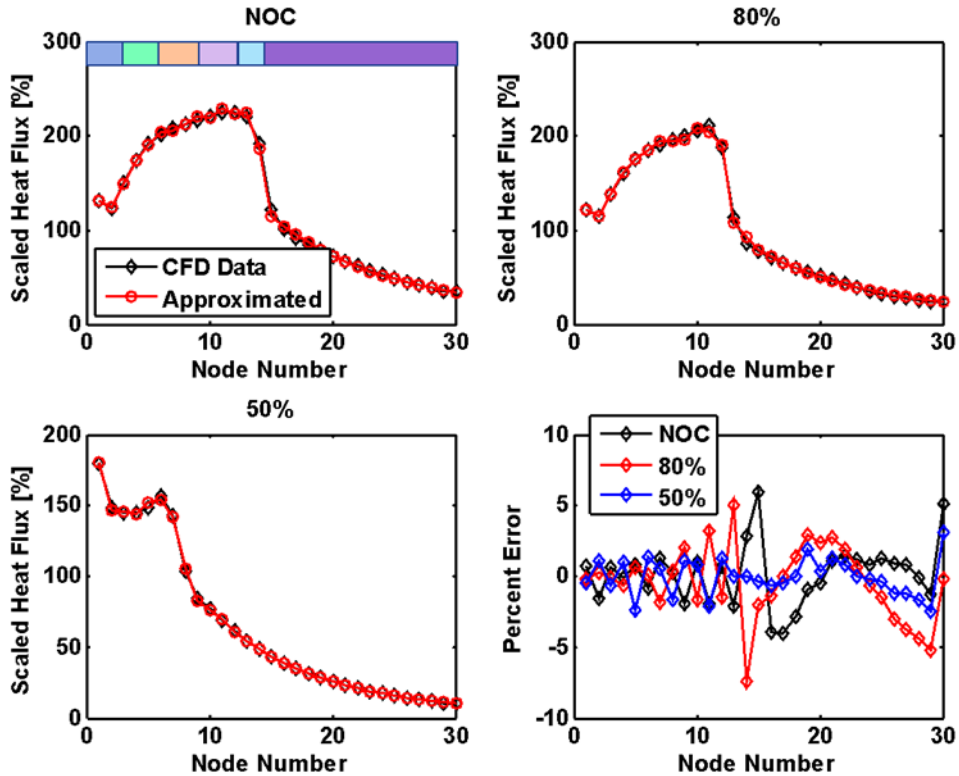


Figure 18: Fitting the RSC model heat fluxes to CFD data by parameterizing the heat transfer coefficients for the un-fouled RSC tubes in case of: NOC, 80% throughput, and 50% throughput. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the percent error in the heat flux fits. The heat flux trends shown here are for a generic model and do not represent any specific site data.

The heat transfer coefficients and the effect of fouling on these coefficients were previously parameterized with respect to flow and the axial location matching 1-D model heat fluxes to heat flux values obtained from a more detailed 3-D computational fluid dynamics model. This analysis was also improved to account for the axially non-uniform fouling. In particular, the heat transfer length was divided into six zones based on the heat transfer profiles to obtain better match along the length for nominal and fully-fouled conditions. The comparison of heat fluxes obtained from the parameterization of the heat transfer coefficients in the 1-D model with six zones and the heat fluxes obtained from a 3-D computational fluid dynamics (CFD) model is presented in Figure 18 for different throughput conditions: normal operating conditions (NOC), 80% throughput, and 50% throughput. The heat flux profile comparisons to the CFD data for the fully-fouled RSC tubes are presented in Figure 19. As can be seen in Figure 18 and Figure 19, with multiple zones having different heat transfer coefficient parameterizations 1-D RSC model's heat flux agrees very well with the CFD data.

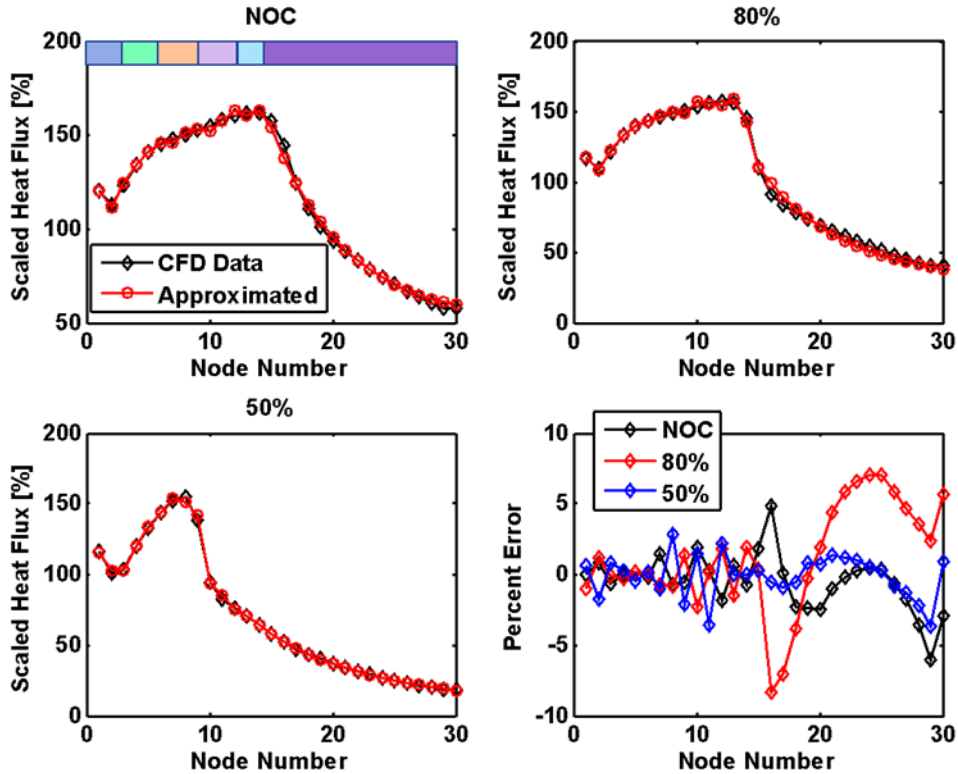


Figure 19: Fitting the RSC model heat fluxes to CFD data by parameterizing the heat transfer coefficients for the fully fouled RSC tubes in case of: NOC, 80% throughput, and 50% throughput. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the percent error in the heat flux fits. The heat flux trends shown here are for a generic model and do not represent any specific site data.

In accordance with the multiple zones utilized for identifying heat transfer coefficient parameterizations, a different fouling scale factor is used for each individual zone. These scale factors vary from 0 to 1, where 0 corresponds to no fouling on the tubes and 1 corresponds to fully fouled RSC tubes. This allows for a non-uniform fouling with independent fouling factors for each of the six zones. The effect of fouling on the temperature profiles inside the RSC is presented in Figure 20. It can be seen that the effect of fouling on the temperature profiles can be significant and can create a difference as much as 20% on the exit temperature between the no fouled and fully fouled cases. In three of the cases presented in Figure 20, fully fouled, half fouled, and no fouling cases, the fouling is uniform across the RSC. As expected, among these cases the no fouling case has the lowest temperature profile (corresponding to most heat extracted from the syngas); fully fouled case has the highest temperature profile; and the temperature profile of the half fouled case falls in between these temperature profiles. Two non-uniform cases are also presented in Figure 20: bottom fouled and top fouled cases, where the bottom three and top four zones have fouling scale factors of 1, respectively, while the other zones are un-fouled. The top half of the nodes in these cases align with the fully fouled and no fouling cases, as expected while the rest of the RSC has different temperature profiles in between the fully-fouled and un-fouled extremes.

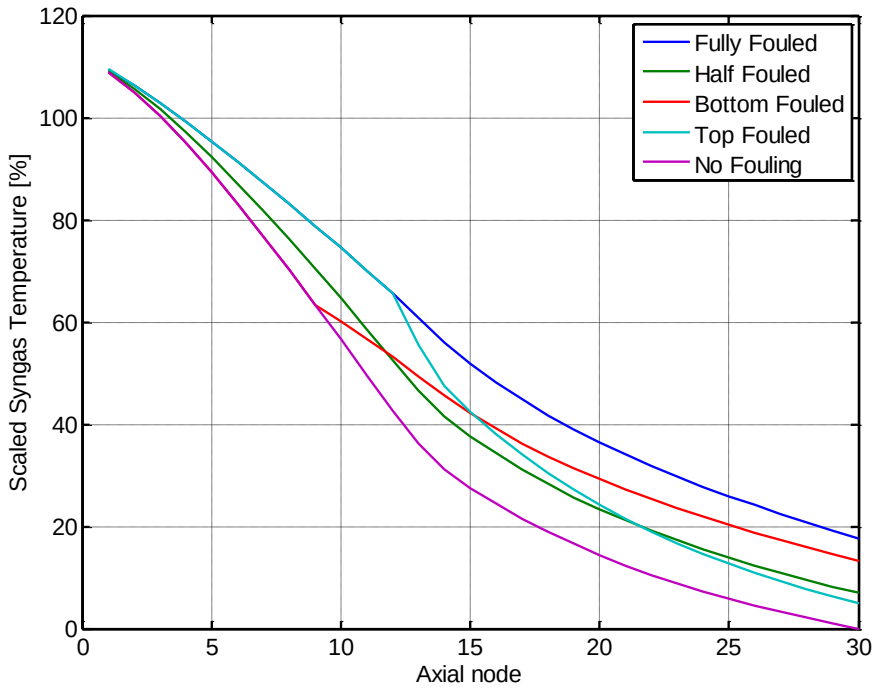


Figure 20: Effect of fouling on syngas temperature profiles inside the RSC. Fully fouled, half fouled, and no fouling cases have uniform fouling while bottom fouled and top fouled cases have non-uniform fouling along the RSC. 1st node corresponds to the top of the RSC and the 30th node corresponds to the bottom of the RSC.

Since fouling on the RSC tubes decreases the heat transferred from the hot syngas to the water inside RSC tubes, the net effect of fouling is present in the amount steam that can be extracted from the RSC drum at a given drum pressure. Figure 21 shows the scaled steam flow from the RSC drum (deviation from the lowest flow for fully fouled case) for different cases of uniform and non-uniform fouling as well as a transient case from no fouling to full uniform fouling. The variation in steam flow is in negative correlation with the degree of fouling on the RSC tubes, as expected.

Heat flux, exit syngas temperature, and RSC drum steam flow are all measurements that can be utilized to track the non-uniform fouling of the RSC tubes. These three classes of measurements can be utilized to determine the extent of fouling through its effect on the heat transfer inside the RSC and they are defined within the updated RSC model. Another measurement that can also be utilized for this purpose is the strain experienced by the mechanical structure at the top in the RSC that supports the weight of the tubes, where the extent of fouling along the entire tube length can be tracked by the increasing strain on the RSC bolts due to the additional weight brought on by the fouling. The strain experienced on this support structure, in terms of added strain from the baseline no-fouling case, is presented in Figure 22 for the same fouling cases of Figure 21. The strain measurement is also defined within the new RSC model.

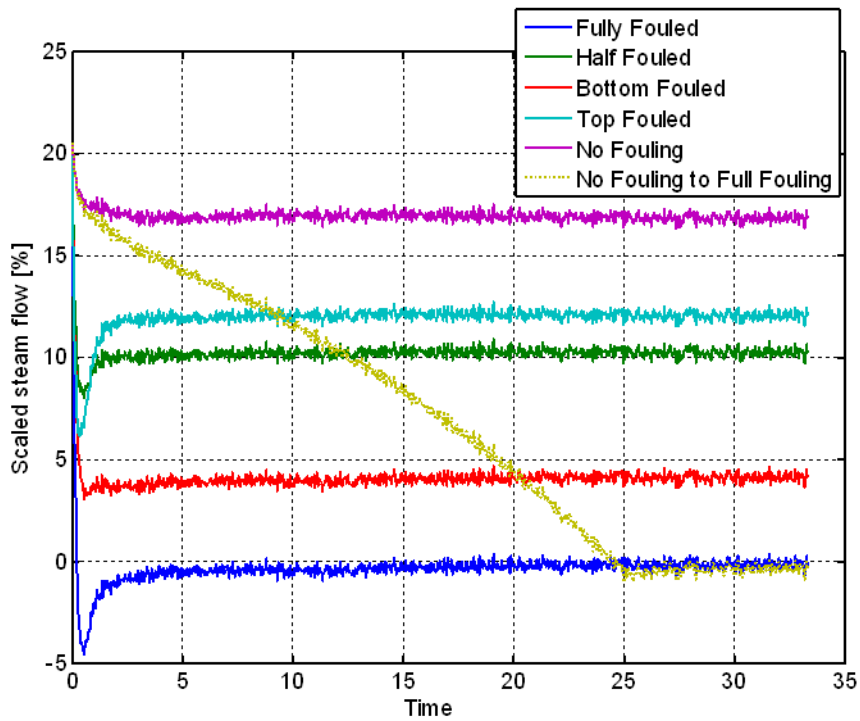


Figure 21: Effect of fouling on the amount of steam flow from the RSC drum.

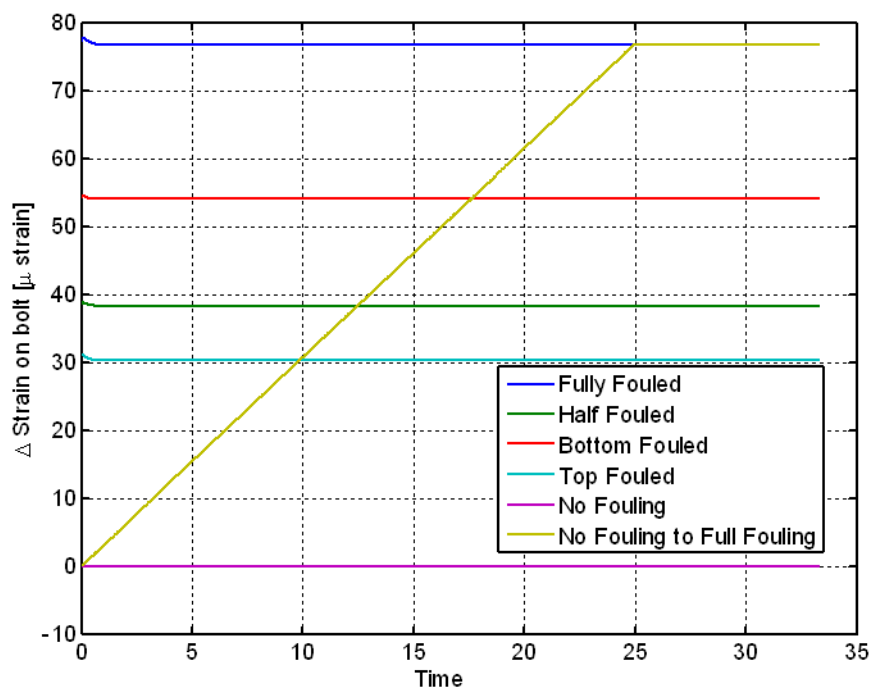


Figure 22: Effect of fouling on the strain on the RSC bolt.

To summarize, the modifications to the RSC model included: (i) improving the heat transfer coefficient correlations for a better fit to CFD data with six zones along the RSC length, (ii) increasing the number of fouling scale factors to allow for non-uniform fouling, one independent factor for each of the six zones, (iii) modeling the effect of cumulative fouling

weight in all the zones on the support structure and resulting strain, and (iv) including heat flux and strain measurements to allow for evaluating the use of these sensors within OSP for RSC fouling monitoring. In order to reduce the size of the gasifier-RSC model, the slag model was simplified. In particular, the previous slag model with a spatial resolution in the axial as well as radiant directions and detailed dynamics of the material and energy balance in the IGCC plant was updated with a coarser spatial resolution model. While the previous slag model provides detailed transient evolution of the liquid and solid slag thickness along the gasifier length, for OSP problem this level of resolution was not required, further this simplification reduces the number of the states and hence the complexity and size of the optimization problem. Also, reducing the order of the slag model has insignificant impact on the steady state gasifier temperature and syngas exit temperature to RSC which are signature of fouling conditions. As expected, with this simplification the error between critical responses (e.g., syngas temperature out of RSC exit, bolt strain, heat fluxes across the RSC tube etc.) for the linear models derived from this simplified model and that of the full nonlinear models were less than 2%.

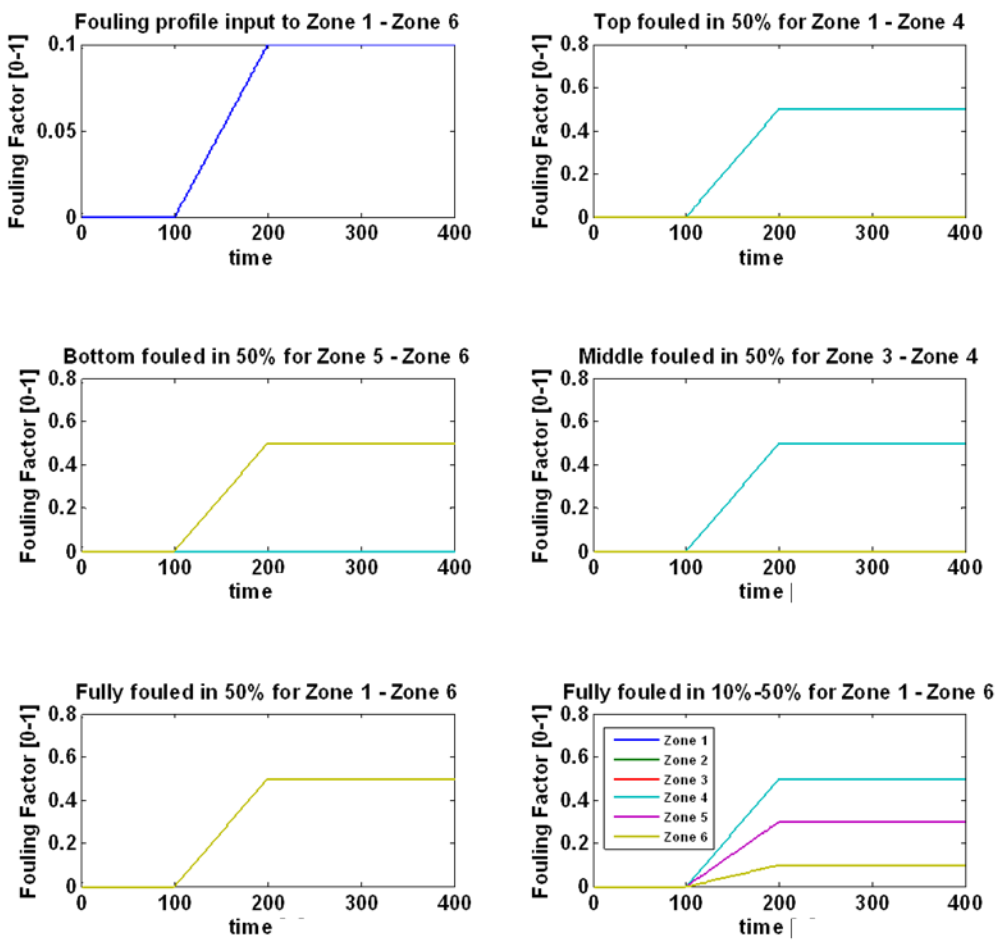


Figure 23: The fouling profiles used in simulation.

In order to apply the OSP algorithms, the model linearization was carried out for the enhanced RSC model which allows for the non-uniform fouling profiles and the cumulative effect of the fouling weight on the strain sensors. The linearization is conducted at steady

states conditions for un-fouled RSC tubes. In the following comparisons, all the linear modes were generated at no fouling condition by setting the fouling factors for all sections to zero. Then the simulations were conducted by applying different (small, large, uniform, non-uniform) fouling profiles as inputs, as shown in Figure 23. All the fouling profiles are modeled as ramps of 100 seconds starting at 100 second to the final value at 200 second. For example, the left top one is the fouling profile which is given to each zone for small non-uniform fouling test, and the rest of the cases are the fouling profiles which are given to specific zones for larger non-uniform/uniform fouling tests.

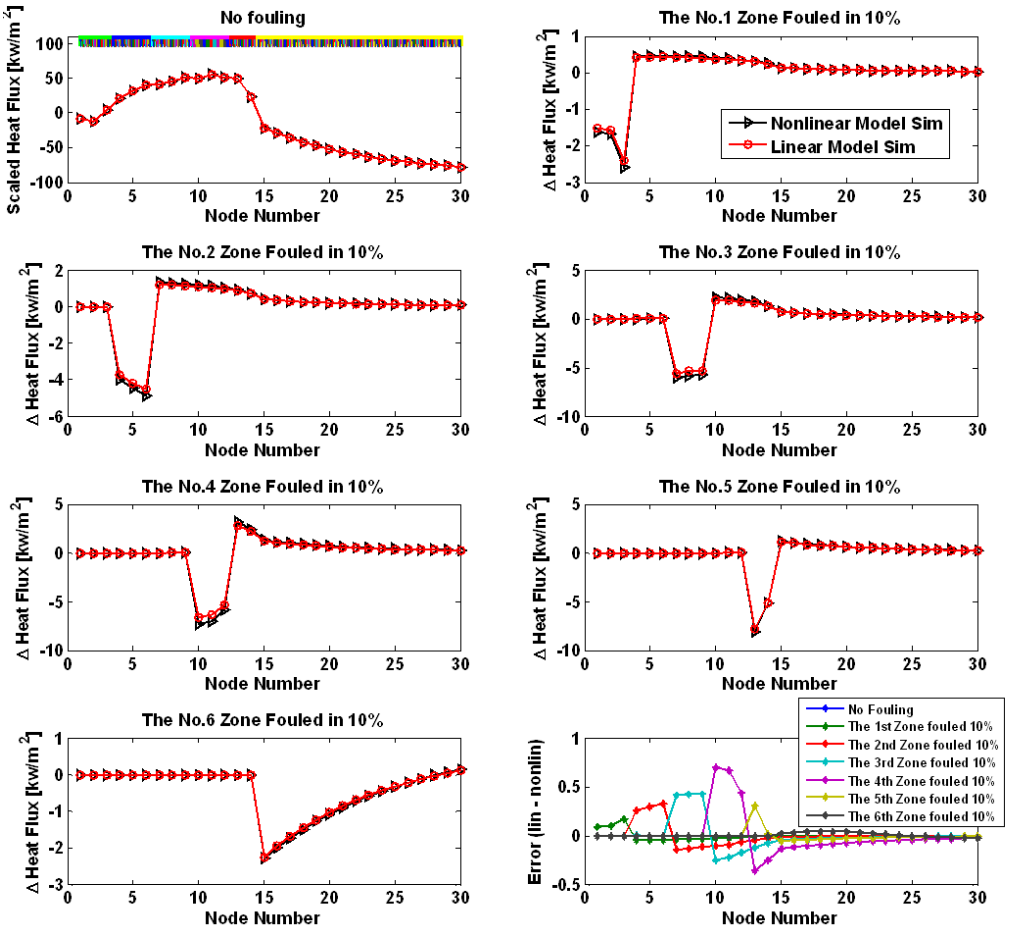


Figure 24: The comparison of the RSC model heat fluxes between linearized model and nonlinear model for un-fouled and small non-uniform fouling (10%) at different zones. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the error in term of deviation for the linear model from the nonlinear model.

The comparison of heat fluxes obtained from the parameterization of the heat transfer coefficients in the nonlinear RSC model with six zones and the heat fluxes obtained from the linearized model is presented in Figure 24 for various fouling profiles: Zero (no) fouling and 10% fouling for each zones.

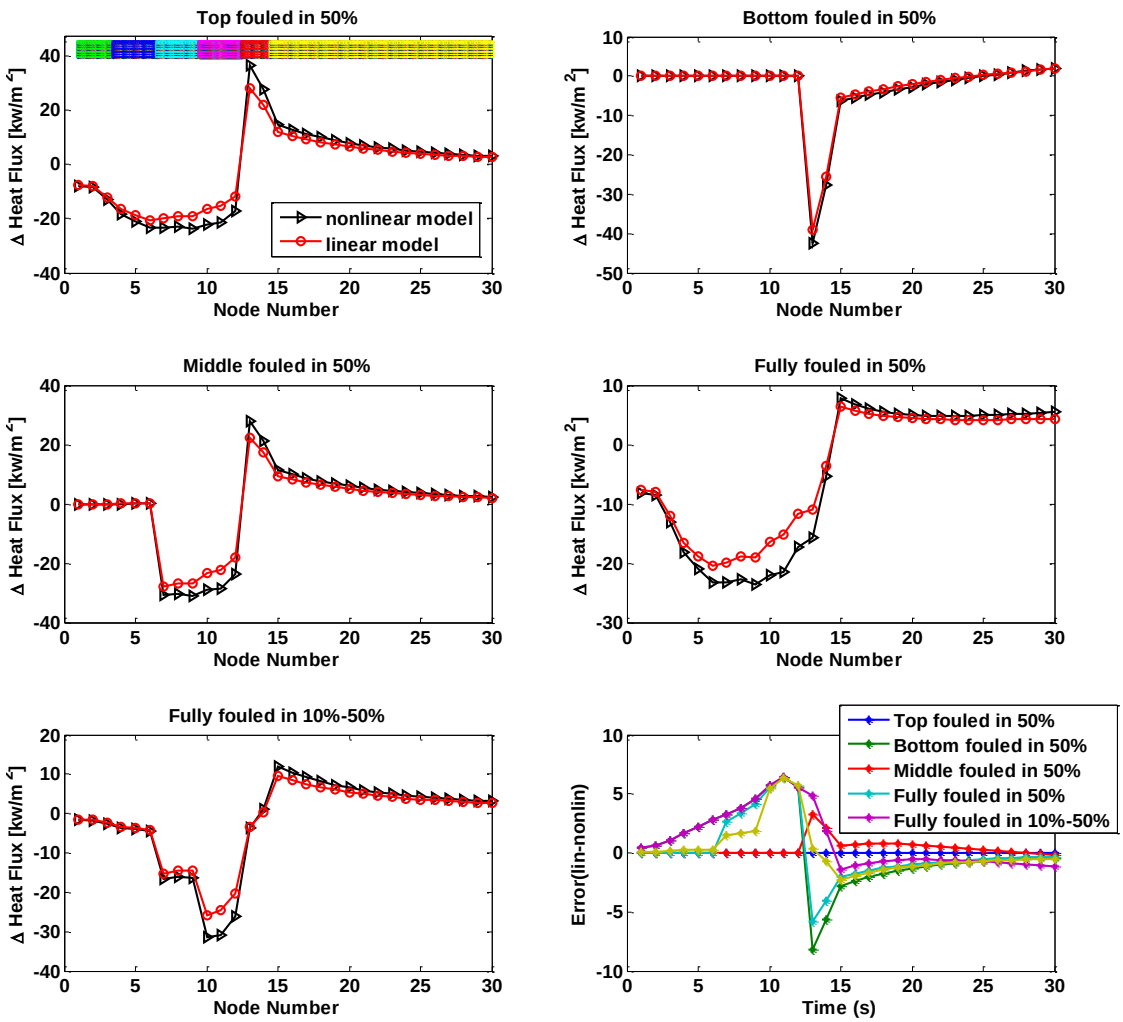


Figure 25: The comparison of the RSC model heat fluxes between linearized model and nonlinear model for non-uniform half fouled zones. The six zones, that have different heat transfer coefficient parameterizations, are highlighted in the top left plot. The bottom right plot shows the error in term of deviation for the linear model from the nonlinear model.

In this simulation, six non-uniform cases are presented as shown in the top left plot in Figure 23: 10% fouling for each zone cases, where the specified zone has fouling scale factor of 0.1 after 200 second, while the other zones are un-fouled. The values of heat flux are recorded at the end of simulation 400 second. It can be seen in Figure 24, the fouling profile is very “observable” from the heat flux, when a small fouling occurs at 10% in each zone. The linearized RSC model’s heat flux agrees very well with the nonlinear model with different heat transfer coefficient parameterizations as shown in the figure. The next six subplots with individual zone fouled 10% describe the heat fluxes deviation from the baseline no-fouling case. The right plot on the bottom shows the error in terms of the deviation of the linear model from the nonlinear model.

The heat flux profile comparisons between linearized and nonlinear RSC model for half-fouled RSC tubes in different zones are presented in Figure 25. In Figure 25, the three

cases presented at the top represent cases where the top section of RSC is fouled, bottom section of RSC is fouled and middle section of RSC is fouled, are all non-uniform cases, where the top four zone, bottom two zone and middle two zone have fouling scale factor of 0.5, respectively, while the fully fouled case is uniform with fouling scaled factor of 0.5 for all the zones. The left bottom one is the non-uniform case, where the fouling scaled factors are 0.1, 0.1, 0.3, 0.5, 0.3, and 0.1, for corresponding six zones. This is close to the real fouling situation happens in IGCC plant. The input fouling profiles are presented in Figure 23 as the last five plots. As expected, the percent error in Figure 25 is relatively larger than the ones in Figure 24. This is resulted from the fact that the linear RSC model is obtained at no-fouling condition, so that it is expected to have better matches when fouling is close to zero. It is also observed that the effect of fouling on the heat flux profiles can be a significant signature for soft sensing.

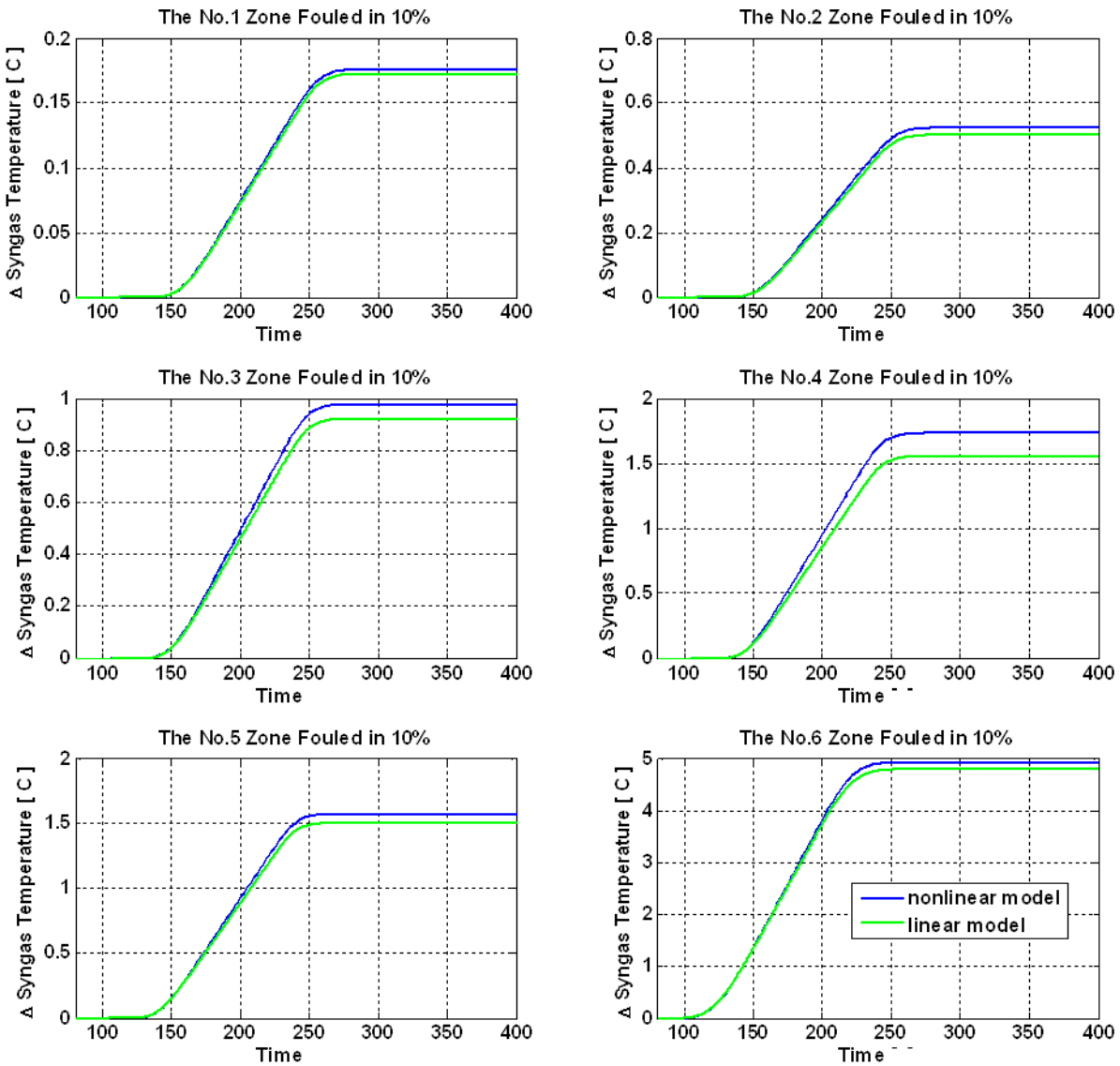


Figure 26: The comparison of the effect of fouling on the exit syngas temperature on the RSC bolt between linearized model and nonlinear model for small non-uniform fouling (10%) at different zones.

Since the fouling on the RSC tubes decreases the heat transferred from the hot syngas to the water inside RSC tubes, the signature of fouling is also present in the exit syngas temperature change. The effect of fouling on the exit syngas temperature of RSC is presented in Figure 26 (deviation from the un-fouled case), for the same fouling profile of Figure 24. Figure 26 also shows that the linear model response matches that of the nonlinear model response well during the transient condition, when the fouling ramps between 100-200 second in the simulation as shown in the first plots of Figure 23. As expected, the exit syngas temperature increases when fouling occurs. This increment is more significant when fouling cumulates to 50% as in Figure 27. It can be seen that the increment of the exit syngas temperature is very “observable” with the cumulated fouling in the transient and steady state conditions, but it is not “observable” for non-uniform fouling profile along the RSC tube length. In another word, it may not be possible to distinguish between a uniform fouled RSC tube and non-uniform fouled RSC tube by measuring the exit syngas temperature alone.

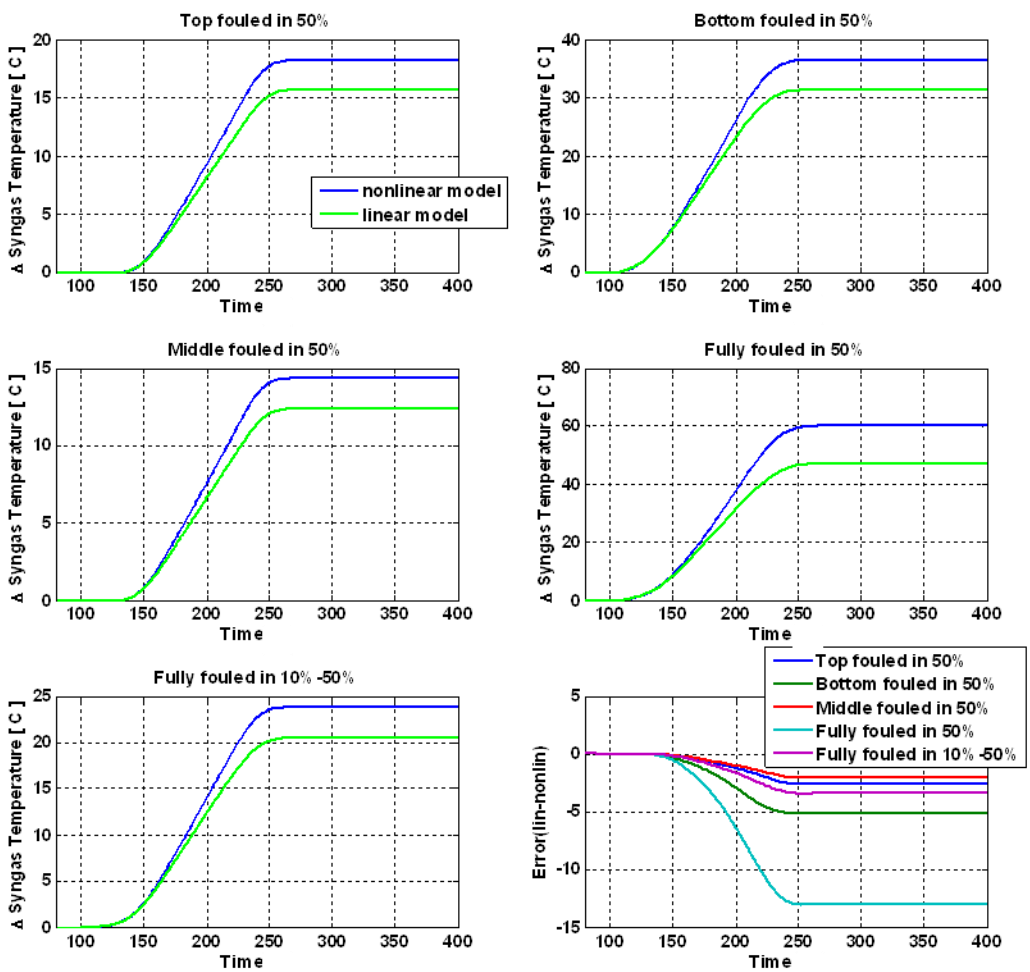


Figure 27: The comparison of the effect of fouling on the exit syngas temperature on the RSC bolt between linearized model and nonlinear model for non-uniform half fouled zones. The bottom right plot shows the error in term of deviation for the linear model from the nonlinear model.

Another signal that can be utilized for tracking the fouling of the RSC tubes is the strain experienced by the mechanical structure at the top that supports the weight of the RSC

tubes, where the extent of fouling along the entire tube length can be tracked by the increasing strain on the RSC bolts due to the additional weight brought on by the fouling. The strain experienced on this support structure, in terms of added strain from the baseline no-fouling case, is presented in Figure 28 for the fouling profiles given as the last five plots in Figure 23, where the fouling for each case is modeled as a 100 second ramps from zero fouling to full fouling. It can be seen that the added stain is very “observable” with the cumulated fouling in the transient and steady state conditions. However, similar to exit syngas temperature the strain alone cannot be used to ascertain fouling profile along the RSC length. The right bottom plot shows the error between the linear model response from the nonlinear ones which indicates that the linearized model agrees very well with the nonlinear model.

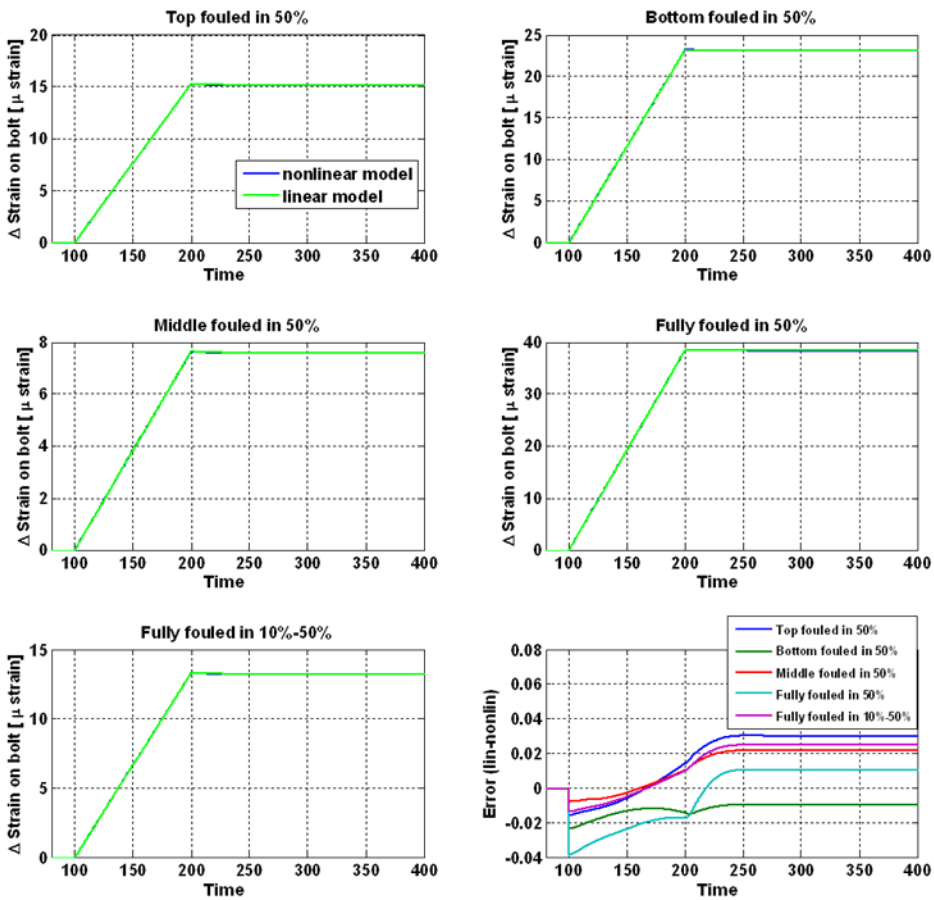


Figure 28: The comparison of the effect of fouling on the strain on the RSC bolt between linearized model and nonlinear model for non-uniform half fouled zones. The bottom right plot shows the error of strain in term of deviation for the linear model from the nonlinear model.

The heat flux, syngas temperature, and strain on the bolt are all measurements that might be utilized to track the fouling of the RSC tubes. These three classes of measurements are outputs of the linearized RSC model. Another measurement that can be utilized for this purpose is the amount of steam extracted from the RSC drum. When fouling occurs, the heat transfer from the hot syngas to the water decreases, and it is expected that the amount of steam extracted from the RSC drum will decrease.

The following simulation compares the steam flow from RSC steam drum between linear and nonlinear overall plant model. The steam flow is measured as the output of the drum. The same fouling profile, as shown in the top left plot in Figure 23, is given to the each zone as the input. Here the fouling is modeled as 1000 seconds ramp between no fouling and full fouling starting at 2000 seconds as more time is need for the overall plant to reach its steady state. The output of the steam flow in drum is presented in Figure 29. As expected, the steam flow matches well between linear and nonlinear cases, and both have negative correlation with the fouling.

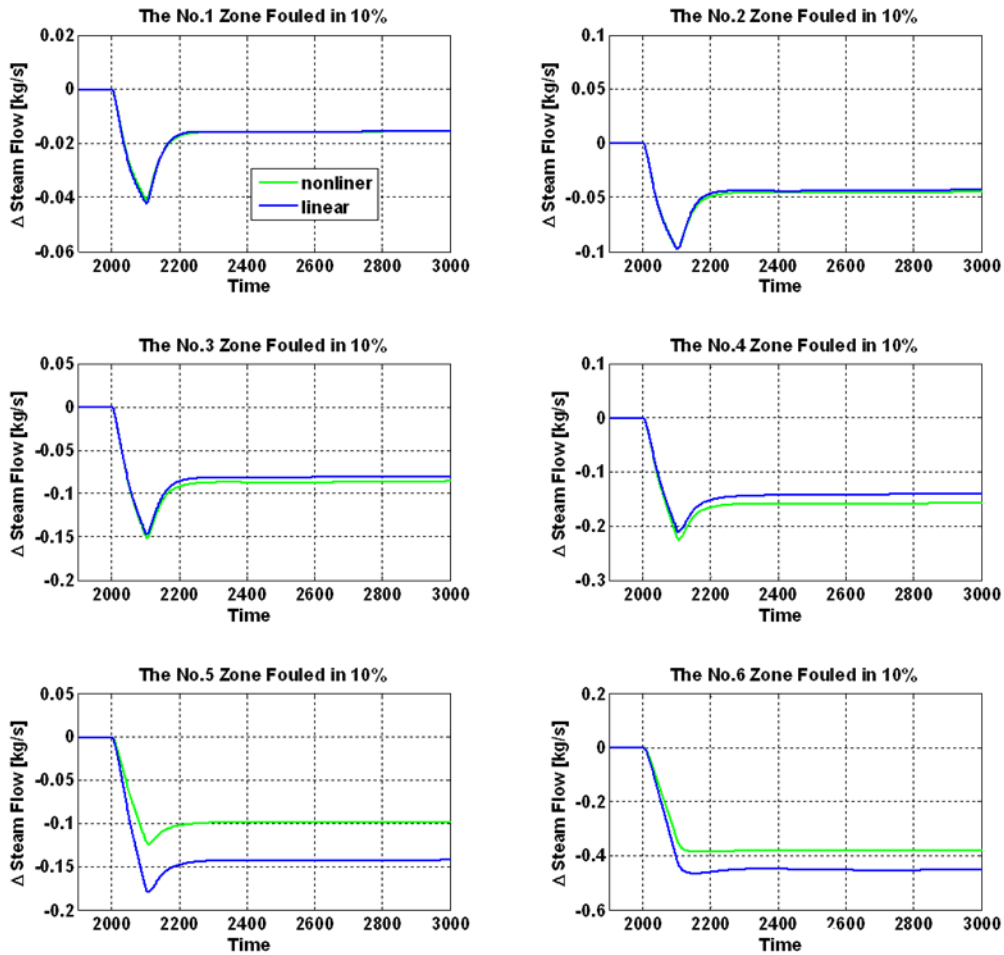


Figure 29: The comparison of the effect of fouling on the exit syngas temperature on the RSC bolt between linearized model and nonlinear model for small non-uniform fouling (10%) at different zones.

The study presented in this section shows that in order to monitor fouling profile along the RSC length the OSP algorithms will have to use combination of RSC syngas temperature at exit, drum steam flow, bolt strain at the mechanical support along with heat flux measurements at certain location along the RSC length.

5 Task 3: Development of OSP Algorithm

This task was focused on developing an OSP algorithm by integrating the estimation and the optimization algorithms discussed earlier. In order to develop a general algorithm for solving an OSP problem, first various optimization algorithms were evaluated.

Recall that the optimal sensor placement problem was formulated as a “cost” minimization problem subject to estimation precision and network reliability constraints in the presence of expected sensor failure rates (Eq. (11)). Mathematically the OSP problem is defined as follows:

Eq. (60)

$$\min_{q,n} c^T q$$
$$C_l^e P_\infty(q) C_l^{e^T} \leq \bar{s}_l,$$
$$r(q) \geq \bar{r}_l$$
$$q_i = \{0,1\}, \forall i \in \{1, N\}\ , \quad n_i = 1,2,3 \ldots., \forall i \in \{1. N\}$$

Where, $r(q)$ is the reliability constraint on the sensor network and defined as either hardware reliability (Eq. (8)), functional reliability (Eq. (9)) or estimation reliability (Eq. (10)) based on the problem at hand.

The OSP problem in Eq. (60) is an INLP problem. As mentioned earlier, there are two fundamentally different approaches for solving an INLP problem: (1) Branch and Bound and (2) Outer Approximation. In the current program both these two approaches were applied to solve the OSP problem. Since the BB method can be applicable to both MINLP as well as INLP problems, the approach was directly implemented for the OSP problem. However based on the analysis described in the literature survey section, the existing OA method could not be directly applied to the OSP problem and hence modifications to the existing method was required and developed as a part of the current program.

The OSP solution methodology was developed and first implemented on a simplified gasifier model for the refractory wear monitoring before applying it to find the solution to OSP problem for condition monitoring of full-fledged 3-D gasifier model refractory wear and RSC fouling profile. . In the rest of the section, the application of the BB method is presented followed by the OA algorithm developments.

Solving the OSP problem for a simplified 1-D gasifier model provided a good and quick mean to understand the applicability and limitations of the proposed methodology for OSP solution. Before going into OSP solution methodology developed, a brief description of the simplified gasifier refractory model is presented.

5.1 Simplified gasifier refractory model for OSP implementation

In the simplified gasifier refractory model, the gasifier refractory geometry and the thermal profiles are assumed to be uniform in the axial as well as azimuthal directions, i.e. the

wear on the hot inner surface is also uniform in these directions. This implies that there is only one unknown uniform wear to be estimated and the thermal profiles vary only in the radial direction from the hot inner surface to the external shell. This 1-D gasifier refractory model has a total of 15 dynamic temperature states arising from the spatially discretized nodes in each of the three refractory brick layers, the insulation and the metal shell. Excluding the inner half of the innermost refractory brick layer, which wears out over the life of the refractory lining between successive repairs, there are approximately 20 locations for temperature sensors corresponding to the respective nodes in the discretized model as well as the interfaces between the successive brick/insulation/shell layers. It should be mentioned that not all the sensor locations are practically viable, these 20 locations are used to specify a rich enough potential sensor locations to implement and test the OSP algorithms and gain practical insights. Further, there are 13 model parameters that are constant but unknown. To account for modeling uncertainties, these are modeled as unknown adder or multiplier, as required, to the nominal known parameters. These parameters include the gasifier core temperature, the heat flux on the hot inner surface, the heat flux from the outer shell to the ambient and thermal conductivities and heat capacities in each of the five material layers (three refractory bricks, insulation and metal shell). The nonlinear 1-D dynamic model is of the form in Eq. (5) with a first-order Markov model for the model parameters. This nonlinear model was linearized at two different steady-state conditions (i) zero wear, i.e. brand new refractory, and (ii) approximately 35% of the inner refractory layer worn out. These linear models were used for initial development, implementation and testing of the OSP algorithms. In particular, the linear model obtained at zero wear condition was used in the initial studies as the sensor placement with a zero wear model was more challenging. This is because as the brick wear progresses and the corresponding thermal profiles get hotter, the effect of the wear is relatively more pronounced on the temperature sensors, hence the zero wear model was the most difficult in terms of meeting desired wear estimation precision requirements.

5.1.1 Monitoring requirement for the simplified gasifier OSP problem

The solution to OSP problem seeks the best (lowest cost) sensor location $q \in \Re^{20}$, that can satisfy the precision and reliability. These sensors are assumed to have sensor cost $c \in \Re^{20}$ and corresponding individual sensor failure rates $r \in \Re^{20}$, as shown in Figure 4 and Figure 5, respectively. The sensors located in the inner most wall while being most expensive also are the most observable and more prone to failures due to the extreme temperature. For this reason, in all our studies with 1D gasifier model, the first 5 innermost possible locations are excluded from sensor placement. With this restriction, there are only 15 possible locations for sensor placement. It must be noted that the sensitivity to precision and survivability of the individual sensors are almost always in conflicting order in the sense that most survivable sensor may not provide best sensitivity to precision. If all the sensors are highly survivable, then a sensor placement that satisfies the precision constraints will also imply a higher reliability.

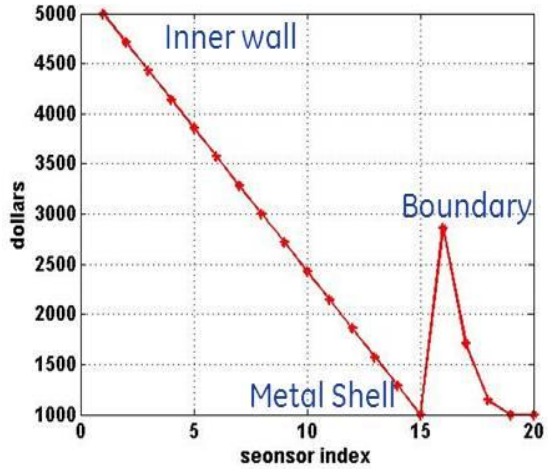


Figure 30: Sensor Cost

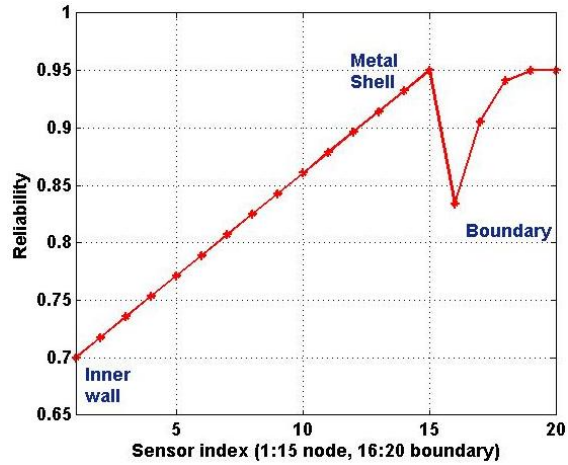


Figure 31: Sensor Failure Rate

The OSP problem for the simplified gasifier model has two versions each associated with functional or estimation reliability formulations. The two formulations are given as follows:

Version 1:

Eq. (61)

$$\min_{q} c^T q$$

$$C^e P_\infty(q) C^{e^T} \leq \left(\frac{2.78}{100}\ t_{Brick}\right)^2$$

$$\sum_{\omega_k \in \Omega_q} \Pr(\omega_k)\, I(\omega_k) \geq 0.9$$

$$q_i = \{0,1\}, \quad for\ \ i = 1, 2, \ldots, 15$$

Version 2:

Eq. (62)

$$\min_{q} c^T q$$

$$C^e P_\infty(q) C^{e^T} \leq \left(\frac{2.78}{100}\ t_{Brick}\right)^2$$

$$\sum_{\omega_k \in \Omega_q} \Pr(\omega_k)\, I(\omega_k) \geq 0.9$$

$$q_i = \{0,1\}, \quad for\ \ i = 1, 2, \ldots, 15$$

Eq. (63)

$$\min_{q,n} c^T n \times q$$

$$C^e P_\infty(q) C^{e^T} \leq \left(\frac{2.78}{100} \ t_{Brick}\right)^2$$

$$1 - \prod_{i=1}^{N} (1 - q_i p_i)^{n_i} \geq 0.9$$

$$q_i = \{0,1\}, \quad for \ i = 1, 2, \dots, 15$$

$$n_i = \{0, n^{Max}\}, \quad for \ i = 1, 2, \dots, 15$$

Where, i corresponds to all possible sensor locations. In Eq. (62) Ω_q is the set of possible sensor failure scenarios associated with a given sensor configuration q and is defined as the power set of the locations where the sensors are placed. For example, if the sensors are located at indices 6 and 7, i.e.,

$$q = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ],$$

Then $\Omega_q \coloneqq [\omega_1, \omega_2, \omega_3, \omega_4]^T$,

Where,

$\omega_1 = [0^{15}], (case\ when\ all\ sensors\ have\ failed)$
$\omega_2 = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ], (sensor\ 7\ has\ failed)$
$\omega_3 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ], (sensor\ 6\ has\ failed)$
$\omega_4 = q\ (no\ sensor\ has\ failed)$

In Eq. (62) $I(\omega_k)$ is the indicator function and $\Pr(\omega_k)$ is probability that a sensor failure can occur given the base sensor configuration. For example, for the case given above, the probability $\Pr(\omega_2)$ implies that term $P(q_6\ working\ \cap q_7 failing)$ and this is not the same as term $P(q_6\ working\ )$.

In Eq. (63), n_i denotes the number of sensors that are required to increase the survival rates (p_i) of the sensor at the given location. The functional reliability (Eq. (9)) and estimation reliability (Eq. (10)) constraint makes the solving the OSP problem very challenging due to following reasons:

1. Convexity, smoothness of objective and constraints

This is essential for both the BB and OA based approaches where the feasible space is approximated through relaxations, the true feasible space of the INLP problem should always remained contained in the relaxed feasible space. While the precision constraint is both smooth and convex for the particular definition of measurement noise covariance matrix R as in Eq. (12), the functional reliability is non-convex and the indicator function $I(\omega_k)$ used in the reliability definition makes the reliability constraint non-smooth. It must be noted that the convexity of the reliability constraint is still guaranteed as long as the set of points that satisfy the precision constraint is a convex set. This is so because, an Indicator functions over a convex set, namely the set of precision feasible points, is a

convex function. The non-smoothness can be alleviated by finding smooth approximations to the indicator function, such as sigmoid functions.

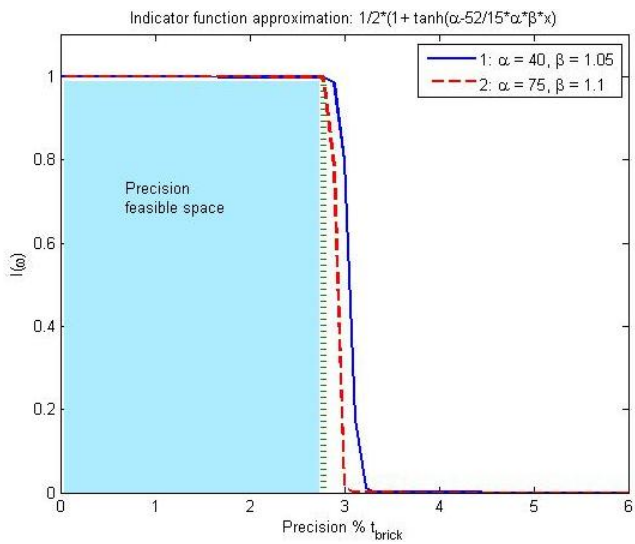


Figure 32: Approximation to indicator function.

Figure 32 shows two such approximations which can influence the solutions of the OSP problem. The looser the approximation (blue solid line) is, the more optimistic the OSP solutions will be, in the sense that actual reliability if computed using the indicator function may not necessarily satisfy the 90% requirement. However, the tighter approximations (red-dashed) will run into the numerical issues associated with derivative computation when using NLP solvers.

2. Computation of the Ω_q set of possible failure scenarios

Another key challenge in enforcing the estimation reliability constraint lies in computing the reliability metric itself, which is a summation over all combinations of the failure scenarios for any given sensor configuration, namely the set Ω_q. For the 1D OSP problems, the potential size of the set can be between $2\ to\ 2^{15}$ and computing this set can be quite expensive. It can be argued that it may not be necessary to evaluate all the sensor failures while solving the OSP problem. The BB method based on integer relaxation solves an NLP or LMI relaxation optimization problem that often returns real-valued decision variables. While the argument above can be made with pure integer solution, it is difficult to interpret real-values. For example, when $q = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ] \in Z^{15}$ the Ω_q has 2^2 elements, while for $q = [0\ 0\ 0\ 0\ 0.1\ 0.2\ 0.3\ 0.4\ 0.001\ 1\ 0\ 0\ 0\ 0\ 0\ ] \in \Re^{15}$, the Ω_q will have 2^6 elements. This means depending upon the relaxed solution, the Ω_q may still have large number of elements. This however is not applicable to OA based method as the OA based approach will always yield integer solutions at each iteration. One way to reduce the effort associated with the Ω set computation can be to consider only those sensors that affect the estimation precision the most, thereby reducing the number of potential failure scenarios. While this would facilitate solving the original OSP problem with a benefit of getting the lowest cost, the additional effort to compute reliability (though reduced) still remains an open question.

In order to solve the OSP problem, given these challenges, approximations were required that would enable getting OSP solutions in a timely manner. The main idea behind these approximations would be to solve a convex part of the OSP problem while ensuring constraint satisfaction by making relevant changes to the BB as well the modified OA framework. The exact approximations depends on the choice of functional or estimation reliability and are described in Section 5.6

As mentioned earlier, three approaches were identified to be the most relevant for solving an OSP program, namely: (1) LMI based formulation of OSP problem that uses branch and bound, (2) Branch and Bound approach to directly solve OSP problem and (2) Outer Approximation. All the approaches were explored to identify the methodology that was both computationally efficient and accurate with the added complexity.

The rest of the section is organized as follows: First, the challenges of the estimation reliability constraint discussed above with respect to smoothness and the Ω set complexity are highlighted using the BB method on the 1D model as a case study, Then the development of the OA-INLP algorithm is presented. The application of the LMI, BB the LMI and the results of the three algorithms are presented and finally the proposed modifications for the reliability constraints are developed and validated on the 1D model .

5.2 Implementation challenges for Estimation Reliability using Branch and Bound based approach

Two approaches have been considered for solving the NLP relaxed problem at every node in the BB framework. These approaches are as follows,

1) Include both precision and reliability constraint in the NLP problem, where the NLPR (Eq. (16)) is given by,

Eq. (64) NLPRF

$$\min_{q} c^T q$$
$$C^e P_\infty(q) C^{e^T} \leq \left(\frac{2.78}{100}\ t_{Brick}\right)^2$$
$$\sum_{\omega_k \in \Omega_q} \Pr(\omega_k)\, I(\omega_k) \geq 0.9$$
$$q_i = [0,1], \quad for\ \ i = 1, 2, \dots, 15$$

2) Include only precision constraint at every node, where the NLPR is given by,

Eq. (65) NLPRP

$$\min_{q} c^T q$$
$$C^e P_\infty(q) C^{e^T} \leq \left(\frac{2.78}{100}\ t_{Brick}\right)^2$$
$$q_l = [0,1], \quad for\ \ l = 1, 2, \ldots, 15$$

In both the approaches, the BB framework remains the same, wherein upper and lower bounds are iteratively generated at every node and convergence is achieved when the bounds are within a specified tolerance. The upper bounds are generated anytime a feasible solution (both integer and nonlinear feasible) is found and the lower bounds are provided by the optimal solutions to the NLPR problem. It can be seen immediately that the approach 2 yields less tight lower bounds than approach 1, which may indicate slower convergence. While the process of enumeration, branching and bounding is the same, the conditions for branching and enumeration differs in the two methods.

The two approaches are pursed in order to understand the tradeoff between the computation effort and accuracy associated with detailed OSP problem with reliability constraint versus finding reasonable approximation using the precision constraint alone.

5.2.1 Approach 1 using NLPRF

The NLPRF is solved at every node and the enumeration is based on the most fractional variable. This means that if the ith index of the solution to the NLPRF, namely, q_i is very close to either 0 or 1, the ith index would be added as a branch in the subsequent search. Even though there is not a systematic procedure for determining the incumbents quickly, depth first branching strategy is adopted for this analysis. The calculation of the reliability metric for real-valued decision values, involved computing Ω_q for all possible sensor failure scenarios for sensor indexes whose value is greater than 0, i.e, $q_i > 0, i = 1,2,..15$. In order to mitigate the associated effort, only the set of sensor locations that are both failure prone and affect the estimation precision are considered. This leads to including only sensor locations at 6, 7, 8, 16 and 17 for evaluating the failure scenarios. It must be noted, that for this particular OSP application there is a clear separation in the precision sensitivity making this trick suitable. However this need not be true for other OSP applications and such a trick may yield overly optimistic solutions.

To smoothen the reliability constraint, the indicator function was approximated using sigmoid function given by $I = \ 0.5(1 + \ tanh(\ \alpha - 3.47\alpha\beta Precision))$, where α, β are parameters used to control the tightness of the indicator function approximation. Two sets of these parameters are used :

1. $\alpha = 40, \beta = 1.05$ is a smoother approximation, but less tight
2. $\alpha = 75, \beta = 1.1$ is a tighter approximation

Table 1: Optimal Sensor Placement

Case	Approximation	Reliability (actual)	Optimal location	Total cost
1	Smooth Approx.	0.79	$q_{6,7} = 1$	6857
2	Tight Approx.	0.9	$q_{6,7,16} = 1$	7857

The optimal sensor locations for cases 1 and 2 are shown in Table 1. The branch and bound (BB) iterates of the upper and lower bounds are shown in Figure 33 and Figure 34 respectively. The following observations can be made:

1. The optimal sensor placement for the 1D problem is to place the sensors at locations 6, 7 and 15 which has a network reliability of 90% and an associated cost with a hypothetical value of 7857.

2. From the table, it can be seen that a smoother approximation yields an overly optimistic solution, where the cost is lower than case 2. While the reliability using the approximation satisfies the 90% requirement, in reality this sensor network is only 79% reliable.

3. From the figures, another important observation is the non-monotonicity of the lower bounds in both cases. Even though this would not affect convergence to the true optimal solution, this indicates potential non-convexity of the reliability constraint with the approximation. With the tighter approximation (Figure 34), the non-monotonic behavior is reduced.

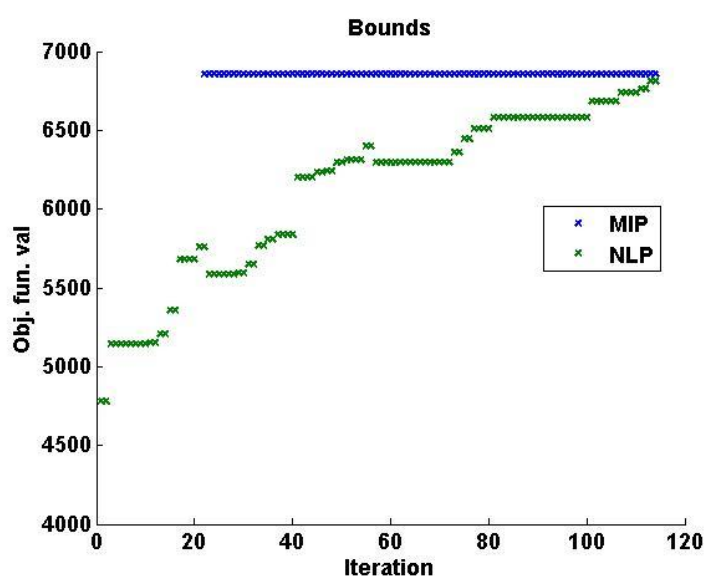


Figure 33: Smoother approximation of the Indicator function

4.

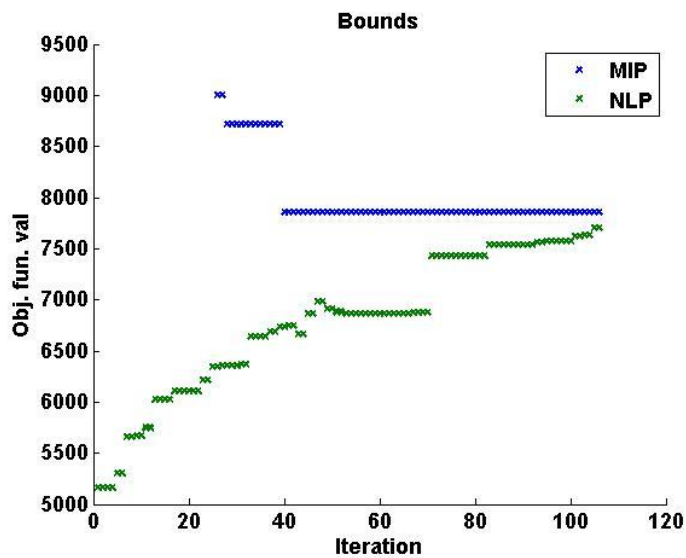


Figure 34: Tight approximation of the Indicator function

5.2.2 Approach 2 using NLPRP

The main idea here is to leverage the optimal sensor solutions obtained by only considering the precision constraint denoted by $q^*_{Precision}$and add redundancies to obtain the optimal solution that would also satisfy the reliability constraint. This rationale holds true because the feasible space of the solutions that satisfy the reliability constraint (F) is a subset of those set of points that satisfy precision constraint (F_P), i.e, $F \subset F_P$. One way to achieve this would be to add multiple sensors which would be a suboptimal option but computationally the least expensive.

The algorithm involves the following steps

1. Using the BB with NLPRP obtain the precision optimal solution. Here there are two incumbents, one for the precision (Z_P^U)and another for the reliability (Z_R^U).

2. If the precision optimal solution is also reliable then convergence is achieved, else

    a. Branch with respect to the highest sensitivity sensor index
    b. Solve the NLPRP whose solution is the lower bound (Z_P^L)
    c. Repeat a, b to convergence defined as $|Z_P^U - Z_P^L| < \epsilon$

The main issue with this approach is that while there is a mechanism of getting the incumbents quickly from the precision optimal solutions, the number of iterations that would be required to converge will be larger if the sensor failure rates are higher. This is so because the lower bounds are the solutions of NLPRP where $c^T q^*_{Precision} < c^T q^*$.

The optimal sensor locations using this approach are 6, 7 and 15. This is the true optimal solution with a cost with a hypothetical value of 7857.00 as shown in the Table 2.

Table 2: Optimal sensor placement by solving NLPRP problem at every node

Rel actual %	Cost	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
90	**7857**															

The BB iterates for this approach are shown in , where it can be seen that it takes about 80 iterations to reach the precision optimal solution, while about 1200 iterations to converge to the final optimal solution. In addition, the incumbent that satisfies the reliability constraint(Z_R^U), is found immediately from the precision optimal solution by the above branching strategy. The number of iterations that will be required will depend on the gap between the OSP solution for the precision only and the reliability problem, i.e., $c^T q^* - c^T q^*_{Precision}$.

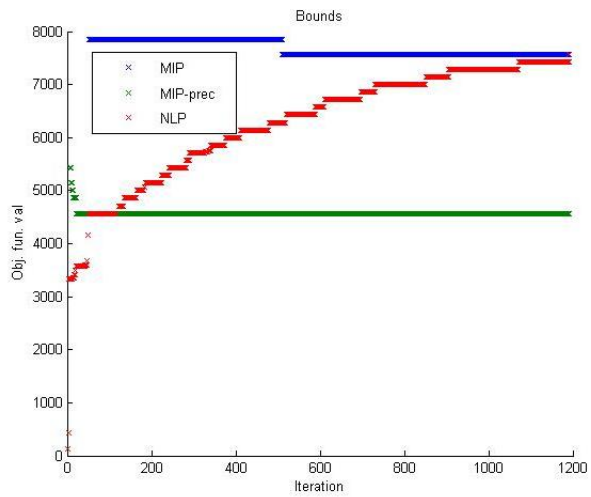


Figure 35: BB iterates for approach 2, MIP: (Z_R^U) Upper bound when reliability and precision are satisfied, MIP_prec: ((Z_P^U) Upper bound when ONLY precision is satisfied, NLP: Solution of NLPRP problem

Table 3: Comparison of optimization times with approach 1 and 2

Method	No. Iterations	Time/Iteration
NLPRR (App. 1)	~110	~165s
NLPRP (App. 1)	~1200	~1s

A comparison () of the OSP solutions along with the optimization time per iteration for both the approaches shows the computational benefits of using approach 2 while being able to achieve the optimal solution. Methods to intelligently restrict the feasible space of NLPRP problem will further bring down this effort by tightening the lower bounds thereby reducing

the total number of iterations required for convergence. Identify metrics that would allow for computing reliability using less number of failure scenarios without loss of accuracy in the reliability computation, one such metric explored in this report is using only the sensors with highest sensitivities.

5.3 Proposed Outer Approximation Based Approach for INLP

Another approach to solving an INLP problem arising in OSP formulation is Outer approximation. Since the OSP problem has a pure-integer decision space, conventional Outer approximation algorithm was needed to be modified. The algorithm development is discussed in this section. In the modified OA), the nonlinear constraints are relaxed to get the lower bounds and the following MOA problem (Integer linear Program) is solved.

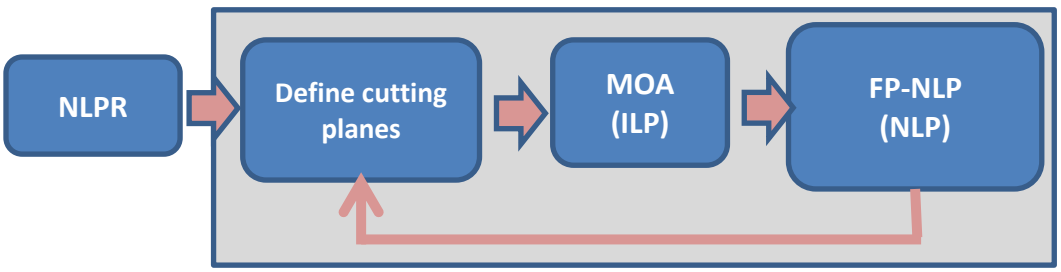


Figure 36: Schematic of OA-INLP method

The feasible space is contracted successively by defining cutting planes at the projections of MOA solutions on the feasible space which are computed as the solution of the feasibility problem (FP-NLP). The MOA and the FP-NLP are defined as follows:

MOA

Eq. (66)

$$
\begin{aligned}
&Z^{LB} = \min_{\alpha,q,y} \ \alpha \\
&c(q^k) + \nabla c^T(q^k)(q - q^k) \leq \alpha \\
&g(q^k) + \nabla g(q^k)(q - q^k) \leq 0 \\
&\sum_{i \in B^k} q_i^k - \sum_{i \in NB^k} q_i^k \leq \left|B^k\right| - 1 \\
&q \in \{0,1\}^n, \ , \alpha \in \Re, k = 1, \dots, K
\end{aligned}
$$

Where, $q^k \in CP$ is the cutting plane which is the solution of the NLPR problem (Eq. (16)) if it is nonlinear feasible or the FP-NLP problem (Eq. (67)) if it is infeasible.

FP-NLP

Eq. (67)

$$CP = \underset{q}{\operatorname{argmin}} \| q - q^{OA} \|$$
$$g(q) \leq 0$$
$$\mathrm{q} \in [0,1]^N$$

Where, q^{OA} is the solution of the MOA problem. It should be noted that the MOA problem is an Integer linear programming problem, while the FP-NLP is a nonlinear programming problem such as NLPR.

In the rest of the section the three methods, namely LMI, BB and OA-INLP are applied to the OSP problem using the 1D model and the results are presented.

5.4 LMI Based Approach

The solution of OSP problem as defined in Eq. (55) is given as:

Eq. (68)

$$\min_{\substack{c_i=1000 \\ q_i \in \{0,1\}, i=6,7,\ldots,20 \\ n_i=1,2,3}} \sum c_i q_i n_i = 12857.14$$
$$\{q_6 = 1, n_6 = 2;\ q_{16} = 1, n_{16} = 2\}$$

The branch and bound iterates for this case, are shown in Figure 37.

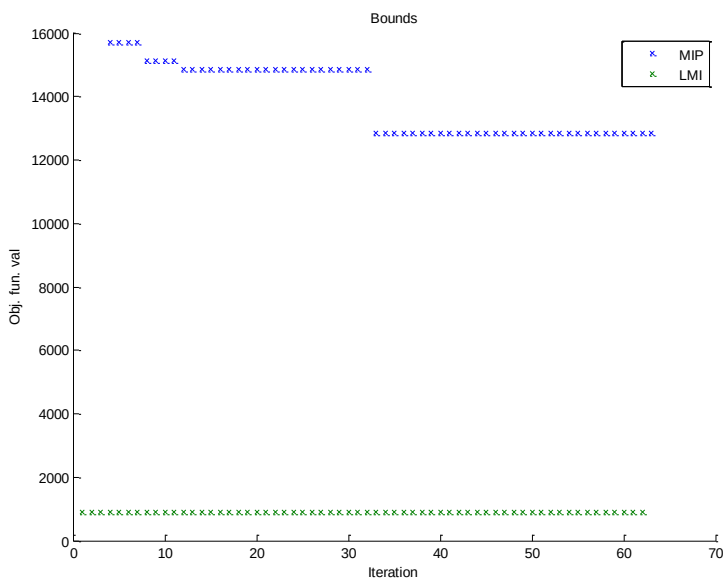


Figure 37: Branch-and-Bound Iteration result with LMI relaxation

As mentioned earlier, the total time required to compute the OSP solution using LMI formulation depends on the algorithm used. Next, an evaluation of the LMI solvers is provided

5.4.1 LMI solver evaluation

The previous section shows the optimal solution for the 1-D gasifier refractory problem by using branch-and-bound with LMI relaxation. The iteration results are generated by using LMI solver *mincx* of LMI Lab. This is a commercial solver developed by Matlab©. This solver is quite efficient for a small/medium size optimization problem. For the medium to large size optimization problem as in OSP for 3-D gasifier refractory model and RSC model, alterative solvers are also evaluated. One of the widely used non-commercial solvers, SDPT3 was investigated for solving medium-large size LMI problems and compared with Matlab© LMI solver.

All those solvers use the interior-point method, but with possible difference in barrier functions, direction determination mechanism, and options setting. LMI lab and SDPT3 solvers yield the same overall result for the sensor placement problem. However, the numbers of the iteration that branch-and-bound takes are different. Table 4 shows the comparison for these solvers.

Table 4: LMI solver evaluation with reliability constraints

	No Para Uncertainty		Para Uncertainty	
size	n = 153, m = 34		n = 465, m = 60	
Solver	Iteration	Time	Iteration	Time
LMI Lab	47	181	63	2200
SDPT3	111	1434	53	2324

The above table shows the CPU time, or the total time, to solve the MIP problem by using branch-and-bound algorithm with LMI relaxation. The LMI lab out performs SDPT3 in terms of time when the size of the problem is small. But, its performance deteriorates significantly as the number of states increases in the presence of parametric uncertainties. However, the SDPT3 solver is relatively slow for solving a small size problem, but the time does not grow significantly compare to that for LMI Lab.

5.4.2 Discussion of LMI implementation

As shown in this section, the standard branch-and-bound algorithm and LMI solvers are implemented to solve the optimal sensor placement problem in 1-D gasifier refractory model. The technique is easy to apply, and efficient for small to medium size problem.

While LMI and branch-and-bound technique is chosen for many advantages, the LMI technique has some practical limitations. One limitation is the size of the optimization problem. Even with SDPT3 the OSP problem using the full 3D gasifier model cannot be solved. However, for certain other applications having small to medium size OSP problem, LMI does provides competitive solutions. Further, the reliability constraint is not easy to formulate as an LMI. However, with the approximate formulation used here, a practical solution to the OSP problem can be obtained.

5.5 Results of OA-INLP on 1D Gasifier Model

The proposed OA for INLP was applied to solve the OSP problem using the 1D gasifier model in order to understand the algorithm applicability. To do so,
Matlab© NLP solver *fmincon* was used for solving NLPR and FP-NLP sub-problems, while the GNU ILP solver *GLPK* was utilized to solve the MOA sub-problem. The optimal solutions and the computation times are shown in Table 5.

Table 5: Optimization Time using *fmincon* for the 1-D Gasifier Model

Method	Constraints	**Optimal location**	**Total Optimization time (s)**	**No. Iterations**
BB-NLP	Precision	$q_{6,16} = 1$	440	40
OA-INLP	Precision	$q_{6,16} = 1$	215	16
BB-NLP	Precision and Estimation Reliability	$q_{6,7,16} = 1$	O(1000)s	110

From the table, it is obvious that the computational effort is of the order of few hundred seconds and increases drastically with the inclusion of the reliability constraint even for a small size model. Even though the OSP for condition monitoring is an offline problem there is still a need for obtaining optimal solutions in a timely manner.

Based on the above analysis, it is clear that to compute OSP solutions in a timely manner modifications to the OA and BB framework are essential and are presented in the sequel.

5.6 Modifications to BB and OA framework for Reliability constraint

The main idea still is the iterative generation of lower and upper bounds for the functional reliability problems defined in (Eq. (8)) and (Eq. (9)) respectively. However instead of solving the associated relaxations, the following properties are exploited to generate the

lower bounds by solving a simpler convex INLP problem, namely the OSP problem with precision as the only constraint.

1. Let Z_L, Z_U, F be the lower and upper bounds and the feasible space of the OSP problem with precision as the only constraint and Z_L^R, Z_U^R, F^R be the lower and upper bounds and the associated feasible space of the OSP problem with precision as well reliability (Functional or estimation). Then, the ordering $Z_L \leq Z_L^R \; and \; Z_U \leq Z_U^R$ holds true since $F \subset F^R$.

Through this ordering the following modifications to the BB and the OA method is proposed. The lower bound computation is exactly the same which the solving a NLPR (BB) or ILP (M-OA) problem. The key modification is in the computation of an incumbent. Three possible cases can occur with the solution of the relaxed problem (lower bound calculation) at each iteration in the BB:

- Case 1: The solution of the NLPR problem satisfies precision constraints alone. In this case the current node is added as a potential node to be explored for further search.
- Case 2: The solution of the NLPR satisfies integer and precision constraints. This case is the one which requires modifications (described in the sequel). In the traditional method, an incumbent would be declared. However since the associated objective function, namely Z_U, lower bounds the true incumbent Z_U^R, the following modifications are made to the existing framework depending on the type of reliability considered.
- Case 3: The solution of the NLPR satisfies integer, precision and reliability constraints. Update the incumbent Z_U^R

Three similar cases occur in the OA methodology after solving the relaxed M-OA problem, which are:

- Case 1: The solution of the M-OA problem satisfies INTEGER constraints alone. In this case, the feasibility restoration problem (FPNLP) problem described in Eq. (67) is solved and a supporting hyper-plane is added to contract the feasible space. This step is exactly the same as in the previously proposed method

- Case 2: The solution of the M-OA satisfies integer and precision constraints. This case is the one which requires modifications (described in the sequel).

- Case 3: The solution of the M-OA satisfies integer, precision and reliability constraints. Update the incumbent Z_U^R

Before presenting the modifications to the proposed algorithm associated with Case 2, a discussion of the fundamental differences between functional and estimation reliability is required. For any given sensor configuration, the functional reliability can be made to satisfy by increasing the number of sensors. Hence any solution of the relaxed problem that solves both precision as well integer constraints can satisfy the functional reliability by computing the optimal (cost-effective) number of sensors required to achieve constraint

satisfaction. This would require solving another optimization problem to compute the number of sensors since the decision space of the OSP problem with precision constraint is a 0/1, but that of the functional reliability also includes the number of sensors.

However the same does not hold true with the estimation reliability since both redundancy as well as observability of the monitored variable need to be improved with respect to a sensor configuration that satisfies precision constraint. This means that while adding sensor at location "l" may increase the sensor redundancy, it need not improve estimation reliability if monitored variable is poorly observable from that specific particular location.

5.6.1 Modification to OSP Formulation for Handling Functional Reliability Constraints

Given the NLPR ${q_k}^{NLPR}$or the MOA ${q_k}^{MOA}$ solution of the relaxed problem at k[th] iteration,

1. Compute the optimal number of sensor n_k^* as the solution to the following optimization problem

Eq. (69)

$$\min_{n}(n \times c)^T q_k^*$$
$$1 - \prod_{i=1}^{N}\left(1 - q_{\{k,i\}}^* p_i\right)^{n_i} \geq \bar{r}_l$$
$$n_i = 1,2,3 \ldots., \forall i \in \{1.N\}$$

   Where, $q_k^* = {q_k}^{NLPR}$ for BB and $q_k^* = {q_k}^{MOA}$ for OA.

2. Update the incumbent as $Z_U^R = (n_k^* \times c)^T q_k^*$

Note that:

- The reliability constraint in the above problem is convex with respect to the number of sensors (n) as opposed to *q* and *n* and can be solved using the proposed INLP solver.
- It must be noted that the reliability constraint is a product of the reliability of the sensor(s) at individual locations. Hence in order to satisfy the functional reliability constraint for the network, the individual probability must satisfy the threshold, i.e, $\left(1 - q_{\{k,i\}}^* p_i\right)^{n_i} \geq \bar{r}$. This implies the minimum number of sensors required at each location can be computed a priori and is computed as

Eq. (70)

$$n^{apriori} = \{n_i | i = (1 - p_i)^{n_i} \geq \bar{r}, i = 1..N\}$$

The modifications for Case 2 with respect to estimation reliability are described next.

5.6.2 Modification to OSP Formulation for Handling Estimation Reliability Constraints

In case of the branch and bound algorithm, given the NLPR solution ${q_k}^{NLPR}$,

1. The current node is added to the list of bud nodes to the explored
2. Branch with respect to the highest sensitivity sensor index at a current sensor configuration or index of the location with highest observability.
3. Enumerate and repeat the steps in BB-schematic

In case of the OA algorithm, at any given iteration *k*, if the MOA solution ${q_k}^{MOA}$ satisfies integer and precision constraint but not the estimation reliability define an integer cut at ${q_k}^{MOA}$ which is defined by the addition of the following constraint

Eq. (71)

$$\sum_{i \in B^k} q_i^k - \sum_{i \in NB^k} q_i^k \leq |B^k| - 1$$

$$Where\, B^k = \{i | q_k^{MOA} = 1\},\, and\, NB^k = \{i | q_k^{MOA} = 0\}.$$

A very important observation is that it is possible to define a supporting hyper plane when a solution does not satisfy the precision constraint, while only an integer cut can be achieved when the estimation reliability is not met. This is so because for the former, the FPNLP which considers the precision constraint can be solved at each iteration to define the support, while solving a restoration for reliability implies solving the optimization problem with the reliability constraint and the associated issues described above. An integer cut being a point wise elimination is not effective as a cutting plane which eliminates a larger area of the infeasible space.

5.6.2.1 Implementation Considerations

While the proposed modifications to the BB and OA methodologies, considers simpler and convex relaxed problems it can be seen immediately that this approach yields less tight lower bounds, which may indicates slower convergence. A tighter lower bound for the functional reliability can be computed by solving the relaxed problem (NLPR or ILP) associated with the optimization problem defined below:

Eq. (72)

$$\min_q (n^{apriori} \times c)^T q$$

$$C_l^e P_\infty(q) C_l^{e^T} \leq \bar{s}_l,$$

$$q_i = \{0,1\}, \quad \forall i \in \{1.N\}$$

Where $n^{apriori}$ is defined in Eq. (70).

5.6.3 Results

An offline optimization was performed to determine the optimal sensor location ($q_i \in \{0,1\}, i = 1..15$) using BB and OA-INLP methods. The comparison of the optimal sensor location, cost and computational times for the functional reliability is shown in Table 6.

Table 6: Functional reliability optimization results

Reliability	Sensor location	Number of Sensor	Opt Cost	No. Iterations
Functional-BB	6,16	2,2	12571.43	49 (58s)
Functional-BB-tighter lower bound (Eq 17)	6,16	2,2	12571.43	35 (42s)
Functional-OA	6,16	2,2	12571.43	110 (145s)
Functional-OA-tighter lower bound (Eq 17)	6,16	2,2	12571.43	34 (52s)

It can be seen that by tightening the lower bound as proposed in Eq.17, the number of iterations along with the computational time is significantly reduced for both the BB and the OA based methodologies. The progress of the iterates, namely the bound generation, using the BB and the OA methods for functional reliability is shown in Figure 38 and Figure 39, respectively where the black x's are the lower bounds of the reduced feasible space while red x's are those without using the proposed modification.

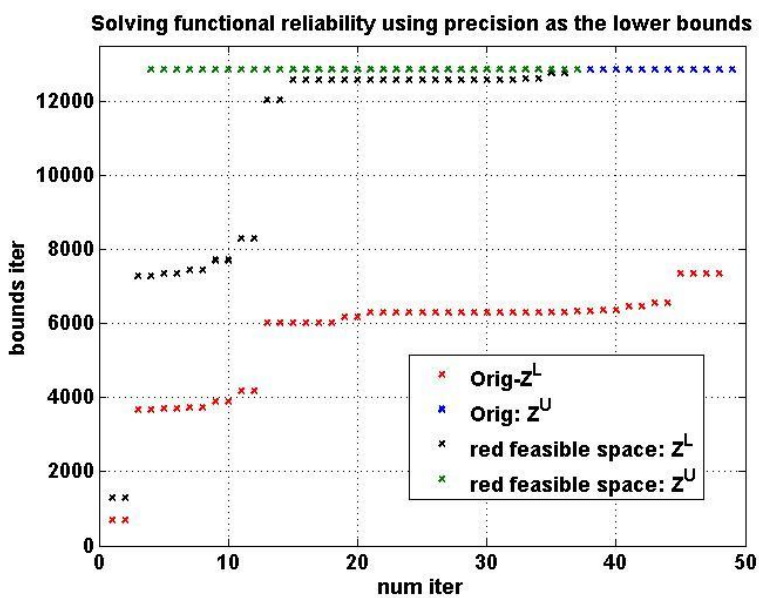


Figure 38: BB method functional reliability

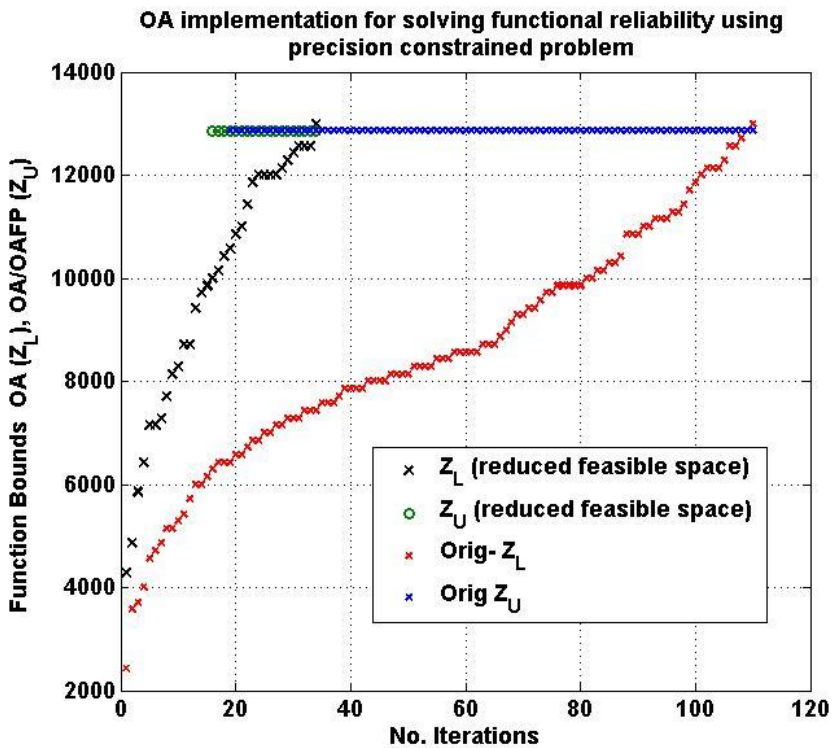


Figure 39: OA method functional reliability

The optimization results for estimation reliability are described in the rest of the section. Two different branching were considered in the BB implementation,

- Branching with sensor index with the following ordering on observability `[6 7 8 16 9 10 11 12 17 13 14 18 15 19 20]`
- Branching with respect to the element which has maximum sensitive to the precision constraint at a given sensor configuration.

The OA method implements the integer cut and no strategy was implemented for obtaining a tighter lower bound for both the OA and the BB algorithms. The comparison of the optimal sensor location, cost and computational times for the functional reliability is shown in Table 7.

Table 7: Estimation reliability optimization results

Reliability	Sensor location	Number of Sensor	Opt Cost	No. Iterations
Estimation-BB (obsv)	6,7,16	1,1	~9714	750 (~420s)
Estimation-BB (sens)	6,7,16	1,1	~9714	1350 (~800s)
Estimation-OA	6,7, 6	1,1	~9714	250 (250s)

It can be seen that the OA method far outperforms the BB algorithm in terms of the computational efficiency while both of them have the same solution accuracy. This is because in the BB the enumeration is used for case 2, while in OA a point wise elimination

is done. The progress of the bounds for the BB is shown in for the two branching strategies and that of OA is shown in , respectively. It can be seen that for OA-INLP, the convergence is achieved once an incumbent is obtained given that there is no associated primal problem due to the pure integer space.

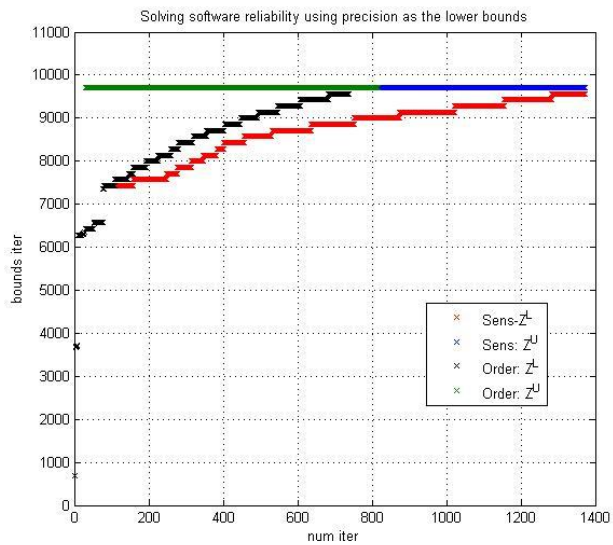


Figure 40: BB implementation of estimation reliability

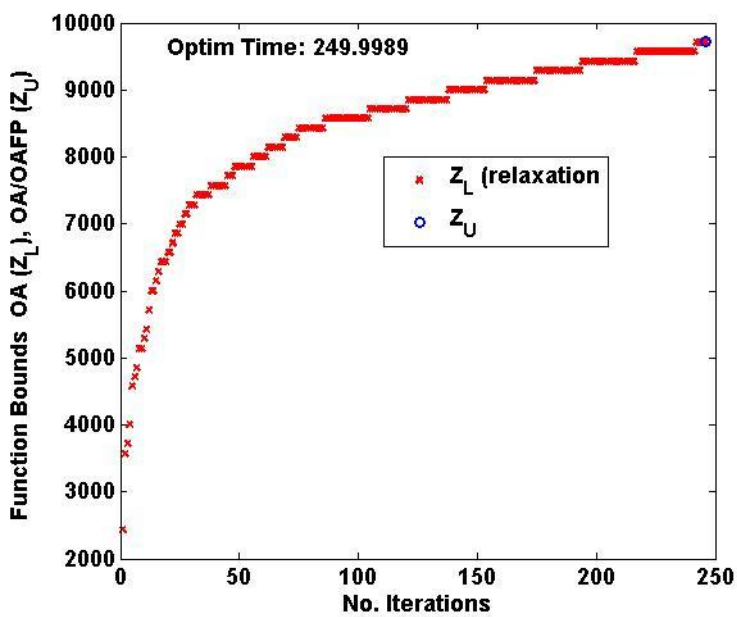


Figure 41: OA implementation for estimation reliability

Given that the underlying nature of the problem is combinatorial as well as nonlinear, a thorough analysis was done in order to identify the set of tools that would achieve solution accuracy in a computationally efficient manner. For any successful implementation of the OSP there are few critical elements:

- **State-of-art NLP solvers:** The solvers for both NLP and ILP problems need to be efficient as well as robust. The state of the art solver for the efficient NLP implementation is the interior-point based *Ipopt* (www.coin-or.org/Ipopt), while the GLPK (http://www.gnu.org/software/glpk/) is considered as the ILP solver

- **Analytical gradients:** While using state-of-art solvers is an important aspect, the solver efficiency increases when as much information can be provided about the underlying NLP problem. One of the most important information is the gradient as well as the Hessian computation. Even though they can be computed using finite differencing, analytical gradient computation greatly enhances the solver efficiency.

5.6.4 Analytical Gradient Computation

The optimization algorithms requires gradient of the cost function as well the constraints. While it is straight-forward to compute the cost gradient (linear in this case), the gradient of the precision and reliability involves computing the sensitivities of the state error covariance matrix $\left(\frac{dP_\infty}{dq}\right)$. For instance the gradient of the precision constraint is given as,

Eq. (73)

$$\frac{ds}{dq} = C^{e} \left.\frac{dP_\infty}{dq}\right|_{q^k} C^{e^T}$$

Where P_∞ is the solution of the ARE given in Eq. (7) It can be seen if the gradient computation is done through finite differencing, the ARE equation has to be solved N+1 times which increases as the number of the sensor (N) as well as the underlying system dynamics (n) increased. In a previous work, an approach for computing the sensitivity equations for LQR design of control systems using continuous time ARE was presented (Borggaard 2004). This approach was extended to compute the covariance sensitivity. While the described approach deals with continuous time ARE, the noise characteristics are easier to describe in terms of covariance matrices. Given that the system under consideration is discrete time, the discrete-time Lyapunov approach was developed. The following proposition computes the covariance sensitivities in an efficient manner thereby providing the analytical gradients.

Proposition

Let (A,C) denote the discrete time state space and (Q,R> 0) denote the process and measurement noise covariance matrices. If $P_\infty \geq 0$ is the solution of the discrete ARE

solution, then the sensitivity $P_{q_i} := \frac{dP_\infty}{dq_i}$ can be computed as the solution of the following discrete-time Lyapunov equation

Eq. (74) $$\hat{A}P_{q_i}\hat{A}^T - P_{q_i} + \hat{Q} = 0$$

Where, $\hat{A}$ and $\hat{Q}$ can take any of the possible two values

a. $\hat{A} = Z(AP_\infty A' + Q)^{-1}A, \hat{Q} = -ZC'R_{q_i}^{-1}CZ,$
b. $\hat{A} = A - AP_\infty \gamma C, \hat{Q} = -AP_\infty C'\gamma R_{q_i} CP_\infty A',$

Where, $Z = [(AP_\infty(q)A^T + Q)^{-1} + C^T R^{-1}(q)C]^{-1}$, $\gamma = (CP_\infty C' + R)^{-1}$, $R_{q_i}^{-1} = \frac{dR^{-1}}{dq_i}$ and $R_{q_i} = \frac{dR}{dq_i}$.

This can be proved in two steps

- If Q,R>0, then we have$(AP_\infty(q)A^T + Q) > 0$ and hence the inverse exists which implies Z>0.

- Using the matrix identity which states that $\forall\Gamma > 0, \frac{d\Gamma^{-1}}{dq_i} = -\Gamma^{-1}d\Gamma/dq_i\Gamma^{-1}$, a and b above, can be obtained by differentiating the following equations respectively and rearranging terms.
  - $P_\infty(q) = [(AP_\infty(q)A^T + Q)^{-1} + C^T R^{-1}(q)C]^{-1}$
  - $P_\infty(q) = AP_\infty(q)A^T - AP_\infty C'\gamma C\, P_\infty A' + Q$

Here, the sensitivity of the measurement noise, namely $R_{q_i}^{-1}$ and R_{q_i} can be readily computed since the matrix is defined as $R^{-1}(q) = diag\left(\frac{q_i}{\sigma_i^2}\right), i = 1..N$. It must be noted that such a computation is possible whenever the *R(q)* is defined as an explicit function of *q*. Comparison of the computational effort and the solution accuracy is presented in Table 8.

Table 8: Comparison of Optimization Time Using Proposed *Lyapunov* Based Method and Finite Differencing

Method	Gradient Calculation	Optimal Cost	Optimal location	Total Optimization time (s)	No. Iterations
BB	Lyapunov	6428.5714	$q_{6,16} = 1$	44	40
BB	Finite-Differencing	6428.5714	$q_{6,16} = 1$	440	40
OA-INLP	Lyapnuov	6428.5714	$q_{6,16} = 1$	23	17
OA-INLP	Finite-Differencing	6428.5714	$q_{6,16} = 1$	215	17

It can be seen that computing covariance sensitivities using solutions of Lyapunov equations improve the algorithm speed by about an order of magnitude.

5.7 Practical Considerations: Constraint Softening for OSP Problem

The motivation for softening the constraints is provided by following example. Consider the refractory wear estimation using the 1-D simplified gasifier model for two different sensor networks are provided in Table 9.

Table 9 : Refractory Wear Estimation for Two Sensor Networks

Sensor set	Refractory wear estimate	Cost	No. Iterations
1: $q_{6,16} = 1$	$0.98\ \beta t_{Brick}$	6428.6	40
2: $q_{6,9} = 1$	$1.02\ \beta t_{Brick}$	6285.7	40

If the constraint was a hard constraint, then feasible space associated with the precision constraint is given as, $\Pi = \{q \in \{0,1\}^N | C^e P_\infty (q) C^{e^T} \leq \beta\ \ t_{Brick}\}$. This would mean that the sensor set 2 which is cost-effective than sensor set 1, will not be selected since it is not a feasible point.

In many practical applications there will always be a tradeoff between cost and the constraints thereby motivating the need to include the constraint softening as a user option. The constraint softening essentially relaxes the feasible space by allowing constraint violation through the problem formulation itself using a slack variable. In order to minimize the violation this slack variable is included to the objective function and the formulation for the soft constrain is given as,

Eq. (75)

$$\begin{gathered} \min_{q,Sl} c^T q + \lambda^{\mathrm{T}} \mathrm{Sl} \\ s_i(q) - Sl_i \leq \bar{s}_i, \mathrm{i} = 1..\mathrm{l} \\ r(q) - Sl_{l+1} \geq \bar{r}_l \\ q_i = \{0,1\}, \quad for\ \ i = 1, 2, \ldots, N \end{gathered}$$

Where, $\lambda \in \Re^{l+1}$is the penalty for constraint violation and $Sl \in \Re^{l+1}$ is the slack variables. The OSP with constraint softening is an MINLP problem as compared to the original INLP problem and includes both real as well as integer variables and hence an extended decision space. It should be noted that for solving the MINLP problem a C++ based implementation for the BB and OA, namely *Bonmin* is available. However, the tool-set developed for the OSP program is Matlab based. Hence the machinery was extended to

include traditional Outer approximation for MINLP problem which is based on the Primal-Master sub-problem (Bonami 2009).

Using the Matlab-based solver which comprises of *Ipopt* (Matlab Interface) and the GLPK as the solvers for NLP and ILP, the machinery was tested by softening the precision constraint for the 1-D refractory wear monitoring problem given as:

Eq. (76)

$$\min_{q,Sl} c^T q + \lambda^{\mathrm{T}} \mathrm{Sl}$$
$$C^e P_\infty(q) C^{e^T} - Sl \leq \beta t_{Brick}$$
$$q = \{0,1\}^{20}, Sl \in \Re$$

The above MINLP problem for constraint softening was solved using both BB and OA approaches for different penalty values and the results are shown in Table 10. Two observations can be made:

- As the slack penalty is increased the constraint violation is reduced which implies that if the penalty value is very high the constraint violation will be zero. The optimal sensor location for a penalty greater than 3.5e4 is the same as the optimal solution with the hard constraint formulation (Eq. (11)). In order to meet the precision requirements more closely, given the increased penalty, more sensors are required and hence the total cost increases. The same trend is observed with the optimization times also due to the fact that an increased penalty tightens the feasible space.

- For applications where constraint violations are not allowed, treating the problem as an INLP problem and using the associated set of tools (BB-INLP, OA_INLP) is more computationally efficient when compared to solving the MINLP problem with a larger penalty as seen in the last row of Table 10.

Many factors affect the performance of the OSP algorithms even after the proposed improvements to the computationally efficiency. The plant size affects the memory requirements; the measurement system affects the decision space and the reliability constraint handling. As mentioned earlier, the estimation reliability (Eq. (10)) is computationally prohibitive even for small systems as shown in the 1D gasifier model case study, due to the evaluation of multiple sensor failure configurations. In order to develop a generic tool-suite that is applicable to solve the OSP problem for small/medium or large scale systems, the two constraints that were considered was the precision and the functional reliability.

Table 10 : Comparison of Optimal Solutions/Time vs Slack With Precision Constraint Softening

Penalty (λ)	**Method / Optimal Solution (q)**	**Slack (Sl)**	**Slack Cost ($\lambda^T Sl$)**	**Sensor cost ($c^T q$)**	**Total cost**	**Time(s)**
1e4	$OA: q_{18} = 1$	0.2358	2358	1142.9	3501.2476	12s
	$BB: q_{18} = 1$	0.2358	2358	1142.9	3501.2476	47
1.5e4	$OA: q_{18} = 1$	0.2358	3537.6	1142.9	4680.42	22
	$BB: q_{18} = 1$	0.2358	3537.6	1142.9	4680.42	145
2e4	$OA: q_6 = 1$	0.09	1799.5	3571.4	5370.9	34
	$BB: q_6 = 1$	0.09	1799.5	3571.4	5370.9	143
>3.5e4	$OA: q_{[6,16]} = 1$	0	0	6428	6428	56
	$BB: q_{[6,16]} = 1$	0	0	6428	6428	162
Hard constraint (N/A)	$OA - INLP : q_{[6,16]} = 1$	[]	[]	6428	6428	23
	$BB: q_{[6,16]} = 1$	[]	[]	6428	6428	44

A Matlab based INLP framework was proposed and the associated machinery was developed using the state-of-art solvers, which comprises of BB-INLP and OA-INLP. With the multitude of the optimization algorithms for solving INLP problem it was decided that instead of one algorithm, a suite of OSP design tool should be developed combining multiple algorithms for solving an OSP optimization problem based on size and requirement of the problem in hand. A description about the features of the framework along with summary of its applicability is shown in the Table 11. A brief summary about the modular OSP tool is presented in Appendix A.2.

Table 11: Summary of the Tool-suite for Solving OSP Problems

Tool Features		**Applicability**
Two Algorithms	BB: 1. LMI 2. Ipopt	Better option when the feasible space is included in the decision space (Eg. Estimation reliability) LMI is a better solver for small scale systems. Ipopt works for large scale systems as well
	OA: 1. Integer cut 2. Feasibility pump	Better than BB when the feasible space approximations through hyper planes are included (Precision and functional reliability)
Three indicators	Precision	• Precision constraint is convex, and evaluated for small, medium and large scale systems • Analytical and efficient computation of Jacobians
	Functional Reliability	Functional reliability for medium and large scale systems (RSC and 3D)
Indicator Enforcement	1. Soft constraint 2. Hard constraint	The two options are there to provide a way to factor the cost vs constraint trade-off.

In the next section the modular tool was tested on solving the OSP problems for the refractory wear monitoring using the full order 3D model as well as the fouling monitoring of the RSC model. The relevant modeling assumptions along with the algorithm performance are discussed below.

6 Task 4: Demonstration of OSP Algorithm Performance

In this task, the developed OSP algorithm in Task3 was applied to find solution to two optimal sensor placement problems – a) OSP problem for condition monitoring of full-fledged 3-D gasifier refractory wear and b) OSP problem for condition monitoring of RSC fouling. The solution (sensor location) thus obtained for each of these problems was used in a model based sensing framework to verify if with these sensor set the variable of interest (refractory wear in case of the gasifier and fouling profile in case of the RSC) could be estimated correctly with the design accuracy.

6.1 Solution to OSP problem for condition monitoring of gasifier refractory wear and RSC fouling

The problem size of gasifier refractory wear OSP problem poses significant challenges due to the target platform namely windows Matlab©, limitation. The estimation algorithm used within the OSP algorithm cannot handle models having more than couple thousand states. The 3-D gasifier model that represents the actual gasifier closely has more than 9000 states and 3000 possible sensor locations. Hence a reduced size gasifier model was identified.

6.1.1 Reduced Size Gasifier Model for the OSP Algorithms

In this section a methodology to represent the larger size gasifier model by an “equivalent” smaller size model for OSP purpose is proposed. This methodology exploits the axial symmetry and the special nature of thermal model that represents the gasifier dynamics. Similar techniques can be applied to other models having similar characteristics.

Consider the gasifier model response to a non-uniform wear profile shown in Figure 15 (Section 4), where a few nodes at mid axial location $z = 5$ and around $\theta = 0$ have a localized wear profile up to ~35% of the inner-most brick layer thickness, while the rest of the bricks have zero wear. The thermal response of a gasifier due to refractory wear exhibits two characteristics:

a) Dominant temperature difference within the gasifier close to the wear location (axial and azimuth). The temperature quickly goes to no-wear value away from this location.
b) The thermal response does not depend on the azimuthal location of the wear.

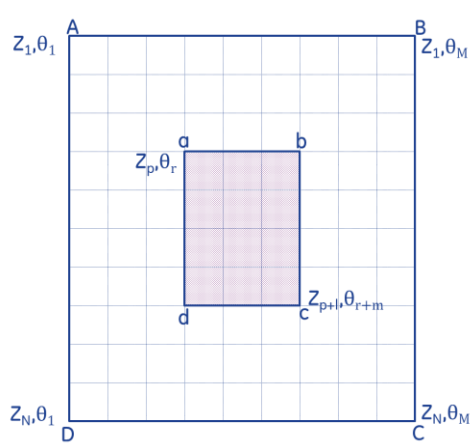


Figure 42: Hot surface of the 3-D refractory model

Due to above two characteristics, it is possible to utilize only part of the gasifier model for the OSP problem and repeat the solution thus obtained to find optimal sensor set for the complete gasifier OSP problem. The large number of the states in complete 3-D gasifier model arises from the fine grid required to adequately represent the thermal gradient

across the refractory. Consider a 3-dimensional refractory block, 2 dimensional hot surfaces of which is denoted by *abcd* in Figure 42. The block is l-grid long in the axial direction and m-grid wide in azimuth direction. The block is a part of the complete refractory, 2 dimension hot surfaces of which is denoted by *ABCD*. The radial dimension the block is same as the original 3-D gasifier model.

The idea is to explore the possibility of representing the complete refractory *ABCD* by a much smaller block *abcd* for the OSP algorithm purpose. Recall that it is the estimator algorithm within the OSP algorithms that is limited by the memory requirement of the target implementation machine. Consider the gasifier model representation within the estimator as in Eq. (3). The term w_k in Eq. (3) represents the process noise and is used to account for un-modeled system dynamics. It is possible to use this term to model the "left out" part of the gasifier *ABCD* while considering the smaller block *abcd* to model the system dynamics in the estimator. For example, the contribution to the state derivatives as in Eq. (3) from the states not included within the block could be accounted as unmodeled dynamics and can be clubbed with noise term w_k for the boundary nodes of the block. This will increase the covariance of the noise for the boundary nodes of the block as shown in Figure 43.

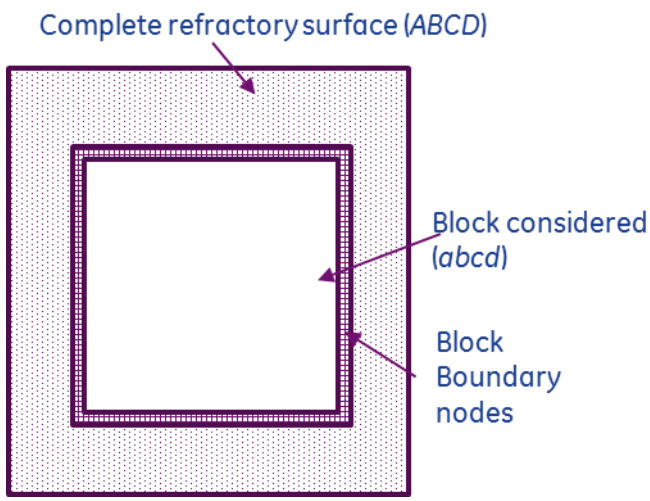


Figure 43: The refractory wear outside the block affects the state derivatives with in the block. That effect can be captured through the process noise model of the block boundary nodes.

The magnitude of this additional noise can be obtained through extensive simulation studies. As an example, consider that the block *abcd* is of size 7x7x15, and the block *ABCD* is of size 20x30x15. In other words the block *abcd* has 735 nodes (or states) whereas the complete gasifier block *ABCD* has 9000 nodes. The maximum wear possible is, say, 4 inches. Now extensive simulation study can be performed where the refractory is given various amount of uniform wear between zero and 4 inches everywhere expect in the smaller block *abcd*. This represents the worst case scenario for the thermal response in the border grids of *abcd* due to sections of the refractory that are ignored in the OSP algorithm. The maximum of all the temperature gradients (i.e., maximum of $\frac{dT}{dt}$, in Eq. (56)) in boundary grid of *abcd* due to various wear magnitude are recorded as shown in Figure 44. This maximum magnitude of temperature gradients, then, is used to compute the 3σ value of the noise term w_k over and above nominal noise term in Eq. (4). In the

figure the nodes are ordered axially (i.e., first fixed radial and azimuth direction and move axially) before moving azimuthally and then finally radially. From the figure it can observed that the maximum modeling error occur at the corner nodes of the blocks as expected because the corner nodes temperature gets affected from both axial as well as azimuthal thermal flow (apart from radial flow of thermal energy), whereas all other boundary nodes only get affected either by the axial or by the azimuthal flow of thermal energy (again, apart from the radial flow of energy) as the case may be.

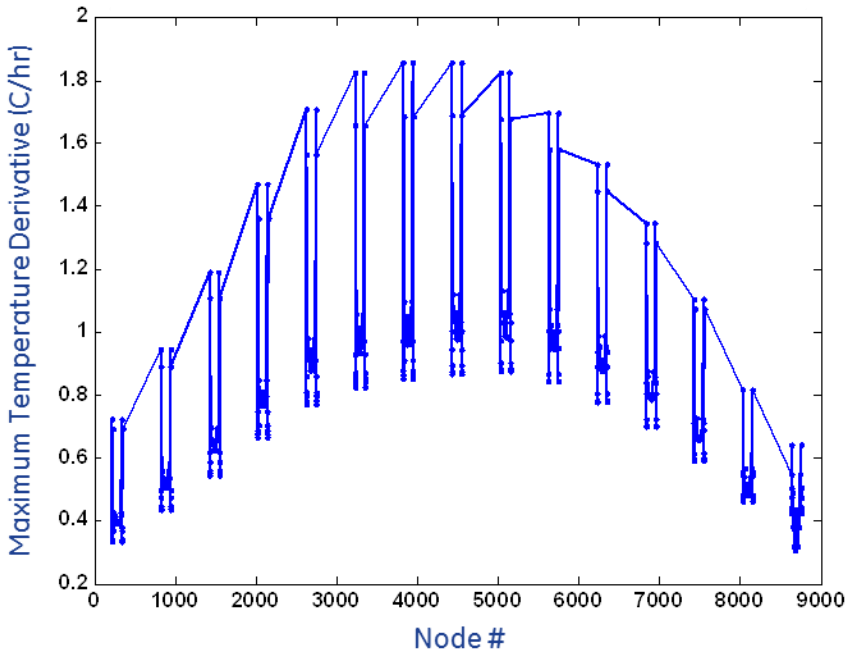


Figure 44: Maximum temperature derivatives for each boundary nodes of the smaller block *abcd,* for various wear scenarios within complete gasifier *ABCD*, are recorded in simulation. The magnitude of the process noise (3σ) for the boundary nodes are increased by the noise magnitude thus obtained.

As mentioned earlier, it is the estimation algorithm that restricts the size of the model that can be used on a given target operating system. To verify that the assumption of using a small size block here for applying the OSP algorithm and then replicating the solution to find the solution of the OSP problem of the full-fledged 3D gasifier is valid a simple experiment was conducted. The precision values for the wear were obtained by solving the Algebraic Riccati equation (Eq. (7)) for the full 3D model as well as that of a small 5x5x15 wedge block carved out from the full model using the same noise covariance matrices R & Q. The histogram of the precision thus obtained in shown in Figure 45. The figure shows that both the histograms are statistically quite similar.

Using the techniques outlined in this section earlier, the model size is reduced for the OSP purpose. One of the design decision required to be made for the OSP problem related to gasifier refractory, is the size of the smaller block *abcd*. From the estimation point of view it could be argued that the small size problem as obtained earlier is “equivalent” to the original larger size problem (though, this is difficult to prove rigorously). However, from the OSP solution point of view it is easy to show that both are not equivalent. To understand

this, consider the limiting case of size one for the smaller block. Due to the absence of any analytical redundancy, in order to do any condition monitoring the OSP algorithm will have only trivial solution of placing sensor on each block in order to meet any non-zero precision requirement. However, for a larger block it is possible to construct scenario where the OSP algorithm can take advantage of the analytical redundancy available in the model and thus reducing the total number of sensor required to meet the same condition monitoring requirements. Intuitively, the size of the smaller block *abcd* should be as large (i.e., close to the complete model *ABCD*) as possible within the constraints of memory requirement for the estimation algorithm. In other words the upper limit on the size of the smaller model *abcd* is dictated by the memory requirement. The lower size limit of the model, however, gets restricted by the added process noise for boundary nodes. This is because; the higher process noise reduces the bandwidth of the estimation filter thus making its convergence very slow. For condition monitoring with respect to gasifier refractory wear, the OSP algorithm uses solution to steady state Riccati equation to compute quantities relevant to precision constraints. It is implicitly assumed that the steady state of the estimation filter is reached within a reasonable time (3 months). Once the solution to the OSP problem involving the smaller block *abcd* is obtained, the solution to the original gasifier OSP problem can be obtained due to azimuthal symmetry of the gasifier model and the gasifier operation.

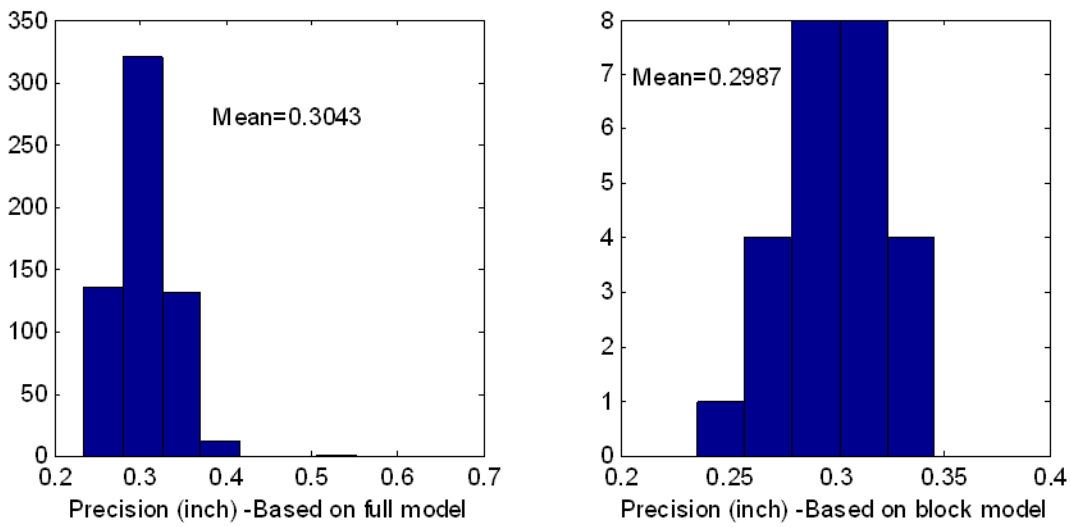


Figure 45: Comparing precision of the wear estimates obtained from a small wedge block of the gasifier to that obatined from the full gasifier

In order to obtain the solution to the gasifier OSP problem following heuristic is proposed:

- First identify an “equivalent” smaller size model.
- Solve the OSP problem for this small size model by imposing the condition monitoring requirement constraints only at nodes away from the boundary as shown in Figure 46. This means the sensors would be placed wherever they are required within the smaller block but the requirement of precision and reliability would be

imposed on the double shaded block as shown in Figure 46. This is done to minimize the effect of artificial boundary created by the smaller block.

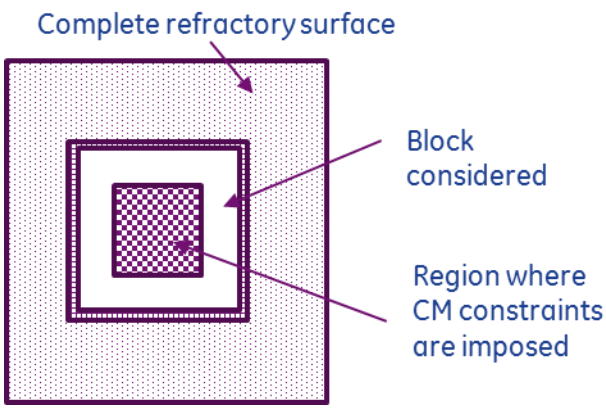


Figure 46: Solve the OSP problem for the block considered by imposing CM requirement constraints only on part of it.

- Extend the sensor placement solution to the entire gasifier by repeating the optimal sensor placement solution obtained for the smaller block and shifting the block so that the region where CM constraints were imposed are just overlapping and taking the union of sensor placement as show Figure 47.

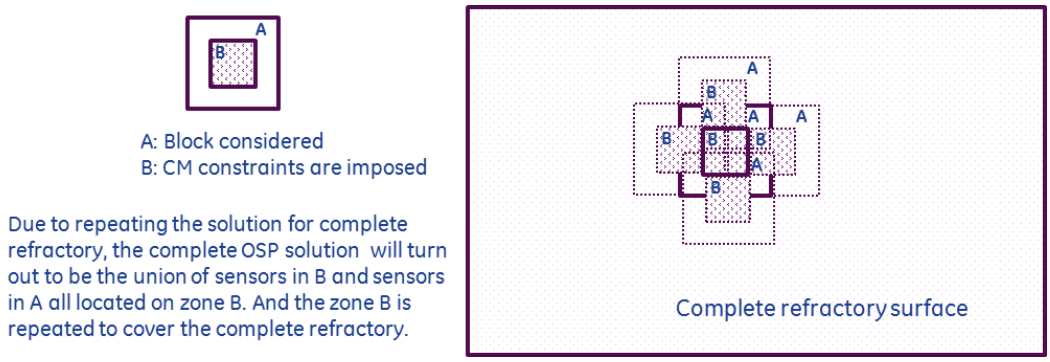


Figure 47: Extend the OSP solution obtained using smaller block to obtain OSP solution for complete refractory gasifier by repeating the solution.

6.1.2 Optimal Sensor Placement Results for Reduced Order 3D Model

Using the above mentioned heuristic, for the purpose of the OSP algorithm, the 3D gasifier layer is represented by a reduced wedge that comprises of a 5 x 5 x 15 block (Figure 42), where there are 5 grids in z and θ directions and 15 in the radial direction leading to 375 temperature states. The temperature measurements are available only at the interfaces thereby leading to a 125 possible measurements. Since the interface layers at 3 and 4 are very close to each other only the measurements at interface 4 is considered leading to a total of 100 possible sensing locations. The linearized system corresponding to the

reduced or 3D model represented by the wedge is obtained by truncating the linearized model for the full 3-D gasifier and is given by,

Eq. (77)

$$x_{k+1} = A_k x_k + w_k, \mathrm{A_k} \in \Re^{n\times n}$$
$$y_k = C_k x_k + v_k\ , C_k \in \Re^{n\times N}$$

Where, $n = n_z n_\theta\, n_r + 2n_z n_\theta = 400$ and N=100. The standard deviation used in the computation of the measurement noise covariance matrix is assumed to be 1% of nominal measurement which is based off the sensor characteristics. The process noise characteristics for the temperature states is a small percentage of the initial state and that of the ramp state is assumed to be a1σ variation from the nominal wear rate, which is computed as the maximum wear over a period of normal maintenance cycle. The solution to the OSP problem for condition monitoring of the 3D gasifier refractory wear is obtained by solving the optimization problem Eq. (11), where the constraint imposed are the precision constraint (Eq. (6)) and the reliability constraint (Eq. (9)). Due to size of the problem, the estimation reliability is prohibitively expensive to compute, where a single computation of the estimation reliability metric would require the precision evaluations 2^{100}times. For the same reason the notion of functional reliability is considered even for the computing the OSP solutions of the reduced order 3D wedge model,

The OSP problem for the wear monitoring using the 3D wedge model is an integer nonlinear programming problem (INLP) with a total of N= 100 integer decision variables with $2\mathrm{n_z n_\theta}$ nonlinear constraints. The large dimensional search space due to the increased number of possible sensor locations, combined with the enumerative nature of the BB algorithm makes the BB method inapplicable to the 3D OSP problem, making the outer approximation only viable alternative.

To compute the OSP solutions, a hypothetical sensor cost similar to the 1D model is considered, where the cost of placing sensors at a given layer is the same and the cost increases as the layers are radially inward closer to the wear surface as shown below. The sensor reliability is maximum at the outer-shell and is minimum in innermost interface given the harsher operating conditions and is shown in Figure 49. The requirement on the estimation precision using the 3D gasifier model is considered to be around 4% of the brick thickness and the reliability is considered to be 90%.

One of the features of the OSP tool developed is that it enables using a-priori knowledge of infeasible sensor locations which can reduce the search space thereby improving the computational speed. This feature is exploited in getting the solutions to the OSP problem , given the large-scale dimensions of the OSP problem, even when utilizing the 3D wedge approximation.

The OSP algorithm was run with an initial guess of placing the sensors at all locations and the optimal sensors placement solutions from the algorithm would be to place sensors at all locations at node 16 as shown in the following figure. While a single sensor would meet the estimation precision, meeting the reliability constraints would require computing the minimum number of sensors requried to increase the survival rates of each sensor. The reliability analysis using the 1D model was extended to the reduced order 3D model. It

was computed that placing 3 sensors at all the locations in the first interface (node 16) with the exception of the indices 6,22 and 25 which required 4 sensors will meet both reliability and precision requirements. The schematic representation of the OSP solutions on the 3D model is given in Figure 50, where the green indicates that a sensor is placed at the indicated location and gray indicates the absence of a sensor at the specified location.

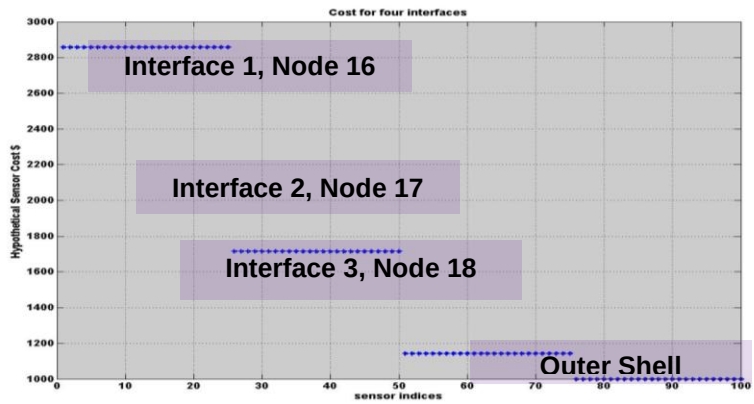


Figure 48: Hypothetical Sensor Cost

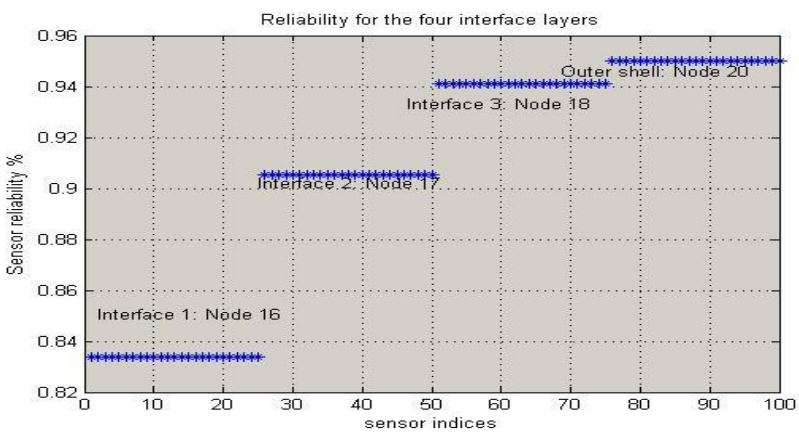


Figure 49: Sensor Reliability

The OA algorithm converged in one iteration within 2950 sec for the 3D model with 100 binary variables and the the progress of the OA iterates, namely the generation of the upper and lower bounds are shown in . Figure 51

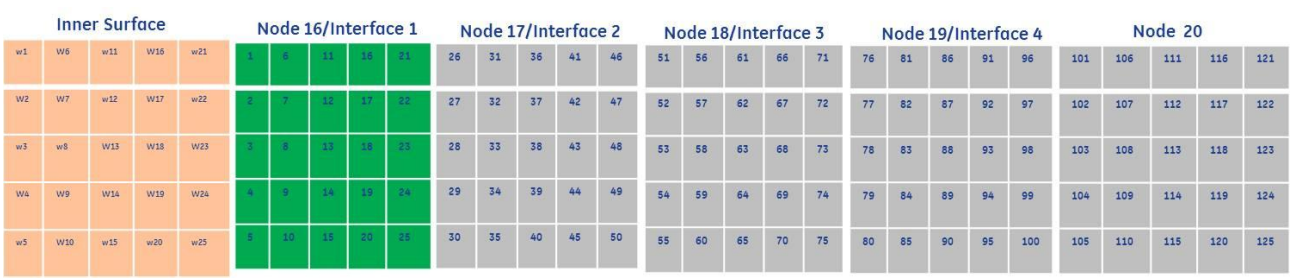


Figure 50: OSP Results for $s \leq .044 t_{brick}. r \geq 0.9$

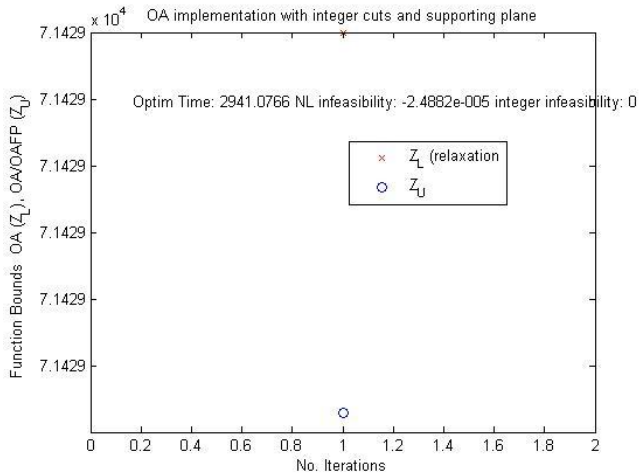


Figure 51: OA bound generation

A comparison of the sensor locations as well as the estimation precision of the initial guess as well as the optimal solutions for the 3D model is shown in Figure 52.

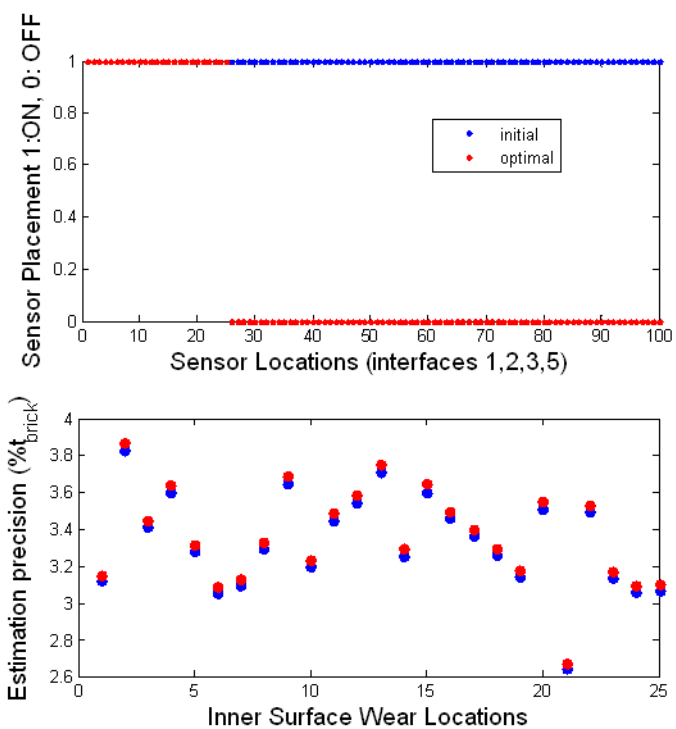


Figure 52: Wear Estimate Precision of the innermost surface

It can be seen that the estimation precision of the optimal sensor network is slightly larger than that of the placing sensors at all locations but at a much reduced cost and satisfies the precision constrain on all the 25 grids on the inner wear surface.

A first glance at the OSP results poses an important question as to why would it be sufficient to place sensors only in the first layer. The following observation are useful in explaining the optimality of the OSP solution.

1. The most important observation is the radial dominance exhibited by the 3D gasifier model, where the temperature states in any given grid are effected by the states in the radial direction as opposed to those in the azimuth (z) or θ directions. The radial dominance necessitates a sensor placement directly behind the wear surface.
2. Even though the strategy is to place the sensor radially behind, how much farther behind will be decided on the estiamtion accuracy desired. For example since the desired accuracy is 4/10”, there placement of sensors at node 16 was the option.

6.1.2.1 Analysis and Remarks

In order to validate the above observations, two results are presented. First a comparison of the estimation accuracies of the OSP solution with different possible sensor networks is presented in order to highlight observation 1 and 2.
Then a comparison of the networks in terms of cost is presented in order to highlight the optimality of the OSP solution. The different networks considered are as follows:

- Initial Guess which is to place the sensors at all 100 locations in interface 1,2,3 and 5 spread across nodes 16, 17, 18 and 20.
- The OSP solution which is to place the sensors in interface 1 alone.
- The sensor network which comprises of placing sensors only in the interface 2.
- Case1 which is place 24 sensors on interface 1 and one sensor in interface 2Case2 which is to move 2 sensors from interface 1 to interface 2.

The sensor network corresponding to the OSP solution is shown in Figure 50 and the networks corresponding to the cases 1 and 2 are shown in Figure 53 and Figure 54.

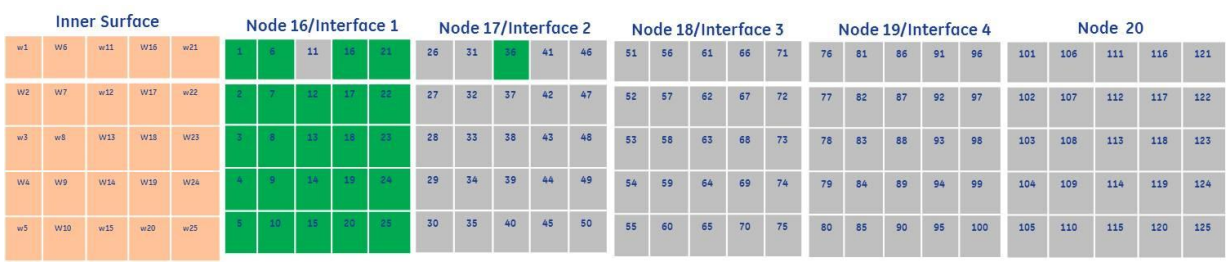


Figure 53: Sensor network for case 1

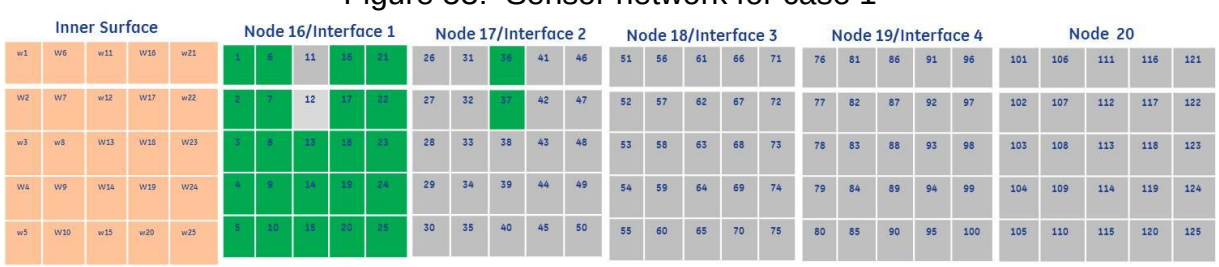


Figure 54: Sensor network for case 2.

It can be seen from Figure 55 that placing all the sensors in interface 2 yields a minimum and maximum estimation accuracy of 6.7% and 10% of brick thickness respectively. This means that it would not be possible to achieve 4/10” requirement on a given wear surface if the sensor behind that surface is placed in the second interface. It can be seen that the precision constraint is violated in sensor networks corresponding to cases 1 and 2, where 1 or 2 sensors are moved from the 1st interface to the 2nd. The constraints are violated in

the surface 10 or 11 (case 1 or 2), where the sensor location has been moved from interface 1 to 2 illustrating the radial dominance on the minimum achievable precision (Observation 2).

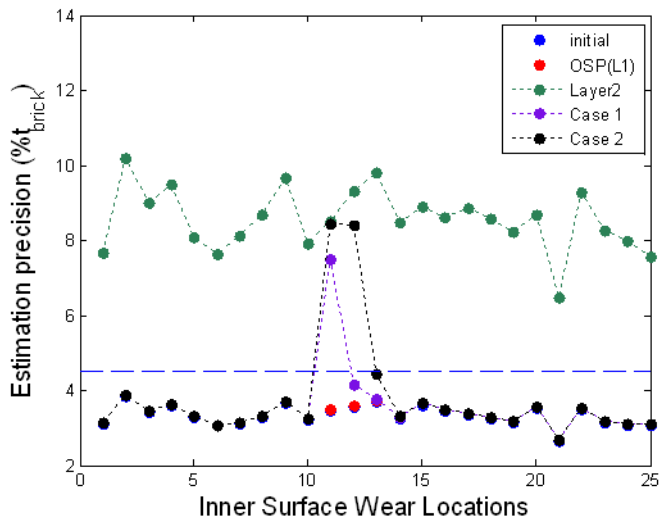


Figure 55: Comparison of the OSP solution with cases 1 - 5

The comparison of cost and estimation precision is shown in Figure 56 for the different sensor networks. It can be seen that the low-cost option while meeting the precision requirement is the OSP solution thereby illustrating the optimality of the solution. For the specific application, constraint softening was not applicable given that the estimation accuracies were quite different at different interfaces, where the interface 2 could meet a threshold of 10% as compared to the desired accuracy of 4.5% of the brick thickness.

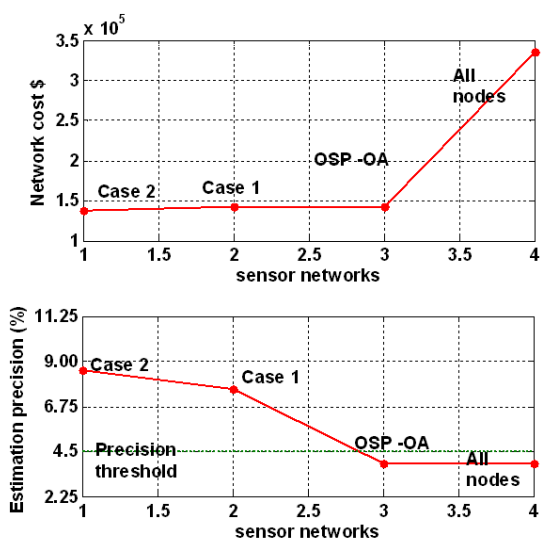


Figure 56: Cost, precision comparison of OSP solution.

6.1.3 OSP Solution to the Detailed 3D Gasifier Model

The sensor placement for the entire refractory layer will be computed by repeating the OSP solutions for the block considered above due to the symmetry of the underlying model. The most important observation from the above analysis is the radial dominance in estimating the wear, where the optimal solution is to place the sensor radially and the location depends on the estimation requirements. Because of the dominance and the symmetry of the 3D model, the OSP solution for the full order model can be obtained by repeating the solution of the 5x5x15 wedge all along the theta and z direction. In the rest of the discussion, the exact solution is presented.

The 3D model consists of 30 grids in the theta direction and 20 grids in the z direction leading to a total of 600 grids for each of the 15 layers. For the rest of the discussion a top-down and left-right numbering is adopted and the following nomenclature is used for the sensor placement. The grids in interface 1 (node 16) are numbered from 1 to 600 and the corresponding optimization variables are denoted by $[q_1..q_{600}] \in \{0,1\}$, where a value of 1 indicates the presence of a sensor at the location and 0 otherwise. Similarly the grids in interfaces 2 through 5 (node 17-20) are numbered from 601 to 3000 and the corresponding optimization variables are denoted by $[q_{601}..q_{30200}] \in \{0,1\}$. Let $n_i, i \in \{1,..,3000\}$ denote the number of sensors at each measurement location. Let I_1 denote the following index sets:

$$I_1 = [21\ 26\ 31\ 36\ 82\ 85\ 87\ 90\ 92\ 95\ 97\ 100\ 121\ 126\ 131\ 136\ 182\ 185\ 187\ 190\ 192\ 195\ 197\ 200\ 221\ 226\ 231\ 236\ 282\ 285\ 287\ 290\ 292\ 295\ 297\ 300\ 321\ 326\ 331\ 336\ 383\ 385\ 387\ 390\ 393\ 395\ 397\ 400\ 421\ 426\ 431\ 436\ 482\ 485\ 487\ 490\ 492\ 495\ 497\ 500\ 521\ 526\ 531\ 536\ 582\ 585\ 587\ 590\ 592\ 595\ 597\ 600]$$

Then the OSP solution for the 3D model, whose schematic is shown in Figure 57, is given as follows:

- $q_i = \begin{cases} 1, \forall i \in \{1..600\} \\ 0, \forall i \in \{601..3000\} \end{cases}$,
- $n_i = \begin{cases} 3, \forall i \in \{1..600\} \backslash I_1 \\ 4, \forall i \in I_1 \\ 0\ \forall i \in \{601..3000\} \end{cases}$

Sensor Placement in all measurement locations in node 16/Interface 1

1	21	41	61	81	101	121	141	161	181	201	221	241	261	281	301	321	341	361	381	401	421	441	461	481	501	521	541	561	581
2	22	42	62	82	102	122	142	162	182	202	222	242	262	282	302	322	342	362	382	402	422	442	462	482	502	522	542	562	582
3	23	43	63	83	103	123	143	163	183	203	223	243	263	283	303	323	343	363	383	403	423	443	463	483	503	523	543	563	583
4	24	44	64	84	104	124	144	164	184	204	224	244	264	284	304	324	344	364	384	404	424	444	464	484	504	524	544	564	584
5	25	45	65	85	105	125	145	165	185	205	225	245	265	285	305	325	345	365	385	405	425	445	465	485	505	525	545	565	585
6	26	46	66	86	106	126	146	166	186	206	226	246	266	286	306	326	346	366	386	406	426	446	466	486	506	526	546	566	586
7	27	47	67	87	107	127	147	167	187	207	227	247	267	287	307	327	347	367	387	407	427	447	467	487	507	527	547	567	587
8	28	48	68	88	108	128	148	168	188	208	228	248	268	288	308	328	348	368	388	408	428	448	468	488	508	528	548	568	588
9	29	49	69	89	109	129	149	169	189	209	229	249	269	289	309	329	349	369	389	409	429	449	469	489	509	529	549	569	589
10	30	50	70	90	110	130	150	170	190	210	230	250	270	290	310	330	350	370	390	410	430	450	470	490	510	530	550	570	590
11	31	51	71	91	111	131	151	171	191	211	231	251	271	291	311	331	351	371	391	411	431	451	471	491	511	531	551	571	591
12	32	52	72	92	112	132	152	172	192	212	232	252	272	292	312	332	352	372	392	412	432	452	472	492	512	532	552	572	592
13	33	53	73	83	113	133	153	173	193	213	233	253	273	293	313	333	353	373	393	413	433	453	473	493	513	533	553	573	593
14	34	54	74	84	114	134	154	174	194	214	234	254	274	294	314	334	354	374	394	414	434	454	474	494	514	534	554	574	594
15	35	55	75	85	115	135	155	175	195	215	235	255	275	295	215	335	355	375	395	415	435	455	475	495	515	535	555	575	595
16	36	56	76	96	116	136	156	176	196	216	236	256	276	296	316	336	356	376	396	416	436	456	476	496	516	536	556	576	596
17	37	57	77	97	117	137	157	177	197	217	237	257	277	297	317	337	357	377	397	417	437	457	477	497	517	537	557	577	597
18	38	58	78	98	118	138	158	178	198	218	238	258	278	298	318	338	358	378	398	418	438	458	478	498	518	538	558	578	598
19	39	59	79	99	119	139	159	179	199	219	239	259	279	299	319	339	359	379	399	419	439	459	479	499	519	539	559	579	599
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400	420	440	460	480	500	520	540	560	580	600

3 sensors 4 sensors

Figure 57: Node 16 (Interface 1) sensor placement

6.2 Optimal Sensor Placement for Condition Monitoring of RSC Fouling

Recall that the fouling profile of the RSC heat transfer tubes were parameterized by using six independent fouling parameters that can vary between zero and one; zero signifying no fouling whereas one signifies fully fouled RSC zone. For monitoring the condition of the RSC heating tube, the sensors are needed to be placed so that these six fouling parameters can accurately be identified. The fouling parameters are modeled as a first-order Markov process. The IGCC-RSC model has a total 207 states from gasifier-slag model, RSC, Drum, Quench, and Scrubber, which makes the OSP problem a medium-large optimization problem. In the previous program (DE-FC26-07NT43094), certain sensors (36 to be precise) were identified that were needed for the control purpose, such as the syngas temperature and pressure measurements form RSC to Quench, steam flow from Drum, water flow from Scrubber etc. For optimal sensor placement problem, it was assumed that this sensor set (36 sensors) that are normally present in an IGCC plant for monitoring and controlling the process will remain available for condition monitoring for RSC tube fouling. Hence, the OSP problem for monitoring RSC fouling seeks the best (lowest cost) additional sensors that can satisfy the precision and reliability constraints. These additional sensors include heat flux sensor and the strain gauge sensor (referred to as ebolt sensor in this report) on the mechanical support structure. Since there are six fouling parameters that needed to be identified, the additional sensors that may be needed for condition monitoring of RSC fouling would be a most six heat flux sensors. As mentioned earlier, the ebolt sensor also has significant signature of the cumulative fouling

and is cheaper compare to a heat flux sensor the OSP problem here involves finding the lowest cost sensor set, $q \in \Re^7$,that can satisfy the precision and reliability constraints.

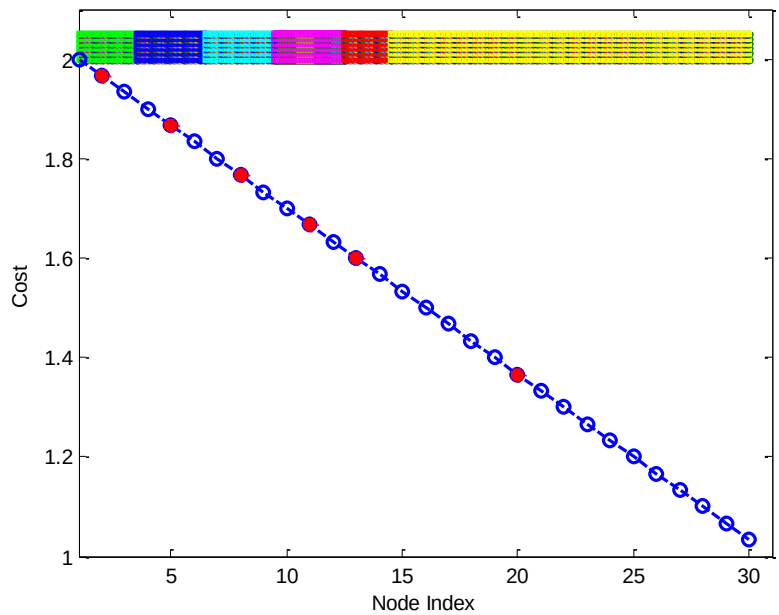


Figure 58: Heat flux sensor location vs. sensor cost

In the RSC model, there are 30 possible locations for the heat flux sensors. The heat flux sensor cost as a function of location is shown in Figure 58.
The cost to install heat flux sensors is assumed to be relatively expensive than other sensors, for example, the strain sensor. Also, the reliability of heat flux sensors along the tube is assumed to vary from each other based on their location. As the syngas gets cooler due to heat transfer inside the RSC tube, the temperature on the top of the RSC is expected to be higher than the bottom. Accordingly, the sensors placed on the top are less reliable than the ones placed on the bottom. The sensors are assumed to have sensor cost $c \in \Re^7$ as shown in Figure 58 and corresponding individual sensor failure rates $r \in \Re^7$ (Figure 59).

In Figure 58 and Figure 59, the x axis shows the node index, indicating 30 heat flux location along the length of RSC tube, with Index 1 being the sensor on the top and Index 30 being the on the bottom. The y axis in Figure 58 shows corresponding sensor cost. The color bar shows the region of zones, for instance the green bar covers the node index from 1 to 3, which implies that zone 1 has three locations for heat flux sensors, etc. As explained previously, we select one sensor in each zone to be the potential location to put sensor, which is plotted as red dot, whereas the blue ones are the locations that won't be considered for sensor locations. The cost of the strain sensor is defined as $c_7 = 1$ with index "7". It must be noted that while the absolute values for the cost is not of any significance here

It is assumed that the survivability of the sensors on the top of the RSC is lower than the ones on the bottom, due to the higher temperature and presence of more corrosive

environment at the top-middle section. For the same reason, the cost of heat flux sensors are assumed to decrease along the length of RSC from top to bottom.

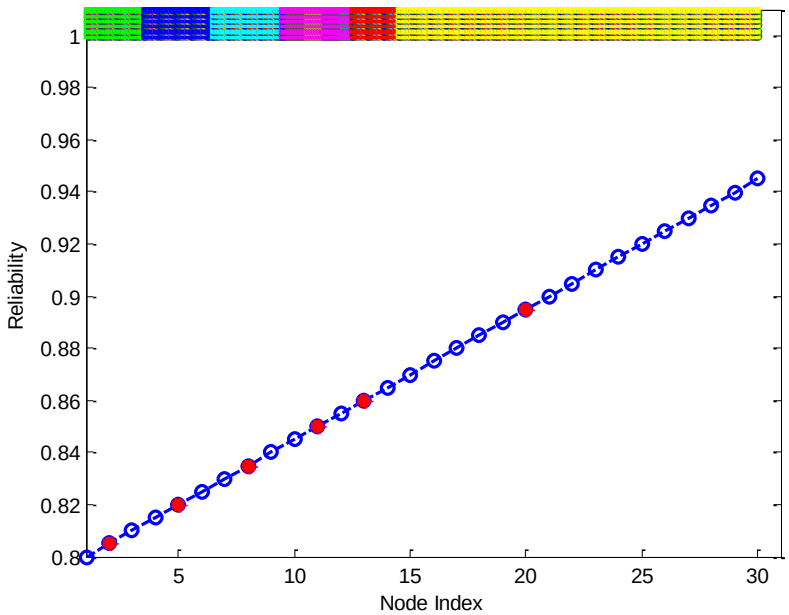


Figure 59: Heat Flux sensor reliability vs. sensor location

The y axis of Figure 59 shows the reliability of sensor at each location. The strain sensor is relatively more reliable than heat flux sensor, and it is defined as $r_7 = 0.95$.

The constraints imposed on the sensor network are defined as follows:

1. Estimation precision : For an fully fouled RSC tube f_{RSC} is given by:

Eq. (78)

$$C^e P_\infty (q) C^{e^T} \leq \frac{10}{100} \ f_{RSC}$$

Where f_{RSC} is defined as the thickness of the fully-fouled RSC.
The OSP problem for condition monitoring of RSC fouling can be categorized as a medium-large scale problem with more than one hundred states for RSC model. The branch and bound based approach was used to solve this problem.

2. Functional reliability:

Eq. (79)

$$r_l^h = 1 - \prod_{i=1}^{n} [1 - r_i^s q_i]^{n_i} \geq 0.9$$

$$q_i = \{0,1\}, for\ i = 1,2, \dots, N;\ n_i = 1,2, \dots, k.$$

In the above expression, *N* denotes the total number of sensor locations n_i denotes the number of sensors at each location $i = 1, \ldots, N$, and r_i^s is the intrinsic reliability of each sensor i used to monitor process *I*; and r_l^h denotes the overall reliability of the sensor network for estimating or monitoring the l^{th}process variable as defined in

To solve the OSP problem with the precision constraint and reliability constraint, a two-step approach is followed in the framework of branch-and-bound algorithm. First the NLP problem is solve with only the precision requirement imposed. This solution gives lower bound in the branch-and-bound algorithm to obtain the integer solution. Such an integer solution is augmented at the leaf node of branch-and-bound algorithm, as in Eq. (56) with implementing adequate redundancy in sensor location to allow meeting the desired precision in the presence of expected sensor failure probabilities.

With only the precision requirement, the following optimal sensor configuration result is obtained.

Table 12: OSP solution using NLP-BB for RSC fouling problem

Location	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Zone 6	bolt
Rel. Cost	1.9967	1.8667	1.7667	1.6667	1.6	1.3667	1
solution							

Eq. (80)

$$\min_{\substack{c_i \\ q_i \in \{0,1\}, i=1,2,\ldots,7}} \sum c_i q_i = 4.0333$$

$$\{q_4 = 1, q_6 = 1, q_7 = 1\}$$

Since the cost of sensors located at bottom is less expensive than the ones on the top and the strain sensor has the least expensive cost, the minimum cost is obtained by selecting two heat flux sensors at Zone No.4 and Zone No.6, together with one strain sensor so that the precision requirement is satisfied with minimum cost. This result is expected from the equal cost test in previous section, which is the confirmation from the OSP algorithm.

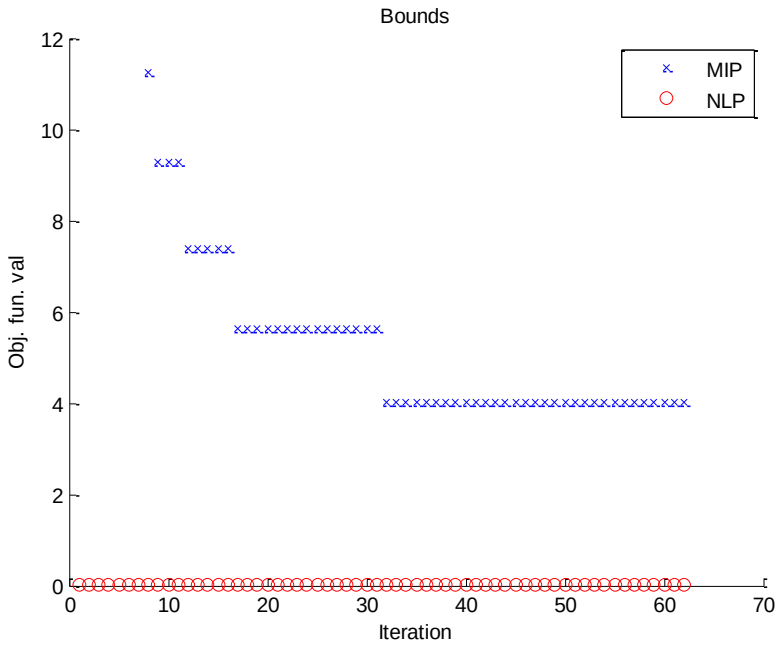


Figure 60: Branch-and-Bound Iteration result with NLP relaxation: precision only

The Figure 60 shows the iteration result by using the Branch-and-Bound and NLP relaxation.

If both, precision and reliability requirements are imposed then the following optimal sensor configuration result is obtained.

Table 13: Complete solution to OSP problem using NLP-BB for RSC fouling problem

Location	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Zone 6	bolt
Rel. Cost	1.9967	1.8667	1.7667	1.6667	1.6	1.3667	1
Reliability	0.805	0.82	0.835	0.85	0.86	0.895	0.95
Solution							
Number				2		1	2

Eq. (81)

$$\min_{\substack{c_i, r_i \\ q_i \in \{0,1\}, i=1,2,\ldots,7 \\ n_i = 1,2,3}} \sum c_i q_i n_i = 6.7$$

$$\{q_4 = 1, n_4 = 2;\ q_6 = 1, n_6 = 1;\ q_7 = 1, n_7 = 2\}$$

The result states that it requires two heat flux sensors at Zone No. 4, one heat flux sensor at Zone No.6 and two strain sensors at bolt to satisfy the precision and reliability requirement.

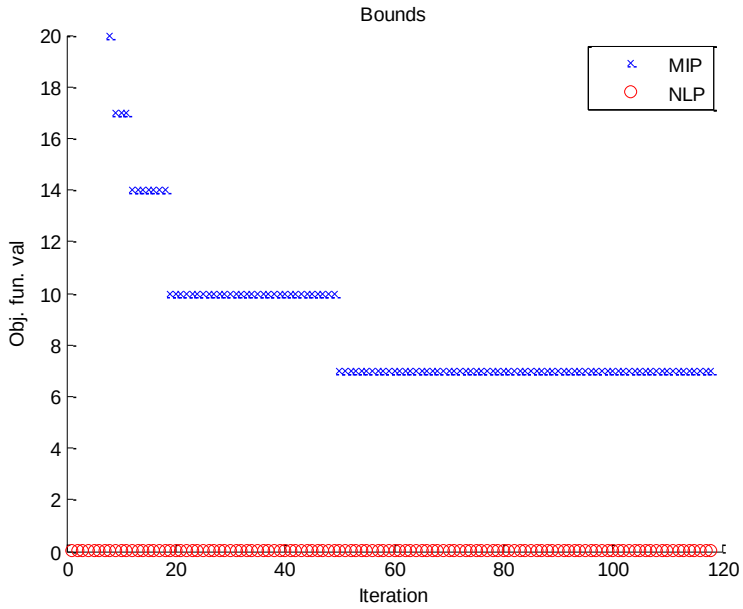


Figure 61: Branch-and-Bound Iteration result with NLP relaxation: unequal cost, precision and reliability

The Figure 61 shows the iteration result by using the Branch-and-Bound and NLP relaxation for unequal cost with precision and reliability requirement.

6.3 Assessing the performance of the OSP algorithm

The application of model based state estimation, which combines information from an analytical process model with direct actuator and sensor measurements forms the basis for model based sensing tool used here to validate the OSP solutions. This section summarizes the general structure of this model based tool used. Using models for corresponding processes the model based sensing framework can be customized for that particular application. Due to matrix size limitation posed by the operating system running the Matlab\Simulink©, the validation tool for OSP algorithm solution for gasifier refractory wear (recall that gasifier model has more than 9000 states) is slightly different than that of the RSC fouling profile. They both are based on extended Kalman filter formulation; however they differ slightly in the way they are implemented. The code for the gasifier is implemented in Matlab© whereas the one for the RSC fouling is implemented in Matlab\Simulink©.

A typical model based sensing system also known as virtual sensing, is shown Figure 11. In virtual sensing, the measurement (actual) from the plant is compared with response (estimated) of the dynamic model of the plant to the same input as the actual plant. Kalman filter uses this difference between the actual and the estimated response of the plant along with the plant model to estimate parameters like refractory wear or RSC fouling profile. The extended Kalman filter formulation is similar to that of Section 4.1.2. This estimator for the gasifier refractory wear is implemented to run in the Matlab© environment uses a detailed nonlinear model of refractory heat transfer coupled with distributed

temperature sensor measurements as described in Section 4.3 to estimate the depth of the wear at each defined grid point (node) in the refractory liner. In addition to demonstrating via simulation that the liner wear can be accurately estimated using the full radial temperature sensor set at each node, the estimator will basically be used to verify the near identical performance accuracy achieved using the reduced sensor set identified by the Optimal Sensor Placement study.

This estimator for the RSC fouling is implemented to run in the Matlab\Simulink© environment uses a linear model derived from the detailed nonlinear physics based model of the gasification section model of an IGCC plant consisting of gasifier, RSC, quench, drum boiler and scrubber very similar to the one used in the previous program (DE-FC26-07NT43094) with having additional heat flux and bolt strain measurement sensors as described in Section 4.4. In addition to demonstrating via simulation that the RSC fouling profile can be accurately estimated using the additional heat sensors, the estimator will basically be used to verify the near identical performance accuracy achieved using the reduced sensor set identified by the Optimal Sensor Placement study.

6.3.1 Assessing solution to OSP problem for CM of gasifier refectory wear

In order to assess solution to OSP problem for condition monitoring of refractory wear a Monte Carlo simulation was conducted where randomly generated wear ramps were used to simulate multiple cases of worn out refractory within the innermost surface of the gasifier. For one such case the randomly distributed wear is shown in Figure 62. The maximum wear is clamped at 4 inch. It should be noted that the wear distribution generated here is completely uncorrelated in the sense that the grids next to each other might have completely different wear; one might have maximum wear of 4 inch whereas the next one might be close to zero inch (e.g., wear grids corresponding to azimuth location 4 and 5 at axial location 16) . In reality this is a highly unlikely scenario. Normally the nearby locations will have similar wear. Nonetheless the wear distribution used here is very challenging for a model based sensor.

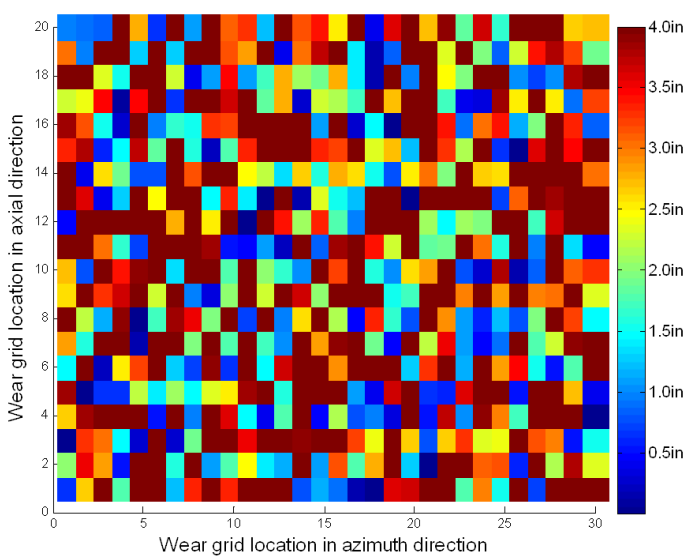


Figure 62: Randomly distributed refractory wear

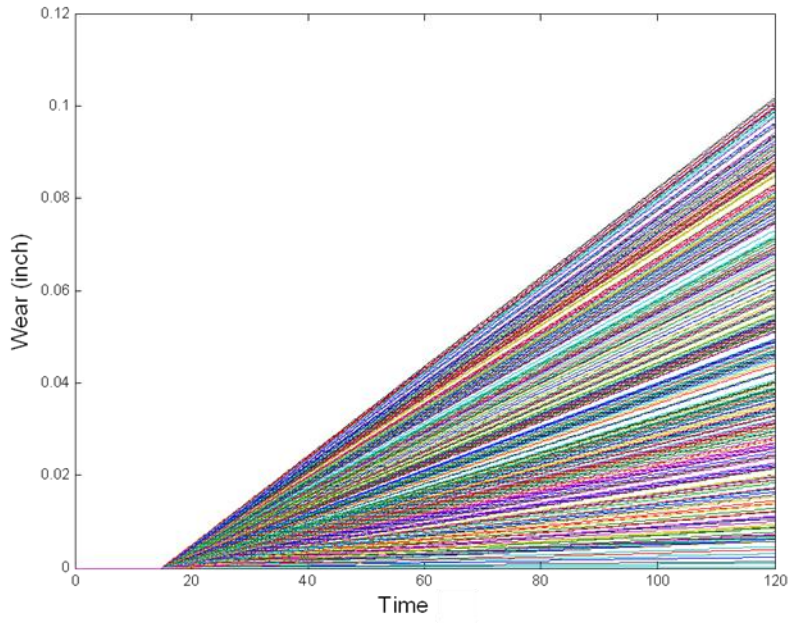


Figure 63: Refractory wear profile

The wear grows from zero level to corresponding maximum level in 120 hours as shown in Figure 63. The assumed wear ramp rate is certainly very large and unrealistic but the computing time needed to run the estimator for longer period imposed a practical constraint for carrying out this work. Hence a faster ramp rate was chosen. Nonetheless the fast ramp rate does not adversely affect the ability to assess the OSP solution.

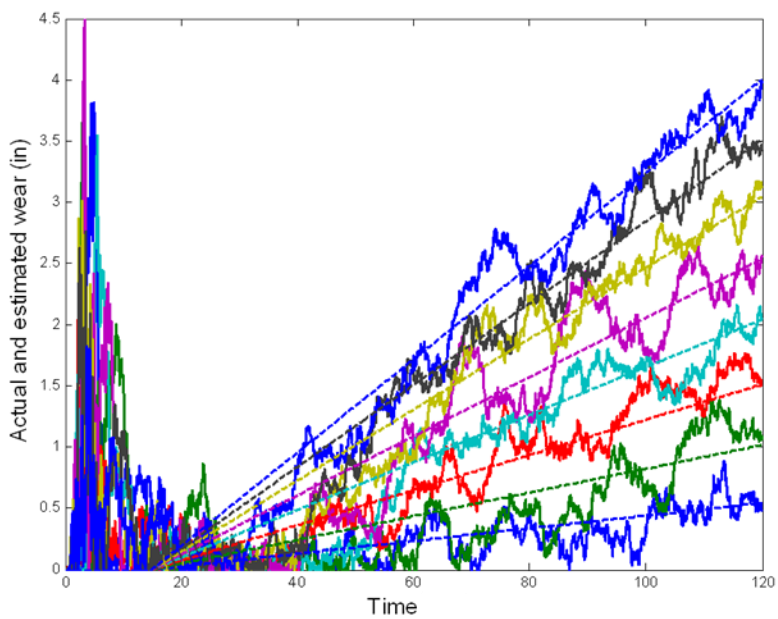


Figure 64: Wear estimate through model based sensing

The temperature responses within the gasifier refractory were generated with these randomly generated wear profile keeping all other input constant. The assumption here is that the time scale in which the input change occurs are too small compare to the time scale related to wear propagation (hours in the simulation). The hardware sensors included in Model based sensing approach were obtained using the solution to OSP problem for the gasifier refractory wear monitoring. Total 40 such randomly generated wear profiles were used for simulation validation study. Figure 64 shows the wear estimate through model based sensing for one typical case out of this 40 random cases. To avoid cluttering only eight wear grid points are shown in the figure. From the figure it can be seen that the model based sensor is able to correctly estimate the wear profile for all the wear grid location. The model based sensor requires some initial time based on sensor tuning but soon latches on to the actual wear profile. Similar responses were observed in all other cases.

In the simulation environment since the actual wear profiles are available, the estimation error statistics can be computed directly. Figure 65 shows the distribution of mean error between the actual wear and the estimated wear obtained through the model based sensing. From the figure it is clear that the mean error is normally distributed with around mean zero and standard deviation much smaller than that of the design precision value as expected. This is so, because the optimizer ensures that the precision constraint is satisfied. Recall that the precision is output error variance which is a stochastic measure.

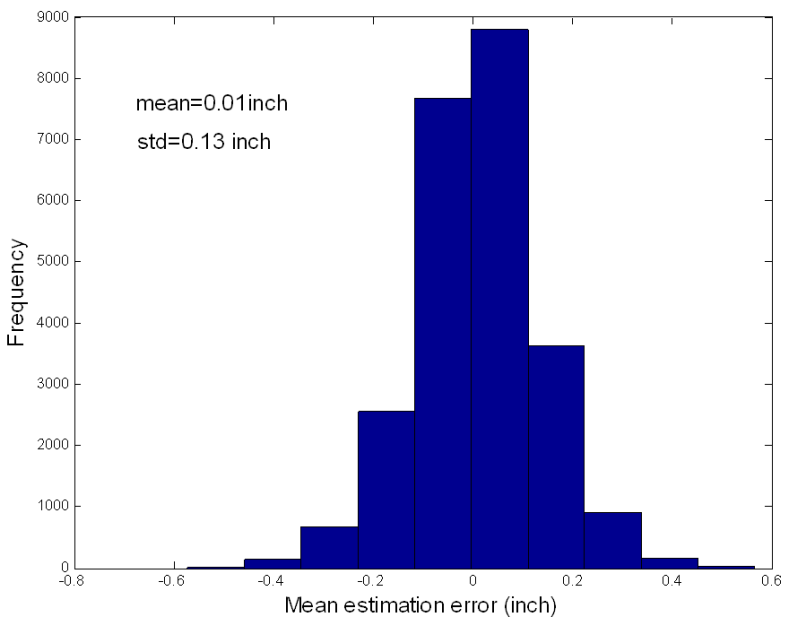


Figure 65: Distribution of the mean wear estimation error

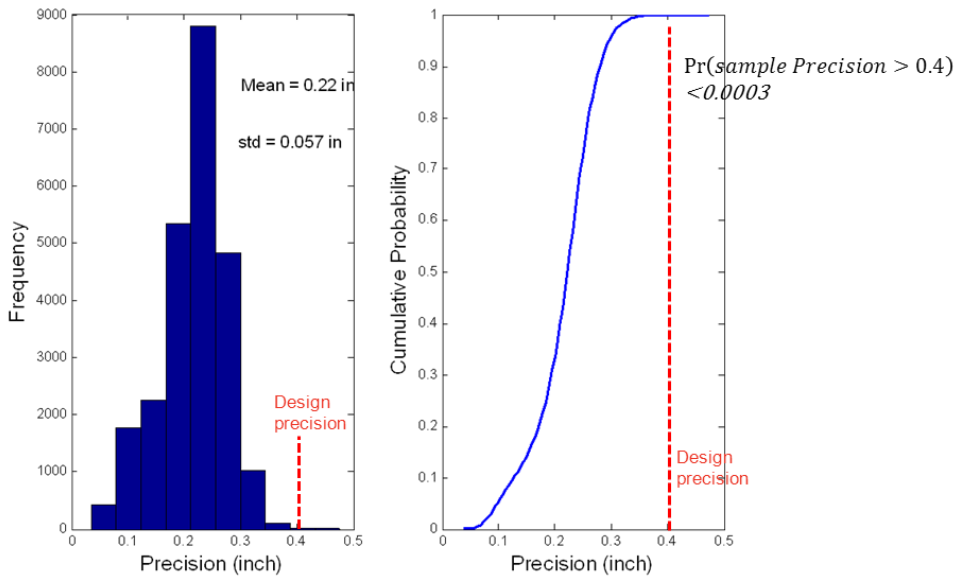


Figure 66: Precision constraint obtained from simulation based on 24600 wear estimates

Further, the individual variance of each estimated wears as obtained from model based sensing approach for all 40 cases was compared with the design value of the precision constraints. Figure 66 shows the comparison of the ensemble standard deviation with that of the precision constraint used in the OSP solution. It can be seen from the figure, that the precision as obtained from the ensemble standard deviation is less than the design precision constraints. Further, Only 7 out of 24600 wear estimates had variance slightly more than the design precision of 0.4 inch. This implies that the probability of having a less accurate estimate than what the sensor system was designed for; using the solution of the OSP algorithm is less that 0.03%. Based on simulation it can be concluded that using the solution to the OSP algorithm for the gasifier refractory wear the model based sensor can accurately estimate the wear.

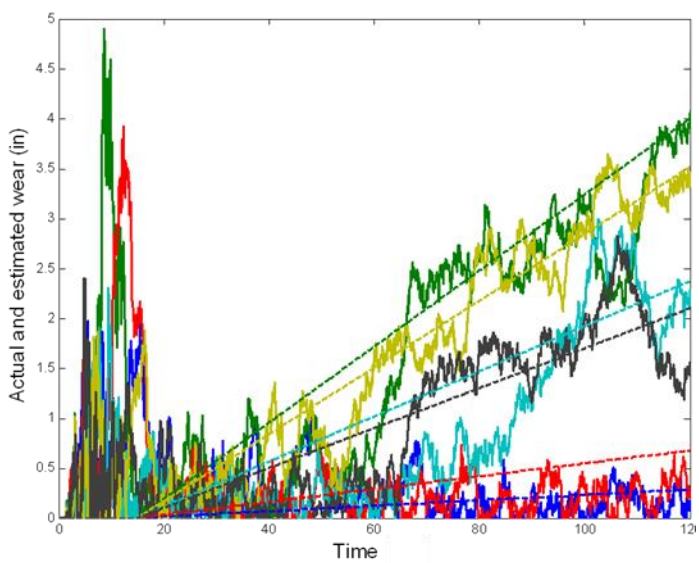


Figure 67: Wear estimate when few sensor from the solution to the OSP problem are moved further away

In order to check optimality of the solution one can remove few sensors from the solution to the OSP problem and do the simulation again. It was observed in the simulation that when a sensor is moved from the sensing location (1st interface) obtained as solution to the OSP problem to further away from the hot surface (2nd interface), the model based sensor was either not able to estimate the wear at the location behind which the sensor is moved from the first interface, correctly specially if the wear magnitude was small or was estimating the wear with reasonably large variance rending such estimate meaningless. Figure 67 shows one such case. In the figure, the wear of maximum magnitude of 0.5 inch could not be estimated correctly. Further the wear estimates have large variance.

6.3.2 Assessing solution to OSP problem for CM of RSC fouling profile

In order to assess solution to OSP problem for condition monitoring of RSC fouling a Monte Carlo simulation was conducted where randomly generated 20 RSC fouling profiles were used to simulate multiple cases of RSC tube fouling for simulation validation study. Recall that to model the RSC fouling profile the fouling was parameterized by six parameters ranging between zero and one. Parameter value being zero implies no fouling whereas one implies full fouling. It should be noted that the fouling distribution generated here is completely uncorrelated in the sense that the zone next to each other might have completely different fouling; one might have maximum fouling whereas the next one might be close to zero. In reality this is a highly unlikely scenario. Normally the nearby locations will have related fouling. Nonetheless the fouling distribution used here is very challenging for a model based sensor.

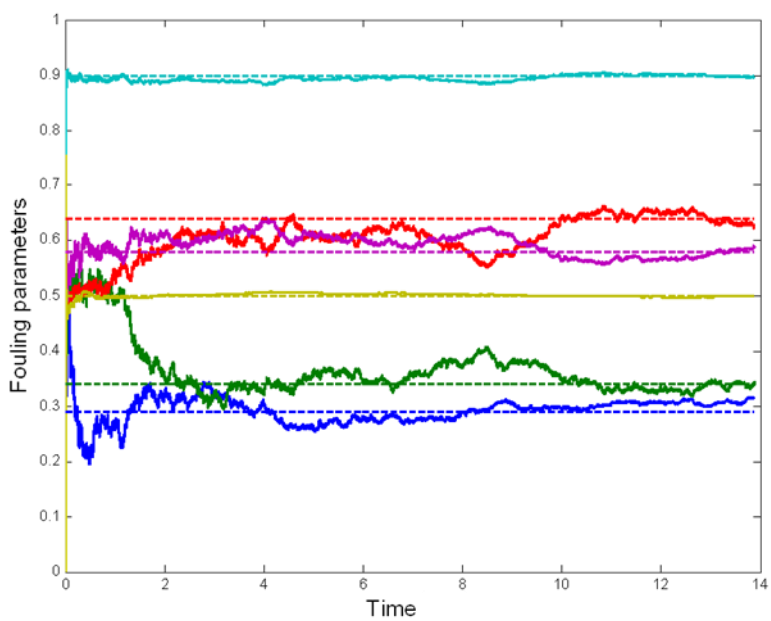


Figure 68: Fouling profiles and fouling parameters estimates

A Monte Carlo simulation was conducted with 20 randomly generated RSC fouling profiles. The plant response was generated with these randomly generated RSC fouling profiles keeping all other input constant. The assumption here is that the time scale in which the input change occurs are too small compare to the time scale related to fouling. Also, since

fouling occurs very slowly, it is practically constant for estimation modeling. Figure 68 compares the actual fouling parameters used in the simulation with that of the fouling estimates for one typical simulated study.

In the figure, the broken lines are the actual fouling parameters whereas the solid lines with same color are the corresponding estimates. The figures shows that the fouling parameter estimates takes around 10 hours to tune themselves and after that they match very well with the actual parameters.

In the simulation environment since the actual fouling parameters are known, the estimation error statistics can be computed directly. Figure 69 shows the distribution of mean error between the actual fouling parameters and the estimated fouling parameters obtained through the model based sensing. From the figure it is clear that the mean error is normally distributed with around mean zero and standard deviation much smaller than the design precision value as expected. This is so, because the optimizer ensures that the precision constraint is satisfied. Recall that the precision is output error variance which is a stochastic measure. In this case around 9% case the ensemble variance was more than the design variance. This is slightly more than expected but most likely this is due to slow convergence of the estimate.

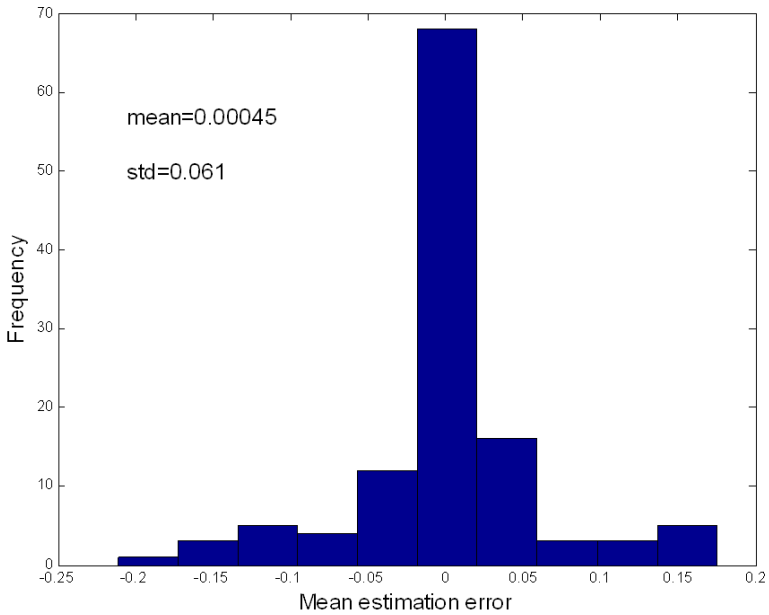


Figure 69: Distribution of mean estimation error for RSC fouling parameters

7 Conclusion

An Integrated Gasification Combined Cycle (IGCC) plant is a large chemical plant coupled to a power generation plant. It is traditionally designed to operate mainly at steady state conditions. The IGCC technology is very important for clean and efficient power generation from coal and is currently being commercialized by GE. One of the key challenges in its commercialization is to increase reliability and availability of an IGCC plant. The critical

components that normally affect the plant availability are gasifier and radiant syngas cooler (RSC) in the core gasification section. These components operate in a corrosive environment at very high temperature (1200-1600 ^{o}C) and pressures (exceeding 40atm) leading to significant degradation in their condition. In an IGCC plant the gasifier is normally checked annually for wear and damage. In the absence of online monitoring system, plant is operated conservatively and is required to shut and cool down for manual inspection and maintenance of the gasifier refractory lining. Each gasifier cool down for manual inspection and the refractory preheat for startup requires roughly 10 days of plant outage. These are periodic outages that the plant must take in order to perform routine maintenance. These outages are often scheduled during a time of year that the customer suffers the least loss of income (for instance, in Florida, the best time for an outage of a power plant would be between high electrical usage for heating and air conditioning like February or March time frame). In that case, shutting down in December or June to replace refractory would be very costly. In addition, there are unscheduled outages too. Some of these unscheduled outages happen if it is felt that the refractory was wearing faster than it should be based on the other indirect indicators. Having no other means of condition monitoring, the plant may be forced to shut down for a week to allow for cool down and inspection of the refractory. This adversely impact plant availability. Similarly, the heat transfer performance in the RSC degrades over time due to fouling buildup from ash and slag, which also affects the plant efficiency. Online condition monitoring is critical to enhancing power generation plant reliability and availability. Furthermore, an online condition monitoring will also enable increased plant efficiency and flexibility when coupled with advanced control strategies such as model-predictive control. Motivated by this, a systematic approach to sensor network design methodology has been developed in this program that supports the development of an online condition monitoring. In particular, this project developed a sensor network design with optimal combination and placement of sensors to enable online monitoring of refractory condition and RSC fouling in order to improve IGCC plant availability and efficiency. The approach is model based and uses Extended Kalman Filter framework along with optimization to optimally place sensor in order to meet certain network criteria e.g., precision and reliability. There were three main tasks in this program. The results and conclusions from each key element of the program are summarized in the next sections.

7.1 Model Enhancement and Technology Review

Inability to get direct measurement due to either unavailability or high cost of hardware sensors pertaining to harsh environment in the core gasification section necessitated indirect or model based sensing approach for condition monitoring of gasifier refractory wear and RSC tube fouling. The overall objective of an optimal sensor placement problem was to design a cost-effective sensor network for a robust monitoring of key process variables in the presence of practical constraints e.g., sensor measurement errors as well as modeling errors, in addition to expected sensor failure rates.

The three main elements to an OSP algorithm were identified. They are: network requirement specifications, an estimator for indirectly “measuring” the quantities of interest based on a model and some direct measurements and an optimizer to optimize the

number of direct measurements needed for the estimator to meet the given network requirement specification.

Two key requirements that the sensor network should satisfy for condition monitoring of a system were identified as, a) the precision requirement that deals with the accuracy of the estimated signal in terms of mean and standard deviation and b) reliability of the network that deals with the accuracy of the estimated signal in the presence of expected failure rate of hardware sensors that make up this sensor network. In addition the physical locations where a possible hardware sensor may be placed are assumed to be given.

The estimator within the OSP framework was chosen to be of Kalman filter type. This was done to systematically account for modeling and sensor error. The network requirement of precision was mapped into parameters related to steady state covariance matrix in the Kalman filter formulation. Multiple notion of network reliability was discussed where he functional reliability was found to be most appropriate for OSP problem formulation with model based sensing approach. Even though enforcing estimation reliability would lead to minimum cost sensor network, the computation complexity of enforcing this constraints was found to be prohibitively high.

With Kalman filter approach for model based estimator the problem of optimal sensor placement was naturally posed as an optimization problem with objective being minimizing sensor network cost subject to precision, reliability and sensor location constraints. The optimizer optimizes the number, type and location of the hardware sensors used within the Kalman filter. This optimization problem is a nonlinear programing problem over integer variable (INLP), the integer decision variable being the zero or one corresponding to “use” or “don’t use” a particular sensor at a given location. Due to the combinatorial nature of INLP problem, the OSP problem is computationally challenging to solve efficiently. An exhaustive literature survey was conducted to shortlist algorithms to solve INLP problem related to OSP formulation. Two algorithms, one based on branch and bound approach and other based on outer approximation techniques were found most suited for solving an OSP problem. These algorithms were shortlisted for further evaluation in the Task3. A slightly different formulation to solve the OSP problem based on Chmielewski et al (2002) work was also shortlisted for evaluation in Task3. This formulation is based on LMI.

Model based sensing requires mathematical model of the process. First-principles physics based models captures the process knowledge through mass and energy balance and is used to relate the measured signals with variable of interests that cannot be measured directly. The dynamic model of the core gasification system developed in a previous DOE funded program (DE-FC26-07NT43094) was extended to make them suitable to use in the OSP formulation. In this program, the key focus was to develop a sensor network with optimal combination and placement of sensors to enable online monitoring of refractory condition and RSC fouling. In this regard the existing 1-D axial gasifier refractory model was extended to model the 3-D temperature variation in the gasifier refractory due to refractory wear. The model was discretized to solve it numerically. The discretization resulted in a very large size gasifier model. Similarly, the RSC model from the previous program (DE-FC26-07NT43094) was enhanced to include axial fouling profile for the RSC tubes. The fouling profile was parameterized by six parameters. These parameters scale

the conventional as well as radiation heat transfer coefficient appropriately to model fouling effect on the heat transfer from syngas to the water/steam with the tube. In additional the strain on the tube supporting structure was included in the model. This bolt strain is a good measure of cumulative fouling on the tubes.

7.2 Development of OSP Algorithm

For development purpose, there are two components of the OSP algorithm- the estimator and the optimizer. These have been integrated under OSP framework. All the algorithms were implemented in Matlab©. A simplified first order gasifier model was used to evaluate various OSP formulations. Even though OSP problem is an offline design problem, an efficient optimizer is needed to find solution to the OSP problem with reasonable time and solution accuracy. Hence the OSP algorithms were evaluated for these two criteria.

All the solvers to OSP algorithms give accurate solution, hence the applicability of these algorithms for practical application basically depend on their ability to solve the OSP problem in reasonable time and memory resources. The LMI based OSP formulation was found to suitable only for a small size problem (problem with less than 20 states). The formulation does not scale up well with the size of the OSP problem. It became computationally inefficient in terms of both, CPU time and memory requirement. Of course the efficiency of the algorithm depends on the solver used to solve the LMI problem. Matlab toolbox based LMI solver, LMI Lab and widely used and freely available solver SDPT3 were evaluated. For a small size problem the Matlab solver was found to be more efficient but its performance degraded with the increase in the size of the problem. However both the solvers do not scale well in terms of memory requirement with the size of the problem.

The standard OSP problem formulation as an INLP problem was evaluated using branch and bound solver as well as OA solvers. A key element of the branch and bound algorithm is to use appropriate relaxation of the original problem to generate lower bound of the optimum solution at each node. In order to solve the OSP problem, integer relaxation was used. The relaxed problem with precision only constraint is a nonlinear convex problem in real decision space with particular scaling of measurement noise covariance matrix. A state-of-art NLP solver IPOPT was used to solve the integer relaxed problem. Branch and bound is a systematic enumeration process. The reliability constraint being non convex cannot be accounted directly in the standard branch and bound algorithm. Hence it was modified to incorporate functional reliability. Branch and bound algorithm is efficient algorithm for solving a small to medium size INLP problem. However, due to enumeration process, this algorithm does not scale well with the size of the problem. This algorithm can still be used to solve small decision space of the order of 50-60 quite efficiently.

The OA based approach uses linear relaxation of the INLP problem and solve relaxed integer linear programming (ILP) problem to obtain the lower bound on the solution and then repeatedly improves the bound. State of art GLPK software was used to solve ILP problem whereas IPOPT was used for solving the NLP problem in OA approach. The standard OA approach deals with mixed integer nonlinear programming problem hence cannot be directly applied to the OSP problem which has only integer decision variables.

The OSP formulation was slightly modified to include real slack variables to make it mixed integer problem. However, the standard MINLP solvers were not found very efficient to solve this problem. In addition, similar to branch and bound, due to non-convexity and non-smoothness of the reliability constraint the OA framework for solving the OSP problem was modified. Further, in order to solve the OSP problem efficiently the standard OA approach was modified to incorporate cutting planes. A technical paper based on this approach was presented in peer reviewed IEEE Conference on Decision and Control (CDC) 2012 conference.

The computational efficiency as well as accuracy of a gradient based NLP solver can be significantly improved if the solver is provided with the analytical derivatives of the cost and the constraints functions. The cost function is linear in decision variable however the precision constraint which is directly related to the solution to an algebraic Riccati equation is a nonlinear function of the decision variable. It was shown that derivative of this solution actually leads to a Lyapunov function and can easily be computed suing Matlab *lyap* command. Use of efficient solvers along with providing solvers with analytical derivatives led to an order of magnitude improvement in the computational efficiency of the algorithm.

Based on various OSP formulation and use of various optimizers to solve the OSP problem it was concluded each formulation and solver has its merit and demerit. The LMI based formulation is easy to follow but does not scale well with the size of the problem. The branch and bound is widely used solver for solving INLP problem but again it is suited well for only small to medium size problem. OA based algorithms are most powerful for large size problem but are computationally more complex. Hence in this program, instead of having one OSP algorithm, a suite of OSP algorithm was developed that make all these various formulation and as well optimizer options available to the user. This modular OSP tool along with modular estimator block will be available to DOE.

7.3 Demonstration of OSP Algorithm Performance

The OSP algorithms developed in Task 3 were applied to design optimal sensor network for condition monitoring of the gasifier refractory wear and RSC fouling profile. The gasifier model and the gasification section model (gasifier, RSC, quench, drum, scrubber, etc.) were used for this purpose.

The gasifier model has more than 9000 states and 3000 possible sensor location for temperature measurement. Due to the size of the gasifier model, The OA based OSP algorithm was used for finding the optimal sensor set. Even with OA based algorithm, it was not possible to apply the OSP algorithm to the full 3-D gasifier model because of the computational limitation of the machine. Exploiting the symmetry of the model about the axial direction, an “equivalent” but smaller and manageable size wedge block was identified and used for applying the OA-based OSP algorithm. Once the optimal sensor location for this wedge block was obtained, the solution for the complete 3-D gasifier was obtained by just repeating the small wedge block optimal solution. In order to estimate refractory wear within 0.4 inch accuracy one would need as many as 600 sensors, all placed in the first interface of the refractory layer. These sensor locations obtained as

solution to the gasifier refractory wear OSP problem are due to the dominance of heat flow in the radial direction.

The temperature sensor locations obtained above was used in the model based sensing approach to verify if indeed it is possible to obtain the refractory wear estimates using these sensor location within the design accuracy of 0.4inch. A Monte Carlo simulation was set up with more than 40 completely random distributed wear profiles on the hot surface of the gasifier refractory. For the refractory hot surface there are 600 grid locations where the wear was monitored. The ensemble variance based on this sample size of more than 24000 wear estimates showed most of the estimate variance were within design limit and the probability that a wear estimate is less than design accuracy was less than 0.03% . Further, to show that the solution thus obtained for the condition monitoring the refractory wear OSP problem is optimal, few thermal sensors were moved further away from the intended location. With this modified sensor locations, the simulation with same random wear profiles, showed that the sensor network was not able to estimate the refractory wear at certain axial and azimuthal locations that corresponded to radial directions from where the sensors were moved. This showed that the OA based OSP algorithm is able to provide correct solution. The specific optimal sensor locations obtained through solving the gasifier refractory wear OSP problem are as such not very important. Due to strong radial heat flux dominance one could have obtained the same solution based on a pure engineering judgment. But that is just due to the nature of this particular problem and it is not always possible. The algorithms to solve OSP problem developed in the current program are generic and can be applied to other applications.

The branch and bound based OSP algorithm was applied to obtain the solution to optimal sensor placement problem for condition monitoring of the RSC tube fouling. It was assumed that the normal sensor set that are usually present and used for controls of the gasifier operation will remain available. The OSP algorithm was used to locate additional heat flux and strain gauge sensors to estimate six fouling parameters that are used to characterized the fouling profile. With this setup the gasification section model has close to 200 states and 31 possible additional sensor locations. The solution to this OSP problem shows that by placing 2 additional heat flux sensor (out of possible 30) and using strain gauge measurement on the required precision on the estimated fouling profile can be obtained. Again, to verify this solution these sensors were used in the model based sensing approach to estimate the fouling profile parameters using a Monte Carlo simulation. In the simulation the model based sensing system was used to estimate fouling profile parameters for 10 randomly generated fouling profile. The simulation shows that in all cases the sensing system was able to estimate all six fouling parameters within the design accuracy. This showed that the branch and bound based OSP algorithm performs as expected and can be used for finding solution to OSP problems of moderate sizes.

8 References

1. E. Balas, S. Ceria and G. Cornu´ejols, “A lift-and-project cutting plane algorithm for mixed 0-1 programs”, Mathematical Programming, Vol. 58 (1993), p. 295–324
2. C. Benqlilou, M. Graells, E. Musulin and L. Puigjaner, “Design and Retrofit of Reliable Sensor Networks,” Ind. Eng. Chem. Res., p 8026-8036, 2004.
3. P.Bonami, L.T.Biegler, A.R.Conn, G. Cornuejols, I.E.Grossman, C.D. Laird, J.Lee, A.Lodi, F.Margot, N.Sawaya, A.Wachter, “An algorithmic framework for convex mixed integer nonlinear programs”, Discrete Optimization, pp. 186-204, 2005.
4. P.Bonami, G. Cornuejols, A. Lodi, F. Margot, “A Feasibility Pump for Mixed Integer Nonlinear Programs,” Mathematical Programming, p.331-352, 2006
5. P.Bonami, M. Kilinc, J.Linderoth, “Algorithms and software for convex mixed integer nonlinear programs,” Technical Report 1664, Computer Science Department, University of Wisconsin Madison, 2009
6. S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, “Linear Matrix Inequalities in Systems and Control Theory,” SIAM, 1994.
7. Samuel Burer, Renato D.C. Monteiro, Yin Zhang , “Rank-Two Relaxation Heuristics for Max-Cut and Other Binary Quadratic Programs”, SIAM Journal on Optimization, Vol12, 2001
8. J.W.Chinneck, “Practical Optimization: A Gentle Introduction,” www.sce.carleton.ca/faculty/chinneck/po.html, 2010
9. D. J. Chmielewski, T. Palmer, and V. Manousiouthakis, “On the theory of optimal sensor placement,” AIChE J., p 1001-1012, 2002.
10. M.A.Duran, I.E.Grossman, “An outer approximation algorithm for a class of mixed-integer nonlinear programs,” Mathematical Programming, Vol. 36, p. 307-339, 1986.
11. M. L. Fisher, “The Lagrangian Relaxation Method for Solving Integer Programming Problems,” Management Science, v 27, p 1-18, 1981.
12. M. L. Fisher, “An Applications Oriented Guide to Lagrangian Relaxation,” Interfaces, v 15, p 10-21, 1985.
13. C.A. Floudas, “Nonlinear and Mixed-Integer Optimization – Fundamentals and Application,” Oxford University Press, 1995.
14. A. M. Geoffrion, “Lagrangean relaxation for integer programming,” Mathematical Programming Study, v 2, p. 82-114, 1974.
15. A.M.Geoffrion, “Generalized Benders Decomposition,” Journal of Optimization Theory and Applications, Vol 10., p237-260, 1974.
16. M.X. Goemans and D.P. Williamson, “Improved approximation algorithms for maximum cuts and satisfiability problems using semidefinite programming”, Journal of the Association for Computing Machinery, Vol.42, p1115–1145, 1995.
17. I.E.Grossman, Z.Kravanja, “Mixed-integer Nonlinear Programming Techniques for process systems engineering,” Computer Chemical Engineering, Vol 1, p 189-204 1995.
18. M. Held and R.M. Karp, “The traveling salesman problem and minimum spanning trees,” Ops. Res., v 18, p 1138-1162, 1970.

19. M. Held and R.M. Karp, “The traveling salesman problem and minimum spanning trees: Part II,” Math. Prog., v 1, p 6-25, 1971.
20. M. Kojima and L. Tuncel, ``Cones of Matrices and Successive Convex Relaxations of Nonconvex Sets" , *SIAM Journal on Optimization* Vol.10, No. 3, p750-778, 2000.
21. M. Kojima, "Semidefifnite programming relaxation and Lagranginan relaxation for polynomial optimization problems, " The Third International Conference on Nonlinear Analysis and Convex Analysis, August 25 - 29, 2003, Tokyo Institute of Technology, Tokyo, Japan. Invited talk. http://www.is.titech.ac.jp/~kojima/talk.html
22. Jean B. Lasserre. “An Explicit Exact SDP Relaxation for Nonlinear 0-1 Programs”. *In Proceedings of IPCO'2001*, p 293~303
23. M. Laurent, “A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming”, Mathematics of Operations Research, Vol. 28, p470-496,2001
24. M. Laurent and F. Rendl, “Semidefinite Programming and Integer Programming”, http://homepages.cwi.nl/~monique/files/chapter1.ps, December 2002.
25. L. Lov´asz and A. Schrijver, “Cones of matrices and set-functions and 0-1 optimization”, SIAM Journal on Optimization, Vol. 1 p166–190, 1991.
26. M. Mevissen and M. Kojima, "SDP Relaxations for Quadratic Optimization Problems Derived from Polynomial Optimization Problems", *Asia-Pacific Journal of Operational Research* Vol 27 (1) p15-38, 2010.
27. G.L.Nemhauser, L.A.Wolsey, “Integer and Combinatorial Optimization,” Wiley, 1988.
28. H. Sherali and W. Adams, “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems”, SIAM Journal on Discrete Mathematics, Vol. 3: p 411–430, 1990.
29. G. Seenumani, D. Dai, R. Lopez-Negrete, A. Kumar, M. Dokucu, R. Kumar, “An Outer-Approximation Based Algorithm for Solving Integer Non-linear Programming Problems for Optimal Sensor Placement”, IEEE-CDC Conference Proceedings, p 4455-4461, 2012.

Appendix A: Modular Tools

As part of this program two modular tools were developed which can also be used for other applications. One of the tool is Matlab/Simulink© based extended Kalman Filter block and the other is the Matlab© based optimal sensor placement tool. In this section both these tools are described briefly.

A.1 Modular Extended Kalman Filter block in Matlab/Simulink©

The EKF block is designed to implement an Extended Kalman Filter (EKF) state estimator in the form of a compilable Simulink block which can be placed in higher level Simulink models such as plant system simulations or control algorithm models, and from which estimator code can be generated for real-time operation on the controller. The modular form of the EKF Simulink block allows the state estimator to be automatically constructed around two user defined models, a plant model and an auxiliary state model that are placed inside modular blocks with pre-specified interfaces. The user defined models can be Simulink or embedded Matlab blocks with custom code, as long as they can be compiled. The top level EKF block, identifying the input/output structure, is shown in Figure 70.

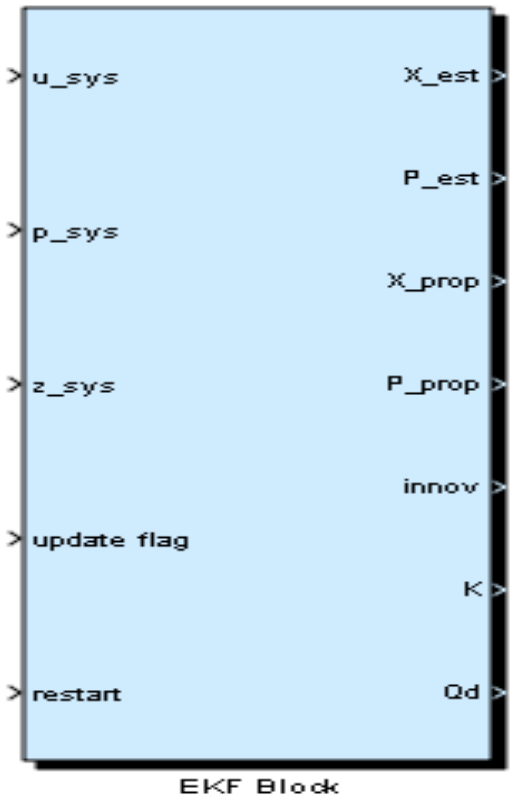


Figure 70: EKF Simulink - top level block

A.1.1 Input and Output Signals

A summary of the Inputs is contained in Table 14

Input	Description	Size (at each time step)	Comments
u_sys	Measured inputs which drive the system	[# of measured inputs x 1]	Set points, operating conditions and other time varying signals that drive system
p_sys	Assumed system parameter values	[# of parameters x1]	Constants (of particular interest) that affect system response
z_sys	Measured outputs of the system	[# of measured outputs x 1]	System output sensor measurements
update flag	Flag to enable an EKF state update with current measurements	scalar	Allows the measurement update to occur at a slower rate than the basic execution rate (if desired)
restart	Flag to reset the EKF estimator	scalar	Allows re-initialization of EKF estimator while running

Table 14: EKF Block Inputs

A summary of the Outputs is contained and Table 15.

Output	Description	Size (at each time step)	Comments
X_est	State Estimate	[# of estimated states x 1]	EKF State estimate is the primary output
P_est	State Error Covariance	[# of est states x # of est states]	For performance monitoring, information or debug
X_prop	State Estimate prior to measurement update	[# of estimated states x 1]	For performance monitoring, information or debug
P_prop	State Error Covariance prior to measurement update	[# of est states x # of est states]	For performance monitoring, information or debug
innov	Measurements minus model predicted measurements	[# of measurements x 1]	For performance monitoring, information or debug
K	Kalman gain matrix	[# of est states x # of meas]	For performance monitoring, information or debug
Qd	Contribution of modelled process noise to State Error Covariance	[# of est states x # of est states]	For performance monitoring, information or debug

Table 15: EKF Block Outputs

The EKF block executes at a fixed time step as specified by the user, and the external system data is input at this fixed rate. The input data signals are connected to the *u_sys*, *p_sys* and *z_sys* ports, where *u_sys* represents the measured inputs driving the plant, *p_sys* represents assumed parameter constants of the plant, and *z_sys* represents the measured outputs of the plant. The distinction between inputs and parameters is arbitrary and is made for organizational convenience only, as either port can contain both constants or time varying signals and in fact all input type signals could be fed in through just one of these ports. The *update_flag* port is used to indicate that the *z_sys* measurements are fresh and should be used by the EKF block to update the state estimates using these measurements. The *restart* port provides the capability to reset the EKF state estimator to the predefined initial conditions and restart the estimation at any point after it is running. The primary output of the EKF block is the state estimate vector, *X_est*., at each time step. *X_est* contains all the plant model states as well as any user defined auxiliary states. The remaining outputs are provided mainly for information or performance monitoring, if desired. *P_est* is the state error covariance matrix at each time step as computed by the EKF process. *X_prop* and *P_prop* are the propagated state vector and state error covariance matrix at each time step, prior to being updated with the measurements, *z_sys*. The innovation vector, *innov*, represents the difference between the actual measurements and their predicted values obtained from the EKF model evaluated at *X_prop*, while *K* is the Kalman Filter Gain matrix which multiplies the innovations to produce the state vector update corrections. Finally, *Qd* is the contribution of the modeled process noise to the state error covariance. (Note that any additional internal signals that are of interest may be easily obtained by simply opening the EKF block in Simulink, adding output ports, and connecting them to the desired signals.)

A.1.2 User Supplied Models

As mentioned, in order to adapt the generic EKF block to a particular application, the user supplies a plant model and an auxiliary state model, which are placed inside standard interface placeholder blocks which exist for that purpose. In addition, some specific information about these models, such as the number of states, inputs, parameters and how they should be combined in the estimator must be input into an initialization file. The plant model captures the essential dynamics and output relationships of the system being observed or controlled, while the auxiliary model is the means to incorporate additional estimator states such as sensor biases, random process models, parameter estimates, etc. that are typically a part of robust system state estimation.

The plant and auxiliary models must adhere to the standard interface as shown below:

$$[\vec{y}_i, \dot{\vec{x}}_i, \vec{x}_{i+1}] = mdl_fnc\{\vec{x}_i, u_i, p_i, w_i\},$$

Where, the model acts as a function that returns the outputs, state derivatives, and one-step-ahead state extrapolation when given state, input, parameter, and noise input values. Note that this function is not in the form of a typical ‘time simulation’, in which the states are computed and maintained internally; rather it requires all state information to be passed into it, so the user supplied models likewise should not contain internal state

memory themselves. Placeholder blocks with the proper interface ports are provided for the models as is shown by the two highlighted model blocks in Figure 71 (this figure represents a several level drill down from the top level EKF block of Figure 70).

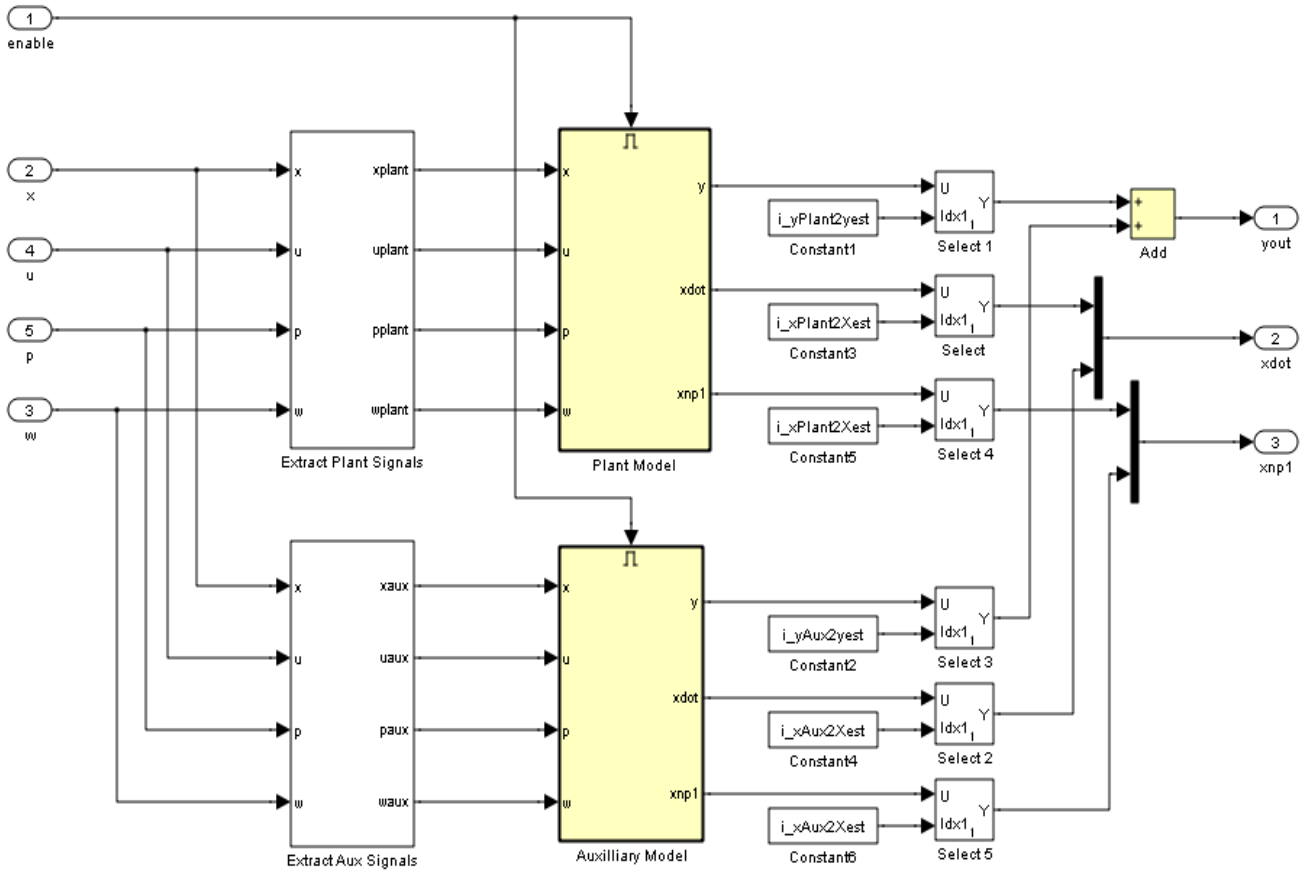
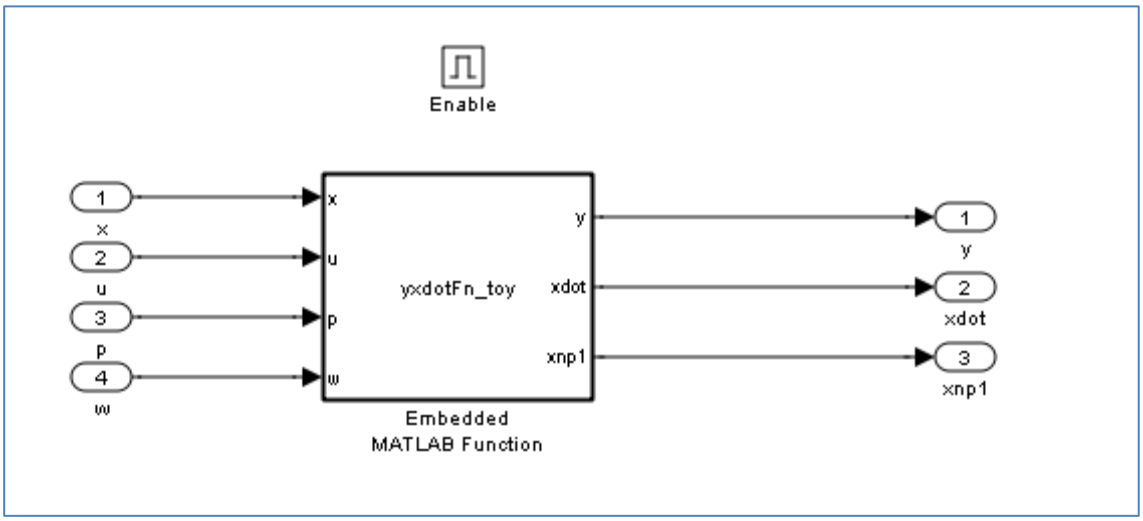


Figure 71: Standard Interface Blocks for User Defined Models

The model placed inside the generic block can be in the form of a Simulink block diagram model or it can be an embedded Matlab block containing the code producing the model function. In either case the result is a Simulink block with the prescribed inputs and outputs, which is simply connected inside to the external ports provided by the generic block. The supplied models must be designed to execute at a fixed time step, specified by the user to accommodate the plant dynamics, and this time step also becomes the top level execution time step. Measurements can be processed by the EKF estimator at a lower sampling rate using the *update flag* signal previously described.

To illustrate, a simple example follows which uses an embedded Matlab block for the plant model, and a Simulink block diagram for the auxiliary model. Figure 72 shows the plant model being ultimately defined by code in an embedded Matlab function block.




```
function
[y,xdot,xnp1]=yxdotFn_toy(x,u,p,w,prop_dT)
%#eml

% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%  Embedded Matlab Code Example
%  Plant as a damped second order system
% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Zeta = p(1);
Wn = p(2);

A = [0 1 ; -Wn^2 -2*Zeta*Wn];
B = [0;Wn^2];
G = [1 0; 0 1];

xdot = A*x + B*u + G*w ;
xnp1 = x + xdot*prop_dT;
y = x(1);
```

Figure 72: User Defined Plant Model as an Embedded Matlab Block

An empty embedded Matlab function block is selected from the Simulink library which, when opened, brings up the embedded Matlab editor into which the desired model code can be entered, as shown. The function definition line identifies the required standard inputs and outputs and these will appear as the ports on the block. In this example, an additional term, *prop_dT* representing the sampling time step, has been included in the call list but does not show up as a block input because it has been declared in the editor (tools/edit data menu) as a (Simulink) parameter rather than a block input. This

mechanism is useful for passing in model or system constants from the workspace without impacting the standard interface definition.

Figure 73 shows the auxiliary model, which is implements a first order Markov random process directly in a fixed time step Simulink diagram.

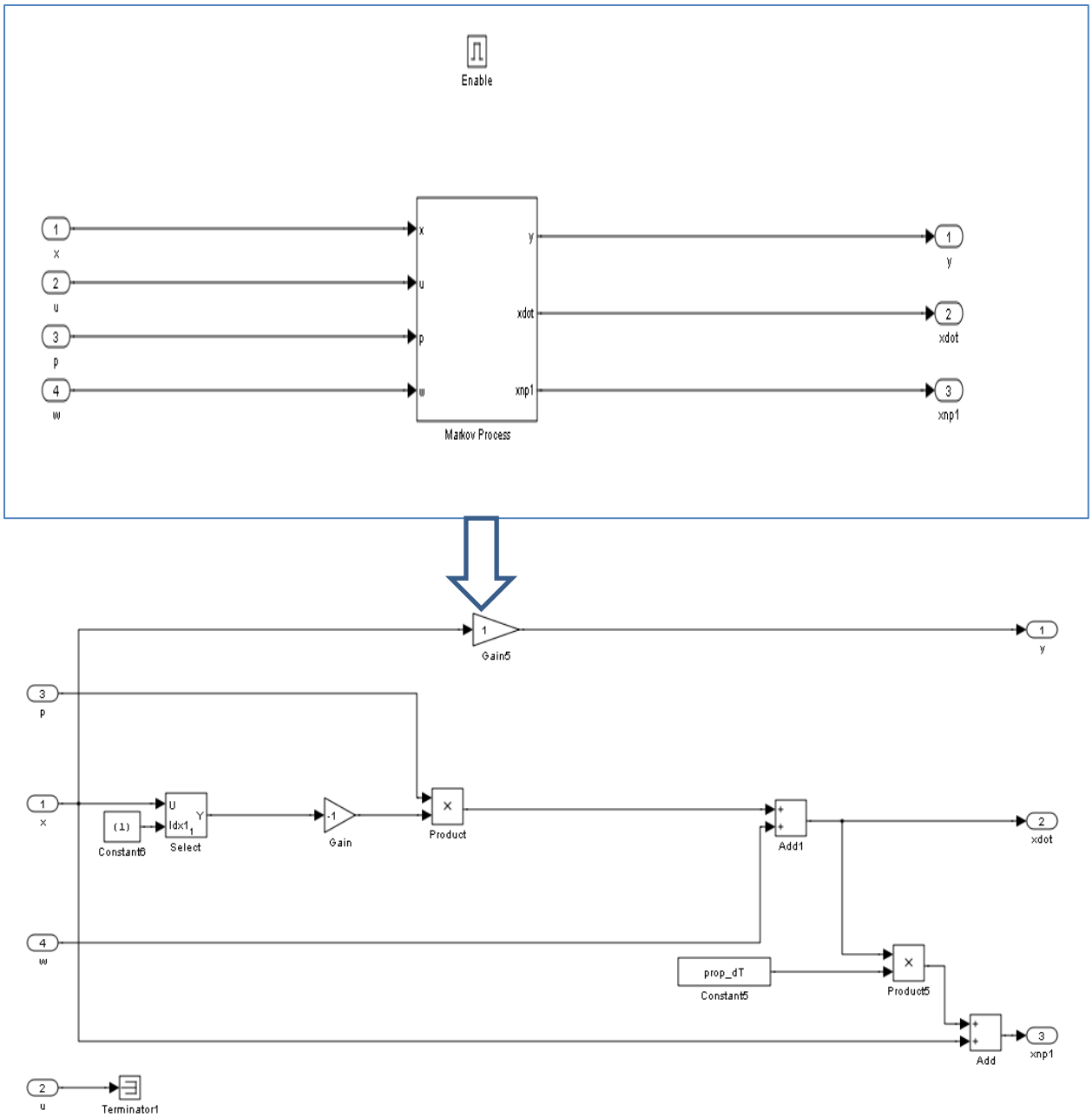


Figure 73: User Defined Auxiliary Model as a Simulink Diagram

A.1.3 EKF Estimator Model

The EKF estimator model is a composite of the plant model and auxiliary model, in which the EKF state vector, *x*, is defined as the plant states followed by the auxiliary states. Likewise, the EKF input, *u*, parameter, *p*, and noise driver vector, *w*, definitions are also the stacked version of the plant and auxiliary model signal names. (This stacked ordering of the two model groups must be adhered to, but within each group, the signals can be reordered, if desired, by the user through the model connection definitions specified in the initialization file.) The EKF measurement vector, *y*, definition, however, will be very specific to each application, as it typically involves measurements which are modeled as functional combinations of plant and auxiliary outputs. For this reason, it was deemed most efficient to require the user to simply construct this functional combination in the Simulink diagram itself, rather than try to make it generically configured from setting in the initialization file. In Figure 71, the small highlighted block in the upper right corner (an 'addition' block) represents this required user construction, which for this example models the single EKF output as the sum of a plant model state (the desired quantity) and an auxiliary model state (a corrupting sensor bias).

The initialization file entries for the simple example are shown below. For presentation purposes, the file sections have been divided into several Tables. Sections 1 through 4 are shown in Table 16, where the user specifies the basic EKF block computational time step, *prop_dT*, section 1 (in units consistent with the user supplied models); specifies the Plant and Auxiliary model dimensions (in terms of the *x*, *u*, *p*, and w vector sizes) in sections 2 and 3; and specifies the size of the desired EKF measurement vector, *y*, and particulars about which noise driver signals contained in *w* should be active (using 1's and 0's in *w2use* to indicate signal vector elements that are to be used or not used, respectively), in section 4. The ability to include many potential noise drivers in the models, but use only subsets of these for specific implementations provides a flexibility and efficiency in that different random process models and process noises can be set up without changing the models and recompiling each time, while the computational burden of model linearization at the unused noise inputs is eliminated.

```
%         Initialization File

% 1.  Specify Simulink Model Fixed Time Step Value
prop_dT = .02;

% 2.  Specify Plant model Info
% Number of Plant states, inputs, params, noise inputs:
%  (Set size to 0, if no signals of that type.)
nx_plant = 2;
nu_plant = 1;
np_plant = 2;
nw_plant = 2;

% 3.  Specify Aux model Info
% Number of Aux states, inputs, params, noise inputs:
%  (Set size to 0, if no signals of that type.)
nx_aux = 1;
nu_aux = 0;
np_aux = 1;
nw_aux = 1;

% 4.  EKF model Info
% Number of Estimator States
nx_ekf = nx_plant + nx_aux;
% Number of deterministic Inputs
nu_ekf = nu_plant + nu_aux;
% Number of parameter constants
np_ekf = np_plant + np_aux;
% Number of noise Inputs
nw_ekf = nw_plant + nw_aux;
% Specify w2use to identify which elements to include in linear model
w2use = [1; 1; 1]; % set to one element if nw_ekf = 0
% Specify mean value of noise signals to linearize about
wnom = zeros(max(nw_ekf,1),1);   % set to one element if nw_ekf = 0
% Number of measurements used
ny_ekf = 1;
```

Table 16: Initialization File Setting for Simple System Example

Table 17 shows the user specifications for the EKF settings and initial conditions. Beyond specifying the standard Kalman Filter quantities such as the initial state vector, *Xest0*, the initial error covariance matrix, *Pest0*, the noise power covariance, *Qc*, and the measurement noise covariance, *R*, the user can customize the perturbation step size selection for the state and noise input model linearization. This is accomplished through *xpertvec_in* for the states, and *wpertvec_in* for the noise drivers, where the rows correspond to each state or noise input, and the four columns identify the factors used to compute the perturbation step sizes. The function of these factors in step size computation is shown in .Table 18

```
% 5.  EKF Settings and Initial Conditions
% Initial State Estimates
Xest0 = [-1; 3; 0];
% Initial State Error Covariance
Pest0 = diag([10^2*ones(2,1); 1^2]);
% Noise Input Signal Power Density
Qc = diag([ .01^2; .01^2; .01^2]);
% Measurement Noise Covariance
R = diag(.01^2);
% Specify the Numerical Differentiation step sizes and limits for x and w
%  Rows correspond to states or noise vector elements
%  Columns (4) specify:
%  [ % of Signal,  floor (abs)   , positive ceiling,   negative ceiling]
xpertvec_in = [.01*ones(nx_ekf,1),[.0001*ones(nx_ekf,1)], ...
              [1000*ones(nx_ekf,1)],[-1000*ones(nx_ekf,1)]];
if nw_ekf>0
wpertvec_in = [.01*ones(nw_ekf,1), .0001*ones(nw_ekf,1)], ...
              [1000*ones(nw_ekf,1), -1000*ones(nw_ekf,1)];
else
wpertvec_in = [0 0 0 0];
end
% Flag to specify Xest output:
% 1 = enable output of propagated states in between measurement updates
% 0 = hold the previous updated states in between measurement updates
Extrapolate_Output = 1;
```

Table 17: Initialization File Setting for Simple System Example

xpertvec_in and wpertvec_in Column Definitions;

[Fraction, Non-Zero Default, Pos Limit, Neg Limit]

where the positive and negative perturbation steps are computed as:

Pos step = min{ max[abs(Fraction*Signal value), Non-Zero Default], Pos Limit}
Neg step = max{ min[-abs(Fraction*Signal value), -Non-Zero Default], Neg Limit}

Table 18: User Specifications of *xpertvec_in* and *wpertvec_in* to Control Perturbation Step Size

In the EKF block operation, EKF estimator signal vectors must be decomposed and routed to the appropriate plant and auxiliary model blocks, and likewise, the output signals from

the model blocks must be gathered and combined to reform the EKF vectors again. Sections 6 and 7 of the Initialization File, shown in Table 19, allow the user to specify the EKF and plant/auxiliary model signal relationships so that the signal routing will be set up automatically in the generic model blocks. For instance, specifying *i_Xest2xPlant* = [1:2] identifies states 1 and 2 of the EKF estimator as states 1 and 2 of the plant model, and specifying *i_Xest2xAux* = [3] identifies states 3 of the EKF estimator as state 1 of the auxiliary model. It is common to have EKF states which are in fact unknown inputs or parameters in the plant model, and the terms such as *i_Xest2uPlant* or *i_Xest2pPlant* are used to specify the EKF states that become plant model inputs, *u*, or parameters, *p*. The accompanying *terms i_uPlant2replace* and *i_pPlant2replace* are then used to identify where these states fit in the input or parameter vectors.

```
% 6.  Specify Signal Indices to translate from EKF x,u,p,w signals
%         to Plant/Aux model x, u, p, w signals
% EKF to PLANT model signals
% States
i_Xest2xPlant = [1:2];  % These elements of EKF state vector are the
%                         Plant model states
% Inputs
i_Xest2uPlant = [0];    % These elements of EKF state vector are the
%                         'estimated' Plant model inputs if any
i_uest2uPlant = [1 ];   % These elements of EKF input vector are the
%                         remaining Plant model inputs
i_uPlant2replace = [0]; % Indices of the estimated inputs in the u vector
% Parameters
i_Xest2pPlant = [0];    % These elements of EKF state vector are the
%                         'estimated' Plant model params if any
i_pest2pPlant = [1 2];  % These elements of EKF param vector are the
%                         remaining Plant model params
i_pPlant2replace = [0]; % Indices of the estimated params in the p vector
% Noise
i_west2wPlant = [1 2 ]; % These elements of EKF noise vector
%                         drive the Plant model process noise
% EKF to AUX model signals
% States
i_Xest2xAux = [3];     % These elements of EKF state vector are the
%                        Aux model states
% Inputs
i_Xest2uAux = [0];    % These elements of EKF state vector are the
%                       'estimated' plant model inputs if any
i_uest2uAux = [0 ];   % These elements of EKF input vector are the
%                       remaining Aux model inputs
i_uAux2replace = [0]; % Indices of the estimated inputs in the u vector
% Parameters
i_Xest2pAux = [0];    % These elements of EKF state vector are the
%                       'estimated' Aux model params
i_pest2pAux = [3];    % These elements of EKF param vector are the
%                       remaining Aux model params
i_pAux2replace = [0]; % Indices of the estimated params in the p vector
% Noise
i_west2wAux = [3];     % These elements of EKF noise vector
%                        drive the Aux model process noise

% 7.  Specify Signal Indices to translate from Plant/Aux model x,y
%               signals to EKF x,y signals
%       Plant/Aux to EKF model signals
% States
i_xPlant2Xest = [1 2];
i_xAux2Xest = [1];
%Outputs
i_yPlant2yest = [1 ];
i_yAux2yest = [1];
```

Table 19: Initialization File Settings for Simple System Example

A.1.4 EKF Block Operation

The two major EKF functions, State Propagation and State Update, are performed by the highlighted blocks of Figure 74. The Propagate block executes at the basic time step interval and propagates the EKF state and covariance forward in time using the plant and auxiliary models, driven by the measured system inputs. When measured system outputs are available, the Update block will execute, correcting the current propagated state and covariance with information contained in the output measurements. The Update block can execute at a lower rate than the Propagate block, and this is controlled by the user through an externally generated update flag signal.

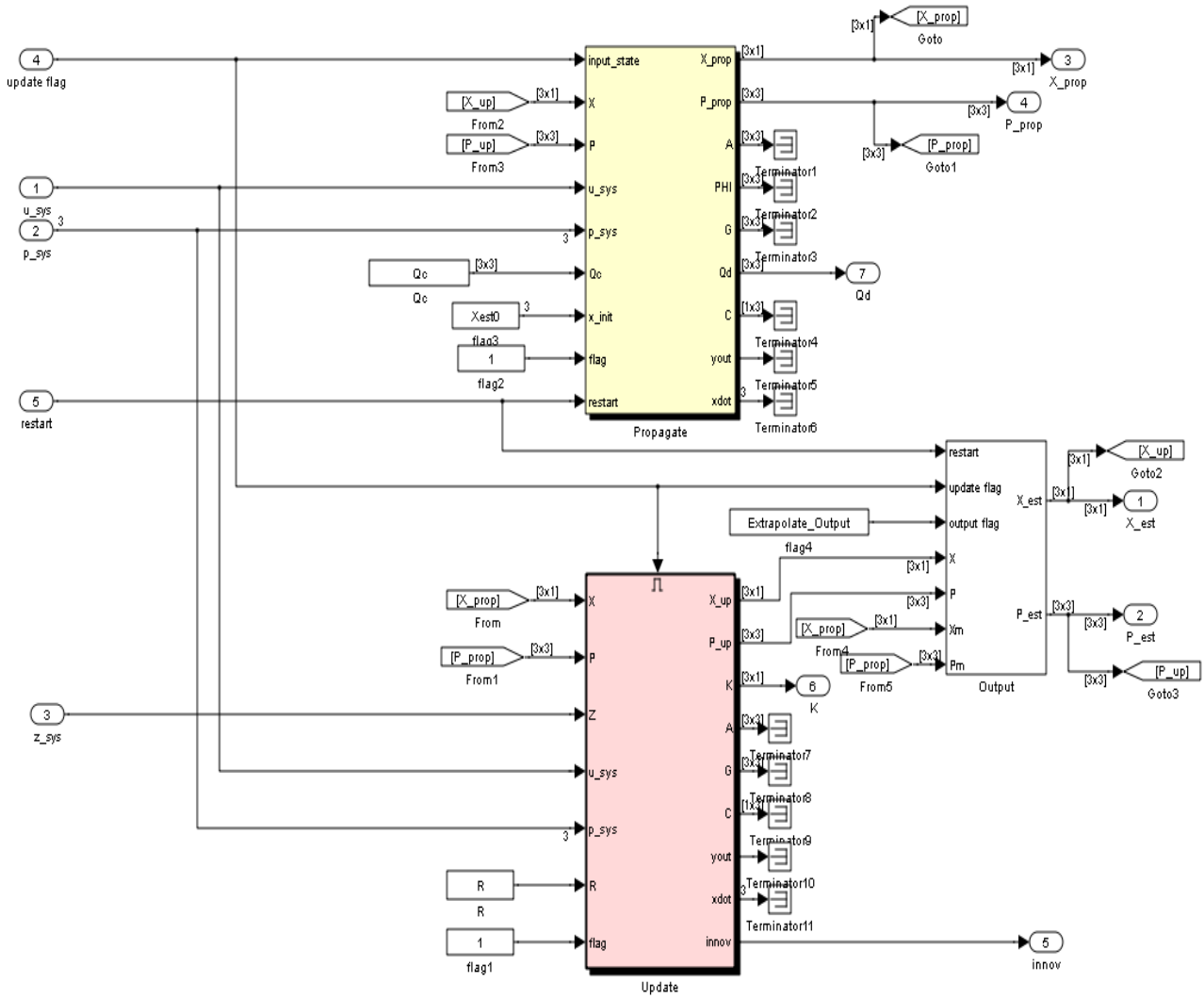


Figure 74: Propagate and Update Functions within the EKF Block

Both the *Propagate* and *Update* functions require that the EKF model provide an output vector, *y*, and state derivative vector, *xdot*; and this is satisfied by the requirement on the

form of the plant and auxiliary models that must be supplied. However, solution of the EKF estimation algorithm also requires linear system approximations about the current estimated state vector and operating point. To perform this linearization, a generic EKF model linearization block was created base on forward and backward signal perturbation. The EKF model linearization block is shown at the left in Figure 75, along with the functions inside it on the right. This block is implemented as a 'while iterator' block, which will execute many times within each basic time step, where the number of iterations will depend on number of EKF states and noise inputs that must be perturbed to produce the linearized Model.

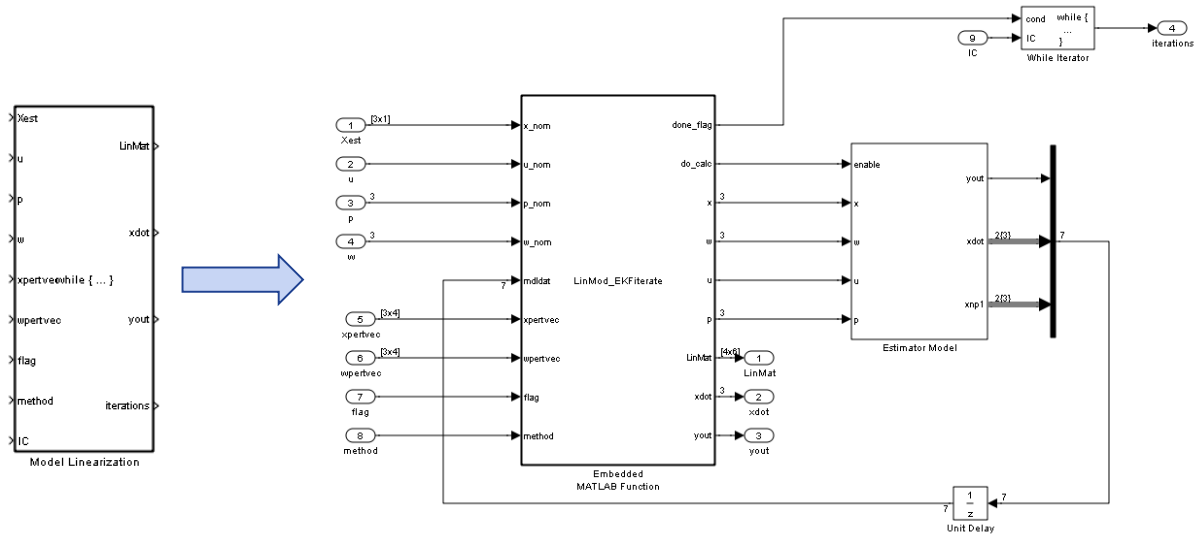


Figure 75: EKF Model Linearization Block

The contents inside the 'Estimator Model' block are shown in is the full EKF model which is shown in . The perturbation of the EKF model and the linear model generation is orchestrated by means of the embedded Matlab Function block in Figure 75.

A.2 Modular OSP Tool in Matlab©

The OSP tool is designed to solve the optimal sensor placement problem, in the form of NLP or LP, with optional search algorithms and optional solvers. The module allows the user to input the OSP problem with pre-specified interfaces. The top level user interface is shown in the following table (Table 20).

```
The Demo of the Optimal Sensor Placement Algorithm
Step-1: System specification
Input system data file name:

Step-2: Include reliability constraint
Check reliability? Y/N [N]: (Otherwise, only precision)

Step-3: Option for choosing OSP formulation
OSP formulation
 3.1) LMI based
 3.2) Standard INLP based
Choose from above options 3.1/3.2:

(if 3.1 chosen)
Step-4: Option for choosing LMI solvers
 4.1) LMI-Lab provided by Matlab
 4.2) LMI-SDPT3

(if 3.2 chosen)
Step-4: Options for optimization algorithm
 4.1)Branch-and-Bound
 4.2)Outer-Approximation
```

Table 20: OSP tool top level user interface for Branch-and-Bound

At the top level user interface, the tool expects users to specify the OSP problem in the Matlab readable MAT file. The following subsection will describe the outline of preparing such a MAT file. The subsequent steps deal with choosing the criteria (reliability), OSP framework and optimization algorithms.

The user can specify the OSP with precision requirement and reliability requirement. The precision criterion is always enforced. If the reliability requirement is not selected, the OSP tool will solve the OSP problem with only precision requirement. The reliability

implemented in the OSP tool is the “functional” reliability as described in Section 4. The “estimation” reliability is not implemented in this modular general purpose tool. The main reason for the choice is that for problems with *N* number of possible sensor locations, the estimation reliability metric requires 2^N precision constraint computations, which even with small scale problems of say *N*=10 leads to 2^{10} computations. The time required to compute this metric is quite prohibitive. Since the intent of modular OSP tool is to help in designing a reliable and precise estimation in the cost-optimal manner, the choice of functional reliability along with the precision constraint will make the tool applicable to a much wider class of problems.

A.2.1 Input Data

The user is required to provide for certain data to setup the OSP problem, which includes 1) the system matrices of the linear or linearized dynamic system in discrete-time domain; 2) the precision requirement; 3) the reliability requirement; and 4) the cost of the sensors in the network. The input data is required to be in mat file format in the Matlab. An example m-file “Setup_OSP.m” has been provided in Table 21 to help the user to prepare the input data required at Step-1 (Table 20) to setup the OSP problem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % This m-file is used to prepare the input mat file required for the OSP
problem with Branch and Bound search Algorithm. This file will create a .mat
file with following data
%-----------------
%
%  1) Ad: Discrete system matrices (N x N)
%  2) Cd: Discrete system matrices (M x N)
%  3) Q: Covariance of process noise w (N x N)
%  4) R: Covariance of the measurement noise v (M x M)
%  5) ReqstPrec: Precision request
%  6) PrecIdx: Index of states on which precision is enforced
%  7) ReqstRelib: Reliability requirement
%  8) RelibVec: Survivability for sensors,(M x 1)
%  9) CostVec: Vector defining the sensor cost (M x 1)
%  10) SensorIdx: Predefine a sensor network to include prior information (M x
1)
%       vector of o, 1 or Nan
%       0: no sensor;
%       1: fixed sensor;
%       Nan: the potential sensor measurement for OSP problem.
%
%
% %%%%%%%%% Example%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1. Define the problem formulation for discrete-time linear system
% system matrix x(k+1) = Ad x(k) + w(k)
%               y(k) = Cd x(k) +v(k)
  Ad = eye(7,7);  % system has 7 states
  Cd = rand(5,7);  % system has 5 output
  Q = eye(7);
  R = eye(5)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2. Estimation requirement for OSP problem
% The OSP problem has multiple precision requirements:
```

```
% 1) precision 0.2 on the 2nd element of states; and
% 2) precision 0.4 on the 5th element of states, then define
  ReqstPrec = [0.2;0.4];
  PrecIdx = [2;5];
%
%%%%%%%%%%%%%%%%%%%%%%%%
% 3. Reliability requirement for OSP problem
%
% The OSP problem requires 80% of reliability of the sensor network
 ReqstRelib = 0.8;

% The sensor survivability at the 1st, 2nd and 3rd measurements are 0.9.
% and at 4th and 5th measurements are 0.8, then define:
  RelibVec = [0.8;0.9;0.9;0.8;0.7];

% 4. Cost of the sensor network
  CostVec = [100; 120; 110; 60; 50];

% 5. Predefine sensor network to include prior information multiple options
% want to indicates there is always sensors at the first measurement location,
% no sensors at the fourth measurement location
% OSP problem is to find potential sensor @2nd, 3rd and 4th measurement location
  sensorIdx = [1; NaN; NaN; 0; NaN];

% 6. Save the outputs in a mat file
  save OSP_trivial_example.mat Ad Cd Q R ReqstPrec PrecIdx ReqstRelib RelibVec
  CostVec sensorIdx
```

Table 21: OSP tool: help file of the Setup_OSP.m

A.2.2 Output Data

The output of the OSP tool is a Matlab structure *SolutionData.* The field of this structure is listed in Table 22.

Output	**Description**	**Size (at each time step)**	**Comments**
SolutionData. Costbest	The cost for the optimal sensor placement solution.	Scalar	With satisfied estimation and/or reliability requirements.
SolutionData. Optbest	Location for optimal sensor placement (solution to OSP problem).	[M x # of possible solutions]	0: no sensor 1: 1 sensor 2: 2 sensors...
SolutionData. Cputime	The total CPU time spent to solve the OSP problem	scalar	In second

Table 22: OSP Tool Outputs

A.2.3 Software Package Needed for the OSP Tool

The OSP tool is Matlab© based but it requires few additional software packages for all the optimization options to work.

For the LMI based OSP formulation the OSP loot uses LMI-Lab, developed by Mathworks and provided in the Matlab Control toolbox, SDPT3, which is the free open source. In the OSP tool, YALMIP is used to integrate SDPT3 in the Matlab environment. The solver IPOPT is implemented to solve NLP relaxation problem and a combination of Ipopt and GNU Linear Programming Kit (GLPK) packages are used to solve the Outer-Approximation problem. SDPT3, YALMIP, IPOPT, GLPK are all freely available and can be downloaded from links below.

The link for SDPT3 can be found as follows,
http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

The link for YALMIP can be found as follows:
http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Main.HomePage
Note that the above software packages get updated from time to time. The OSP tool has been tested based on the latest release when the report is generated.

In the OSP tool, SDPT3-4.0 is tested with release YALMIP R20130128

The link for IPOPT can be found as follows:
http://www.cs.ubc.ca/~pcarbo/ipopt-for-matlab/.
Further details about the exact algorithm and other options can be found in the following link: http://www.coin-or.org/Ipopt/documentation/node10.html

The link for GLPK can be found as follows:
http://www.gnu.org/software/glpk/#TOCdocumentation