Final Report

Development of PUNDA
(Parametric Universal Nonlinear Dynamics Approximator)
Models for Self-Validating Knowledge-Guided
Modelling of Nonlinear Processes in
Particle Accelerators & Industry

DE-FG02-04ER86225

Dr. Bijan Sayyar-Rodsari, PI
Dr. Carl Schweiger and Dr. Eric Hartman

Pavilion Technologies, Inc., Austin, TX 78758

October 4, 2007

Summary

Optimization-Based Control (OBC) is a widely adopted control technology that optimizes forecasts of process behavior, over the manipulatable process inputs, . The forecasting is accomplished with a process model. The *accuracy* and *computational efficiency* of the process model are critical to the feasibility of the OBC approach. While "accuracy" determines whether process response is correctly anticipated, "computational efficiency" determines the frequency at which an OBC solution can be reliably applied. The main modelling challenge is how to accommodate the need for higher model accuracy while limiting the computational cost of real-time optimization. Our Phase II STTR research has pursued a systematic methodology for building **A**ccurate and **C**omputationally **E**fficient (ACE) models using an innovative parametric modelling framework known as **P**arametric **U**niversal **N**onlinear **D**ynamics **A**pproximator (PUNDA).

The main activities during the two year period of the Phase II STTR project may be divided in three categories:

1. **Software Development:** The software framework for constrained training of PUNDA models has been developed. Efforts are made to integrate the newly developed training framework with Pavilion's commercially available modelling, optimization, and control software (Known as Pavilion8).
2. **Nonlinear Dynamic System Identification**[1]**:** Varying parameters of nonlinear dynamic systems are identified using PUNDA model structure. In addition to synthetically generated data, real process data from a fermentation process in a bio-reactor has been used to demonstrate the applicability of the PUNDA modelling framework in nonlinear dynamic identification problems.

   (a) Both continuous and discrete representations of the nonlinear dynamic systems are examined in our study:

      i. For continuous domain representation, parameters such as gain, K, and residence time, τ, that are commonly used to describe dynamic behavior of the system are identified.

      ii. For discrete domain representation, $a_{1,2}/b_{1,2}$ for first and second order difference equations are identified.

   (b) Both nonlinear input/output and nonlinear state-space representations of a nonlinear process are studied in the course of this STTR Phase II project.

3. **Emittance Modelling for rf Photoinjectors:** Given beam size measurements for a range of quadrupole magnet currents for Gun Test Facility (GTF) at Stanford Linear Accelerator Center (SLAC), beam size has been modelled as a function of quad magnet current using a PUNDA structure. Three different scenarios have been considered here:

   (a) The transport matrix between the exit of the linac section and the spectrometer screen is fully known.

   (b) The transport matrix is identified at the same time the beam matrix parameters are identified. Physically meaningful constraints on how the quadrupole current affects transport matrix elements are used to guide the optimizer's search for best beam parameters.

   (c) Using first principles knowledge, the transport matrix is modelled as an explicit function of quadrupole magnet current. Two specific cases are examined here:

      i. The relationship between quad magnet current and the exerted magnetic field is assumed known.

[1] To the best of our knowledge this is the first systematic attempt in identification of nonlinear systems with varying dynamics.

ii. A neural network is used to model the relationship between quad magnet current and the exerted magnetic field.

4. **Beam Lifetime Modelling for Synchrotron Light Sources:** Given a fundamental model of beam lifetime in which Touschek, Bremsstrahlung, and Coulomb effects on beam loss are reflected, a PUNDA model is developed to capture the dependence of the beam lifetime on gap voltage (V_{rf}), vertical scraper position (Y_s), dynamic vacuum pressure (P_{dyn}) and number of bunches (M_b). The PUNDA model is shown to correctly predict electron beam loss under various operating conditions of the synchrotron light source.

The results of our Phase II STTR research on emittance modelling for rf photoinjectors and beam lifetime modelling for synchrotron light sources were presented as three papers at the 2007 meeting of Particle Accelerator Conference in Albuquerque, NM.

1 - Introduction

Optimization-Based Control (OBC) is a widely adopted control technology that optimizes forecasts of process behavior, over the manipulatable process inputs. The forecasting is accomplished with a process model (*i.e.* a mathematical representation of the process), and, therefore, the model is a key component of any optimization-based control strategy [1]. Optimization-based control is "*the most natural and in some cases the only methodology for control of systems that are governed by constrained dynamics* [2]" and therefore, is increasingly the control methodology of choice in difficult nonlinear applications that range from biochemistry [3], medicine [4], and process industries [5], to aerospace [6].

The optimization problem at the heart of OBC is an *open-loop* finite-horizon optimal control problem that is solved repeatedly. Therefore, the feedback in an OBC controller is only an *implicit* feedback that is produced by receding horizon implementation (*i.e.* applying a fraction of the optimal trajectory for manipulatable process inputs, measuring process outputs, and solving the optimization problem again at the next step). With an implicit feedback, *accuracy* and *computational efficiency* of the process model essentially determine whether an OBC controller will be successful. "Accuracy" determines whether the OBC controller correctly anticipates process response. "Computational efficiency" determines whether real-time optimization can produce an acceptable solution by the time the OBC controller is expected to commit its next move. The main challenge in modelling for OBC stems from the need to build highly accurate models that are computationally efficient. This STTR project was indeed motivated by the fact that a systematic methodology for building **A**ccurate and **C**omputationally **E**fficient (ACE) models was absent in prior research.

In Phase I, we introduced **P**arametric **U**niversal **N**onlinear **D**ynamics **A**pproximator (PUNDA) models [7], as a framework for building ACE models. As shown in Figure 1, a PUNDA model is formed by a series connection of a Static Nonlinear Mapping (SNM) block and a Parametric Nonlinear Model (PNM) block. The parameters, $\vec{p}$, in the PNM block may vary as a function of process inputs, $\vec{u}$. A parametric multi-input multi-output (MIMO) difference equation is the default choice for the PNM block, although other parametric nonlinear models may be used. Neural Networks (NN) are a preferred choice for the SNM block, which is fully determined by the NN weights and biases. Constrained optimization determines the NN weights and biases such that the modelling error (the objective function) is minimized, subject to any imposed constraints. This constrained optimization is known as the *training* of the PUNDA model.

The main technical objective of the Phase I research was to demonstrate that PUNDA models are accurate and computationally efficient (ACE) models that can be successfully trained using constrained optimization. In Phase II we have further explored this technical objective and have systematically examined its suitability to model nonlinear dynamic systems. In particular, We have used PUNDA framework to model beam emittance as a function of quadrupole magnet current, I_2, given real measurements of beam size at Gun Test Facility (GTF) at SLAC. We have also used a PUNDA structure to study beam loss in synchrotron light sources, and have developed models for beam lifetime that capture the dependence of the beam lifetime on gap voltage (V_{rf}), vertical scraper position (Y_{s}), dynamic vacuum pressure (P_{dyn}) and number of bunches (M_{b}).

2 - Background Information

This section provides background information for complex system modelling. In particular a brief mathematical description of the PUNDA model is provided to facilitate the discussion of research results in Sections 3.

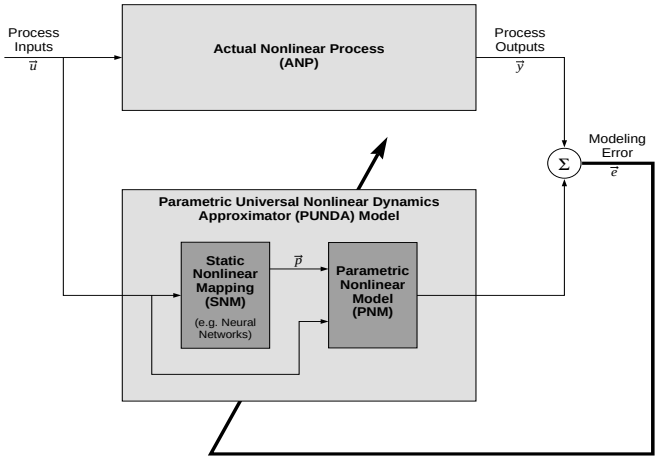

Fig. 1: Diagram of the modelling with a PUNDA model. In this diagram: (a) ANP block represents the actual nonlinear process to be modelled whose inputs/outputs (I/O) are measured. Often I/O measurements are augmented with additional information about the process. (b) The PUNDA model is formed by the series connection of the static nonlinear mapping (SNM) block and the parametric nonlinear model (PNM) block. Both steady state and dynamic models may be used for the PNM block. The diagonal arrow through the PUNDA model indicates that the "modelling error" is the objective function to be minimized during the training of the PUNDA model.

2.1 - Background Information on Complex System Modelling

The overwhelming success of model-based optimization and control in all aspects of modern life (aerospace and transportation, materials and processing, biology and medicine, robotics, and information and networks) has given mathematical modelling a critical role in all fields of engineering and physics. "*Mathematical models may be developed along two routes (or a combination of them). One route is to split up the system, figuratively speaking, into subsystems, whose properties are well understood from previous experiences. This basically means that we rely on laws of nature and other well-established relationships that have their roots in earlier empirical work. These subsystems are then joined mathematically and a model of the whole system is obtained. ... The other route to mathematical models is directly based on the experimentation. Input and output signals from the system are recorded and subjected to data analysis in order to infer a model* [8]." Literature often refers to the first route as First-Principles (FP) modelling, while the second route is commonly referred to as Empirical Modelling (EM) (although empirical data is also involved in building FP models).

The accumulated experience over the past several decades has revealed the strengths and weaknesses of these two routes when applied to real-world complex systems.

For the FP route it has become clear that:

1. FP models are built based on the science behind the process, and hence are better suited for representing general process behavior over the entire operation regime.
2. FP information is often incomplete/inaccurate.
3. FP model parameters must often be tuned before use in optimization/control.
4. FP models are computationally expensive (particularly when model output is not explicit[2]), and are often only feasible for real-time optimization/control when the process is slow.

[2] An example is $\mathcal{G}(y_k, u_k, x_k) = 0$ where the output vector y_k is an implicit function of input vector u_k and state vector x_k. An internal solver is therefore needed to solve for y_k at each interval.

5. When the process changes, modification of the FP model is, in general, expensive. Designed experiments may be needed to obtain the necessary data.

For the EM route it has become clear that:

1. Since data captures the non-idealities of the *real* process, an EM model can be more accurate where data is available.
2. The available data is often highly correlated and in this case process data alone is not sufficient to unambiguously break these correlations. This is particularly visible when process operation is recipe-dominated[3].
3. Designed experiments are often needed to produce the necessary uncorrelated data for EM modeling. Designed experiments disrupt normal plant operation and hence are highly undesirable.
4. Certain regions of operation are typically avoided, and hence representative data for those regions will not be available.

The complementary strengths and weaknesses of these two modelling routes are widely recognized [9, 10, 11], and the value of an approach that allows for their strengths to complement one another is generally acknowledged [11, 12].

Logically, there are two basic ways to combine EM and FP models:

1. *Parallel Combination*, Figure 2.a, where the residuals, or errors, of an FP model are modelled by an EM[4], and the outputs of the two models are added together to give the combined output.

2. *Series Combination*, Figure 2.b, where an EM model directly rectifies a primary cause of FP model errors, rather than simply compensating for them. Rather than fitting the FP parameters, $\vec{p}$, to data as constants, as is the common practice, an EM is used to model the parameters, $\vec{p}$, as functions of the process inputs. The output of the EM model is thus input to the FP model[5]. A powerful aspect of this structure is that the capturing of the previously unknown ($\vec{u}$-$\vec{p}$) relationship by the EM model requires no data for the parameters, $\vec{p}$, which is generally unavailable, but requires only data for the inputs/outputs of the combined model.

In comparing the two approaches, the parallel EM acts as a "clean-up" model to compensate for cumulative inaccuracies in the FP model. This approach simply observes the errors of the FP model "from the outside" and compensates for inaccuracies without improving the structure or accuracy of the FP model itself. The series approach directly addresses insufficiencies in FP models "from the inside", improving their accuracy and extrapolation capability by adding detail to their structure.

In comparing the training of the EM blocks in the two approaches, a major distinction is that the outputs of the EM model in the parallel combination are directly measurable, while the outputs of the EM model in the series combination (*i.e.* FP parameter values, $\vec{p}$) are not. Consequently, the objective function in the parallel combination cannot be formulated explicitly in terms of the parameters, $\vec{p}$; instead,

[3]For example, in a linear system with 2 inputs and 1 output, a recipe may require two inputs to move simultaneously, one to increase by one unit and the other to decrease by one unit. Even if the output is known to increase by one unit, the sign and value of the gains from the two inputs to the output can not be uniquely determined based on this data *alone*.

[4]In the literature on combined models, NN models have by far been the primary choice for empirical modelling, and are the only construct to be considered for the SNM block in Figure 26 in this proposal.

[5]As a simple example, in the non-ideal gas model $PV = zT$, the compressibility factor z is traditionally fitted as a different constant for different regions of T, P, and V. In the series combination, z is instead modelled as an (unknown) function of T and P (and also mixture composition).

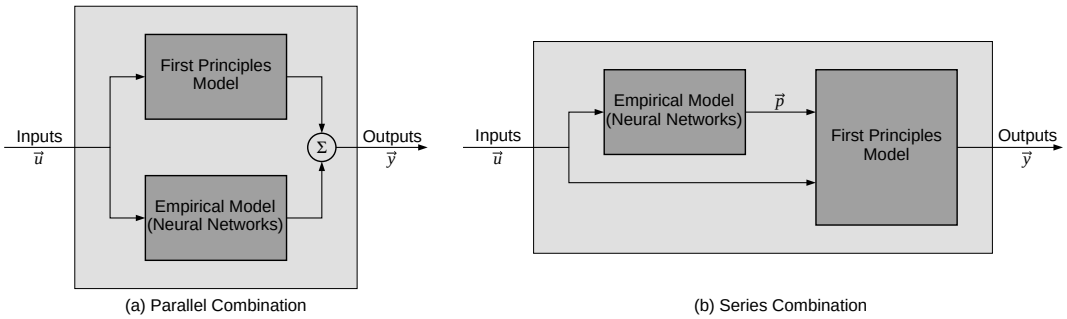

Fig. 2: Block diagram of the two basic ways FP and EM models may be combined.

the objective function is the error of the FP model outputs. This, in general, constitutes a more difficult optimization problem than the parallel combined models.

The reported research on combined models, despite their logical appeal and straightforward structure, is surprisingly limited. In addition it falls significantly short for our purposes with respect to algorithmic issues, both technical and theoretical, that are indispensable for a model-based control system to be capable of *general* real-world applications. In our Phase II proposal we pointed out that research prior to our Phase I efforts was unable to build combined models in the following situations:

1. *Only partial FP model is available:* For example, in a multi-input/multi-output process, the lack of the FP model for even one input/output pair is enough to prevent prior research from building a series combined model for the process that is appropriate for optimization and control, even if representative measurements of all process inputs/outputs are available. In process industries, for instance, the efforts to complete FP models in such cases are extremely expensive and often require outside expertise, leaving the available FP information essentially unusable.

2. *FP model only implicitly describes the relationship between inputs/states/parameters/outputs:* Prior research simply assumes that process outputs are always explicitly defined by the FP model, which is generally only rarely the case. Building combined models with an implicit FP model presents significant algorithmic and optimization challenges (see research issue #4 in this section), which is nowhere addressed in prior research.

3. *Higher-order fidelity of the combined model is needed:* Often, accurate first and/or second order derivatives of the process outputs with respect to the process inputs are highly critical in model-based optimization/control applications. Available degrees of freedom in NN models may be exploited to enforce desired first/second order derivative behavior in a combined model. Prior research is completely silent on how these higher order constraints may be imposed during the training of a combined model.

Yet, in almost all real-world scenarios of complex system modelling, one or more of these challenges arise. This STTR project was indeed motivated by the need to enable combined modelling under such challenging scenarios. Consequently, our research in the course of the STTR project has focused on demonstrating the applicability of the PUNDA approach to complex system modelling and on documenting the challenges with constrained training of these models to the extent necessary for the development of a commercial software for this purpose under realistic scenarios.

We conclude this background section with a brief mathematical description of the PUNDA models.

2.2 - A Description of PUNDA models

Assume the following general discrete-domain description of a nonlinear dynamic process:

$$\begin{cases} x_k &= F_k\left(u_k, x_{k-1}, \rho_k\right) \\ y_k &= G_k\left(u_k, x_{k-1}, \rho_k\right) \end{cases} \tag{1}$$

where $x_k \in \mathcal{R}^{N_x \times 1}$ is the state vector, $u_k \in \mathcal{R}^{N_u \times 1}$ is the input vector, $y_k \in \mathcal{R}^{N_y \times 1}$ is the output vector, and $\rho_k \in \mathcal{R}^{N_p \times 1}$ is the physical parameter vector at time k. Note that, for clarity of the derivation, x_k and y_k are defined as explicit functions of state/input/parameters.

A PUNDA model for the process is motivated by the *Taylor series expansion* of Eq. (1) around its current operating point, $\left(x^c, u^c, y^c, \rho^c\right)$:

$$\begin{cases} x_k &= x^c + \lambda_x\left(\delta x_{k-1}\right) + \lambda_u\left(\delta u_k\right) + \lambda_\rho\left(\delta \rho_k\right) \\ &\quad + \lambda_{xx}\left(\delta x_{k-1}\right)\left(\delta x_{k-1}\right) + \lambda_{uu}\left(\delta u_k\right)\left(\delta u_k\right) + \lambda_{\rho\rho}\left(\delta \rho_k\right)\left(\delta \rho_k\right) \\ &\quad + \lambda_{xu}\left(\delta x_{k-1}\right)\left(\delta u_k\right) + \lambda_{ux}\left(\delta u_k\right)\left(\delta x_{k-1}\right) + \lambda_{\rho x}\left(\delta \rho_k\right)\left(\delta x_{k-1}\right) \\ &\quad + \lambda_{x\rho}\left(\delta x_{k-1}\right)\left(\delta \rho_k\right) + \lambda_{u\rho}\left(\delta u_k\right)\left(\delta \rho_k\right) + \lambda_{\rho u}\left(\delta \rho_k\right)\left(\delta u_k\right) + \text{H.O.T} \\ y_k &= y^{ic} + \mu_x\left(\delta x_{k-1}\right) + \mu_u\left(\delta u_k\right) + \mu_\rho\left(\delta \rho_k\right) \\ &\quad + \mu_{xx}\left(\delta x_{k-1}\right)\left(\delta x_{k-1}\right) + \mu_{uu}\left(\delta u_k\right)\left(\delta u_k\right) + \mu_{\rho\rho}\left(\delta \rho_k\right)\left(\delta \rho_k\right) \\ &\quad + \mu_{xu}\left(\delta x_{k-1}\right)\left(\delta u_k\right) + \mu_{ux}\left(\delta u_k\right)\left(\delta x_{k-1}\right) + \mu_{\rho x}\left(\delta \rho_k\right)\left(\delta x_{k-1}\right) \\ &\quad + \mu_{x\rho}\left(\delta x_{k-1}\right)\left(\delta \rho_k\right) + \mu_{u\rho}\left(\delta u_k\right)\left(\delta \rho_k\right) + \mu_{\rho u}\left(\delta \rho_k\right)\left(\delta u_k\right) + \text{H.O.T} \end{cases} \tag{2}$$

where the coefficients $\lambda_x, \lambda_u, \cdots$ and $\mu_x, \mu_u, \cdots$ are the coefficient matrices in the Taylor series expansion[6], and H.O.T refers to the higher order terms in the Taylor series expansion. Note that

(a) Using Eq. (2), the nonlinear process is described with a nonlinear Parametric MIMO Difference Equation (PMDE).

(b) Around its current operating condition, one can approximate the nonlinear process in Eq. (1) with any given degree of accuracy by simply including enough terms from its Taylor series expansion.

(c) The coefficients of the Taylor series expansion are essentially functions of process inputs[7].

In Phase I, we proposed to build a PUNDA model for the process described by Eq. (1) as follows:

1. Use the PMDE description in Eq. (2), with an appropriate number of terms, as the PNM block in the PUNDA model.
2. Treat the elements of the coefficient matrices $\lambda_x, \lambda_u, \cdots$ and $\mu_x, \mu_u, \cdots$ as the parameter vector P in the PUNDA model[8].
3. Use an empirical model (NN in particular) to capture the functional dependency of these parameters on process inputs.

[6]Given FP model in Eq. (1), the coefficient matrices in Eq. (2) are well-defined *analytical* function. For example:

$$\lambda_x = \frac{\partial F_k}{\left(\partial x_{k-1}\right)}, \quad \lambda_u = \frac{\partial F_k}{\left(\partial u_k\right)}, \quad \lambda_{uu} = \frac{1}{2}\frac{\partial F_k^2}{\left(\partial u_k\right)\left(\partial u_k\right)}, \quad \lambda_{ux} = \frac{1}{2}\frac{\partial F_k^2}{\left(\partial u_k\right)\left(\partial x_{k-1}\right)},$$
$$\mu_x = \frac{\partial G_k}{\left(\partial x_{k-1}\right)}, \quad \mu_u = \frac{\partial G_k}{\left(\partial u_k\right)}, \quad \mu_{uu} = \frac{1}{2}\frac{\partial G_k^2}{\left(\partial u_k\right)\left(\partial u_k\right)}, \quad \mu_{ux} = \frac{1}{2}\frac{\partial G_k^2}{\left(\partial u_k\right)\left(\partial x_{k-1}\right)},$$

[7]Assuming that the process in Eq. (1) is asymptotically stable, the effect of the initial conditions on process behavior will decay to zero and hence it is valid to consider coefficient matrices as functions of process inputs only.

[8]Note that due to the truncation of the PMDE model to a finite number of terms, the coefficient matrices could in general be different from Taylor series coefficients.

4. Given process data and constraints derived from FP process knowledge, solve a constrained optimization problem to train the empirical model.

For Phase I of this STTR project, we focused on a case where the state vector, x_{k-1}, was fully constructed as a function of past process inputs/outputs. Assuming that $x_{k-1} = S_{k-1}\left(u_{k-1}, u_{k-2}, \cdots, y_{k-1}, u_{k-2}, \cdots\right)$, the output equation in Eq. (1) may be expressed as:

$$y_k = G_k\left(u_k, S_{k-1}\left(u_{k-1}, u_{k-2}, \cdots, y_{k-1}, u_{k-2}, \cdots\right), \rho_k\right) \tag{3}$$

for which the Taylor series expansion reduces to:

$$y_k = y^{\text{init}} + \sum_{i=1}^{Y_{\text{past}}} A_i \delta y_{k-i} + \sum_{i=0}^{U_{\text{past}}} B_i \delta u_{k-i} \tag{4}$$

where $Y_{\text{past}}/U_{\text{past}}$ are the number of past outputs/inputs required to fully construct the state vector, and A_i/B_i are the coefficient matrices of appropriate dimension.

Our Phase I research demonstrated that (a) the input/output model of Eq. (4) is widely applicable, and (b) a truncation of the PMDE model to only a few terms is sufficient to build accurate models for complex nonlinear processes[9]. For Phase II, we applied the PUNDA modelling framework to a state-space model of batch fermentation process in a bio-reactor using real process data. The results of this study, reported in Section 3, demonstrate the applicability of the PUNDA models to the state space representation of the complex nonlinear processes.

3 - Phase II STTR Research Results

In this section, we present the results of our research and development in the course of this Phase II STTR project. The software development efforts that have enabled constrained training of the PUNDA models are described in Section 3.1. The application of PUNDA models in the identification of varying parameters of nonlinear dynamic systems is described in Section 3.2. The application of PUNDA models in the modelling of transverse beam matrix given transverse beam size measurements are described in Section 3.3. The results of the beam lifetime study in synchrotron light sources are presented in Section 3.4. The application of the PUNDA modelling framework to the state-space representation of a nonlinear process is described in Section 3.5.

[9] In particular, we used the PMDE description of Eq. (4) with $Y_{\text{past}} = 1$ to build a PUNDA model for gas composition in a real polymerization reactor with 11 inputs and 6 outputs (with correlated measured data and only partial FP information).

3.1 - Software Development for Constrained Training of PUNDA Models

A modeling and optimization framework has been developed for the purpose of creating combined models and implementing the constrained training techniques. The modeling component of the framework is used to describe explicit, multi-input, multi-output mappings. The optimization component allows the user to describe an optimization problem by identifying the variables, objectives, and constraints for the optimization along with the computational model and the solver to be used. The optimization framework addresses both the training of a model to determine its parameters as well as the use of the model in other optimization settings.

3.1.1 - Modeling Component: The modeling framework has been developed to facilitate the construction and usage of nonlinear models for use in a broad range of optimization and control applications. The base component of the modeling framework is the *Numeric Mapping*, which is an abstract base class for the various model types.[10] The general mathematical representation of the Numeric Mapping is the following multi-input, multi-output operator:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$$

where $\boldsymbol{x}$ is the vector of inputs, $\boldsymbol{f}$ is the operator, and $\boldsymbol{y}$ is the vector of outputs. Although the mapping must behave as thought it has an explicit representation, models with implicit representations such as

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) = 0$$

can also be addressed. This requires that the method for computing the values of $\boldsymbol{y}$ as a function of $\boldsymbol{x}$ (such as an iterative solver) be contained within the mapping itself. Focusing on explicit models helps improve the computational efficiency since the outputs can be computed directly from the model inputs. Because the implicit form requires an iterative solver to compute the model outputs, model evaluation times can increase significantly. (Using the implicit form also complicates the usage of the model in the optimization framework; however, there are ways to address this issue.)

The basic development concepts for the Numeric Mapping are the following:

1. Provide a base modeling component that handles the computational aspects of the model including the output and gradient computations.

2. Serve as the model interface for the description of optimization problems that use gradient-based solvers.

3. Compute gradients automatically using both sensitivity (forward) and adjoint (backward) computation.

   (a) Allow arbitrary variable vectors for sensitivity calculations.

   (b) Allow arbitrary function vectors for adjoint calculations.

Two implementations of the Numeric Mapping are the Expression Mapping, and the Multi-Layer Perceptron (MLP) Mapping for Neural Networks.

[10] The terms "mapping" and "model" are interchangeable in this context.

3.1.2 - Expression Mapping: The *Expression Mapping* is a Numeric Mapping that allows models to be written using mathematical expressions. It employs a modeling language that allows the user to write the models in ASCII text using a basic editor. The Expression Mapping was developed with the following goals:

1. Allow well-defined mathematical models to be written in a clear, concise textual form.
2. Implement automatic differentiation for efficient gradient calculations.
3. Provide a framework for incorporating other models (especially MLP mappings) within the expressions.

The Expression Mapping uses a declarative language that describes the computations that must be made in order to compute the outputs from the given values of the inputs. In order to function within the modeling framework, the Expression Mapping must be differentiable and thus the expressions must adhere to rather strict modeling rules in order to maintain differentiability. The Expression Mapping has the following features:

1. Employs a clear, concise modeling language (written using the parser generator tool ANTLR).
2. Allows the declaration of local variables to help improve model clarity.
3. Allows output and local variables to be used as inputs in other expressions:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$$
$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$$

4. Allows the user to write the expressions in any order (Expressions are sorted to determine the appropriate computations sequence and detect algebraic loops).
(.)
5. Allows the declaration and usage of variables with arbitrary number of dimensions.
6. Allows explicit indexing over variables and index arithmetic.
7. Allows indexing operations over sets:
   (a) Indexing for set functions (summation).
   (b) Indexing over expressions (for each notation).
8. Includes basic mathematic operators (`+, -, *, /, ^`).
9. Includes numerous built-in functions (trigonometric, log, exp, etc.). (Allows new functions to be added rather easily.)
10. Allows user-defined functions.
    (a) Used to describe multi-input, single-output mappings.
    (b) Called as a function within an expression.
    (c) Uses argument position for input assignment.
11. Allows other Numeric Mappings to be used within the expressions.

(a) Used to describe multi-input, multi-output models.

(b) Called as a stand-alone statement.

(c) Uses labels for the association of variables to the inputs and outputs.

12. Allows the creation of composite models consisting of multiple Numeric Mappings within the Expression Mapping.

13. Employs error checking and provides error messages to assist in writing and debugging models.

The Expression Mapping is configured as a XML document that contains all of the information about the model in a loosely structured format. The elements of the XML document are the set declarations, input declarations, output declarations, parameter declarations, and the expression text. The set declarations define indexed sets that are used for the dimensions of the variables and for performing indexing operations. The input and output declarations are used to define the variables that can be connected to other mappings. The parameters are values that can be used be the mapping, but can not be wired to an external source. The expression text contains the statements that describe that mathematical representation of the model. The basic statement types that are used within the expression are the following:

1. Set declaration: The sets that will be used in the expression along with the definition of the elements of the set.

2. Variable declaration: The local variables that will be used in the expressions along with their dimensionality.

3. Function declaration: The user-defined functions.

4. Expression assignment: The actual expressions that indicate the mathematical computations to be performed.

5. Mapping assignment: Statements for using a mapping within the expressions.

Several examples of some basic mathematical entities and their corresponding modeling language representation are described in Table 1. A full XML document of an Expression Mapping is shown in Figure 3.

Table 1: Modeling language representation of mathematical constructs.

Mathematical Representation	Modeling Language Representation
$I = \{1 \dots 5\}$	`set I = |1:5|;`
$J = \{1 \dots 10\}$	`set J = |1:10|;`
$x \in \mathbb{R}^{5\times 10}$	`real x[I,J];`
$y \in \mathbb{R}^5$	`real y[I];`
$z \in \mathbb{R}^5, v \in \mathbb{R}$	`real z[I], v;`
$y_i = \sum_{j\in J} x_{ij} \quad \forall i \in I$	`y[i] = sum( j in J | x[i,j] ) foreach i in I;`
$z_i = \log\left(\frac{1}{1+c_i y_i^2}\right) \quad \forall i \in I$	`z[i] = log( 1/(1+c[i]*y[i]^2)) foreach i in I;`
$v = \sum_{i\in I} \frac{z_i}{y_i}$	`v = sum( i in I| z[i]/y[i] );`
$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{w})$	`g( in1:x, in2:w; out1:y );`

```
<mapping>
  <set name='I'>
    <elements type='range' start='0' stop='9'/>
  </set>
  <set name='J'>
   <elements type='range' start='0' stop='1'/>
  </set>
  <set name='K'>
    <elements type='range' start='0' stop='2'/>
  </set>
  <input name='x1' isScalar='false'>
    <dimension type='ID'  set='I'/>
  </input>
  <input name='x2' isScalar='false'>
    <dimension type='size' size='10'/>
  </input>
  <input name='x3' isScalar='false'>
    <dimension type='size' size='2'/>
    <dimension type='size' size='2'/>
    <dimension type='size' size='3'/>
  </input>
  <input name='x4' isScalar='false'>
    <dimension type='size' size='3'/>
    <dimension type='size' size='4'/>
  </input>
  <output name='y1' isScalar='false'>
    <dimension type='ID'  set='I'/>
  </output>
  <output name='y2' isScalar='false'>
    <dimension type='ID'  set='I'/>
  </output>
  <output name='y3' isScalar='false'>
    <dimension type='ID'  set='J'/>
    <dimension type='ID'  set='J'/>
    <dimension type='ID'  set='K'/>
  </output>
  <output name='y4' isScalar='false'>
    <dimension type='ID'  set='K'/>
  </output>
  <output name='y5'/>
  <output name='y6' isScalar='false'>
    <dimension type='ID'  set='I'/>
  </output>
  <parameter name='d'  isScalar='false'>
    <dimension type='ID'  set='I'/>
    <values>[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]</values>
  </parameter>
  <expressions>
    set L = |0:3|;
    real v[I];
    y1[i] = x1[i]+x2[i] forall i in I;
    y2[i] = x1[i]*x2[i] forall i in I;
    y3[i,j,k] = 2*x3[i,j,k] forall i in J, j in J, k in K;
    y4[k] = sum( l in L | x4[k,l] ) forall k in K;
    y5 = sum( k in K | y4[k] * sum(i in J| sum(j in J| y3[i,j,k]) ) );
    v[i] = (d[i] - x1[i])^2 forall i in I;
    y6[i] = v[i] forall i in I;
  </expressions>
</mapping>
```

Figure 3: XML document containing an Expression Mapping.

Dynamic modeling is currently addressed by treating the dynamic dimension as an algebraic dimension and using the index arithmetic to compute values of a variable at an index position as a function of the values at other (previous) index positions. This is suitable for modeling purposes where the index positions are fixed and absolute. However, this would not work for using the model online where the time dimension is relative and constantly changing.

Some of the future work for the Expression Mapping include (a) implementation of matrix/vector notation, (b) implementation of true dynamics, and (c) implementation of gain computations.

3.1.3 - Multi-Layer Perceptron (MLP) Mapping: The MLP Mapping provides a way to describe neural network models in a clear, concise format without having to describe the specific mathematical formulation representing the neural network. The MLP Mapping consists of layers, connections, inputs, outputs, gains. The MLP is configured as an XML document, and the elements of the document are the following:

Layer A layer in an MLP consists of an array of nodes and is configured by its `name`, `type`, `size`, and whether it has a `bias`. The name is an identifying string that is used by the connections to identify the layers used in the connection. The type of layer indicates the function to be used in each node within the layer. (All nodes within a layer must have the same function type.) The available function types are `identity`, `linear`, `quadratic`, `sigmoid`, `tanh`, `gaussian`, and `logcosh`. The size of the layer is an integer that indicates how many nodes are in the layer. Each layer also indicates whether or not the nodes have a bias by using a boolean flag. (The identity type is the same as linear with no bias.) The value information included with the layer holds the values for the bias for each node in the layer in the order of the nodes. Example:

```
<layer name="layer0" type="identity" size="3"/>
<layer name="layer1" type="tanh" bias="true" size="5">
  5.4, 4.3, 3.2, 2.1, 1.0
</layer>
<layer name="layer2" type="linear" bias="true" size="2">
  1.2, 2.3
</layer>
```

Connection A connection indicates the connectivity of the layers and is configured by identifying the names of the source (`srcName`) and destination (`dstName`) layers. The value information included in the connections contain the weights from the nodes in the source layer to the nodes in the destination layer. Example:

```
<connection srcName="layer0" dstName="layer1">
  1.1, 1.2, 1.3, 1.4, 1.5,
  2.1, 2.2, 2.3, 2.4, 2.5,
  3.1, 3.2, 3.3, 3.4, 3.5
</connection>
<connection srcName="layer1" dstName="layer2">
  1.1, 1.2,
  2.1, 2.2,
  3.1, 3.2,
  4.1, 4.2,
  5.1, 5.2,
</connection>
```

Input The input identifies a particular node in a layer as being a input to the MLP. It is configured by identifying the `name` of the input, the `layer` that contains the node, the index of the `node`, as well as the `scale` and `bias` for the input. Example:

```
<input name="x1" layer="layer0" scale="1.750" bias="0.03"/>
<input name="x2" layer="layer0" scale="327.1" bias="65.7"/>
<input name="x3" layer="layer0" scale="29.48" bias="-3.2"/>
```

Output The output identifies a particular node in a layer as being an output of the MLP. It is configured in the same way as the input. Example:

```
<output name="y1" layer="layer2" scale="273.1" bias="37.6"/>
<output name="y2" layer="layer2" scale="2.38" bias="0.19"/>
```

Gain The gain element informs the MLP that it needs to compute a gain within the MLP as an output. It is configured by indicating the `order` of the gain (either 1, 2 or 3), the name of the output, `dy`, and depending on the order of the gain, the inputs, `da`, `db`, `dc`. Given this information, the MLP will compute a gain that corresponds to $\frac{dy}{da}$ if the order is 1, $\frac{d^2y}{dadb}$ if the order is 2, and $\frac{d^3y}{dadbdc}$ if the order is 3.

The gain configuration can also specify a type as either `local` (default), `lower`, or `upper`. The `local` computes the value of the gain, while the `lower` and `upper` type computes the lower and upper bounds on the gain through the MLP. This type only works with the first order gains. Example:

```
<gain name="dy1dx1" dy="y1" da="x1"/>
<gain name="dy1dx2" dy="y1" da="x2"/>
<gain name="d2y1dx12" order="2" dy="y1" da="x1" db="x1"/>
<gain name="d2y1dx1dx2" order="2" dy="y1" da="x1" db="x2"/>
```

The implementation of the MLP Mapping has placed particular attention to computational efficiency in both output and gradient computation.

3.1.4 - Optimization Component: The optimization framework is used to describe and solve nonlinear optimization problems. The framework is based on gradient-based nonlinear programming algorithms such as Sequential Quadratic Programming (SQP) and Generalized Reduced Gradient (GRG) methods. An optimization formulation consists of the problem description, the model used for the optimization, and the solver used to solve the problem. The optimization problem is described by identifying the model inputs that are the variables along with their bounds, the model outputs that are constraints along with their bounds, and the model output that is the objective. The model that is used in the optimization is any Numeric Mapping representation and may correspond to a broad range of optimization problems including training problems, model identification, set-point computations, and control problems.

The optimization problem is the component that describes how the optimization model is to be used in the optimization problem. It specifies the variables, objectives, and constraints for the optimization formulation:

- Objective value
- Solution Status: optimal, infeasible, not solved, etc.
- Solver name: which solver was used to obtain solution.

- Variables: inputs of the optimization model that are to be used as variables along with their upper and lower bounds
  - values
  - lower and upper bounds
  - states
  - Lagrange multipliers
  - reduced gradients.
- Objectives: outputs of the optimization model that are to be used as objectives along with their linear weighting coefficients
  - values
  - linear coefficients
- Constraints: outputs of the optimization model that are to be used as constraints along with their upper and lower bounds
  - values
  - lower and upper bounds
  - states
  - Lagrange multipliers

The optimization model is a Numeric Mapping. Some of the inputs to the Numeric Mapping are used as decision variables, some of the outputs are used as objectives, and some of the outputs are used as constraints. The optimization model serves as the computational engine that computes the values and gradients of the objectives and constraints as functions of the decision variables.

The Optimization Solver identifies the solver to be used along with its specific set of parameters (e.g. tolerances, algorithm parameters, iteration limits, algorithm modes). The solvers that are currently implemented are SNOPT, LSGRG, and SQP.

The mathematical form of the Optimization Formulation is

$$\begin{aligned} \underset{\boldsymbol{u}\in\boldsymbol{U}}{\text{optimize}} \quad & \boldsymbol{c}^T\boldsymbol{z} \\ \text{subject to} \quad & \boldsymbol{u}^L \le \boldsymbol{u} \le \boldsymbol{u}^U \\ & \boldsymbol{z}^L \le \boldsymbol{z} \le \boldsymbol{z}^U \\ & \boldsymbol{z} = \boldsymbol{\Psi}(\boldsymbol{u}) \end{aligned}$$

$\boldsymbol{\Psi}(\boldsymbol{u})$	Numeric Mapping (may have other inputs/outputs)
$\boldsymbol{u}$	variables
$\boldsymbol{z}$	objectives and constraints
$\boldsymbol{c}$	objective linear coefficient
$\boldsymbol{u}^L, \boldsymbol{u}^U$	variable lower and upper bounds
$\boldsymbol{z}^L, \boldsymbol{z}^U$	constraint lower and upper bounds

The components of the optimization formulation are described in Figure 4.

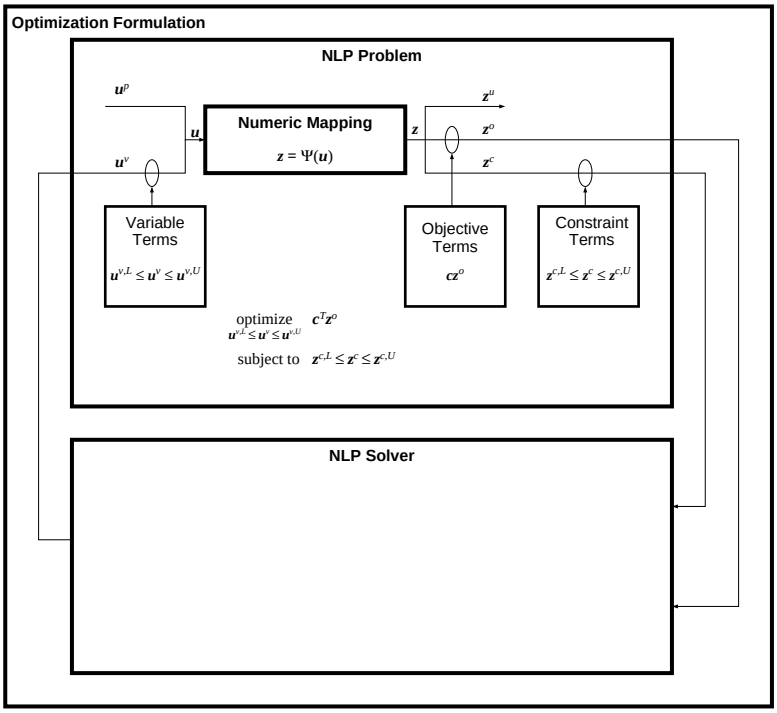

Figure 4: The block diagram for the optimization problem that is used in the constrained training of a PUNDA model. u^v is the decision vector determined by the nonlinear solver (NLP solver).

3.1.5 - Training: A *training problem* is an optimization problem that determines the parameters of a model such that the outputs of the model for a given set of input data closely match the corresponding output data. A training problem is formulated in the optimization framework by creating a harness around the model to be trained where the input data is provided to the model and the corresponding outputs are computed one data point at a time. The model outputs are then used along with the corresponding output data to compute the metrics for the training problem. This creates a nonlinear mapping between the model parameters (weights) and the metrics for the training that establish the quality of the model for given values of the parameters. These metrics typically consist of the sum-square-error objective and any constraints imposed on the training problem.

The optimization model for the training problem is depicted in Figure 4. The modeling framework provides flexibility in the description of the training problem:

1. MLP weights, first-principle model parameters, or both can be determined.

2. Constraints can be written as functions of the data, model outputs, and model parameters.

3. Constraints can have any mathematical form.

4. Constraints can be written over any subset of the data.

5. The training model can be described in a variety of ways.

The data set used for the training is contained within the nonlinear mapping, and the accumulation functions over the data set are explicitly written as functions over that set. The model and objective/constraint expressions are all within the single set of expressions.

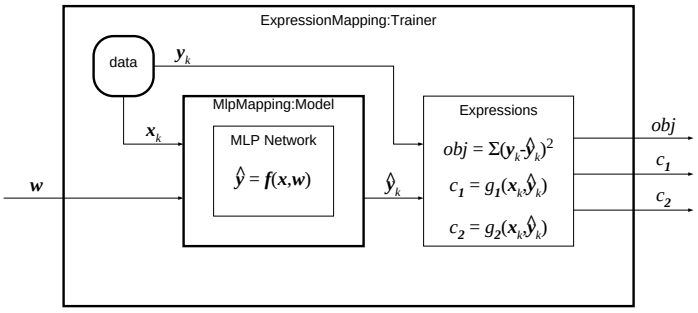

Figure 5: The block diagram for the training of a NN model $y = f(x, w)$ where x is the input vector to the NN model and y is the NN output vector.

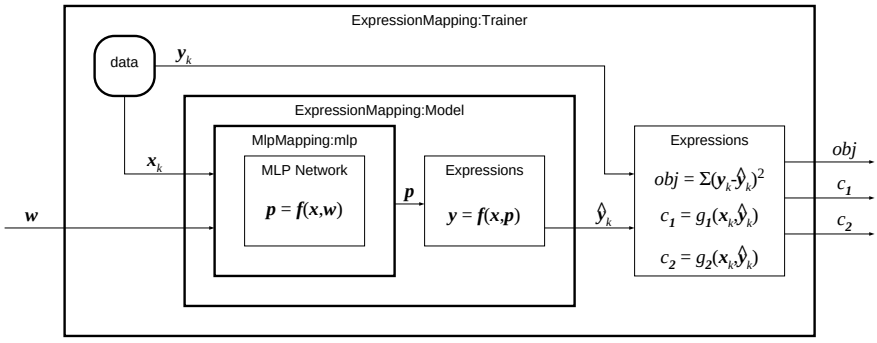

Figure 6: The block diagram for the model used for training of a combined model.

Figure 5 shows the case for the training of a neural network model. Note that in Figure 5, MLP refers to Multi-Layer Perceptron where w is the vector of NN weights and biases to be determined by the NLP solver. Figure 6 shows when the parameters of a parametric nonlinear model, p, are modeled by a NN block in a PUNDA structure (as described in the previous section). Note that in Figure 6:

1. The NN model is defined as $p = f(x, w)$ where x is the input to the NN block, w is the vector of weights and biases for the NN model, and p is the varying parameter of the nonlinear parametric model $y = f(x, p)$.

2. The optimizer does not directly onfluence the output of the NN model, *i.e.* p. Instead, only the output of the parametric nonlinear model, y, is directly measured.

3. The error function the optimization problem will attempt to minimize is formed based on the output of the parametric nonlinear model, $y - \hat{y}$.

4. Appropriate constraints are imposed to ensure the combined model trained by the optimizer is physically meaningful.

3.2 - Nonlinear Dynamic Model Identification using PUNDA Models

To fully investigate the challenges involved in successful training of PUNDA models, for the first year of this project we focused on synthetic data for a set of first and second order dynamic systems with varying dynamics. Furthermore, we only focused on input/output description of the nonlinear dynamic systems, leaving the investigation of state-space models to the second year of this project. We studied the identification problem for both continuous and discrete domain representations of the nonlinear system and developed interesting insight into the complexity of the identification problem.

In what follows we present representative results from our experimentations and highlight the versatility of the proposed PUNDA framework for nonlinear system identification.

3.2.1 - Continuous Domain Representation - First Order Model: We considered the following dynamic system for our study:

$$\tau(\cdot)\frac{dy(t)}{dt} + y(t) = K(\cdot)u(t) \tag{5}$$

where parameters $\tau(\cdot)$ and $K(\cdot)$ vary as a function of process input $u(\cdot)$:

$$\tau(u) = 5\tanh(u(t)) + 0.5 \tag{6}$$

$$K(u) = 5\tanh\left(\frac{u(t)}{4}\right) + 5 \tag{7}$$

The nonlinear system in Eq.(5) was simulated in Matlab where system response to variations in input signal $u(t)$ was recorded.

To identify the varying parameters of this nonlinear model, the following PUNDA structure was constructed:

1. The SNM block in Figure 26 is a neural network block with $u(\cdot)$ as input, and the parameters $\tau(\cdot)$ and $k(\cdot)$ as its outputs.

2. The differential equation described by Eq. (5) constitutes the PNM block in Figure 26. In addition to system input $u(\cdot)$, the PNM block receives varying parameters $\tau(\cdot)$ and $k(\cdot)$ from the NN block, and produces $y(\cdot)$ as its output.

Typical results for the training of the PUNDA model are shown in Figures 7 and 8. Note that the richness of the input signal (and the absence of noise in measurements) has resulted in perfect identification of varying parameters and accurate predication capability for the PUNDA model. Nevertheless, this underscores the versatility of the PUNDA structure for the identification of varying parameters in a nonlinear dynamic system.

3.2.2 - Continuous Domain Representation - Second Order Model: We considered the following dynamic system for our study of second order parametric models:

$$(\tau_1(\cdot)\tau_2(\cdot))\frac{d^2y(t)}{dt^2} + (\tau_1(\cdot) + \tau_2(\cdot))\frac{dy(t)}{dt} + y(t) = K(\cdot)u(t) \tag{8}$$

where parameters $\tau_1(\cdot)$, $\tau_2(\cdot)$, and $K(\cdot)$ vary as a function of process input $u(\cdot)$:

$$\tau_1(u) = 5\tanh(u(t)) + 0.5 \tag{9}$$

$$\tau_2(u) = 8\tanh(\exp(-u(t))) \tag{10}$$

$$K(u) = 5\tanh\left(\frac{u(t)}{4}\right) \tag{11}$$

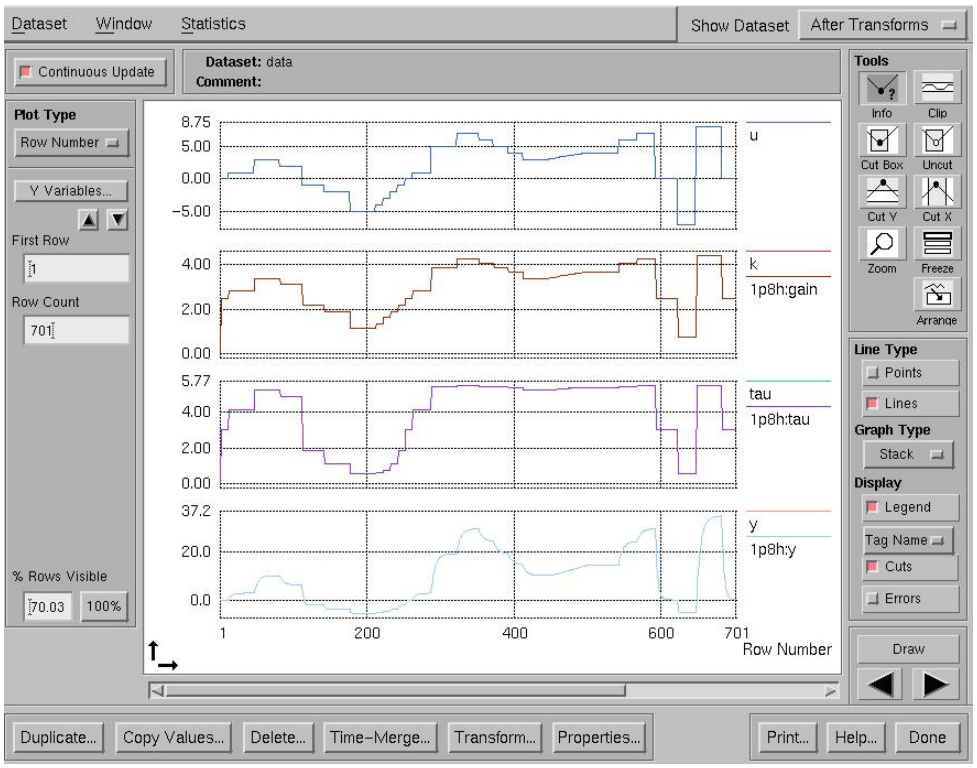

Figure 7: Parameter and output prediction with a PUNDA model for a first order overdamped system of Eq. (5).

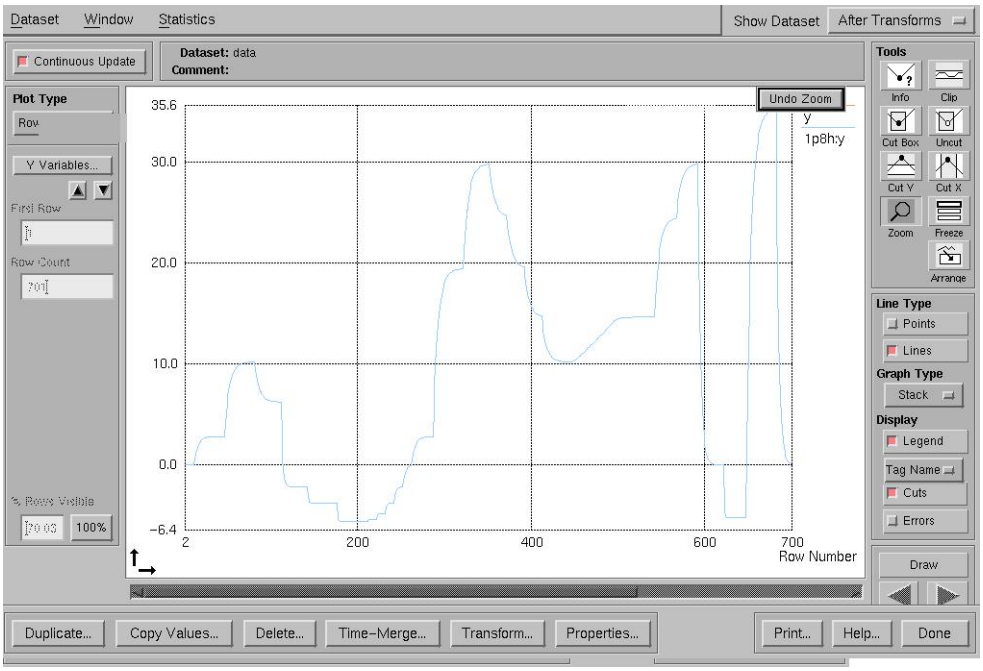

Figure 8: A closeup look at the output prediction with the identified model of Figure 7.

The nonlinear system in Eq.(8) was simulated in Matlab where system response to variations in input signal $u(t)$ was recorded.

To identify the varying parameters of this nonlinear model, the following PUNDA structure was constructed:

1. The SNM block in Figure 26 is a neural network block with $u(\cdot)$ as input, and the parameters $\tau_1(\cdot)$, $\tau_2(\cdot)$, and $k(\cdot)$ as its outputs.

2. The differential equation described by Eq. (8) constitutes the PNM block in Figure 26. In addition to system input $u(\cdot)$, the PNM block receives varying parameters $\tau_1(\cdot)$, $\tau_2(\cdot)$, and $k(\cdot)$ from the NN block, and produces $y(\cdot)$ as its output.

Typical results for the training of the PUNDA model are shown in Figures 9 and 10. Note that not only has the PUNDA model been able to reproduce process output $y(\cdot)$, it has been able to correctly identify variations in gain and process residence times. A closer look at parameter identification is provided in Figure 10 where the predicted values for $\tau_1(\cdot)$ is compared against the actual value.

3.2.3 - Discrete Domain Representation - Second Order Model: In this section a typical result for the identification of varying parameters in a discrete domain representation of a dynamic system is presented. The discrete model for a second order system may be described as:

$$y(k) = -a_1(\cdot)y(k-1) - a_2(\cdot)y(k-2) + b_1(\cdot)u(k-1-D) + b_2(\cdot)u(k-2-D) \tag{12}$$

where D is the delay and varying parameters $a_{1,2}/b_{1,2}$ are related to the continuous domain parameters of Eq. (8) as follows:

$$a_1(\cdot) = -\left(e^{-\frac{T}{\tau_1}} + e^{-\frac{T}{\tau_2}}\right) \tag{13}$$

$$a_2(\cdot) = +\left(e^{-\left(\frac{T}{\tau_1}+\frac{T}{\tau_2}\right)}\right) \tag{14}$$

$$b_1(\cdot) = +\left(A\left(1-e^{-\frac{T}{\tau_1}}\right) + B\left(1-e^{-\frac{T}{\tau_2}}\right)\right) \tag{15}$$

$$b_2(\cdot) = -\left(Ae^{-\frac{T}{\tau_2}}\left(1-e^{-\frac{T}{\tau_1}}\right) + Be^{-\frac{T}{\tau_1}}\left(1-e^{-\frac{T}{\tau_2}}\right)\right) \tag{16}$$

where $A = k\frac{\tau_1}{\tau_1-\tau_2}$ and $B = k\frac{-\tau_2}{\tau_1-\tau_2}$.

For this example, the continuous domain parameters of the dynamic system are assumed to vary as follows:

$$\tau_1(u) = 5\tanh(u(t)) + 0.5 \tag{17}$$

$$\tau_2(u) = -8\tanh(u^2(t) - 2u(t)) + 8.5 \tag{18}$$

$$K(u) = 5\tanh\left(\frac{u(t)}{4}\right) \tag{19}$$

Typical results are shown in Figures 11 and 12. As Figure 11 indicates, the PUNDA modelling framework offers the ability to identify $a_{1,2}/b_{1,2}$ with acceptable accuracy. The stability of the discrete model is used as a constraint for the training of the discrete model. There is a good agreement between the identified and actual values for $a_{1,2}/b_{1,2}$, with system output perfectly modelled.

Parameter identification for the discrete model proves to be the most challenging in the examples we have tested. Therefore, it offers a more challenging platform to examine the effect of the measurement

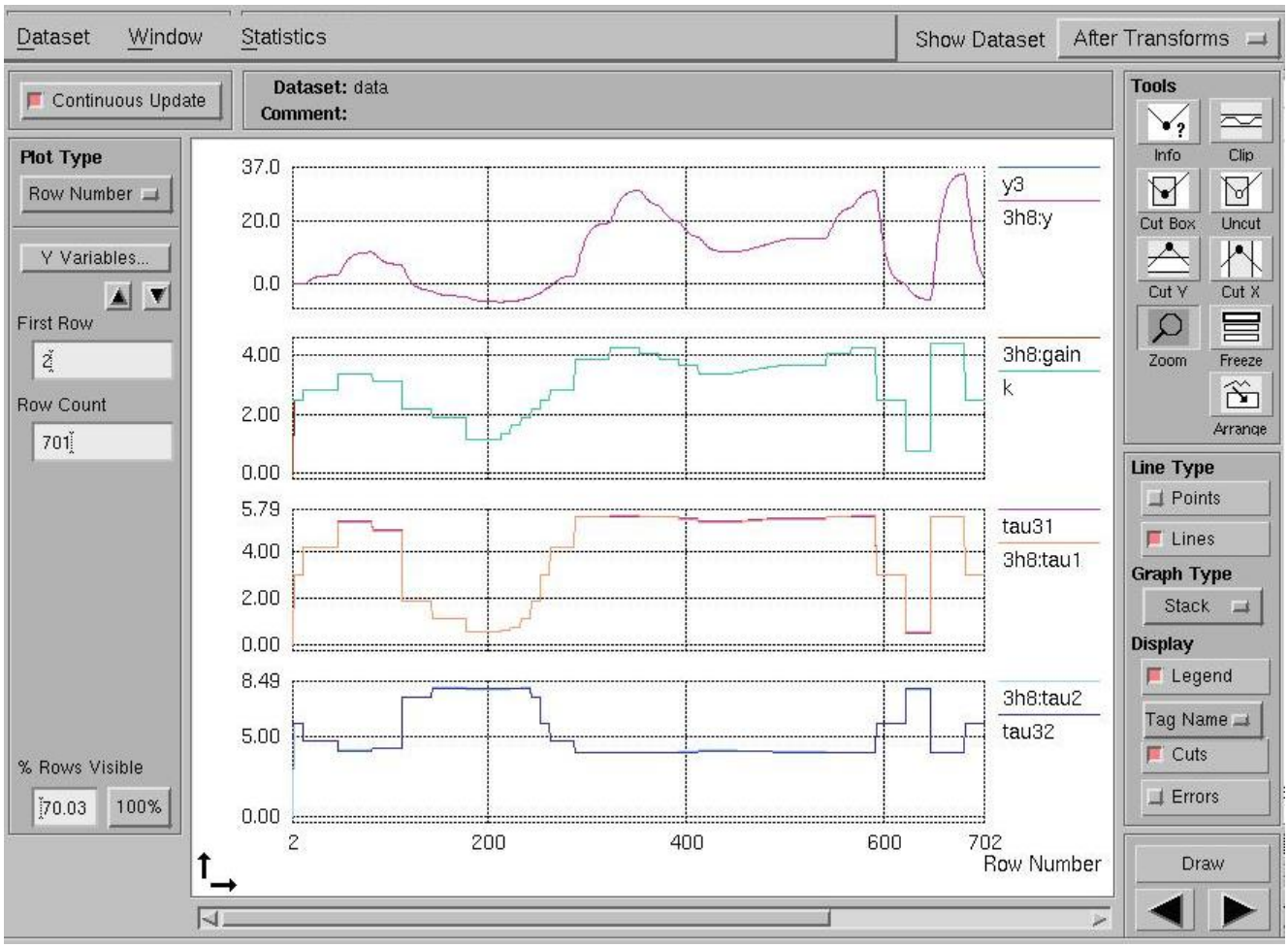

Figure 9: Parameter and output prediction with a PUNDA model for the second order overdamped system of Eq. (8).

noise in parameter identification. Figure 13 demonstrates that the presence of measurement noise results in increased error in the identified parameters, even though overall quality of parameter ID is extremely desirable. Figure 14 highlights the fact that an increase in the noise level will result in reduced accuracy of the parameter ID. Nevertheless, even with a noise with three times the standard deviation of the measured output signal the parameter ID is still acceptable. We plan to study the methodologies by which the effect of noisy measurement signals may be mitigated.

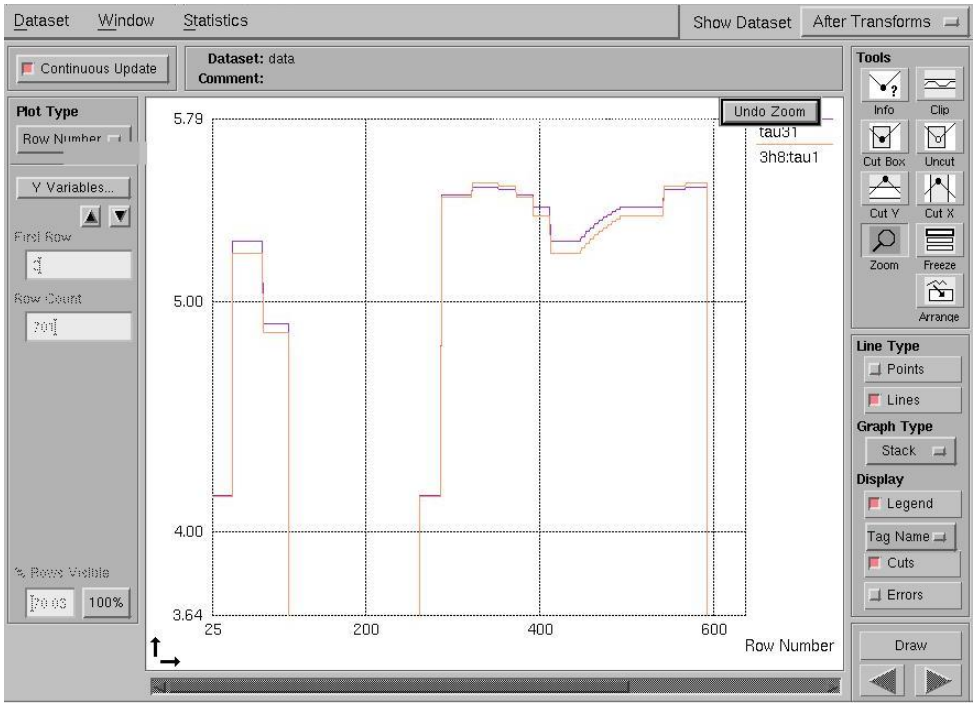

Figure 10: A closeup look at the identified $\tau_1(\cdot)$ in Figure 9 by the NN block in the PUNDA structure.

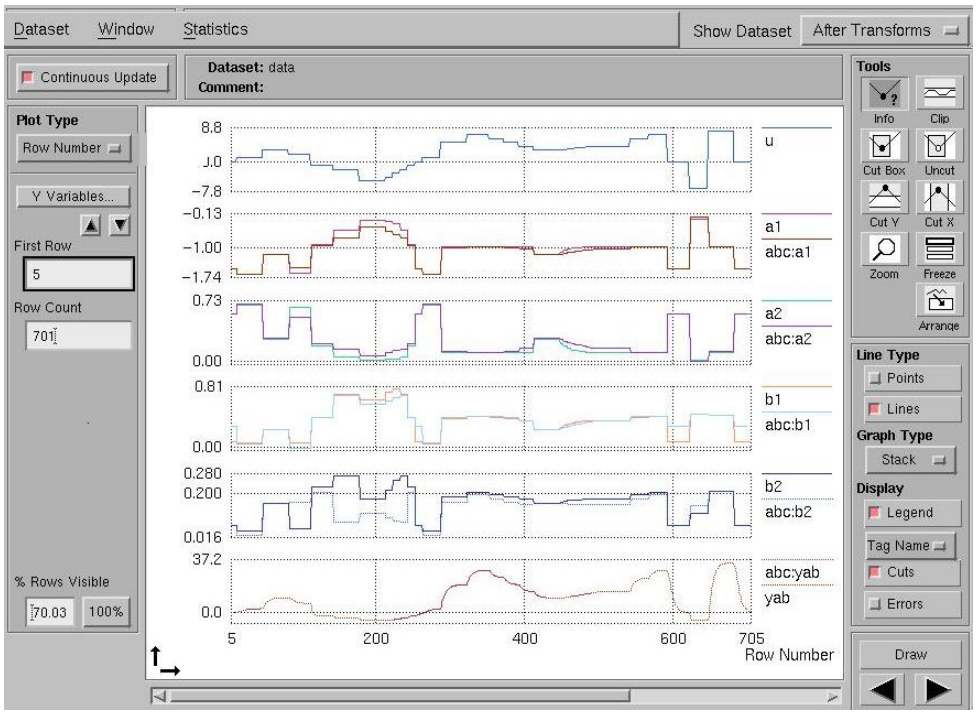

Figure 11: Parameter and output prediction with a PUNDA model for the second order overdamped system of Eq. (12).

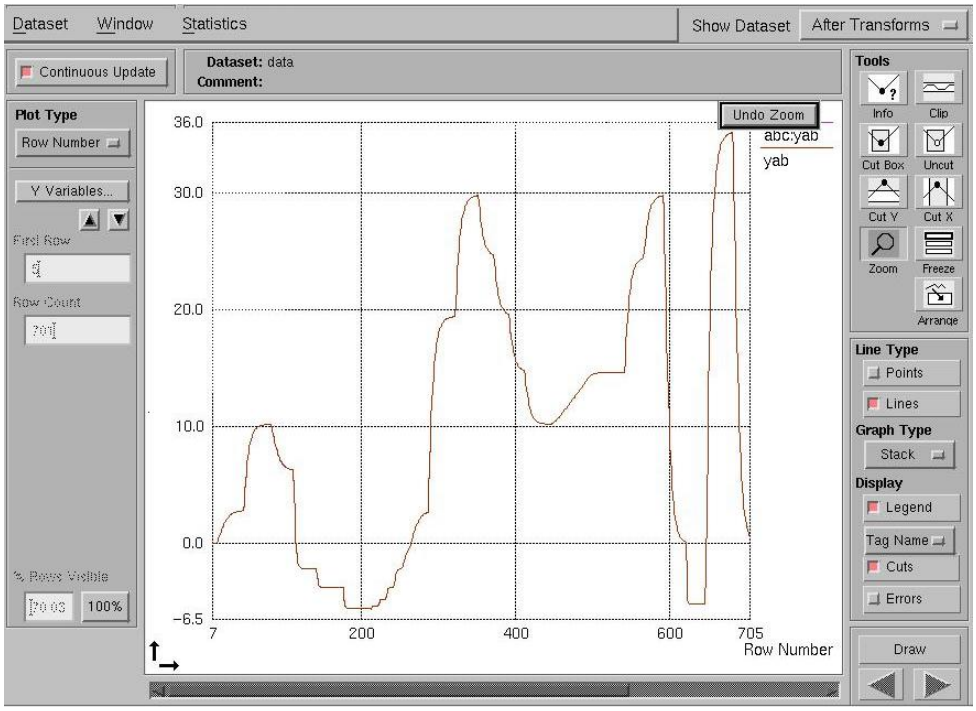

Figure 12: A closeup look at the output prediction in Figure 11 by the NN block in the PUNDA structure.

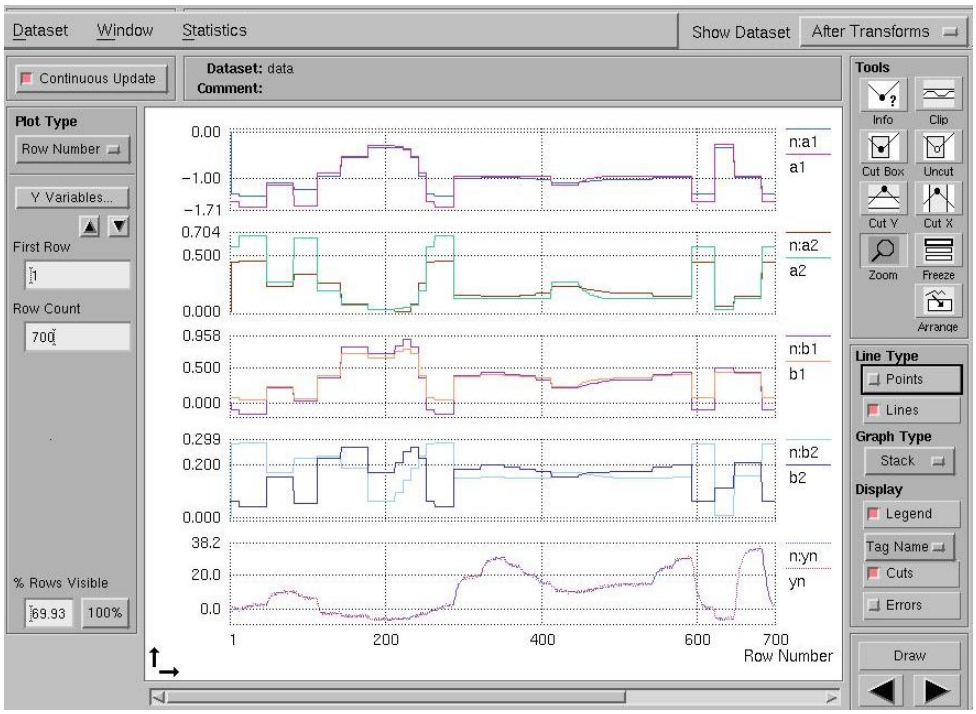

Figure 13: Parameter and output prediction with a PUNDA model for the second order overdamped system of Eq. (12) when the output measurement was subject to measurement noise. The noise in this case was 0.1 times the standard deviation of the output signal.

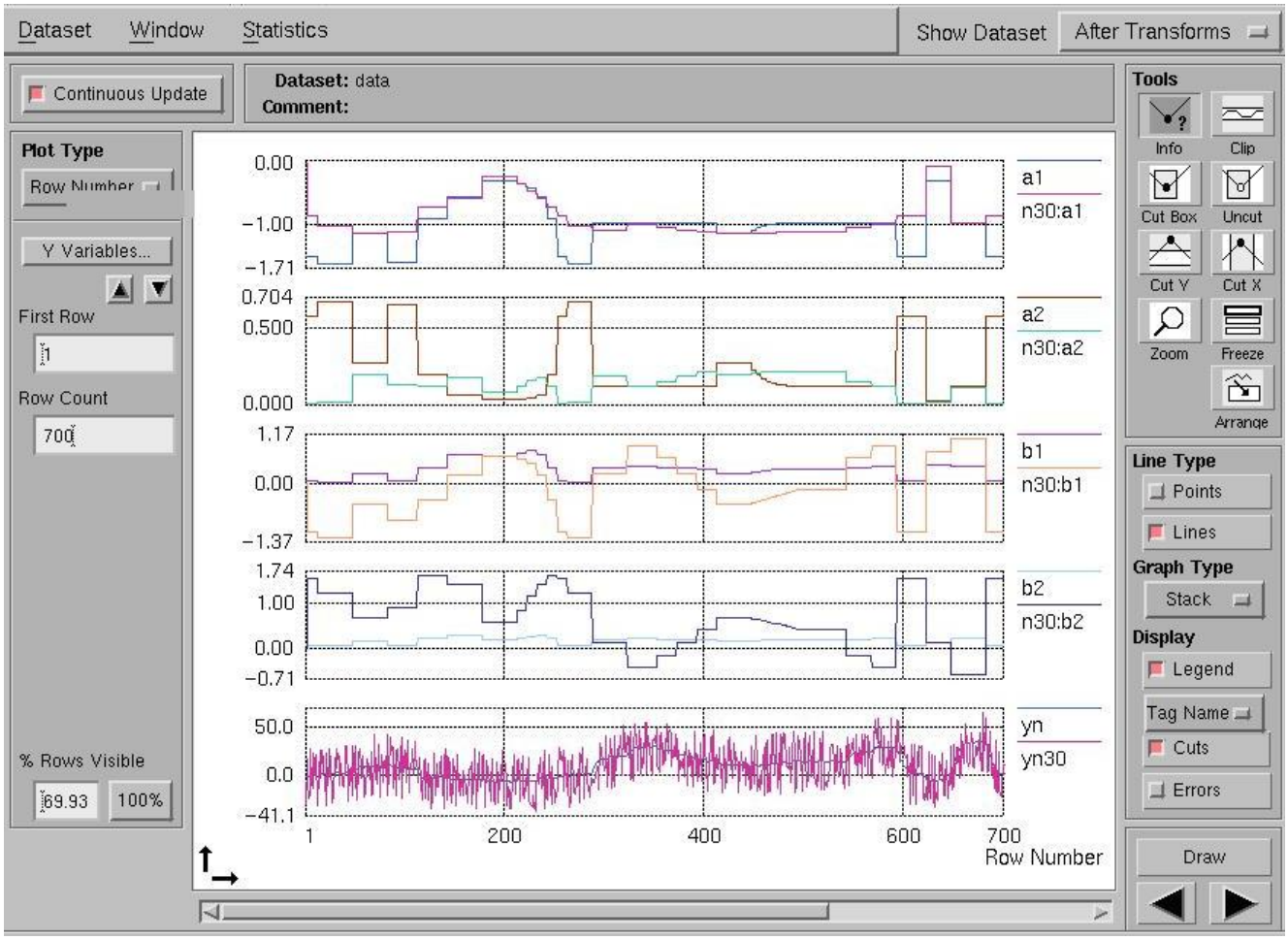

Figure 14: Parameter and output prediction with a PUNDA model for the second order overdamped system of Eq. (12) when the output measurement was subject to measurement noise. The noise in this case was 3.0 times the standard deviation of the output signal.

3.3 - Beam Matrix Identification using PUNDA Models

The Linac Coherent Light Source (LCLS) at SLAC requires the RF photoinjector to consistently produce a high-brightness electron beam (*i.e.* an electron beam with high current and low emittance). Free Electron Laser (FEL) applications have been the primary driving force behind this requirement. To achieve the desired beam quality, there have been systematic studies of both the transverse and longitudinal beam properties from the RF photocathode gun at SLAC Gun Test Facility (GTF) [13, 14]. One main objective of these studies has been to identify the factors affecting beam quality and to establish systematic methodologies for minimizing beam emittance. At the moment however, reliable models that are suitable for emittance minimization are not fully developed [15, 13].

The schematic diagram for longitudinal and transverse beam measurements at SLAC GTF[11] is shown in Figure 15. The measurement technique, known as phase/quadrupole scans [14], measures the energy spectra/size of the beam at the exit of the linac section as a function of linac phase. The purpose of the analysis is to identify the longitudinal/transverse beam matrix at the exit of the linac. It is noted that these are not the same as those at the gun exit, because of continued evolution of the bunch as it drifts from the gun to the linac [13, 16].

One objective of this Phase II STTR project is to develop accurate and computationally efficient models for electron beam such that systematic optimization of beam properties (*e.g.* beam emittance) is feasible. Our first year effort has focused on building PUNDA models for transverse properties of the beam and to examine the applicability of PUNDA modelling framework for systematic optimization scenarios.

3.3.1 - GTF Transport Matrix from Linac Exit to Spectrometer: In this section, the transport matrix for transverse motion of the beam from the exit of Linac to spectrometer is provided briefly, as we have used the transport matrix between Linac exit and spectrometer as the first-principles model for the training of the PUNDA models[12]. A detailed description of the transport matrices for various components in a beamline may be found in [17].

1. The transport matrix for the first quadrupole magnet (vertical quad) is defined as follows:

$$T_1 = \begin{bmatrix} \cosh(k_1 l_{eff}) & \frac{1}{k_1}\sinh(k_1 l_{eff}) \\ k_1 \sinh(k_1 l_{eff}) & \cosh(k_1 l_{eff}) \end{bmatrix} \tag{20}$$

   where $k_1 = 1.27$ ($m^{(-1)}$) is the strength of the magnetic field for the vertical quad (which is kept constant for all measurements), and $l_{eff} = 0.077$ (m) is the effective length of the quad.

2. The transport matrix for the drift space between two quads is defined as follows:

$$T_2 = \begin{bmatrix} 1 & l_{qtq} \\ 0 & 1 \end{bmatrix} \tag{21}$$

   where $l_{qtq} = 0.110(m)$ is the effective length between the two quads.

3. The transport matrix for the second quadrupole magnet (horizontal quad) is defined as follows:

$$T_3 = \begin{bmatrix} \cos(k_2 l_{eff}) & \frac{1}{k_2}\sin(k_2 l_{eff}) \\ -k_2 \sin(k_2 l_{eff}) & \cos(k_2 l_{eff}) \end{bmatrix} \tag{22}$$

   where $k_2(I_2) = \left(\frac{167.23 I_2 + 43.17}{30.50}\right)^{0.5}$ ($m^{(-1)}$) is the strength of the magnetic field described as a nonlinear function of quad current, and $l_{eff} = 0.077$ (m) is the effective length of the quad.

[11] The SLAC GTF beamline consists of a 1.6 cell S-band gun of the BNL/SLAC/UCLA design followed by a 3 meter linac section. The drive laser is a Nd:Glass CPA laser with a regenerative amplifier that provides 2 ps (fwhm) gaussian UV pulses to the cathode with an approximately uniform, 2 mm transverse profile.

[12] The transport matrices and the parameters for these matrices were provided by Dr. J. Schmerge.

4. The transport matrix for the drift space between the second quad and the spectrometer is as follows:

$$T_4 = \begin{bmatrix} 1 & l_{d1} \\ 0 & 1 \end{bmatrix} \tag{23}$$

where $l_{d1} = 1.835(m)$ is the effective drift length between the quad and the spectrometer section.

5. The transport matrix for the spectrometer section is:

$$T_5 = \begin{bmatrix} A_{spec} & B_{spec} \\ C_{spec} & D_{spec} \end{bmatrix} \tag{24}$$

where:

$$A_{spec} = 1 - (\alpha + \Delta\alpha)\tan(\psi_{ent}) \tag{25}$$

$$B_{spec} = \rho\frac{E_{slice}}{E_{proj}}(\alpha + \Delta\alpha) \tag{26}$$

$$C_{spec} = \frac{1}{\rho\frac{E_{slice}}{E_{proj}}}(\alpha+\Delta\alpha)\tan(\psi_{ent})(\tan(\psi_{exit})+\Delta\alpha)-\tan(\psi_{ent})-\tan(\psi_{exit})-\Delta\alpha \tag{27}$$

$$D_{spec} = 1-(\alpha+\Delta\alpha)(\tan(\psi_{exit})+\Delta\alpha) \tag{28}$$

and $\alpha = \frac{\pi}{3}$, $\rho = 0.400$ (m), $\psi_{ent} = 2.53 \times \frac{\pi}{180}$, $\psi_{exit} = 13.49 \times \frac{\pi}{180}$, $E_{slice} = 30.50$ (Mev), $E_{proj} = 30.50$ (Mev), and $\Delta\alpha = \alpha\left(\frac{E_{slice}}{E_{proj}} - 1\right)$.

6. The transport matrix for the drift space from spectrometer bend to the screen is as follows:

$$T_6 = \begin{bmatrix} 1 & l_{d2} \\ 0 & 1 \end{bmatrix} \tag{29}$$

where $l_{d2} = 0.419(m)$.

With the transport matrices for various components of the beam line well defined, the overall transport matrix between the exit of the linac section and the spectrometer screen may be defined as follows:

1. The transfer matrix for the quad section is as follows:

$$T_{quad} = \begin{bmatrix} A_{quad} & B_{quad} \\ C_{quad} & D_{quad} \end{bmatrix} = T_3 \times T_2 \times T_1 \tag{30}$$

2. With the transfer matrix for the quad section well defined the overall transfer matrix between the linac exit and the spectrometer screen may be defined as follows:

$$T = \begin{bmatrix} A_T & B_T \\ C_T & D_T \end{bmatrix} \tag{31}$$

where

$$A_T = A_{spec}\left[A_{quad} + l_{d1}C_{quad}\right] + B_{spec}C_{quad} + \cdots \\ l_{d2}C_{spec}\left[A_{quad} + l_{d1}C_{quad}\right] + l_{d2}D_{spec}C_{quad} \tag{32}$$

$$B_T = A_{spec}\left[B_{quad} + l_{d1}D_{quad}\right] + B_{spec}D_{quad} + \cdots \\ l_{d2}C_{spec}\left[B_{quad} + l_{d1}D_{quad}\right] + l_{d2}D_{spec}D_{quad} \tag{33}$$

$$C_T = C_{spec}\left[A_{quad} + l_{d1}C_{quad}\right] + D_{spec}C_{quad} \tag{34}$$

$$D_T = C_{spec}\left[B_{quad} + l_{d1}D_{quad}\right] + D_{spec}D_{quad} \tag{35}$$

	σ^e_{11}	σ^e_{12}	σ^e_{22}	Objective Function
Known A_T&B_T	2.62035	1.55237	1.40828	9.8272985503E+01
Identified A_T&B_T	2.699999E+00	1.400000E+00	1.600000E+00	8.5382085177E+00
Identified α&β	3.484610E+00	1.857201E+00	1.473184E+00	9.8027962242E+01

Table 2: Optimization results for different scenarios.

With the A_T matrix in hand, the beam matrix at the exit of the linac section may be related to the beam matrix at the spectrometer screen as follows:

$$\begin{bmatrix} \sigma^s_{11} & \sigma^s_{12} \\ \sigma^s_{21} & \sigma^s_{22} \end{bmatrix} = \begin{bmatrix} A_T & B_T \\ C_T & D_T \end{bmatrix} \begin{bmatrix} \sigma^e_{11} & \sigma^e_{12} \\ \sigma^e_{21} & \sigma^e_{22} \end{bmatrix} \begin{bmatrix} A_T & B_T \\ C_T & D_T \end{bmatrix}^{(T)} \tag{36}$$

where the superscript 's' indicates the beam matrix parameters at the spectrometer screen while the superscript 'e' indicates the beam matrix parameters at the exit of the linac.

3.3.2 - Identifying Beam Matrix Using PUNDA Model[13]: For the results presented here, we applied the PUNDA models shown in Figures 16-18. Representative experimentations carried out during first year of this STTR project may be summarized as follows:

1. For our first test, we used the first principles models of Eqs. (32)-(35) for the *"F.P. for Transport Matrix"* block in Figure 16. The constrained optimization problem sought the optimal value of the beam matrix, *i.e.* σ^e_{11}, σ^e_{12}, and σ^e_{22} given the measurements of the horizontal beam size at spectrometer screen, $(\Delta X)^2$. The xml file developed for this optimization problem is provided in Appendix 1. The beam matrix parameters and the sum-squared error in predicting the beam size at the spectrometer screen are reflected in Table 2. Figure 19 shows the measured beam size versus the modelled beam size. Our results are consistent with the results reported in [18] by SLAC scientists.

2. For the second test, we used a NN model to estimate A_T and B_T as functions of quad current I_2, at the same time that we identified the beam matrix at the exit of the linac (see Figure 17). Note that the first principles model relates the beam size at the spectrometer screen to the beam size at the exit of the linac as follows (see Eq. (36) for full mapping of the beam matrix):

$$(\Delta X)^2 = A_T^2\sigma^e_{11} + 2A_TB_T\sigma^e_{12} + B_T^2\sigma^e_{22} \tag{37}$$

The beam matrix parameters and the sum-squared error in predicting the beam size at the spectrometer screen are reflected in the second row in Table 2. Figure 20 compares the PUNDA model prediction of beam size against the measured beam size. Figures 21 and 22 compare the identified A_T and B_T elements of the transport matrix to those computed from first principles models. It is important to point out that:

   (a) The results presented here are obtained using the following constraints, for $I_2 \in [0, \cdots, 3.9]$, extracted from FP model for beam matrix:

$$\frac{\partial A_T}{\partial I_2} \leq 0 \tag{38}$$

$$\frac{\partial B_T}{\partial I_2} \leq 0 \tag{39}$$

   and no other information from FP model is used for the training[14].

[13]The details of data acquisition for the data used in this section is outlined in [18].

[14]We will investigate the role of more FP information in model quality during second year of the project.

(b) Only 14 data points were available for the training of the PUNDA model. We anticipate more data will improve the quality of the model.

(c) This PUNDA model has produced a better fit to the beam size measurements compared to that in Figure 19.

3. For the third test, we used the first principles models of Eqs. (32)-(35) for the *"F.P. for Transport Matrix"* block in Figure 18, and we used the constrained optimization to identify the parameters of the I_2 to K_2 mapping at the same time that we identified the parameters of the beam matrix. We assumed (based on input from SLAC scientists), the following relationship between $I_2 - k_2$:

$$k_2(I_2) = \left(\frac{\alpha I_2 + \beta}{30.50}\right)^{0.5} \tag{40}$$

The xml file for this optimization problem is provided in Appendix C. The optimization yields $\alpha = 147.230$ and $\beta = 63.170$[15] with the optimal value for beam matrix and the error function reflected in Table 2. Figures 23-25 demonstrate the results of the optimization for this case.

(a) With only 14 data points available further improvements in PUNDA model quality could only be achieved if the optimization problem is properly constrained with more FP information.

(b) This example demonstrates the possibility of using a PUNDA structure to identify unknown/uncertain parameters in a FP model at the same time that beam matrix parameters are determined given beam size measurements.

4. We also attempted to model $I_2 - K_2$ mapping with a NN. This is an example of using PUNDA models to identify an unknown nonlinear mapping in the FP models (the potential hysteresis effect in this case). The available data, however, was not sufficient to simultaneously produce a meaningful model for $I_2 - K_2$ mapping (*i.e.* identify NN weights and biases for this mapping)and identify beam matrix parameters. We plan to obtain more data from SLAC for this purpose.

We would like to conclude this section with two main observations regarding the applicability of the PUNDA models in GTF applications:

1. PUNDA structure offers a framework in which both beam data and first principles models may be used to complement one another. Furthermore, the measured data may be used to fine tune the FP model.

2. A trained PUNDA model may be systematically used to find optimal operation conditions for the GTF. For our first test case, for example, the PUNDA model may be used to find the optimal quad current for which a desired $(\Delta X, \Delta X')$ may be achieved. The xml file for doing so is provided in Appendix D.

[15] In the original first principles model provided by SLAC scientists $\alpha = 167.23$ and $\beta = 43.17$.

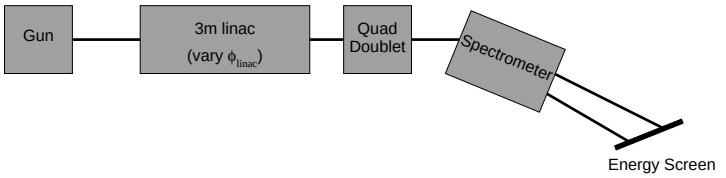

Fig. 15: The block diagram for longitudinal and transverse beam measurements at SLAC GTF. A full description of the beamline is provided in [18].

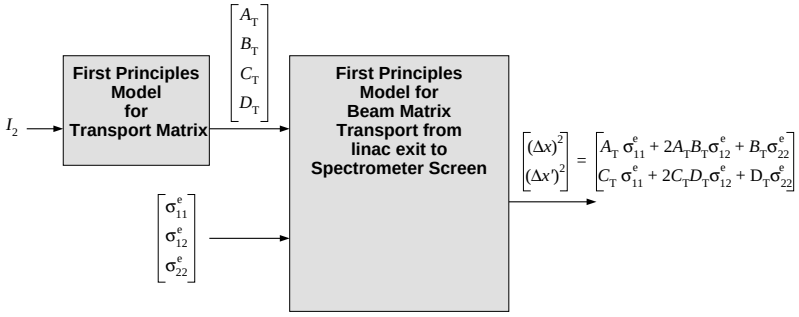

Fig. 16: Block diagram representation of the PUNDA model that is used for beam matrix identification with full FP model available.

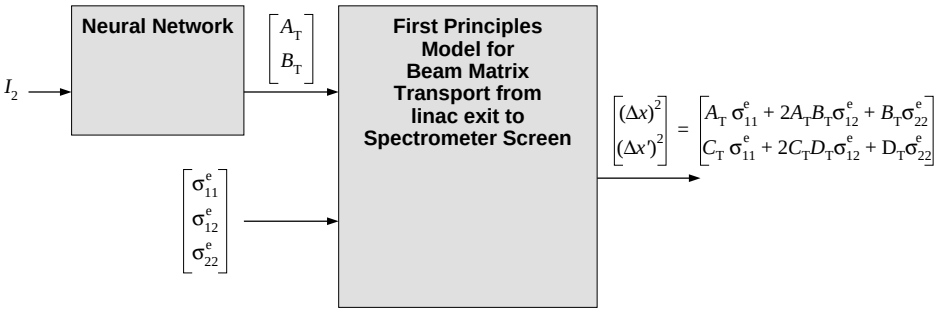

Fig. 17: Block diagram representation of the PUNDA model that is used for simultaneous identification of beam matrix and transport matrix elements A_T and B_T. Note that we have used a NN model to capture the nonlinear mapping between I_2 and the transport matrix parameters.

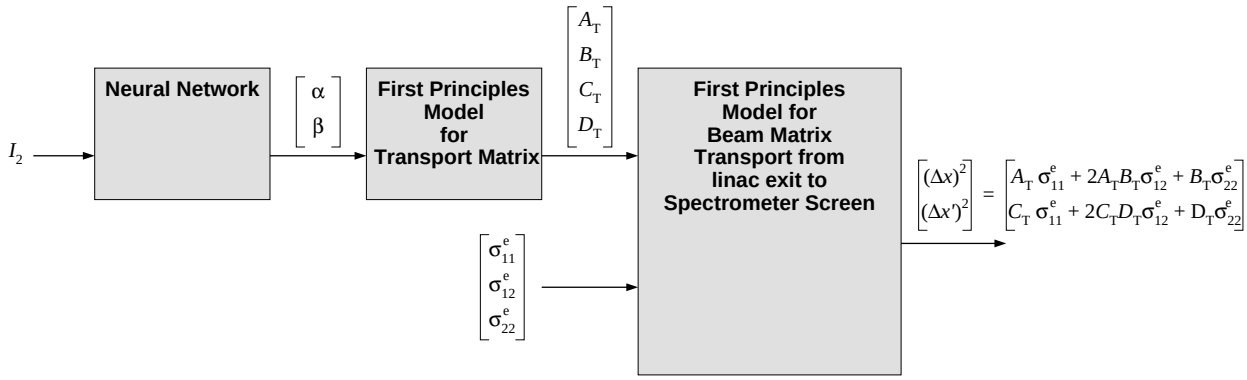

Fig. 18: Block diagram representation of the PUNDA model that is used for simultaneous identification of beam matrix and the $I_2 - K_2$ mapping parameters $\alpha\&\beta$. Note that for the results reported in case 3, Figures 23-25, the NN block in this figure is a fixed identity mapping.

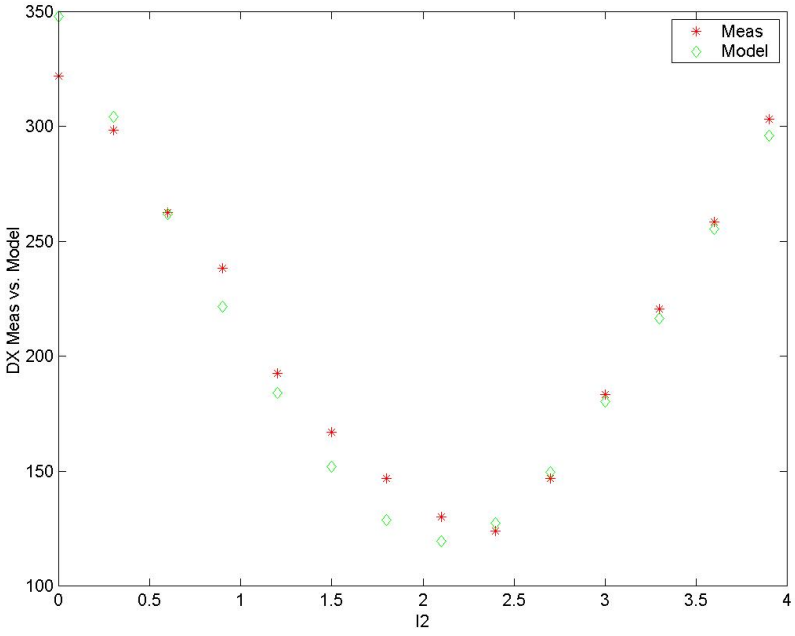

Fig. 19: Comparison of measured beam size vs. the beam size predicted by the PUNDA model.

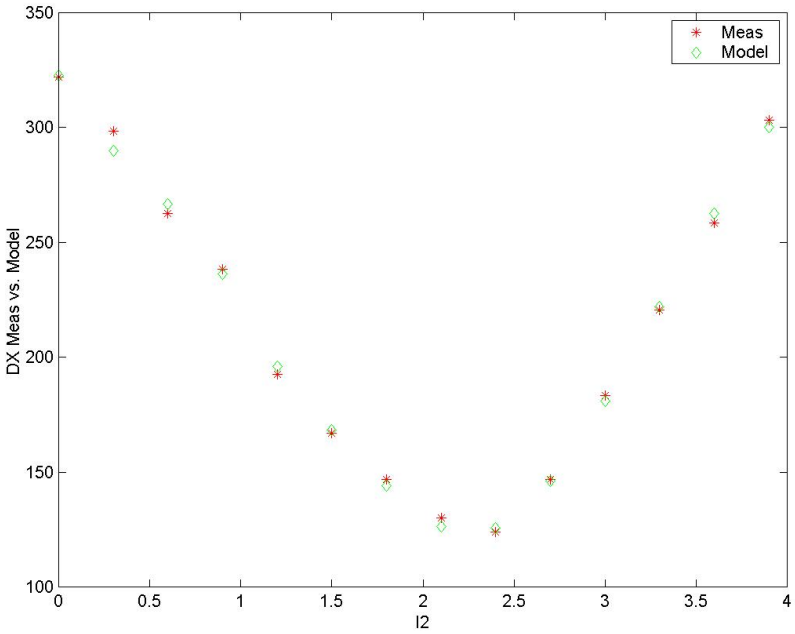

Fig. 20: Comparison of measured beam size vs. the beam size predicted by the PUNDA model.

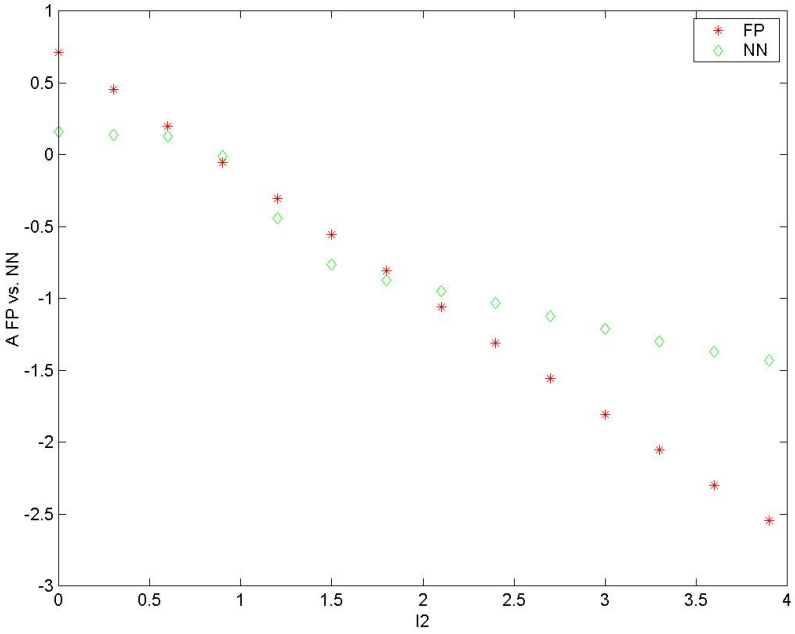

Fig. 21: Comparison of identified values for A_T vs. the A_T values generated by the FP model.

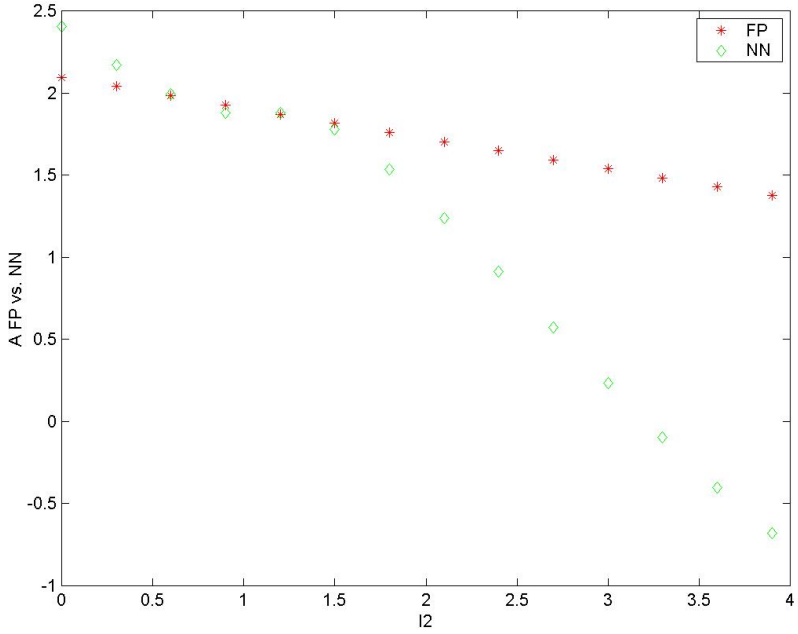

Fig. 22: Comparison of identified values for B_T vs. the B_T values generated by the FP model.

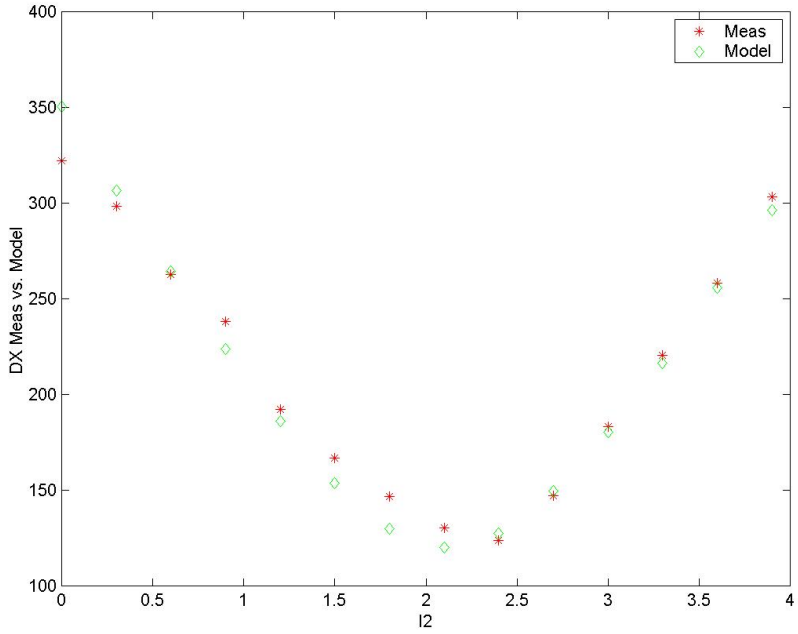

Fig. 23: Comparison of measured beam size vs. the beam size predicted by the PUNDA model for the case where beam matrix parameters and α and β in $I_2 - K_2$ mapping are identified simultaneously.

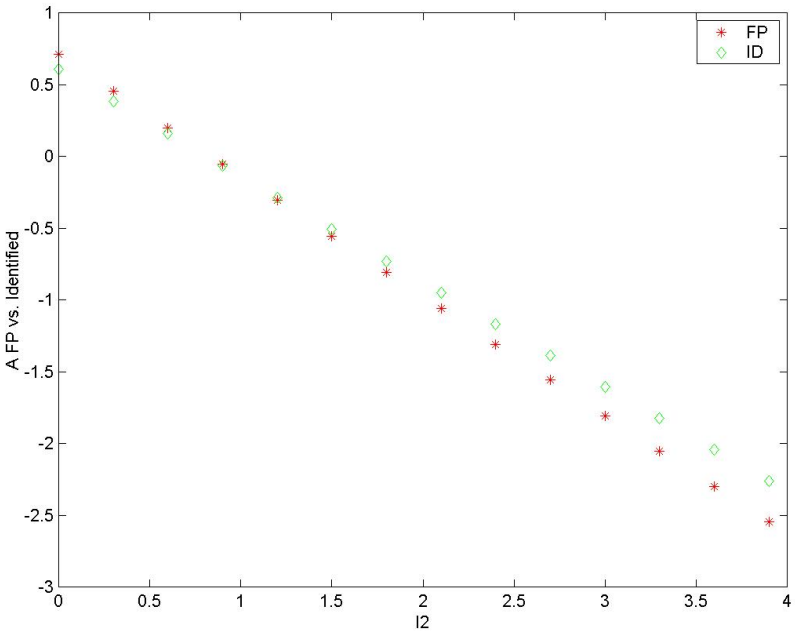

Fig. 24: Comparison of identified values for A_T vs. the A_T values generated by the FP model for the case discussed in Figure 23.

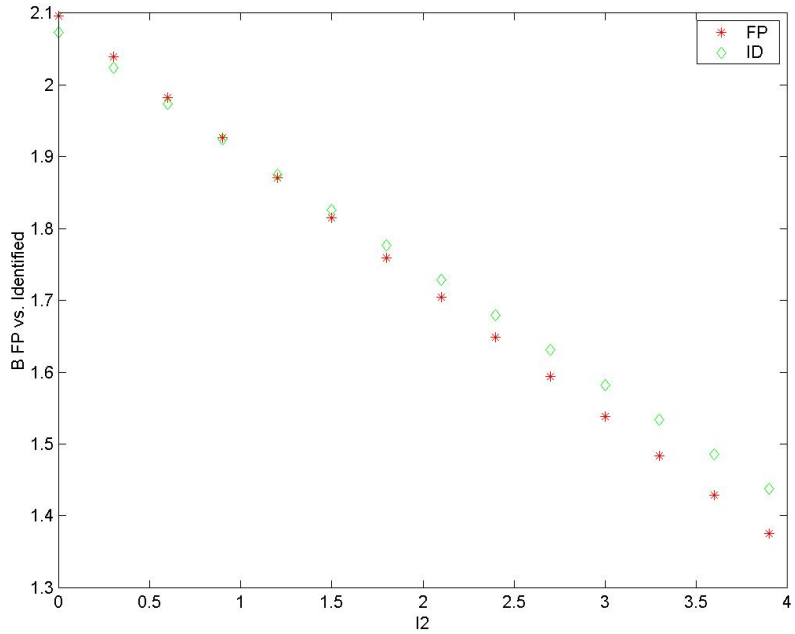

Fig. 25: Comparison of identified values for B_T vs. the B_T values generated by the FP model for the case discussed in Figure 23.

3.4 - Beam Lifetime Study using PUNDA Models

Synchrotron light is used for a wide variety of scientific disciplines ranging from physical chemistry to molecular biology and industrial applications. The synchrotron light is radiated from a relativistic electron beam circulating in a storage ring particle accelerator [19, 20, 21]. As the electron beam circulates, random single-particle collisional processes lead to decay of the beam current in time. As the electron beam decays, so does the intensity of the synchrotron light resulting in detuned optics in the photon transport lines (*e.g.* mirrors, gratings, slits), changes in material properties of the experimental sample, degradation of detector performance and uncertainties in data reduction. Hence, at all synchrotrons, a premium is placed on delivering constant photon beam intensity to the photon beam lines [22].

Efforts to systematically model the electron beam loss in synchrotron light sources have therefore been of primary interest. At SPEAR3, for example, every two weeks, up to 48 hrs of beam time is allocated for machine development studies, including programs to measure, characterize, and mitigate electron beam loss. In this section a framework for the systematic modeling of electron beam loss is presented. This framework, known as **P**arametric **U**niversal **N**onlinear **D**ynamics **A**pproximator (PUNDA), consists of a series connection of a Nonlinear Empirical Model (NEM) block and a Parametric First-principles Model (PFM) block (see Figure 26). The parameters, $\vec{P}$, in the PFM block may vary as a function of process inputs, $\vec{u}$. For the beam loss model of interest in this study, the instantaneous electron beam current, Eq. (41), constitutes the PFM block, where the characteristic beam decay time constant is the varying parameter. A neural network (NN) model forms the NEM block. This NN model is trained to capture the variation in the characteristic beam decay time constant as a function of variation in vertical scraper position (Y_s), RF voltage (V_{rf}), initial beam current (I_0), and total number of bunches (M_b).

The ultimate goal of this study is to build accurate and computationally efficient models that quantify beam loss from electron-gas scattering (elastic and inelastic) and intrabeam scattering (electron-electron). Once such models are constructed, an optimization-based approach may be adopted to: *(a)* minimize beam loss in standard modes of operation, *(b)* optimize synchrotron performance based on forecasts of beam loss, *(c)* provide design guidelines for performance at high beam current and top-up mode of operation, and *(d)* provide design guidelines for installation of future small-gap undulators.

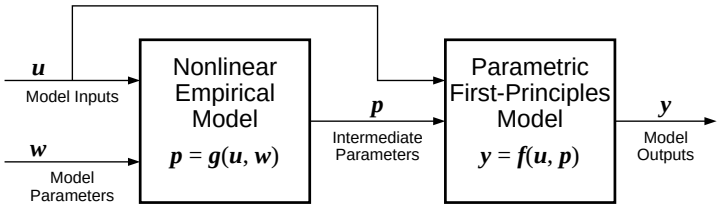

Figure 26: Block diagram of a PUNDA model. The PUNDA model is formed by the series connection of a *Nonlinear Empirical Model* (NEM) block and a *Parametric First-principles Model* (PFM) block.

One objective of this Phase II STTR project is to develop accurate and computationally efficient models for electron beam such that systematic optimization of beam properties (*e.g.* beam emittance) is feasible. Our first year effort has focused on building PUNDA models for transverse properties of the beam and to examine the applicability of PUNDA modelling framework for systematic optimization scenarios.

3.4.1 - Physics of Electron Beam Loss: The physics behind electron beam loss is conceptually straight-forward but nevertheless a highly non-linear process. In principle, the electron beam current decays in time due to (a) elastic electron-gas collisions (Coulomb scattering) (b) inelastic electron-gas collisions (Bremsstrahlung scattering), and (c) intrabeam electron-electron collisions [23, 24, 25, 26]. In this section, global models for electron beam loss are briefly described.

At any given time t, the instantaneous electron beam current may be written as:

$$I_t(t) = I_0 e^{\frac{-t}{\tau}} \tag{41}$$

where I_0 is the initial beam current, and τ is the characteristic beam decay time constant. Due to the uncorrelated nature of the collisional processes, the characteristic decay time depends on the individual contributions from Coulomb, Bremsstrahlung, and Intrabeam sources for scattering. Given that we can only measure the net beam decay time, (τ), the gas and intrabeam components must be inferred from an array of measurements under different experimental conditions. For the simulation study in this paper, τ is assumed to be a non-linear function of vertical scraper position (y_s), RF voltage (V_{rf}), initial beam current (I_0), and total number of bunches (M_b).

Over longer time periods of time, electron beam loss deviates from pure exponential and is governed by a more general rate equation:

$$\frac{dN_e}{dt} = \sum_i A_i \, N_e \, N_i \, \sigma_i \tag{42}$$

where N_e is the number of electrons, N_i is the number of scattering centers, σ_i is the scattering cross section for each type of collision, and A_i are characteristic proportionality constants. The cross sections σ_i quantify the probability of particle loss for each collision process. Note that integration of the rate equation, $\frac{dN}{dt} = -\alpha N^2$, yields a beam decay profile in time:

$$N(t) = \frac{N_0}{1 + N_0 \, \alpha \, t} \sim N_0 e^{-\frac{t}{\tau}} \tag{43}$$

for short times t. The long-term decay curve is more complicated than the exponential expression for instantaneous decay because the density of scattering centers is reduced roughly in proportion to circulating beam current.

The objective of the current study is to build PUNDA models that accurately predict the electron beam decay as a function of electron beam parameters and synchrotron operating parameters. The training of the NN block in the PUNDA model will be constrained by first-principles models (*i.e.* particle collision physics) for each scattering mechanism, and hence the trained model will be physically meaningful.

3.4.2 - Simulation Results: For the simulation study in this paper the following first-principles model is used to describe the electron beam loss:

$$I_t(t) = \frac{I_0 e^{-bt}}{1 + \left(\frac{a}{b}\right) I_0 \, (1 - e^{-bt})} \tag{44}$$

where I_0 is the initial beam current, and a and b are parameters of the parametric model for electron beam loss that are functions of the beam operating conditions.

The parameter $a = a_T + a_B + a_C$, is affected by Touschek (a_T), Bremsstrahlung (a_B), and Coulomb (a_C) effects on beam loss [27] which in turn depend on gap voltage (V_{rf}), vertical scraper position (Y_{s}), dynamic vacuum pressure (P_{dyn}) and number of bunches (M_{b}). The parameter $b = b_B + b_C$ is affected by Bremsstrahlung (b_B) and Coulomb (b_C) collisions due to the base pressure.

For the simulation results shown in this section, four major parameters, *i.e.* Y_{s}, V_{rf}, M_{b}, and I_0 are varied over an operation range consistent with that at SPEAR3. Electron beam loss is simulated using Eq. (44). Random noise is added to the simulated electron beam current to reflect imperfect current measurements. The noisy electron beam current is then used to construct a PUNDA model where Eq. (44)

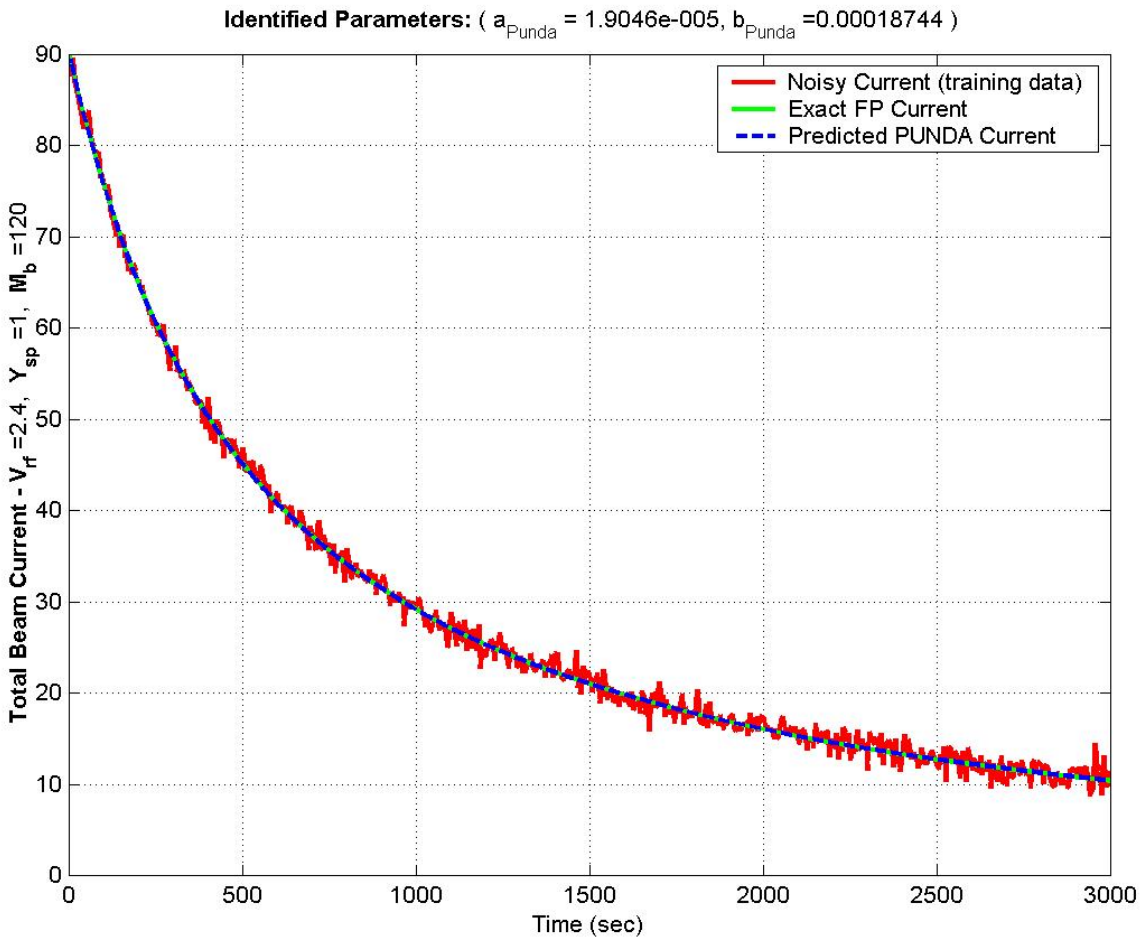

Figure 27: Prediction of electron beam decay using PUNDA model. The beam decay is accurately predicted while acceptable estimates of the decay model parameters are obtained. The actual parameter values used in the FP model of Eq. (44) are $a_{FP} = 1.8992e-005$ and $b_{FP} = 0.00018964$.

constitutes the *Parametric First Principles Model* block, and a NN constitutes the *Nonlinear Empirical Model* block. The combined PUNDA model is trained via constrained optimization, identifying appropriate parameter values for the FP model (*i.e.* a and b), at the same time that the decay in electron beam current is modeled. Figure 27-30 capture typical simulation results.

PUNDA structure offers a framework in which both beam data and first principles models may be used to complement one another. The nonlinear empirical model block may be used to capture the less known aspects of the beam decay that is reflected in the operation data but is not fully explained by first-principles information. Once a PUNDA model is verified to capture the beam loss in a particle accelerator, the model can be used to identify potential sources of beam loss in real-time.

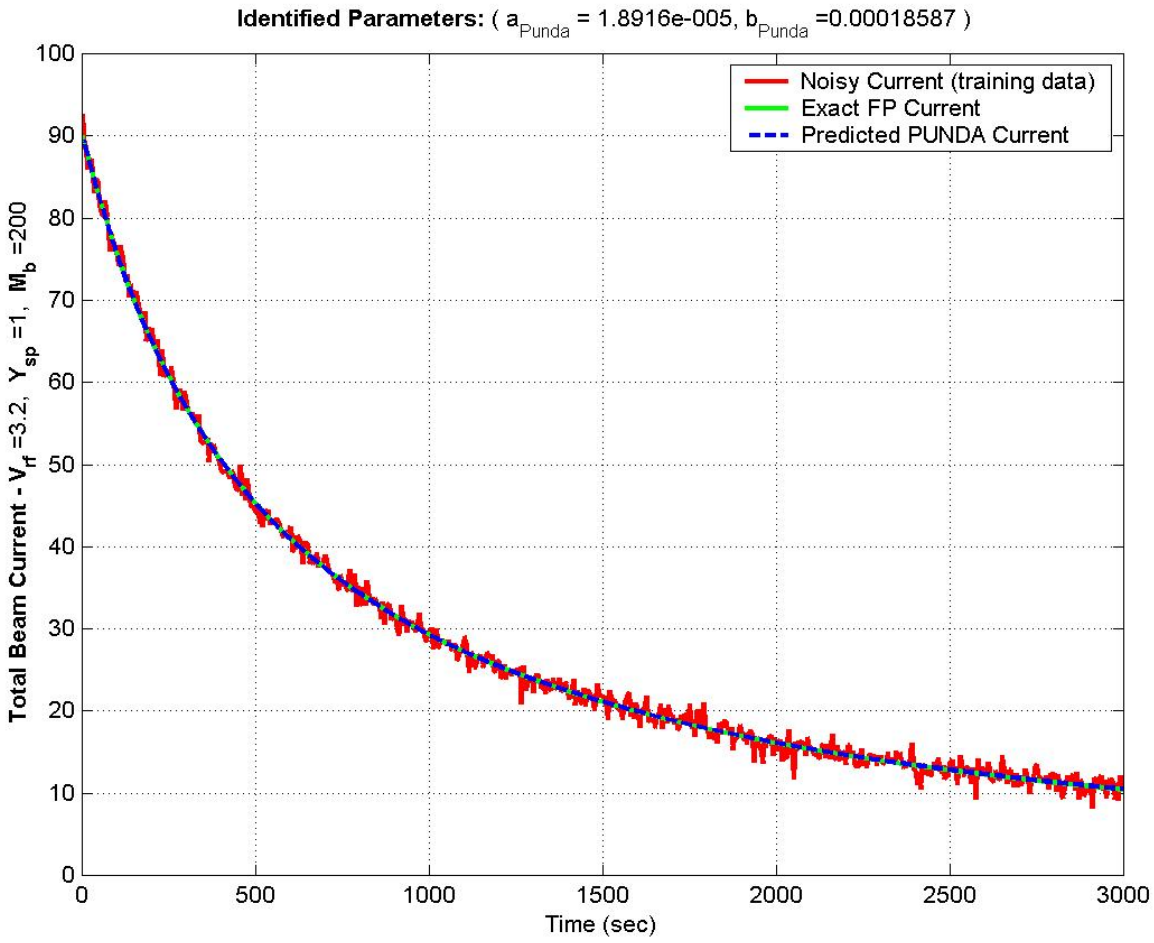

Figure 28: Prediction of electron beam decay using PUNDA model. The beam decay is accurately predicted while acceptable estimates of the decay model parameters are obtained. The actual parameter values used in the FP model of Eq. (44) are $a_{FP} = 1.882e-005$ and $b_{FP} = 0.0001905$.

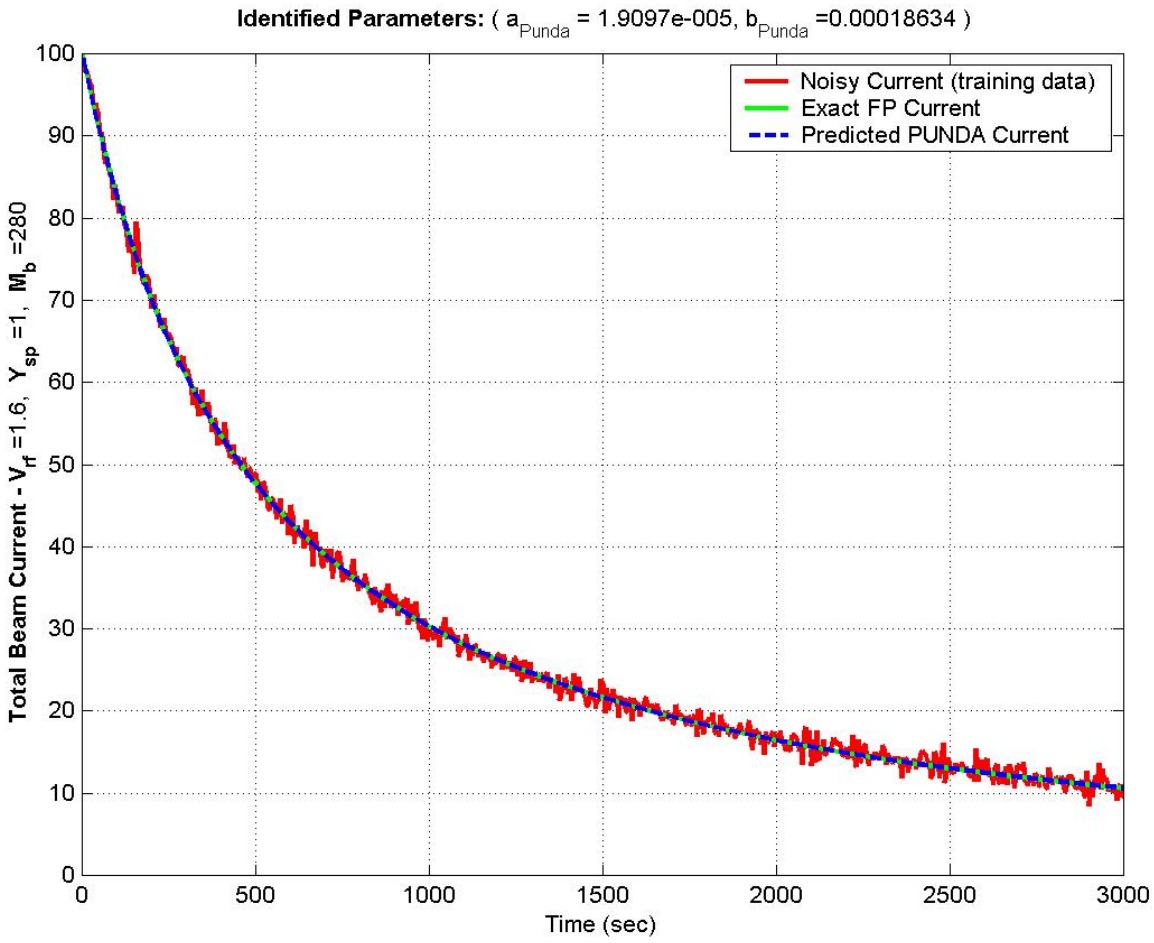

Figure 29: Prediction of electron beam decay using PUNDA model. The beam decay is accurately predicted while acceptable estimates of the decay model parameters are obtained. The actual parameter values used in the FP model of Eq. (44) are $a_{FP} = 1.9058e-005$ and $b_{FP} = 0.00018842$.

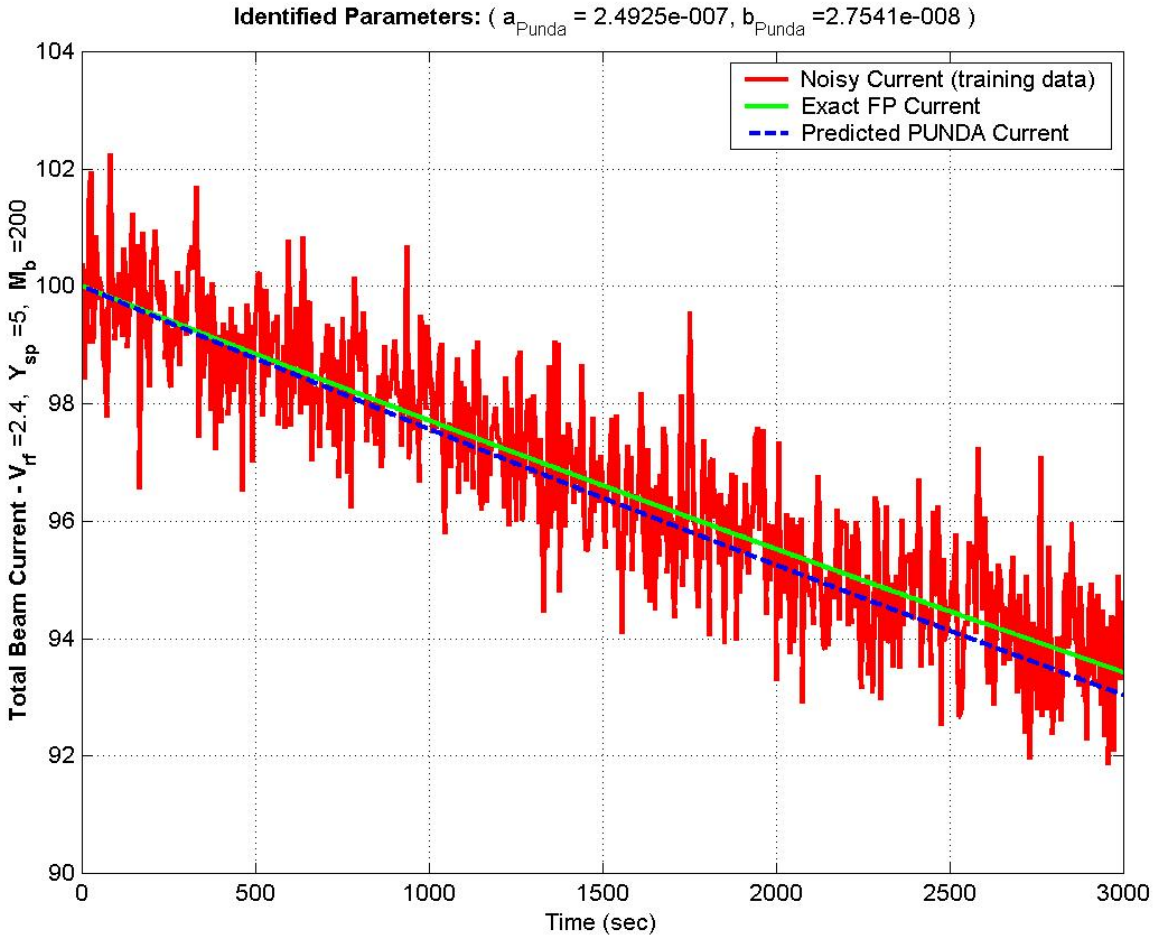

Figure 30: Prediction of electron beam decay using PUNDA model. The beam decay is accurately predicted while acceptable estimates of the decay model parameters are obtained. The actual parameter values used in the FP model of Eq. (44) are $a_{FP} = 2.4925e-007$ and $b_{FP} = 2.437e-006$.

3.5 - State-Space Representation of Bio-Reaction using PUNDA Models

In this section we will briefly describe the application of the PUNDA modeling framework to a bio-fermentation process in pharmaceutical industry. Using real process data, we will show that the PUNDA modeling framework provides an environment in which measured process data and fundamental process knowledge are optimally utilized to build accurate and computationally efficient models.

3.5.1 - Detailed Analysis of Fermentation Operating Data: Biochemical reactors are used to produce a large number of intermediate and final products of mass consumption in industries such as pharmaceuticals, food, and biofuels [28]. In its simplest form, a reaction in a biochemical reactor involves two components: biomass and substrate. The biomass consists of cells that consume the substrate. In a fermentation process for example, the cells consume sugar and produce alcohol. Modeling equations for bio reaction are obtained by writing out the material balances for different components participating in the bioreaction. In particular,

1. The biomass material balance can be written as:

$$\text{rate of accumulation} = \text{in by flow} - \text{out by flow} + \text{biomass generation} \tag{45}$$

   where biomass generation can be modeled as biomass concentration, x_1, multiplied by a production rate for the biomass, μ_b.

2. The substrate material balance can be written as:

$$\text{rate of accumulation} = \text{in by flow} - \text{out by flow} - \text{substrate consumption} \tag{46}$$

   where substrate consumption can be modeled as substrate concentration, x_2, multiplied by a consumption rate for the substrate[16], μ_s.

Due to the strict confidentiality agreements between Pavilion Technologies, Inc. and its pharmaceutical customer, the details of the bio-fermentation process from which the data for PUNDA modeling is obtained will not be discussed in this report. The fermentation process is a lengthy process currently requiring more than a few days for each run. Each batch undergoes several operating phases based on cell growth. Batch kinetics varies during various stages of the batch, and an appropriate model must account for these variations. Process inputs (*e.g.* feed, air, coolant, and valve openings) are manipulated (based on a predetermined recipe or under closed loop control) such that appropriate growth of biomass is ensured in the course of the bio-fermentation process. The goal of our study is to develop enough of an insight into the bio-fermentation process such that a strategy for the improved operation of the batch can be devised. In this report we will briefly present the results of the simulation study and demonstrate that the PUNDA models are correctly predicting the progress of the fermentation process.

3.5.2 - Model Development: During model development process, the customer provided Pavilion with a brief description of the process, along with the measurements for several runs of the batch process. The model development process is composed of the follwoing steps:

(a) The analysis of the data and correlating the data to the process description provided by the customer.

(b) Developing a candidate parameterized FP model, for the FP model block in PUNDA structure.

[16]The consumption rate is sometimes modeled as the ratio of the production rate to the yield of the bioreaction [28]. For this study however we adopted a more direct approach identifying the consumption rate directly from measured fermentation data.

(c) Developing a candidate nonlinear model that constitutes the empirical model block in the PUNDA structure.

(d) Training the PUNDA model using a nonlinear optimization algorithm that searches for the optimal values of the parameters in the empirical modeling block given measured process data and known operation/FP model constraints.

Biomass growth and glucose concentration were modelled as a function of batch initial-conditions and inputs. The ultimate goal of the modeling exercise was to predict the amount of harvestable active material at each point during the batch.

Figure 31 demonstrates a typical dataset over approximately 5 batches. To identify which inputs are most relevant to the desired outputs (*i.e.* biomass growth, glucose concentration, and titre (amount of harvestable active material), nonlinear correlations in the data was examined. A typical correlation plot is shown in Fig. 32.

As Fig. 32 indicates, the maximum correlation of the measured input data to Titre is 0.415 which is not high. Our modeling exercise revealed that building a model based on direct process measurements produces poor quality models for Titre.

Fig. 33 shows the PUNDA model for the prediction of the bio-reaction. In this PUNDA model, several parameters of a candidate FP model are modelled as static nonlinear functions of process inputs. The block diagram in Fig. 34 then uses the PUNDA model for bio-reaction as a component of a larger PUNDA model that produces accurate predictions of Titre.

Fig. 35 shows the prediction quality of the PUNDA models against the actual measured data. Fig. 36 shows the predictions of process response with an alternative inout (feed) profile. Predictions demonstrate that the batch length could be shortened with a more stable feed profile.

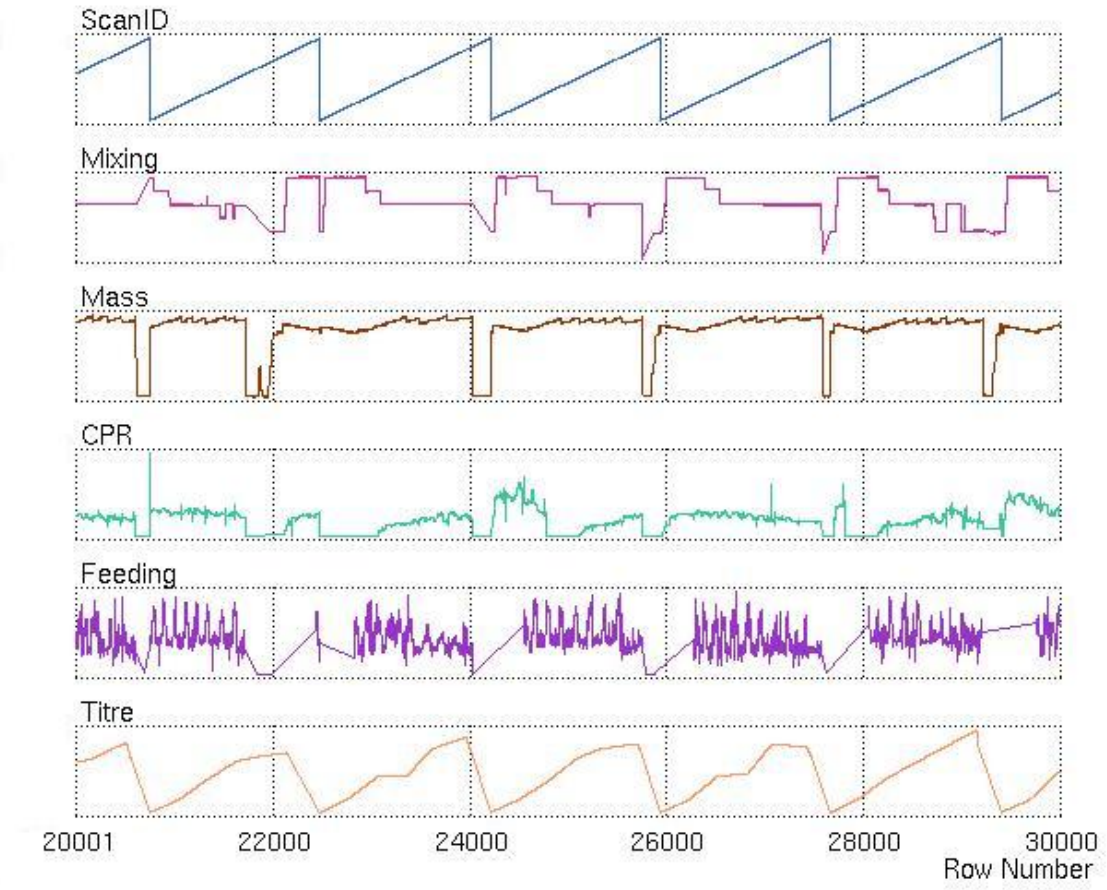

Fig. 31: Typical Batch data for the bio-fermentation process. Variable ranges are masked due to the proprietary nature of the Pavilion's customer data.

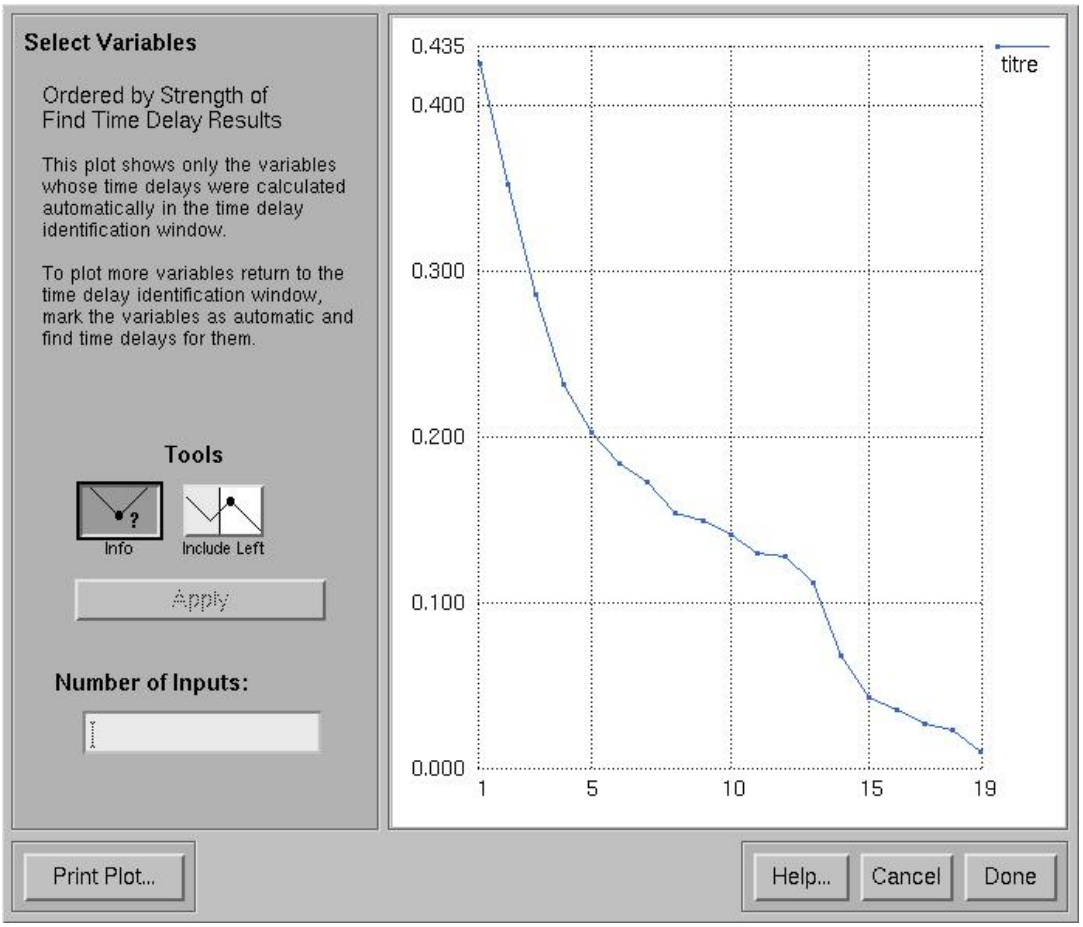

Fig. 32: plot demonstrating the result of Mutual Information analysis for Titre model. The nonlinear correlation of the various process inputs to the Titre measurements is ranked and plotted in this screen capture. The highly correlated variables are total feed, intCPR, mixing, mass, aeration, DO, motoramp, and glucose.

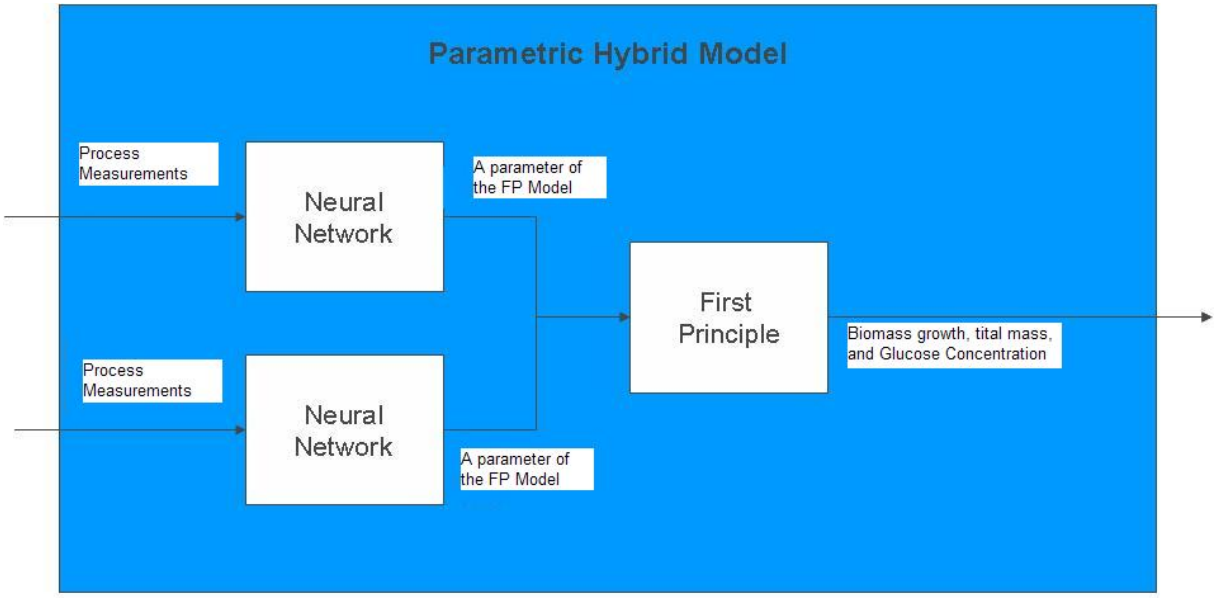

Fig. 33: Block diagram of the PUNDA model for the prediction of biomass growth, total mass accumulation, and glucose concentration.

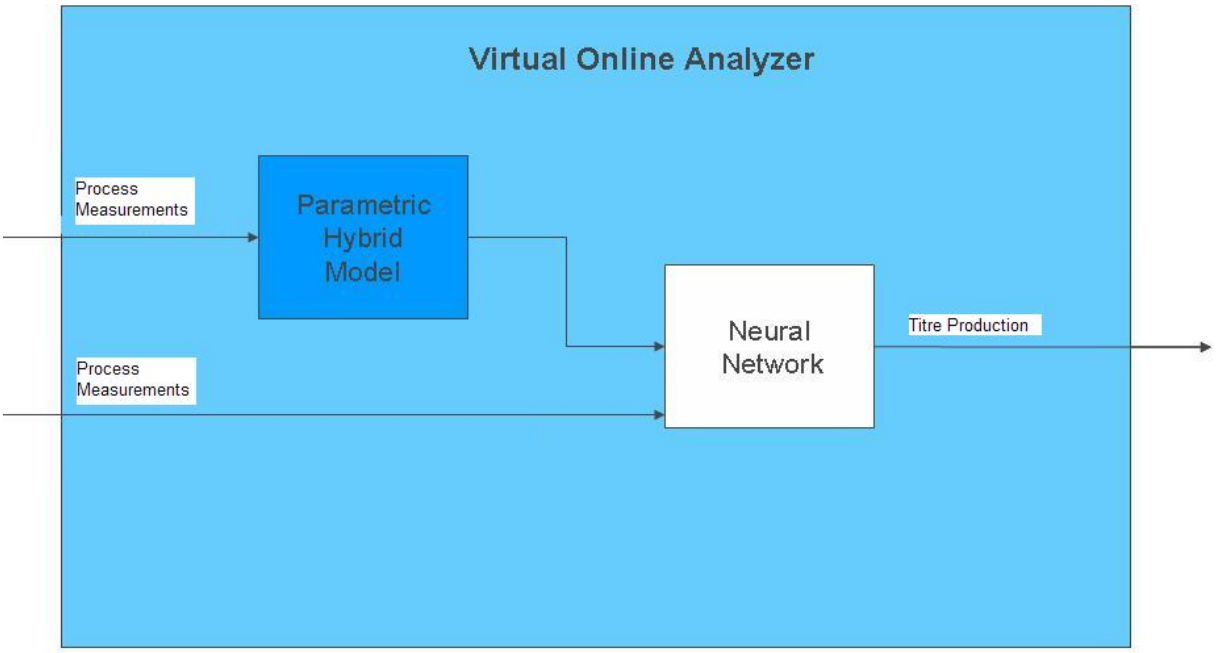

Fig. 34: Block diagram of the PUNDA model for the prediction of Titre. Note that in addition of direct process measurements, the PUNDA model for the bio-reaction process is also used to predict Titre.

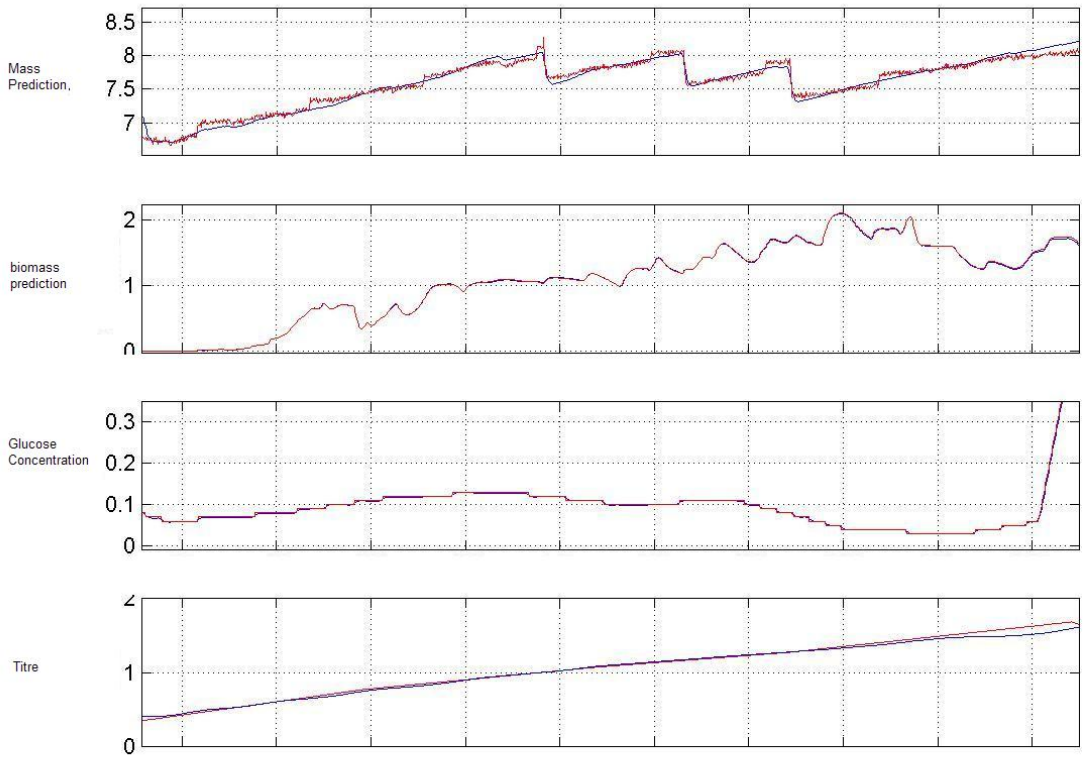

Fig. 35: Prediction Results for the bio-reaction in the fermenter as well as the total titre.

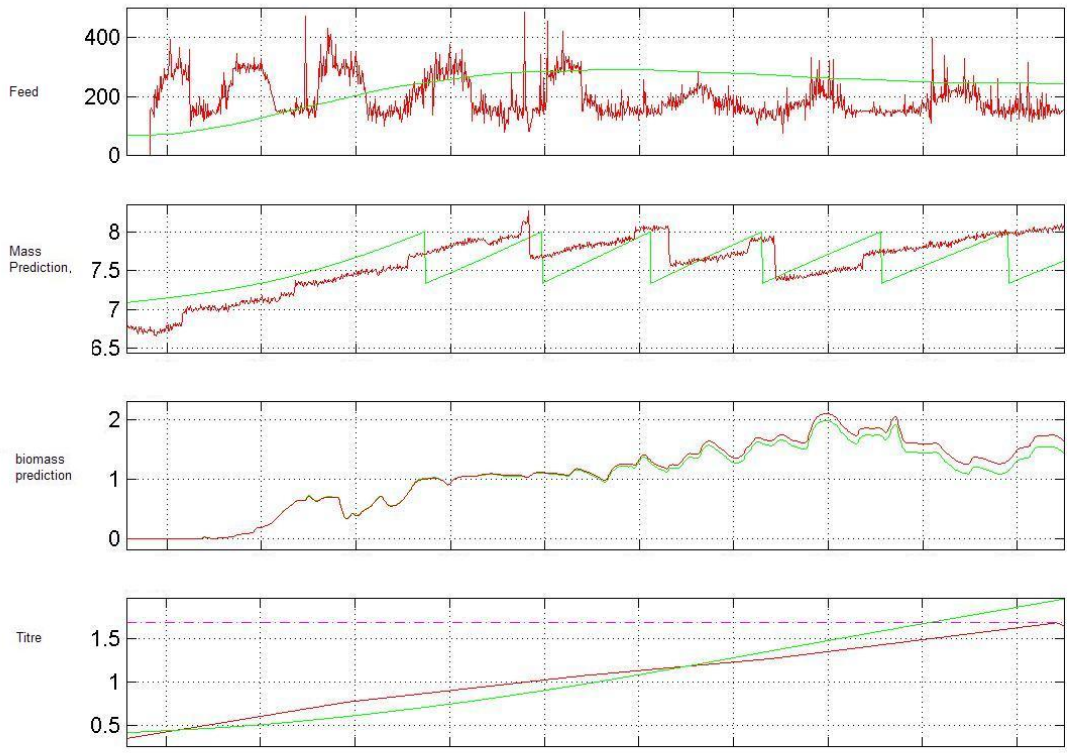

Fig. 36: Prediction Results for the bio-reaction in the fermenter as well as the total titre with an alternative feed profile.

References

[1] J. Rawlings, "Tutorial Overview of Model Predictive Control," *IEEE Control Systems Magazine*, vol. 20, pp. 38–52, June 2000.

[2] W. Dunbar and R. Murray, "Model Predictive Control of Coordinated Multi-Vehicle Formations," *Conference on Decision and Control*, 2002.

[3] G. Erdem, S. Abel, M. Morari, M. Mazzotti, and M. Morbidelli, "Online Optimization Based Feedback Control of Simulated Moving Bed Chromatographic Units," *Chemical and Biochemical Engineering Quarterly*, vol. 18, pp. 319–328, 2004.

[4] E. Martinoni, C. Pfister, K. Stadler, P. Schumacher, D. Leibundgut, T. Bouillon, T. Böhlen, and A. Zbinden, "Model-Based Control of Mechanical Ventilation: Design and Clinical Validation," *British Journal of Anaesthesia*, vol. 92, pp. 800–807, 2004.

[5] S. Qin and T. Badgwell, "A Survey of industrial model predictive control technology," *Control Engineering Practice*, April 2002.

[6] R. Murray, J. Hauser, A. Jadbabaie, M. Milam, W. Dunbar, and R. Franz, *Online Control Customization via Optimization Based Control*. John Wiley and Sons: Software Enabled Control: Information Technologies for Dynamical Systems, G. Balas and T. Samad, editors, 2002.

[7] B. Sayyar-Rodsari, E. Plumer, E. Hartman, K. Liano, and C. Axelrud, "Parametric Universal Nonlinear Dynamics Approximator and Use," *Pending Patent Application*, 2004.

[8] L. Ljung, *System Identification, Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[9] M. Jordan and D. Rumelhart, "Forward Models: Supervised Learning with a Distal Teacher,," *Cognitive Sci.*, vol. 16, p. 307, 1992.

[10] F. Cubillos and E. Lima, "Identification and Optimizing Control of a Rougher Flotation Circuit using an Adaptable Hybrid Neural Model," *Minerals Eng.*, vol. 10, p. 707, 1997.

[11] M. Thompson and M. Kramer, "Modeling Chemical Processes Using Prior Knowledge and Neural Networks," *AIChE Journal*, vol. 40, p. 1328, 1994.

[12] D. Psichogios and L. Ungar, "A Hybrid Neural Network-First Principles Approach to Process Modeling," *AIChE Journal*, vol. 38, p. 1499, 1992.

[13] D. Dowell, P. Bolton, J. Clendenin, P. Emma, S. Gierman, C. Limborg, B. Murphy, and J. Schmerge, "Longitudinal Measurements at the SLAC Gun Test Facility," *SLAC-PUB-9541*, September 2002.

[14] D. Dowell, P. Bolton, J. Clendenin, S. Gierman, C. Limborg, B. Murphy, and J. Schmerge, "Analysis of Slice Emittance Measurements For the SLAC Gun Test Facility," *SLAC-PUB-10727*, May 2003.

[15] J. Schmerge, J. Castro, J. Clendenin, D. Dowell, S. Gierman, and R. Hettel, "Emittance and Quantum Efficiency Measurements from a 1.6 cell S-Band Photocathode RF Gun with Mg Cathode," *SLAC-PUB-10763*, September 2004.

[16] J. Schmerge, P. Bolton, J. Clendenin, D. Dowell, S. Gierman, C. Limborg, and B. Murphy, "6D Phase Space Measurements at the SLAC Gun Test Facility," *SLAC-PUB-9681*, March 2003.

[17] K. Brown, F. Rothacker, D. Carey, and C. Iselin, "TRANSPORT: A Computer Program for Designing Charged Particle Beam Transport Systems," *SLAC-91, Rev 2*, May 1977.

[18] J. Schmerge, J. Clendenin, D. Dowell, and S. Gierman, "GTF Transverse and Longitudinal Emittance Data Analysis Technique," *LCLS-TN-05-19*, July 2005.

[19] J. Jackson, *Classical Electrodynamics*. New York: John Wiley and Sons, 1975.

[20] K. Kim, "Characteristics of Synchrotron Radiation," *Physics of Particle Accelerators, Eds. M. Month and M. Dienes, AIP Conf. Proc.*, p. 184, 1989.

[21] A. Hofmann, *The Physics of Synchrotron Radiation*. Cambridge University Press, 2004.

[22] M. Cornacchia, "Requirements and Limitations on Beam Quality in Synchrotron Radiation Sources," *CERN 90-03*, 1990.

[23] C. Bocchetta, "Lifetime and Beam Quality," *Invited Lectures at the CERN Accelerator School on Synchrotron Radiation and Free Electron Lasers*, June 1997.

[24] H. Bruck, *Accelateurs Circulaires de Particules*. Paris: Presses Universitaires de France, 1969.

[25] H. Wiedemann, *Particle Accelerator Physics II*. Berlin: Springer Verlag, 1996.

[26] A. Sorensen, "Introduction to Intrabeam Scattering," *CERN Accelerator School, CERN 87-10*, 1987.

[27] J. Corbett and et al, "Electron Beam Lifetime in SPEAR3: Measurement and Simulation," these proceedings.

[28] B. Bequette, *Process Dynamics - Modeling, Analysis, and Simulation*. New Jersey: Prentice Hall, 1998.

Appendix A

The xml file for the optimization problem to find beam matrix parameters at the exit of the linac section given horizontal beam size measurements at the screen, with the assumption of full knowledge of the transport matrix for each quad current is as follows:

```
<formulation>
  <problem classname='com.pav.pdas.optimization.common.NLPProblem' direction='minimize'>
    <variable label='s11' modelVariable='s11' lower='0' upper='10'
              useRandomInitial='true' randomSeed='11'
              randomLower='0' randomUpper='10'/>
    <variable label='s12' modelVariable='s12' lower='0' upper='10'
              useRandomInitial='true' randomSeed='12'
              randomLower='0' randomUpper='10'/>
    <variable label='s22' modelVariable='s22' lower='0' upper='10'
              useRandomInitial='true' randomSeed='21'
              randomLower='0' randomUpper='10'/>
    <objective label='obj' modelVariable='obj'/>
  </problem>

  <mapping classname="com.pav.pdas.model.expression.ExpressionMapping">
    <data name='d'
          file='fitdata.csv'
          rowCount='14' columnCount='23' />
    <input name='s11'/>
    <input name='s12'/>
    <input name='s22'/>
    <output name='obj'/>
    <output name='dx_model'>
      <dimension set='K'/>
    </output>
    <expressions>
      set K = |1:14|;
      set J = |1:23|;

      real data[K,J] = d;

      real A[K];
      real B[K];

      real dx_measured[K];
      real sdx_measured[K];
      real dx2_model[K];

      real error_linear[K];
      real error_linear_ns[K];
      real error_nonlinear[K];
      real error_nonlinear_ns[K];

      real s11_s, s12_s, s22_s;

      A[k] = data[k, 9] foreach k in K;
      B[k] = data[k,10] foreach k in K;

      dx_measured[k]  = data[k,15] foreach k in K;
      sdx_measured[k] = data[k,16] foreach k in K;

      s11_s = s11/1e8;
      s12_s = s12/1e8;
      s22_s = s22/1e8;

      dx2_model[k] = A[k]^2*s11_s + 2*A[k]*B[k]*s12_s + B[k]^2*s22_s   foreach k in K;
      dx_model[k] = 1e6*sqrt(dx2_model[k])   foreach k in K;

      foreach k in K
      {
        error_linear[k]       = ( dx_model[k]^2 - dx_measured[k]^2 )/( 2*dx_measured[k]*sdx_measured[k] );
        error_linear_ns[k]    = ( dx_model[k]^2 - dx_measured[k]^2 );
        error_nonlinear[k]    = ( dx_model[k]   - dx_measured[k]   )/( sdx_measured[k] );
        error_nonlinear_ns[k] = ( dx_model[k]   - dx_measured[k]   );
```

```
      }

      obj = sum( k in K | error_linear[k]^2 );
    </expressions>
  </mapping>

  <solver classname='com.pav.pdas.optimization.snopt.SnoptDirect'>
    <parameter name='major feasibility tolerance' value='1e-6' />
    <parameter name='major optimality tolerance' value='1e-6' />
    <parameter name='major print level' value='1' />
    <parameter name='print file' value='6' />
    <parameter name='solution' value='true' />
    <parameter name='solution file' value='6' />
  </solver>

</formulation>
```

Appendix B

The xml file for the optimization problem to find beam matrix parameters at the exit of the linac section along with the transport matrix elements given horizontal beam size measurements at the screen is as follows:

```
<formulation>
  <problem classname='com.pav.pdas.optimization.common.NLPProblem' direction='minimize'>
    <variable label='s11' modelVariable='s11' lower='2.0' upper='3.0' useRandomInitial='true' randomSeed='11'
                              randomLower='2.63'         randomUpper='2.64'/>
    <variable label='s12' modelVariable='s12' lower='1.0' upper='2.0' useRandomInitial='true' randomSeed='12'
                              randomLower='1.57'         randomUpper='1.58'/>
    <variable label='s22' modelVariable='s22' lower='1.0' upper='2.0' useRandomInitial='true' randomSeed='21'
                              randomLower='1.43'             randomUpper='1.44'/>
    <variable label='weights' modelVariable='weights' lower='-10' upper='10' useRandomInitial='true' randomSeed='31'
                              randomLower='-0.01'            randomUpper='0.01'/>
    <constraint label='maxdAdc' modelVariable='maxdAdc' upper='0' />
    <constraint label='maxdBdc' modelVariable='maxdBdc' upper='0' />

    <objective label='obj' modelVariable='obj'/>
  </problem>

  <mapping classname="com.pav.pdas.model.expression.ExpressionMapping">
    <data name='d'  file='fitdata.csv'
                rowCount='14'
              columnCount='23' />
    <mapping classname='com.pav.pdas.model.mlp.MlpMapping' name='mlp'>
      <input  name='c'    layer='layer0' node='0' scale='1'/>
      <output name='A'    layer='layer2' node='0' scale='1'/>
      <output name='B'    layer='layer2' node='1' scale='1'/>
      <gain   name='dAdc' order='1' type='local' dy='A' da='c' />
      <gain   name='dBdc' order='1' type='local' dy='B' da='c' />
      <layer  name='layer0' size='1' type='identity'/>
      <layer  name='layer1' size='5' type='tanh'/>
      <layer  name='layer2' size='2' type='linear'/>
      <connection srcName='layer0' dstName='layer1'/>
      <connection srcName='layer1' dstName='layer2'/>
    </mapping>

    <input name='weights'>  <dimension set='I'/>  </input>
    <input name='s11'/>
    <input name='s12'/>
    <input name='s22'/>
    <output name='obj'/>
    <output name='dx_model'> <dimension set='K'/> </output>
    <output name='A'> <dimension set='K'/> </output>
    <output name='B'> <dimension set='K'/> </output>
    <output name='dAdc'> <dimension set='K'/> </output>
    <output name='dBdc'> <dimension set='K'/> </output>
    <output name='maxdAdc' />
    <output name='maxdBdc' />

    <expressions>
      set K = |1:14|;
      set J = |1:23|;
      set I = |0:21|;

      real data[K,J] = d;

      real current[K];
    real sdAdc[K];
    real sdBdc[K];

      real dx_measured[K];
      real sdx_measured[K];
      real dx2_model[K];

      real error_linear[K];
      real error_linear_ns[K];
      real error_nonlinear[K];
```

```
      real error_nonlinear_ns[K];

      real s11_s, s12_s, s22_s;

      current[k]      = data[k, 1] foreach k in K;

      dx_measured[k]  = data[k,15] foreach k in K;
      sdx_measured[k] = data[k,16] foreach k in K;

      s11_s = s11/1e8;
      s12_s = s12/1e8;
      s22_s = s22/1e8;

      mlp( weights[i] = weights[i] foreach i in I; c=current[k] ::
                              A[k]=A; B[k]=B; dAdc[k]=dAdc; dBdc[k]=dBdc) foreach k

in K;

    sdAdc[k] = dAdc[k] * 1.2 foreach k in K;
    sdBdc[k] = dBdc[k] * 5.4 foreach k in K;

      dx2_model[k] = A[k]^2*s11_s + 2*A[k]*B[k]*s12_s + B[k]^2*s22_s   foreach k in K;
      dx_model[k] = 1e6*sqrt(dx2_model[k])   foreach k in K;

      foreach k in K
      {
        error_linear[k]       = ( dx_model[k]^2 - dx_measured[k]^2 )/( 2*dx_measured[k]*sdx_measured[k] );
        error_linear_ns[k]    = ( dx_model[k]^2 - dx_measured[k]^2 ) ;
        error_nonlinear[k]    = ( dx_model[k]   - dx_measured[k]   )/( sdx_measured[k] );
        error_nonlinear_ns[k] = ( dx_model[k]   - dx_measured[k]   );
      }

      obj = sum( k in K | error_linear[k]^2 );

    maxdAdc = sum( k in K | max( sdAdc[k]+1e-2, 0)^2 );
    maxdBdc = sum( k in K | max( sdBdc[k]+1e-2, 0)^2 );

    </expressions>
  </mapping>

  <solver classname='com.pav.pdas.optimization.snopt.SnoptDirect'>
    <parameter name='major feasibility tolerance' value='1e-6' />
    <parameter name='major optimality tolerance' value='1e-6' />
    <parameter name='major step limit' value='0.75' />
    <parameter name='major print level' value='1' />
    <parameter name='print file' value='6' />
    <parameter name='solution' value='true' />
    <parameter name='solution file' value='6' />
  </solver>

</formulation>
```

Appendix C

The xml file for the optimization problem to find beam matrix parameters at the exit of the linac section along with α and β in I_2-K_2 mapping, given horizontal beam size measurements at the screen is as follows:

```
<formulation>
  <problem classname='com.pav.pdas.optimization.common.NLPProblem' direction='minimize'>
    <variable label='s_11' modelVariable='s_11'             lower='0.0'       upper='10000'
              useRandomInitial='true' randomSeed='11' randomLower='0.0' randomUpper='10'/>
    <variable label='s_12' modelVariable='s_12'             lower='0.0'       upper='10000'
              useRandomInitial='true' randomSeed='12' randomLower='0.0' randomUpper='10'/>
    <variable label='s_22' modelVariable='s_22'             lower='0.0'       upper='10000'
              useRandomInitial='true' randomSeed='21' randomLower='0.0' randomUpper='10'/>
    <variable label='slope_q3' modelVariable='slope_q3'     lower='160.87'    upper='160.87'
              useRandomInitial='true' randomSeed='31' randomLower='0.0' randomUpper='500'/>
    <variable label='int_q3' modelVariable='int_q3'         lower='49.0'      upper='49.0'
              useRandomInitial='true' randomSeed='32' randomLower='0'   randomUpper='100'/>
    <variable label='slope_q4' modelVariable='slope_q4'     lower='147.23'          upper='187.23'
              useRandomInitial='true' randomSeed='41' randomLower='167.23'    randomUpper='167.23'/>
    <variable label='int_q4' modelVariable='int_q4'         lower='23.17'          upper='63.17'
              useRandomInitial='true' randomSeed='42' randomLower='43.17'    randomUpper='43.17'/>
    <objective label='obj' modelVariable='obj'/>
  </problem>

  <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='trainer'>
    <data name='d' file='fitdata.csv' rowCount='14' columnCount='23' />

    <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='Transfer'>
      <input name='I_2'/>
      <input name='s_11'/>
      <input name='s_12'/>
      <input name='s_22'/>
      <input name='slope_q3'/>
      <input name='int_q3'/>
      <input name='slope_q4'/>
      <input name='int_q4'/>
      <output name='A_T'/>
      <output name='B_T'/>
      <output name='C_T'/>
      <output name='D_T'/>
      <output name='dx'/>
      <output name='dxp'/>

      <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='QuadTransfer'>
        <input name='I_2'/>
        <input name='slope_q3'/>
        <input name='int_q3'/>
        <input name='slope_q4'/>
        <input name='int_q4'/>
        <output name='A_T'/>
        <output name='B_T'/>
        <output name='C_T'/>
        <output name='D_T'/>
        <parameter name='E_slice' value='30.50'/>
        <parameter name='A_spec' value='0.953728998305'/>
        <parameter name='B_spec' value='0.418879020479'/>
        <parameter name='C_spec' value='-0.682448955752'/>
        <parameter name='D_spec' value='0.748783408240'/>
        <parameter name='l_eff' value='0.07677'/>
        <parameter name='I_Qy' value='0'/>

        <expressions>

          real l_qtq;
          real l_drift1, l_drift2;
          real K_1, K_2;
          real A_quad, B_quad, C_quad, D_quad;

          l_drift1 = 1.873-l_eff/2;
          l_drift2 = 0.4191;
          l_qtq    = (4.36-4.173)-l_eff;
```

```
      K_1      = sqrt( ( slope_q3*I_Qy + int_q3 )/E_slice );
      K_2      = sqrt( ( slope_q4*I_2  + int_q4 )/E_slice );

      A_quad =         cos(K_2*l_eff)*( cosh(K_1*l_eff)     + l_qtq*K_1*sinh(K_1*l_eff) )
             +   K_1/K_2*sin(K_2*l_eff)*  sinh(K_1*l_eff);
      B_quad =         cos(K_2*l_eff)*( sinh(K_1*l_eff)/K_1 + l_qtq*    cosh(K_1*l_eff) )
             +   1/K_2*sin(K_2*l_eff)*  cosh(K_1*l_eff);
      C_quad =    -K_2*sin(K_2*l_eff)*( cosh(K_1*l_eff)     + l_qtq*K_1*sinh(K_1*l_eff) )
             +     K_1*cos(K_2*l_eff)*  sinh(K_1*l_eff);
      D_quad =    -K_2*sin(K_2*l_eff)*( sinh(K_1*l_eff)/K_1 + l_qtq*    cosh(K_1*l_eff) )
             +         cos(K_2*l_eff)*  cosh(K_1*l_eff);
      A_T    = A_spec*( A_quad + l_drift1*C_quad ) + B_spec*C_quad
             + l_drift2*C_spec*( A_quad + l_drift1*C_quad) + l_drift2*D_spec*C_quad;
      B_T    = A_spec*( B_quad + l_drift1*D_quad ) + B_spec*D_quad
             + l_drift2*C_spec*( B_quad + l_drift1*D_quad ) + l_drift2*D_spec*D_quad;
      C_T    = C_spec*( A_quad + l_drift1*C_quad ) + D_spec*C_quad;
      D_T    = C_spec*( B_quad + l_drift1*D_quad ) + D_spec*D_quad;

    </expressions>
  </mapping>

  <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='BeamTransfer'>
    <input name='A_T'/>
    <input name='B_T'/>
    <input name='C_T'/>
    <input name='D_T'/>
    <input name='s_11'/>
    <input name='s_12'/>
    <input name='s_22'/>
    <output name='dx'/>
    <output name='dxp'/>
    <expressions>
      real dx2, dxp2;

      dx2  = A_T^2*s_11 + 2*A_T*B_T*s_12 + B_T^2*s_22;
      dxp2 = C_T^2*s_11 + 2*C_T*D_T*s_12 + D_T^2*s_22;

      dx  = 1e2*sqrt(dx2);
      dxp = 1e2*sqrt(dxp2);
    </expressions>
  </mapping>

  <expressions>

    QuadTransfer( I_2 = I_2;
                  slope_q3=slope_q3; int_q3=int_q3;
                  slope_q4=slope_q4; int_q4=int_q4
                  ::
                  A_T = A_T; B_T = B_T; C_T = C_T; D_T = D_T );
    BeamTransfer( A_T = A_T; B_T = B_T; C_T = C_T; D_T = D_T;
                  s_11 = s_11; s_12 = s_12; s_22 = s_22
                  ::
                  dx = dx; dxp = dxp);
  </expressions>
</mapping>

<input name='s_11'/>
<input name='s_12'/>
<input name='s_22'/>
<input name='slope_q3' value='160.87'/>
<input name='int_q3' value='49.0'/>
<input name='slope_q4' value='167.23'/>
<input name='int_q4' value='43.17'/>
<output name='obj'/>
<!-- These are listed as outputs to see their values at the solution. -->
<output name='dx_model'>
  <dimension set='K'/>
</output>
<output name='dxp_model'>
  <dimension set='K'/>
</output>
<output name='A_T'>
```

```
      <dimension set='K'/>
    </output>
    <output name='B_T'>
      <dimension set='K'/>
    </output>
    <output name='C_T'>
      <dimension set='K'/>
    </output>
    <output name='D_T'>
      <dimension set='K'/>
    </output>
    <expressions>
      set K = |1:14|;
      set J = |1:23|;

      real data[K,J] = d;

      real I_2[K];
      real dx_measured[K];
      real sdx_measured[K];
      real error_linear[K];
      real error_linear_ns[K];
      real error_nonlinear[K];
      real error_nonlinear_ns[K];

      I_2[k]          = data[k, 1] foreach k in K;
      dx_measured[k]  = data[k,15] foreach k in K;
      sdx_measured[k] = data[k,16] foreach k in K;

      Transfer( s_11 = s_11;
                s_12 = s_12;
                s_22 = s_22;
                slope_q3=slope_q3; int_q3=int_q3;
                slope_q4=slope_q4; int_q4=int_q4;
                I_2 = I_2[k]
                ::
                A_T[k] = A_T;
                B_T[k] = B_T;
                C_T[k] = C_T;
                D_T[k] = D_T;
                dx_model[k] = dx;
                dxp_model[k] = dxp
              ) foreach k in K;

      foreach k in K
      {
        error_linear[k]       = ( dx_model[k]^2 - dx_measured[k]^2 )/( 2*dx_measured[k]*sdx_measured[k] );
        error_linear_ns[k]    = ( dx_model[k]^2 - dx_measured[k]^2 );
        error_nonlinear[k]    = ( dx_model[k]   - dx_measured[k]   )/( sdx_measured[k] );
        error_nonlinear_ns[k] = ( dx_model[k]   - dx_measured[k]   );
      }

      obj = sum( k in K | error_nonlinear[k]^2 );

    </expressions>
  </mapping>

  <solver classname='com.pav.pdas.optimization.snopt.SnoptDirect'>
    <parameter name='major feasibility tolerance' value='1e-6' />
    <parameter name='major optimality tolerance' value='1e-6' />
    <parameter name='major step limit' value='0.1' />
    <parameter name='major print level' value='1' />
    <parameter name='print file' value='6' />
    <parameter name='solution' value='true' />
    <parameter name='solution file' value='6' />
    <parameter name='scale option' value='2' />
  </solver>

</formulation>
```

Appendix D

The xml file for the optimization problem to find optimal quad setting given a desired operating condition for the beam $\left(\Delta X_{des}, \Delta X'_{des}\right)$:

```
<formulation>
  <problem classname='com.pav.pdas.optimization.common.NLPProblem' direction='minimize'>
    <variable label='I_2' modelVariable='I_2' lower='0' upper='10' useRandomInitial='true' randomSeed='11'
                                              randomLower='0'         randomUpper='10'/>
    <objective label='obj' modelVariable='obj'/>
  </problem>

  <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='setpoint'>
    <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='Transfer'>
      <input name='I_2'/>
      <input name="s_11" value="2.6367971055566173" />
      <input name="s_12" value="1.5738242270790272" />
      <input name="s_22" value="1.435441121463094" />
      <output name='A_T'/>
      <output name='B_T'/>
      <output name='C_T'/>
      <output name='D_T'/>
      <output name='dx'/>
      <output name='dxp'/>

      <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='QuadTransfer'>
        <input name='I_2'/>
        <output name='A_T'/>
        <output name='B_T'/>
        <output name='C_T'/>
        <output name='D_T'/>
        <parameter name='E_slice' value='30.50'/>
        <parameter name='slope_q3' value='160.87'/>
        <parameter name='int_q3' value='49.0'/>
        <parameter name='slope_q4' value='167.23'/>
        <parameter name='int_q4' value='43.17'/>
        <parameter name='A_spec' value='0.953728998305'/>
        <parameter name='B_spec' value='0.418879020479'/>
        <parameter name='C_spec' value='-0.682448955752'/>
        <parameter name='D_spec' value='0.748783408240'/>
        <parameter name='l_eff' value='0.07677'/>
        <parameter name='I_Qy' value='0'/>

        <expressions>

          real l_qtq;
          real l_drift1, l_drift2;
          real K_1, K_2;
          real A_quad, B_quad, C_quad, D_quad;

          l_drift1 = 1.873-l_eff/2;
          l_drift2 = 0.4191;

          l_qtq = (4.36-4.173)-l_eff;

          K_1 = sqrt( ( slope_q3*I_Qy + int_q3 )/E_slice );
          K_2 = sqrt( ( slope_q4*I_2  + int_q4 )/E_slice );

          A_quad =         cos(K_2*l_eff)*( cosh(K_1*l_eff)     + l_qtq*K_1*sinh(K_1*l_eff) )
                 + K_1/K_2*sin(K_2*l_eff)*  sinh(K_1*l_eff);

          B_quad =         cos(K_2*l_eff)*( sinh(K_1*l_eff)/K_1 + l_qtq*    cosh(K_1*l_eff) )
                 +   1/K_2*sin(K_2*l_eff)*  cosh(K_1*l_eff);

          C_quad =    -K_2*sin(K_2*l_eff)*( cosh(K_1*l_eff)     + l_qtq*K_1*sinh(K_1*l_eff) )
                 +     K_1*cos(K_2*l_eff)*  sinh(K_1*l_eff);

          D_quad =    -K_2*sin(K_2*l_eff)*( sinh(K_1*l_eff)/K_1 + l_qtq*    cosh(K_1*l_eff) )
                 +         cos(K_2*l_eff)*  cosh(K_1*l_eff);
```

```
        A_T = A_spec*( A_quad + l_drift1*C_quad ) + B_spec*C_quad
            + l_drift2*C_spec*( A_quad + l_drift1*C_quad) + l_drift2*D_spec*C_quad;

        B_T = A_spec*( B_quad + l_drift1*D_quad ) + B_spec*D_quad
            + l_drift2*C_spec*( B_quad + l_drift1*D_quad ) + l_drift2*D_spec*D_quad;

        C_T = C_spec*( A_quad + l_drift1*C_quad ) + D_spec*C_quad;

        D_T = C_spec*( B_quad + l_drift1*D_quad ) + D_spec*D_quad;

      </expressions>
    </mapping>

    <mapping classname='com.pav.pdas.model.expression.ExpressionMapping' name='BeamTransfer'>
      <input name='A_T'/>
      <input name='B_T'/>
      <input name='C_T'/>
      <input name='D_T'/>
      <input name='s_11'/>
      <input name='s_12'/>
      <input name='s_22'/>
      <output name='dx'/>
      <output name='dxp'/>
      <expressions>
        real s_11s, s_12s, s_22s;

        s_11s = s_11/1e8;
        s_12s = s_12/1e8;
        s_22s = s_22/1e8;

        real dx2, dxp2;

        dx2  = A_T^2*s_11s + 2*A_T*B_T*s_12s + B_T^2*s_22s;
        dxp2 = C_T^2*s_11s + 2*C_T*D_T*s_12s + D_T^2*s_22s;

        dx  = 1e6*sqrt(dx2);
        dxp = 1e6*sqrt(dxp2);
      </expressions>
    </mapping>

    <expressions>

      QuadTransfer( I_2 = I_2 :: A_T = A_T; B_T = B_T; C_T = C_T; D_T = D_T );
      BeamTransfer( A_T = A_T; B_T = B_T; C_T = C_T; D_T = D_T;
                    s_11 = s_11; s_12 = s_12; s_22 = s_22
                    ::
                    dx = dx; dxp = dxp);
    </expressions>
  </mapping>

  <input name='I_2'/>
  <input name='dx_desired' value='200'/>
  <input name='dxp_desired' value='140'/>
  <output name='dx_model'/>
  <output name='dxp_model'/>
  <output name='obj'/>
  <expressions>
    Transfer( I_2 = I_2 :: dx_model = dx; dxp_model = dxp );
    obj = (dx_desired - dx_model)^2 + (dxp_desired - dxp_model)^2;
  </expressions>
</mapping>

<solver classname='com.pav.pdas.optimization.snopt.SnoptDirect'>
  <parameter name='major feasibility tolerance' value='1e-6' />
  <parameter name='major optimality tolerance' value='1e-6' />
  <parameter name='major step limit' value='0.1' />
  <parameter name='major print level' value='1' />
  <parameter name='print file' value='6' />
  <parameter name='solution' value='true' />
  <parameter name='solution file' value='6' />
</solver>
```

```
</formulation>
```