Systems & Control Transactions

Research Article
Foundations of Computer Aided Process Design (FOCAPD 2024)
Breckenridge, Colorado, USA. July 14-18, 2024
Peer Reviewed Conference Proceeding

PSE PRESS

Guaranteed Error-bounded Surrogate Framework for Solving Process Simulation Problems

Chinmay M. Aras[a], Ashfaq Iftakher[a], and M. M. Faruque Hasan[a,b]*

[a] Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, Texas 77843-3122, USA
[b] Texas A&M Energy Institute, Texas A&M University, College Station, Texas 77843-3122, USA
* Corresponding Author: hasan@tamu.edu.

ABSTRACT

Process simulation problems often involve systems of nonlinear and nonconvex equations and may run into convergence issues due to the existence of recycle loops within such models. To that end, surrogate models have gained significant attention as an alternative to high-fidelity models as they significantly reduce the computational burden. However, these models do not always provide a guarantee on the prediction accuracy over the domain of interest. To address this issue, we strike a balance between computational complexity by developing a data-driven branch and prune-based framework that progressively leads to a guaranteed solution to the original system of equations. Specifically, we utilize interval arithmetic techniques to exploit Hessian information about the model of interest. Along with checking whether a solution can exist in the domain under consideration, we generate error-bounded convex hull surrogates using the sampled data and Hessian information. When integrated in a branch and prune framework, the branching leads to the domain under consideration becoming smaller, thereby reducing the quantified prediction error of the surrogate, ultimately yielding a solution to the system of equations. In this manner, we overcome the convergence issues that are faced by many simulation packages. We demonstrate the applicability of our framework through several case studies. We first utilize a set of test problems from literature. For each of these test systems, we can find a valid solution. We then demonstrate the efficacy of our framework on real-world process simulation problems.

Keywords: Surrogate Model, Modelling and Simulations, Algorithms, Data-Driven

INTRODUCTION

Systems of equations arise in many fields of practical interest such as engineering, economics, physics, and chemistry. A system of equations in N -dimensions is represented as:

$$\boldsymbol{F}(\boldsymbol{x}) = (f_1(x), f_2(x), \ldots\ldots f_N(x))^T$$

Simulation of chemical processes involve numerically solving such systems of equations. Solving such a system means finding a vector $\boldsymbol{x}$ such that $\boldsymbol{F}(\boldsymbol{x}) = 0$ where $\boldsymbol{x} = (x_1, x_2, x_3 \ldots\ldots, x_n)$,which is not trivial, but there are many existing methods that have been developed to solve such systems traditionally. These methods can be categorized under three general classes: i) Homotopy continuation methods; ii) Interval-Newton methods; iii) Newton-type methods.

Homotopy continuation methods involve starting with a system of equations $\boldsymbol{G}(\boldsymbol{x})$, whose solution is known, embedded in a homotopy function $\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{t})$ along with the original system of equations $\boldsymbol{F}(\boldsymbol{x})$ where t is the homotopy parameter. The idea here is to solve a series of problems as t increases from 0 to 1 to obtain the solutions of $\boldsymbol{F}(\boldsymbol{x})$ by following the solution paths starting from $\boldsymbol{G}(\boldsymbol{x})$ [1,2]. Interval-Newton methods involve finding intervals of the solution space where solutions can exist with mathematical guarantee [2,3].

Newton's methods [4] or Newton-type methods are iterative techniques that generate increasingly improved approximations of the root with each iteration. These methods have a faster rate of convergence but there are a few issues associated with Newton-type methods [5], e.g., i) Newton-type methods require computation of the Jacobian while some require the inverse of the Jacobian

as well which may be computationally expensive to obtain for large systems of equations at every iteration; ii) This may also lead to numerical instabilities with convergence if the Jacobian is very close to zero; iii) The convergence of such methods also depends upon the initialization. It is difficult to know apriori if a particular initialization converges to a root within a desired neighborhood. If initialized at a point which is far away from a root, convergence to that desired root may not be obtained.

Several other algorithms have been developed which do not exactly fall under any one of these categories but are nevertheless intelligent algorithms to find solutions to systems of equations. Grosan and Abraham [6] developed an evolutionary algorithm where they convert the system of equations into a multi-objective optimization problem and use pareto dominance relationships to find solutions. Ramos and Vigo-Aguiar [5] developed an approach which involves replacing a non-linear equation with 2 associated equations with the idea being that the system of associated equations is easier to solve than the original equation using Newton's methods. Ramos and Monteiro [7] further developed this approach to solve systems to non-linear equations.

This work aims to propose and study the GEMS framework which we have developed to solve systems of equations. GEMS stands for Guaranteed Error-bounded Modeling of Surrogates which involves replacing each of the original C^2-continuous equations by surrogates. The maximum error between these equations and their respective surrogates is bounded by an error metric, ϵmax, which can be derived through the theory of guaranteed estimators and using interval arithmetic methods [8]. One of the key features of the GEMS framework is that it does not depend on the Jacobian of the system of equations but does require a bound on the diagonal elements of the Hessian. Another key feature is that is allows the user to define domain bounds to selectively search the space for solutions. Lastly, it defines a convergence criterion which ensures that a solution is found.

The contents of this work are organized as follows: the methodology section introduces the overall flow of the framework along with the theoretical requisites. In the next section, we present case studies to demonstrate the applicability of the framework. The computational results are reported in the next section, finally followed by the conclusions.

METHODOLOGY

We propose the GEMS framework to solve systems of equations with one of its key ideas being that it can check whether a solution exists in a search space using sampled data, the error metric ϵmax, and interval arithmetic. From the point of view of the user, GEMS takes in input data in the form of the systems of equations $\boldsymbol{F}(\boldsymbol{x})$ and the global domain bounds and returns a solution to $\boldsymbol{F}(\boldsymbol{x})$. When provided with a system of equations and the global domain bounds, the framework first checks for the existence of a solution in the global search space. If a solution can exist, the search space (domain) is then divided into two subproblems which are explored. Exploration of a subproblem involves checking for the existence of a solution which, if exists, necessitates the evaluation of the fitness for the subproblem. The domain of the subproblem with the best fitness is then set as the domain to be explored next. This is done iteratively until a solution is found. A high-level flow of the GEMS framework is shown in Figure 1.

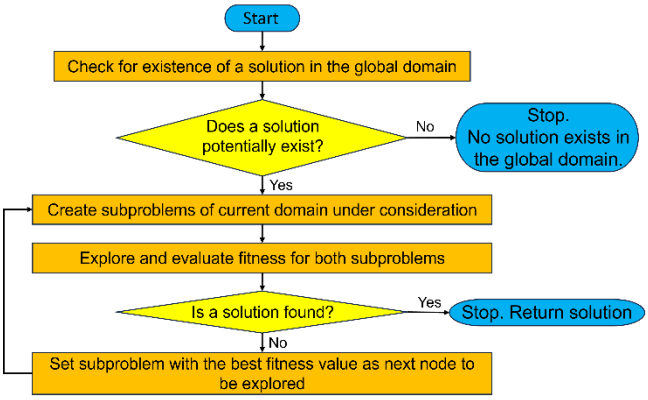

Figure 1: Overall flow of the GEMS framework.

The subsequent subsections explain the thoeries and subroutines that are utilized in the GEMS framework.

Data Sampling and Generation of Convex Hull Surrogates

GEMS replaces the original system of equations with convex hull surrogates. Given a search space i.e., a bounded domain, the functions are evaluated (sampled) at the domain bounds so that the set of points used to generate the convex hull can cover even the extremities of the search space. An N-dimensional space has 2^N vertices. The number of vertices increases drastically as N increases thereby making it impractical to have a function evaluation at every vertex for higher dimensional problems. Caratheodory's theorem states that given an N-dimensional space, if any point x lies within a convex hull defined by a set of points, then it is possible to express x as a convex combination of at most $N+1$ points in that set of points [9]. In other words, $N+1$ points are sufficient to generate a convex hull surrogate in any domain. When obtaining $\boldsymbol{F}(\boldsymbol{x})$ is expensive, this theorem can be used to reduce the number of required function evaluations. We consider both cases where 2^N and $N+1$ points are used to generate the convex hull surrogates. To illustrate, the 2-D functions $f_1(x,y) = x^2 + y^2 - 25$ and $f_2(x,y) = (x-6)^2 + y^2 - 9$ with their convex hull

surrogates $c_1(x,y)$ and $c_2(x,y)$ which are constructed using the vertex evaluations are shown in Figure 2.

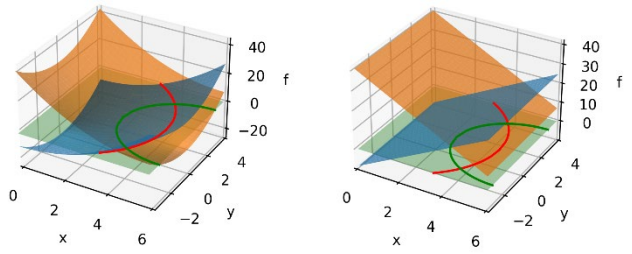

Figure 2: $f_1(x,y)$ (blue) and $f_2(x,y)$ (orange) shown in a given domain (left). Their respective convex hulls $c_1(x,y)$ (blue) and $c_2(x,y)$ (orange) shown over the same domain (right). The red and green curves represent the zeros of $f_1(x,y)$ and $f_2(x,y)$ respectively.

Interval Arithmetic

We use our in-house subroutine to compute Hessian bounds for a C^2-continuous function in a closed form involving the standard arithmetic, trigonometric, logarithmic, and exponential operators. We first generate the computational graph for a given function, and then implement interval arithmetic rules in all nodes of the graph to efficiently compute bounds on the output. An automatic differentiation type scheme is then used to compute bounds on the gradients. The computational graph that corresponds to the gradient is dynamically created to store all dependencies. Using these graphs, we then automatically compute bounds on the Hessian. A key advantage of this approach is the guaranteed estimation of Hessian bounds that can not be achieved using pointwise evaluation techniques. In some cases, we also use interval arithmetic on the computational graph to compute a lower bound, $lbmin$ and upper bound, $ubmax$ for each function in the case where enough samples aren't available.

Guaranteed Estimators

Now that Hessian bounds can be obtained, the concept of guaranteed estimators is utilized which is used to compute the error metric ϵmax. Namely, we use the guaranteed edge-concave underestimator [10], $L(x)$ defined as:

$$L(x) = f(x) - \sum_{i=1}^{N} \theta_i^L \left(x_i - x_i^G\right)^2 \qquad (1)$$

where $\theta_i^L = max\left\{0, \left[\frac{d^2f}{dx^2}\right]^U\right\}$

The edge-concave definition for a function f is equivalent to:

$$\frac{d^2f}{dx^2} \leq 0 \qquad \forall i = 1, \ldots . N \qquad (2)$$

For an edge concave function, $\theta_i^L = 0 \; \forall \; i = 1, \ldots . N$

Similarly, we use the edge-convex overestimator, $U(x)$ defined as [11]:

$$U(x) = f(x) + \sum_{i=1}^{N} \theta_i^U \left(x_i - x_i^G\right)^2 \qquad (3)$$

where $\theta_i^U = min\left\{0, -\left[\frac{d^2f}{dx^2}\right]^L\right\}$

The edge-convex definition for a function f is equivalent to:

$$\frac{d^2f}{dx^2} \geq 0 \qquad \forall i = 1, \ldots . N \qquad (4)$$

For an edge convex function, $\theta_i^U = 0 \; \forall \; i = 1, \ldots . N$

x_i^G is the point from which the estimator is generated. The estimators are illustrated in Figure 3.

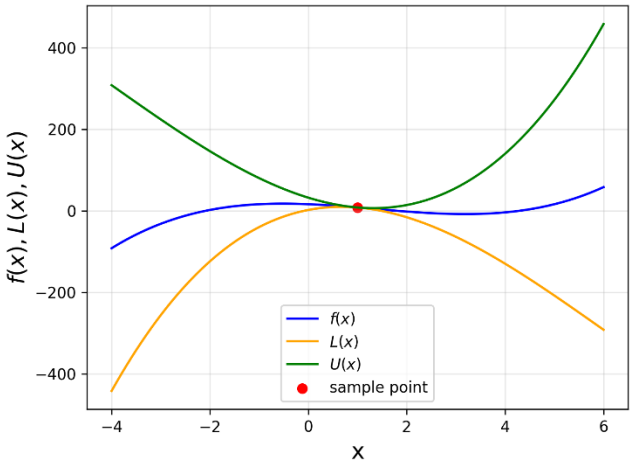

Figure 3: The function $f(x)$ shown along with its guaranteed underesimator $L(x)$ (shown in orange) and its guaranteed overestimator $U(x)$ (shown in green) generated from the sample point (shown in red).

Error-boundedness

ϵmax is the maximum possible error between the equations and their respective surrogates and therefore it is desired to have the smallest ϵmax as possible. If the closed form of the equations is convoluted in the sense that it contains nested terms, interval arithmetic may result in extremely large (loose) values of ϵmax.

Based on the nature of the function i.e., based on its θ_i^L and θ_i^U values, each of the functions in the original system can be classified into one of three categories:

i) Category 1 - The function is neither edge-convex nor edge-concave in all dimensions,
ii) Category 2 - The function is both edge-concave and edge-convex in all dimensions,
iii) Category 3 - The function is edge-convex or

edge-concave in some but not all dimensions.

The value for ϵmax for each of these categories when we consider the set of all 2^N vertices and then a subset of $N+1$ of those 2^N are shown in Tables 1 and 2 respectively.

Table 1: ϵmax values when all 2^N vertices are evaluated.

Category 1	$\epsilon max = \sum_{i=1}^{N} (\theta_i^L + \theta_i^U)(x_U^i + x_M^i)$
Category 2	$\epsilon max = \max\{f_{eval}\} - \min\{f_{eval}\}$
Category 3	$\epsilon max = f_{IAmax} - f_{IAmin}$

Table 2: ϵmax values when $N+1$ vertices are evaluated.

Category 1	$\epsilon max = \sum_{i=1}^{N} (\theta_i^L + \theta_i^U)(x_U^i + x_M^i)$
Category 2	$\epsilon max = f_{IAmax} - f_{IAmin}$
Category 3	$\epsilon max = f_{IAmax} - f_{IAmin}$

Here, $\max\{f_{eval}\}$ is the maximum of the vertex function evaluations, $\min\{f_{eval}\}$ is the minimum of the vertex function evaluations, f_{IAmax} is the upper bound on the function obtained through interval arithmetic and f_{IAmin} is the lower bound on the function obtained through interval arithmetic.

Solving the system of surrogates

Now that we have our system of error-bounded convex hull surrogates, we wish to find a point x^* in the search space such that the distance τ between furthest the convex hull surrogate and the zero "plane" is minimized. The following linear programming (LP) model CH enables us to do that:

$$\min \tau$$

$$s.t. \qquad x_n = \sum_{v \in cv} \lambda_{j,v} \hat{x}_{v,n} \quad \forall\, n \in N, j \in J \qquad (5)$$

$$CH_j = \sum_{v \in cv} \lambda_{j,v} f_j(\hat{x}_{v,n}) \quad \forall\, j \in J \qquad (6)$$

$$\sum_{v \in cv} \lambda_{j,v} = 1 \quad \forall\, j \in J \qquad (7)$$

$$CH_j \leq \tau \quad \forall\, j \in J \qquad (8)$$

$$-CH_j \leq \tau \quad \forall\, j \in J \qquad (9)$$

$$0 \leq \lambda_{j,v} \leq 1 \quad \forall\, j \in J, v \in V \qquad (10)$$

In this formulation, Equations 5, 6 and 7 are referred to as the convex combination constraints. Equations 7 and 10 ensure that the point lies inside the domain under consideration. Here, J is the set of functions and V is the set of vertices.

Note that, $\hat{x}_{v,n}$ is the known co-ordinates of vertex v in dimension n and $f_j(\hat{x}_{v,n})$ is the value of the function evaluation for a function j at the point $\hat{x}_{v,n}$. $\lambda_{j,v}$ is the weight assigned to vertex v for the function j.

Solving this formulation, we obtain a point $(\boldsymbol{x}^*)$ at which we evaluate fitness defined as follows:

$$Fitness = \sum_{j=1}^{N} f_j^2(\boldsymbol{x}^*)$$

This fitness allows us to define a metric for quantifying the "closeness to zero" or quality of the obtained solution. It is desirable to have a fitness function value close to zero.

Branch and Prune

With the theoretical requisites established, GEMS involves a branch and prune subroutine (which is inspired by the classical branch and bound) where, given an initial search space, the idea is to repeat the following steps iteratively:

1) Prune regions of the search space based of the existence of a solution in that space.
2) Branching to generate two subproblems if a solution could exist.
3) Choose subproblem with the best (smallest) fitness value to be explored next.

Each subproblem in the branch and prune tree is also called a node. The current node is branched at the midpoint of the domain of the variable having the longest edge. For systems of equations where the domain bound ranges are unequal, we scale the domain bounds relative to the global domain bounds and branch at the midpoint of the domain of the variable having the scaled longest edge. To check whether a solution exists in each search space (each node), the GEMS framework does the following: for each function, we perform a loose pruning check which involves computing a lower bound, $lbmin$ and an upper bound, $ubmax$ based solely using interval arithmetic. For any function, if its lower and upper bounds have the same sign, it is not possible for that function to have a solution in that space. Each function has a distinct lower and upper bound. If any of the functions have $lbmin$ and $ubmax$ of the same sign, it is impossible for the entire system of equations to have a solution in that space, thereby allowing us to discard that node from further consideration. Note that, we are using the $lbmin$ and $ubmax$ based solely on IA which are loose. However, this check helps eliminate unnecessary computation. We move to the next step of node exploration if a solution can exist in the node i.e., if it cannot be pruned based on the loose pruning condition. We then perform function evaluations and Hessian bound estimations to obtain tighter $lbmin$ and $ubmax$ values for another tighter pruning check. Figure 4 shows the node exploration subroutine. The $lbmin$ and $ubmax$ for each of the categories when we consider the set of all 2^N vertices are

shown in table 3. Similarly, the values of $lbmin$ and $ubmax$ when we consider a subset of $N+1$ vertices are shown in table 4. Note that for functions which are both edge-concave and edge-convex in all dimensions (Category 2), when we have function evaluations at all 2^N vertices, Tardella's theorem [12,13] enables us to obtain tightest $lbmin$ and $ubmax$.

Table 3: $lbmin$ and $ubmax$ values when all 2^N vertices are considered.

Category 1	$lbmin = \min\{f_{eval}\} - \epsilon max$ $ubmax = \max\{f_{eval}\} + \epsilon max$
Category 2	$lbmin = \min\{f_{eval}\}$ $ubmax = \max\{f_{eval}\}$
Category 3	$lbmin = f_{IAmin}$ $ubmax = f_{IAmax}$

Table 4: $lbmin$ and $ubmax$ values when all $N+1$ vertices are considered.

Category 1	$lbmin = \min\{f_{eval}\} - \epsilon max$ $ubmax = \max\{f_{eval}\} + \epsilon max$
Category 2	$lbmin = f_{IAmin}$ $ubmax = f_{IAmax}$
Category 3	$lbmin = f_{IAmin}$ $ubmax = f_{IAmax}$

If $lbmin$ and $ubmax$ have different signs, a solution may possibly exist, and we move to the next step of node fitness evaluation. The node fitness evaluation involves generation of the convex hull surrogates and using the LP formulation to find the point at which we evaluate fitness. If the fitness of a node falls below a certain threshold (<1e-4) we have found a point for which the value of each function in $\boldsymbol{F}(\boldsymbol{x})$ is extremely close to zero which basically means we have found a solution to the original system of equations.

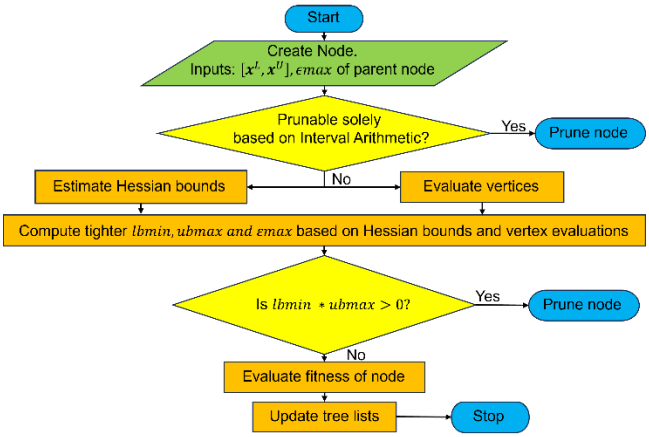

Figure 4: The node exploration subroutine.

Convergence

As ϵmax reduces, the error between the function and its convex hull surrogate reduces. The error metric ϵmax is dependent on the Hessian bounds and domain sizes which means the ϵmax of a child node can never be greater than the ϵmax of its parent. In other words, the error-bound for any subproblem is less than or equal to its parent problem. The sum of all $\epsilon maxs$ for the child node will always be greater than the sum of $\epsilon maxs$ of the parent node. It can be considered an equivalent to the gap in the classical branch and bound technique used in optimization. Also, having a small enough value of ϵmax can lead to efficient pruning of sub-regions.

This concludes the overall flow of the GEMS framework.

CASE STUDIES

We consider some test problems having a single solution as well as a few benchmark problems from literature along with examples of systems of equations arising from test problems in chemical engineering literature.

Test problems

1) Hypersphere systems – The following system of equations can be scaled to N-dimensions while still maintaining the characteristic of having a single solution. We consider the 10, 50 and 100-dimensional systems for our studies. Here, R is the radius of the hyperspheres.

$$\boldsymbol{F_{HSN}}(\boldsymbol{x}) = \begin{bmatrix} \sum_{i=1}^{N} x_i^2 - \mathrm{R} = 0 \\ (x_1^2 - 2\mathrm{R})^2 + \sum_{i=2}^{N} x_i^2 - \mathrm{R} = 0 \\ \sum_{\substack{i=1 \\ i \neq j-1}}^{N} x_i^2 + \left(x_j - \mathrm{R}\right)^2 - \mathrm{R} = 0, \forall\, j = (3, \ldots., N) \end{bmatrix}$$

Benchmark systems in literature

1) Interval Arithmetic [14] – The following system has been considered as one of the benchmark problems from interval arithmetic:

$$\boldsymbol{F_{IA}}(\boldsymbol{x}) = \begin{bmatrix} x_1 - 0.25428722 - 0.18324757x_4x_3x_9 = 0 \\ x_2 - 0.37842197 - 0.16275449x_1x_{10}x_6 = 0 \\ x_3 - 0.27162577 - 0.16955071x_1x_2x_{10} = 0 \\ x_4 - 0.19807914 - 0.15585316x_7x_1x_6 = 0 \\ x_5 - 0.44166728 - 0.19950920x_7x_6x_3 = 0 \\ x_6 - 0.14654113 - 0.18922793x_8x_5x_{10} = 0 \\ x_7 - 0.42937161 - 0.21180486x_2x_5x_8 = 0 \\ x_8 - 0.07056438 - 0.17081208x_1x_7x_6 = 0 \\ x_9 - 0.34504906 - 0.19612740x_{10}x_6x_8 = 0 \\ x_{10} - 0.42651102 - 0.21466544x_4x_8x_1 = 0 \end{bmatrix}$$

2) Neurophysiology Application [15] – The

following system arises from an application in neurophysiology:

$$\boldsymbol{F_N(x)} = \begin{bmatrix} x_1^2 + x_3^2 = 1 \\ x_2^2 + x_4^2 = 1 \\ x_5 x_3^2 + x_6 x_4^3 = c_1 \\ x_5 x_1^3 + x_6 x_2^3 = c_2 \\ x_5 x_1 x_3^2 + x_6 x_2 x_4^2 = c_3 \\ x_5 x_3 x_1^2 + x_6 x_4 x_2^2 = c_4 \end{bmatrix}$$

where the constants c_i can be arbitrarily chosen. We consider all $c_i = 0$.

Systems arising form process engineering models

1) Combustion Application [16] – The following system arises from the combustion of propane in air. The stoichiometric equation is:

$$C_3H_8 + \frac{R}{2}(O_2 + 4N_2) \rightarrow Products$$

where R is the relative amounts of air to fuel. Combustion Application – The following system arises from the combustion of propane in air. The variables x_i for $i = 1$ to 10 are the number of moles of product i formed per every mole of propane combusted.

$$\boldsymbol{F_{CA}(x)} = \begin{bmatrix} x_1 + x_4 = 3 \\ 2x_2 + 2x_5 x_6 x_7 = 8 \\ 2x_3 + x_9 = 40 \\ 2x_1 + x_2 + x_4 + x_7 + x_8 + x_9 + 2x_{10} = 10 \\ x_1 x_5 = K_5 x_2 x_4 \\ \sqrt{40} x_6 x_1^{0.5} = K_6 \sqrt{x_2 x_4 N_{Tot}} \\ \sqrt{40} x_7 x_4^{0.5} = K_7 \sqrt{x_1 x_2 N_{Tot}} \\ 40 x_8 x_4 = K_8 x_1 N_{Tot} \\ \sqrt{40} x_9 x_4 = K_9 x_1 \sqrt{x_3 N_{Tot}} \\ 40 n_{10} n_4^2 = K_{10} n_1^2 N_{Tot} \end{bmatrix}$$

where,

$$N_{Tot} = \sum_{i=1}^{10} x_i$$

The first for equations in the system arise form the mole balances while the rest arise from the equilibrium relations. K_i for $i = 5$ to 10 are the equilibrium constants.

2) Chemical Equilibrium [17] – The mathematical structure of the combustion application system is analysed and reformulated by using the notion of 'element variables' (surrogates for atomic combinations). The first four equations arise from the reformulation of the mole balances in the combustion application system whereas the equilibrium relations reduce to a single equation because of the element variable substitutions which leads to the following system:

$$\boldsymbol{F_{CEQ}(x)} = \begin{bmatrix} x_1 x_2 + x_1 - 3x_5 = 0 \\ 2x_1 x_2 + x_1 + x_2 x_3^2 + R_8 x_2 - R x_5 \\ +2R_{10} x_2^2 + R_7 x_2 x_3 + R_9 x_2 x_4 = 0 \\ 2x_2 x_3^2 + 2R_5 x_3^2 - 8x_5 \\ +R_6 x_3 x_4 + R_7 x_2 x_3 = 0 \\ 2x_4^2 - 4R x_5 + R_9 x_2 x_4 = 0 \\ x_1(x_2 + 1) + R_{10} x_2^2 + x_2 x_3^2 + R_8 x_2 \\ +R_5 x_3^2 + x_4^2 - 1 + R_6 x_3 + R_7 x_2 x_3 \\ +R_9 x_2 x_4 = 0 \end{bmatrix}$$

Note that the variables x_i for $i = 1$ to 5 have been redefined along with the constant R for $i = 5$ to 10 to account for the element variable substitutions.

3) Recycle and Purge in the Synthesis of Ammonia –The ammonia synthesis process involving a recycle and a purge stream in considered. Given the feed stream specifications and the reactor conversion, it is desired to calculate the overall conversion of nitrogen and the ratio of moles of gas purged to the moles of gas leaving the condenser.

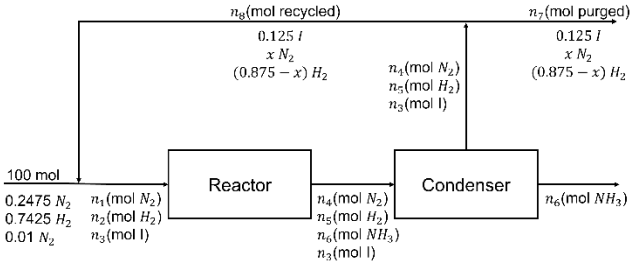

Figure 4: Flowsheet for the ammonia synthesis process.

The following system of equations arises from the overall mole balances, stoichiometric relations, and material balances at the mixing and split points.

$$\boldsymbol{F_{AS}(x)} = \begin{bmatrix} 1 - 0.125 n_6 = 0 \\ 148.5 - 3n_5 - 1.75 n_6 + 2n_6 x = 0 \\ 49.5 - 2n_6 x - n_5 = 0 \\ n_6 - 0.75 n_1 = 0 \\ 0.5 n_1 - n_5 = 0 \\ n_6 - x n_7 - n_4 x = 0 \\ 74.25 + 0.875 n_7 - x n_7 - n_2 = 0 \\ 1 + 0.125 n_7 - n_3 = 0 \end{bmatrix}$$

The domain ranges considered for each of these problems are shown in Table 5.

Table 5: Domain bounds for the test problems

Problem Instance	Domain Ranges
10D-hyperspheres	$x_i \in [-5,5]\ \forall\ i = 1, \ldots 10$

50D-hyperspheres	$x_i \in [-5,5]\ \forall\ i = 1, \ldots 50$
100D-hyperspheres	$x_i \in [-5,5]\ \forall\ i = 1, \ldots 100$
Interval Arithmetic	$x_i \in [-4,4]\ \forall\ i = 1, \ldots .10$
Neurophysiology Application	$x_1 \in [-10,5]$ $x_2 \in [-10,6]$ $x_3 \in [-10,7]$ $x_4 \in [-10,8]$ $x_5 \in [-10,9]$ $x_6 \in [-10,9]$
Combustion Application	$x_i \in [-5,5]\ \forall\ i = 1, \ldots 10$
Chemical Equilibrium	$x_i \in [-5,5]\ \forall\ i = 1, \ldots 5$
Ammonia Synthesis	$n_i \in [0,500]\ \forall\ i = 1, \ldots .7$ $x \in [0,1]$

COMPUTATIONAL EXPERIMENTS AND RESULTS

All runs were performed on an Intel Xeon Gold 6248R 3GHz processor running on Linux (CentOS 7). The algorithm was developed in Python v3.6.8 and used CPLEXv20.1.0.1 in GAMSv35.1 to solve the LP model CH for the system of surrogates.

Table 6: Fitness of solution obtained, number of nodes explored and number of function evaluations for the test problems.

Instance	Fitness of Solution	Nodes Explored	Function Evaluations
10-D hyperspheres	7e-5	228	1487
50-D hyperspheres	1e-4	1278	35827
100-D hyperspheres	4e-5	813	51713
Interval Arithmetic	1e-5	64.48	1823
Neurophysiology Application	6e-5	307	1301
Combustion Application	1e-4	21650	97856
Chemical Equilibrium	8e-5	837	3149
Ammonia Synthesis	1e-8	383	54483

We have reported the solutions for each of the test problems using a sampled data set of $N + 1$ vertices except for the system of equations arising from the ammonia synthesis model where we have used 2^N vertices. The solution of the LP (to minimize the distance to the furthest convex hull surrogate) highly depends on choice of the $N + 1$ sampled vertices. In other words, the choice of the set of the sampled data affects the path of the branch and prune search.

CONCLUSIONS

In this work, we report a data-driven branch and prune approach that has the potential to unlock new avenues for finding solutions to systems of equations without the requirement of an initial guess. Furthermore, the user can specify the domain bounds if solutions within a desired search space are required. Multiple potential sampling strategies need to be investigated which may aid in faster convergence. Further work is needed to reduce the number of required function evaluations.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge support from the NSF CAREER (CBET-1943479), and the DOE AIChE RAPID Institute (DE-EE0007888-09-03) grants. Part of the research was conducted with the computing resources provided by Texas A&M High Performance Research Computing.

REFERENCES

1. Wayburn, T. L., & Seader, J. D. Homotopy continuation methods for computer-aided process design. Computers & Chemical Engineering, 11(1), 7-25. (1987)
2. Maranas, C. D., & Floudas, C. A. Finding all solutions of nonlinearly constrained systems of equations. Journal of Global Optimization, 7, 143-182. (1995)
3. Hansen, E. R., & Greenberg, R. I. An interval Newton method. Applied Mathematics and Computation, 12(2-3), 89-98. (1983)
4. Remani, C. Numerical methods for solving systems of nonlinear equations. Lakehead University Thunder Bay, Ontario, Canada, 77. (2013)
5. Ramos, H., & Vigo-Aguiar, J. The application of Newton's method in vector form for solving nonlinear scalar equations where the classical Newton method fails. Journal of computational and applied mathematics, 275, 228-237. (2015)
6. Grosan, C., & Abraham, A. A new approach for solving nonlinear equations systems. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(3), 698-714. (2008)
7. Ramos, H., & Monteiro, M. T. T. A new approach based on the Newton's method to solve systems of nonlinear equations. Journal of Computational and Applied Mathematics, 318, 3-13. (2017)
8. Alefeld, G., & Mayer, G. Interval analysis: theory and applications. Journal of computational and

applied mathematics, 121(1-2), 421-464. (2000)
9. Floudas, C. A. Deterministic global optimization: theory, methods and applications (Vol. 37). Springer Science & Business Media. (2013)
10. Hasan, M. M. F. An edge-concave underestimator for the global optimization of twice-differentiable nonconvex problems. Journal of Global Optimization, 71(4), 735-752. (2018)
11. Iftakher, A., Aras, C. M., Monjur, M. S., & Hasan, M. F. Data-driven approximation of thermodynamic phase equilibria. AIChE Journal, 68(6), e17624. (2022)
12. Tardella, F. On a class of functions attaining their maximum at the vertices of a polyhedron. Discrete applied mathematics, 22(2), 191-195. (1988)
13. Tardella, F. On the existence of polyhedral convex envelopes (pp. 563-573). Springer US. (2004)
14. Moore, R. E. Methods and applications of interval analysis. Society for Industrial and Applied Mathematics. (1979)
15. Verschelde, J., Verlinden, P., & Cools, R. Homotopies exploiting Newton polytopes for solving sparse polynomial systems. SIAM Journal on Numerical Analysis, 31(3), 915-930. (1994)
16. Shacham, M., Brauner, N., & Cutlip, M. B. A web-based library for testing performance of numerical software for solving nonlinear algebraic equations. In Computer Aided Chemical Engineering (Vol. 9, pp. 291-296). Elsevier. (2001)
17. Meintjes, K., & Morgan, A. P. A methodology for solving chemical equilibrium systems. Applied Mathematics and Computation, 22(4), 333-361. (1987)