1st AIAA Multidisciplinary Design Optimization Specialist Conference, 18 - 21 April 2005, Austin, Texas

Application of an Interior Point Sequential Approximate Optimization Method to an Autonomous Underwater Vehicle

Neal M. Patel* Shawn E. Gano† John E. Renaud‡

University of Notre Dame, Notre Dame, Indiana, 46556-5637, U.S.A.

Victor M. Peréz§

General Electric CIAT, Santiago de Querétaro, Querétaro, 76030, México

Jay D. Martin¶

Applied Research Laboratory, State College, PA 16804, U.S.A.

Simulation based design has become a major tool in the design of automotive, aerospace and consumer products. Designers are faced with the continuous challenge of reducing manufacturing costs and design cycle times while improving the systems performance and reliability. Simulation based design plays an increasingly prominent role in facilitating the conceptualization and realization of products under these competitive conditions. Single discipline simulations used for analysis are being coupled together to create complex coupled simulation systems. The computational cost of executing a single complex coupled simulation make these problems very expensive for optimization. In some cases, the engineer has to design under a time constraint and therefore the optimization process may be terminated prematurely. For this case an algorithm that provides a feasible design at each iteration would be desirable. In this paper, we will give a brief description of the AUV simulation model as well as detail the use of an Interior Point Trust-Region Sequential Approximate Optimization (IP-TR-SAO) method to an Autonomous Underwater Vehicle Simulation that is both computationally expensive and noisy; an ideal application of the IP-TR-SAO framework.

Nomenclature

$\mathbf{A}$	Gradient matrix of active constraints.
b_i	Relaxation level for ith constraint.
f	Objective function.
$\mathbf{g}$	Inequality constraint vector.
g_i	i^{th} inequality constraint.
G	Relaxed inequality constraint vector.
$\mathbf{h}$	Equality constraint vector.
$\mathbf{H}$	Relaxed equality constraint vector.
L	Lagrangian function.

*Graduate Student, Aerospace and Mechanical Engineering, e-mail: npatel@nd.edu, Student Member AIAA.
†Graduate Student, Aerospace and Mechanical Engineering, e-mail: sgano@nd.edu, Student Member AIAA.
‡Professor, Aerospace and Mechanical Engineering, e-mail: John.E.Renaud.2@nd.edu, Associate Fellow AIAA.
§Ph.D, GE Transportation, Aircraft Engines, victor.perez@ae.ge.com, Member AIAA.
¶Research Assistant, Manufacturing Systems, P.O. Box 30, jdm111@psu.edu, Member AIAA.

r	Penalty parameter.
y	Vector of states.
$\mathbf{x}$	Vector of design variables.
x^k	Design point at k^{th} iteration.
f	Response surface approximation to f.
Δ	Trust region radius.
Λ	Vector of Lagrange multipliers.
Φ	Augmented Lagrangian function.
Ψ_i	Alternative form for inequality constraint g_i.
ρ	Trust region ratio.
τ	Relaxation parameter.

I. Introduction

A torpedo dynamics simulation model was developed as part of a University of Notre Dame-Penn State Applied Research Laboratory (ARL) collaboration. The ARL had originally developed an unclassified two variable design problem for an Autonomous Underwater Vehicle (AUV) that proved to be overly simplistic. Therefore a model that includes a dynamics model of a torpedo was developed, using the ARL model for system analysis calls. The simulation model includes a six-degree of freedom dynamics model of a simple underwater vehicle, which referred to as the AUV, modeled in Mathworks' SIMULINK which is executed from MATLAB. This model includes added mass and cross terms as well as cross-flow drag to accurately model the nonlinear dynamics. The model is one big feedback loop that is driven by the evading vehicle's position relative to the pursuing vehicle and the subsequent pursuit.

To make the model more complex, simple differential game theory is integrated into the simulation by incorporating a pursuit game. The pursuer is the modeled vehicle that uses a simple autonomous controller to follow a target. This target is the evader, a non-dynamically modeled, unintelligent vehicle that travels on a programmed path. This vehicle can be launched at different orientations and depths. The pursuing vehicle first attempts to orient itself in the direction of the target to get a lock on it using its sonar. Subsequently, the vehicle travels towards the target, while updating the expected target location and employing strategy along the way using the sonar information. A mission is deemed successful when the pursuer travels within a set radius of the evading craft. The idea behind this game theory approach is that a probability of success of a given AUV over a range of missions can be determined. Furthermore, a set of design metrics can be measured, a design objective can be computed, and a merit function can be computed.

The model is coupled with the ARL's AUV problem as a system analysis before each set of simulations to determine the torpedo design parameters and sizings. Specifically, weights and available power of subsystems and components, efficiencies based on propulsor types, sonar configurations, and various other subsystems can be optimized.

Computing the probability that the AUV is successful requires the simulation to be run many times and therefore is computationally expensive. The framework makes use of a new adaptive experimental design (AED) approach for meta-modeling. The novel AED meta-modeling technique reduces from $O(n^2)$ to $O(n)$ the amount of data needed to construct meta-models. This is particularly important for the conceptual design optimization of systems that involve very large simulation codes such as the AUV. A new interior point approach for trust region managed sequential meta-model optimization has been developed to insure that feasibility is maintained throughout the optimization process. This facilitates the delivery of a consistent and feasible design when subject to reduced design cycle time constraints. In order to deal with infeasible starting points which will be the case for the example AUV problem, probability one homotopy methods will be used to relax constraints and push designs toward feasibility.

In this paper an overview of the IP-TR-SAO framework[12] is given, followed by a description of the AUV problem. Then the IP-TR-SAO framework is applied to the AUV problem; the results are compared to solving the AUV problem using sequential quadratic programming, a standard optimization method.

II. IP-TR-SAO Methodology

In this section, an approximate interior point trust region based sequential approximate optimization algorithm is presented.[12] The method produces a feasible design (for the surrogate problem) after each approximate optimization. The development of the algorithm has a two-fold purpose. The first is to generate approximately feasible iterates along the optimization path so the designer can stop the algorithm at any point and obtain a usable design. The second is to speed up convergence, helping the minimization subproblems locate the true minimum by including explicitly the constraints. The main idea of the algorithm is simple. If a (surrogate) feasible starting point is given, the algorithm should give a (surrogate) feasible point at each iteration. Though this sounds like a simple requirement to comply with, it is a very tough problem for general nonlinear optimizers. The problem is that requiring feasibility for the iterates forces the algorithm to move along the constraints. The use of line searches makes the algorithm move between feasibility and infeasibility with small steps. In SAO the problem is avoided since no line search is performed. Instead, approximations of the objective function and constraints are built, and then an optimization is performed over the approximations. The optimizer is free to move over the local design space instead of being restricted to a search direction. The step size depends on how good the approximations are.

The proposed algorithm is inspired by the trust region augmented Lagrangian framework of Rodriguez *et al.*[15,16] What makes the present algorithm different is that the minimization subproblem explicitly contains the approximated constraints to generate a feasible iterate. The Lagrange multipliers are updated at each successful iteration and the constraints are relaxed to handle infeasible starting points.

The algorithm assumes that the initial starting point is a (surrogate) feasible design. This assumption will help to develop the general framework. The more general case of an infeasible starting point will be handled by the use of homotopy methods to control the relaxation of the constraints.[12]

A. Sequential Approximate Optimization

A SAO framework is intended to solve the constrained optimization problem of the form:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \geq 0, \\ & \mathbf{h}(\mathbf{x}) = 0, \\ & \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}, \end{aligned} \tag{1}$$

The main idea of the SAO is to build simple response surface approximations of the objective function and constraints, valid for a local region. An optimization is performed over this approximation within the local limits. The approximation is updated every iteration until convergence is achieved.

The algorithm begins at any iteration k with a starting point $\mathbf{x}_k$. The objective function, constraints and their gradients are evaluated in the design characterization step. Then local move limits are defined. The move limits define the region where the approximation will be valid. Within this move limits a database is constructed. Once a RSA is constructed using the information in the database, an optimization is performed over the approximations. The new candidate point is evaluated and either accepted or rejected. After this new move limits are set and the optimization goes on. A flowchart of the general sequential approximate optimization is shown in Figure 2.

In the sequential approximate optimization strategies of Wujek *et al.*[25,26] and Rodríguez *et al.*,[15,16] which were modified for DOE based sampling by Peréz *et al.*,[10,11] the framework constructs second order response surface approximations of the objective function and constraints within a local trust region. The approximations are constructed using exact first order information at the current design point. Therefore the work of constructing a response surface involves only the fitting of the second order terms which are referred to as the Hessian-RS matrix H or simply Hessian.

Alexandrov *et al.*[2] proved the convergence of a SAO framework in which the move limits are managed by a trust region approach and the local approximation matches the function and the gradient at the current design point. The trust region augmented Lagrangian framework implemented in this study has this characteristics and is proved to converge.[15] This framework uses an augmented Lagrangian approach for driving the optimization and a trust region methodology to manage the move limits.

In the present research the trust region augmented Lagrangian framework of Rodríguez *et al.*[15] is implemented. For a better understanding of the present investigation, a more detailed description of the response surface approximation, the trust region approach and the database generation is presented.

B. The Response Surface Approximation

The quadratic function approximation used in the framework for f at the k^{th} iteration is given by:

$$\tilde{f}^k(\mathbf{x}) = f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T \Delta\mathbf{x}^k + \frac{1}{2}\Delta\mathbf{x}^k \mathbf{H}_f^k \Delta\mathbf{x}^k, \tag{2}$$

where $\Delta\mathbf{x}^k = \mathbf{x} - \mathbf{x}^k$. Both $f(\mathbf{x}^k)$ and $\nabla f(\mathbf{x}^k)$ are known and only the second order information $\mathbf{H}_f^k$ is computed. Gradient information can be obtained in analytical form provided by the user, or computed via finite differences or the global sensitivity equations[20] in the case of a MDO problem.

It is worth to note here the difference between the true Hessian of the function and the Hessian-RS of the approximation $\tilde{f}^k$ denoted here as $\mathbf{H}_f^k$. The true Hessian is the second order information evaluated at a given design point, in this case $\mathbf{x}^k$. The Hessian-RS or simple Hessian, as referred to in this paper, constructs a quadratic approximation of the function in the local region. The true Hessian is never computed nor used in this SAO framework. It is too expensive to compute by finite differences and assumed to be difficult to obtain in analytical form. Whenever the word Hessian is used in this paper, it refers to the matrix of second order coefficients in the quadratic approximation.

The total number of coefficients in a full second-order model is $1+n+n(n+1)/2$. However in the quadratic model considered here, zero and first order information are known, therefore the number of coefficients in $\mathbf{H}_f^k$ is $n(n+1)/2$ where n is the number of design variables in the problem. The minimum size of the database required to compute the approximate model (2) is therefore $n(n+1)/2$ or $O(n^2)$ in general terms. The coefficients in the Hessian matrix are computed using the least squares method.

The least squares method assumes an independent and normally distributed error in the model to be fitted. This is obviously not true for computer experiments. Sacks *et al.*[18] point out that *In the absence of independent random errors, the rationale for least-squares fitting of a response surface is not clear.* However he also indicates that *least squares can be viewed as curve fitting.* Simpson *et al.*[19] also underline the fact that in deterministic computer experiments, the model should interpolate and not smooth along the data points. In this particular application, the quadratic model is used in the curve fitting sense, within a local region controlled by the trust region methodology. The general trend is captured by the approximation. When approaching convergence, the trust region method guarantees the adequacy of the quadratic model by shrinking the move limits. Furthermore, if variable fidelity data is used,[17] smoothing along the data points is a necessity. Other methods have been proposed to compute the coefficients of the Hessian as the minimum bias estimation method.[7]

C. Trust Region Methodology

The trust region approach[1–3,5,25,26] is based on the use of a trust region ratio ρ to monitor how well the current approximation is found to represent the actual design space. Consider an unconstrained optimization problem. $\tilde{\phi}^k$ is an approximation to the function ϕ around the point $\mathbf{x}^k$ at the k^{th} iteration. The move limits are defined by the region $\|\mathbf{x} - \mathbf{x}^k\|_p \leq \Delta^k$ and the p norm defines the shape of the region. Δ^k is known as the trust region radius.

The minimization of $\tilde{\phi}^k$ subject to the defined trust region gives a new *candidate point* $\mathbf{x}_*^{k+1}$. The trust region ratio ρ is computed based on the information of the new candidate point:

$$\rho^k = \frac{\tilde{\phi}^k(\mathbf{x}_*^{k+1}) - \tilde{\phi}^k(\mathbf{x}^k)}{\phi(\mathbf{x}_*^{k+1}) - \phi(\mathbf{x}^k)}. \tag{3}$$

ρ is simply the ratio of the actual change in the function to the change predicted by the approximation. The closer the value of ρ to one, the better the approximation $\tilde{\phi}^k$ mimics the behavior in the descent direction of ϕ. The approximate minimization forces $\phi(\mathbf{x}_*^{k+1}) \leq \phi(\mathbf{x}^k)$. A negative ρ indicates a poor approximation and the new point does not decrease the function, therefore the point is rejected and the trust region radius reduced. If the value of ρ is greater than zero, the point is kept: $\mathbf{x}^{k+1} = \mathbf{x}_*^{k+1}$. The trust region radius Δ is updated according to the following principles:

$$\Delta^{k+1} = \begin{cases} c_1\Delta^k & \text{if } \rho^k < R_1 \\ c_2\Delta^k & \text{if } \rho^k > R_2 \\ \Delta^{(t)} & \text{otherwise} \end{cases} \tag{4}$$

Typical values for the limiting range constants are $R_1 = .25$ and $R_2 = .75$. The trust region multiplication factors c_1 and c_2 are commonly set to be .25 and 2 respectively.

D. Database Generation

In SAO, at every iteration a new database is computed. The points to be evaluated can be generated using an optimization based sampling as in References[14, 16, 25, 26] or using more traditional design of experiments (DOE) strategies.[10, 11, 15, 17] Each design point can be sampledevaluated by evaluatingcomputing the system analysis(SA), or in the case of multidisciplinary optimization problems, by sampling the linearly decoupled simulation codes or contributing analysis (CAs)[11, 17] generating the so called *variable fidelity* data.

Several techniques have been developed to efficiently sample the design space and generate a response surface approximation. Among the common techniques to generate an experimental design are the traditional DOE arrays such as full factorial experiments (FF), central composite designs (CCD), fractional factorials,[7] Latin hypercubes and their extensions,[6, 22] and orthogonal arrays (OA's).[4, 8] Some quality improvement computations such as D-optimality[13, 21] are too large or too complex for a SAO framework. OAs are an excellent choice for computer experiments because they are easy to generate. Owen[8] has compiled a suite of programs to generate a broad class of OA's for different number of levels and design variables. Furthermore the number of design points is relatively small. In the construction of RSAs for MDO one important consideration is the dimensionality of the problem, where large problems may impose the *curse of dimensionality*. While small problems can be easily dealt with by traditional sampling techniques (FF, CCD), as tthe number of variables increases, the complexity of the sampling does too. In the case of a second order RSA, the number of the parameters to be fitted is $O(n^2)$. FF and CCD generate design arrays with order $O(2^n)$, while some OA's have order $O(n^2)$ or even $O(n)$.

In this paper orthogonal arrays have been chosen as experimental arrays. Once the experimental array is defined for the optimization problem, the local move limits set the values for each design point. Therefore, at each iteration a complete new set of points is defined by the same experimental array.

E. Adaptive Experimental Design

The most important difference, from an experimental point of view, between traditional laboratory experiments and the computational experiments embedded in SAO, is that in the latter the experiment is repeated several times at different locations, up to convergence or stopping of the algorithm. At each new iteration a new sampling is performed of the same system but in a new sampling region. However, at each new iteration one is not completely blind about the behavior of the system, as the previous information about the nature of the response surface, and the fitted coefficients of the previous local approximation are available. Due to the highly nonlinear nature of MDO problems, the design space is not expected to have the same response from the starting sampling region to the final one, and therefore the performance of a fixed experimental

array may vary through the process. This fact was acknowledged in the research of Rodríguez *et al.*,[17] where the OA's were randomized to avoid having a fixed experimental array not capturing the true interactions of the design variables.

The main drawback for SAO is the size of the database required to construct the quadratic approximation. The number of parameters needed to fit a full second order approximation is $O(n^2)$. This problem is known as the *curse of dimensionality*. An adaptive experimental design that takes advantage of the information from the previous local approximation to modify the size of the experimental array required for the next sampling appears to hold promise for improving the efficiency of SAO algorithms. In this paper information already available from the previous approximation is used to reduce the size of the data required to construct a full quadratic model, while maintaining the quality of the approximation. As a result, the total cost of the optimization is reduced.

A reduction in the cost of the optimization can be obtained if for some iterations instead of an $O(n^2)$ sampling, only an $O(n)$ would be performed. This can be done by neglecting the off-diagonal terms in the Hessian matrix $\tilde{H}_f^k$. While this certainly reduces the order of data to be queried to $O(n)$, the quality of the approximation might decrease considerably. These neglected terms could be an important component of the second order information and thus it would end up being a poor representation of the system response. Thus, a canonical transformation forces the off-diagonal terms of a Hessian to vanish. If for a given point the canonical transformation matrix is known, then it can be assumed to be invariant for at least some iterations.

Once the full matrix of second order terms is approximated for a given design point, can be transformed into its canonical form. In the canonical space the off-diagonal components of the Hessian vanish. Once the canonical transformation matrix has been found it can be assumed, at least for some number of iterations, that the curvature of the function will be invariant, so the transformation matrix is kept. In the next iterations, in the transformed design space, only the main diagonal of the second order matrix is fitted ($O(n)$ sampling). As a result, back in the original space a full matrix is obtained. The transformation matrix can be kept as long as the curvature of the function does not change too much. In that case, the full matrix of second order terms has to be updated and a new transformation matrix is computed.This technique is called in this paper adaptive experimental design (AED).

1. *Adaptive Experimental Design Methodology*

Lets assume a single function f is being approximated. The quadratic approximation at the k^{th} iteration is represented by (2). The matrix of second order terms $\mathbf{H}_f^k$ is symmetric, hence similar to a real diagonal matrix via an orthogonal transformation matrix, whose columns are orthonormal eigenvectors. The eigenvalue decomposition of the Hessian matrix is defined as:

$$\mathbf{H}_f^k = \mathbf{U}_f^k \mathbf{D}_f^k {\mathbf{U}_f^k}^T. \tag{5}$$

The matrix of normalized eigenvectors $\mathbf{U}_f^k$ *defines the orientation of the curvature while the diagonal matrix of real eigenvalues* $\mathbf{D}_f^k$ *defines the magnitude of the curvature along the eigenvectors orientation.* This transformation and its significance is illustrated in Figure 1.
Using $\mathbf{U}_f^k$ as the transformation matrix, the design variables $\mathbf{x}$ can be transformed to the canonical variables $\mathbf{x}_U$. The canonical form of the quadratic approximation is:

$$\tilde{f}^k(\mathbf{x}_U) = f(\mathbf{x}_U^k) + \nabla f_U^T(\mathbf{x}_U^k)\Delta\mathbf{x}_U^k + \frac{1}{2}\Delta{\mathbf{x}_U^k}^T \mathbf{D}_f^k \Delta\mathbf{x}_U^k. \tag{6}$$

where

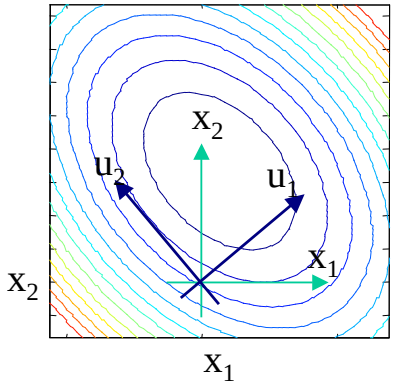

Figure 1. Transformation to canonical form.

$$\mathbf{x}_U^k = \mathbf{U}_f^{k\,T}\mathbf{x}^k, \tag{7}$$
$$\Delta\mathbf{x}_U^k = \mathbf{x}_U - \mathbf{x}_U^k, \tag{8}$$
$$f(\mathbf{x}_U^k) = f(\mathbf{x}^k), \tag{9}$$
$$\nabla f_U(\mathbf{x}_U^k) = \mathbf{U}_f^{k\,T}\nabla f(\mathbf{x}^k). \tag{10}$$
$$\tag{11}$$

Now assume that a full Hessian is known for a previous nearby iteration s, $s < k$. Its corresponding eigenvector matrix is $\mathbf{U}_f^s$. If the points x^k and x^s are close enough, it is safe to assume that $\mathbf{U}_f^k$ and $\mathbf{U}_f^s$ are similar. Therefore to compute the Hessian matrix $\mathbf{H}_f^k$ one just need to calculate the eigenvalue matrix $\mathbf{D}_f^k$ by sampling the design space with an $O(n)$ experimental array. This experimental array is sampled in the original space. Then the database is transformed to the canonical space $\mathcal{U}^s$ where the eigenvalue matrix $\mathbf{D}_f^k$ is computed. Finally a back-transformation returns the new *full* Hessian matrix in the original space:

$$\mathbf{H}_f^k = \mathbf{U}_f^s\mathbf{D}_f^k\mathbf{U}_f^{s\,T}. \tag{12}$$

The eigenvector matrix $\mathbf{U}_f^s$ can be used for several iterations. If at any point the approximation is bad, then a new eigenvector matrix is computed. This is done by constructing a full Hessian using an $O(n^2)$ experimental array. In this paper the criteria for a *bad* approximation is a failure in the trust region test as described in Section C. The trust region test fails when the new candidate point increases the value of the function. The bad approximation can be a result of a trust region too large or a bad model. Therefore a trust region reduction plus a transformation matrix update have to be performed.

In summary, the user requires two experimental arrays, one with $l_1 \geq n(n+1)/2$ ($O(n^2)$) design points and one with $l_2 \geq n$ ($O(n)$) design points. At the first iteration, the local design space defined by the trust region is sampled using the $O(n^2)$ EA. A full Hessian is approximated and its eigenvector matrix kept. For the next iterations the local design space is sampled using the $O(n)$ EA. By the use of the AED technique described above, a full Hessian matrix is obtained. When the trust region test fails ($\rho^k < 0$) then the algorithm resets the transformation matrix by sampling the $O(n^2)$ EA.

The modified SAO using the AED methodology proposed is presented in Figure 3. For simplicity the algorithm mentions a single function to be approximated, however it is equally applicable to a fully constrained problem.

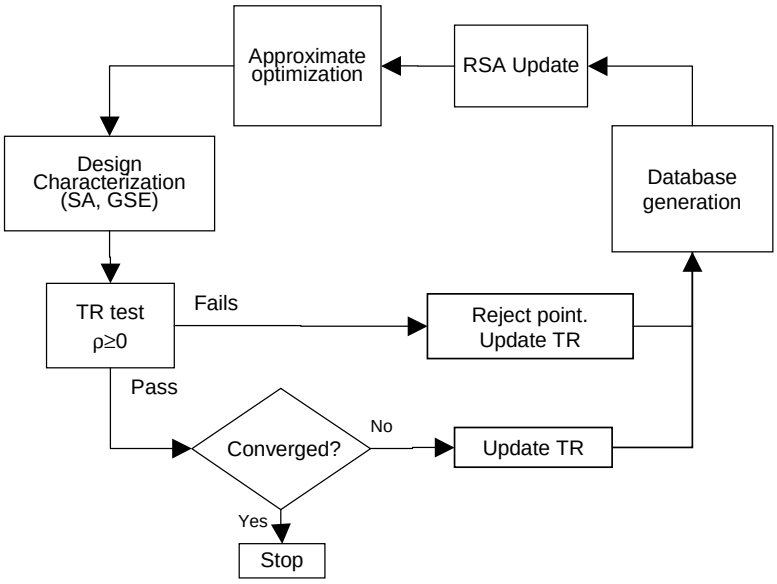

Figure 2. Sequential approximate optimization algorithm

F. Handling Infeasible Starting Points with Homotopy

Most Interior Point NLP algorithms assume a start from a feasible point, and provide the user with a different algorithm to reach feasibility in case the starting point is non-feasible. In the algorithm being developed in this research, the use of a two step approach for the optimization, will be avoided by relaxing the constraints when an infeasible starting point is encountered. Probability one homotopy methods[23,24] will be used to relax the constraints and push the design point back to feasibility. The proposed scheme will relax only the inequality constraints, however, it is possible and straightforward to extend this approach to equality constraints. The relaxed constraints take the following form:

$$\begin{aligned} &\text{Choose} \quad b_i > 0 \\ gr_i(\mathbf{x}) &= g_i(\mathbf{x}) + (1-\tau)b_i \geq 0 \end{aligned} \tag{13}$$

Where b_i is a constant and τ drives the relaxed constraint gr_i to the actual constraint by gradually adjusting $\tau = 0 \rightarrow \tau = 1$. The approximate minimization subproblem can be solved with respect to the relaxed constraints.

$$\begin{aligned} \min \quad & \tilde{\theta} \\ \text{s.t.} \quad & \tilde{\mathbf{gr}} \geq 0 \\ \text{s.t.} \quad & \tilde{\mathbf{hr}} = 0 \\ & \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \end{aligned} \tag{14}$$

Note that the resulting point will be feasible with respect to the relaxed inequality constraints. This is referred to as a *relaxed feasible point*.

When the step passes the trust region test, a new design point $(\tau + \Delta\tau, x_\tau + \Delta x_\tau)$ will be predicted by the use of homotopy curve tracking techniques. This predicted point then becomes the starting point for solving the next iteration and should be a good prediction of the next iterate.

Note that by using the constraint relaxation technique we are now able to use the *same* interior point algorithm even if the AUV starting point is infeasible.Moreover, the homotopy curve tracking technique, pushes

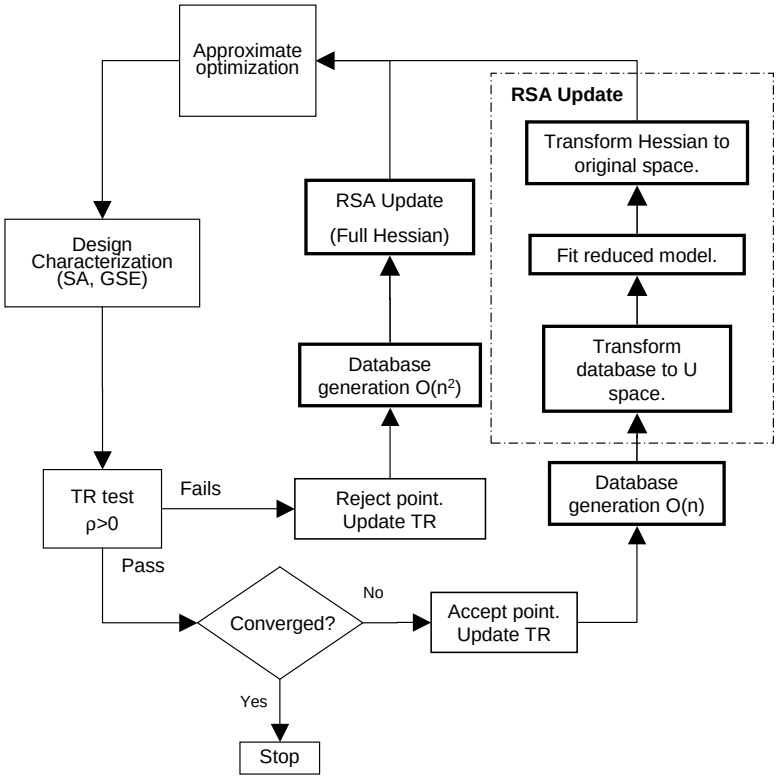

Figure 3. Sequential approximate optimization with AED.

the design to feasibility while maintaining a descent direction in θ.

III. AUV Model

The Autonomous Underwater Vehicle (AUV) simulation was created in SIMULINK and consists of an AUV model and a simplified evading torpedo that has a limited number of maneuvers that it can perform to pursue its target. The pursuing AUV attempts to neutralize the other torpedo given a design and sizings. For given performance characteristics the simulation was run for many different scenarios which resulted in a probability of success based on the number of times the AUV was successful in reaching the evading torpedo before it reached its target.

The simulation includes a full six degree of freedom dynamic model of an AUV and was developed to evaluate the effectiveness of such a craft given a set of physical attributes. These attributes include but are not limited to: speed, mass, moments of inertia, control gains in the auto-pilot, and target detection capabilities. A full description of the model is given in Patel *et al.*[9] The effectiveness of the AUV is based upon the probability that it can successfully complete a given mission and how quickly it can complete this mission.

This mission is simply to hit an oncoming torpedo before it was able to reach its target. The objective of such a mission is to maximize the probability of successfully hitting the torpedo before the torpedo reaches its own target. In order to calculate this probability, the simulation is run with many different starting configurations including: different speeds of the on coming torpedo, evasive maneuvers of the on-coming torpedo, and also various spacial orientations between the AUV and the enemy torpedo.

Computing the probability that the AUV is successful requires the model to be run many times and, therefore, is computationally prohibitive. Also, since the objective function is based at least in part on a discrete

outcome, hit or miss, this function is not as smooth as most optimizers require. To address both of these issues the IP-TR-SAO framework is utilized.

The pursuing underwater vehicle is the dynamically modeled vehicle that pursues the evading vehicle by employing guidance control and strategy. The following assumptions are used in this simulation regarding the performance of the modeled vehicle:

1. Rigid body with equal mass distribution
2. No environmental disturbances
3. Constant gravity at any depth
4. Constant density of the surrounding medium
5. Constant thrust propulsion
6. Neutral buoyancy

In each simulation sequence, the pursuing vehicle starts at a specified initial position with a given initial velocity as if it were launched from a defending vessel.

A. Mission Details

To optimize the performance of the AUV model, it is simulated using a game theory approach. In these simulations, we define the craft described in this paper as the pursuing craft. A trajectory of a second craft, defined as the evader, is also simulated using simple linear equations. This is done to reduce the computing time. Therefore, it is an unintelligent representation of a moving target that does not intelligently evade its opponent, whereas the pursing AUV intelligently pursues this target.

In each mission, the evader travels along a path that ultimately ends at its target, while the pursuing AUV starts from this target, tracks, and attempts to hit the pursuing vessel before it reaches its destination. Idealistically, this is a simulation of a submarine defending itself against an incoming torpedo by firing an 'anti-torpedo' torpedo as a countermeasure. Multiple attack scenarios are executed within each set of simulations.

More specifically, these scenarios entail various maneuvers used by the evader to unintelligently evade the pursuer at different positions of attack (angle and depth) and different speeds of the evader. These different orientations are illustrated in Figure 4. Due to symmetry, the oncoming torpedo needs only to be positioned on one side of the semicircle that is defined by the radius around the evader's target (which can also been seen as the pursuer's starting point). At the start of each simulation, the pursuer is given the initial position of the evader.

The evader model includes four path scenarios: a (1) straight line path, (2) barrel roll maneuver, (3) curved trajectory, and (4) weaving path. These paths are modeled using simple line and sinusoid equations. These paths are shown in Figure 5, where (1) and (2) are climbing towards the pursuer and defended vessel, and (3) and (4) attack from the same depth. To further simplify the dynamics of the evading vehicle paths, the evader begins at the origin, although the end result of the simulation given an initial orientation and position for the pursuer is as if the evading vehicle is starting at varied positions.

In addition to varying starting orientation, the speed, the height from which the evading vehicle attacks, and the path type make up the possible simulation scenarios can each be varied. The performance of the pursuing vehicle based on the percentage of hits and average minimum encounter distance (for misses) for the scenarios mentioned. This is what makes up the objective function to be optimized. The motivation behind these simulated mission is to test the AUV under the actual scenario that a real AUV may encounter.

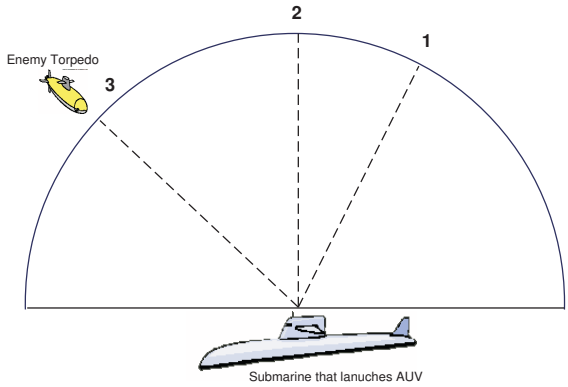

Figure 4. Mission scenarios

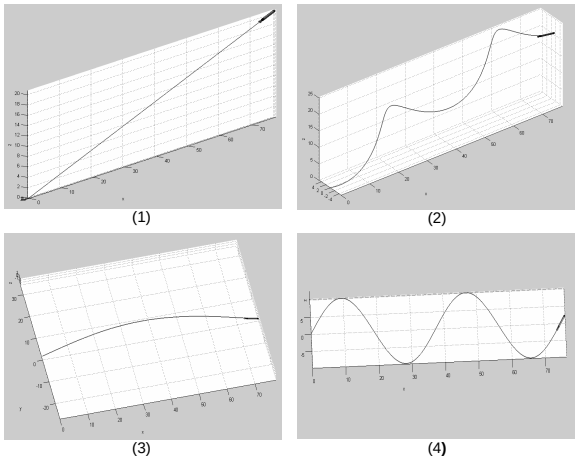

Figure 5. The evader's paths

IV. Optimization of the AUV

Using the dynamics model of the AUV presented in this paper, optimization can be performed to design a AUV based on its performance for a given mission. The details of the optimization process are explained in this section.

A. Problem Statement

The ARL at Penn State required a more complex AUV problem to be developed for unclassified AUV design. Therefore, the d-AUV model was developed to add complexity to a relatively simplistic sizing model the ARL had developed. The d-AUV model is coupled to the ARL's original AUV model as a system analysis call for componential sizings before each set of simulations are executed.

For each set of simulations, three metrics are stored: (1) the number of successful 'hits' by the pursuer of the evader or the probability of successful missions, P_{kill}, (2) the average minimum separation distance during each mission, $Dist_{kill}$, and (3) the time required for each mission, $Time_{kill}$. We wish to maximize the probability of success of the AUV while minimize the minimum distance separation and mission time.

The objective is to achieve an increase the probability of success for a given mission, while reducing the time for each mission for quicker strike. Constraints are placed on the length of the d-AUV. Furthermore, if the AUV is unsuccessful in it's attempt, it should travel as close as possible to the evading vessel. The design

problem in standard form is given as follows

$$\begin{aligned} \min_{\mathbf{x}} \quad & f = -P_{kill} + Time_{kill} + Dist_{kill} \\ \text{s.t.} \quad & 4.00 < L(\mathbf{x}) < 4.75 \\ & 0.75 < x_1 < 2 \\ & 13,000 < x_2 < 20,000 \\ & 0.7 < x_3 < 1.1 \\ & 0 < x_4 < 0.3 \\ & 0.7 < x_5 < 1.1 \\ & 0 < x_6 < 0.3 \\ & 15 < x_7 < 30 \\ & 0 < x_8 < 0.03 \\ & 0 < x_9 < 0.03 \end{aligned}$$

where L is the length of d-AUV and the design variables x_1-x_9 are defined in Table 1.

B. Optimization Methodology and Results

The initial problem, developed at the ARL for external release, was a simple linear two variable design problem. By adding dynamics and a control system, the new problem includes seven more design parameters: four control gains and three strategy parameters.

On average, using a computer with a Pentium 4 1.8 GHz processor, each d-AUV objective function call requires approximately 12-14 minutes depending upon the given design and simulation scenario.

For each simulation used for this paper, the evader starts from each of 3 different yaw positions relative to the submarine located on a 150 meter semi-circle as seen in Figure 4. The yaw angles used here are also shown in Figure 4. From each of these yaw positions three different starting height of the torpedo are used one is level with the submarine and the other two are 10 meters above and below submarine. Two more variations are used for each of these spacial starting points and those are the evading maneuvers used by the torpedo and also the speed at which the torpedo can travel. Four different evading maneuvers were used: straight path to the submarine, barrel rolling path, a half-sinusoidal path, and a two dimensional weave in the plane of the submarine. The evading torpedo was also given two speeds which were held constant for a single simulation these speeds were $10\frac{m}{s}$ and $25\frac{m}{s}$. The 3 starting yaws, 3 heights, 4 paths, and 2 different speeds result in 72 different simulation runs. For each simulation if the AUV comes within a radius of twice its length to the torpedo it is considered a successful mission. If the torpedo evades the AUV and hits the submarine the run is considered a failed mission. In either case the total simulation time to either hit or miss and also the closest encounter distance are calculated. The final results for the optimization runs using the IP-TR-SAO framework and SQP are presented in Table 1.

V. Conclusions

In this paper, the IP-TR-SAO framework is applied to a recently developed problem that is both computationally intensive and highly nonlinear. The IP-TR-SAO methodology employs an interior point approach that insures that approximate feasibility is maintained throughout the optimization process as well as a sequential approximate optimization a series of local minimizations are performed over local response surface approximations of the design space. The method also utilizes a globally convergent probability-one homotopy theory to control the relaxation allowed to each violated constraint. The algorithm has a two-fold purpose. The first is to generate approximately feasible iterates along the optimization path so the designer can stop the algorithm at any point and obtain a usable design. The second is to speed up conver-

Design variable	Variable name	Initial design	SQP solution	IP-TR-SAO solution
x_1	payload length	1.5	1.985	1.909
x_2	thrust	16,000	14,000	16,000
x_3	rudder P gain	0.9	1.049	1.044
x_4	rudder I gain	0.1	0.174	0.175
x_5	elevator P gain	0.9	0.800	0.800
x_6	elevator I gain	0.1	0.750	0.029
x_7	strategy - lead distance	20.0	20.55	20.327
x_8	strategy - C_1	0.2	0.0152	0.0120
x_9	strategy - C_2	0.2	0.0172	0.0170
f^*	-	269.2	-15.20	-23.09
Function calls	-	-	367	190

Table 1. SQP optimization of exact and approximated design for the AUV problem

gence, helping the minimization subproblems locate the true minimum by including explicitly the constraints.

The application of the framework on the AUV design problem demonstrates its capabilities and efficiency. Starting from an infeasible point, the algorithm has demonstrated that it still converges to a feasible design. Furthermore, compared to the SQP algorithm, it converges in nearly half the time.

Acknowledgments

This research effort was supported in part by the following grants and contracts: ONR Grant N00014-02-1-0786, NSF Grant DMI-0114975, and a Grant from the AFRL / DARPA through Anteon Corporation contract F33615-98-D-3210.

References

[1]Natalia M. Alexandrov. On managing the use of surrogates in general nonlinear optimization and mdo. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Multidisciplinary Analysis & Optimization Symposium*, number AIAA 98-4798, pages 720 – 729, St. Louis, MO, September 1998.

[2]Natalia M. Alexandrov, J. E. Dennis, Robert Michael Lewis, and V. Torczon. A trust region framework for managing use of approximation models in optimization. *Structural Optimization*, 15(1):16–23, February 1998.

[3]J. E. Dennis and Victoria Torczon. Managing approximation models in optimization. In N. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*, Proceedings in Applied Mathematics Series, pages 330–347. SIAM, 1997.

[4]A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays. Theory and applications.* Springer Series in Statistics. Springer-Verlag, 1999.

[5]Michael R. Lewis. A trust region framework for managing approximation models in engineering optimization. In *Proceedings of the 6th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization Symposium*, number AIAA 96-4101, pages 1053 – 1055, Bellevue, WA, September 4-6 1996.

[6]M. D. McKay, W. J. Conover, and R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

[7]R. H. Myers and D. C. Montgomery. *Response Surface Methodology. Process and Product Optimization Using Designed Experiments.* John Wiley & Sons, Inc., New York, N. Y., 1995.

[8]A. B. Owen. Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, 2(2):439–452, July 1992.

[9]N. M. Patel, S. E. Gano, J. E. Renaud, J. D. Martin, and M. Yukish. Simulation model of an autonomous underwater vehicle for design optimization. In *Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, number AIAA 2004-2034, Palm Springs, California, 19 - 22 April 2004.

[10]V. M. Pérez and J. E. Renaud. Decoupling the design sampling region from the trust region in approximate optimization.

In *Proceedings of the International Mechanical Engineering Congress & Exposition*, volume 63 of *AD*, Orlando, FL, November 5-10 2000.

[11]V. M. Pérez, J. E. Renaud, and S. E. Gano. Constructing variable fidelity response surface approximations in the usable feasible region. In *Proceedings of the 8th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization Symposium*, number AIAA 2000-4888, Long Beach, CA, September 6-8 2000.

[12]V. M. Perez, J. E. Renaud, and L. T. Watson. An interior-point sequential approximate optimization methodology. *Structural and Multidisciplinary Optimization*, 27(5):360–370, July 2004.

[13]F. Pukelsheim. *Optimum Design of Experiments*. Chapman & Hall, New York, 1995.

[14]J. E. Renaud and G. A. Gabriele. Approximation in nonhierarchic system optimization. *AIAA Journal*, 32(1):198–205, January 1994.

[15]J. F. Rodríguez, J. E. Renaud, and L. T. Watson. Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data. *Structural Optimization*, 15(3-4):141–156, June 1998.

[16]J. F. Rodríguez, J. E. Renaud, and L. T. Watson. Trust region augmented Lagrangian methods for sequential response surface approximation and optimization. *Journal of Mechanical Design*, 120(1):58–66, March 1998.

[17]José F. Rodríguez, Victor M. Pérez, Dhanesh Padmanabhan, and John E. Renaud. Sequential approximate optimization using variable fidelity response surface approximations. *Structural and Multidisciplinary Optimization*, 22:24–34, 2001.

[18]J. Sacks, W. J. Welch, T. J Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, November 1989.

[19]T. W. Simpson, J. Peplinski, P. N. Koch, and J. K. Allen. On the use of statistics in design and the implications for deterministic computer experiments. In *Proceedings of the 1997 ASME Design Engineering Technical Conference*, volume DETC97/DTM-3881, Sacramento, CA., September 14-17 1997.

[20]Jaroslaw Sobieszczanski-Sobieski. A linear decomposition method for large optimization problems- blueprint for development. Technical Report TM-83248-1982, NASA, 1982.

[21]R. C. St. John and Draper N. R. D-optimality for regression design: A review. *Technometrics*, 17:15–23, 1975.

[22]B. Tang. Orthogonal array-based latin hypercubes. *J. Amer. Statist. Assoc.*, 88:1392–1397, 1993.

[23]L. T. Watson. Theory of globally convergent probability-one homotopies for nonlinear programming. *SIAM Journal on Optimization*, 11(3):761–780, February 2000.

[24]L. T. Watson and R. T. Haftka. Modern homotopy methods in optimization. *Computer Methods in Applied. Mechanics and Engineering*, 74(3):289–305, September 1989.

[25]B. A. Wujek and J. E. Renaud. A new adaptive move-limit management strategy for approximate optimization, part i. *AIAA Journal*, 36(10):1911–1921, October 1998.

[26]B. A. Wujek and J. E. Renaud. A new adaptive move-limit management strategy for approximate optimization, part ii. *AIAA Journal*, 36(10):1922–1937, October 1998.