
Optimization Under Adaptive Error Control for a Contact Tank Reactor

Bart G. van Bloemen Waanders* · Brian R. Carnes

Received: date / Accepted: date

Abstract Optimization problems constrained by complex dynamics can lead to computationally challenging problems especially when high accuracy and efficiency are required. We present an approach to adaptively control numerical errors in optimization problems approximated using the finite element method. The discrete adjoint equation serves as a key tool to efficiently compute both parameter sensitivities and goal-oriented error estimates at the same discretized levels. By using a recovery method for the error estimators, we avoid expensive higher order adjoint calculations. We nest the adaptivity of the mesh within the optimization algorithm, which is responsible for converging both the state and optimization algorithms and thereby allowing the reuse of state, parameters, and reduced Hessian in subsequent optimization iterations. Our approach is demonstrated on a parameter estimation problem for contamination transport in a contact tank reactor. Significant efficiency and accuracy improvements are realized in comparison to uniform grid refinement strategies and black-box optimization methods. A flexible and maintainable software interface was developed to provide access between the underlying linear algebra of a production simulator and advanced numerical algorithms such as optimization and error estimation.

Keywords optimization, PDE constrained optimization, error estimation, adjoint, adaptivity, parameter estimation, contact tank reactor

1 INTRODUCTION

One of the key goals of numerical simulation is to approximate complex physics as accurately as possible while maintaining computational efficiency. This can be achieved through the advancement of numerical methods such as linear solvers, nonlinear solvers, time integrators, preconditioners, and parallelization. More fundamentally however, the accuracy of numerical simulation is strongly dependent on the appropriate use of discretization techniques and mesh refinement which almost always accomplishes higher levels of solution accuracy. But simply refining meshes becomes computationally expensive, especially if multiple forward simulations are required as part of more detailed optimization studies. Therefore to maintain computational tractability, one can only afford to sparingly refine the grid, preferably in a way that is guided by the dynamics of the problem. This can be accomplished mathematically using the adjoint formulation, which encompasses the necessary information to drive both the optimization and grid refinement problems. Despite many technical advancements, several important technical issues remain when coupling optimization and adaptivity, consisting of the computational expensive nature of calculating an additional adjoint on higher order meshes, the lack of established algorithms to calculate error estimation within an optimization context, and the challenges associated with software implementation of intrusive algorithms in production simulation codes. In this paper we address these issues by 1) demonstrating recovery methods applied to adjoint based error estimators as an inexpensive alternative to higher order adjoint solutions, 2) reusing Hessian, state and optimization variables after each adaption cycle, 3) leveraging embedded optimization methods to efficiently combine adaptivity and optimization algorithms, and 4) enabling a generalized interface to mitigate the complexities of interfacing advanced numerical algorithms into production codes.

Significant research has been conducted in the area of a posteriori error estimation for finite element discretizations, especially for engineering responses of interest, such as surface fluxes, average values on subdomains or surfaces, and point values. The underlying tool in nearly all of these approaches, beginning with the work of Becker

* Correspondence to: bartv@sandia.gov

and Rannacher [Becker and Rannacher(1996)], is to *computationally* make use of an auxiliary linear adjoint problem. Weighting the local finite element residuals with the adjoint error yields both global error estimates on the error in the responses of interest and local error indicators that can be used to drive adaptive mesh refinement. This approach was subsequently pursued for linear elliptic problems [Paraschivoiu et al(1997)Paraschivoiu, Peraire, and Patera, Prudhomme and Oden(1999)], optimization [Becker and Kapp(1998), Becker et al(2000)Becker, Kapp, and Rannacher, Bangerth(2008)], and more general nonlinear systems of PDEs [Estep et al(2000)Estep, Larson, and Williams]. Recent reviews of adjoint-based error estimation can be found in [Giles and Süli(2002), Bangerth and Rannacher(2003)]. One of the main research issues is how to reduce the high computational cost of approximating the adjoint solve with a higher order method, while still preserving sufficient accuracy in the error estimators. We present a new approximation approach for the adjoint solve using recovery methods that is very computationally efficient when compared to solving the adjoint using a higher order method.

Optimization techniques have been studied for several decades and more recently, efficient large scale algorithms have demonstrated impressive computational performance [Akçelik et al(2005)Akçelik, Biros, Drăgănescu, Ghattas, Hill, and van Bloemen Waas]. The use of these algorithms require access to the linear algebra infrastructure of the simulator which typically is not readily available, especially in production codes. Different levels of interfaces can be considered and the choice depends on a balance of implementation effort versus desired performance. For the most efficient algorithm, first and second order sensitivity information need to be provided to the optimization algorithm to realize potentially quadratic convergence rates, whereas at the opposite end of the spectrum the optimizer calculates the objective function gradient through finite difference techniques requiring very little from the simulator (merely forward simulations) but at a significant performance cost. In this paper, we show performance comparisons for different interface strategies from the “black box” to the “simultaneous analysis and design” (SAND) approach (a.k.a. all-at-once approach). It is the SAND strategy however that not only significantly improves the computational efficiency but also provides the algorithmic flexibility to accommodate adaptivity within the optimization algorithm.

Progress on algorithms for large scale optimization with adaptivity in parallel environments has been hampered by the complexity associated with the implementation process. As one of the few research efforts, Bangerth [Bangerth(2008)] demonstrates large scale optimization algorithms with h -adaptivity applied to inversion in 3D optical tomography. However, his finite element environment was appropriately designed from the outset to accommodate adaptivity and access to the linear algebra infrastructure. Such capabilities are typically not available in existing legacy production codes. Short of completely refactoring, the algorithms in Bangerth’s work cannot be conveniently encapsulated and efficiently transferred to production codes. As part of this research, one of our goals was to create a general interface so that existing optimization libraries and adaptivity capabilities can be seamlessly used by any simulation code that adopted the interface. However, just creating an interface for optimization and adaptivity is still not sufficient to ensure the longevity of such an interface. The primary code developers are typically focussed on the enhancement of the forward prediction mode and not on the maintenance of interfaces for optimization or error estimation. A general interface must therefore also appeal to other nonlinear numerical algorithms (such as time integrators and nonlinear solvers) that are in direct support of the forward prediction. Our interface is designed to accomplish this and although a detailed description of the object oriented design of our interface is beyond the scope of this paper, the importance of these implementation issues warrants the inclusion of a brief description of our design.

In the remainder of the paper we present the algorithms to perform optimization and discuss different implementation strategies. A performance comparison is presented using flow and transport datasets. The practical difficulties associated with the theoretical error for the optimality conditions are discussed and the dual-weighted residual approach is justified. Our adaptive process makes use of the adjoints in a recovery method to augment the solution field with higher order information. This approach is verified by comparing it to an analytic solution for convection-diffusion dynamics. A description of our implementation approach is included to emphasize the intrusive nature of the implementation and the added complications of attempting this in multiple codes and production systems. After the verification section, the physics of our example dataset are explained followed by numerical results for both two and three dimensions. We summarize the effectiveness of adaptivity in the optimization context in addition to showing the performance gains for our embedded optimization and adaptivity methods which are both supported by a single adjoint calculation on the same discrete space. Numerical studies were performed in serial and parallel for two and three dimensional datasets, respectively.

2 OPTIMIZATION METHODOLOGIES

We start by defining our algorithms to solve large scale optimization problems and by identifying appropriate solution strategies that are extensible to leverage adaptivity. Suppose that the forward model is described using a semilinear variational statement: given a value of the parameter $p \in \mathcal{I}$, find the solution $u = u(p) \in V$:

$$A(u, p)(v) = 0, \quad v \in V. \quad (1)$$

where the exact form of the functional spaces V and A are problem dependent. The parameter p can also belong to a function space; here, for simplicity we assume that the parameter space is finite dimensional, or $p \in \Pi \equiv \mathbb{R}^n$.

In order to define the optimization problem, we need a cost functional F that depends on the solution u and the parameters p . The goal of the optimization problem is to find (u^*, p^*) :

$$F(u^*, p^*) = \min_{u, p} F(u, p) \quad (2)$$

subject to the constraint in (1). A classical way to solve this problem is to introduce a Lagrange multiplier field, ϕ , also known as the adjoint state, and form a Lagrangian functional \mathcal{L} that combines the objective function with the state equation:

$$\mathcal{L}(u, p, \phi) \equiv F(u, p) + A(u, p)(\phi). \quad (3)$$

The stationarity of \mathcal{L} is derived by taking variations with respect to the adjoint (ϕ), state (u), and optimization parameter (p). The following system of equations represent the first-order necessary conditions for optimality (suppressing the dependence on (u, p) for clarity in our notation):

$$\begin{cases} L_\phi \\ L_u \\ L_p \end{cases} = \begin{cases} A \\ F_u + A_u^T \phi \\ F_p + A_p^T \phi \end{cases} = \mathbf{0} \quad \begin{array}{l} \text{state equation} \\ \text{adjoint equation} \\ \text{optimization equation} \end{array} \quad (4)$$

This system of equations is typically nonlinear and therefore requires a linearization step, which can be achieved through Newton's method. This system of equations for the Newton updates is called the Karush-Kuhn-Tucker (KKT) system:

$$\begin{bmatrix} L_{uu} & L_{up} & A_u^T \\ L_{pu} & L_{pp} & A_p^T \\ A_u & A_p & 0 \end{bmatrix} \begin{Bmatrix} d_u \\ d_p \\ d_\phi \end{Bmatrix} = - \begin{Bmatrix} L_u \\ L_p \\ L_\phi \end{Bmatrix} \quad (5)$$

where L_{uu} is the Hessian operator of the Lagrangian with respect to the u variable. Different algorithms can solve these optimality conditions and the right choice depends on several issues, most importantly on the size of the optimization space, complexity of the constraints, and the affordability of the implementation effort. The most difficult one to implement is a *full space* method in which (5) is solved directly. The most notable obstacles are the need for second derivatives and special preconditioning [Biros and Ghattas(2005a), Biros and Ghattas(2005b)]. Neither requirement is tractable in most production codes. An approximation to the Hessian could be considered such as BFGS or SR1 updating methods [Nocedal and Wright(2000)], which simplifies the requirements considerably. A popular alternative is to eliminate state and adjoint variables, thereby reducing the system to a manageable one in just the inversion parameters. Approaches of this type are known as *reduced space* methods.

Several important variants of reduced space methods can be considered. A nonlinear elimination variant of a reduced space method would solve the nonlinear state equation (1) for given p for the state variable u . Knowing the state then permits solution of the adjoint equation for the adjoint variable ϕ . Finally, with the state and adjoint known, the parameter p is updated via an appropriate linearization of the optimization equation. This loop is repeated until convergence. As an alternative to such nonlinear elimination, one often prefers to follow the Newton strategy of *first* linearizing the optimality system, and *then* eliminating the state and adjoint updates via block elimination on the linearized state and adjoint equations. The resulting Schur complement operator is known as the *reduced Hessian*, and the equation to which it corresponds can be solved to yield the parameter update. After applying appropriate discretizations, the above described methods require access to the linear algebra in addition to the optimization algorithm directly communicating with the simulator. In this paper, we have adopted the Newton strategy which exposes a variety of linear objects to the optimization/adaptivity algorithm. In particular, we reuse the reduced Hessian after adapting the mesh and realize significant performance improvements (see Section 6).

To accommodate optimization algorithms as part of a simulation code can be a challenging undertaking. A range of non-standard linear algebra objects are needed including objective functions, inequality constraints, sensitivity information and a mechanism for the optimization algorithm to control the iterative loop. A decoupled approach is therefore a convenient initial approach to making use of optimization. This often referred to as the *black box* interface and requires very little information from the underlying simulator. Some basic data needs to be exchanged between optimization and simulator (usually through the file system) such as the objective function value, changes to the design parameters and globalization data. The gradient of the objective function is calculated through finite differences across the entire simulator and although very expensive computationally for many design variables, the interface is trivial. The original

optimization problem (2) is reformulated by eliminating the state variable and constraints as an unconstrained optimization problem:

$$F(u(p)^*, p^*) = \min_p F(u(p), p) \quad (6)$$

A logical improvement over the *black box* approach is to substitute direct or adjoint based sensitivities for the finite difference calculations. It is different from the intrusive approach described above in that there is still no direct interface and therefore the simulator is converged at each optimization iterations. In the numerical results section, we present a performance comparison for the black box with finite difference, black box with adjoints, and a reduced space approach. Unfortunately, the decoupled algorithms do not lend themselves to efficient use of adaptivity. As the optimization algorithm steers the simulator to convergence there is no direct interface to communicate adjoints or any other objects between optimizer and forward simulator. A fully coupled approach on the other hand provides the necessary conduits between the forward simulator and optimization algorithm to exchange adjoints, Hessians, objective function, and any other pertinent information. Before outlining our algorithmic strategy, the error estimates for the KKT system (4) and the approximation approaches are explained.

3 OPTIMIZATION AND ERROR ESTIMATION

Our goal of the adaptive error control is to minimize the error in the objective function $F(u, p)$ using an adjoint equation which is identical to the second equation in (4) used in optimization. Below we present an approach for using the same discrete adjoint to drive both algorithms. However, the adjoint for optimization is solved in the same functional space as the forward problem and by finite element orthogonality, the resulting weighted residual calculation for the error estimate would be zero. This then suggests a need for duplicate adjoint calculations, each in different functional spaces, which is unfortunately computationally expensive. Our formulation proposes a recovery method whereby higher order information is extracted from an adjoint solution on the same functional space as the optimization problem. This will not result in the same levels of accuracy in comparison to an adjoint solved in a higher functional space but we show that this approximation appears sufficient to steer the mesh adaptivity. Furthermore, highly accurate adjoints in the early stages of the optimization process will likely not justify the high cost-benefit ratio.

3.1 Finite element approximation and error estimation

The continuous first-order necessary conditions for optimality in (4) must be approximated in practice. Because of our interest in error estimation and adaptive mesh refinement, we employ the adaptive finite element method [Ainsworth and Oden(2000)]. Let $V_h \subset V$ be a finite element approximation space based on conforming elements of fixed polynomial degree $r \geq 1$. The mesh is only required to be locally quasi-uniform [Ainsworth and Oden(2000)]. The finite element approximation of the optimality conditions is then: find $(U, \Phi, P) \in V_h \times V_h \times \Pi$:

$$\begin{aligned} A(U, P)(v) &= 0, & v \in V_h \\ F_u(U, P)(v) + A_u(U, P)(v, \Phi) &= 0, & v \in V_h \\ F_p(U, P) + A_p(U, P)(\Phi) &= 0 \end{aligned} \quad (7)$$

We are interested in the error of the objective function

$$\mathcal{E}(U, P) \equiv F(u, p) - F(U, P).$$

An a posteriori error estimate for this error was derived by Becker and Kapp [Becker and Kapp(1998)] which involves the exact solution. For the case of a fixed finite-dimensional parameter space, this estimate takes the form

$$\begin{aligned} \mathcal{E}(U, P) &= \frac{1}{2} \{ A(U, P)(\varepsilon) \\ &\quad + F_u(U, P)(e) + A_u(U, P)(e, \Phi) \\ &\quad + F_p(U, P)(\xi) + A_p(U, P)(\xi, \Phi) \} + R_3, \end{aligned} \quad (8)$$

where the remainder term R_3 is cubic with respect to the errors

$$e \equiv u - U, \quad \varepsilon \equiv \phi - \Phi, \quad \xi \equiv p - P. \quad (9)$$

A lower order approximation can be defined by

$$\mathcal{E}(U, P) = A(U, P)(\varepsilon) + R_2, \quad (10)$$

where the remainder R_2 is only second order [Bangerth and Rannacher(2003)]. The form in (10) avoids approximating the state and inversion operators and thereby significantly simplifies the implementation in large production finite element codes. Consequently, a loss in accuracy is realized as a result of the remainder term increasing from third to second order ($R_3 \rightarrow R_2$). The lower order error estimate requires the exact solution to the adjoint equation:

$$A_u(u, p)(v, \phi) = -F_u(u, p)(v), \quad v \in V. \quad (11)$$

Since the exact solution is unknown, this problem is further approximated by replacing the exact state u and parameter p by the approximate solution U and parameter P : find $\hat{\phi} \in V$:

$$A_u(U, P)(v, \hat{\phi}) = -F_u(U, P)(v), \quad v \in V. \quad (12)$$

3.2 Approximations to the adjoint problem

In order to derive a computable error estimate we need to approximate the continuous adjoint problem in (12). The simplest way to do this is to use the same approximation space V_h and solve for $\Phi \in V_h$

$$A_u(U, P)(v, \Phi) = -F_u(U, P)(v), \quad v \in V_h. \quad (13)$$

The solution to this problem is exactly the same as the adjoint component of the solution to the full discrete optimality problem (7), which is potentially convenient since it has already been computed. However, because of the Galerkin orthogonality, this would give a zero approximation of the error if substituted for ϕ in (10). The ideal approach to calculate the solution ϕ is to approximate (12) using a higher order spatial approximation space \tilde{V}_h . This can be done, for example, by increasing the polynomial degree of the finite element space V_h or by refining the underlying mesh. Then the adjoint weights are approximated using the higher order approximation $\tilde{\Phi} \in \tilde{V}_h$

$$\varepsilon \approx \tilde{\Phi} - \Phi$$

This approach has the advantages of typically being quite accurate, due to the use of a higher order method. It can also be expensive, due to the higher order adjoint solve, and very difficult to implement in existing production finite element codes.

Various other less expensive approaches have been proposed using postprocessing of the approximate solution (U, Φ, P) . In these cases, the error weights are approximated using some smoothing operators that only depend on the computed approximate solution and the problem data. For self-adjoint differential operators, Paraschivoiu et al. [Paraschivoiu et al(1997)Paraschivoiu et al] used local Neumann problems on refined patches of elements to generate upper and lower bounds on the error in linear functionals. This work was improved by Prudhomme and Oden [Prudhomme and Oden(1999)], who used techniques from generating upper and lower bounds on the error in the energy norm to derive sharper bounds on the error in linear functionals. For more general partial differential equations, Becker and Rannacher [Becker and Rannacher(1996)] proposed an interpolation method for estimating first and second order derivatives of the adjoint solution computed on the same finite element mesh. Several options for approximating the adjoint were explored by Larsson et al. [Larsson et al(2002)Larsson, Hansbo, and Larsson] including approximating the adjoint *error* using a hierarchical higher order approximation with the lower order basis functions removed. They also considered approximations of the adjoint error on local patches of elements, as was later done by Carnes and Carey [Carnes and Carey(2008)].

In this work, we approximate the adjoint weights using recovery procedures. Value and gradient recovery have been used in finite elements for some time, beginning with the key work of Zienkiewicz et al. [Zienkiewicz and Zhu(1987)] and more recently by Wiberg et al. [Wiberg and Li(1994)] and Ovall [Ovall(2007)]. We refer the reader to the book by Ainsworth and Oden for a more detailed review [Ainsworth and Oden(2000)]. The value recovery is based on a least squares polynomial fit of nodal values on a patch of elements around an element. For elements of degree greater than one, this method generally produces a higher order field approximation on the element. However, for linear elements, the accuracy may not be greatly improved. Thus, this approach may be sub-optimal when terms involving the value of the adjoint error ($\phi - \phi_h$) are large.

Our choice of using a recovery method was primarily motivated by computational efficiency of solving the adjoint on the same mesh plus inexpensive post-processing. In addition our approach was convenient to implement in our production finite element code. The approach uses local operators defined on V_h that can recover higher order approximations of functions in V_h . The form of the adjoint weights can be expressed as:

$$\phi - \Phi \approx r_h(\Phi) - \Phi, \quad \nabla(\phi - \Phi) \approx R_h(\nabla\Phi) - \nabla\Phi$$

We employ a standard approach based on patches of elements around a vertex node (See Figure 1). We sample the finite element gradients on the elements and fit a polynomial through the sampled values using a least squares fit. For the case of linear finite elements in 2D, the sampling points are the element midpoints and the polynomial basis is $\{1, x, y\}$. Then the nodal values are used to define a global recovered gradient in V_h^2 .

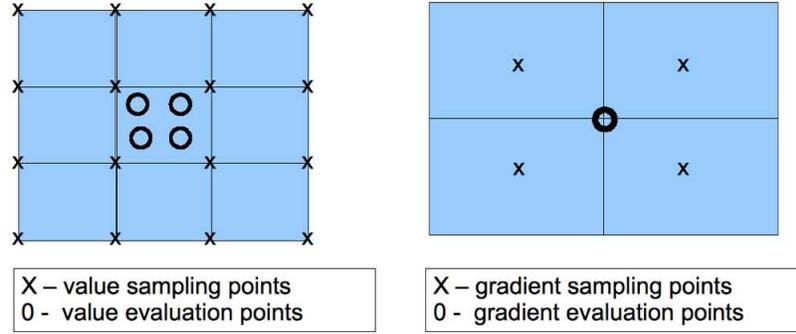


Fig. 1 Sampling points for value and gradient recovery for linear finite elements

4 ALGORITHMS FOR OPTIMIZATION UNDER ADAPTIVITY

To perform optimization under adaptive error control the choice of the algorithm is partially dictated by the form of the coupling between the application code and the optimizer. In the decoupled “black box” approach, the optimization loop launches the application code whenever the state and parameter sensitivities need to be evaluated. Adaptivity can be done by the application code, but the adapted mesh and state cannot be re-used for the next optimization step. Furthermore, our experience has shown that in this case the optimizer has difficulty reaching a stable minimum when the mesh is changing due to adaptivity.

Another possibility is to adapt the mesh at every step and thereby converging the mesh as part of the optimization loop. Bangerth [Bangerth(2008)] demonstrates this approach for problems with distributed parameters. The forward simulator, optimization algorithm and the adaptivity mechanism are closely integrated. A SAND optimization implementation solves for feasibility and optimality simultaneously, providing the necessary access to all the linear objects at any point of the forward prediction, optimization and adaptivity algorithms. Although our work also makes use of the SAND optimization approach, we do not converge the mesh as part of the optimization loop. The primary reason for not using this approach was because the goal of our implementation effort was to develop an interface sufficiently flexible to accommodate off-the-shelf optimization libraries and other advanced numerical algorithms. Adaptivity at each optimization iteration would have required optimization algorithms tailored to handle this and shifted the focus of our development efforts.

In our approach, adaptivity is performed as an outer loop around an inner optimization loop, which allows a fixed (adapted) mesh to be used for optimization. Our integrated optimization solution strategy provides, after each outer loop iteration step, the necessary access to reuse various linear objects such as the reduced Hessian, state and optimization variables. Instead of recalculating with a prescribed initial guess, the optimization now starts from a much improved starting point after each outer loop iteration. We show in section 6 the computational improvement of this reuse mechanism. The stopping criteria for the outer loop should ideally be set by comparing the error estimate for the objective function with a prescribed tolerance. Because of the uncertain quality of the error estimates that we compute, we instead use a fixed number of refinement levels. Algorithm 1 outlines our implementation strategy.

Algorithm 1 Optimization under adaptive error control

```

Given an initial parameter value  $p_0$  and state  $U_0$ 
Set initial Hessian  $H_0 = I$ 
for  $k = 1$  to number of adaption levels  $n$  do
  while optimization not converged do
    calculate adjoint sensitivities
    perform step computation
    globalize with line search
    update optimization and state variables  $(U_k, p_k)$ 
  end while
  Compute the adjoint based error estimate for  $F(U_k, p_k)$ 
  Adapt the mesh using error indicators
  Prolong state from old mesh to new  $U_{k+1} = U_k$ 
  Update Parameters  $P_{k+1} = P_k$ 
  Update Hessian  $H_{k+1} = H_k$ 
end for

```

In order to make use of the error estimator in adaptivity, a strategy is needed to decide which elements to refine/coarsen. The inputs are the element error contributions, which are the restriction of the integrals in the estimate (see equation 10) to a mesh element K . These element indicators, denoted η_K , can be either positive or negative. This confounds most adaptive strategies, which are based on refining elements with large indicators, and coarsening those with small indicators.

Our approach to adaptivity is to compute two basic statistics on the element error indicators: the mean μ and standard deviation σ . The idea behind the adaptivity is that refinement should concentrate on those η_K that are outliers. This is defined formally as: mark an element K for refinement if

$$|\eta_K - \mu| > \theta_R \sigma, \quad (14)$$

where θ_R is a free parameter. Typically we have used θ_R from 0.5 to 1.5, with decreasing θ_R yields more adaptivity. In the study we keep θ_R fixed at 0.5. We do not apply any coarsening, since we are only concerned with stationary problems. However, the above approach can be extended as: mark an element K for coarsening if

$$|\eta_K - \mu| < \theta_C \sigma, \quad (15)$$

with θ_C a free parameter.

5 IMPLEMENTATION

The implementation of analysis algorithms into production simulation codes poses numerous challenges. First, production codes historically are designed to perform only forward predictions. The linear algebra representation is typically designed with this in mind and accessing non-standard numerical objects requires extensive refactoring. For example in the case of optimization algorithms, the adjoint calculation requires a transpose of the Jacobian which is not a standard operation, especially in the parallel context (higher order adjoint solves for error estimators are even more difficult to implement). Furthermore, the terms in the general error estimate formula (8) are not included in typical production codes. Even the simpler error expression in (10) requires the integration of the finite element residual against special adjoint weight functions. This can be done using the standard element assembly process, but requires the code to support swapping the nodal test functions with a single adjoint weight function for every possible term in the residual. Second, each production code presents unique implementation styles with different concrete linear algebra infrastructures. Thus for legacy production codes, implementing analysis algorithms directly in the code would duplicate implementation efforts. This could be simplified if a general purpose, standardized interface were to be adopted by all the legacy codes. Third, such specialty interfaces are difficult to maintain because the advancement of production codes is centered around the forward prediction mode and the responsibility for maintaining these specialty interfaces may become quickly outdated as the development of the forward simulation code advances and changes. Our proposed solution for all the above mentioned issues is to design an interface sufficiently flexible and extensible to accommodate different underlying linear algebra infrastructures and to enable a range of numerical algorithms of interest to the developers.

To this end, we have developed an interface that is sufficiently general to provide a conduit from a range of advanced numerical algorithms (ANAs) to different underlying linear algebra infrastructures. Underlying the design is the premise that the interface is stateless, lightweight, and extensible. Our stateless interface does not maintain temporary copies to any vector or matrix objects but instead manipulates pointers. The interface is designed so that any input and output variables can be easily added or deleted to accommodate any algorithm. This is a critical feature because the interface ideally should not only be used in specialized ANAs (such as optimization) but also algorithms central to the forward simulator (i.e. nonlinear solver, time integration, etc).

5.1 Model Evaluator

The Thyra package in Trilinos [Heroux(2009)] contains a set of interfaces and supporting code that define basic interoperability mechanisms between different types of numerical software. The foundation of all of these interfaces are the mathematical concepts of vectors, vector spaces, and linear operators, as well as interfaces to various linear and nonlinear solvers. To address the communication from ANA to concrete application, the ModelEvaluator class is introduced (Fig 2). This design is based on the 'decorator' design pattern which makes it possible to extend (decorate) the functionality of a class at run time. This works by adding a new decorator class that wraps the original class in addition to combining component pointers as field to the decorator class, initializing these pointers in the component constructor, and redirecting component methods to the pointers. For additional details see [Gamma et al(1994)Gamma, Helm, Johnson, and Vlissides]. The essence of the ModelEvaluator class lies in the definition of input, output and evaluation methods from which a variety of input and output parameters can be defined for different algorithms.

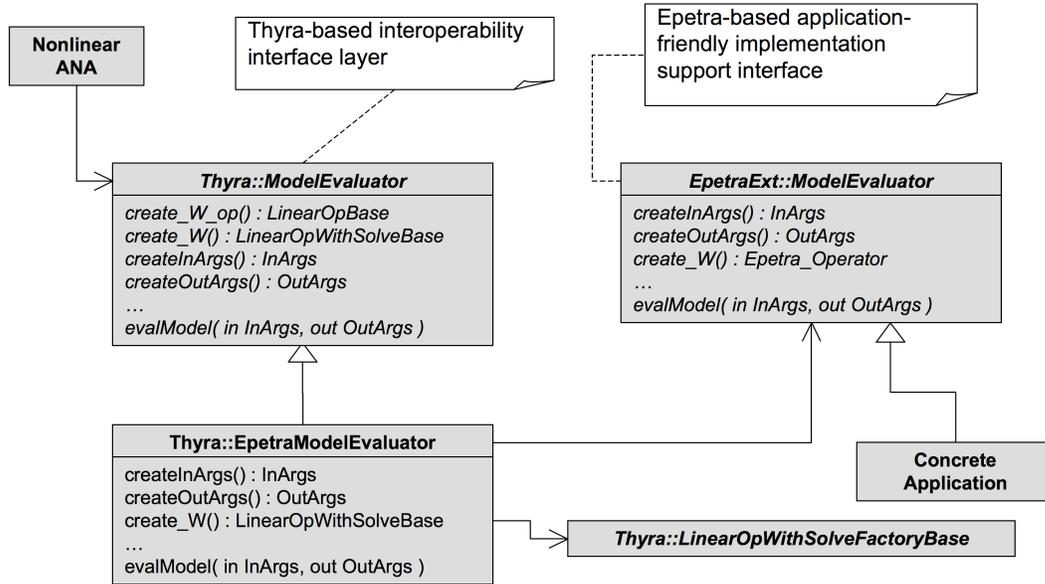


Fig. 2 The Trilinos Thyra::Model Evaluator UML class diagram. Trilinos::Epetra is the underlying vector and matrix parallel class. The ModelEvaluator class is part of the Trilinos::Thyra package which is a facility to manage and support interfaces for numerical software.

5.2 Legacy Production Simulator

The forward simulation models were implemented in a computational mechanics framework called Sierra in which the Aria package is responsible for the thermal and fluid capabilities. The Sierra framework (see Section 7 in [Biegler et al(2003)]Biegler, Ghattas, He) was designed to provide common finite element services and thereby allow for an efficient concentration on the physics development. Parallelism, mesh adaptivity, contact and multiphysics management components are among the many complex features that are available within this environment. However, the framework was designed primarily to enable the forward prediction mode which consequently creates significant implementation challenges to incorporate analysis algorithms. Aria is capable of first and second order finite elements on locally refined (*h*-adaptive) meshes. The supported physics used were the incompressible flow and transport modules. The adjoint was implemented and solved by making use of the solver capabilities from the Trilinos framework. In addition to optimization and error estimation, the Thyra::ModelEvaluator interface will also enable advanced time integration, and uncertainty quantification in the near future.

6 NUMERICAL RESULTS

Optimization and adaptivity algorithms present significant implementation challenges but, as this section will show, these disadvantages are offset by impressive accuracy and performance improvements. The embedded nature of SAND methods enable adjoint based error estimation to drive adaptivity which further improves the overall computational efficiency in addition to improving the accuracy of both the forward and the optimization problems. In this numerical results section, our goal is to demonstrate these algorithms on non-trivial examples within a production type simulation code. We target two and three dimensional datasets that describe flow and transport dynamics for a contact reactor tank used in water treatment. Navier Stokes and convection-diffusion-reaction partial differential equations are implemented in a parallel finite element framework with embedded optimization under adaptivity. First, the recovery method will be verified by comparing adjoint calculations using higher order elements to the recovery method with simple convection-diffusion dynamics. Second, the performance of SAND versus NAND optimization interfaces will be compared, followed by the performance and accuracy of these interfaces combined with uniform (combined with a NAND interface) and adaptive (combined with a SAND interface) refinement strategies. A two-dimensional flow and transport problem forms the basis for our numerical experiments. Third and finally, a three dimensional dataset for a subsection of the contact reactor tank will demonstrate our implementation in parallel, in addition to a demonstration of the reuse of certain linear algebra objects to help accelerate the convergence of the optimization problem.

6.1 Prototype Two Dimensional Problem with Solution Verification

We compare the accuracy of our recovery-based approach to a higher order adjoint solve on a simple transport problem in which the error is calculated using a known analytical solution. The model represents stationary transport of a species by convection-diffusion as follows:

$$\begin{aligned} u \cdot \nabla c - \epsilon \Delta c &= f & \text{in } \Omega &\equiv (0, 2) \times (0, 1), \\ c &= 0 & \text{on } \Gamma_{in} &\equiv \{x = 0\}, \\ c &= 1 & \text{on } \Gamma_{out} &\equiv \{x = 2\}, \\ -\epsilon \nabla c \cdot n &= 0 & \text{on } \Gamma_o &\equiv \{y = 0, 1\} \end{aligned} \quad (16)$$

The velocity field is chosen to be parabolic with $u \equiv (4y(1-y), 0)$. The dimensionless parameter ϵ is equal to the inverse of the Peclet number (Pe) and is set to 0.01 to prevent stability problems from highly convective dominated dynamics, which would require some form of stabilization. Although stabilization is available in the SIERRA framework we elected to use mild convective transport conditions instead of complicating our implementation with a stabilized formulation. The infinite dimensional problem is approximated using bilinear basis functions on quadrilateral elements.

Using methods of manufactured solution [Roache(1998), 2] the source term f is chosen so that the exact solution is given by

$$u(x, y) = \frac{1 - \exp((1 + x(x-2)y^2(1-y)^2)x/\epsilon)}{1 - \exp(2/\epsilon)}$$

The response function J is defined to be the average value of the species across the entire domain:

$$J(c) \equiv \frac{1}{|\Omega|} \int_{\Omega} c \, dx.$$

We employ two metrics to compare the performance of the error estimators. The first is the standard effectivity index, which is defined to be the ratio of the error estimator to the exact error. Ideally this number should be close to one. The second is the error reduction under adaptivity, when compared to uniform mesh refinement. In Figure 3(a) we plot the effectivity ratio for both the recovery method (denoted as Q1R) and the higher order approach with bi-quadratic elements (denoted as Q2) under uniform and adaptive refinement. For both the recovery and higher order adjoint the residuals associated with surface flux boundary conditions can be neglected although in general these weighted residual contributions should be included.

For uniform refinement, the Q2 estimator effectivity tends to about 1.02, while the Q1R estimator only tends approximately to the value of about -5.3. When adaptivity is used, the effectivity of both the Q1R and Q2 estimators becomes more volatile because the meshes are much more irregular. However, the Q2 estimator eventually stabilizes, while the Q1R estimator still appears to oscillate on the finest adaptive meshes. From this comparison, the Q1R estimator does not appear to achieve reasonable effectivity values whereas the Q2 estimator eventually settles on more stable quantities. However, in Figure 3(b) the error reduction from the Q1R is significantly better than uniform refinement and provides equal error reduction as the Q2 estimator. Moreover, the error reduction from the Q1R estimator is more monotone than that obtained from the Q2 estimator. We conclude that the Q1R estimator can drive adaptivity although additional work is required to achieve appropriate effectivity values. Accordingly in our numerical experiments, the number of refinement levels is set a priori and not dynamically determined with an effectivity tolerance.

6.2 Application to a Model for Transport in a Contact Tank Reactor

Our recovery approach efficiently calculates error estimators using the optimization adjoint. To further demonstrate this capability on a relatively complex problem, we select an appropriate optimization problem constrained by convection-diffusion-reaction transport of a species in a contact tank reactor, which is used in water treatment. In this section the details of the contact problem are described and in subsequent sections this dataset will be used to perform numerical experiments. Wang et al. [Wang and Falconer(1998)] developed a two dimensional finite difference model of the flow and transport in order to investigate transport of a tracer. They focused primarily on resolving the fluid flow with different turbulence models, concluding that solute transport predictions depends on the accuracy of the hydrodynamics. Although in Wang's study the flow is turbulent, we have reduced Reynolds number to the laminar case to allow for a more simplified investigation of adaptivity and optimization for transport without the complications of turbulence. In addition the chemical reactions which consume the reactant species were assumed to be first order and located on prescribed surfaces.

For the contact tank, the boundary $\Gamma \equiv \partial\Omega$ is divided into four parts: the inflow Γ_{in} , for which we specify a parabolic fluid velocity and constant species concentration; the outflow Γ_{out} , for which we specify an open flow boundary condition on the flow and a zero diffusive flux condition on the species concentration; the surface reaction Γ_{rxn} ,

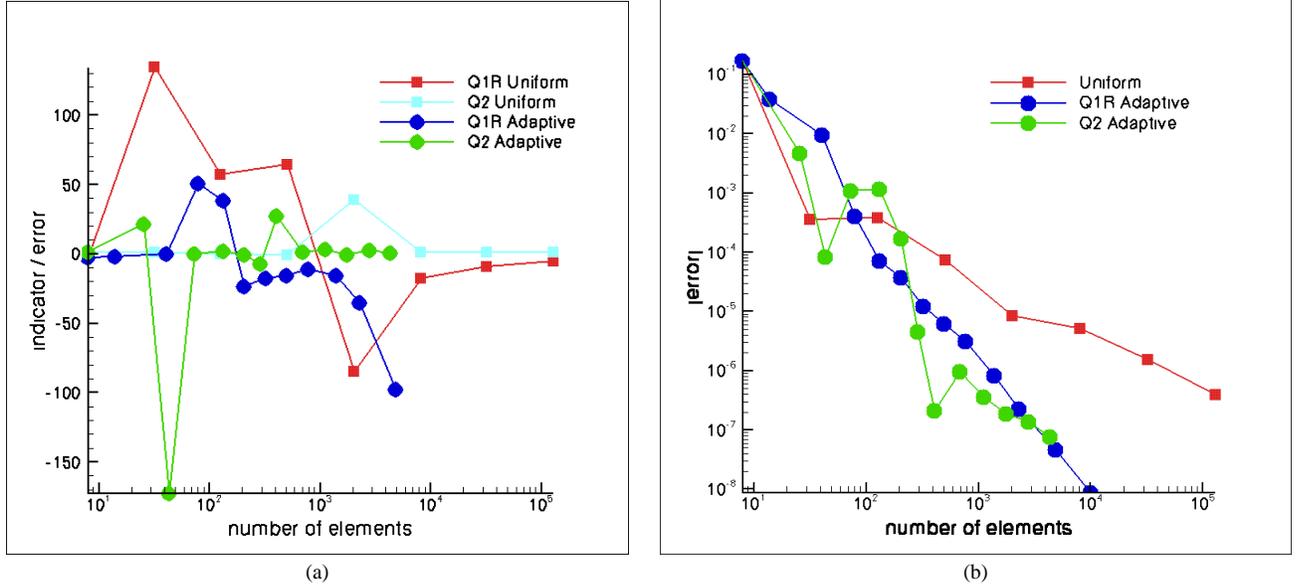


Fig. 3 Results of the two dimensional verification problem. (a) Effectivity ratio for linear (Q1R) and quadratic (Q2) meshes, under uniform and adaptive mesh refinement. (b) Exact error under uniform adaptive refinement for Q1R and Q2.

for which we specify a first order reaction for the species (the flow boundary condition is no slip); and the remaining surface Γ_o for which we also specify a zero diffusive flux condition on the species and a no slip condition on the flow.

The mathematical model for the flow is defined by the stationary incompressible Navier Stokes equations along with appropriate boundary conditions. Neglecting gravity, these can be formulated on a domain Ω as follows.

$$\begin{aligned}
 \rho u \cdot \nabla u - \mu \Delta u + \nabla p &= 0 && \text{in } \Omega, \\
 \nabla \cdot u &= 0 && \text{in } \Omega, \\
 u &= u_{in} && \text{on } \Gamma_{in}, \\
 u &= 0 && \text{on } \Gamma_{rxn} \cup \Gamma_o, \\
 \{-p I + \mu (\nabla u + \nabla u^t)\} \cdot n &= \mu n \cdot \nabla u^t && \text{on } \Gamma_{out}.
 \end{aligned} \tag{17}$$

The actual contact tank geometry consists of a flow domain with a single inlet and outlet. The domain has multiple turns at right angles to form a serpentine structure. We plot the computed flow field for $Re = 100$ in Figure 4. The channel was extended at the outlet (not shown) in order to allow the fluid to return to a near fully developed flow. The reaction zones were located where the flow would be in closest proximity to the walls, in order to increase mass transport.

Because our interest is in the species transport, we only consider the flow as an auxiliary problem that provides input to the transport through the fluid velocity. In order to avoid numerical errors from under-resolved flow, the flow equations (17) were solved on a fine grid using finite element spaces consisting of quadratic velocities and continuous linear pressures. This solution was then interpolated to the grids (both uniform and adaptive) where the following transport equation was solved:

$$\begin{aligned}
 u \cdot \nabla c - D \Delta c &= 0 && \text{in } \Omega, \\
 c &= c_{in} && \text{on } \Gamma_{in}, \\
 -D \nabla c \cdot n &= 0 && \text{on } \Gamma_o \cup \Gamma_{out}, \\
 -D \nabla c \cdot n &= k c && \text{on } \Gamma_{rxn}.
 \end{aligned} \tag{18}$$

The dominant dimensionless groups for equations (17)-(18) are the Reynolds number $Re \equiv \frac{\rho U L}{\mu}$, the Peclet number $Pe \equiv \frac{U L}{D}$, and a third dimensionless group denoted by $\Pi \equiv \frac{k L}{D}$. Here U is defined to be the maximum inlet velocity u_{in} , L is the width of the flow channel, and k is a reference surface reaction rate constant. In Table 1 we specify the baseline parameters for the contact tank model.

The solution to the steady state transport problem defined by (18) can be expressed in abstract form as in (1). To do this, we define the function spaces $V^{c_{in}} \equiv \{v \in H^1(\Omega) : v|_{\Gamma_{in}} = c_{in}\}$ and $V \equiv \{v \in H^1(\Omega) : v|_{\Gamma_{in}} = 0\}$. The parameters are the set of reaction coefficients k_j that are specified as constants on the set of surfaces that make up Γ_{rxn} . The weak solution is obtained by finding $c = c(k) \in V^{c_{in}}$:

$$A(c, k)(v) = 0, \quad v \in V. \tag{19}$$

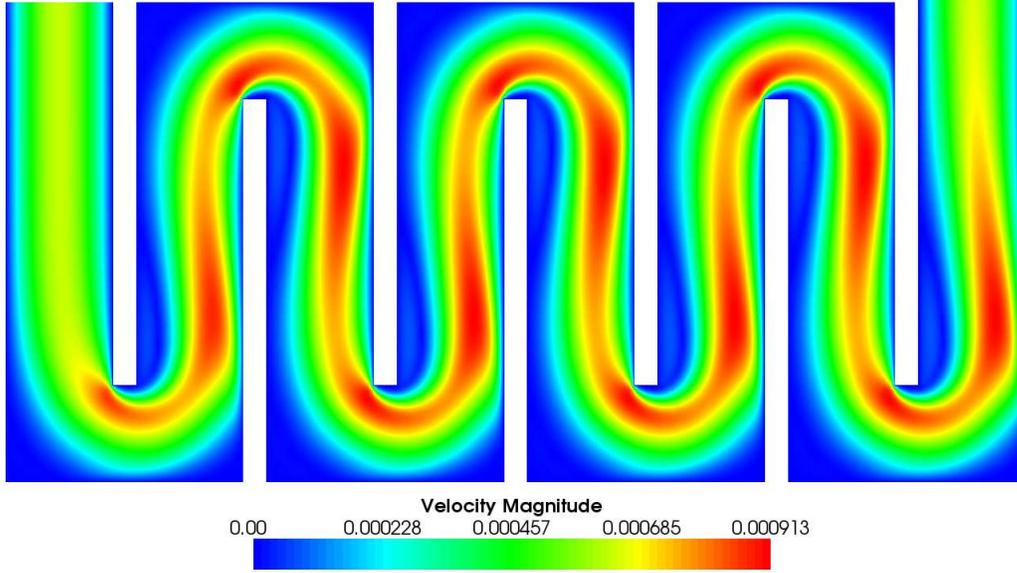


Fig. 4 Two dimensional contact reactor tank with a flow field for $Re = 100$. Resolved flow was achieved by extensions at the outlet.

Name	Value	Units	Description	Name	Value	Units	Description
Re	100	-	Reynolds number	μ	1.307e-3	[Pa-s]	Viscosity
Pe	100	-	Peclet number	u_0	6.330e-4	[m/s]	Initial velocity
Π	3.160	-	-	c_{in}	1.0	-	Inlet concen.
L	0.21	[m]	Length	D	1.329e-6	[m ² /s]	Diffusivity
ρ	9.832e+2	[kg/m ³]	Density	k	2.0e-5	[m/s]	Reaction rate

Table 1 Nominal parameters for the contact tank model. Re is the Reynolds number, the Peclet number Pe represents a ratio of convection and diffusion, and the dimensionless number Π represents the ratio of reaction and diffusion.

where the operator A is defined by

$$A(c, k)(v) \equiv (u \cdot \nabla c, v) + (D \nabla c, \nabla v) + \langle k c, v \rangle_{\Gamma_{rxn}}, \quad (20)$$

and we have used the usual notations for integral inner products $(v, w) \equiv \int_{\Omega} v w \, dx$ and $\langle v, w \rangle_{\Gamma} \equiv \int_{\Gamma} v w \, ds$.

By choosing appropriate finite dimensional spaces $V_h^{c_{in}} \subset V^{c_{in}}$ and $V_h \subset V$ for the trial and test functions, respectively, we can define the Galerkin finite element approximation: find $C = C(k) \in V_h^{c_{in}}$:

$$A(C, k)(v) = 0, \quad v \in V_h. \quad (21)$$

This abstract form provides a mapping to our algorithmic description in the preceding sections.

6.3 Optimization of Multiple Reaction Parameters to Fit a Prescribed Concentration on Reaction Surfaces

We compare the performance of the various optimization approaches described in Section 2, with the exception of the full space approach. In addition, we compare the performance and accuracy of using either uniform or adaptive grids for the reduced space SAND approach.

Our test problem contains six reaction parameters. The goal of the optimization problem is to solve an inverse problem by reconciling the differences between prescribed and numerical concentration profiles. The area where the comparison is made is along the total reaction surface, which in this example consists of six disjoint surfaces, each with its own constant reaction rate (See Figure 5). The function that we fit is a linear function of x that decreases along the overall flow direction:

$$c_{rxn}(x) \equiv 1 - x/4. \quad (22)$$

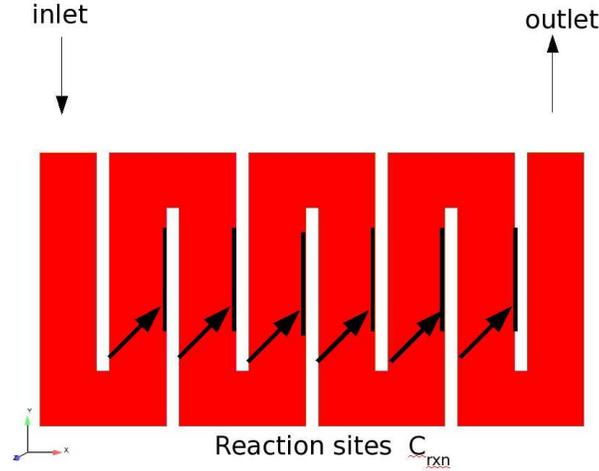


Fig. 5 Reaction sites are marked by the arrows in the two dimensional contact tank. These sites are represented as six disjoint sidesets of the mesh

Since the length of the domain in the x -direction is two, this should result in a concentration profile from approximately one to one half in the x -direction. The response function in this case is defined by

$$J(c) \equiv \frac{1}{2} \int_{\Gamma_{\text{rxn}}} |c - c_{\text{rxn}}|^2 dx. \quad (23)$$

The solutions to the forward and adjoint problem are shown in Figure 6 for $\text{Re}=\text{Pe}=100$ and at the optimal parameter values. The forward solution exhibits large gradients near the surfaces where reactions occur, as well as near the various corner singularities. The adjoint has similar dynamics, only reversed, and exhibits plumes that flow off the reaction surface in the upwind direction. These plumes mask the serpentine like features in the adjoint solution.

We compare the various optimization approaches – black box with finite difference sensitivities (BB-FD), black box with analytic sensitivities (BB), and reduced space with analytic sensitivities (RS), using both uniform and adaptive meshes. In all cases, the optimal parameters from the coarser mesh are used to initialize the optimization of the finer mesh. However, only the RS approach is able to reuse the solution state and reduced space Hessian approximation as discussed in Section 4.

In Table 2 we can see that there are significant differences in the computational cost. Most expensive is the BB-FD

DoFs	$J(c) \times 1e3$	% Error	Total Computational Time [s]		
			BB-FD	BB	RS
893	4.99553	147.6	1537	176	24
3217	2.61738	29.75	3646	461	41
12161	2.15112	6.64	9253	1283	264
47233	2.05408	1.83	–	12077	3481
186113	2.01717	–	–	–	73101

Table 2 Optimization results for the contact tank using six parameters and uniform meshes. The error is with respect to the 186113 Dof case as the truth model. BB-FD represents the black box finite difference interface, BB represents the black box with analytic sensitivity case, and RS represent the reduced space optimization approach. All our numerical results were performed on a Intel Xeon 2.66 GHz processor, running RedHat Linux Enterprise release version 4.0.

approach. Here the cost can be more than an order of magnitude slower than the BB approach. This is because of the excessive number of function evaluations needed as well as the lower accuracy of the finite difference derivatives. The BB approach was about a factor of four to six slower than the RS approach. The latter method likely was faster than the black box case with analytic sensitivities because of the improved algorithm which allows infeasible paths toward the optimal parameters in addition to the elimination of repetitive pre and post-processing of the simulator. These results are consistent with past studies that have demonstrated the computational advantages of SAND methods over black box implementations [van Bloemen Waanders et al(2002)van Bloemen Waanders, Bartlett, Long, Boggs, and Salinger, Akçelik et al(2005)Akçelik, Biros, Drăgănescu, Ghattas, Hill, and van Bloemen Waanders]. It is clear that avoiding the repetition of converging the forward simulator provides the SAND approach with significant computational advantages.

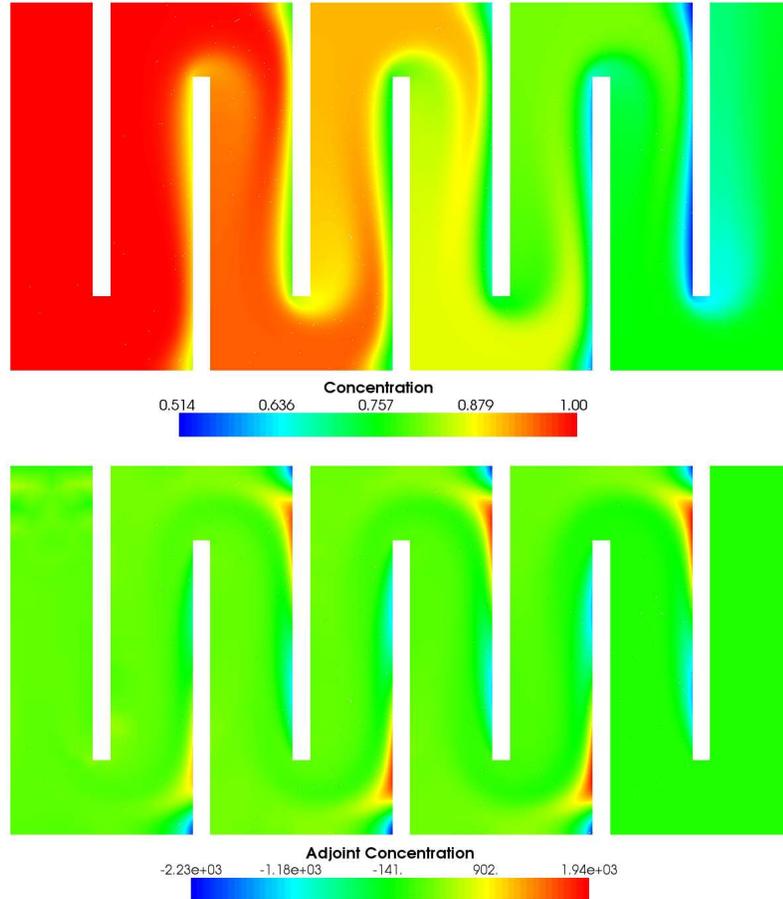


Fig. 6 Concentration solutions for the forward (top) and adjoint (bottom) for the multiple parameter case. The plumes off the reaction sites in the adjoint solution mask the serpentine feature similar to the forward solution.

However, another advantage to the SAND approaches is that adaptivity can be implemented. We compared the RS approach for both uniform and adaptive grids, with adaptivity driven by the error estimator defined in Section 3. In Table 3 we report the values and relative error of the objective function. We clearly see that with adaptivity, the accuracy in the objective function is improved by orders of magnitude over what is computable using uniform grids. Moreover, optimization under adaptivity is more efficient. To reach approximately two percent relative error using uniform grids takes about one hour (3481 s); using adaptivity, a similar accuracy can be obtained in about one to three minutes.

DoFs	RS, Uniform Refine			DoFs	RS, Adaptive Refine		
	$J(c) \times 1e3$	% Error	Time [s]		$J(c) \times 1e3$	% Error	Time [s]
893	4.99553	147.6	24	1425	3.79773	88.26	27
3217	2.61738	29.75	41	2896	2.35480	16.73	40
12161	2.15112	6.64	264	5020	2.09260	3.73	65
47233	2.05408	1.83	3481	10724	2.01748	0.012	172
186113	2.01717	-	73101	24166	2.02137	0.20	702
-	-	-	-	34796	2.01767	0.02	2763
-	-	-	-	119322	2.01726	-	14437

Table 3 Optimization results for the two dimensional contact tank using six parameters and adaptive meshes. The Reduced Space (RS) is used to compare the accuracy and performance for uniform and adaptive mesh refinement. The error is with respect to the finest grid.

To appreciate the improvements in efficiency and accuracy, the relative error versus computational cost is plotted for all the approaches – BB-FD, BB, and RS (both uniform and adaptive refinement) – on a single graph in Figure 7. Several conclusions can be drawn: first, the restriction to uniform meshes results in a limiting slope (dashed line) for the error

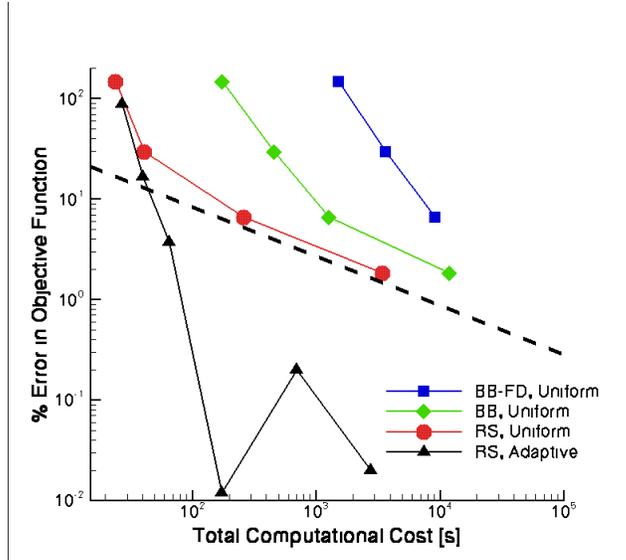


Fig. 7 Cost versus accuracy for the BB, BB-FD and RS (uniform and adaptive) strategies. The dashed line represents the limiting slope for uniform refinement.

reduction (when plotted as log-log). This slope is a function of the smoothness of the exact forward and adjoint solutions, which is reduced in this case because of the number of geometric singularities in the problem occurring at re-entrant corners. Second, the slope is much better when adaptive refine is used, actually closer to the optimal second order slope that is observed for problems with smooth solutions on uniform meshes. As a result, the adaptive approach can realize levels of accuracy not practically obtainable using uniform meshes. The RS adaptive results exhibit an anomalously low objective function error value at approximately 200 seconds. We believe this to be a result of the somewhat random nature of the adaptivity process.

Two of the adaptive grids used for refinement levels four and six are shown in Figure 8. Adaptivity is concentrated along the reaction surfaces and in regions where the adjoint solution plumes are located. No adaptivity occurs near the outlet because the adjoint is approximately zero.

6.4 Large Scale Optimization of a 3D Contact Tank Model

Although a detailed three dimensional study is beyond the scope of this paper, the implementation of any algorithms in our computational framework must be functional in multiple dimensions and operate in parallel. Therefore, in this section our algorithms are demonstrated on a subsection of a three dimensional contact tank dataset (Figure 9) with an increased number of inversion parameters (eighteen) solved in parallel using eight Intel Xeon 2.66 GHz processors. For this 3D case, the optimization is performed to match a different constant constant value on the upper (0.9) and lower (0.8) reaction surfaces. The solution concentration is plotted in Figure 9.

Our numerical experiments are limited to a comparison of uniform refinement and adjoint based adaptive refinement, all within the reduced space SAND optimization context. As shown in Table 4, the adaptive refinement algorithm achieves the same error as the uniform refinement, but with an order of magnitude less number of degrees of freedom. This translates then into more than an order of magnitude improvement in computational efficiency. It should be noted that the total cost of the adjoint based error estimator using the recovery post-processing approach was generally less than 10 percent of the total computational cost.

As a final calculation, a comparison was performed to assess the benefits of re-using the reduced Hessian matrix, which in this problem is an 18×18 dense matrix. When the reduced Hessian is not re-used, it is initialized as an identity matrix. We see in Table 5 that even for uniform problems, computational cost savings of a single optimization solve on the finest mesh can be as much as 60%. When adaptive refinement is used, the cost savings can be as high as 70%. Since most of the computation is done on the finest grid, we conclude that re-use of objects such as the reduced Hessian, in addition to the state, between levels of mesh refinement can significantly improve the efficiency of algorithms for optimization under adaptivity.

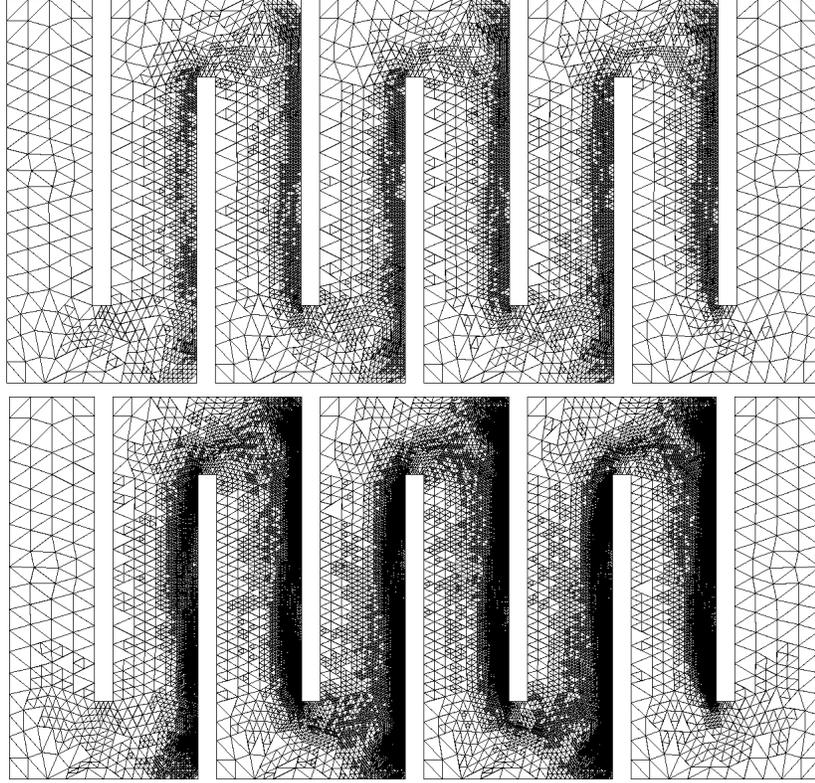


Fig. 8 Adaptive meshes used at optimal value of reaction rate parameters for 10,724 dofs (top) and 34,796 dofs (bottom)

DoFs	RS, Uniform Refine			DoFs	RS, Adaptive Refine		
	$J(c) \times 1e3$	% Error	Time [s]		$J(c) \times 1e3$	% Error	Time [s]
532	1.23231	56.43	18	532	1.23231	56.43	18
3367	3.12091	10.34	69	1260	2.74336	3.01	39
23725	3.06178	8.25	995	4473	2.97430	5.16	118
177625	2.92271	3.33	8441	15237	2.89203	2.25	364
-	-	-	-	60047	2.84875	0.72	1587
-	-	-	-	234159	2.82843	-	9812

Table 4 Optimization results for the 3D contact tank using 18 parameters using Reduced Space (RS) comparing both uniform and adaptive refinement. Error are calculated with respect to the finest grid.

RS, Uniform Refine			RS, Adaptive Refine		
DoFs	Re-use	Iden	DoFs	Re-use	Iden
550	26	26	550	26	26
3385	18	19	914	21	22
23743	26	30	2722	17	30
177643	12	31	9517	12	29
-	-	-	36098	9	29

Table 5 Comparison of iteration counts for both re-use of the reduced Hessian and initialization with an identity matrix for both uniform and adaptive refinement

7 CONCLUSIONS

We have presented an approach for implementing goal-oriented adaptivity and optimization in a production finite element code. The adjoint is central to both calculating an optimal solution and error estimation for mesh adaptivity. To avoid finite element orthogonality, ideally the adjoint for estimating errors should be determined on a higher functional space than the corresponding adjoint for sensitivity calculations. This poses more computational demands and therefore we have developed a recovery process that allows the higher functional space adjoint to be approximated by polynomial projection. This error indicator avoids additional adjoint calculations and was shown to be a viable tool for driving adap-

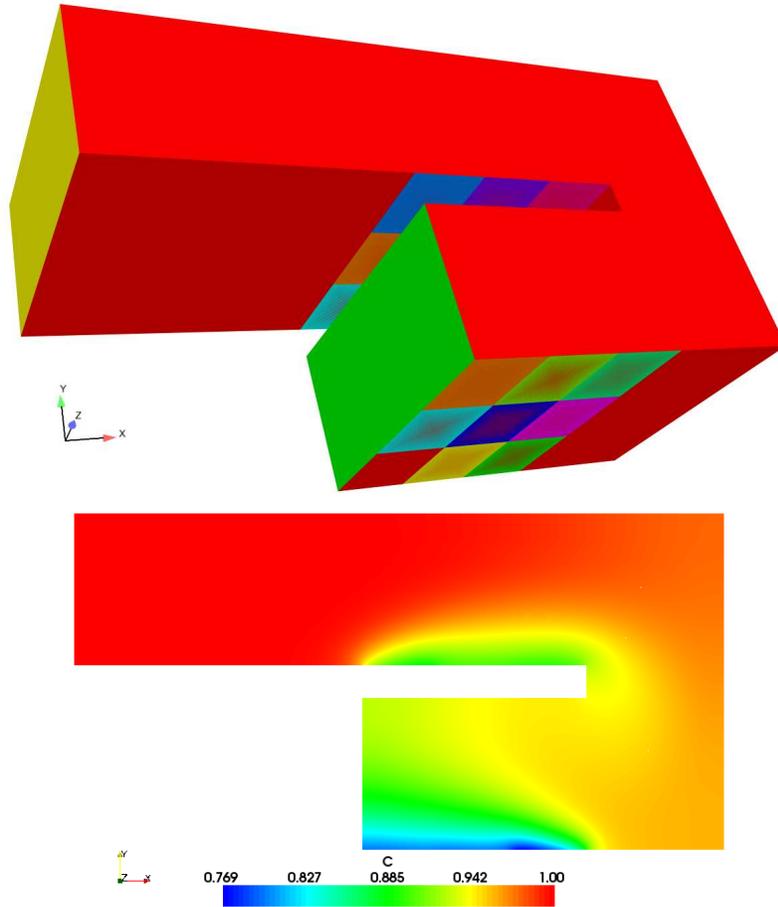


Fig. 9 Three dimensional dataset for a subsection of the contact tank; 18 reaction sites are indicated by the colored squares (top). Concentration profile is shown along the centerline (bottom).

tivity. Numerical experiments showed that effectivity could not be relied upon to terminate the adaptivity loop. However future work is planned to investigate improvements.

The SAND optimization approach provides significant computational advantages over a black-box interface, in addition to a convenient environment for adjoint based error estimator for adaptivity. The SAND optimization interface was shown to be compatible with adaptivity using a nested approach. Moreover, reduced space optimization methods can be accelerated through the re-use of the state and parameter variables, as well as the approximate reduced Hessian, when transferred from coarse to fine grids.

We outlined the implementation requirements needed to enable optimization under adaptivity in production simulation codes. This was accomplished through a ModelEvaluator abstract interface from the Trilinos library. The ModelEvaluator interface provides a conduit between advanced numerical algorithms and the underlying linear algebra native to the simulator. Besides enabling off-the-shelf optimization libraries, other numerical algorithms can be efficiently interfaced including those algorithms that are essential to the forward simulator, such as nonlinear solvers and time integration methods. Not only are duplicate implementation efforts avoided but this interface is more likely to be maintained by those responsible for the forward simulation codes.

Finally, the effectiveness of our approach was demonstrated on a 2D convection-diffusion-reaction problem from the water treatment community. It should be noted however that our methods and interfaces are generally applicable to a wide variety of physics and production simulation codes. Furthermore, our numerical tests showed improved accuracy in the optimization solution. Finally, we demonstrated our algorithms on a 3D parallel dataset with an increased number of optimization parameters.

Acknowledgements We thank Roscoe Bartlett for designing the Model Evaluator interface in addition to supporting the implementation of this interface into the Sierra::Aria production simulator.

References

- [Ainsworth and Oden(2000)] Ainsworth M, Oden JT (2000) A posteriori error estimation in finite element analysis. Wiley-Interscience, New York
- [Akçelik et al(2005)Akçelik, Biros, Drăgănescu, Ghattas, Hill, and van Bloemen Waanders] Akçelik V, Biros G, Drăgănescu A, Ghattas O, Hill JC, van Bloemen Waanders BG (2005) Dynamic data driven inversion for terascale simulations: real-time identification of airborne contaminants. In: Proceedings of SC2005, IEEE/ACM, Seattle, WA
- [Bangerth(2008)] Bangerth W (2008) A framework for the adaptive finite element solution of large-scale inverse problems. *SIAM J Sci Comput* 30(6):2665–2989
- [Bangerth and Rannacher(2003)] Bangerth W, Rannacher R (2003) Adaptive Finite Element Methods for Differential Equations. Birkhäuser Verlag
- [Becker and Kapp(1998)] Becker R, Kapp H (1998) Optimization in pde models with adaptive finite element discretization. In: et al HB (ed) Proceedings ENUMATH 97 (Heidelberg), World Scientific, pp 147–155
- [Becker and Rannacher(1996)] Becker R, Rannacher R (1996) A feed-back approach to error control in finite element methods: Basic analysis and examples. *East-west J Numer Math* 4:237–264
- [Becker et al(2000)Becker, Kapp, and Rannacher] Becker R, Kapp H, Rannacher R (2000) Adaptive finite element methods for optimal control of partial differential equations: basic concept. *SIAM J Control Optim* 39(1):113–132
- [Biegler et al(2003)Biegler, Ghattas, Heinkenschloss, and van Bloemen Waanders eds] Biegler L, Ghattas O, Heinkenschloss M, van Bloemen Waanders eds B (2003) Large-Scale PDE-Constrained Optimization. Springer Verlag Lecture Notes in Computational Science and Engineering
- [Biros and Ghattas(2005a)] Biros G, Ghattas O (2005a) Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver. *SIAM Journal on Scientific Computing*
- [Biros and Ghattas(2005b)] Biros G, Ghattas O (2005b) Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: The Lagrange Newton solver, and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*
- [van Bloemen Waanders et al(2002)van Bloemen Waanders, Bartlett, Long, Boggs, and Salinger] van Bloemen Waanders B, Bartlett R, Long K, Boggs P, Salinger A (2002) Large scale non-linear programming for pde constrained optimization. Tech. Rep. 2002-3198, Sandia National Laboratories
- [Carnes and Carey(2008)] Carnes B, Carey G (2008) Local boundary value problems for the error in fe approximation of non-linear diffusion systems. *Int J Numer Methods Engrg* 73(5):665–684
- [Estep et al(2000)Estep, Larson, and Williams] Estep D, Larson M, Williams RD (2000) Estimating the error of numerical systems of reaction-diffusion equations. *Mem Amer Math Soc* 146(696):viii+109
- [Gamma et al(1994)Gamma, Helm, Johnson, and Vlissides] Gamma E, Helm R, Johnson R, Vlissides J (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series
- [Giles and Süli(2002)] Giles M, Süli E (2002) Adjoint methods for pdes: A posteriori error analysis and postprocessing by duality. In Iserles, A (ed) *Acta Numerica* Cambridge: Cambridge University
- [Heroux(2009)] Heroux M (2009) Trilinos. Tech. rep., Sandia National Laboratories, <http://trilinos.sandia.gov/>
- [Larsson et al(2002)Larsson, Hansbo, and Runesson] Larsson F, Hansbo P, Runesson K (2002) Strategies for computing goal-oriented *a posteriori* error measures in non-linear elasticity. *Int J Numer Meth Engrg* 55:879–894
- [Nocedal and Wright(2000)] Nocedal J, Wright SJ (2000) Numerical Optimization. Springer
- [Ovall(2007)] Ovall JS (2007) Function, gradient, and hessian recovery using quadratic edge-bump functions. *SIAM J Numer Analysis* 45(3):1064–1080
- [Paraschivoiu et al(1997)Paraschivoiu, Peraire, and Patera] Paraschivoiu M, Peraire J, Patera A (1997) A posteriori finite element bounds for linear-functional outputs of elliptic partial differential equations. *Comput Methods Appl Mechanics Eng* 150(1-4):289–312
- [Prudhomme and Oden(1999)] Prudhomme S, Oden JT (1999) On goal-oriented error estimation for elliptic problems: application to the control of pointwise errors. *Comput Methods Appl Mech Engrg* 176:313–331
- [Roache(1998)] Roache PJ (1998) Verification and Validation in Computational Science and Engineering. Hermosa, Albuquerque, NM
- [Wang and Falconer(1998)] Wang H, Falconer RA (1998) Simulating disinfection processes in chlorine contact tanks using various turbulence models and high-order accurate difference schemes. *Water Research* 32(5):1529–1543, URL <http://www.sciencedirect.com/science/article/B6V73-3T8GSW3-T/2/1192912eb811346d28c0050958f21cfc>
- [Wiberg and Li(1994)] Wiberg NE, Li XD (1994) Superconvergent patch recovery of finite-element solution and a posteriori l_2 error estimate. *Commun Numer Methods Engrg* 10:313–320
- [Zienkiewicz and Zhu(1987)] Zienkiewicz O, Zhu J (1987) A simple error estimator and adaptive procedure for practical engineering analysis. *Int J Numer Methods Engrg*