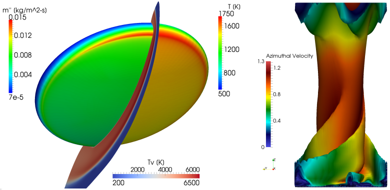
Implementing Generalized Adjoint Capabilities in libMesh

Roy H Stogner[1], Vikram V Garg[1]

[1] The University of Texas at Austin

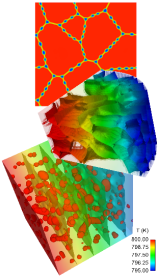
July, 2017

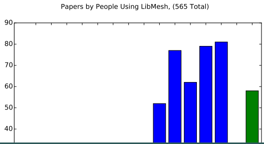
libMesh Community



Scope

- Free, Open source
  - LGPL2 for core
- 45 Ph.D. theses, 507 papers (81 in 2016)
- ~ 10 current developers
- $110 - 240$ current users?





Adjoint-enabled codes

- libMesh library: http://libmesh.github.io/

Adjoint Equation

- Variational problem (semilinear in v; typically nonlinear in u)

$$\text{Given } \mathcal{R} : U \times V \to \mathbb{R},\ \text{find } u \in U \text{ s.t.}$$
$$\mathcal{R}(u, v) = 0 \quad \forall\, v \in V$$

Adjoint Equation

- Variational problem (semilinear in v; typically nonlinear in u)

$$\text{Given } \mathcal{R} : U \times V \to \mathbb{R}, \text{ find } u \in U \text{ s.t.}$$
$$\mathcal{R}(u, v) = 0 \quad \forall\, v \in V$$

- Given Quantity of Interest (QoI) $Q : U \to \mathbb{R}$, the adjoint problem is defined as:

$$\text{Find } z \in V \text{ s.t. } \mathcal{R}_u(u_h, z)(v) = Q_u(u_h)(v) \quad \forall\, v \in U$$

Adjoints for Sensitivity Analysis

Forward Sensitivity

$$Q' = \left.\frac{dQ}{dp}\right|_{p=p_0} = Q_p + Q_u(u_p)$$

Adjoint Sensitivity

$$Q' = Q_p + \mathcal{R}_u(u, z(u,p_0); p_0)u_p = Q_p - \mathcal{R}_p(u, z(u,p_0); p_0)$$

- For forward sensitivity, we need N_p solves for $\frac{\partial u}{\partial p}$
- Adjoints only require per-parameter residual sensitivity
- `ParameterVector` enables FDM sensitivities for codes where analytic are unimplemented:

$$\frac{\partial \mathcal{R}}{\partial p} \approx \frac{\mathcal{R}(u, z(u,p_0); p_0 + \Delta p) - \mathcal{R}(u, z(u, p_0 - \Delta p); p_0)}{2\Delta p}$$

- `ParameterAccessor` subclasses can shim into third party libraries' setter/getter methods

Hessian Calculation: Forward+Adjoint

Sensitivity Evaluation

$$
\begin{aligned}
q''_{kl} = & q_{p_k p_l} + Q_{\boldsymbol{u} p_k}(\boldsymbol{u};\boldsymbol{p})\boldsymbol{u}_{p_l} + Q_{\boldsymbol{u} p_l}(\boldsymbol{u};\boldsymbol{p})\boldsymbol{u}_{p_k} + Q_{\boldsymbol{uu}}(\boldsymbol{u};\boldsymbol{p})\boldsymbol{u}_{p_k}\boldsymbol{u}_{p_l} - \\
& \mathcal{R}_{p_k p_l}(\boldsymbol{u},\boldsymbol{z};\boldsymbol{p}) - \mathcal{R}_{\boldsymbol{u} p_k}(\boldsymbol{u},\boldsymbol{z};\boldsymbol{p})\boldsymbol{u}_{p_l} - \mathcal{R}_{\boldsymbol{u} p_l}(\boldsymbol{u},\boldsymbol{z};\boldsymbol{p})\boldsymbol{u}_{p_k} - \\
& \mathcal{R}_{\boldsymbol{uu}}(\boldsymbol{u},\boldsymbol{z};\boldsymbol{p})\boldsymbol{u}_{p_k}\boldsymbol{u}_{p_l}
\end{aligned}
$$

- N_p linear solves for $\boldsymbol{u_p}$ and N_q linear solves for $\boldsymbol{z}$
- $\mathcal{O}\left(N_q N_p^2\right)$ dot products

Hessian-Vector Product Calculation

Sensitivity Evaluation

$$\sum_l q''_{kl} c_l = \sum_l c_l Q_{p_k p_l}(\boldsymbol{u};\boldsymbol{p}) + Q_{\boldsymbol{u}p_k}(\boldsymbol{u};\boldsymbol{p}) \left(\sum_l c_l \boldsymbol{u}_{p_l} \right) - \mathcal{R}_{p_k}\left(\boldsymbol{u}, \sum_l c_l \boldsymbol{z}^l; \boldsymbol{p} \right) - \mathcal{R}_{\boldsymbol{u}p_k}(\boldsymbol{u},\boldsymbol{z};\boldsymbol{p}) \left(\sum_l c_l \boldsymbol{u}_{p_l} \right) - \sum_l c_l \mathcal{R}_{p_k p_l}(\boldsymbol{u},\boldsymbol{z};\boldsymbol{p})$$

- No direct dependence on $\boldsymbol{u}_{pl}$, just its weighted sum
  - 1 linear solve using weighted sums of $\mathcal{R}_{pl}$
- No direct dependence on $\boldsymbol{z}^l$, just its weighted sum
  - N_q linear solves using weighted sums of $Q_{\boldsymbol{u}p_l}(\boldsymbol{u}) - \mathcal{R}_{\boldsymbol{u}p_l}(\boldsymbol{u},\boldsymbol{z})$
  - Much cheaper than full Hessian for large N_p

Adjoint Error Estimation

`AdjointRefinementEstimator`

Error estimate for target QoI:

$$Q(u) - Q(u_h) = -\mathcal{R}(u_h, z) + \text{H.O.T.}$$

z must be approximated: h/p refinement, discrete adjoint

`AdjointResidualErrorEstimator`

Using Galerkin orthogonality, ,

$$Q(u) - Q(u_h) = \mathcal{R}(u, z) - \mathcal{R}(u_h, z) + \text{H.O.T.}$$

Adjoint Error Estimation

AdjointRefinementEstimator

Error estimate for target QoI:

$$Q(u) - Q(u_h) = -\mathcal{R}(u_h, z) + \text{H.O.T.}$$

z must be approximated: h/p refinement, discrete adjoint

AdjointResidualErrorEstimator

Using Galerkin orthogonality, and Taylor expansion for $R(u, v)$,

$$\begin{aligned} Q(u) - Q(u_h) &= \mathcal{R}(u, z) - \mathcal{R}(u_h, z) + \text{H.O.T.} \\ &= \mathcal{R}_u(u_h, z - z_h)(u - u_h) + \text{H.O.T.} \end{aligned} \tag{1}$$

Adjoint Error Estimation

AdjointRefinementEstimator

Error estimate for target QoI:

$$Q(u) - Q(u_h) = -\mathcal{R}(u_h, z) + \text{H.O.T.}$$

z must be approximated: h/p refinement, discrete adjoint

AdjointResidualErrorEstimator

Using Galerkin orthogonality, and Taylor expansion for $R(u, v)$,

$$\begin{aligned} Q(u) - Q(u_h) &= \mathcal{R}(u, z) - \mathcal{R}(u_h, z) + \text{H.O.T.} \\ &= \mathcal{R}_u(u_h, z - z_h)(u - u_h) + \text{H.O.T.} \end{aligned} \tag{1}$$

Inexpensive goal-oriented error indicator:

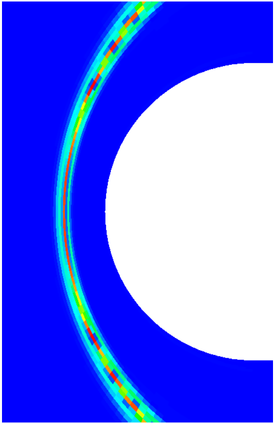
$$|Q(u) - Q(u_h)| \leq ||\mathcal{R}_u||_{ij} ||z - z_h||_j ||u - u_h||_i + \text{H.O.T.} \tag{2}$$

Adjoint-weighted Residual, Hypersonic Case

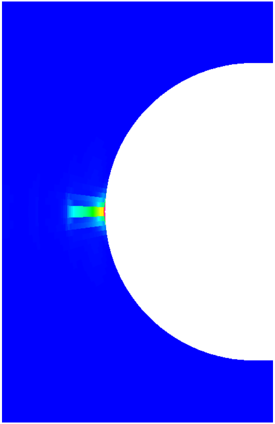
Flow: 5-species reacting viscous air, 3.7km/s inflow, adiabatic cylinder

QoIs: Peak Surface Temperature/Pressure

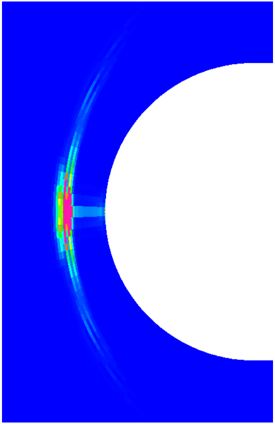
Plots: Cell-by-cell error on initial graded structured mesh



Primal (Forward) Error



Dual (Adjoint) Error



Combined QoI Error

Multiphysics Weighting

WeightedAdjointResidualErrorEstimator

$$
\begin{aligned}
|Q(u) - Q(u_h)| &\leq |\mathcal{R}_u(u_h, z - z_h)(u - u_h)| \\
&= \left| \sum_j \mathcal{R}_u^j(u_h, z - z_h)(u - u_h) \right| \\
&\leq \left|\left| z^j - z_h^j \right|\right| M_{ij} \left|\left| u^i - u_h^i \right|\right|
\end{aligned}
$$

Example

Stokes Flow

$$Q(\boldsymbol{u}) - Q(\boldsymbol{u}_h) \approx \int_\Omega \nabla(\boldsymbol{u} - \boldsymbol{u}_h) \cdot \nabla(\boldsymbol{u}^* - \boldsymbol{u}_h^*) \, dx$$
$$- \int_\Omega \nabla \cdot (\boldsymbol{u} - \boldsymbol{u}_h) \, (p^* - p_h^*) \, dx - \int_\Omega \nabla \cdot (\boldsymbol{u}^* - \boldsymbol{u}_h^*) \, (p - p_h) \, dx$$

Matrix Form

$$|Q(\boldsymbol{u}) - Q(\boldsymbol{u}_h)| \leq \begin{bmatrix} e(u_{1,1}) \\ e(u_{1,2}) \\ e(u_{2,1}) \\ e(u_{2,2}) \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e(u^*_{1,1}) \\ e(u^*_{1,2}) \\ e(u^*_{2,1}) \\ e(u^*_{2,2}) \end{bmatrix}$$
$$+ \begin{bmatrix} e(u_{1,1}) \\ e(u_{2,2}) \\ e(p) \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} e(u^*_{1,1}) \\ e(u^*_{2,2}) \\ e(p^*) \end{bmatrix}$$

Example

Navier Stokes Flow

$$Q(\boldsymbol{u}) - Q(\boldsymbol{u}_h) \approx Re \int_\Omega \left(\boldsymbol{u} : \nabla \left(\boldsymbol{u} - \boldsymbol{u}_h\right)\right) \cdot \left(\boldsymbol{u}^* - \boldsymbol{u}_h^*\right) \, dx$$

$$+ \int_\Omega \nabla(\boldsymbol{u} - \boldsymbol{u}_h) \cdot \nabla(\boldsymbol{u}^* - \boldsymbol{u}_h^*) \, dx$$

$$- \int_\Omega \nabla \cdot (\boldsymbol{u} - \boldsymbol{u}_h) \, (p^* - p_h^*) \, dx - \int_\Omega \nabla \cdot (\boldsymbol{u}^* - \boldsymbol{u}_h^*) \, (p - p_h) \, dx$$

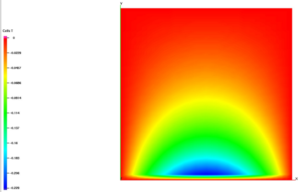
Matrix Form

$$|Q(\boldsymbol{u}) - Q(\boldsymbol{u}_h)| \leq \begin{bmatrix} e(u_{1,1}) \\ e(u_{1,2}) \\ e(u_{2,1}) \\ e(u_{2,2}) \end{bmatrix}^T \begin{bmatrix} u_1 & 0 & 0 & 0 \\ 0 & u_2 & 0 & 0 \\ 0 & 0 & u_1 & 0 \\ 0 & 0 & 0 & u_2 \end{bmatrix} \begin{bmatrix} e(u_1^*) \\ e(u_2^*) \\ e(u_1^*) \\ e(u_2^*) \end{bmatrix} + \ldots$$

Flux Quantities of Interest

Simple weighted flux integration example

$$
\begin{aligned}
R(u) &= \nabla \cdot \vec{\sigma} + f \\
R(u,v) &= -\left(\vec{\sigma}, \nabla v\right)_{\Omega} + \\
&\qquad \left(\vec{\sigma} \cdot \vec{n}, w\right)_{\Gamma_N} + \left(f, v\right)_{\Omega} \\
Q(u^h) &= \left(\vec{\sigma}(u^h) \cdot \vec{n}, w\right)_{\Gamma_D}
\end{aligned}
$$



- Multiphysics QoIs (e.g. lift, drag, heat flux, etc.) can each be expressed via choice of w non-zero variable(s)
- Suboptimal convergence rates
- Adjoint solutions *diverge* in H^1

Superconvergent Flux Quantities of Interest

Residual-based flux integration example

$$
\begin{aligned}
R(u) &= \nabla \cdot \vec{\sigma} + f \\
R(u,v) &= -(\vec{\sigma}, \nabla v)_\Omega + (\vec{\sigma} \cdot \vec{n}, w)_{\Gamma_N} + (f, v)_\Omega \\
Q(u) &= (\vec{\sigma} \cdot \vec{n}, w)_{\Gamma_D} = R(u, L) \quad \forall \;\; L|_{\Gamma_D} = w
\end{aligned}
$$

- `FunctionBase` subclasses provide w
- Multiphysics QoIs (e.g. lift, drag, heat flux, etc.) can each be expressed via choice of w non-zero variable(s)
- `DofMap` extends/projects user weight w into U_h for each QoI
- `FEMSystem` evaluates weighted flux QoI terms automatically
- `FEMSystem` subclasses can add non-weighted-flux terms to the same QoI

Weighted Flux Adjoint Problems

Adjoint Dirichlet conditions

$$\text{Find } z \in V \text{ s.t. } \mathcal{R}_u(u_h, z)(v) = 0 \quad \forall v \in U$$
$$z|_{\Gamma_D} = w$$

- `DofMap` extends/projects w into U_h for each QoI
- `FEMSystem` evaluates Q, Q_u based on supplied w
- `FEMSystem::assembly()` heterogeneous (or homogeneous, or no) constraints: $Ku = f \implies C^T K C y = C^T (f - Kh)$

Flux-Adjoint-based results

$$|Q(u) - Q(u_h)| = -\mathcal{R}(u_h, z - L) + \text{H.O.T.}$$
$$Q' = Q_p(u) - \mathcal{R}_p(u, z - L))$$

Stabilization

- Stabilization: Essential for injecting information from unresolved scales into numerical solution
- Indispensible in CFD, wide variety of stabilization schemes used
- Would like to combine stabilization with adjoint based AMR

The problem

- Stabilization schemes transform weak residual structure, essential to defining the adjoint problem
- Weak residual now changes with mesh resolution, makes analysis tricky
- Adjoint-of-discretization may no longer commute with discretization-of-adjoint

Stabilized Formulation

Consider an SUPG stabilized residual formulation,

$$\mathcal{R}^{\tau(h,p)}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{v}^{\boldsymbol{h}}) = 0 \quad \forall \boldsymbol{v}^{\boldsymbol{h}} \in \boldsymbol{V}^{\boldsymbol{h}} \tag{3}$$

where,

$$\begin{aligned}\mathcal{R}^{\tau(h,p)}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{v}^{\boldsymbol{h}}) &= \mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{v}^{\boldsymbol{h}}) \\ &\quad + < \tau(h,p)\mathcal{S}(\boldsymbol{u}^{h,\tau(h,p)}), L^*(\boldsymbol{v}^{\boldsymbol{h}}) >\end{aligned} \tag{4}$$

Stabilization & Model Refinement

- Stabilization often seen as a numerical scheme, which adds ‘artificial diffusion’
- From an operator perspective, stabilization can be seen as a modified (upscaled) model of underlying physics

Error Estimate

$$Q(\boldsymbol{u}) - Q(\boldsymbol{u}^{h,\tau(h,p)}) = -\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{z}) + H.O.T. \tag{5}$$

Adjoint z satisfies,

$$\mathcal{R}_{\boldsymbol{u}}(\boldsymbol{w}, \boldsymbol{z}; \boldsymbol{u}^{h,\tau(h,p)}) = Q_{\boldsymbol{u}}(\boldsymbol{w}; \boldsymbol{u}^{h,\tau(h,p)}) \quad \forall \boldsymbol{w} \in \boldsymbol{U} \tag{6}$$

Approximate adjoint satisfies,

$$\mathcal{R}_{\boldsymbol{u}}^{\tau(h,p)}(\boldsymbol{w^h}, \boldsymbol{z}_{\tau(h)}^{h,p}; \boldsymbol{u}^{h,\tau(h,p)}) = Q_{\boldsymbol{u}}(\boldsymbol{w^h}; \boldsymbol{u}^{h,\tau(h,p)}) \quad \forall \boldsymbol{w^h} \in \boldsymbol{U^h} \tag{7}$$

Differences with Galerkin case

- If $\boldsymbol{z}_{\tau(h)}^{h,p}$ an equal order $\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{z}_{\tau(h)}^{h,p}) \neq 0$
- Reason: $\mathcal{R} \neq \mathcal{R}^{\tau}$, stabilized solutions are not Galerkin orthogonal
- $\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{z}_{\tau(h)}^{h,p}) = \frac{dQ}{d\tau}(\boldsymbol{u}^{h,\tau(h,p)})[\tau(h,p)]$

- $\mathcal{R}_p^{\tau}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{z}_{\tau(h)}^{h,p})$ is still used for sensitivities, because it matches the forward problem.
- $\mathcal{R}^{\tau}(\boldsymbol{u}^{h,\tau(h,p)}, \boldsymbol{z}_{\tau(h)}^{h,p})$ is awful for error estimates, because it matches the forward problem!
- `AdjointRefinementErrorEstimator` may be provided access to both $\mathcal{R}^{\tau}$ and $\mathcal{R}$

1d convection diffusion

$$-u'' + \alpha u' = 0$$
$$u(0) = 1$$
$$u(1) = 0$$

α = 1000, inflow from left, SUPG stabilization

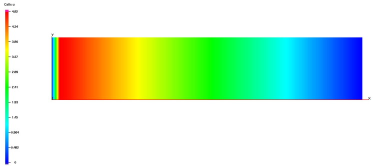
Table: Effectivities for error estimator $\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, z)$, with different approximations of z

Q	$\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, z)$	$\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, z^{h+,p}_{\tau(h+)})$	$\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, z^{h+,p}_{\tau(h)})$	$\mathcal{R}(\boldsymbol{u}^{h,\tau(h,p)}, z^{h,p+}_{\tau(h)})$
$\int_0^1 u(x)\, dx$	1.0	1.0	1.0	1.03
$\int_0^1 u(x)\, w(x)\, dx$[1]	1.0	0.99	0.98	1.01
$\alpha u_x(1)$	1.0	1.0	1.0	1.0
$u(0.996)$	0.98	0.05	0.03	0.05
$u(0.996)$ 10 h/p refinements	1.0	0.99	0.12	0.06

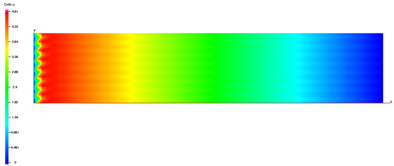
[1] $w(x) = \pi^2 sin(\pi x) - \alpha\pi cos(\pi)$

2d convection diffusion

- 2d Advection Diffusion, with SUPG []



(a) Adjoint consistent (1st order Lagrange)

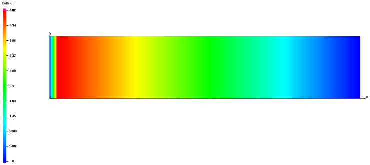


(b) Adjoint inconsistent (2nd order Lagrange)

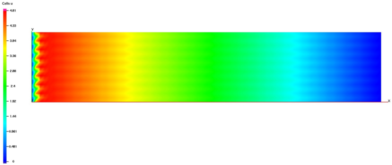
Figure: Adjoint solves obtained for the convection diffusion problem for QoI: $\int_\Omega u(\boldsymbol{x})\, d\boldsymbol{x}$

2d convection diffusion

- 2d Advection Diffusion, with SUPG []
- Oscillations if 2nd order elements used, go away if 1st order used



(a) Adjoint consistent (1st order Lagrange)



(b) Adjoint inconsistent (2nd order Lagrange)

Figure: Adjoint solves obtained for the convection diffusion problem for QoI: $\int_{\Omega} u(\boldsymbol{x})\, d\boldsymbol{x}$

2d convection diffusion

- Precise error estimates if 1st order elements used
- Oscillations seen with 2nd order elements impact error estimate accuracy, despite more dofs and actual error being lower

Table: Impact of adjoint consistency on error estimate accuracy

Consistency	*DoFs*	True Error	Effectivity
Adjoint Consistent	5127	0.00952744	1.00
Adjoint Inconsistent	20375	0.00464844	0.788

2d advection diffusion + Flux QoI

Boundary flux QoI,

$$Q(u) = \mathcal{R}(u, L) \tag{8}$$

where the lift L is given as,

$$L = y\,(1 - y)\,x \tag{9}$$

Table: Mesh dependence of Error Estimates and Effectivities for flux QoI $\mathcal{R}(u, L)$

h	$\mathcal{R}(\cdot, z_{\tau(h+)}^{h+,p} - L)$	Effectivity	$\mathcal{R}^{\tau(h+,p+)}(\cdot, z - L)$
0.5	0.08287	1.23	0.04672
0.25	0.07767	1.07	0.04242
0.125	0.07126	1.02	0.04015
0.0625	0.06083	1.01	0.03639
0.03125	0.04519	1.01	0.02894