Unification of reduced-space and full-space methods for large-scale design optimization

Anugrah Jo Joshy

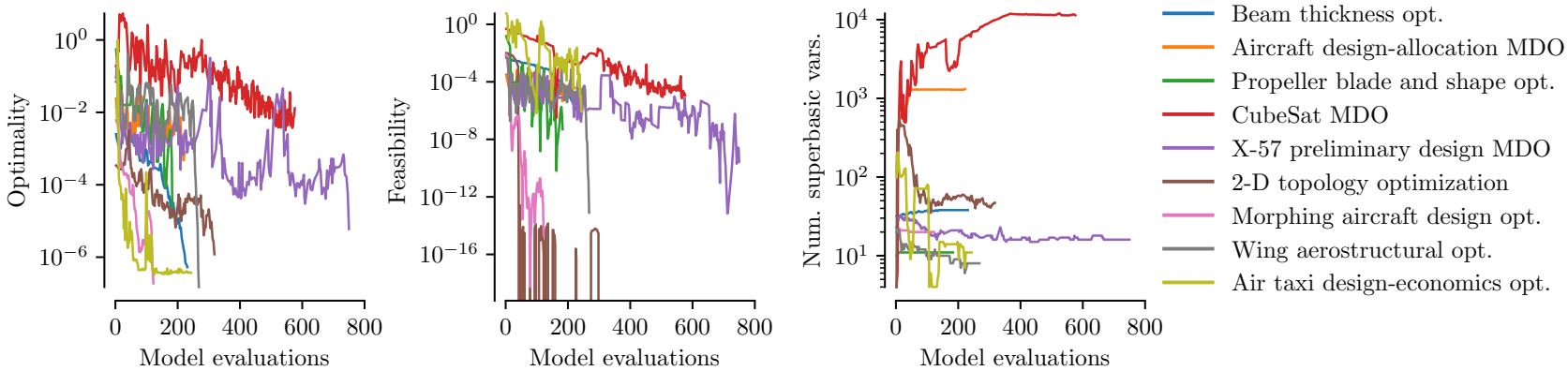
University of California San Diego

Outline

Motivation

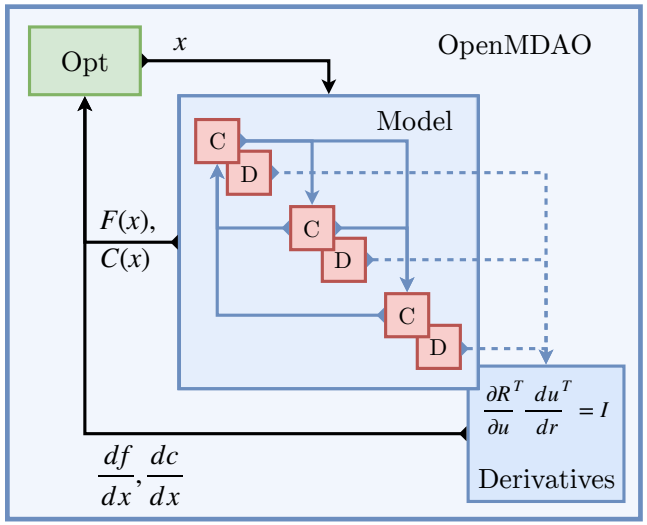
State-of-the-art in LSDO

- State-of-the-art gradient based optimizers
- MAUD(Modular Analysis and Unified Derivatives) for multi-disciplinary derivative computation
- OpenMDAO integrates gradient-based optimizers and MAUD
- Enables solution in only hundreds of model evaluations



State-of-the-art in LSDO: Challenges

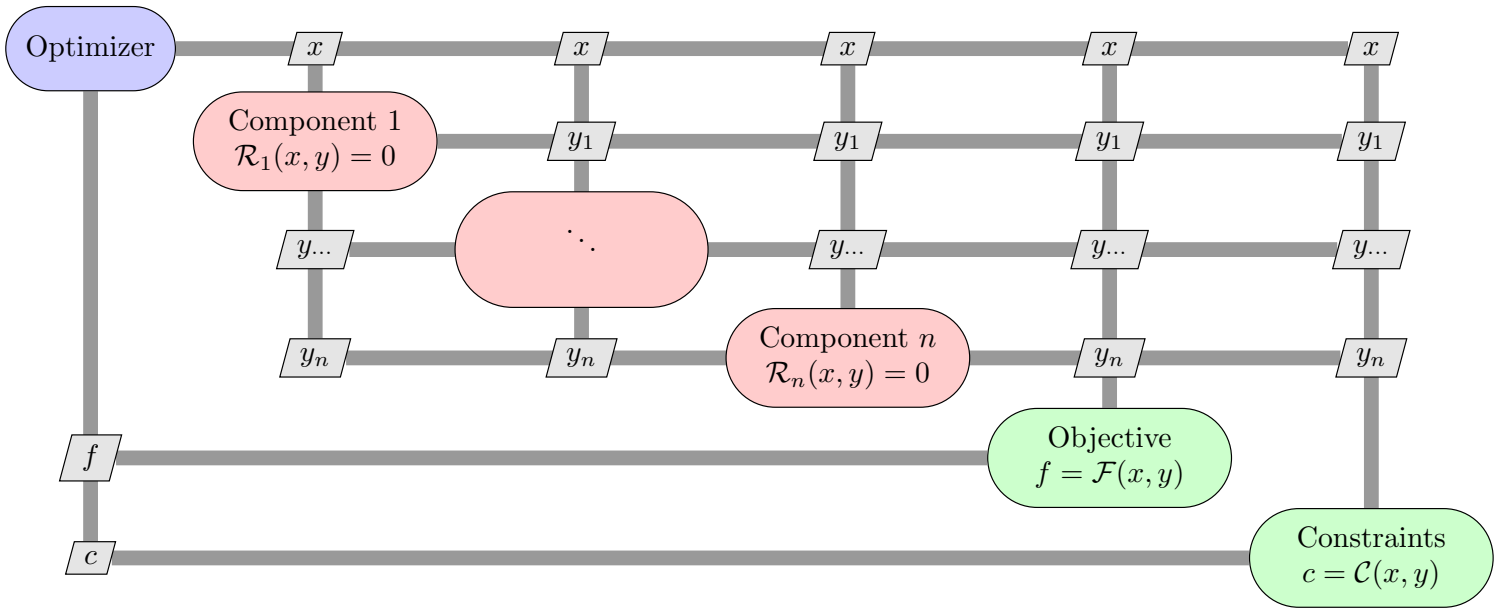
- Computational cost $\propto$ number of model evaluations
- Not fast enough to be adopted into an industrial setting
- OpenMDAO couples optimizers and models but both are still independent: Model acts as a black-box that outputs objective, constraints and their gradients
- A reduction in the number of model evaluations requires an intrusive approach where the internal components of the model are exposed to the optimizer



Optimization Formulations

Reduced-space method

Also known as multidisciplinary feasible(MDF) or nested analysis and design(NAND)



$$
\begin{aligned}
\min_{x} \quad & \mathcal{F}(x, \mathcal{Y}(x)) \\
\text{s.t.} \quad & \mathcal{C}(x, \mathcal{Y}(x)) = 0 \\
& x \geq 0, \\
\text{with} \quad & \mathcal{R}(x, \mathcal{Y}(x)) = 0,
\end{aligned}
$$

Reduced-space method

- State variables corresponding to implicit equations are solved within the model.
- Non-linear solvers within a discipline will solve the residuals $R(x, y) = 0$ to compute y.
- Only x constitutes the design variables and if $x \in \mathbb{R}^n$, the design space is n-dimensional and hence the name reduced-space.

Define the Lagrangian as:

$l = \mathcal{L}(x, \lambda)$ where $\mathcal{L}(x, \lambda) = \mathcal{F}(x, \mathcal{Y}(x)) + \lambda^T \mathcal{C}(x, \mathcal{Y}(x))$.

We obtain the following optimality conditions:

$$\frac{\mathrm{d}l}{\mathrm{d}x} = \left(\frac{\partial \mathcal{F}}{\partial x} - \frac{\partial \mathcal{F}}{\partial y}\ \frac{\partial R}{\partial y}^{-1}\ \frac{\partial \mathcal{R}}{\partial x}\right) + \lambda^T \left(\frac{\partial \mathcal{C}}{\partial x} - \frac{\partial \mathcal{C}}{\partial y}\ \frac{\partial R}{\partial y}^{-1}\ \frac{\partial \mathcal{R}}{\partial x}\right)$$
$$\frac{\mathrm{d}l}{\mathrm{d}\lambda} = \mathcal{C}(x, \mathcal{Y}(x))$$

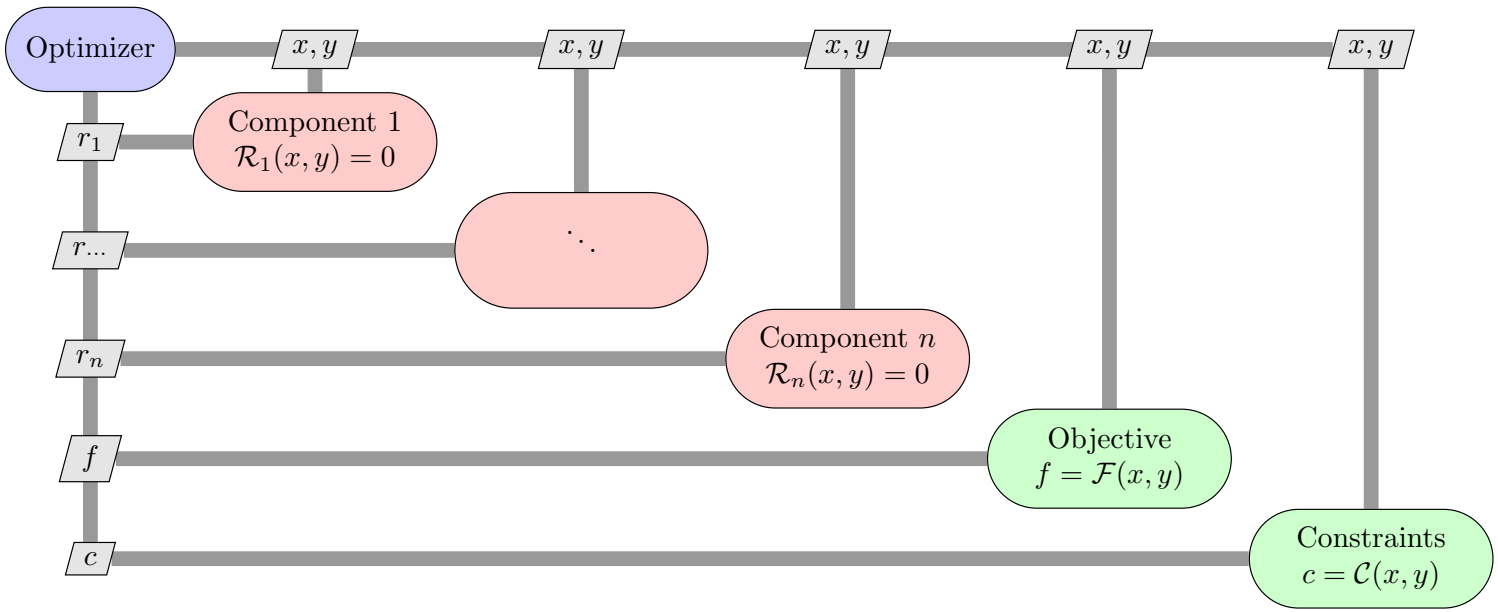
The KKT system is given by

$$\begin{bmatrix} l_{xx} & c_x{}^T \\ c_x & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -l_x \\ -\mathcal{C}(x, \mathcal{Y}(x)) \end{bmatrix},$$

where $l_{xx} = d^2 l/dx^2$, $c_x = dc/dx$, and $l_x = dl/dx$.

Full-space method

Also known as simultaneous analysis and design(SAND)



$$
\begin{aligned}
\min_{x,y} \quad & \mathcal{F}(x, y) \\
\text{s.t.} \quad & \mathcal{C}(x, y) = 0 \\
& \mathcal{R}(x, y) = 0 \\
& x \geq 0,
\end{aligned}
$$

Full-space method

- State variables corresponding to implicit equations are treated as design variables.
- Residuals $R(x, y) = 0$ are treated as additional constraints and the responsibilty of computing y is on the optimizer.
- Both x and y constitutes the design variables and if $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^k$, the design space is $(n + k)$-dimensional and hence the name full-space.

FS: Optimality and KKT conditions

Define the Lagrangian as:

$$m = \mathcal{M}(x, y, \psi, \lambda) = \mathcal{F}(x, y) + \psi^T \mathcal{R}(x, y) + \lambda^T \mathcal{C}(x, y).$$

We obtain the following optimality conditions:

$$\begin{aligned} \frac{\mathrm{d}m}{\mathrm{d}x} &= \frac{\partial \mathcal{F}}{\partial x} + \psi^T \frac{\partial \mathcal{R}}{\partial x} + \lambda^T \frac{\partial \mathcal{C}}{\partial x} & \frac{\mathrm{d}m}{\mathrm{d}\psi} &= \mathcal{R}(x, y) \\ \frac{\mathrm{d}m}{\mathrm{d}y} &= \frac{\partial \mathcal{F}}{\partial y} + \psi^T \frac{\partial \mathcal{R}}{\partial y} + \lambda^T \frac{\partial \mathcal{C}}{\partial y} & \frac{\mathrm{d}m}{\mathrm{d}\lambda} &= \mathcal{C}(x, y) \end{aligned}$$

The KKT system is given by

$$\begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x{}^T & \mathcal{C}_x{}^T \\ m_{yx} & m_{yy} & \mathcal{R}_y{}^T & \mathcal{C}_y{}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ \mathcal{C}_x & \mathcal{C}_y & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_\psi \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x, y) \\ -\mathcal{C}(x, y) \end{bmatrix}$$

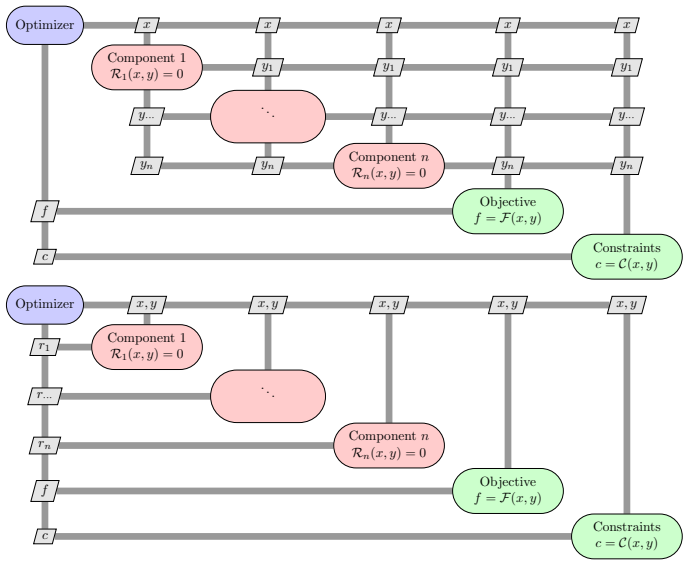
Reduced-space vs full-space

Reduced-space

$$\begin{aligned} \min_{x} \quad & \mathcal{F}(x, \mathcal{Y}(x)) \\ \text{s.t.} \quad & \mathcal{C}(x, \mathcal{Y}(x)) = 0 \\ & x \geq 0, \\ \text{with} \quad & \mathcal{R}(x, \mathcal{Y}(x)) = 0, \end{aligned} \tag{3}$$

Full-space

$$\begin{aligned} \min_{x,y} \quad & \mathcal{F}(x, y) \\ \text{s.t.} \quad & \mathcal{C}(x, y) = 0 \\ & \mathcal{R}(x, y) = 0 \\ & x \geq 0, \end{aligned} \tag{4}$$



	Reduced-space	Full-space
Design space	x	x and y
Robustness	Robust	Convergence issues
Model evaluation time	Higher	Lower
Optimization problem	Easier	Harder
Efficiency	Not efficient	Not always efficient

SURF - Strong Unification of Reduced-space and Full-space methods

SURF: Basic Idea

- State variables corresponding to implicit equations are solved to specified tolerances within the model.
- Solvers within a discipline will solve the residuals $R(x, y) \approx 0$ to compute an approximate y.
- x constitutes the design variables and y is a hybrid variable.
- A hybrid variable lies somewhere in the spectrum between a design variable and a state variable.
- Its location in the spectrum can be changed by adjustimg the solver tolerances.
  A zero tolerance on the non-linear solver is same as the reduce-space method while solving with infinite tolerance (equivalent to not solving $R(x, y) \approx 0$) in the same as a full-space method.

SURF: Algorithm

Algorithm 1 SURF (strong unification of reduced-space and full-space)

SURF unifies reduced and full space for a simplified, equality-constrained optimization setting.

1: **loop**
2: Assemble A, b, $\tilde{M}^{-1}$
3: Solve $\tilde{M}^{-1}Ap = \tilde{M}^{-1}b$
4: Compute α via a line search
5: Update $\begin{bmatrix} x_+, y_+, \psi_+, \lambda_+ \end{bmatrix}^T = \begin{bmatrix} x, y, \psi, \lambda \end{bmatrix}^T + \alpha p$
6: Update y and ψ by inexactly solving $\mathcal{R}(x, y) = 0$ and $\mathcal{R}_y^T \psi = -\mathcal{F}_y^T - C_y^T \lambda$
7: Update the approximation to A
8: **end loop**

Note: p is the search direction and α is the step size, $Ap = b$ is the FS KKT system, and M^{-1} and $\tilde{M}^{-1}$ are exact and approximate preconditioners for A such that $M = M_1 M_2 M_3 M_4$ and

$$A = \begin{bmatrix} m_{xx} & m_{xy} & \mathcal{R}_x{}^T & C_x{}^T \\ m_{yx} & m_{yy} & \mathcal{R}_y{}^T & C_y{}^T \\ \mathcal{R}_x & \mathcal{R}_y & 0 & 0 \\ C_x & C_y & 0 & 0 \end{bmatrix}, b = \begin{bmatrix} -m_x \\ -m_y \\ -\mathcal{R}(x,y) \\ -C(x,y) \end{bmatrix}, M_1 = \begin{bmatrix} 0 & 0 & \mathcal{I} & 0 \\ 0 & \mathcal{I} & 0 & 0 \\ \mathcal{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{I} \end{bmatrix}, M_2 = \begin{bmatrix} \mathcal{R}_y & 0 & 0 & 0 \\ m_{yy} & \mathcal{R}_y^T & 0 & 0 \\ m_{xy} & \mathcal{R}_x^T & \mathcal{I} & 0 \\ C_y & 0 & 0 & \mathcal{I} \end{bmatrix}$$

$$M_3 = \begin{bmatrix} \mathcal{I} & 0 & \mathcal{R}_y^{-1}\mathcal{R}_x & 0 \\ 0 & \mathcal{I} & \mathcal{R}_y^{-T} m_{yx} - \mathcal{R}_y^{-T} m_{yy} \mathcal{R}_y^{-1} \mathcal{R}_x & \mathcal{R}_y^{-T} C_y^T \\ 0 & 0 & m_{xx} - m_{xy}\mathcal{R}_y^{-1}\mathcal{R}_x - \mathcal{R}_x^T \mathcal{R}_y^{-T} m_{yx} + \mathcal{R}_x^T \mathcal{R}_y^{-T} m_{yy} \mathcal{R}_y^{-1} \mathcal{R}_x & C_x^T - \mathcal{R}_x^T \mathcal{R}_y^{-T} C_y^T \\ 0 & 0 & C_x - C_y \mathcal{R}_y^{-1} \mathcal{R}_x & 0 \end{bmatrix}, M_4 = \begin{bmatrix} 0 & \mathcal{I} & 0 & 0 \\ 0 & 0 & \mathcal{I} & 0 \\ \mathcal{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathcal{I} \end{bmatrix}$$

SURF: Algorithm

- Lines 2 and 3 represent the assembly and solution of the full-space KKT system with a preconditioner.
- Lines 4 and 5 computes the step size and applies the step.
- At this updated point, the non-linear system is solved inexactly and the value of y is updated again. Subsequently, we compute ψ using an equation that comes from setting dm/dy to zero.
- Line 6 unifies the reduced space and full space formulations.
  - If line 6 is skipped, SURF becomes a full space algorithm.
  - If the two systems in line 6 are exactly solved SURF becomes a reduced space algorithm.
  - If line 6 uses inexact solvers, SURF becomes a hybrid.
- The equivalence of the full-space and reduced-space formulations are proven when the systems in line 6 are solved exactly.

SURF: Features

- Switching between full-space and reduced-space is now simple.
- SURF allows easy access to a continuous spectrum of hybrid methods by just changing the inexact solver tolerances.
- The choice on this spectrum can be made on a variable-by-variable basis.

	Reduced-space	Full-space	SURF
Design space	x	x and y	x and hybrid y
Robustness	Robust	Convergence issues	Robust
Model evaluation time	Higher	Lower	Close to FS
Optimization problem	Easier	Harder	In between
Efficiency	Not efficient	Not always efficient	Efficient

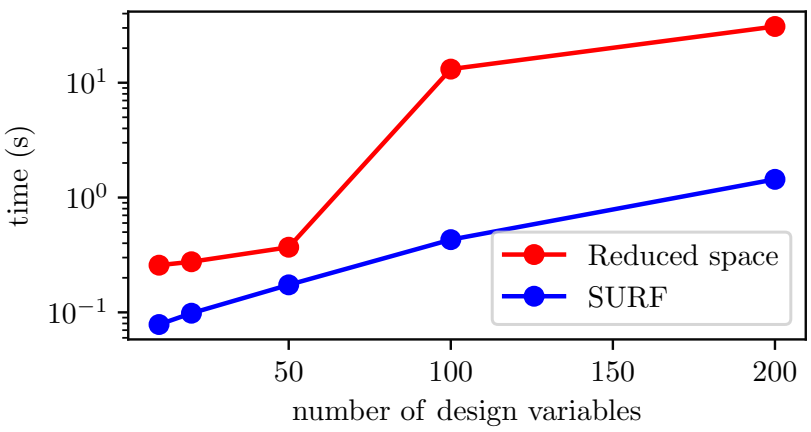
SURF: FEA problem

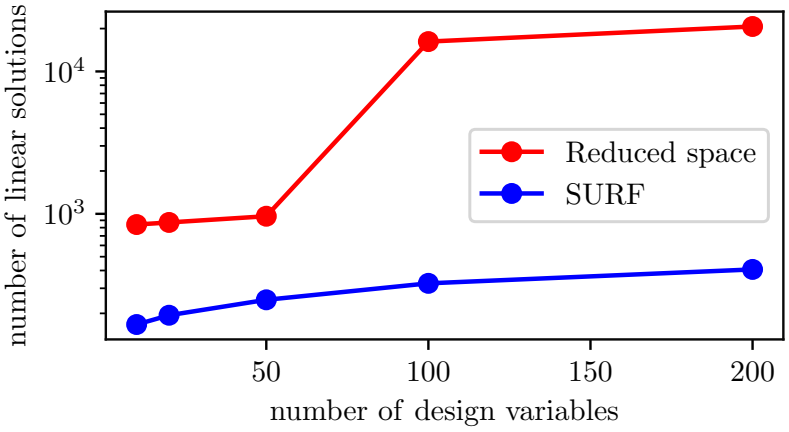
- SURF is applied to a beam thickness optimization problem
- The nonlinear, equality-constrained optimization problem is

$$\begin{aligned} \min_{x} \quad & F^T d \\ \text{s.t.} \quad & V(x) = V_0 \end{aligned} \qquad \text{with} \quad K(x,d)d - F = 0,$$

  where d is the displacement vector, F is the force vector, $V(\cdot)$ is the volume function, V_0 is the allowable volume, and K is the function that computes the stiffness matrix.
- The problem is solved using reduced-space and SURF optimizers. For SURF, we use pre-selected inexact solver tolerances.

SURF: FEA problem





- SURF with a hybrid formulation is, on average, an order of magnitude more efficient than the RS formulation in time and number of model evaluations across various numbers of beam elements.

Future work

Future work

- Extend the unification to inequality constrained optimization.
  - Unification with active-set method
- Incorporating quasi-Newton updates into the SURF algorithm
  - Unification with BFGS
- Adaptive hybrid selection - how to adaptively select solver tolerances during optimization.

Questions