Parallelization Strategies for Efficiently Computing CAD-based Sensitivities for Design Optimization

John F. Dannenhoffer, III *
Aerospace Computational Methods Laboratory, Syracuse University, Syracuse, New York, 13244

Multi-disciplinary analysis and optimization (MDAO) has been a long-standing goal in the aerospace community. In order to employ MDAO effectively, one needs to be able to compute the sensitivity of the objective function with respect to the driving parameters in a robust and efficient manner. As models get very large, there is a need to compute these sensitivities in parallel, especially since most optimization methods already employ parallel solvers.

Contained herein is a study of two different parallelization strategies for efficiently computing sensitivities. They are compared on a model problem, where the objective is to find the CAD-like design parameters that most-closely match a set of given mass properties. Although quite simple compared with CFD-based optimizations, this model problem allows one to really examine the efficiency and robustness of the parallelization strategies.

The results are that for small design changes, a linearized approach to the geometry can be very effective. But for large changes, a non-linear approach, involving rebuilds and computing the sensitivities in parallel with the flow solver, has been found to be the best approach.

I. Model Problem

A model problem was developed to be able to easily examine the effect of various parallelization strategies for design optimization. The overall process was patterned after a `FUN3D`-like design process[1]. But instead of using a full CFD solver, a surrogate was used.

- Starting with a parametrically-defined configuration with initial design parameters ($\vec{D}$) as defined in a `.csm` file for the Engineering Sketch Pad (`ESP`)[2]
- find the design parameters ($\vec{D}$) that minimizes the sum-squared difference between the prescribed target mass properties ($\vec{M}_{\text{tgt}}$) and the computed mass properties ($\vec{M}$)

Specifically, the objective function O is defined by

$$O = ||\vec{M}_{\text{tgt}} - \vec{M}||$$

The particular optimization scheme used here is the Levenberg-Marquardt (LM) scheme[3], which is useful for large least-squares problems. The basic LM equation is given by:

$$\left[J^T J + \lambda \text{ diag}\left(J^T J\right)\right]\left(\vec{D}_{\text{new}} - \vec{D}\right) = J^T\left(\vec{M}_{\text{tgt}} - \vec{M}\right)$$

and

$$\Delta\vec{X} = \frac{\partial X}{\partial D}\left(\vec{D}_{\text{new}} - \vec{D}\right)$$

where the Jacobian is defined by:

$$J \equiv \frac{\partial O}{\partial D} = \frac{\partial O}{\partial X}\frac{\partial X}{\partial D}$$

The mass properties are computed by integrating over the surface of the body, which turns into sums over the triangles that tessellate the surfaces. For each triangle, whose verticies are (x_0, y_0, z_0), (x_1, y_1, z_1), and (x_2, y_2, z_2), first

*Associate Professor, Mechanical and Aerospace Engineering, AIAA Associate Fellow.

compute

$$\begin{aligned}
\bar{x} &= (x_0 + x_1 + x_2)/3 \\
\bar{y} &= (y_0 + y_1 + y_2)/3 \\
\bar{z} &= (z_0 + z_1 + z_2)/3 \\
A_x &= [(y_1 - y_0)(z_2 - z_0) - (z_1 - z_0)(y_2 - y_0)]/2 \\
A_y &= [(z_1 - z_0)(x_2 - x_0) - (x_1 - x_0)(z_2 - z_0)]/2 \\
A_z &= [(x_1 - x_0)(y_2 - y_0) - (y_1 - y_0)(z_2 - x_0)]/2
\end{aligned}$$

Then the surface area is computed by accumulating

$$\text{area} = \sum_{\text{tris}} \sqrt{A_x^2 + A_y^2 + A_z^2}$$

The volume can be computed by a discrete form of the divergence theorem:

$$\text{vol} = \sum_{\text{tris}} (\bar{x} A_x + \bar{y} A_y + \bar{z} A_z)/3$$

Sums needed to compute the centroids, which again are derived from the divergence theorem, are then

$$\begin{aligned}
Q_x &= \sum_{\text{tris}} \bar{x}(\bar{x} A_x/2 + \bar{y} A_y + \bar{z} A_z)/3 \\
Q_y &= \sum_{\text{tris}} \bar{y}(\bar{y} A_y/2 + \bar{z} A_z + \bar{x} A_x)/3 \\
Q_z &= \sum_{\text{tris}} \bar{z}(\bar{z} A_z/2 + \bar{x} A_x + \bar{y} A_y)/3
\end{aligned}$$

After all the triangles have been visited, the centroid values are then

$$\begin{aligned}
\text{xcg} &= Q_x/\text{vol} \\
\text{ycg} &= Q_y/\text{vol} \\
\text{zcg} &= Q_z/\text{vol}
\end{aligned}$$

Similar equations are used for the moments of inertia.

II. Initial Optimization Scheme

Initially a straightforward implementation was used, as outlined in Fig. 1.

This scheme is implemented in MPI[4] with a client-server architecture; the operations in the large box are implemented on the client whereas the other execute on the server. The various routine names described below are part of the Application Programming Interface (API) for `ESP`[2].

Processing starts at the top left, where a configuration is loaded (`ocsmLoad`) and the initial design variables ($\vec{D}$) are defined. The configuration is then built (`ocsmBuild`), producing surface grid points ($\vec{X}$). These are sent to the various clients, via MPI, where each client computes (via surface integrals) its contribution to the mass properties ($\vec{M}$). Each client then sends its contributions to the mass properties back to the server, where they are combined. The optimizer then computes the objective function (O). If converged, a message is broadcast to the clients to stop and the server stops. Otherwise if this current optimization step has improved the objective function, the derivative of the objective function with respect to the various mass properties ($\partial O/\partial \vec{M}$) is computed and broadcast to the clients. Each client then computes its contribution (via an adjoint calculation) to the derivative of the objective function with respect to the surface grid points for which it is responsible ($\partial O/\partial \vec{X}$). In the meantime, the client has been computing the sensitivity

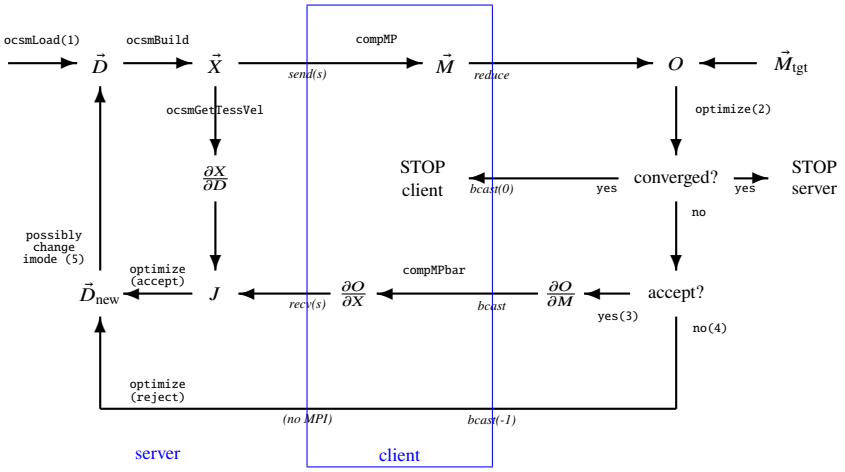


Fig. 1 Basic Levenberg-Marquardt optimization scheme.

of each grid points with respect to the design variables ($\partial\vec{X}/\partial\vec{D}$). (These sensitivities are computed analytically, as described in [5, 6].) These results are then combined with the results that are sent from the clients to for the Jacobian matrix. The LM equation is then employed to produce a new set of design parameters ($\vec{D}_{\text{new}}$) and the whole process repeats. (If the optimization did not produce a better result, the LM scheme increases it λ parameter and the adjoint calculations in the clients are skipped).

To better understand Fig. 1, notes have been added:

- (1) Initialization
  - `imode = 0`
  - `initOpt = 1`
  - `lambda = 1`
- (2) Optimization
  - write history
  - `if (initOpt == 1)`
    - store `dbest`
    - `initOpt = 0`
    - `compAdj = 1`
    - `lambda = 1`
- (3) Accept step (`imode=0`)
  - update `dbest`
  - `compAdj = 1`
  - decrease `lambda`
- (4) Reject step (`imode=0`)
  - revert to `dbest`
  - `compAdj = 0`
  - increase `lambda`

III. Computational Results for Initial Optimization Scheme

To assess this process, it was applied to four test cases. The first test case is a simple 'box" example, where the design variables ($\vec{D}$) and mass properties ($\vec{M}$) are given by:

$$\vec{D} = \begin{bmatrix} \text{xbeg} \\ \text{dx} \\ \text{ybeg} \\ \text{dy} \\ \text{zbeg} \\ \text{dz} \end{bmatrix} \qquad \vec{M} = \begin{bmatrix} \text{area} \\ \text{vol} \\ \text{xcg} \\ \text{ycg} \\ \text{zcg} \\ \text{Ixx} \\ \text{Iyy} \\ \text{Izz} \\ \text{Ixy} \\ \text{Ixz} \\ \text{Iyz} \end{bmatrix}$$

Fig. 2 shows the initial and target configurations and the convergence history for this case. As can be seen, the optimization scheme converges very quickly for this case.

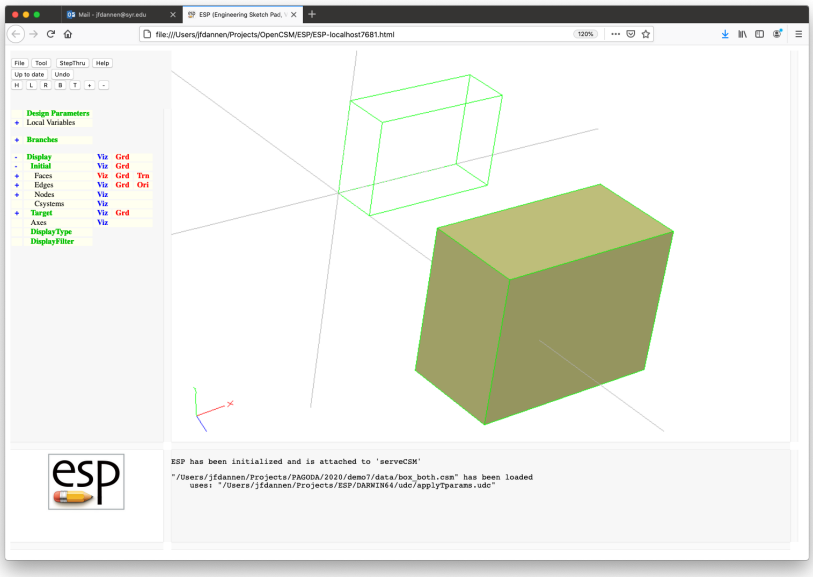
The second test case is a cylinder, where:

$$\vec{D} = \begin{bmatrix} \text{xcent} \\ \text{ycent} \\ \text{zcent} \\ \text{length} \\ \text{radius} \end{bmatrix} \qquad \vec{M} = \begin{bmatrix} \text{area} \\ \text{vol} \\ \text{xcg} \\ \text{ycg} \\ \text{zcg} \\ \text{Ixx} \\ \text{Iyy} \\ \text{Izz} \\ \text{Ixy} \\ \text{Ixz} \\ \text{Iyz} \end{bmatrix}$$

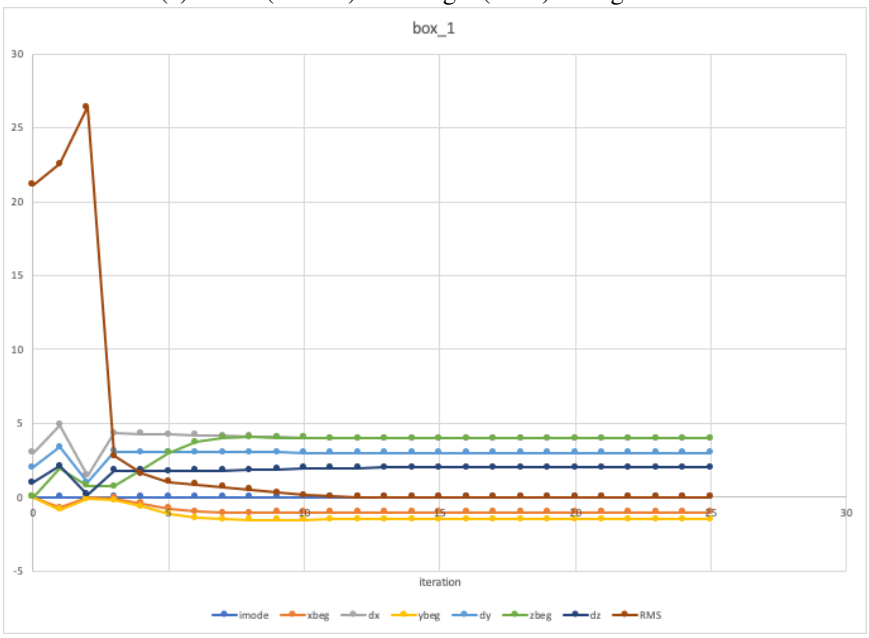
The initial and final configurations and convergence history are shown in Fig. 3. Again rapid convergence can be seen.

The third test case is a simple wing, where:

$$\vec{D} = \begin{bmatrix} \text{area} \\ \text{aspect} \\ \text{taper} \\ \text{sweep} \\ \text{dihedral} \end{bmatrix} \qquad \vec{M} = \begin{bmatrix} \text{area} \\ \text{vol} \\ \text{xcg} \\ \text{ycg} \\ \text{zcg} \\ \text{Ixx} \\ \text{Iyy} \\ \text{Izz} \\ \text{Ixy} \\ \text{Ixz} \\ \text{Iyz} \end{bmatrix}$$

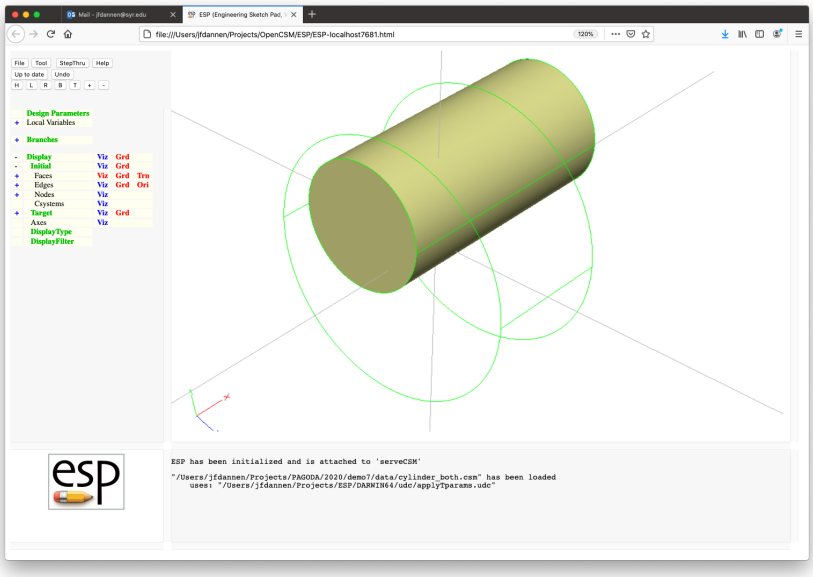


(a) Initial (outline) and Target (solid) configurations

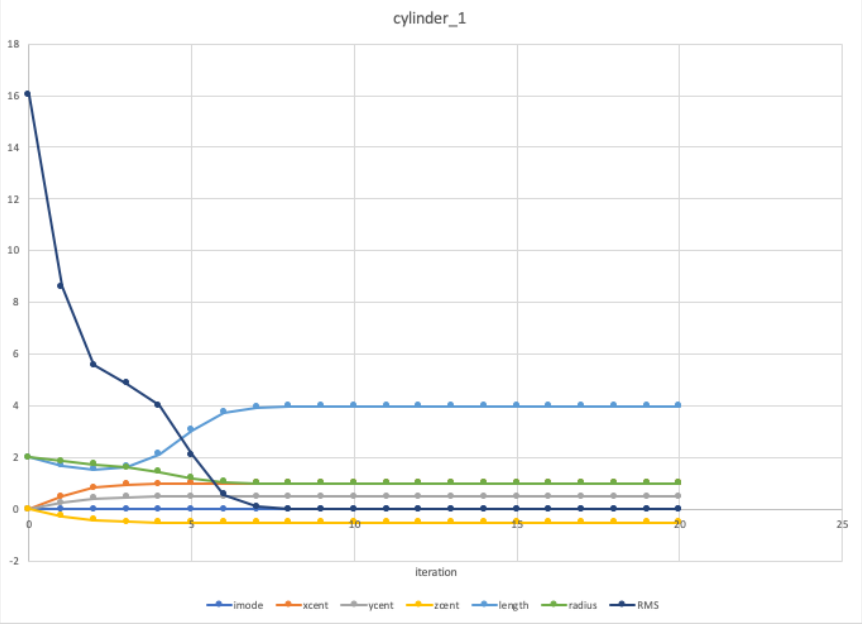


(b) Convergence History

Fig. 2 Box test case.

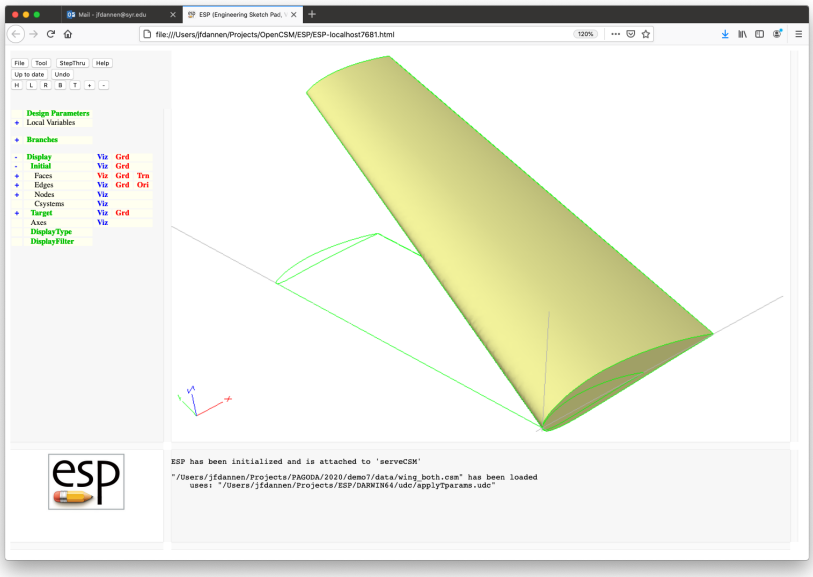


(a) Initial (outline) and Target (solid) configurations

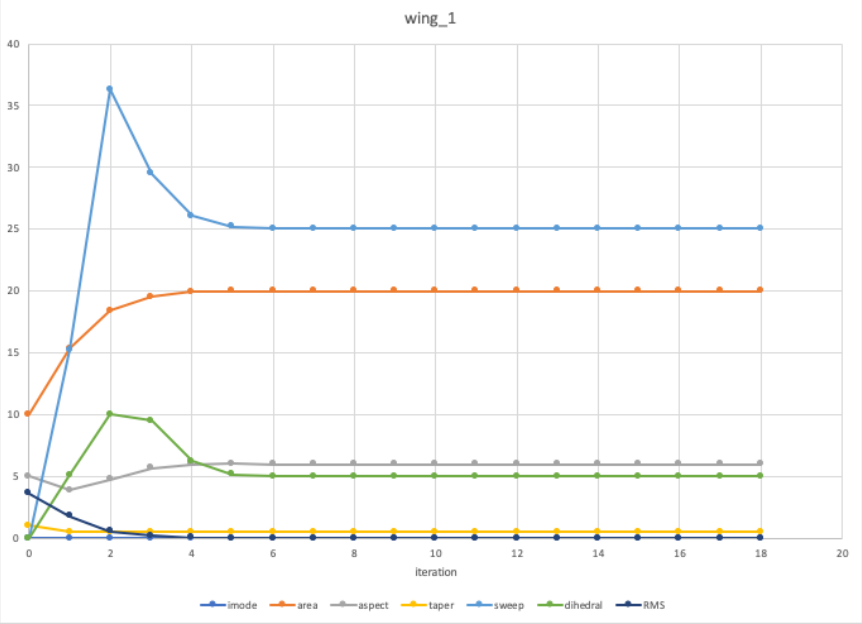


(b) Convergence History

Fig. 3 Cylinder test case



(a) Initial (outline) and Target (solid) configurations



(b) Convergence history

Fig. 4 Wing test case

As seen in Fig. 4, this case also converges very quickly.

The final test case, a boxy-plane is defined by:

$$\vec{D} = \begin{bmatrix} \text{fuse : length} \\ \text{fuse : width} \\ \text{fuse : height} \\ \text{wing : fxroot} \\ \text{wing : fzroot} \\ \text{wing : chord} \\ \text{wing : span} \\ \text{wing : thick} \end{bmatrix} \qquad \vec{M} = \begin{bmatrix} \text{area} \\ \text{vol} \\ \text{xcg} \\ \text{ycg} \\ \text{zcg} \\ \text{Ixx} \\ \text{Iyy} \\ \text{Izz} \\ \text{Ixy} \\ \text{Ixz} \\ \text{Iyz} \end{bmatrix}$$

and shown in Fig. 5. This time the convergence is not quite as good, but still acceptable.

IV. Modified Optimization Scheme

Even though the optimization worked quite well in all cases, it required a complete rebuild of the configuration (`ocsmBuild`) and computation of the tessellation sensitivities (`ocsmGetTessVel`) at every optimization cycle. Therefore a revised optimization scheme was proposed, as shown in Fig. 6.

Notes for the revised scheme are:

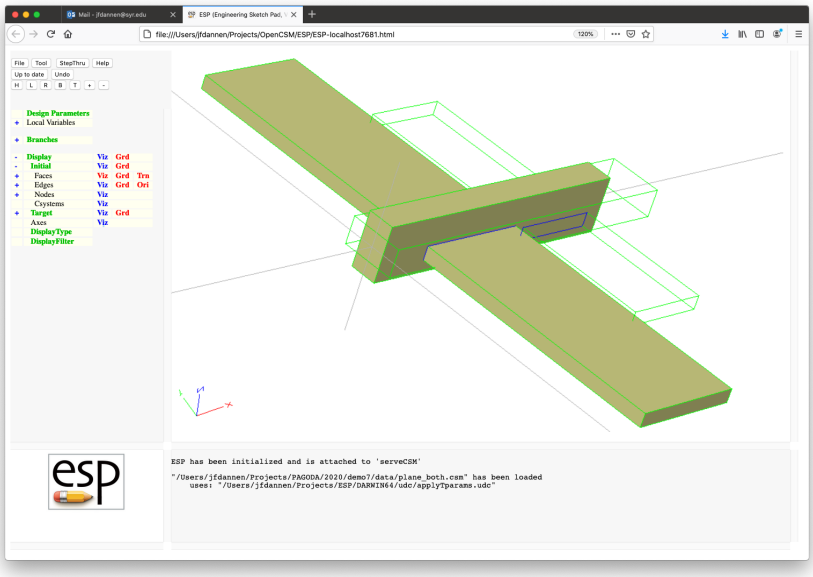
- (5) Possibly change `imode`
  - `if (rms < f*rms[0] and compAdj==1)`
    - increment `imode`
  - `if (imode >= nmode)`
    - `imode = 0`
- (6) Converged (`imode>0`)
  - `imode = 0`
  - `initOpt = 0`
- (7) Reject step (`imode>0`)
  - revert to `dbest`
  - increase `lambda`
  - `imode = 0`
  - `compAdj = 0`
  - `initOpt = 0`
- (8) Accept step (`imode>0`)
  - `compAdj = 1`
  - decrease `lambda`

The major difference between the original and revised scheme is the fact that in the revised scheme, the new geometry ($\vec{X}$) is produced by the sensitivity at the prior stage. This guarantees consistency between the sensitivities and the new shape. But, it requires that the shape changes be relatively small (that is, in the linear range).

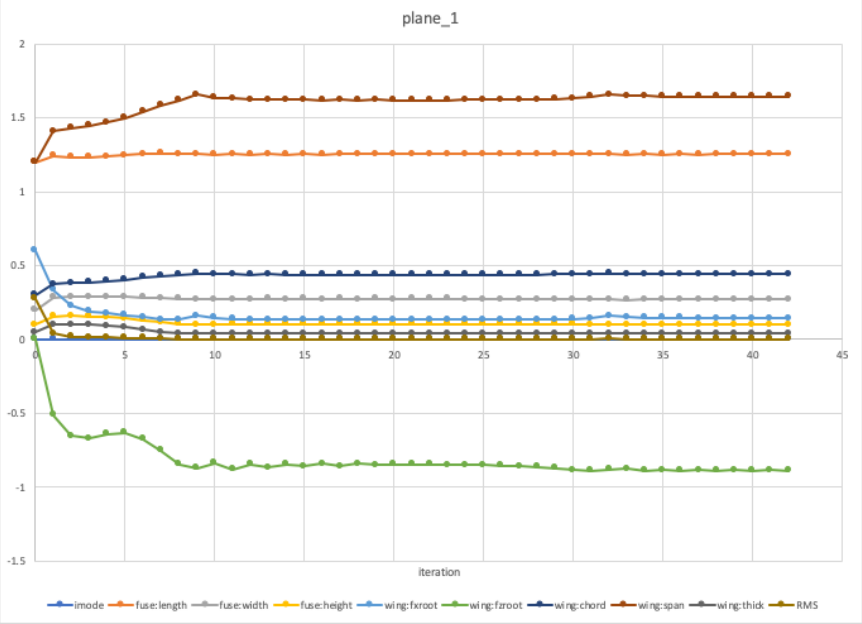
V. Computational Results for Modified Optimization Scheme

Fig. 7 shows the effect of taking this shortcut `nmode-1` times. As can be seen, for small `nmode` there can be some savings.

The results for for all test cases are shown in Table 1. Shown are the number of builds, the number of times that the mass property calculations were performed, the number of steps accepted and rejected by the optimizer, and the number



(a) Initial (outline) and Target (solid) configurations



(b) Convergence History

Fig. 5 Boxy-plane test case

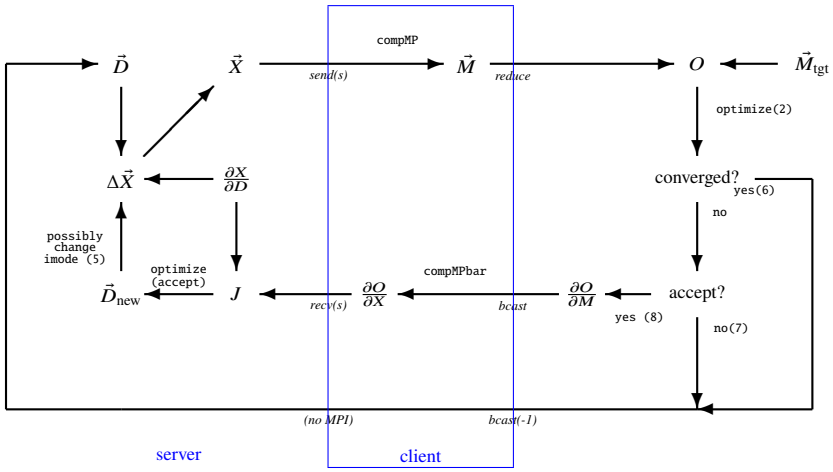
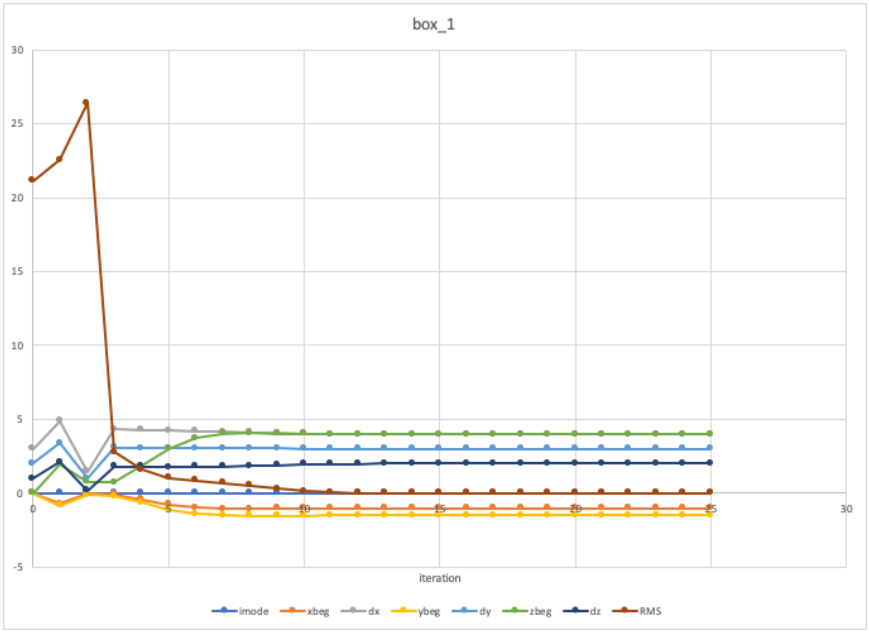


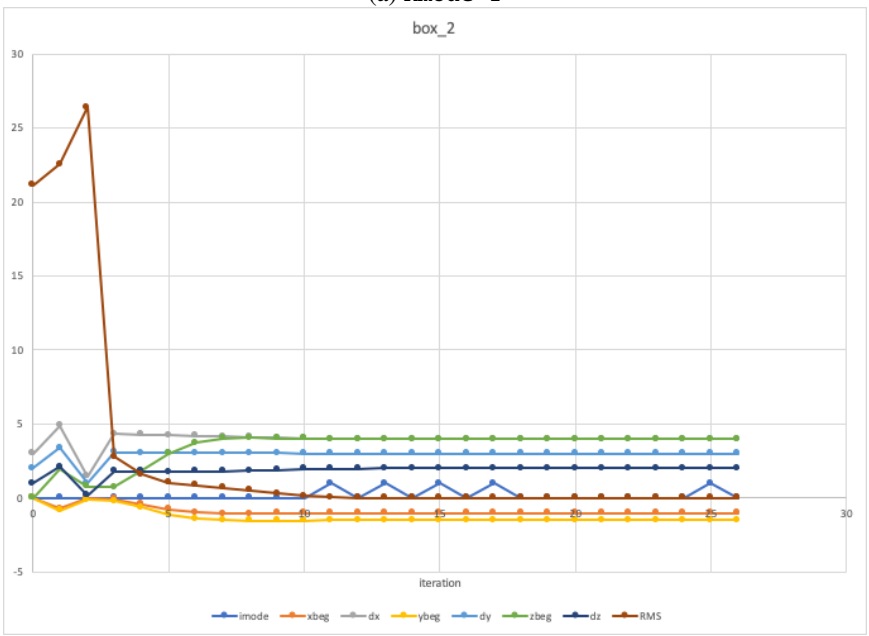
Fig. 6 Revised optimization scheme

Table 1 Summary of results of various `nmode` for all test cases.

case	nmode	#build	#massprop	#accept	#reject	#adjoint
box	1	26	26	20	5	20
	2	22	27	17	9	16
	5	16	26	19	6	19
	10	12	21	17	3	17
cylinder	1	21	21	13	7	13
	2	17	21	14	6	14
	5	18	34	28	5	25
	10	15	29	21	7	21
wing	1	19	19	10	8	10
	2	55	77	49	27	49
	5	14	29	22	6	22
	10	12	29	22	6	22
plane	1	43	43	20	22	20
	2	46	47	21	25	21
	5	43	45	21	23	21
	10	43	45	21	23	21

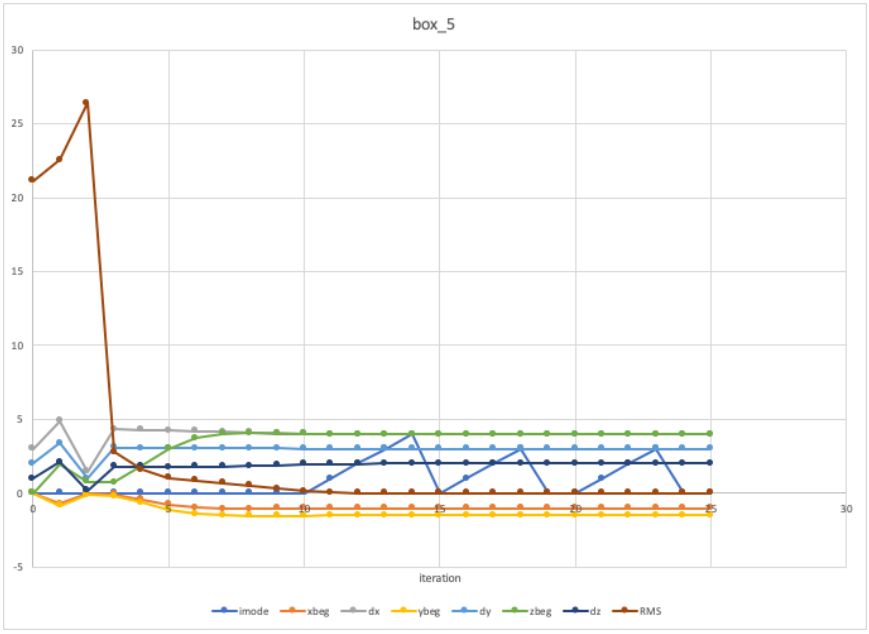


(a) nmode=1

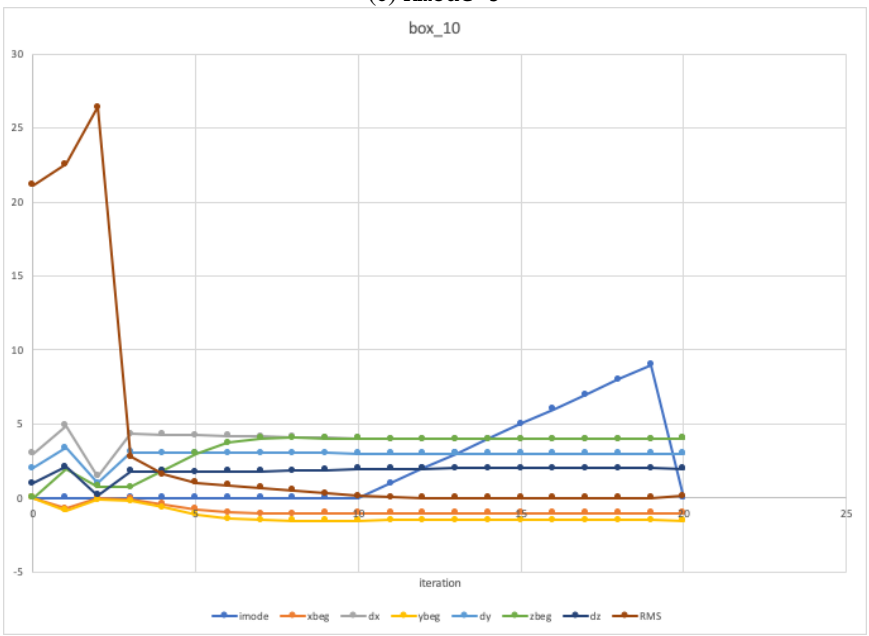


(b) nmode=2

Fig. 7 Convergence histories for box test case.



(c) nmode=5



(d) nmode=10

Fig. 8 Convergence histories for box test case. (continued)

of adjoint calculation. As can be seen, for cases where the changes are somewhat "linear", the revised scheme holds promise. But for cases where non-linearities are important, the revised scheme is not very effective.

VI. Conclusions

Two different optimization schemes were examined for use in an the adjoint-based optimization process.

The first scheme requires a re-build of the configuration at each optimization cycle, but is suitable in cases where the changes in the design variables is large. Fortunately the time to rebuild is generally much smaller that the time required for the flow solution and it adjoint, so re-building has a negligible impact of the overall solution time. In this first scheme, the needed sensitivities can be computed simultaneously with the flow solution, so only a very small amount of work is needed to multiply these sensitivities by the flow's adjoin solution.

The second scheme essentially linearizes the configuration (with respect to the design variables), obviating the need for a re-build. Also, in this second scheme, the sensitivities can be computed once and re-used during each optimization cycle. While much more computationally efficient than the first scheme, it only works when the changes to the design variables are small and those changes result in "almost linear" changes to the model.

A third scheme, which is a hybrid between the first two, was also examined. But the computational results show little gain.

Acknowledgments

This work was conducted under NASA Grant NNX16AQ15A, with William Jones as the Technical Monitor.

References

[1] Nielsen, E.J., "Adjoint-Based Aerodynamic Design of Complex Aerospace Configurations", ASME 2016-7573, 2016.

[2] Haimes, R. and Dannenhoffer, J.F., "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry", AIAA-2013-3073, June 2013.

[3] Marquardt, D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters", *SIAM Journal on Applied Mathematics*, 11 (2): 431-441, 1963.

[4] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI* (2nd ed). MIT Press, 1999.

[5] Dannenhoffer, J.F., and Haimes, R., "Design Sensitivity Calculations Directly on CAD-Based Geometry", AIAA-2015-1370, January 2015.

[6] Dannenhoffer, J.F., and Haimes, R., "Using Design-Parameter Sensitivities in Adjoint-Based Design Environments", AIAA-2017-0139, January 2017.