Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods

Patrick LeGresley

Department of Aeronautics and Astronautics
Stanford University

August 25, 2005

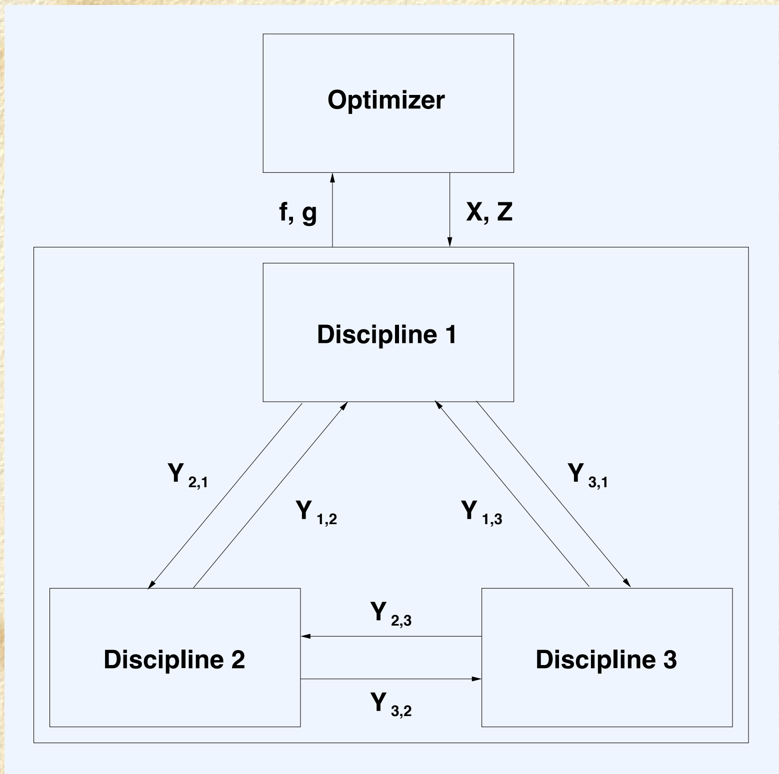
Outline

- Motivation
- Proper Orthogonal Decomposition (POD)
  - Solution Approximation with POD
  - Bi-Level Integrated System Synthesis (BLISS) with POD
- Future Research Areas

Motivation

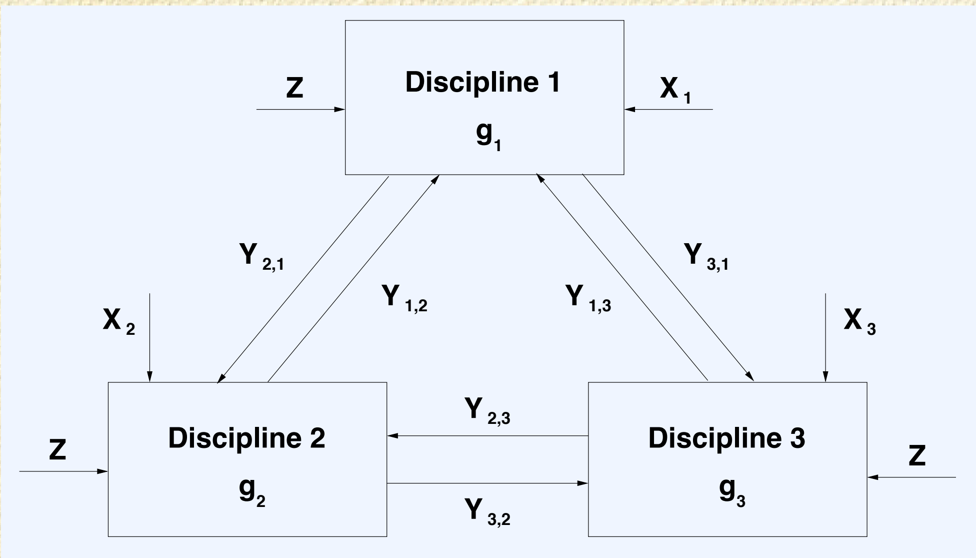
- Obvious need for multidisciplinary design in aerospace vehicles.
- A variety of design architectures exist which have trade offs between implementation cost and computational efficiency, applicability to computationally costly objective functions, large numbers of design variables, etc.
- Approximation techniques may be helpful to alter the characteristics of a given architecture.

Monolithic



- Direct optimization of multidisciplinary analysis.
- Coupling variables, Y, absent from optimization problem.
- Primary challenge for gradient based optimization is obtaining sensitivities of coupled system.
- No discipline autonomy.

Bi-Level Integrated System Synthesis (BLISS)



- Discipline optimization problems formulated from Global Sensitivity Equations (GSE).
- Tighter coupling than CO.
- Gradients rather expensive to compute.

ROM and Approximation Models

- Reduced order or approximation models offer the opportunity to improve the computational performance of design decomposition methods.

- Variety of methods

  - Response surfaces

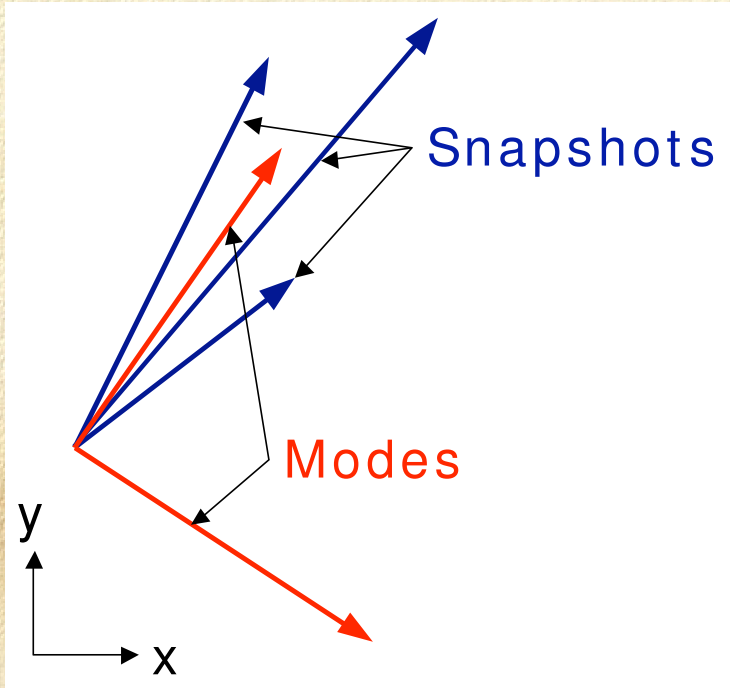
  - Fourier series

  - Variable fidelity models

  - Proper Orthogonal Decomposition (POD)

Proper Orthogonal Decomposition

- Method for computing the *optimal* **linear basis** (*modes*) for representing a sample set of data (*snapshots*).



$$u_M = \sum_{j=1}^{M} a_j \varphi_j$$

Survey of POD Use in Fluids

- Post processing for identification of coherent structures in turbulence (Lumley, 1967).
- ROM for control law design (Rediniotis et al., 1999)
- Unsteady aerodynamic and aeroelastic behavior (Hall et al., 1999).
- Flutter prediction and non-linear panel response (Beran and Pettit, 2000-2001).
- Unsteady shocks in a nozzle (Lucia et al., 2001).

POD - Linear Expansion

- We are seeking finite dimensional representations of a function in terms of a basis (modes) which allows a *linear* approximation to be constructed

$$u_M = \sum_{j=1}^{M} a_j \varphi_j$$

- Assume we have a set of N sample or representative datasets (snapshots), $\{u^k\}$, and we want to choose the basis functions such that they best describe a typical member of the sample data.

POD - Basis Optimality

- The basis functions should be chosen so that they maximize the averaged projection of our ensemble of functions onto the basis functions

$$\max_{\varphi} \frac{\langle |(u,\varphi)|^2 \rangle}{\|\varphi\|^2}$$

where $\langle \cdot \rangle$ is an averaging operator, $|\cdot|$ denotes the modulus, and $\|\cdot\|$ is the L^2 norm given by

$$\|f\| = (f,f)^{\frac{1}{2}}$$

POD - Autocorrelation Functions

- This is a constrained optimization problem where the function to be maximized is

$$J[\varphi] = \langle |(u,\varphi)|^2 \rangle - \lambda(\|\varphi\|^2 - 1)$$

- The problem can have multiple local maxima corresponding to each of the basis functions and can be shown to require that

$$\int_\Omega \langle u(x)u(x')\rangle \varphi(x')dx' = \lambda\varphi(x)$$

such that the optimal basis is composed of the eigenfunctions where the autocorrelation function is $\langle u(x)u(x')\rangle = R(x,x')$.

POD - Finite Dimensions

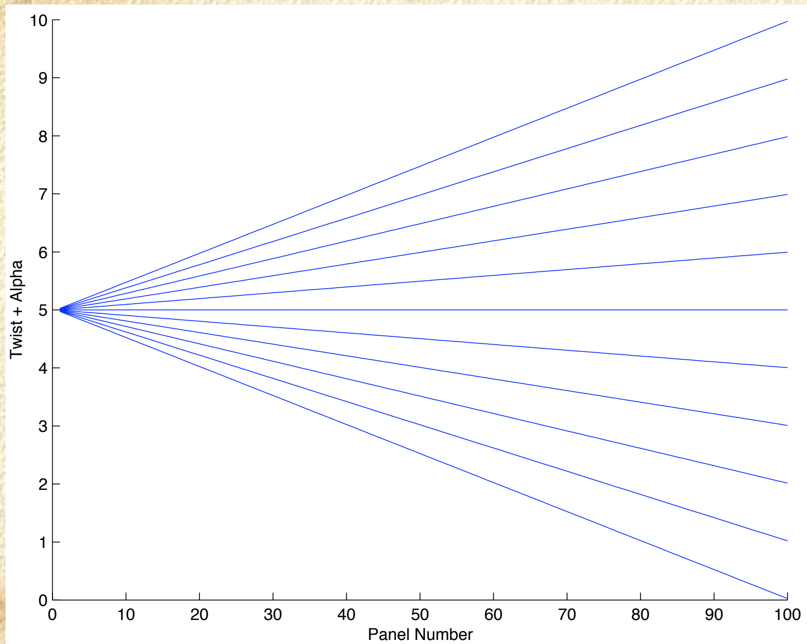
- For numerical computations the ensemble of functions becomes a group of vectors and the autocorrelation function becomes an autocorrelation tensor

$$R = \langle u \otimes u \rangle$$

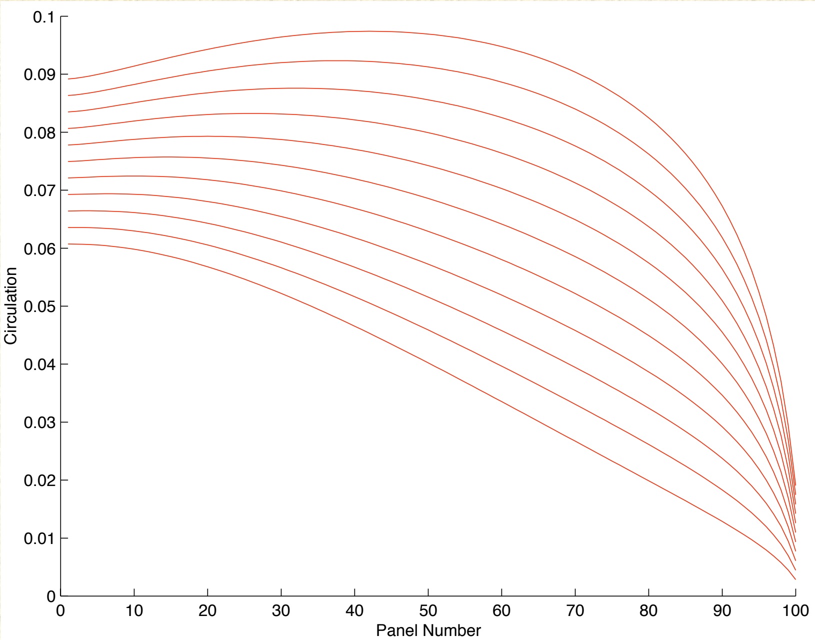
  where the eigenvectors of the problem are the principal axes of the data.

- Method of snapshots replaces computation of the autocorrelation matrix to an *M x M* eigenvalue/eigenvector problem, where *M* is the number of snapshots.

Simple POD Example

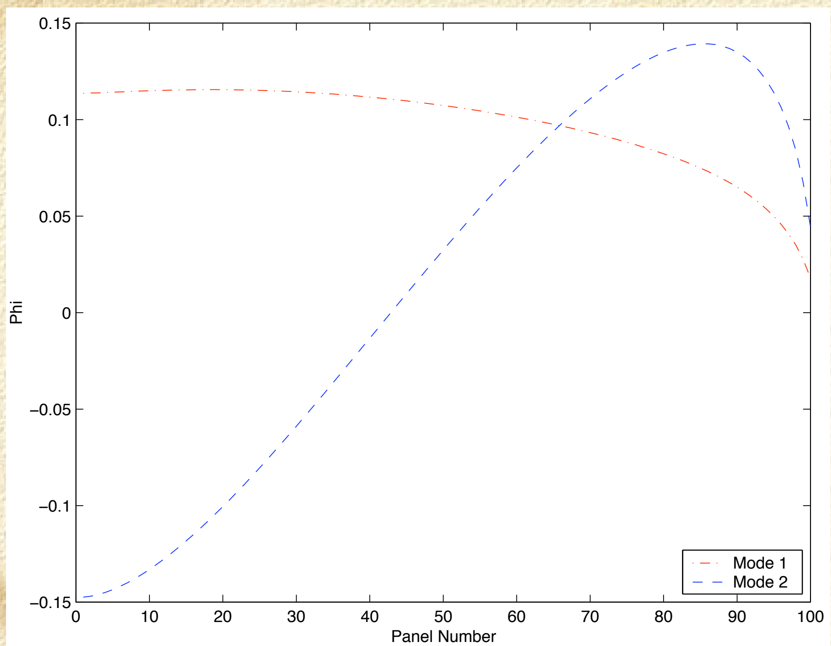


Sample Twist Distributions

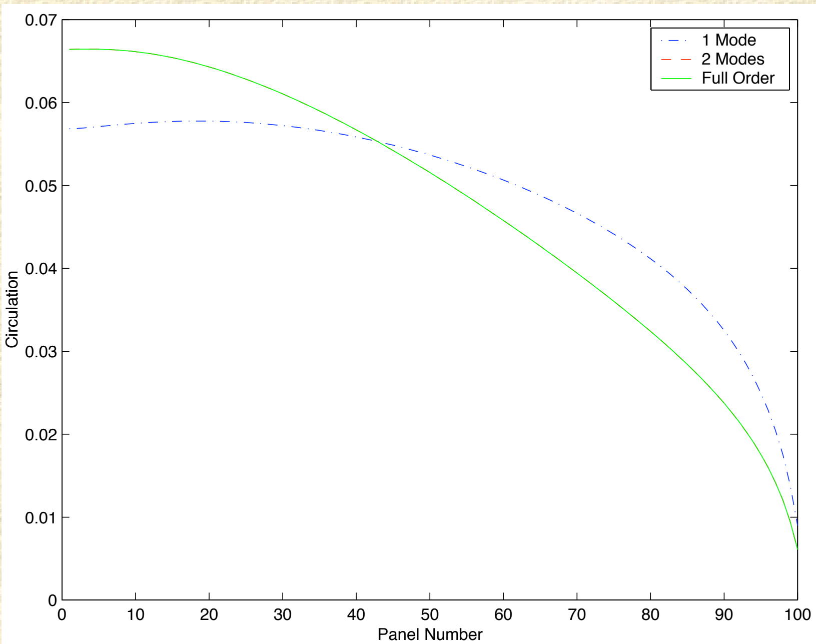


Corresponding Circulation Distributions

Simple POD Example (cont.)



Modes (POD basis)



Solution Approximation

Solution Approximation

- Conventional CFD solver computes solution to discrete approximation of flow equations such as

$$\frac{d}{dt}(w_{ij}V_{ij}) + R(w_{ij}) = 0,$$

which is applied to each volume in the mesh and there are an equal number of equations

- and degrees of freedom.

- Using POD the state vector is represented as

$$w(x,y) = \sum_{i=1}^{M} \eta_i \varphi_i(x,y).$$

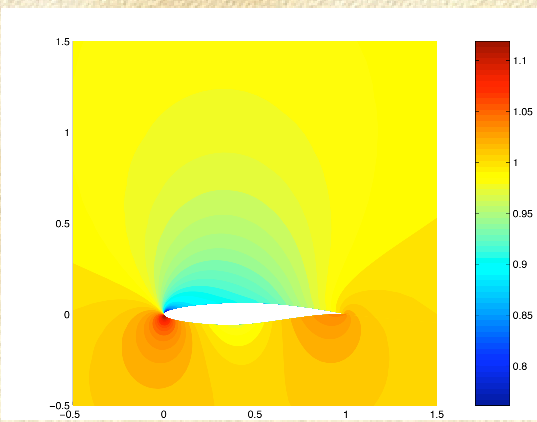
Solution Approximation (cont.)

- For steady problems our governing equations are now of the form

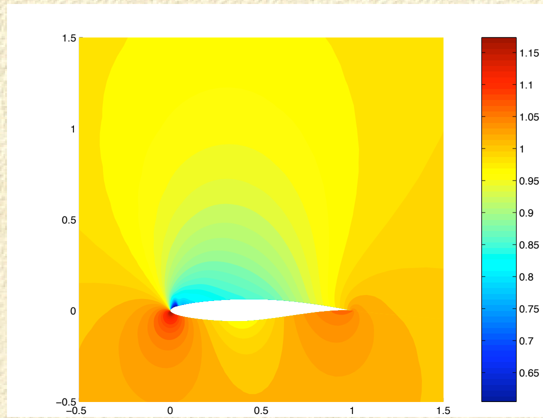
$$R(\eta) = 0$$

- The system is overdetermined as we will typically have many fewer unknown coefficients in the POD expansion than degrees of freedom in the original problem.
- Any of a number of weighted residual methods can be applied - collocation, least squares, Galerkin, etc.

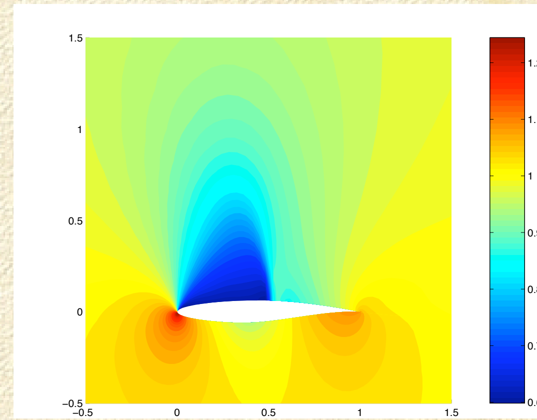
Example Snapshots and Modes



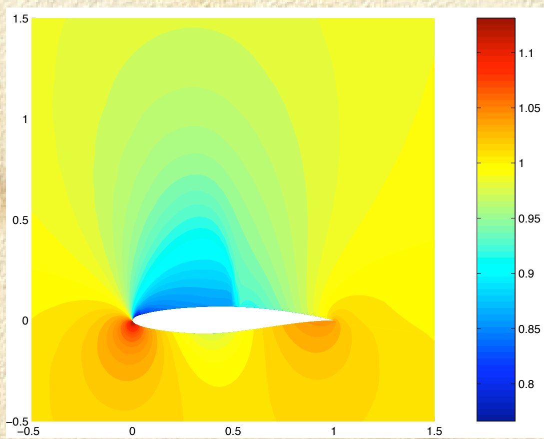
Density, M=0.50



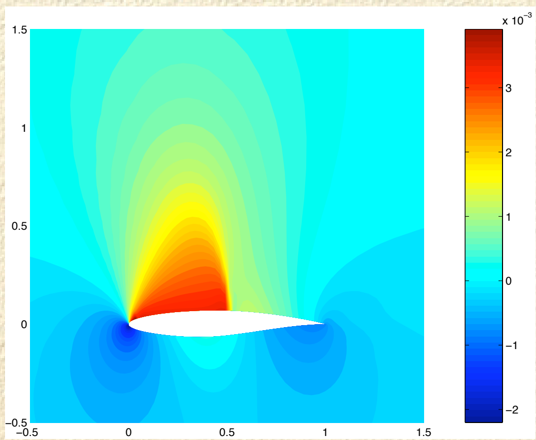
Density, M=0.60



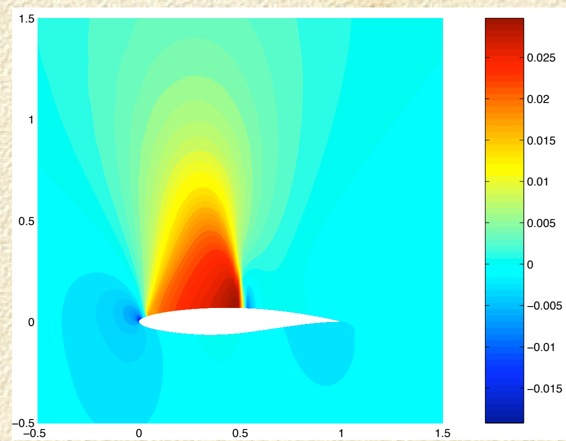
Density, M=0.70



Density Average

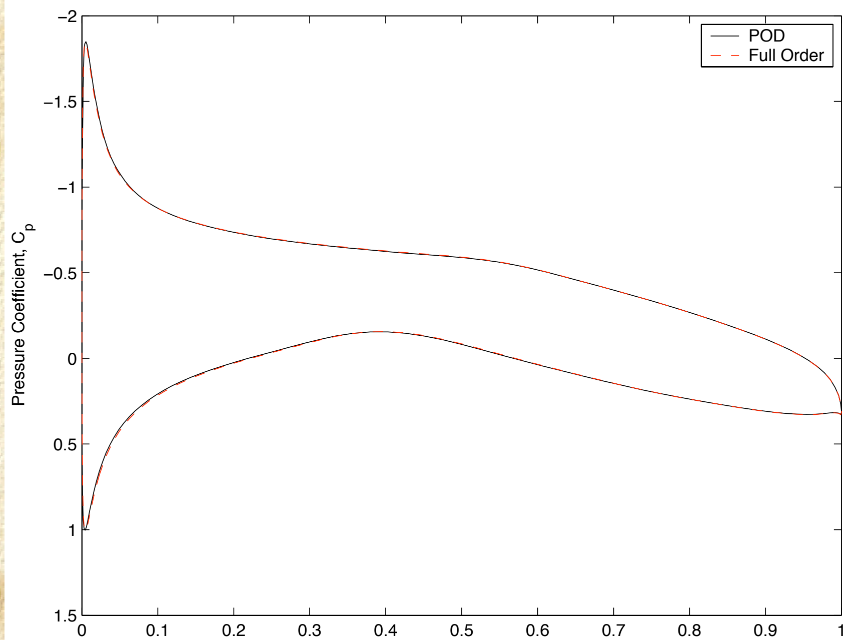


Density Mode 1

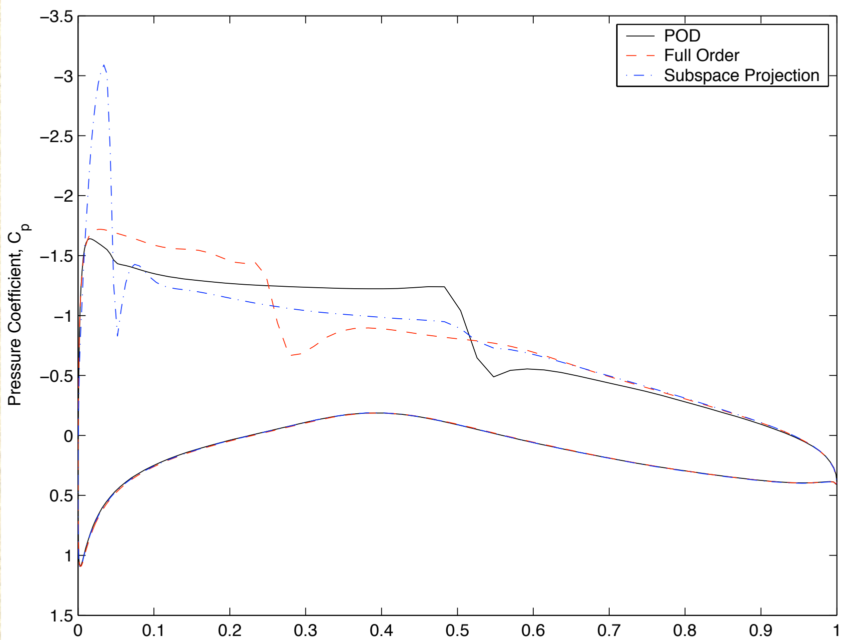


Density Mode 2

Sample Results



Pressure Coefficient, M=0.35



Pressure Coefficient, M=0.67

Domain Decomposition

- If the basis used to generate the approximate solutions is in general sufficient, the solution will be good in most of the domain with just a small portion of the domain which cannot be resolved using the POD basis.
- Lucia et al. (2001) used an *a priori* decomposition for shocks in a nozzle.
- Can we do the decomposition, as needed, in real-time ?

Adjoints for Error Estimation

- Adjoint solutions can be used to generate superconvergent functionals or error bounds (Pierce and Giles, 2000).
- Venditti and Darmofal (2001) used a discrete formulation to drive a grid adaptation procedure.
- The cost of these error estimations are about the same as solving the problem we are trying to approximate.

Error Estimation

- Similar technique can be applied on subsets of the domain to prioritize which portions of the domain to augment with additional basis functions.

- Let subscript *POD* denote the solution as computed using the POD basis alone

$$w_{POD} = \sum_{m=1}^{M} \eta_m \varphi_m(x)$$

Error Estimation (cont.)

- And let *DD* denote the solution using the POD basis functions plus some additional basis functions with local support only, $\hat{\varphi}_n$

$$w_{DD} = \sum_{m=1}^{M} \eta_m \varphi_m(x) + \sum_{n=1}^{N} \hat{\eta}_n \hat{\varphi}_n(x)$$

where the addition of the new basis functions represents the decomposition of the problem into POD and full order subdomains.

Error Estimation (cont.)

- The residual operators representing the governing equations for this system are

$$\begin{Bmatrix} \hat{R}_k(\eta) \\ R_k(\hat{\eta}_n) \end{Bmatrix} = R_{DD} = 0$$

- Define as w_{DD}^{POD} the solution using the original POD expansion transferred to the new basis, the obvious transfer being that η is unchanged and the $\hat{\eta}$ are zero.

Error Estimation (cont.)

- The residual operator is then expanded as

$$\mathrm{R}_{DD}(w_{DD}) = R_{DD}(w_{DD}^{POD}) + \left.\frac{\partial R_{DD}}{\partial w_{DD}}\right|_{w_{DD}^{POD}} (w_{DD} - w_{DD}^{POD}) + \cdots$$

- This can be inverted to give an approximation to the change in the state vector

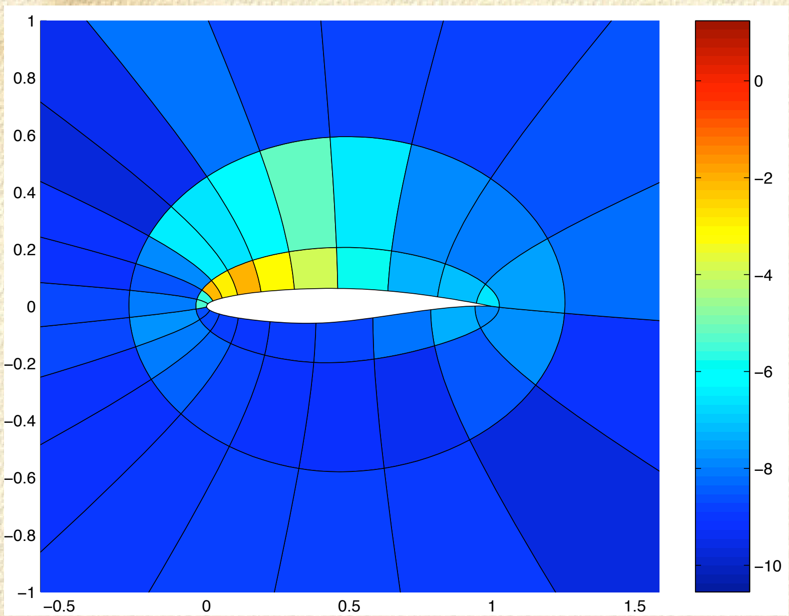
$$(w_{DD} - w_{DD}^{POD}) \approx -\left[\left.\frac{\partial R_{DD}}{\partial w_{DD}}\right|_{w_{DD}^{POD}} \right]^{-1} R_{DD}(w_{DD}^{POD})$$

- The change in the state vector is the first order estimate of the change in the solution if one were to decompose the problem into a POD and full order subdomain.

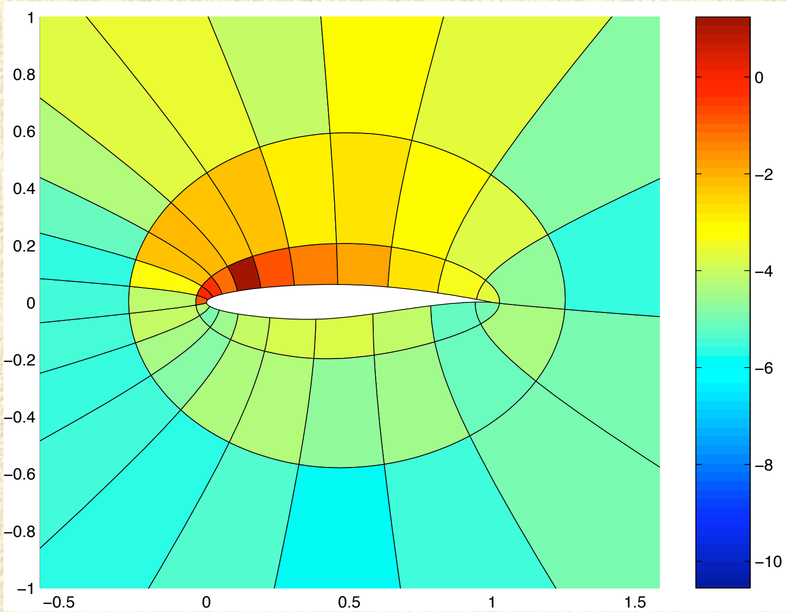
Error Estimation (cont.)

- Estimated error by introducing new basis functions in blocks of 8 by 8 cells (in a domain which was 160 by 32 cells).
- To characterize the need to introduce the new basis functions the change in the state vector was used to compute the corresponding change in pressure.
- Each block was assigned a value equal to the norm of the change in pressure for the cells inside that block.

Error Estimation Results

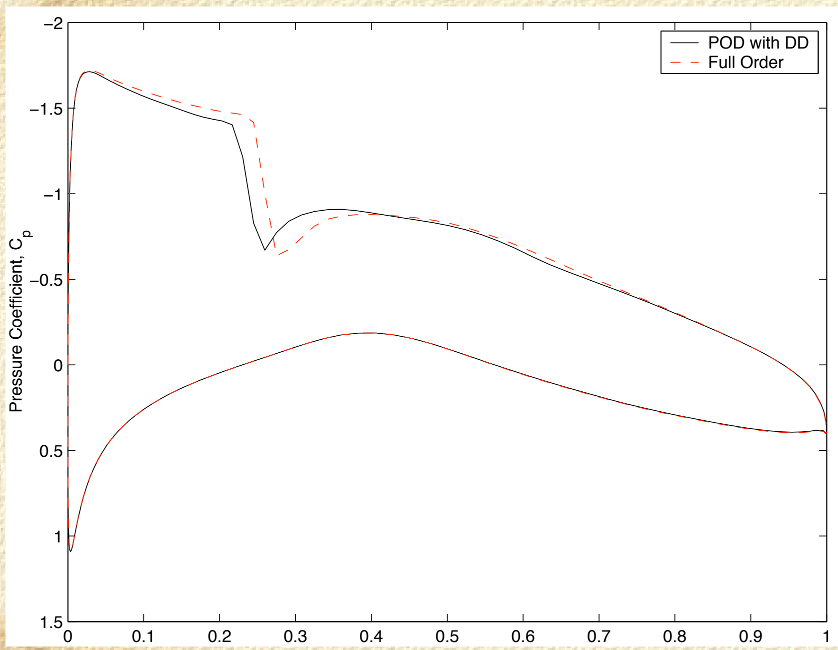


Error Estimation,
M = 0.60

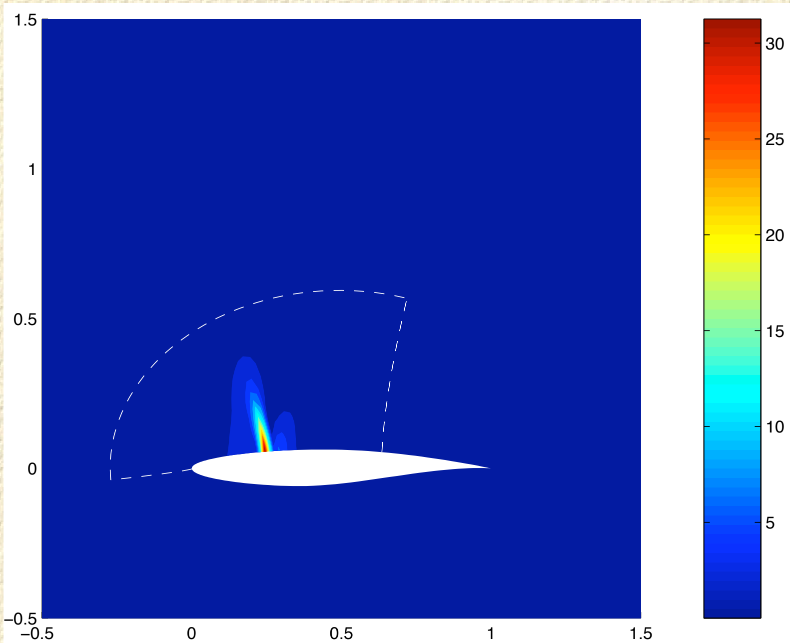


Error Estimation,
M = 0.67

Domain Decomposition Results



Pressure Coefficient, M=0.67



Percentage Error in Pressure Coefficient, M=0.67

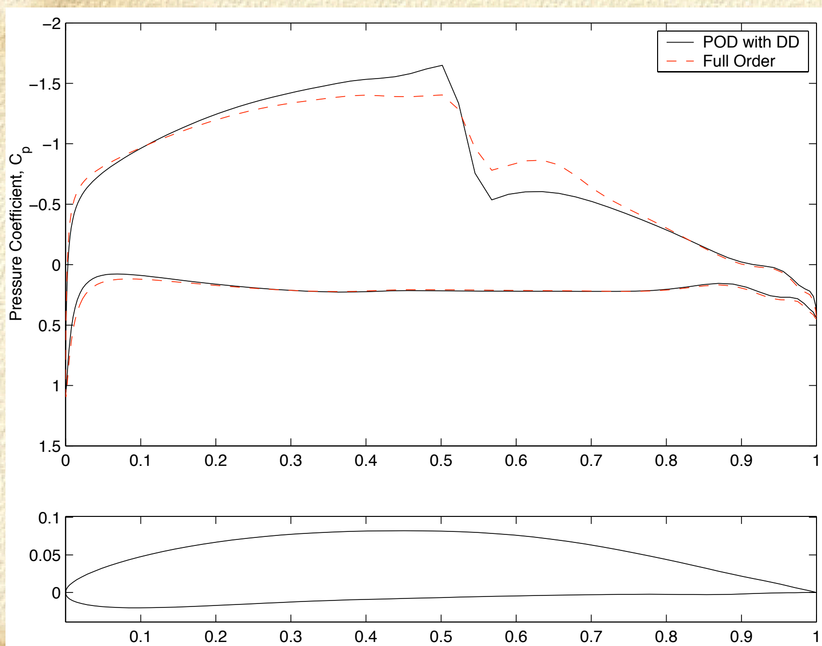
Computational Costs

- Cost of solving full order equations is essentially unchanged in domain decomposition, but now there are many fewer of them.
- Cost savings in 2-D are in the range of 50-75%.
- More compelling in 3-D where an approximate solution could be 4 to 5 times faster.

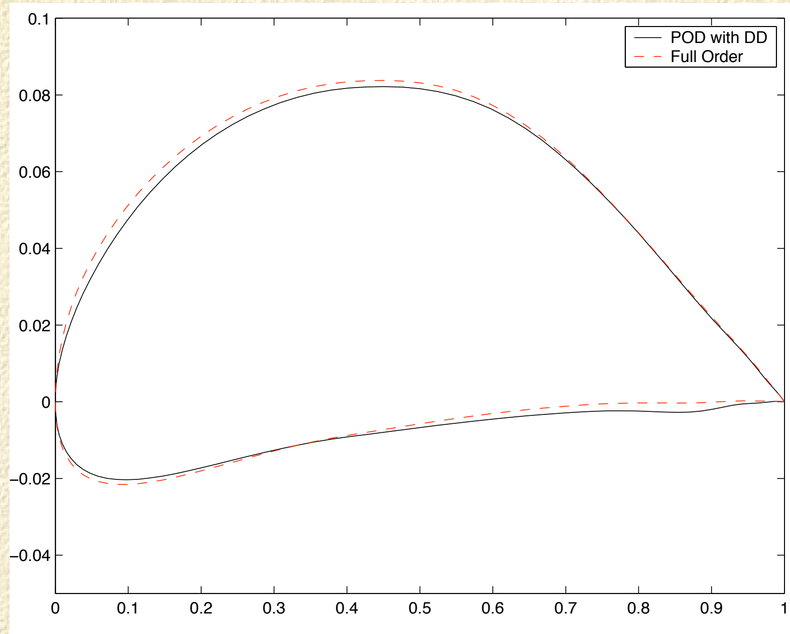
Drag Minimization

- Used as a model problem only.
- Objective is to minimize drag at M=0.67 and fixed lift coefficient.
- Optimization using POD model constructed from snapshots of baseline NACA 4410 airfoil, various geometrically perturbed variations (with bump functions), and two different angles of attack.
- Gradients from finite differencing of reduced order model.

Drag Minimization Results



Comparison of Full Order and POD Pressure Coefficient, 15 iterations



Geometry Comparison, 15 iterations

Contributions

- Theoretical background, implementation, and applications of POD as a reduced order model for systems of non-linear partial differential equations with an emphasis on *design applications*.

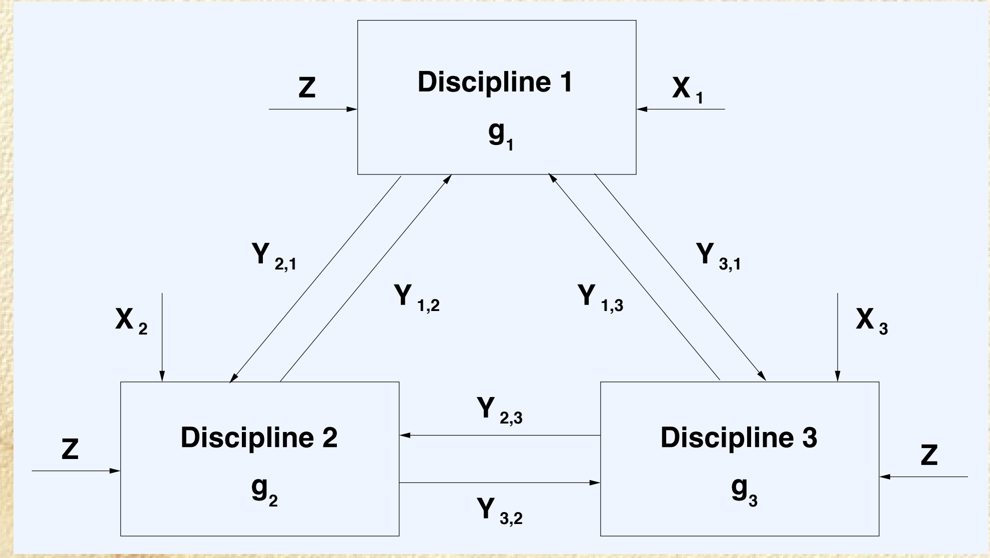
- General least squares residual method applicable to non-linear, systems of equations.

- Use of a *dynamic* domain decomposition yields an approximation method with variable accuracy and computational cost.

Bandwidth Reduction in BLISS

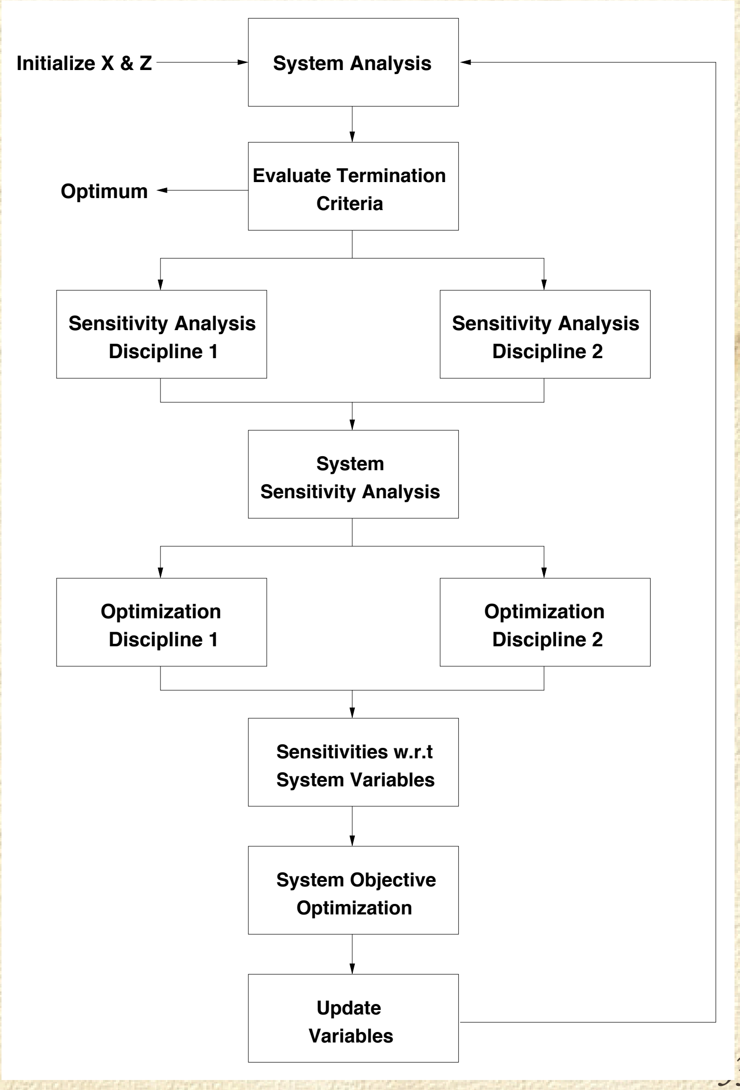
- Typically accomplished using response surfaces (Kodiyalam & Sobieski, 2001).

- POD can be viewed as a bandwidth reduction method, replacing the interaction variables with the more compact scalar coefficients in the POD expansion.



BLISS

- Design variables separated into global (Z) and discipline (X).
- Synthetic objective functions for each discipline based on Global Sensitivity Equations.
- Disciplines optimize w.r.t. X variables while satisfying local constraints.
- System optimization to update Z.



BLISS - Sensitivity Analysis

- Need to formulate objectives for discipline optimizations, where improvements will result in the minimization of the overall system objective function, Φ.

- Total derivatives of Y wrt the *j*-th local design variable in discipline *r* are computed using the Global Sensitivity Equations (GSE)

$$[A]\left\{\frac{dY}{dx_{r,j}}\right\} = \left\{\frac{\partial Y}{\partial x_{r,j}}\right\}$$

BLISS - Sensitivity Analysis (cont.)

- A is a square matrix of dimensionality equal to the number of interaction variables and made up of sub-matrices as follows for the case of 3 disciplines

$$A = \begin{bmatrix} I & A_{1,2} & A_{1,3} \\ A_{2,1} & I & A_{2,3} \\ A_{3,1} & A_{3,2} & I \end{bmatrix}$$

  where I is the identity matrix and $A_{r,s}$ is the matrix of the gradients of interaction variable outputs with respect to interaction variable inputs

$$A_{r,s} = -\frac{\partial Y_r}{\partial Y_s} \quad r,s = 1,2,3$$

BLISS - Gradient Computation

- Costly computation that has dependence on the number of design variables and the number of coupling variables.
- It is expected that where applicable, efficient methods such as adjoints are used.
- Where one needs to compute sensitivities of a large number of inputs to a large number of outputs there are no computationally efficient means to do so - order reduction may be very advantageous.

BLISS - Objective Expansion

- With the system sensitivity analysis computed from the GSE the overall system objective function can be expanded in a Taylor series as a function of X

$$\Phi = \Phi_0 + \frac{d\Phi}{dX_1}^T \Delta X_1 + \frac{d\Phi}{dX_2}^T \Delta X_2 + \frac{d\Phi}{dX_3}^T \Delta X_3 + \cdots$$

- The first order change in the overall objective function due to a change in the local design variables is

$$\Delta\Phi = \frac{d\Phi}{dX_1}^T \Delta X_1 + \frac{d\Phi}{dX_2}^T \Delta X_2 + \frac{d\Phi}{dX_3}^T \Delta X_3 = \sum_i \frac{d\Phi}{dX_i}^T \Delta X_i \;\; i = 1, 2, 3$$

BLISS - Discipline Optimization

- The contribution of the *i*-th discipline is

$$\phi_i = \frac{d\Phi}{dX_i}^T \Delta X_i$$

- The optimization problem for each subproblem, such as for discipline two, is

Given:	X_2, Z, and $Y_{2,1}$, $Y_{2,3}$
Find:	ΔX_2
Minimize:	ϕ_2
Satisfy:	$G_2 \leq 0$

BLISS - System Level Gradients

- We have completed the discipline optimizations and now proceed with the system level optimization using the design variables Z.
- Need the derivatives of Φ with respect to Z.
- BLISS/A computes these derivatives by a modified GSE.
- BLISS/B uses an algorithm that computes them from the Lagrange multipliers of the discipline optimizations.

BLISS - System Level Optimization

- Once we have computed the derivatives using either BLISS/B or BLISS/A the overall system level optimization may be performed

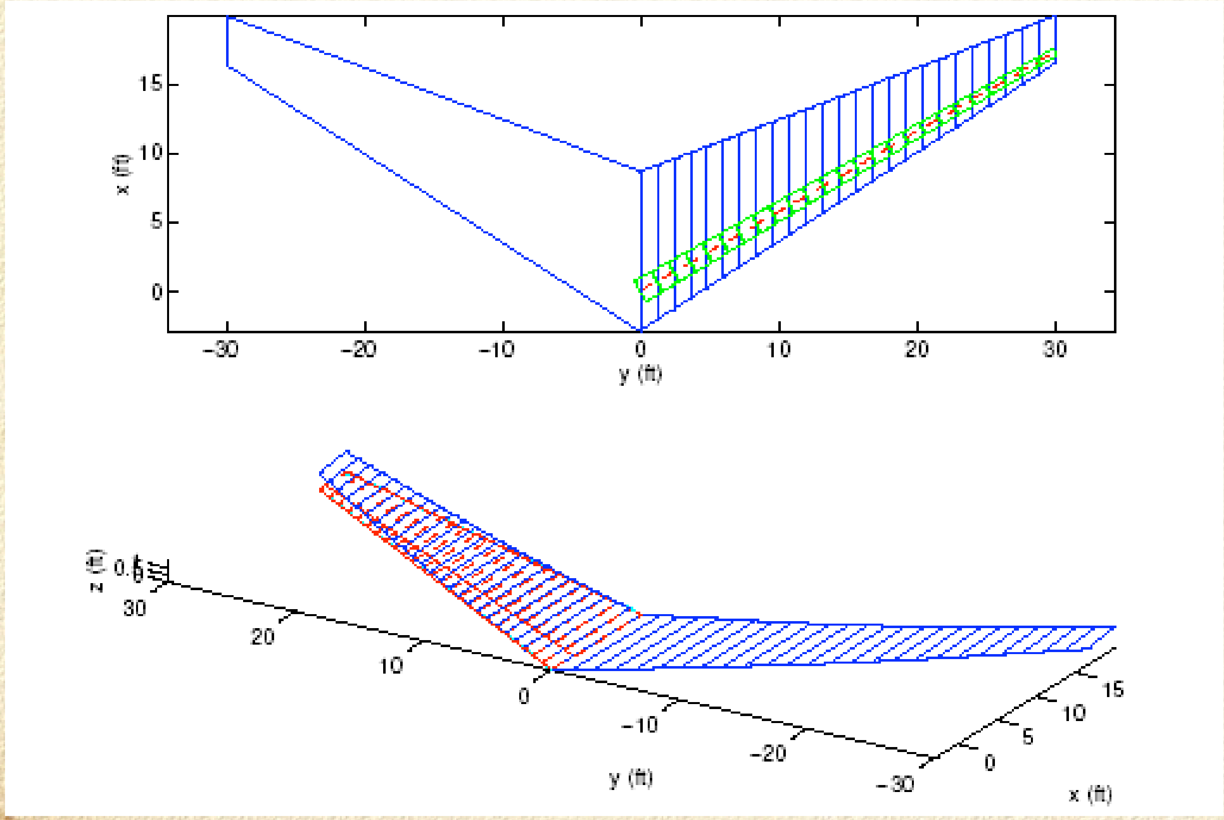
Given:	Z and Φ_0
Find:	ΔZ
Minimize:	$\Phi = \Phi_0 + \dfrac{d\Phi}{dZ}^T \Delta Z$
Satisfy:	$ZL \leq Z + \Delta Z \leq ZU$
	$\Delta ZL \leq \Delta Z \leq \Delta ZU$

- Multidisciplinary analysis is repeated with the updated values of the discipline and global design variables.

BLISS - Low-Fidelity

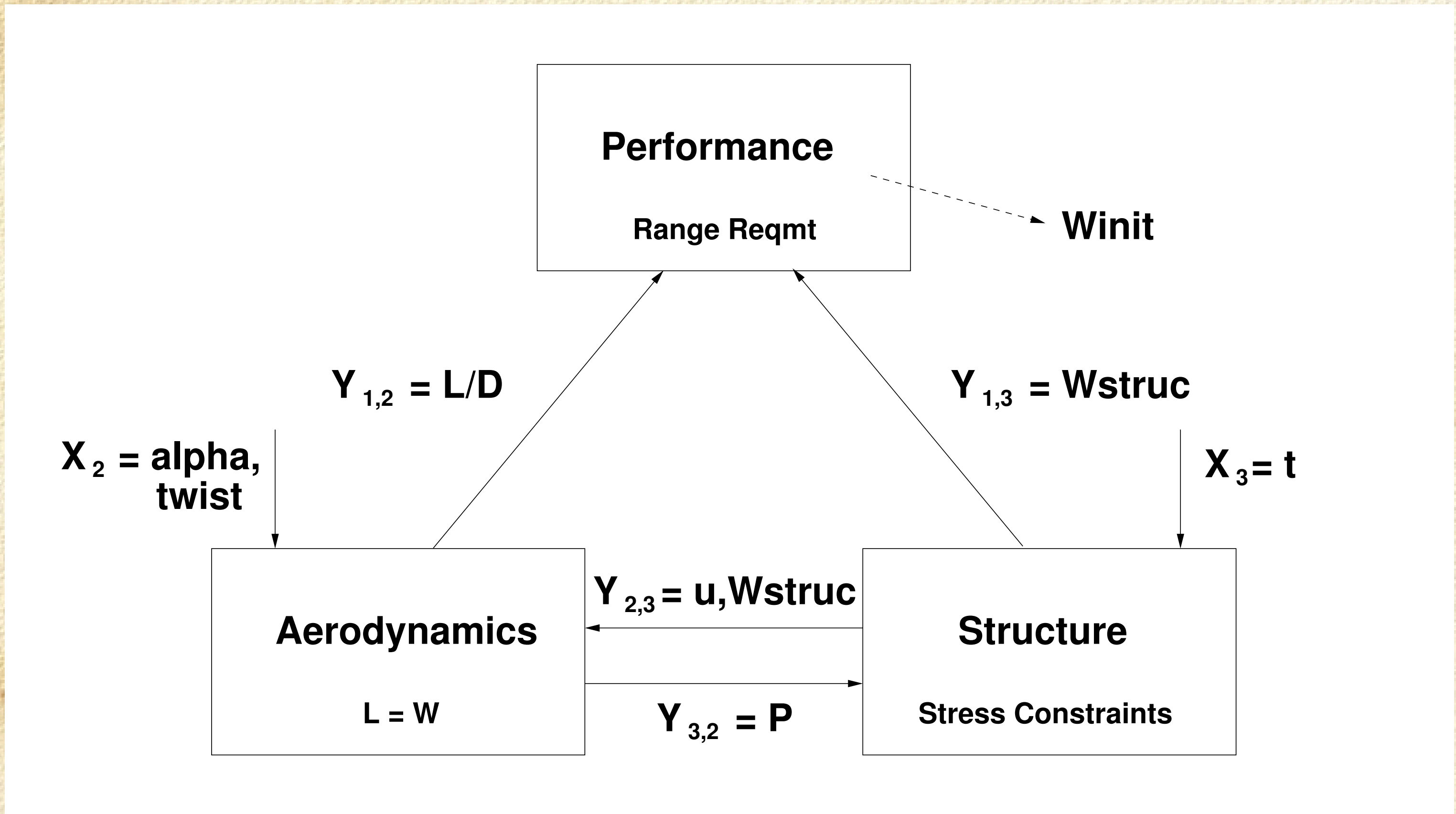
- Three discipline system:
  - Aerodynamics discipline consisting of a 1-D vortex panel code.
  - Structures discipline with a single wing spar with a circular cross section modeled by tubular beam finite elements.
  - Mission performance discipline that computes the takeoff weight to satisfy a range requirement via the Breguet range equation.

BLISS - Low-Fidelity (cont.)

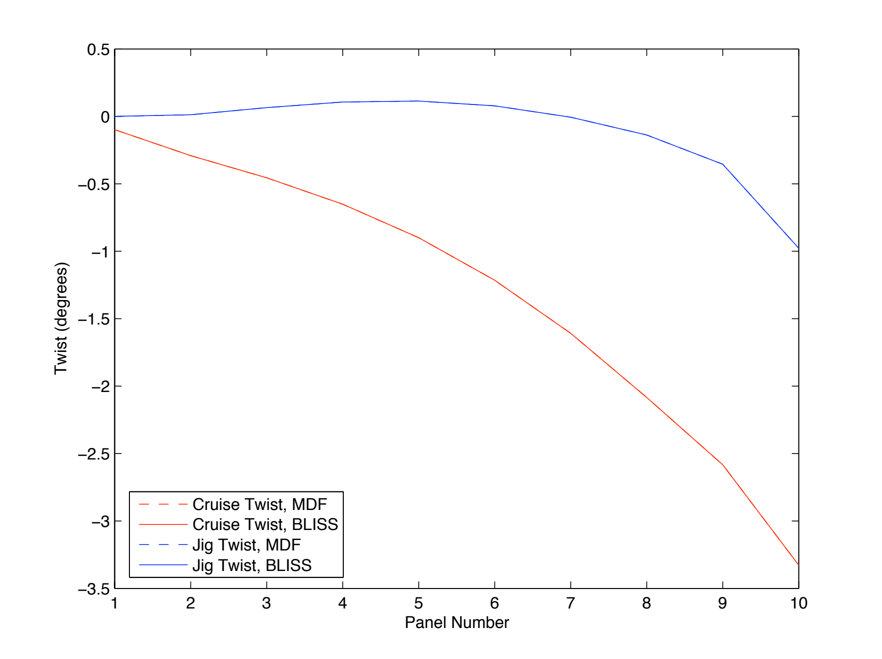


From Martins, 2002.

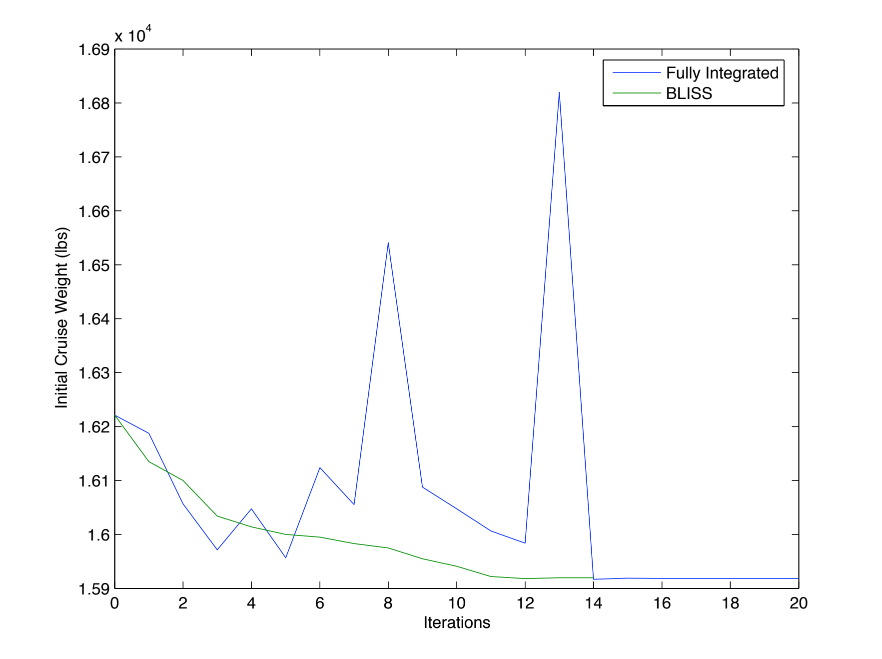
BLISS - Low-Fidelity Interactions



BLISS - Low-Fidelity Results



Jig and Cruise Twists



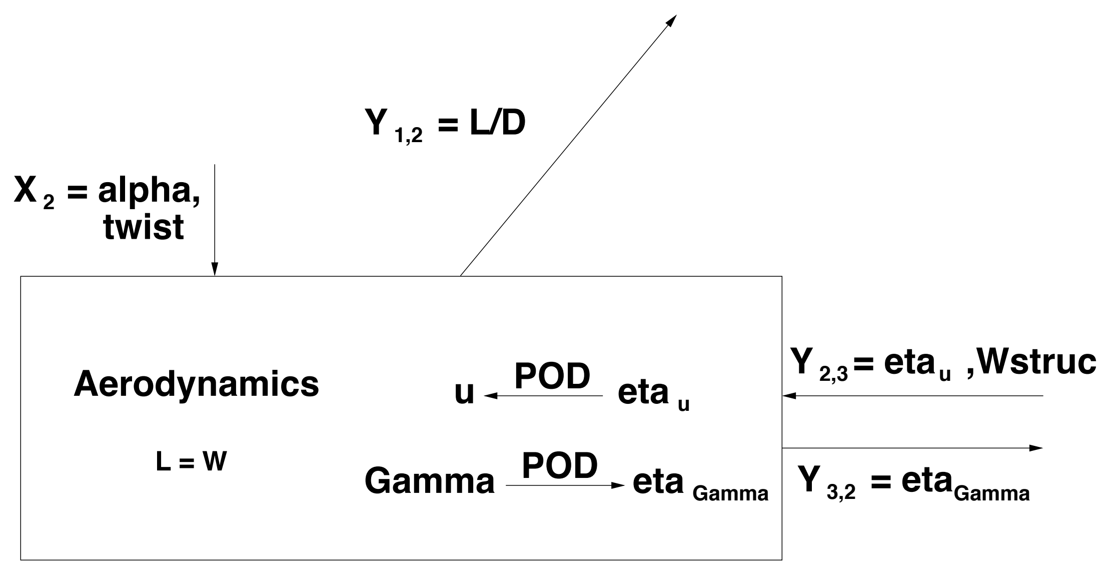
Convergence

BLISS - Low-Fidelity Results (cont.)

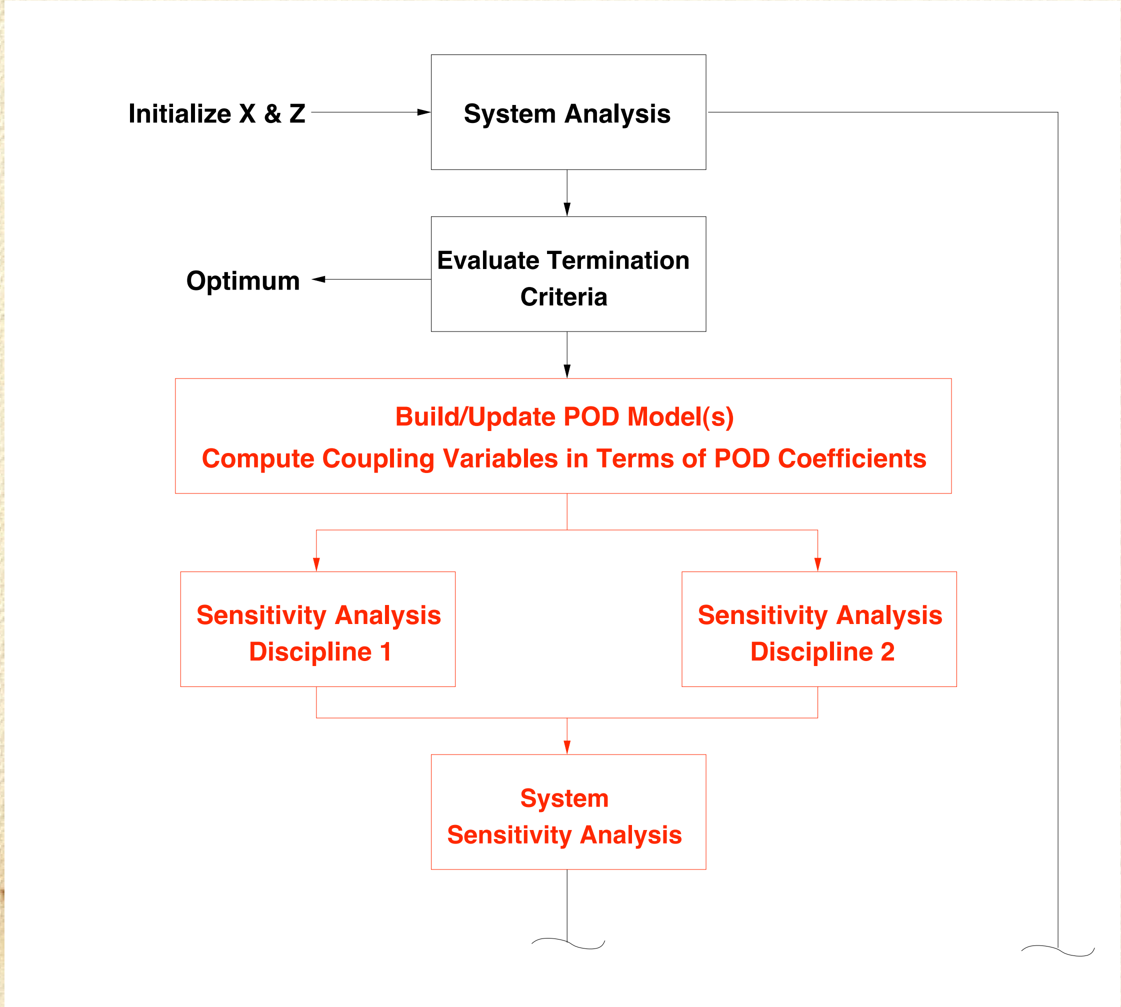
- Values of the design variables are indistinguishable and the objective function matched to within half a pound.
- Computational time was over eight times greater for BLISS compared to Monolithic.
- BLISS (like other decomposition algorithms) suffers from a computational cost with a strong dependence on the number of interaction variables.

BLISS/POD

- Replace high dimensionality quantities like loads and displacements with POD representation for purposes of computing gradients of interactions.



BLISS/POD Flowchart



BLISS/POD (cont.)

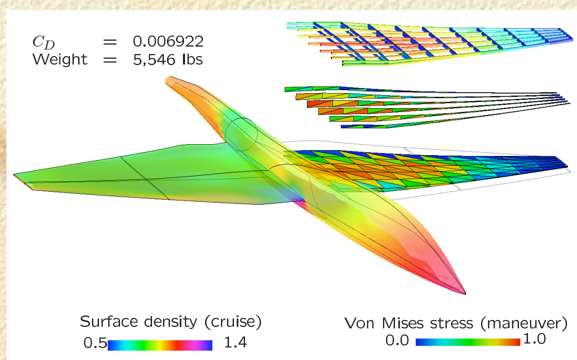
- Sensitivity analysis is now in terms of POD modal coefficients.
- Discipline analyses remain unchanged.
- Can directly compute error in function representation.
- Reduced order model can be updated or refined as necessary.

BLISS/POD Model Update

- Need an initial model
  - Extra multidisciplinary analyses and/or single discipline analyses.
  - Design of (Computer) Experiments
- For subsequent iterations can update as necessary based on analyses from the previous iteration, etc.

High Fidelity Test Case

ASSET Research



Martins (2002)

Performance		
Cruise Mach number	1.5	
Range	5,300	nm
Take-off gross weight (TOGW)	100,000	lbs
Zero-fuel weight (ZFW)	47,500	lbs
Cruise altitude	51,000	ft
Cruise lift coefficient	0.1	
Cruise drag coefficient	0.0116	
Cruise TSFC	0.86	

Wing Geometry		
Reference Area	1750	ft^2
Aspect ratio	3.0	
Taper ratio	0.218	

Objective Function

- From the Breguet range equation we develop an objective to maximize linearized range

$$I = \alpha C_D + \beta W$$

where $\alpha/\beta = 3.04 \times 10^6$ from the performance specifications and C_D is computed at a cruise $C_L = 0.1$ and the structure is sized for a $C_L = 0.2$ maneuver.

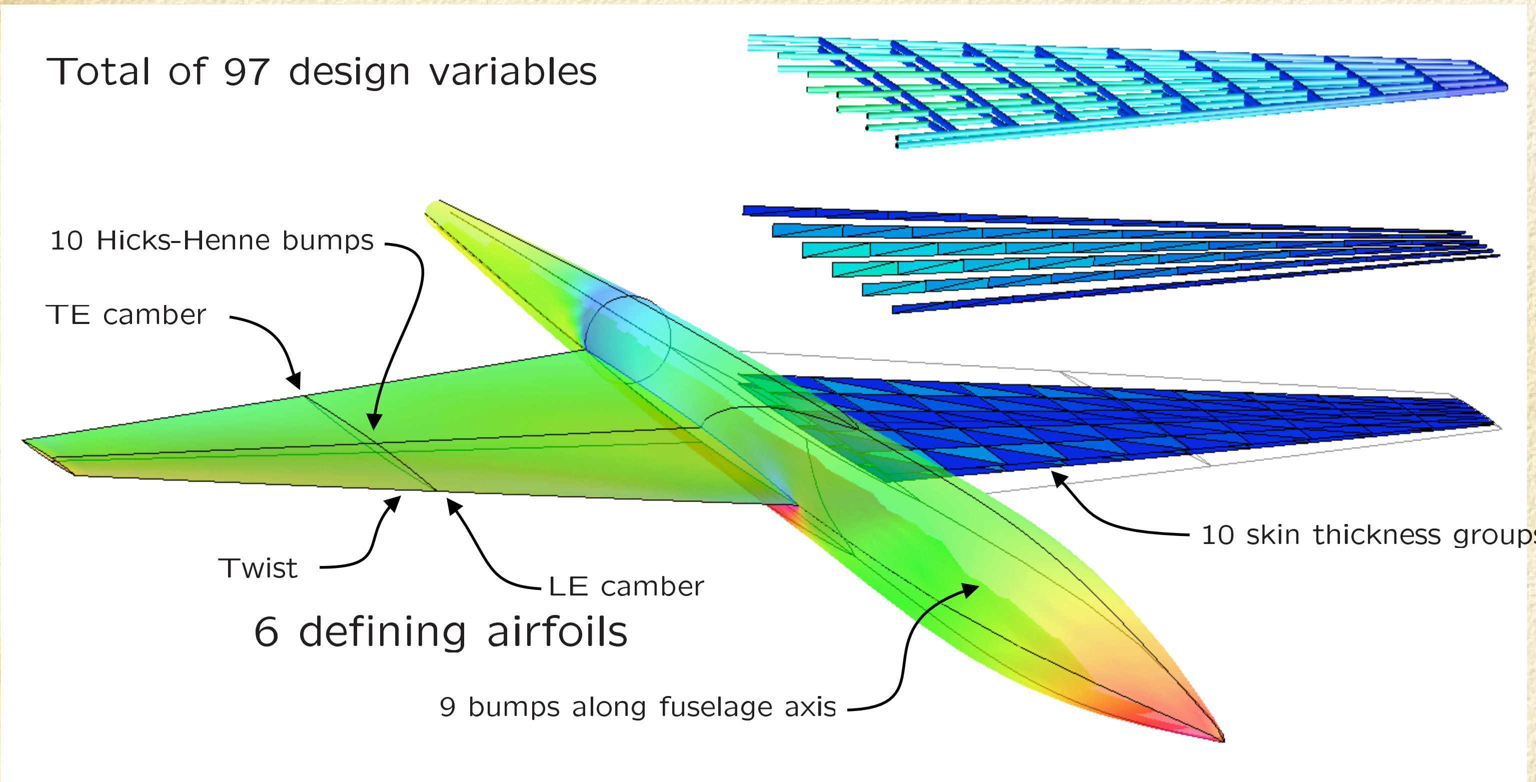
Constraints

- Target cruise and maneuver lift coefficients.
- Number of stress constraints is equal to the number of elements in the structural model.
- Kreisselmeier-Steinhauser (KS) function is used to 'lump' the constraints

$$KS(g_m) = -\frac{1}{\rho} ln\left(\sum_m e^{-\rho g_m}\right)$$

  where g_m is the structural constraint which must be non-negative.
- Minimum gauge skin thickness.

Design Variables



From Martins, 2002.

Analysis and Design Software

- SNOPT - large-scale, nonlinear, constrained optimization
- SUmb - massively parallel flow solver for external and internal flow
- FEAP - general purpose finite-element code for complex structures
- Aerosurf - custom implementation of CAD-type capabilities for aircraft configurations

Analysis and Design Software (cont.)

- WARP - Volumetric mesh deformation
- Load and displacement transfer

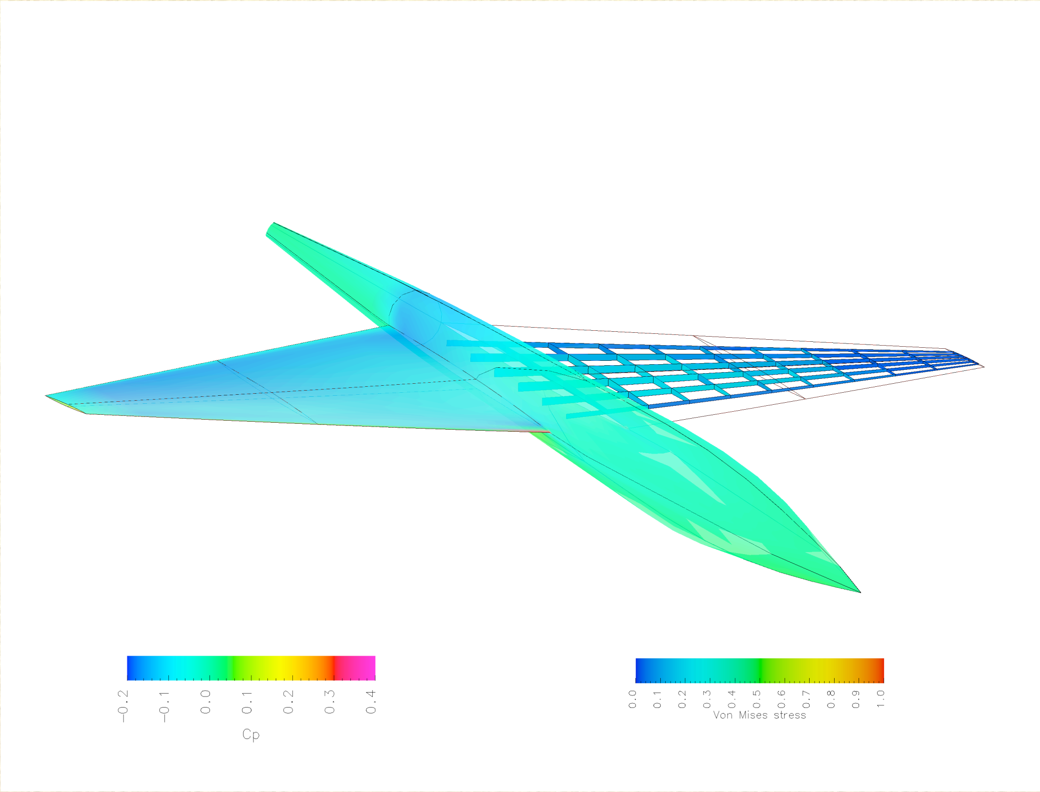
Software Integration

- Using Python, an interpreted object-oriented language with similarities to Java, Matlab, etc.
- Momentum in the scientific computing community - *Numeric*, *f2py*, *pyMPI*, etc.
- Creating a modular, flexible, and easy to use environment - *pyMDO* - for our analysis and design tools.

BLISS/POD Results

- $193 \times 33 \times 49$ cell CFD mesh.
- 132 node, 330 element wing structural model.
- Initial POD model of 15 modes based on the two multidisciplinary analyses, an intermediary aerodynamic analysis, plus four perturbations of the structural displacements for each load case.
- Updated POD model with the new system analyses after each system level iteration, retaining 15 highest modes.

Baseline Configuration



Cruise pressure coefficient on the left and maneuver stresses on the right.

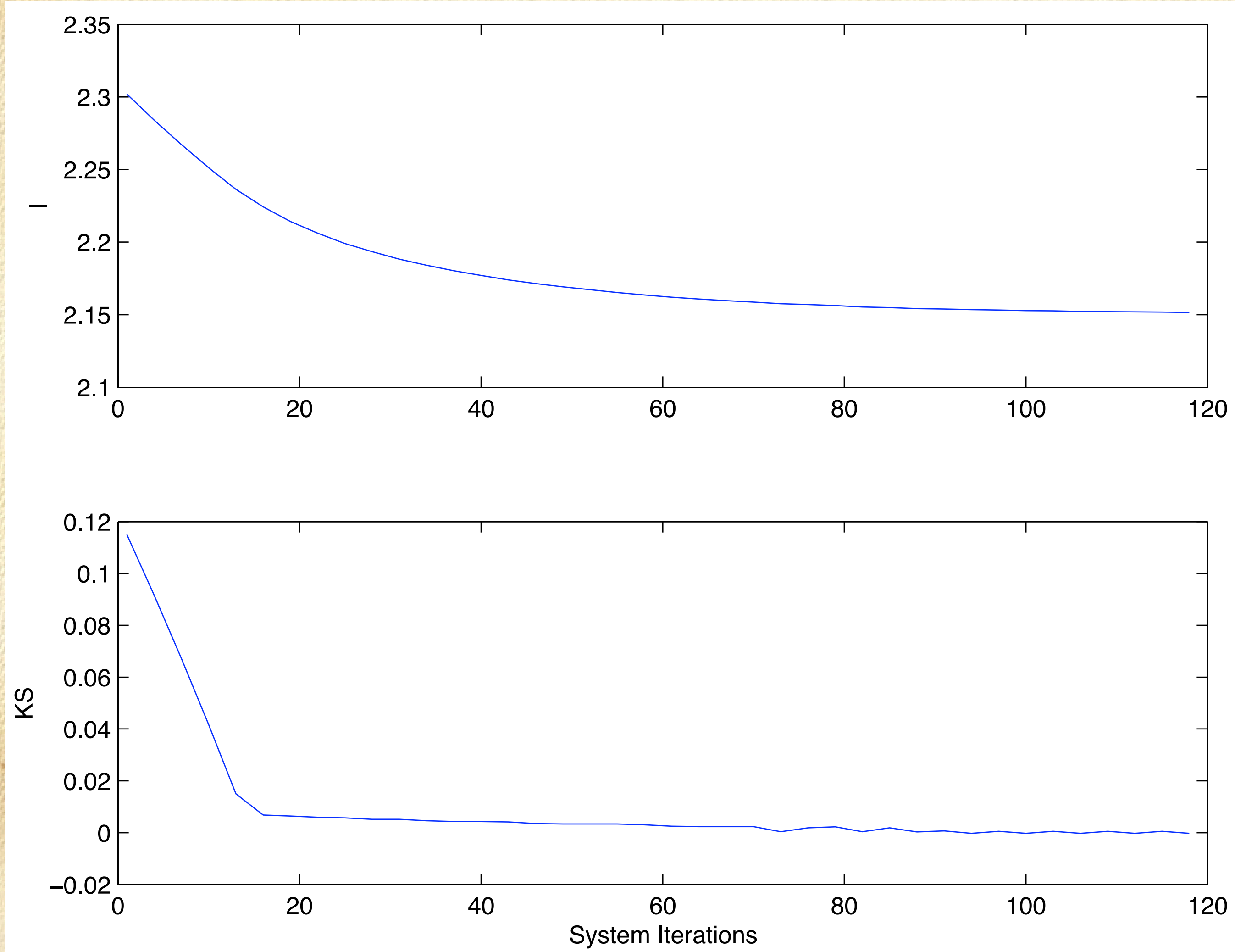
Computational Cost

Per system level iteration and per load case.

	Aerostructural Analyses	Aerodynamic Analyses	Structural Analyses
System Analysis	1		
POD Model		0 to 2	3 to 5 per aero.
Sensitivity Analysis			
A		$0.1M_{disp}$	$0.1M_{loads}$
$\frac{\partial Y}{\partial x}$		M_{loads} *	M_{disp}
Discipline Optimization		$O(1)$	$O(n_{x_{struc}})$

* Projected cost based on the use of an adjoint formulation.

Convergence History

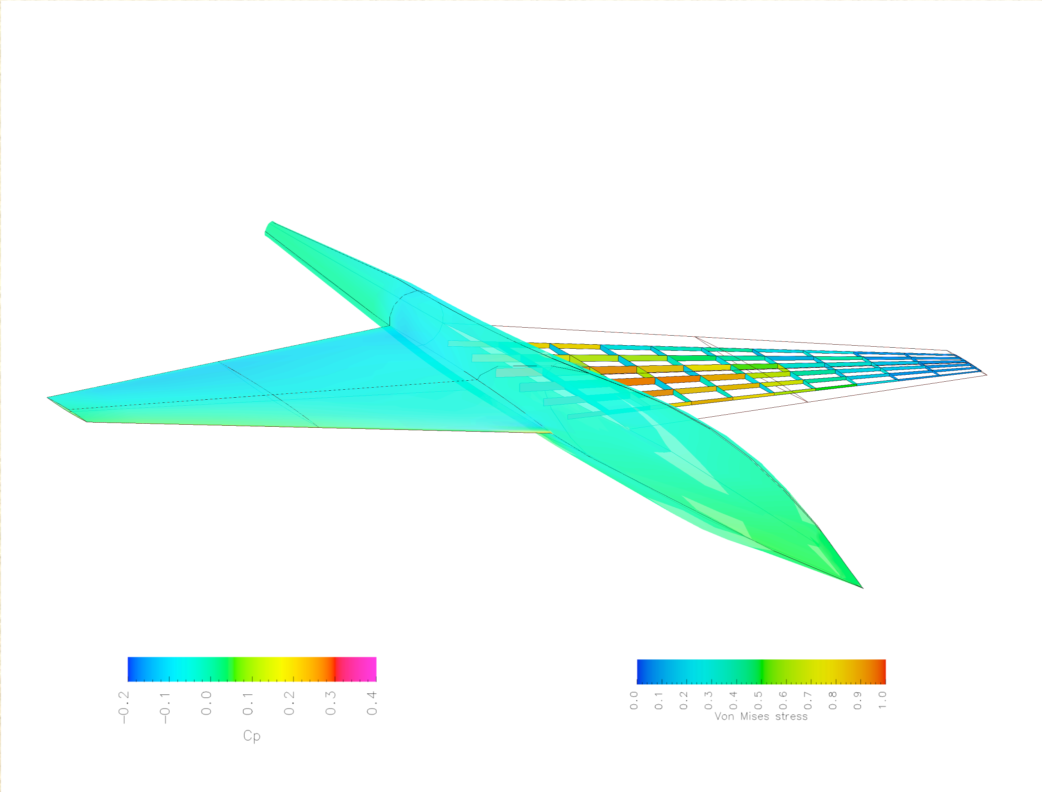


Optimization Results

	C_D (counts)	KS	$\sigma_{max}/\sigma_{yield}$	ZFW (lbs)	Range (nm)
Coupled Adjoint*					
Baseline	73.95	1.15×10^{-1}	0.87	47,500	6,420
Optimization	69.22	-2.68×10^{-4}	0.98	43,761	7,361
BLISS/POD					
Baseline	74.80	2.44×10^{-1}	0.85	47,550	6,338
Optimization	72.04	6.17×10^{-5}	0.99	43,965	7,275

* From Martins, 2002.

Optimized Configuration



Cruise pressure coefficient on the left and maneuver stresses on the right.

Comparison with Monolithic

- Solutions differ by approximately 2 drag counts, 200 lbs structural weight, and 100 nm in range.
- Computational cost per system level iteration approximately 3.4 times higher for BLISS/POD compared to monolithic with coupled adjoint.
- More design iterations required, for an overall computational cost that was 7.9 times higher.
- *Computationally feasible* though, while <u>BLISS alone is not</u>.

Contributions

- Improved MDO decomposition algorithm was developed by incorporating POD as a reduced bandwidth interface between disciplines.

- More loosely coupled decomposition algorithm with more discipline autonomy and opportunities for parallelism can be used.

- Important for scalability of disciplines modeled at a high-level of fidelity.

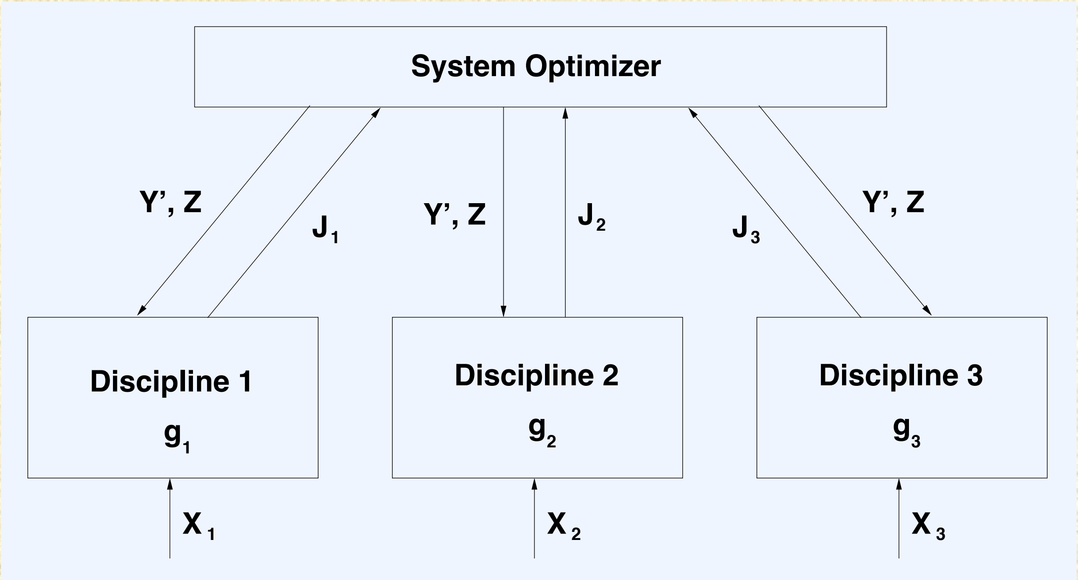
Future Research Areas

- Defining alternate optimality criteria for the POD basis when we are interested in the accuracy of its response to varying input parameters, design variables, etc.
- Techniques for constructing, monitoring, and updating reduced order models for MDO.
- Use the solve of the approximate Global Sensitivity Equations to compute approximate coupled gradients and perform a more conventional integrated optimization

Acknowledgments

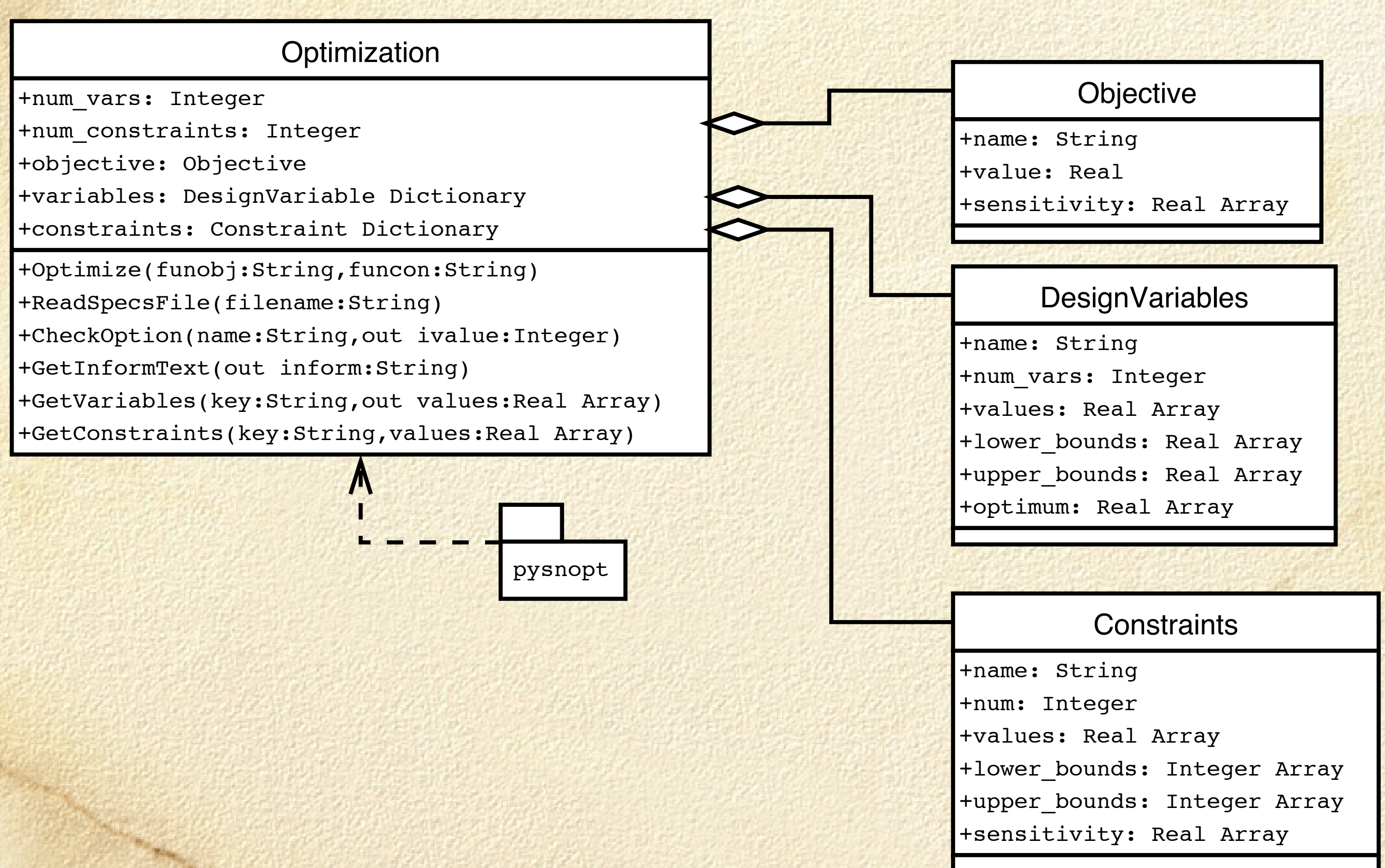
- Juan J. Alonso
- Ilan Kroo, Brian Cantwell, Sanjiva Lele, and Michael Saunders
- Funding
  - Stanford Graduate Fellowship (SGF) Program
  - NASA
  - CITS / DoE ASC

Collaborative Optimization (CO)



- Optimization problems for each discipline, coupled by system level optimizer.
- May exhibit poor convergence for tightly coupled problems.

pysnopt


Optimization
+num_vars: Integer
+num_constraints: Integer
+objective: Objective
+variables: DesignVariable Dictionary
+constraints: Constraint Dictionary
+Optimize(funobj:String,funcon:String)
+ReadSpecsFile(filename:String)
+CheckOption(name:String,out ivalue:Integer)
+GetInformText(out inform:String)
+GetVariables(key:String,out values:Real Array)
+GetConstraints(key:String,values:Real Array)
pysnopt
Objective
+name: String
+value: Real
+sensitivity: Real Array
DesignVariables
+name: String
+num_vars: Integer
+values: Real Array
+lower_bounds: Real Array
+upper_bounds: Real Array
+optimum: Real Array
Constraints
+name: String
+num: Integer
+values: Real Array
+lower_bounds: Integer Array
+upper_bounds: Integer Array
+sensitivity: Real Array

pysnopt Usage

Minimize: $A = x_1 x_2$

With respect to: $x_1,\, x_2$

Subject to: $x_1^2 + x_2^2 = 1$

```
# load the Optimization class
import optimization
problem = optimization.Optimization('Max Area','coords':  2,'circle':  1)
# Set the starting guess for the optimization (defaults to zero)
problem.variables['coords'].value[:]  = 0.01
# Set the lower bounds for the design variables
# (defaults are -inf and +inf respectively)
problem.variables['coords'].lower[:]  = 0.
# Set the lower and upper bounds for the constraints
problem.constraints['circle'].lower[0] = 1.
problem.constraints['circle'].upper[0] = 1.
# Define the objective and constraint functions as well

def funobj(mode, x, f_obj, g_obj):
    f_obj[0] = x[0]*x[1]
    g_obj[0] = x[1]
    g_obj[1] = x[0]
    return
def funcon(mode, njac, x, f_con, g_con):
    f_con[0] = x[0]**2 + x[1]**2
    g_con[0,0] = 2.*x[0]
    g_con[0,1] = 2.*x[1]
    return

# Run the optimizer
problem.Optimize(funobj, funcon)
```

Additional Python Modules

- pysumb
- pyfeap
- pydx
- Aerostructure class

```
def RunIterations(self, num_iterations):
    self.flow.RunIterations(10)
    cfd_loads = self.flow.GetSurfaceLoads('Wing')
    csm_loads = self.TransferLoads(cfd_loads)
    self.structure.SetLoads('test', csm_loads)
    self.structure.CalculateDisplacements('test')
    csm_dispts = self.structure.load_cases['test'].surf_displacements
    cfd_dispts = self.TransferDisplacements(csm_dispts)
    # Warp the mesh and pass the new coordinates to SUmb
    ijk_blnum = self.flow.GetMesh().GetSurfaceIndices('Wing')
    cfd_xyz = self.xyz_wing_orig + cfd_dispts
    self.meshwarping.Warp(cfd_xyz, ijk_blnum[3,:], ijk_blnum[0:3,:])
    for n in range(1,self.flow.GetMesh().GetNumberBlocks()+1):
        # Get the new coordinates from warp
        [blocknum,ijkrange,xyz_new] = self.meshwarping.GetCoordinates(n)
        blocknum = self.comm_world.bcast(blocknum)
        ijkrange = self.comm_world.bcast(ijkrange)
        xyz_new = self.comm_world.bcast(xyz_new)
        # Give them to SUmb
        self.flow.GetMesh().SetCoordinates(blocknum,ijkrange,xyz_new)
    return
```