# Metacomputing Analysis and Design in SORCER

## 17th ISPE International Conference on Concurrent Engineering
## Cracow, Poland
## 6-10 Sept, 2010

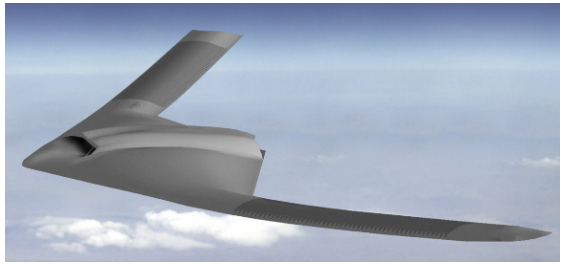


*Mike Sobolewski PJIIT*
*Ray Kolonay*
*US Air Force Research Lab*

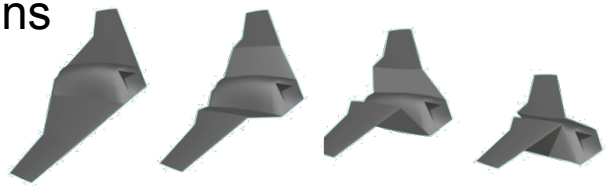
# AFRL Air Vehicles Directorate
# Revolutionary Aerospace Vehicles

*AFRL Air Vehicles Directorate Multidisciplinary Sciences & Technology Center*

Extreme Environments

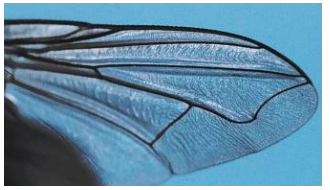


Large Deformations



Time Varying Shape

MAVs



***Higher, Farther, Faster to***
***Lower, Closer, Slower***

# How to Capture the Physics Driving the Design Early in the Process

*AFRL Air Vehicles Directorate Multidisciplinary Sciences & Technology Center*

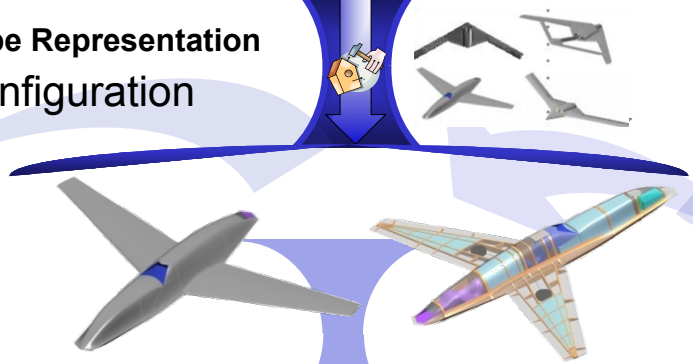
10000s configurations

Conceptual Design Studies

Technology Suite

**Prototype Representation**

10s configuration



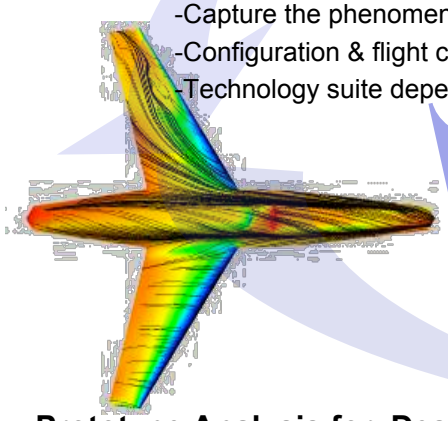
Dialable Fidelity

-Capture the phenomena driving the design

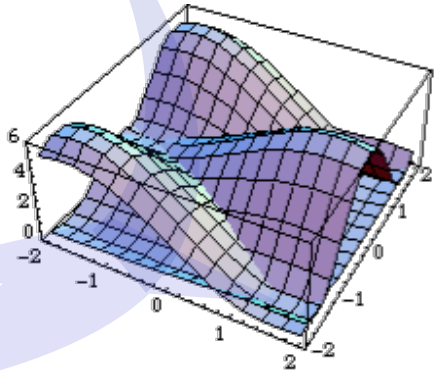
-Configuration & flight condition dependent

-Technology suite dependent

Prototype Experimental Validation



**Prototype Analysis for Design**

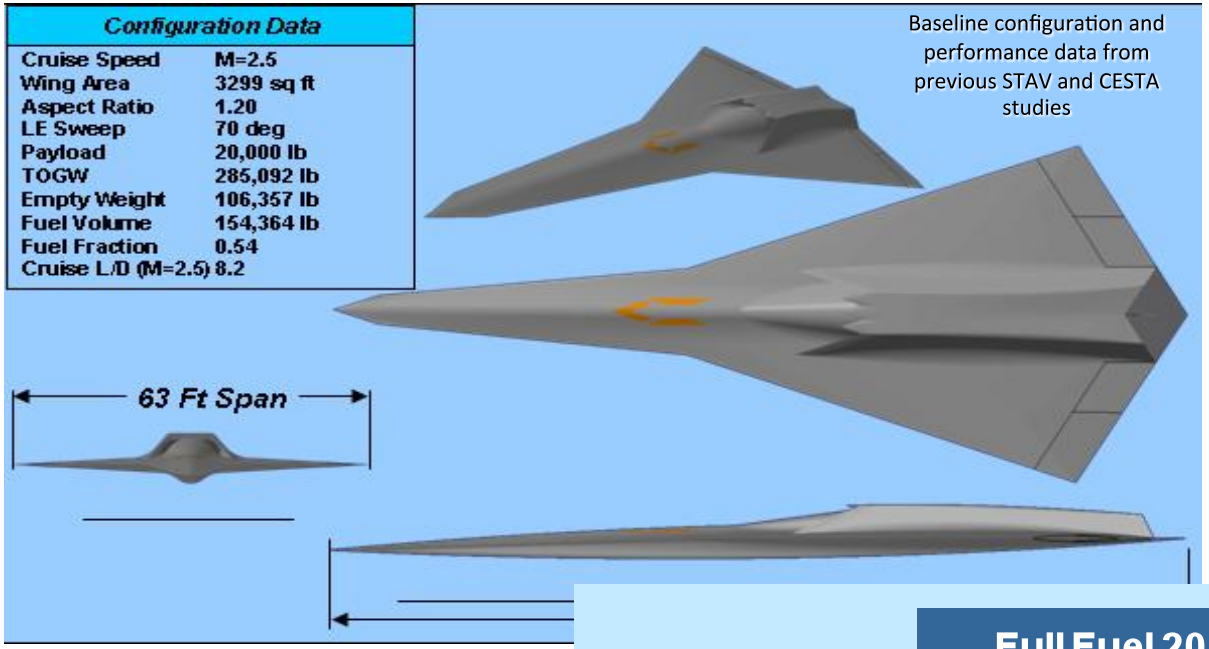


**Prototype Design Space Exploration**

***Get more (and Better) Information … and get it Earlier***

# Typical Target Application
# LRS Configuration

AFRL Air Vehicles Directorate Multidisciplinary Sciences & Technology Center




Requirements:

Range = 4000 nmi

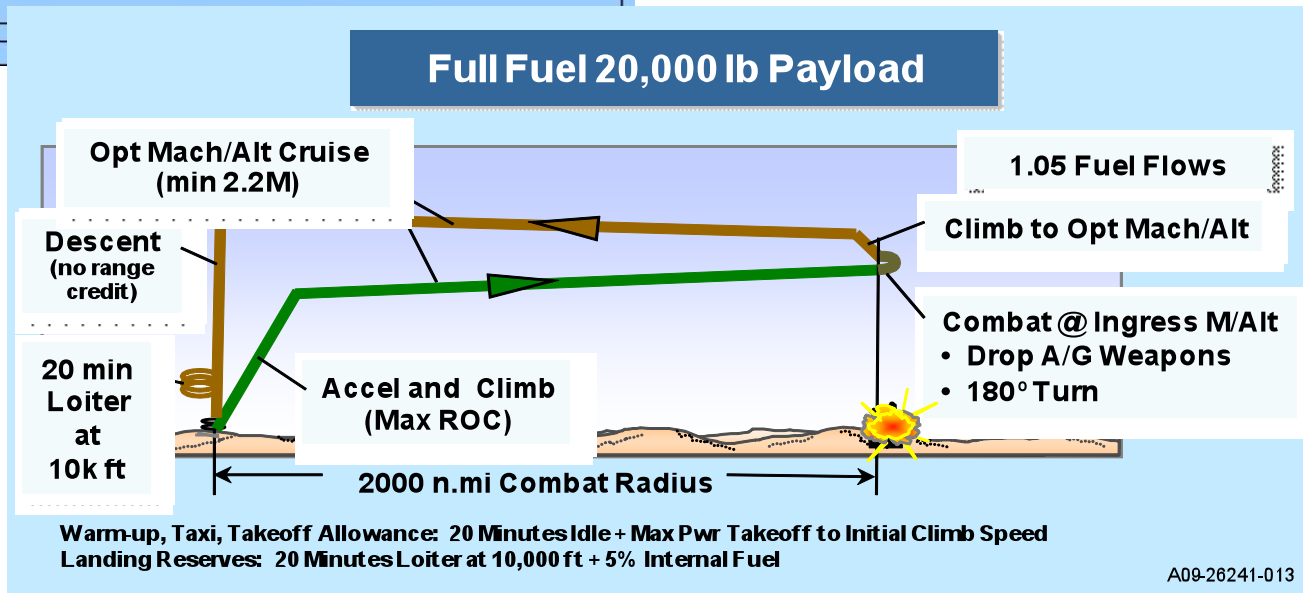
Payload = 5-10 % weight fraction

Maneuver Loads at Cruise – 2.5 g

Cruise speed: Mach 2.0

Cruise L/D: 8.5-9

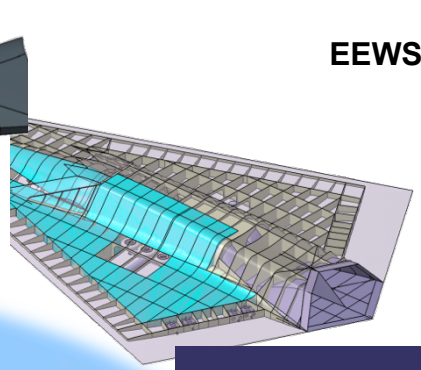
Level 1 Flying Qualities



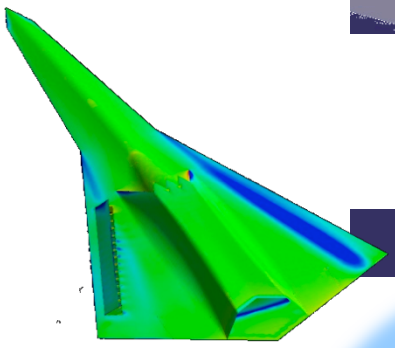
# LRS "Design Drivers"

*Multidisciplinary Sciences & Technology Center*

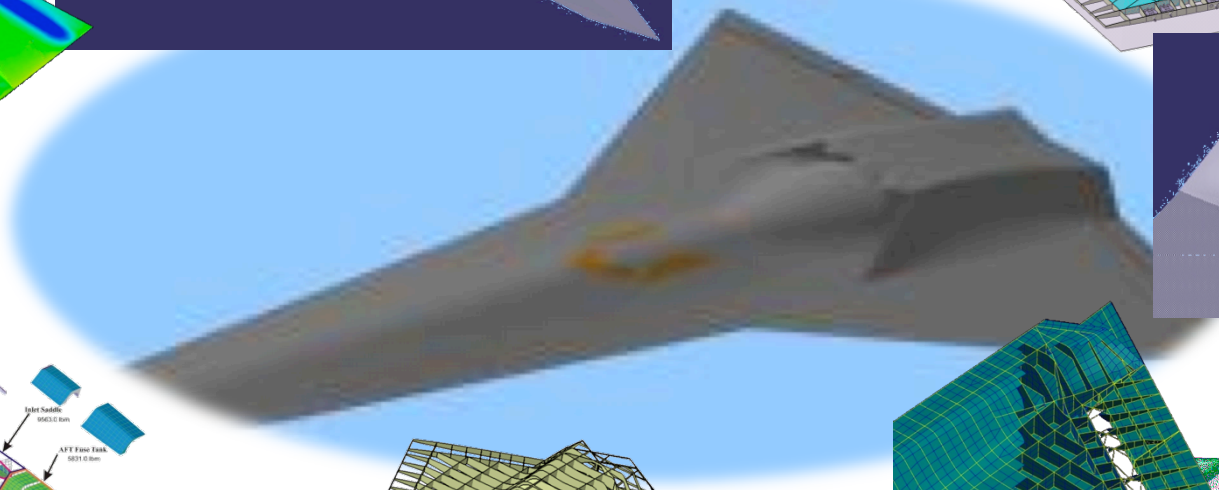
**Subsystems & fuel management**

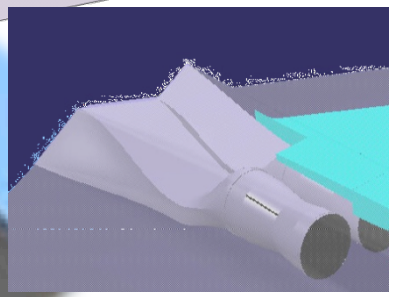


**EEWS**

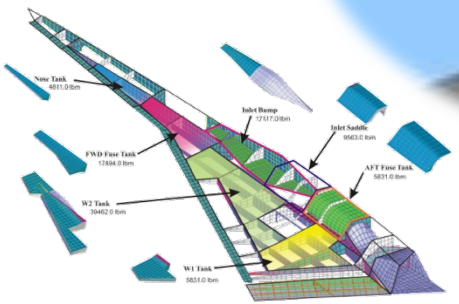


**Aerodynamics**

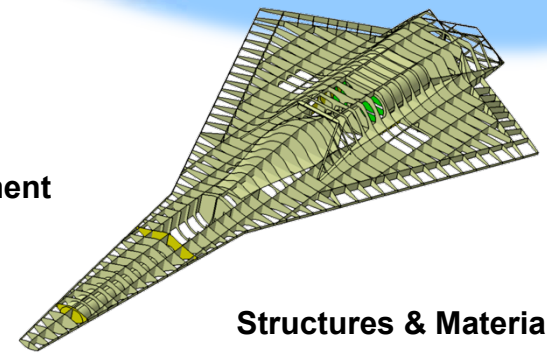




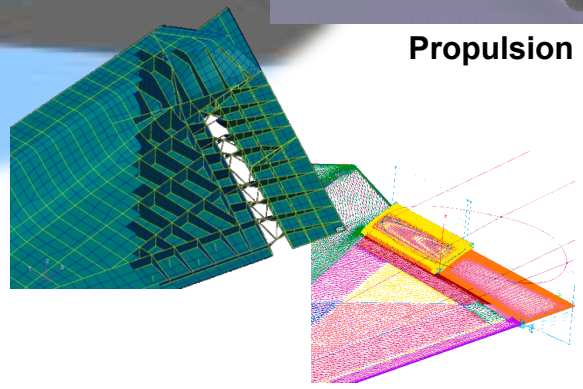
**Propulsion**



**System Thermal Management**



**Structures & Materials**



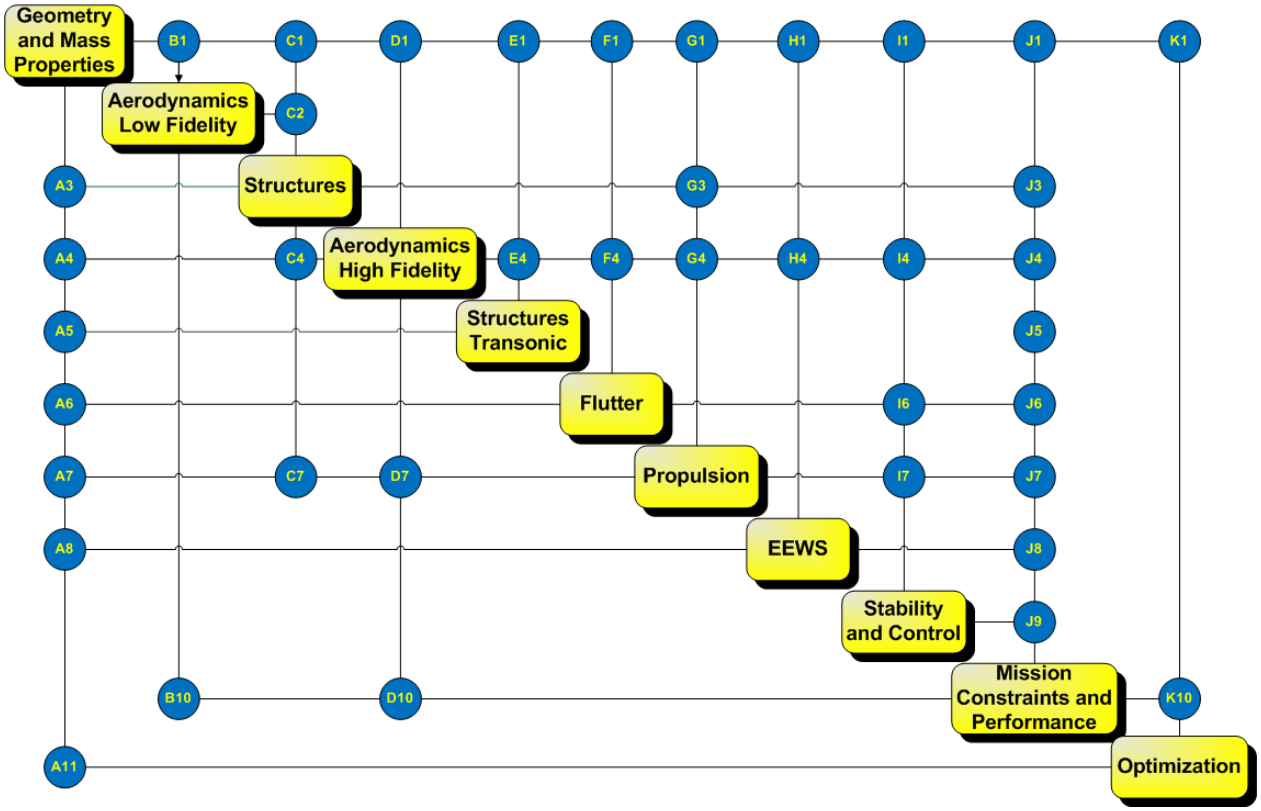
**Stability & Control**

# System Level MDA/MDO

AFRL Air Vehicles Directorate Multidisciplinary Sciences & Technology Center

## "Best in Class Approach (BCA)"

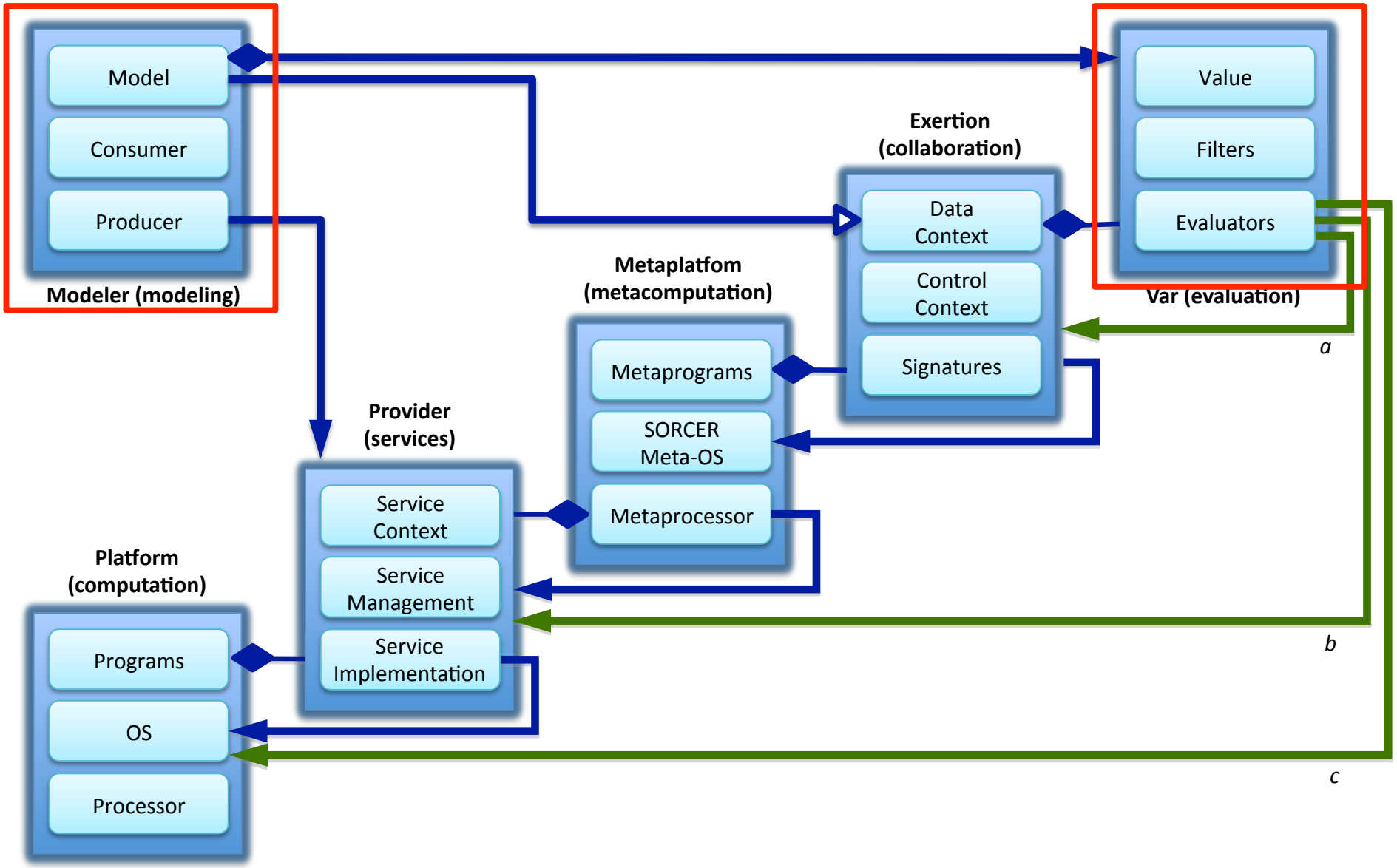
**Components are one or more engineering computational applications that need to be "glued" together to execute a process.** (BTW, components may be on a distributed heterogeneous network)




***Multi-fidelity, Multidisciplinary Computations Require Real Time Seamless Access to Applications, Data & Compute Resources by ALL***

# Model Based Computing

SORCER




# SORCER Models



SORCER Models support

- Analysis – (ResponsModel & ParametricModel)
  - multidisciplinary response analysis
  - multi-fidelity response analysis
  - multi-fidelity response sensitivities

- Design Space Exploration (OptimizationModel)
  - Optimization
  - Multidisciplinary Optimization


# SORCER Model

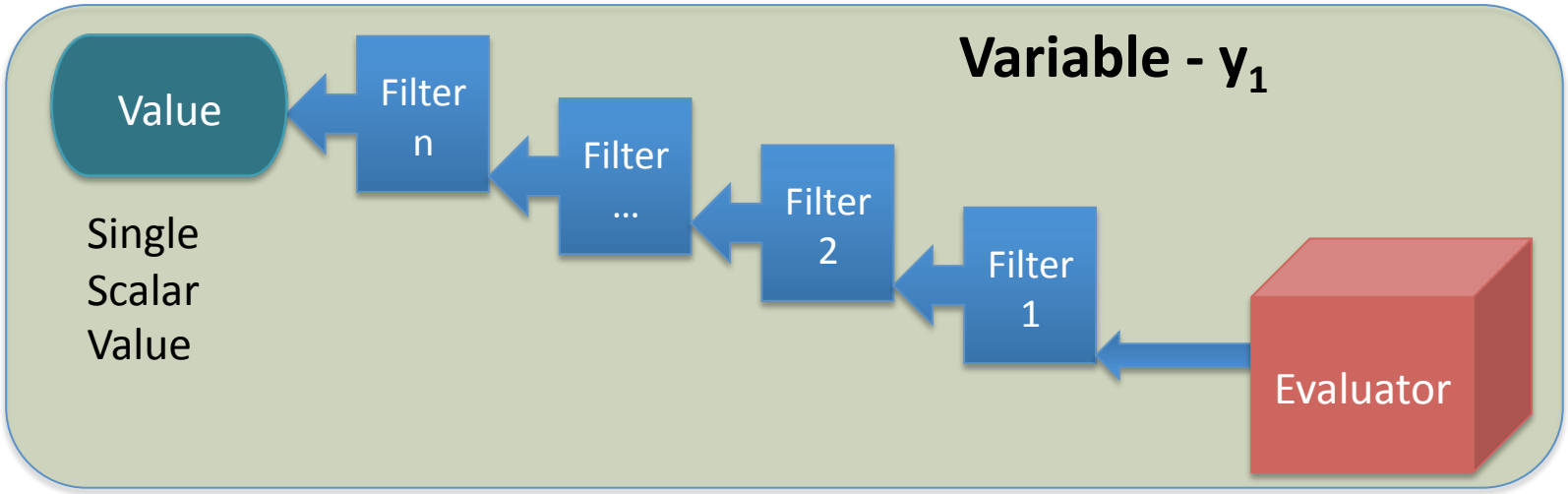


- ***Models –*** consists of a collection of ***Variables***.
  - Current available models – ResponseModel(including sensitivities) , ParametricModel,, OptimizationModel
- ***Variables -*** can be dependent on other *Variables* enabling distributed **functional programming**. Variables can have multiple evaluators enabling **multi-fidelity calculations** for a specific variable's value.
  - Current Variable Types - DESIGN, RESPONSE, DERIVATIVE, GRADIENT, RANDOM, BOUNDED, LINKED, PARAMTER
- ***Evaluators –*** are used to determine the value of variables and their partial derivatives(chain rule works) with respect to their dependencies (Variables).
  - Current Evaluator Types – ModelEvaluator, ExertionEvaluator, ExpressionEvaluator, GroovyEvaluator, JepEvaluator, MethodEvaluator
- ***Filters -*** are used to map the results of *Evaluators* to *Variable* Values. Think unix shell piping. Filters can be concatenated *n* times.
  - Current Filter Types – BasicFileFilter, ContextFilter, FileFilter, GrepFilter, ListFilter, MapFilter, ObjetFilter, PatternFilter, TextFilter

***All are Objects***

# Variable - Value, Filters, Evaluators

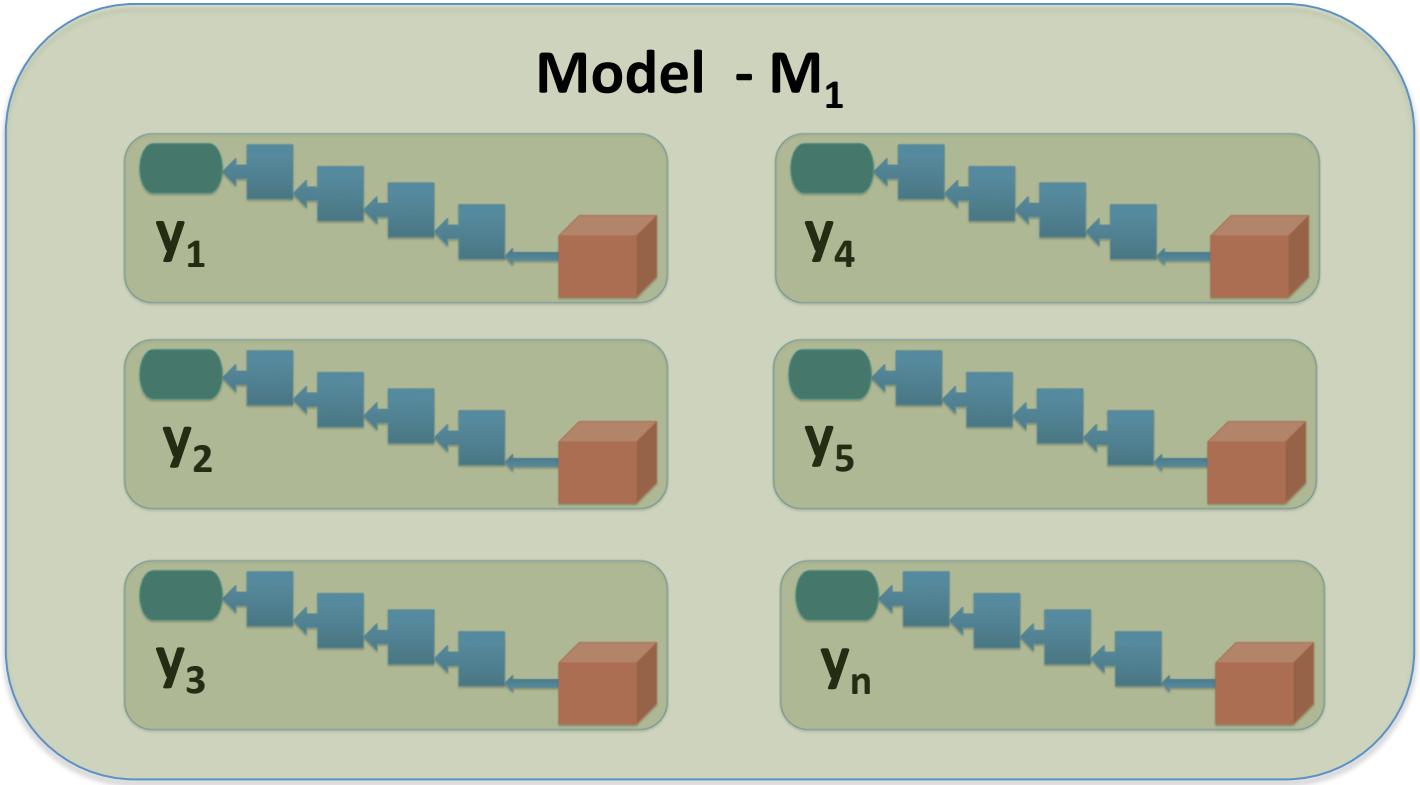





Computational Entity. Creating Large Amounts of Data

# Model is a Collection of Variables





# Rosen-Suzuki Functions



Independent Varibles - $\{x_1, x_2, x_3, x_4\}$

Functions

$$f(x_1,x_2,x_3,x_4) = x_1^2 - 5x_1 + x_2^2 - 5x_2 + 2x_3^2 - 21x_3 + x_4^2 + 7x_4 + 50$$

$$g_1(x_1,x_2,x_3,x_4) = x_1^2 + x_1 + x_2^2 - x_2 + x_3^2 + x_3 + x_4^2 - x_4 - 8.0$$

$$g_2(x_1,x_2,x_3,x_4) = x_1^2 - x_1 + 2x_2^2 + x_3^2 + 2x_4^2 - x_4 - 10.0$$

$$g_3(x_1,x_2,x_3,x_4) = 2x_1^2 + 2x_1 + x_2^2 - x_2 + x_3^2 - x_4 - 5.0$$

## *Model Definition*

```
ResponseModel rm = ResponseModel("Rosen-Suzuki Model",
                                 designVars("x", 4),   // creates 4 real design variables x1-x4
                                 responseVars("f"),   // creates 1 response variable f
                                 responseVars("g", 3)   //creates 3 response variables g1-g3
                   );
```

# Rosen-Suzuki Functions



## Model Configuration

```
// configure response vars
Evaluator fe = evaluator("fe", "x1^2-5.0*x1+x2^2-5.0*x2+2.0*x3^2-21.0*x3+x4^2+7.0*x4+50.0");
rm.setResponseEvaluator("f", fe);
fe.addArgs( om.getDesignVars("x1", "x2", "x3", "x4"));

Evaluator g1e = evaluator("g1e", "x1^2+x1+x2^2-x2+x3^2+x3+x4^2-x4-8.0");
rm.setResponseEvaluator("g1", g1e);
g1e.addArgs(om.getDesignVars("x1", "x2", "x3", "x4"));

Evaluator g2e = evaluator("g2e", "x1^2-x1+2.0*x2^2+x3^2+2.0*x4^2-x4-10.0");
rm.setResponseEvaluator("g2", g2e);
g2e.addArgs(om.getDesignVars("x1", "x2", "x3", "x4"));

Evaluator g3e = evaluator("g3e", "2.0*x1^2+2.0*x1+x2^2-x2+x3^2-x4-5.0");
rm.setResponseEvaluator("g3", g3e);
g3e.addArgs(om.getDesignVars("x1", "x2", "x3", "x4"));
```

# Rosen-Suzuki Functions



## Model Use

```
// you can set the value of the independent variables
rm.setDesignVarValues(set("x1", 2.0), set("x2", 3.0), set("x3", 4.0), set("x4", 5.0));

// get the values of the responses for the new x values
rm.getRepsonses();
Data Table [x1,x2,x3,x4,  f, g1,g2,g3]
           [2.0, 3.0, 4.0, 5.0,  46.,  44.0, 71.0, 24.0]

// additional examples of using the api
Var x4 = rm.getDesignVar("x4");
logger.info("\n\nx4 value: " + x4.getValue());
x4 value: 5.0

// examine the responseVar f
Var f = rm.getResponseVar("f");
ExpressionEvaluator ee = (ExpressionEvaluator) f.getEvaluator();
logger.info("\n\nexpression:\n" + ee.getName() + "/" + ee.getExpression());
fe/x1^2-5.0*x1+x2^2-5.0*x2+2.0*x3^2-21.0*x3+x4^2+7.0*x4+50.0
logger.info("\n\nresponse:\n" + ee.getName() + "/" + f.getValue());
fe/46.0
```

# SORCER Variables



SORCER Variables are distinguished by four attributes: ***Type, Kind, ValueType,*** and ***MathType***

SORCER Variable ***Types*** (only one)

- DESIGN (DEFAULT)
- RESPONSE

SORCER Variable ***Kinds*** (one or more combinations)

- LINKED
- PARAMETER
- BOUNDED
- RANDOM
- CONSTRAINT
- OBJECTIVE
- INDEPENDENT

SORCER Variable ***ValueType*** (only one)

- INTEGER
- LONG
- DOUBLE (DEFAULT)
- FLOAT
- STRING
- OBJECT
- UNKNOWN
- UNDEFINED

SORCER Variable ***MathType*** (one or more combinations)

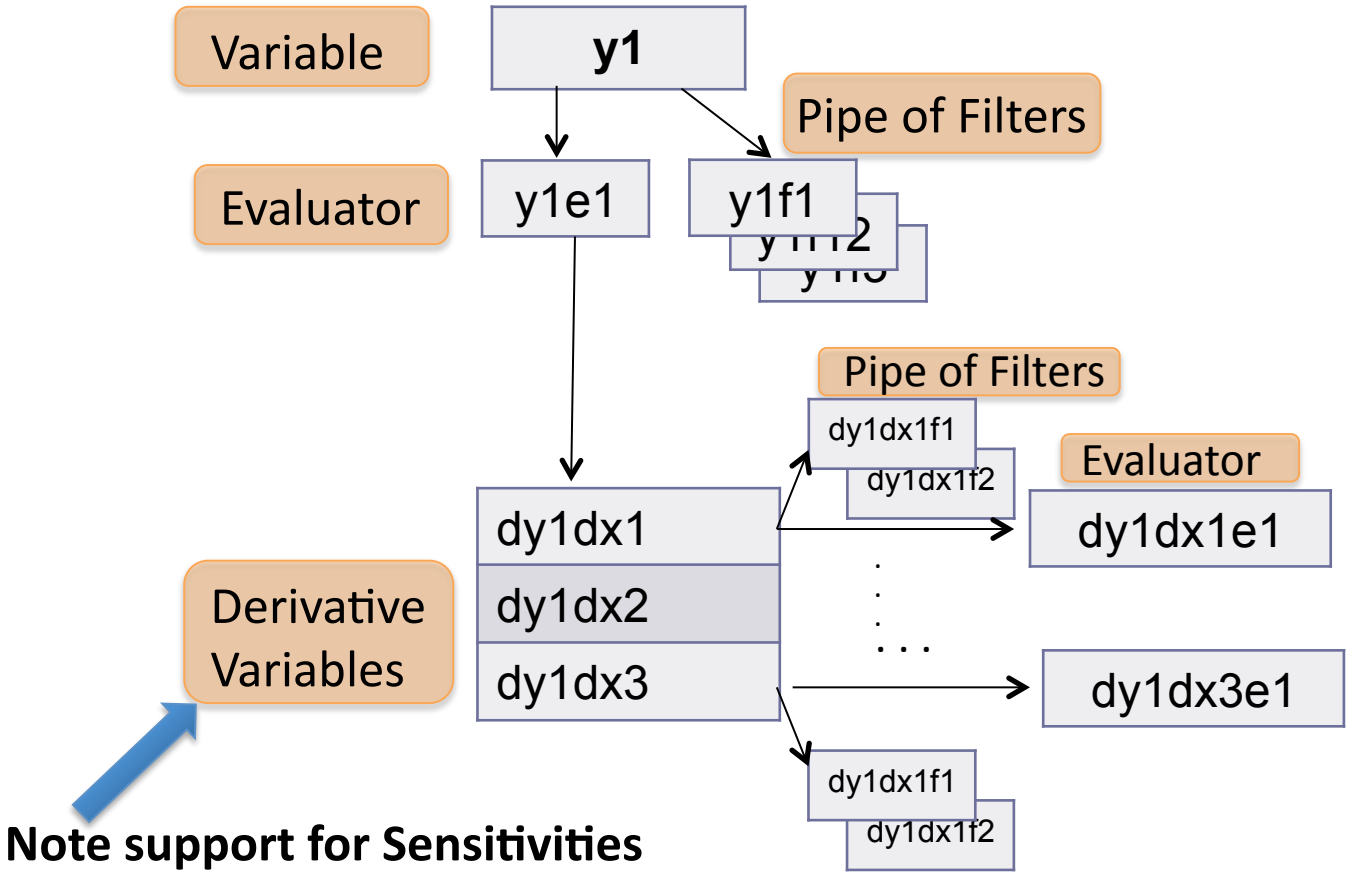
- CONTINUOUS (DEFAULT)
- DISCRETE
- DISCRETE_WITH_ORDER
- DISCRETE_WITH_MATH
- DISCRETE_NO_ORDER
- PROBLEM_PARAMETER
- REAL

Example: skinThickness: Type:DESIGN, Kinds:BOUNDED,RANDOM, ValueType:DOUBLE, MathType: REAL, CONTINUOUS

# Basic Variable Structure

SORCER

$$y_1(x_1, x_2, x_3)$$

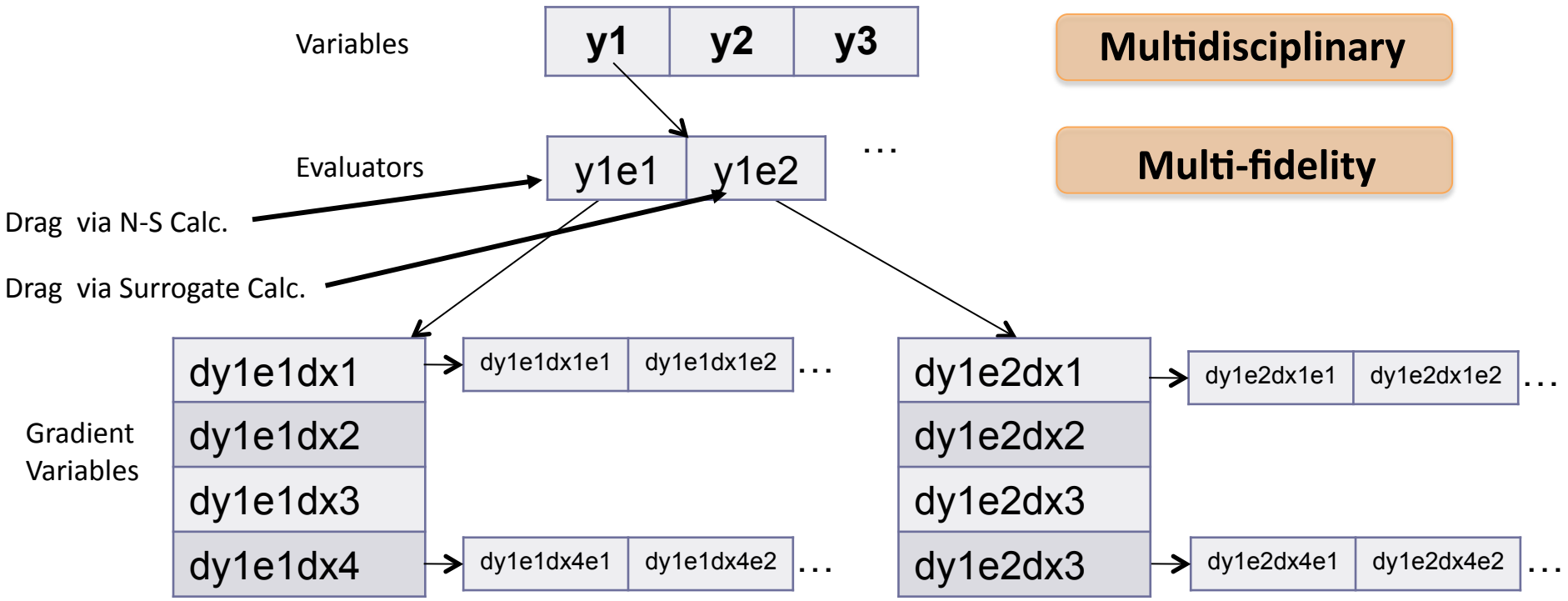


# Advanced Variable Structure

SORCER

**Functions of Functions**

$$y_1(x_1, x_2, x_3), \quad y_2(x_4, x_5, x_6, y_1), \quad y_3(x_6, x_7, x_8, y_2)$$



# Rosen-Suzuki Function Optimization Definition



**Minimize**

$$f = x_1^2 - 5x_1 + x_2^2 - 5x_2 + 2x_3^2 - 21x_3 + x_4^2 + 7x_4 + 50$$

**Subject to:**

$$g_1 = x_1^2 + x_1 + x_2^2 - x_2 + x_3^2 + x_3 + x_4^2 - x_4 - 8.0 \le 0.0$$

$$g_2 = x_1^2 - x_1 + 2x_2^2 + x_3^2 + 2x_4^2 - x_4 - 10.0 \le 0.0$$

$$g_3 = 2x_1^2 + 2x_1 + x_2^2 - x_2 + x_3^2 - x_4 - 5.0 \le 0.0$$

$$-100.0 \le x_1 \le 100.0$$

$$-100.0 \le x_2 \le 100.0$$

$$-100.0 \le x_3 \le 100.0$$

$$-100.0 \le x_4 \le 100.0$$

# Objective Function, Constraints and Sensitivities



## Objective Sensitivities:

$$f = x_1^2 - 5x_1 + x_2^2 - 5x_2 + 2x_3^2 - 21x_3 + x_4^2 + 7x_4 + 50$$

$$\frac{\partial f}{\partial x_1} = 2x_1 - 5, \quad \frac{\partial f}{\partial x_2} = 2x_2 - 5, \quad \frac{\partial f}{\partial x_3} = 4x_3 - 21, \quad \frac{\partial f}{\partial x_4} = 2x_4 + 7$$

## Constraints Sensitivities:

$$g_1 = x_1^2 + x_1 + x_2^2 - x_2 + x_3^2 + x_3 + x_4^2 - x_4 - 8.0 \le 0.0$$

$$\frac{\partial g_1}{\partial x_1} = 2x_1 + 1.0, \quad \frac{\partial g_1}{\partial x_2} = 2x_2 - 1.0, \quad \frac{\partial g_1}{\partial x_3} = 2x_3 + 1.0, \quad \frac{\partial g_1}{\partial x_4} = 2x_4 - 1.0$$

$$g_2 = x_1^2 - x_1 + 2x_2^2 + x_3^2 + 2x_4^2 - x_4 - 10.0 \le 0.0$$

$$\frac{\partial g_2}{\partial x_1} = 2x_1 - 1.0, \quad \frac{\partial g_2}{\partial x_2} = 4x_2, \quad \frac{\partial g_2}{\partial x_3} = 2x_3, \quad \frac{\partial g_2}{\partial x_4} = 4x_4 - 1.0$$

$$g_3 = 2x_1^2 + 2x_1 + x_2^2 - x_2 + x_3^2 - x_4 - 5.0 \le 0.0$$

$$\frac{\partial g_3}{\partial x_1} = 4x_1 + 2.0, \quad \frac{\partial g_3}{\partial x_2} = 2x_2 - 1.0, \quad \frac{\partial g_3}{\partial x_3} = 2x_3, \quad \frac{\partial g_3}{\partial x_4} = -1.0$$

# Rosen-Suzuki Function Optimization



## Optimization Model Definition

```
OptimizationModel om = optimizationModel("DesignExploration Model",
                designVars(vars(loop(4), "x", 0.0,-100., 100.)),
                responseVars("f"),
                responseVars("g",3),
                objectiveVars(var("fo", "f", Target.min )),
                constraintVars(var("g1c", "g1", Relation.lt, 0.0),
                               var("g2c", "g2", Relation.lt, 0.0),
                               var("g3c","g3", Relation.lt, 0.0)))
);
```

# Rosen-Suzuki Function Optimization



## Optimization Model Configuration

### Sensitivity Evaluators for the Objective

$$\frac{\partial f}{\partial x_1} = 2x_1 - 5, \quad \frac{\partial f}{\partial x_2} = 2x_2 - 5, \quad \frac{\partial f}{\partial x_3} = 4x_3 - 21, \quad \frac{\partial f}{\partial x_4} = 2x_4 + 7$$

```
Evaluator dfedx1e1 = evaluator("dfedx1e1", "2.0*x1-5.0",args(om.getDesignVars("x1")));

Evaluator dfedx2e1 = evaluator("dfedx2e1", "2.0*x2-5.0",args(om.getDesignVars("x2")));

Evaluator dfedx3e1 = evaluator("dfedx3e1", "4.0*x3-21.0",args(om.getDesignVars("x3")));

Evaluator dfedx4e1 = evaluator("dfedx4e1", "2.0*x4+7.0",args(om.getDesignVars("x4")));

List<Evaluator> feg1 = list(dfedx1e1, dfedx2e1, dfedx3e1, dfedx4e1);
om.setGradientEvaluators("f", "fe", "feg1",  feg1);
```

# Rosen-Suzuki Function Optimization

SORCER

## Optimization Model Configuration

### Sensitivity Evaluators for the Constraints

$$\frac{\partial g_1}{\partial x_1} = 2x_1 + 1.0, \quad \frac{\partial g_1}{\partial x_2} = 2x_2 - 1.0, \quad \frac{\partial g_1}{\partial x_3} = 2x_3 + 1.0, \quad \frac{\partial g_1}{\partial x_4} = 2x_4 - 1.0$$

Evaluator dg1edx1e1 = *evaluator("dg1edx1e1", "2.0*x1+1",args(om.getDesignVars("x1")));*

Evaluator dg1edx2e1 = *evaluator("dg1edx2e1", "2.0*x2-1.0",args(om.getDesignVars("x2")));*

Evaluator dg1edx3e1 = *evaluator("dg1edx3e1", "2.0*x3+1.0",args(om.getDesignVars("x3")));*

Evaluator dg1edx4e1 = *evaluator("dg1edx4e1", "2.0*x4-1.0",args(om.getDesignVars("x4")));*

List<Evaluator> g1eg1 = *list(dg1edx1e1, dg1edx2e1, dg1edx3e1, dg1edx4e1);*
om.setGradientEvaluators("g1", "g1e", "g1eg1", g1eg1);

# Rosen-Suzuki Function Optimization



## Optimization Model Configuration

### Sensitivity Evaluators for the Constraints

$$\frac{\partial g_2}{\partial x_1} = 2x_1 - 1.0, \quad \frac{\partial g_2}{\partial x_2} = 4x_2, \quad \frac{\partial g_2}{\partial x_3} = 2x_3, \quad \frac{\partial g_2}{\partial x_4} = 4x_4 - 1.0$$

Evaluator dg2edx1e1 = *evaluator("dg2edx1e1", "2.0*x1-1",args(om.getDesignVars("x1")));*

Evaluator dg2edx2e1 = *evaluator("dg2edx2e1", "4.0*x2",args(om.getDesignVars("x2")));*

Evaluator dg2edx3e1 = *evaluator("dg2edx3e1", "2.0*x3",args(om.getDesignVars("x3")));*

Evaluator dg2edx4e1 = *evaluator("dg2edx4e1", "4.0*x4-1.0",args(om.getDesignVars("x4")));*

List<Evaluator> g2eg1 = *list(dg2edx1e1, dg2edx2e1, dg2edx3e1, dg2edx4e1);*
om.setGradientEvaluators("g2", "g2e", "g2eg1", g2eg1);

# Rosen-Suzuki Function Optimization



## Optimization Model Configuration

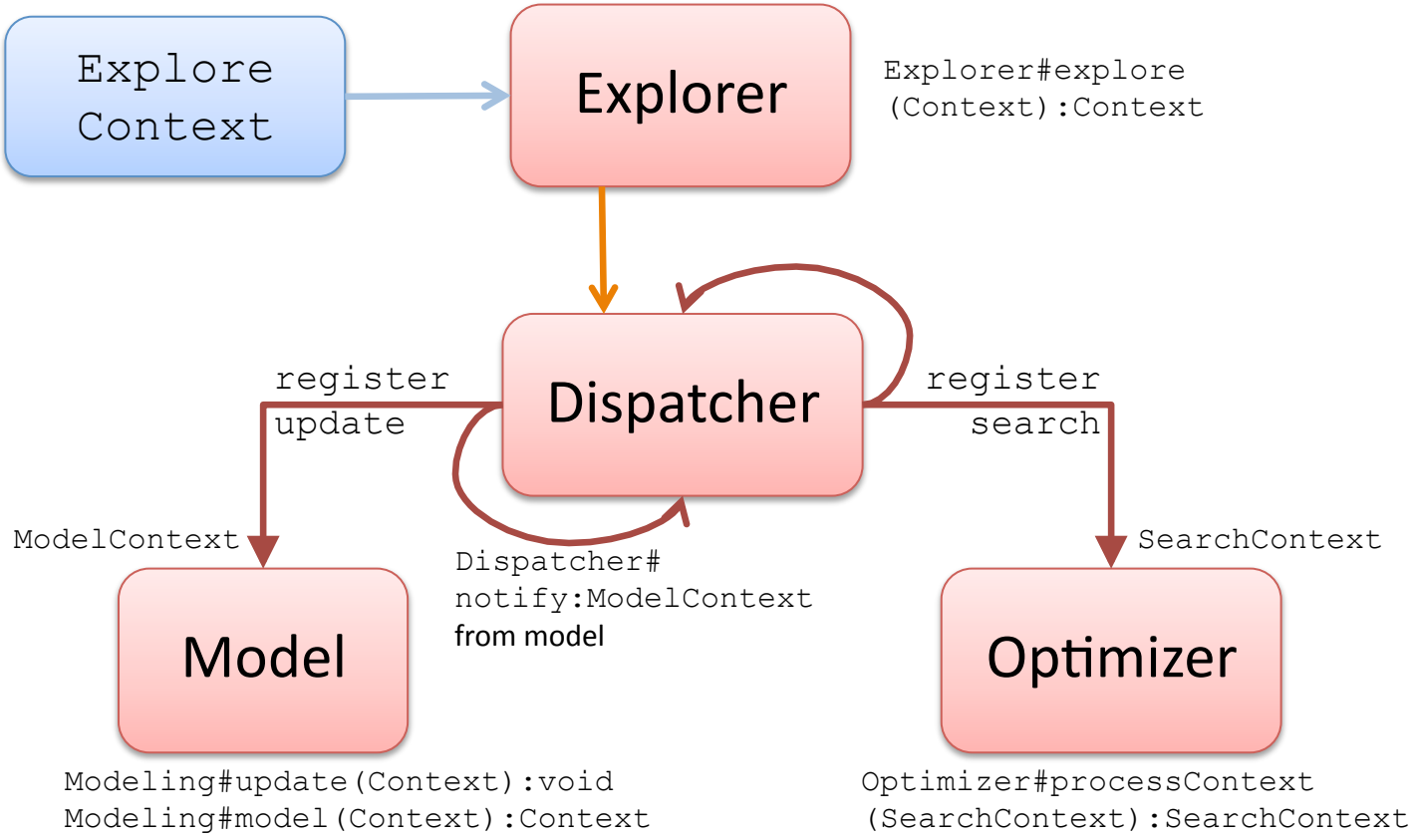
### Sensitivity Evaluators for the Constraints

$$\frac{\partial g_3}{\partial x_1} = 4x_1 + 2.0, \quad \frac{\partial g_3}{\partial x_2} = 2x_2 - 1.0, \quad \frac{\partial g_3}{\partial x_3} = 2x_3, \quad \frac{\partial g_3}{\partial x_4} = -1.0$$

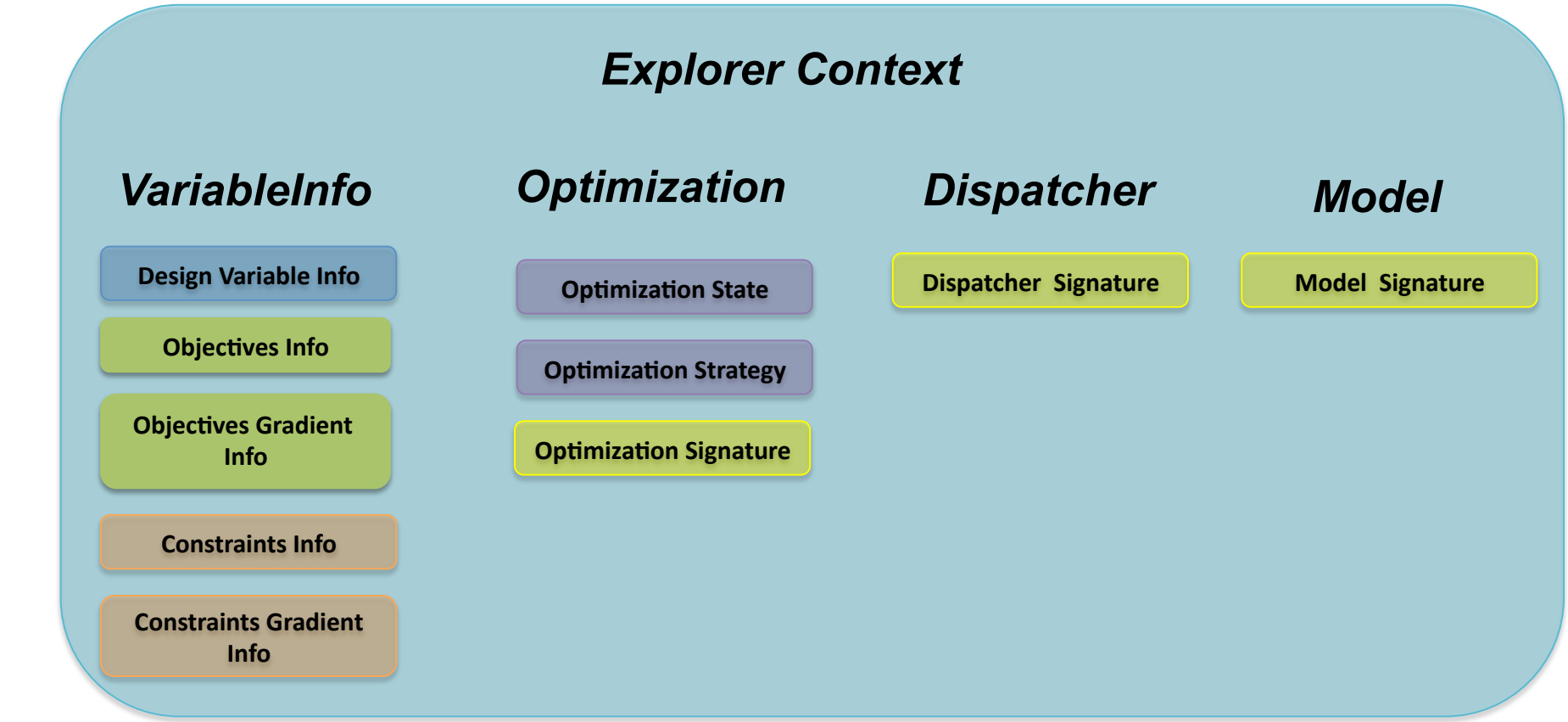
```
Evaluator dg3edx1e1 = evaluator("dg3edx1e1", "4.0*x1+2.0",args(om.getDesignVars("x1")));

Evaluator dg3edx2e1 = evaluator("dg3edx2e1", "2.0*x2-1.0",args(om.getDesignVars("x2")));

Evaluator dg3edx3e1 = evaluator("dg3edx3e1", "2.0*x3",args(om.getDesignVars("x3")));

Evaluator dg3edx4e1 = evaluator("dg3edx4e1", "-1.0");

List<Evaluator> g3eg1 = list(dg3edx1e1, dg3edx2e1, dg3edx3e1, dg3edx4e1);
om.setGradientEvaluators("g3", "g3e", "g3eg1", g3eg1);
```

# Rosen-Suzuki Function Optimization

## Exploration



# Explorer Context (Data)




# Rosen-Suzuki Function Optimization

SORCER

## Explorer Context

```
ExploreContext exploreContext = new ExploreContext("Rosen-Suzuki");
VarInfoList designInfo = varsInfo(varInfo("x1", 1.0), varInfo("x2", 1.0), varInfo("x3", 1.0), varInfo("x4", 1.0));
exploreContext.setDesignVarsInfo(designInfo);

exploreContext.setObjectivesInfo(null);
exploreContext.setObjectivesGradientInfo(null);

exploreContext.setConstraintsInfo(null);
exploreContext.setConstraintsGradientInfo(null);

ConminStrategy cmnStrategy=new ConminStrategy(new File(System.getProperty("conmin.strategy.file")));
exploreContext.setOptimizerStrategy(cmnStrategy);
//exploreContext.setOptimizerState(cmnState); // state initialized in Dispatcher after first Model update


// specify the Model
exploreContext.setModelSignature(signature("createModel", RosenSuzukiModelCreator.class, Type.INTRAPROCESS));

//specify the Optimizer
exploreContext.setOptimizerSignature(signature(null, ConminOptimizerJNA.class, Type.INTRAPROCESS));

// specify the explorer dispatched
exploreContext.setDispatcherSignature(signature(null, RosenSuzukiDispatcher.class, Type.INTRAPROCESS));
```

# Rosen-Suzuki Function Optimization

SORCER

## Explorer Execution

```
outContext = (ExploreContext)explorer.explore(exploreContext);

********* CONMIN STATE **************

CONMIN Iteration # =  29

Objective Fuction fo = 6.00260795945281

Design Variable Values
   x1 = 2.5802964087086235E-4 x2 = 0.9995594642481355 x3 = 2.000313835134211 x4 = -0.9986692050113675

Constraint Values
   g1c = -0.002603585246998996 g2c = -1.0074147118087602 g3c = 4.948009193483927E-7
T
Termination Criterion
    ABS(OBJ(I)-OBJ(I-1))   LESS THAN DABFUN = 5.0E-5 FOR 3 ITERATIONS

Evaluation Statistics
   Number of Objective Evaluations = 88
   Number of Constraint Evaluations = 88
   Number of Objective Gradient Evaluations = 29
    Number of Constraint Gradient Evaluations = 29
```

**Analytic Optimum Results**
**fo  = 6.0**
**X1=0.0, x2=1.0, x3=2.0, x4=-1.0**
**g1c=0.0, g2c=-1.0, g3c=0.0**


# Rosen-Suzuki Function Optimization



***DEMO***

# MSTC SORCER Applications to Date

SORCER

- Hi-Fidelity Aeroelastic Analysis
  - Euler (AVUS), FEM (MSCNASTRAN), MDICE
- Hi-Fidelity Induced Drag minimization
  - Euler (AVUS), Optimization (CONMIN, MATLAB)
- 2-D large amplitude airfoil motion trajectory Optimization
  - Unsteady Vortex Lattice Method (in-house UVLM), Opti (DOT)
- Shape and sizing aeroelastic optimization
  - Euler (AVUS), FEM (MSCNASTRAN), MDICE, Opti (DOT) FD sensitivities

- Demonstrated tight integration with C, C++, and FORTRAN using JNA (can make direct calls to any shared object)
- Network configuration
  - Linux workstations
  - Linux cluster(2), Mac cluster(1)
  - SGI Irix
  - Windows
  - Mac Desktop
  - Mac laptop

# Induced Drag Optimization
# Motivation

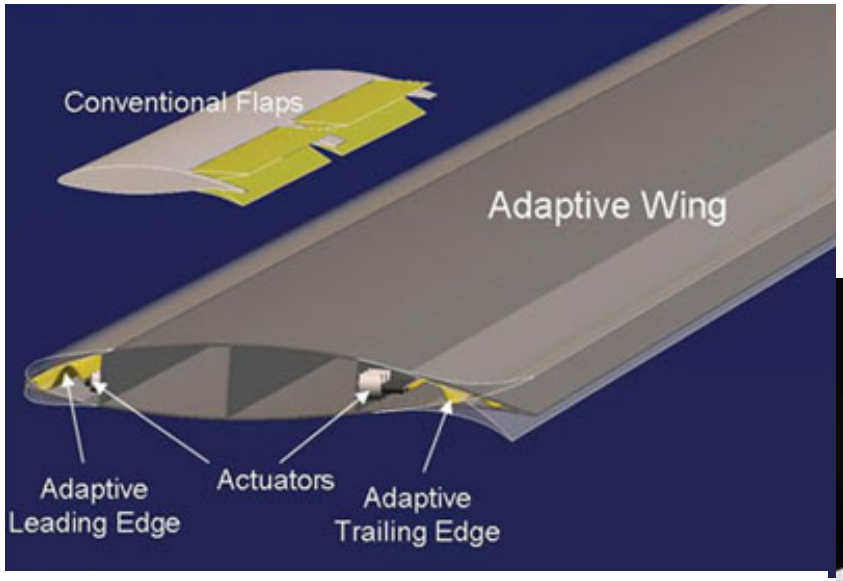

Long duration on station

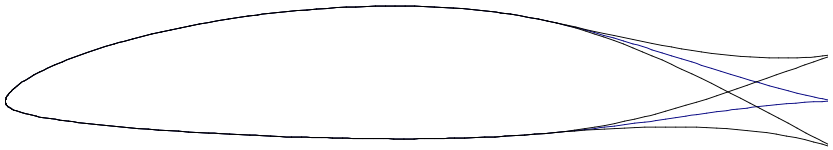
Large changes of $C_L$ over the mission

# A Solution

SORCER

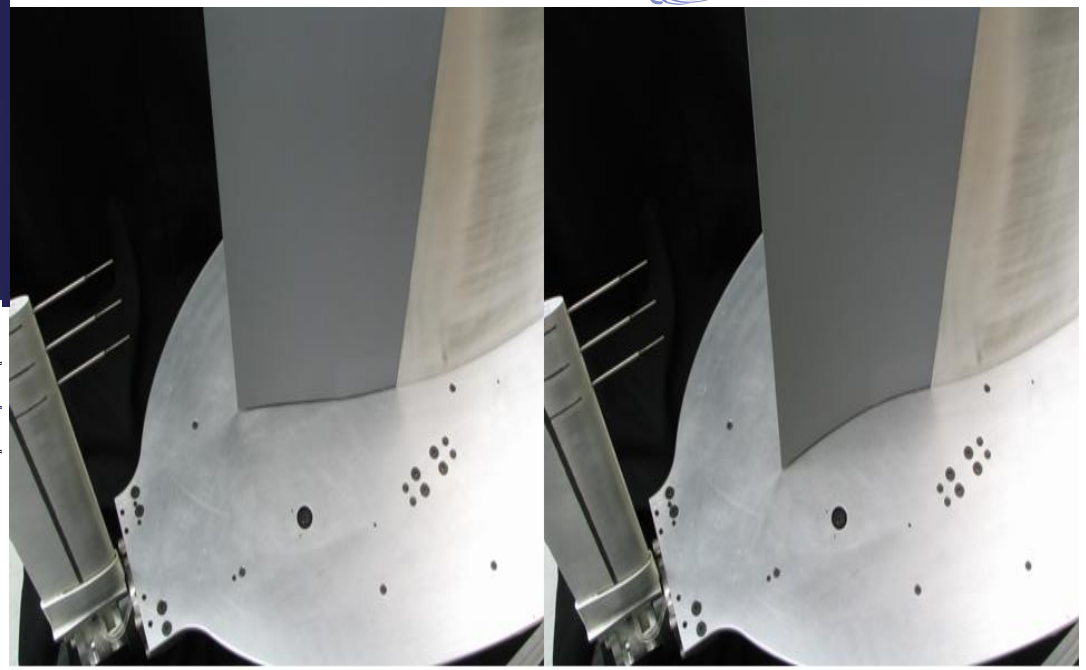
- Utilize conformal control surfaces to enhance laminar flow, minimize the Drag



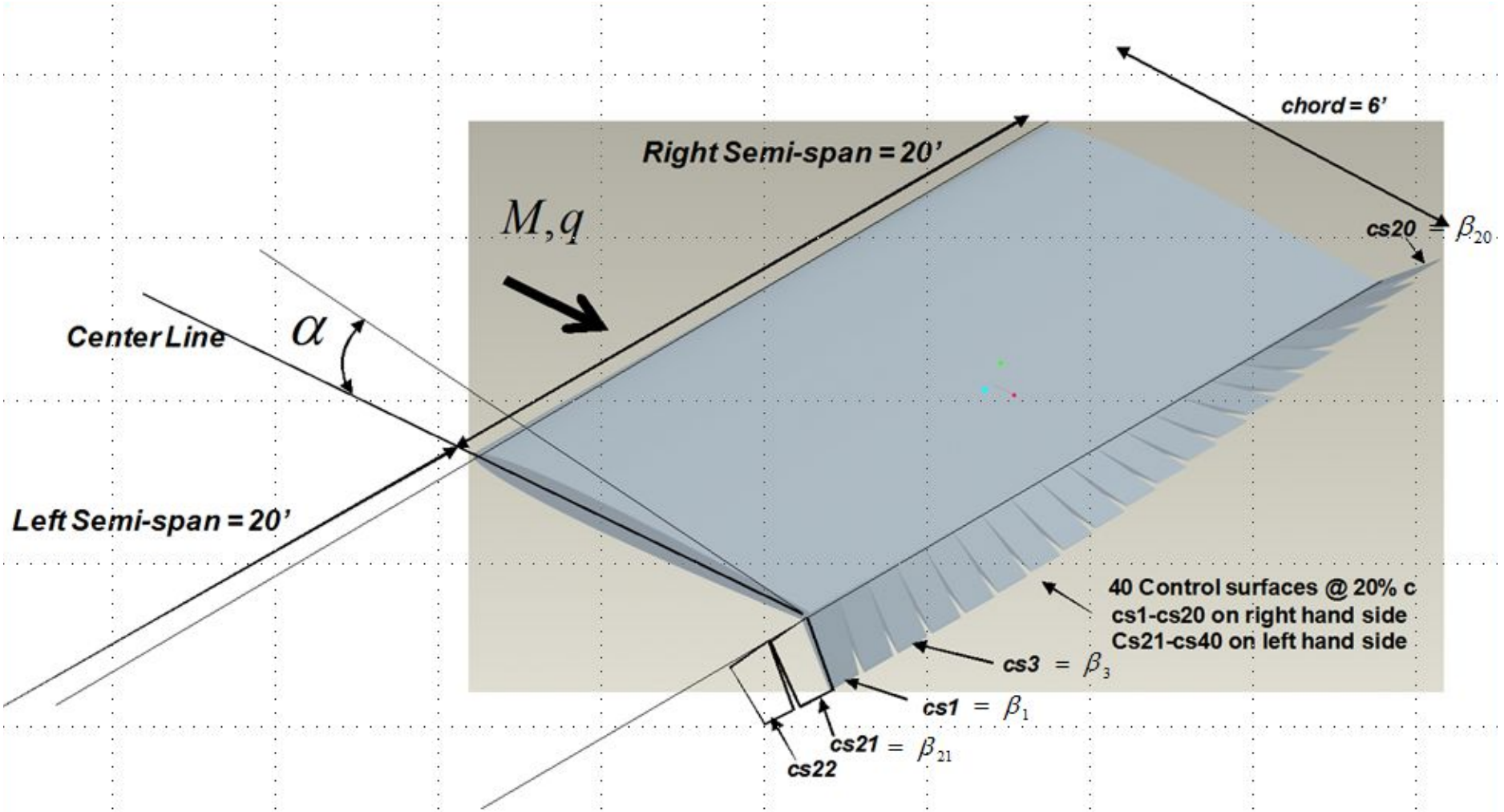
Adaptive Compliant Wing

FlexSys Inc.



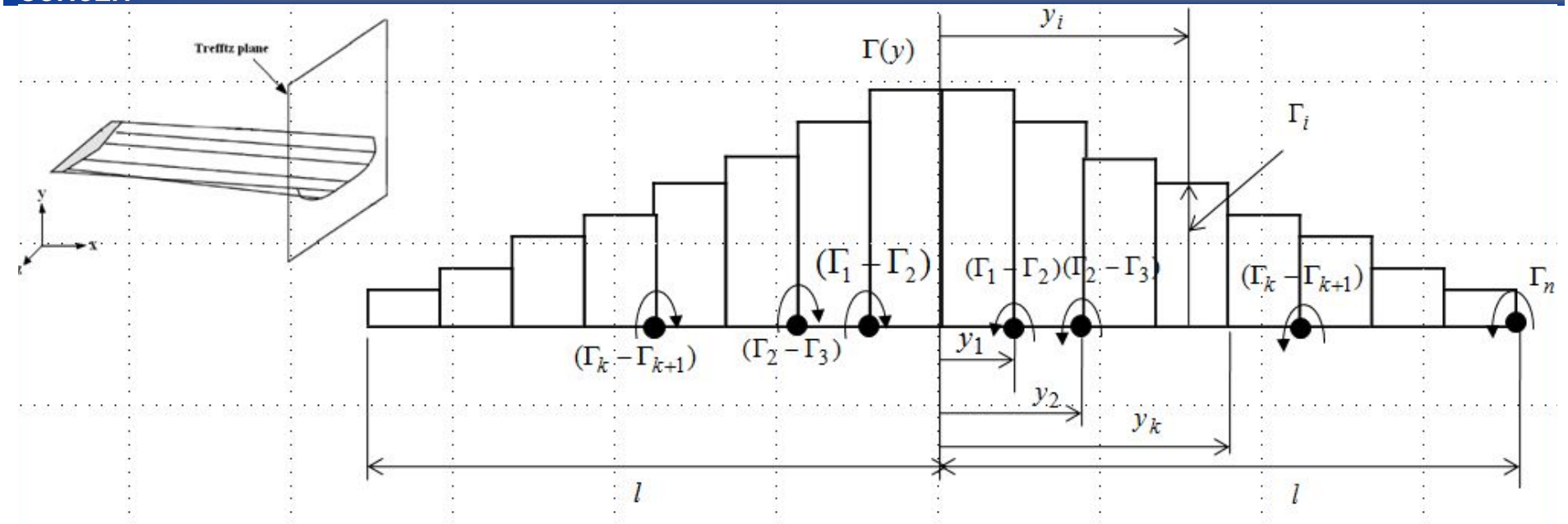
# Modeling Conformal Control Surfaces

SORCER



# Trefftz-Plane Induced Drag With Euler Aerodynamics

SORCER




$$D_I = \frac{-1}{4\pi\rho_\infty V_0^2}\sum_{i=1}^{n} Lpus_i \Delta y_i \sum_{k=1}^{n}\left[Lpus_k - Lpus_{k+1}\right]\left[\frac{1}{y_i - y_k} - \frac{1}{y_i + y_k}\right]$$

$Lpus_j(\alpha,\beta_i)$ – Lift per unit span (Euler calculations)

$\alpha$ - angle of attack

$\beta_i$ - control surface deflection of $i$th surface

Design Variables $\bar{x} = \{\alpha, \beta_i\}^T$

$$Lpus_k = \rho_\infty V_\infty \Gamma_k$$ (Kutta-Joukowski theorem)

# Optimization Problem



$$\text{Minimize}: \quad D_I(Lpus_i(\alpha,\beta_i,xl_i))$$

$$\text{Such That}: \quad L_T(Lpus_i(\alpha,\beta_i,xl_i)) = L$$

$$-5.0 \le \alpha \le 5.0$$

$$-10.0 \le \beta_i \le 10.0$$

$$\text{Design Variables } x_i = \{\alpha,\beta_{1-20}\}^T$$

$$\text{Linked Variables } xl_i = \{\beta_{21-40} = \beta_{1-20}\}^T$$

$$\text{Response Variables}: \; D_I(Lpus_i(\alpha,\beta_i,xl_i)),\; L_T(Lpus_i(\alpha,\beta_i,xl_i)),\; Lpus_i(\alpha,\beta_i,xl_i)$$

$D_I(Lpus_i(\alpha,\beta_i,xl_i))$ and $L_T(Lpus_i(\alpha,\beta_i,xl_i))$ are functions of fuctions and $Lpus_i(\alpha,\beta_i,xl_i)$ is a function of the independent and linked design variables

**At Transonic Mach #'s Computation of Lpus requires Euler Solution**

# Evaluators for Induced Drag and Sensitivities



$$D_I = \frac{-1}{4\pi\rho_\infty V_0^2}\sum_{i=1}^{n} Lpus_i \Delta y_i \sum_{k=1}^{n}\left[Lpus_k - Lpus_{k+1}\right]\left[\frac{1}{y_i - y_k} - \frac{1}{y_i + y_k}\right]$$

**Exact – Perform Euler solution for** $Lpus_i, \nabla Lpus_i$ By FD Euler Solutions

**Standard One Point Approximation (SOA)** – Perform Euler solutions $D_I(\bar{x}^0), \nabla D_I(\bar{x}^0)$

$$\tilde{D}_I(\bar{x}) = D_I(\bar{x}^0) + \nabla D_I(\bar{x}^0)\{\bar{x} - \bar{x}^0\} \qquad \frac{\partial \tilde{D}_I(\bar{x})}{\partial x_i} = \frac{\partial D_I(\bar{x}^0)}{\partial x_i} = \text{constant}$$

**Modified One Point Approximation (MOA)** – Perform Euler solution $Lpus_i(\bar{x}^0), \nabla Lpus_i(\bar{x}^0)$

$$\tilde{D}_I(\bar{x}) = \frac{-1}{4\pi\rho_\infty V_0^2}\sum_{i=1}^{n} \tilde{L}pus_i \Delta y_i \sum_{k=1}^{n}\left[\tilde{L}pus_k - \tilde{L}pus_{k+1}\right]\left[\frac{1}{y_i - y_k} - \frac{1}{y_i + y_k}\right] \qquad \frac{\partial \tilde{D}_I(\bar{x})}{\partial x_i} \neq \text{constant}$$

$$\tilde{L}pus_i(\bar{x}) = \tilde{L}pus_i(\bar{x}^0) + \nabla \tilde{L}pus_i(\bar{x}^0)\{\bar{x} - \bar{x}^0\}$$

**Kriging** – Perform Euler solutions (hundreds) for $D_I$ Kriging Model

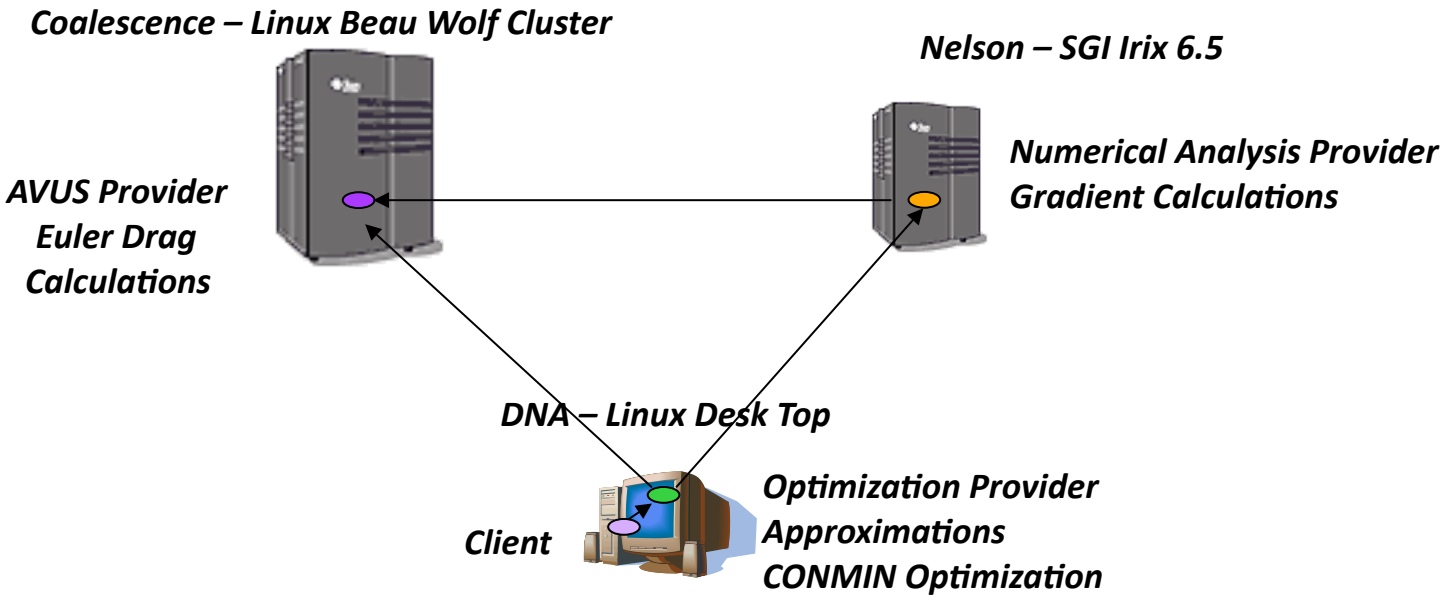
# Induced Drag Optimization Model

**SORCER**

```
OptimizationModel omodel = optimizationModel("Induced Drag",

designVars(vars(loop("i",1,20),"beta$i$", 0.0, -10.0, 10.0)),
linkedVars(names(loop("i",21,40),"betal$i$")),
designVars(var("alpha", 5.0, -5.0, 10.0)),
designVars("mach", "gamma", "pstatic"),

responseVars(var("DI",
        realization(
                evaluation("DIExacte",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("DIExacteg1"))),
                    evaluation("DISOAe",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("DISOAeg1"))),
                evaluation("DIMOAe",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("DIMOAeg1"))),
                evaluation("DIKrige",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("DIKrigeg1"))))),
responseVars(var("LT",
        realization(
                evaluation("LTExacte",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("LTExacteg1"))),
                evaluation("LTSOAe",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("LTSOAeg1"))),
                evaluation("LTMOAe",
                        differentiation(wrt(names(loop("i",1,20),"beta$i$"), "alpha","q"), gradient("LTMOAeg1"))))),
responseVars(loop("i",1,20),"Lpus$i$",
        realization(
                evaluation("Lpus$i$Exacte",
                        differentiation(wrt(names(loop("k",1,20),"beta$k$"), "alpha"), gradient("Lpus$i$Exacteg1"))),
                evaluation("Lpus$i$SOAe",
                        differentiation(wrt(names(loop("k",1,20),"beta$k$"), "alpha"), gradient("Lpus$i$SOAeg1"))))),
responseVars(var("q",
    realization(evaluation("qExact", differentiation(wrt("mach","gamma","pstatic"), gradient("qExactg1"))))),

objectiveVars(var("DIo", "DI", Target.min )),
constraintVars(var("LTc", "LT", Relation.eq, 1000.0))
);
```

# SORCER Network Configuration





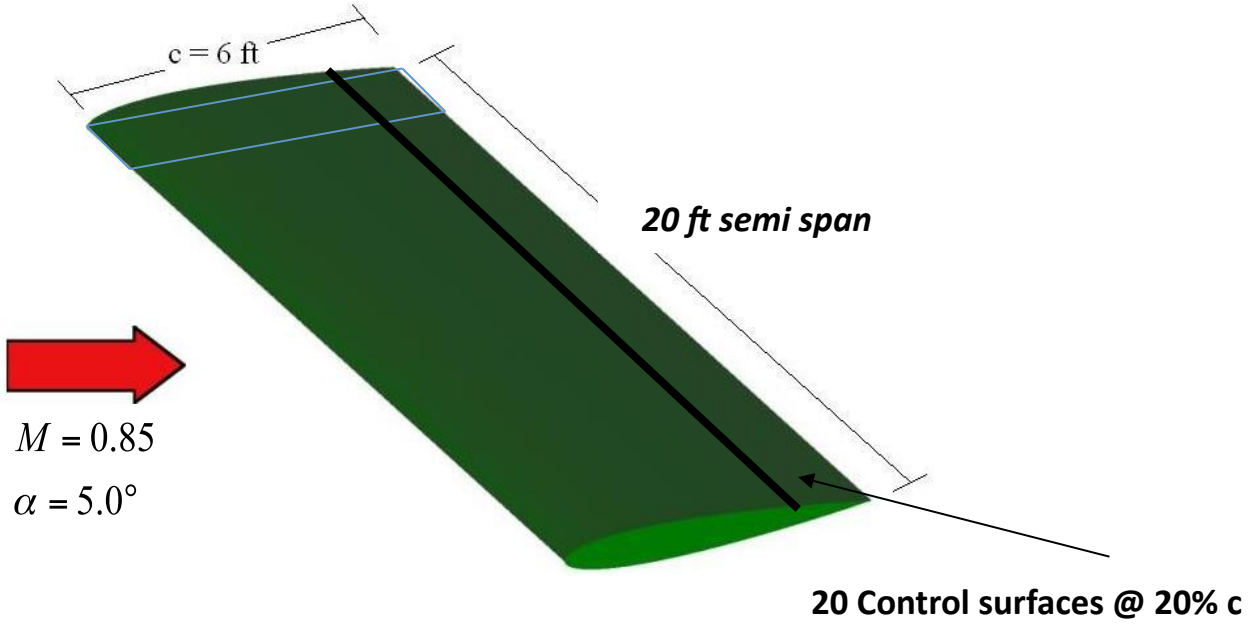

Sobolewski, M., Kolonay, R "Federated Grid Computing with Interactive Service-oriented Programming," International Journal of Concurrent Engineering: Research & Applications, Vol. 14, No 1 2006 pp. 55-66

# Example Case

SORCER

## *Goland Wing Planform with a NACA 0006 Airfoil*

## *(Euler Calculations- AVUS (1.5 million cells)*



# Modified First Order Taylor Series Approximation



**Comparison of Traditional and Modified First Order Taylor Series Expansion**

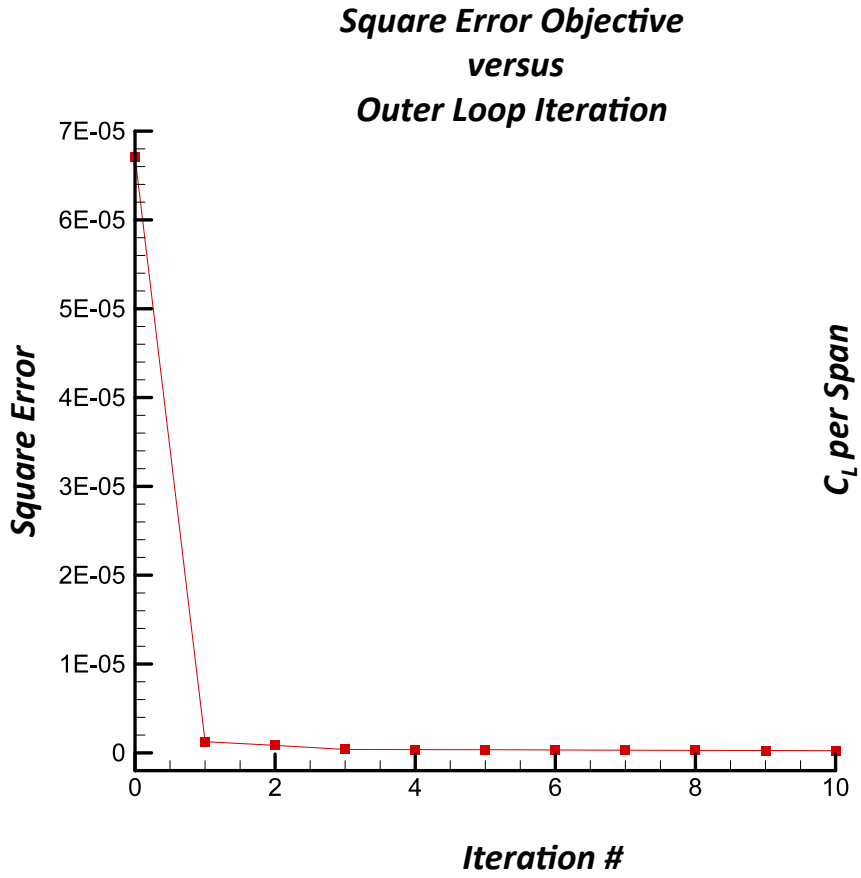
$V^0 \quad \alpha^0 = 5.0, M = 0.85, \beta_i^0 = 0.0$

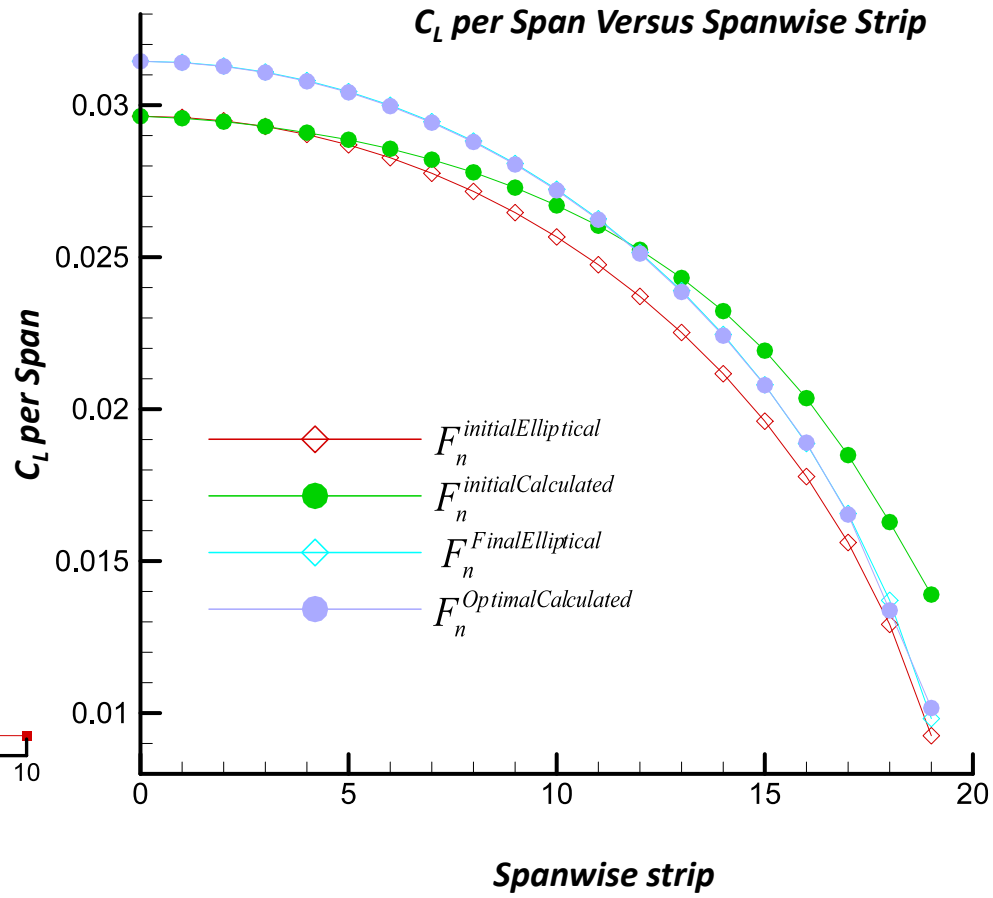
$V^j \quad \alpha^j = 5.0, M^j = 0.85, \beta_i^j = \{0.79,0.94,0.98,0.98,0.95,0.80,0.76,0.73,0.60,0.52,0.36,$
$0.10,-0.08,-0.19,-0.34,-0.78,-1.20,-0.84,-1.71,-8.91\}$

| | Value | % Diff |
|---|---|---|
| Exact Function | 1.18388E-03 | 0.0% |
| Tradition First Order Taylor Series Expansion | -4.380E-04 | -137% |
| Modified First Order Taylor Series Expansion | 1.18708E-03 | 0.27% |

# Goland Wing M=0.85, Alpha=5.0

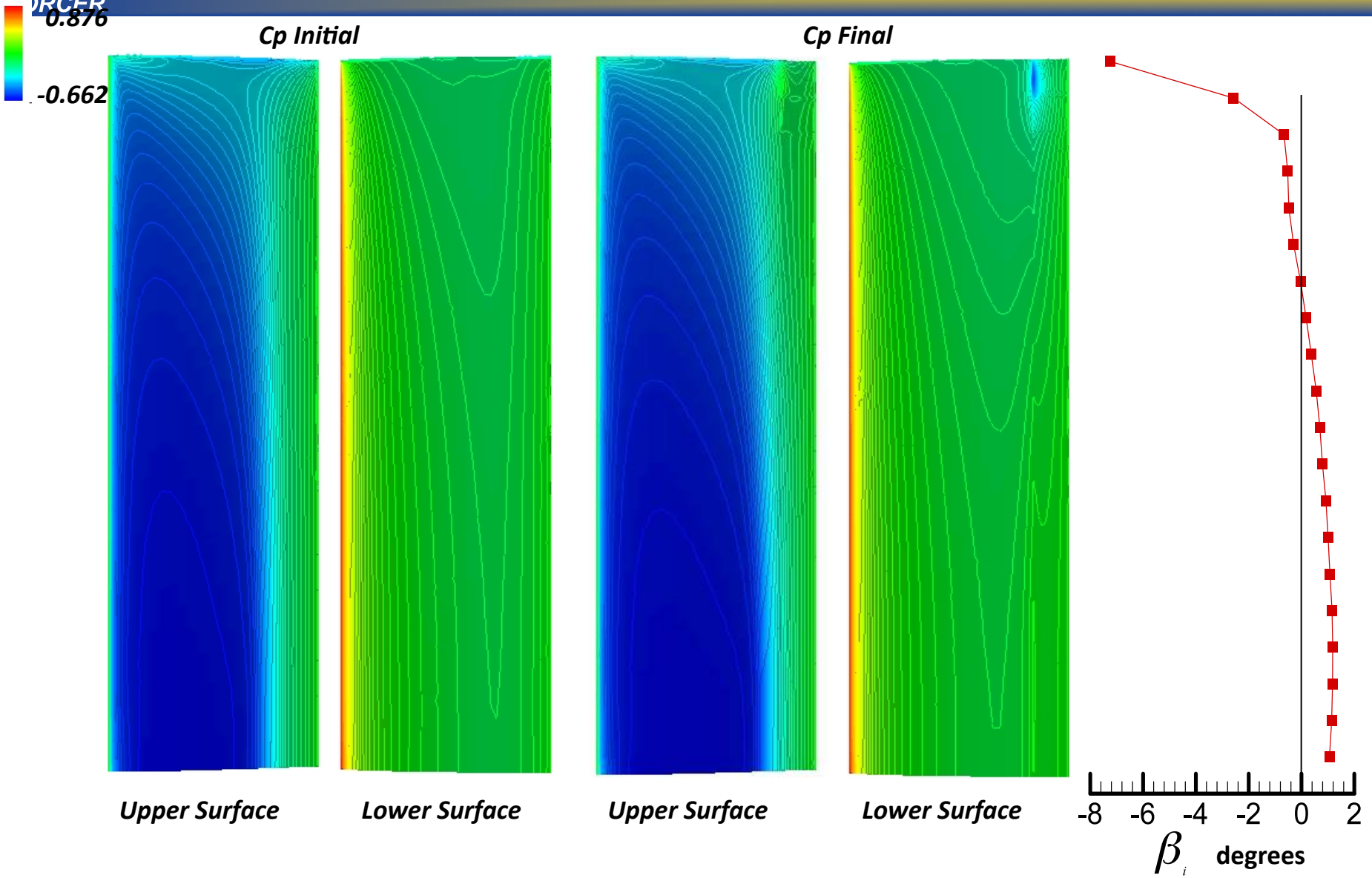
SORCER




*Lift constraint satisfied to within 0.3%

# Goland Wing M=0.85, Alpha=5.0



# Shape and Sizing Aeroelastic Optimization

SORCER

**Minimize**
**Objective Function**
**Subject to**

$$f(\beta_i) = W_{Structural}$$

**Inequality Constraints**

$$g_1(\beta_i) = 1.0 - \frac{C_{M_{roll}}}{C_{M_{roll},ref}}$$

$$g_j(\beta_i) = 1.0 - \frac{\sigma_{VonMises}}{\sigma_{MaxAllowable}} \le 0 \qquad j = 2, N_{Constr.}$$

**Side Constraints**

$$\beta_i^L \le \beta_i \le \beta_i^U \qquad i = 1, N_{DesignVariables}$$

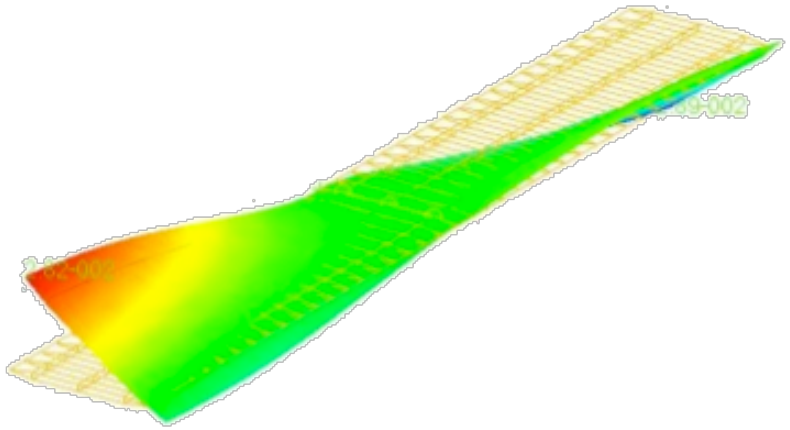
$\beta_i$ - Structural sizes, and planform sweep

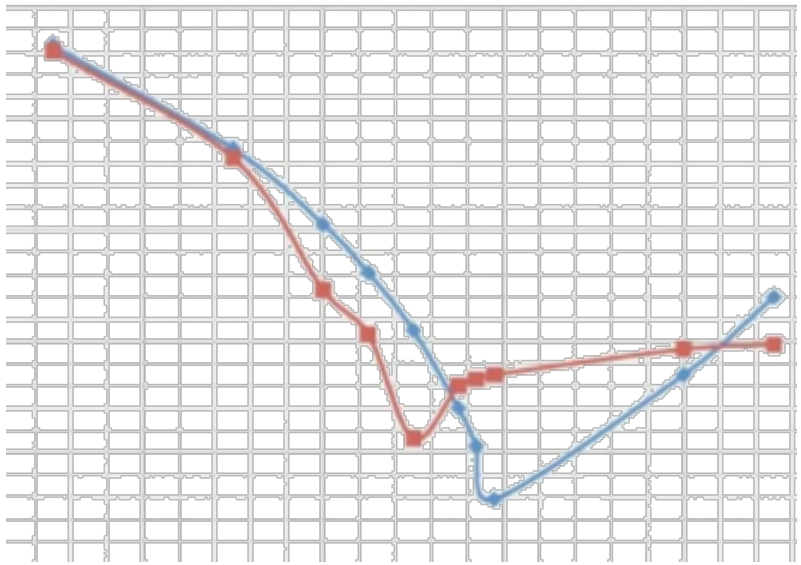
**Ernest Thompson – AFRL/RBSD**

# Min weight shape & sizing opti

SORCER

- Euler Static Aeroelastic  Calculations

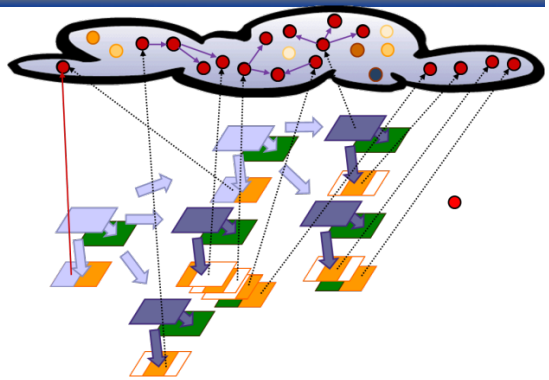





**Ernest Thompson – AFRL/RBSD**

# Sorcer network configuration for min weight (shape and sizing)

SORCER



Coalescence

MDICE

Avus

RBSDZ0010143

Microsoft Windows

NMEDA

MDICE

Avus

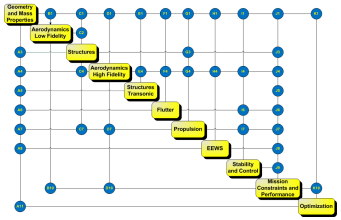
Whittle

Gridgen

DOT

NASTRAN

Client



Gladius

NASTRAN

StructureTransform

Xserv2

MDICE

Avus

Gothmog

Java

JavaSpace

**Ernest Thompson – AFRL/RBSD**

# Concluding Remarks



- Aerospace Vehicle Design requires executing large scale system level MDA/MDO in a heterogenous computing environment
- SORCER exposes Sun's Jini connection Technology to create a Model Oriented Programming (MOP) and computing environment for large scale MDA/MDO
- SORCER Functionality
  - Create providers and expose them as jini services.
  - Exertion Oriented Programming – Ability to orchastrate Services
  - Models for analysis, sensitvitities, and design
  - VEF – Functional demand driven computations
  - Space Computing

# *Comments & Questions?*