# Surrogate Modeling Using SU2

## Pointwise® and SU2 Joint Workshop

## Sept 29-30, 2014

Trent Lukaczyk

Aeronautics & Astronautics Department (Stanford University)

The Open-Source CFD Code

aerospacedesignlab

I) PROBLEM SETUP
II) DESIGN OF EXPERIMENTS
III) DATA PROCESSING
IV) OPTIMIZATION

SU2
The Open-Source CFD Code
aerospacedesignlab

# A Simple Script

This script runs a simple drag polar

```
import SU2

# load config
config = SU2.io.Config('naca0012.cfg')

# set file state
state = SU2.io.State()
state.find_files(config)

# lists for drag polar
angles = [-4.,-2.,0.,2.,4.]
drags, lifts  = [], []
```

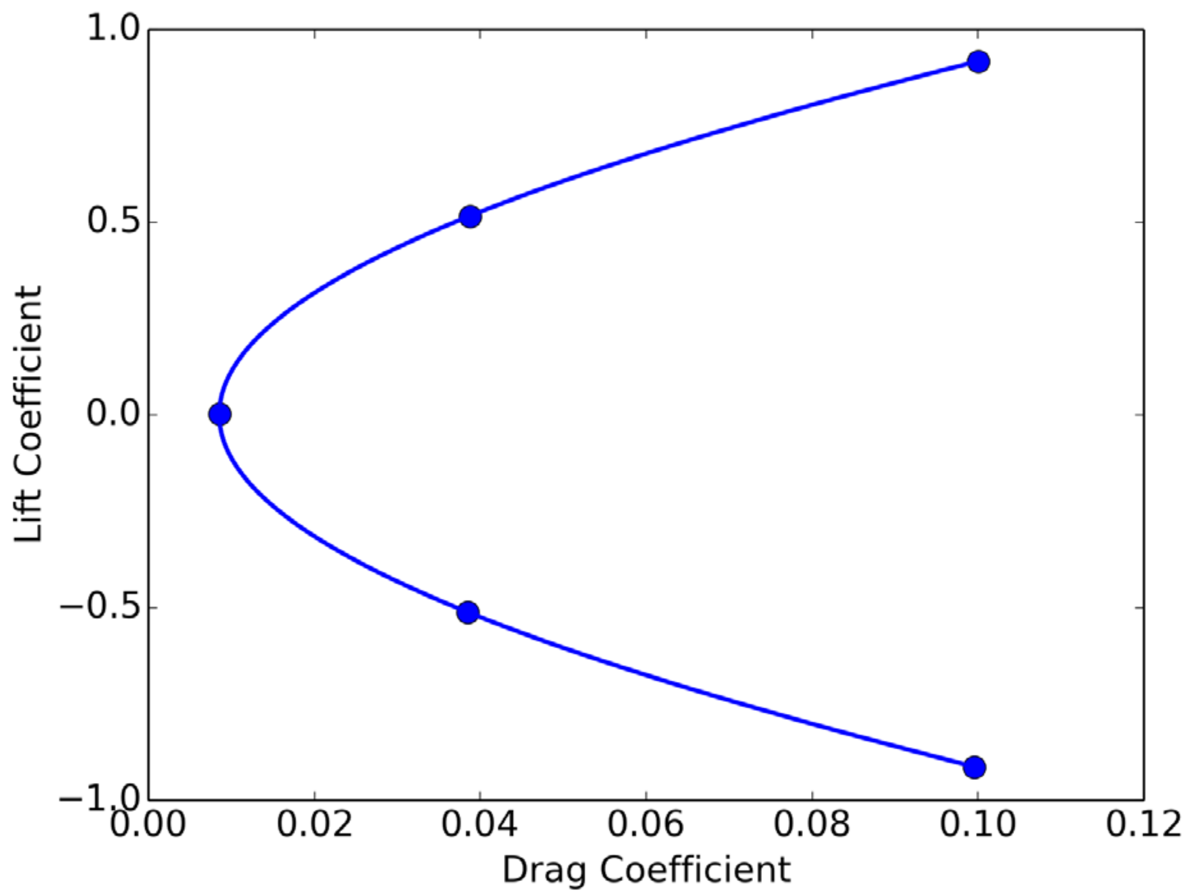
# A Simple Script

This script runs a simple drag polar

```
# iterate angles
for angle in angles:

    # local config and state
    konfig = copy.deepcopy(config)
    ztate  = copy.deepcopy(state)

    # set angle of attack
    konfig.AoA = angle

    # run su2
    drag = SU2.eval.func('DRAG',konfig,ztate)
    lift = SU2.eval.func('LIFT',konfig,ztate)

    # update data lists
    drags.append(drag)
    lifts.append(lift)
```

# A Simple Script

The resulting drag polar plot



# NACA 0012 Optimization Problem

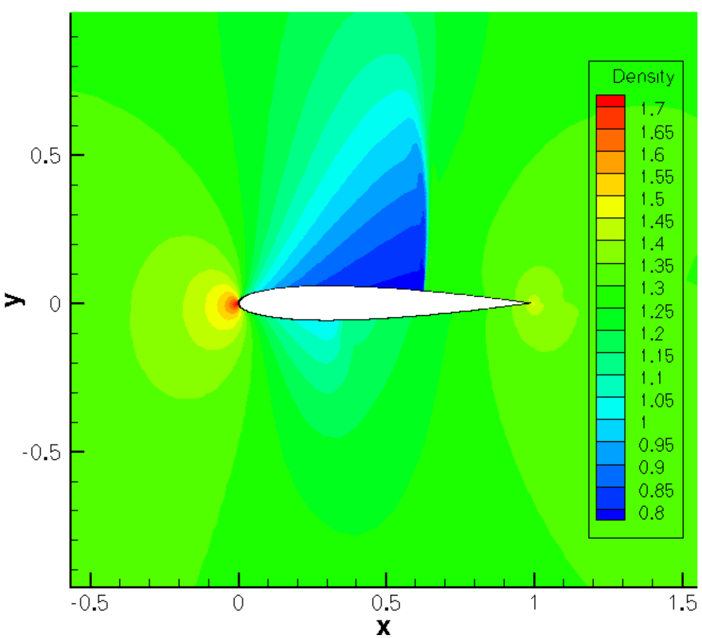
Ma=0.8, AoA=1.25$^0$

Euler second order

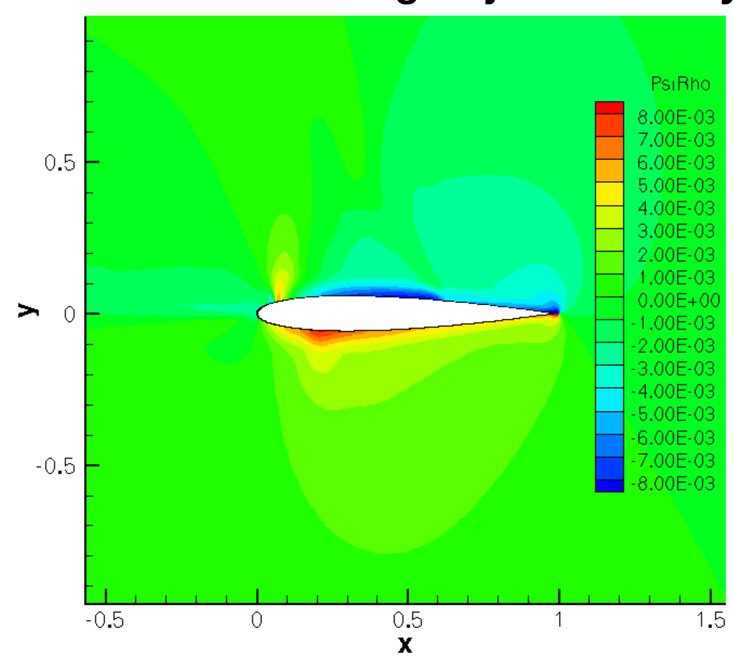
Surface based continuous adjoint formulation

Hicks-Henne bump function design variables

**Contours of Density**



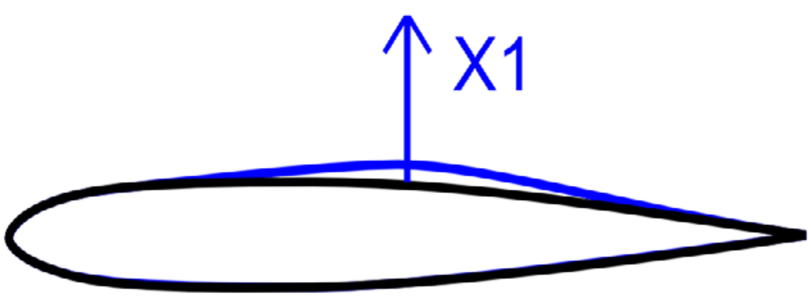

**Contours of Drag Adjoint Density**



## Problem Setup

Minimize drag while maintaining a minimum lift

Vary the airfoil's shape with one Hicks Henne Bump Function




Min. $C_D(X1)$

s.t. $C_L(X1) > 0.3200$

# SU2 Project Setup

SU2 has a python object that can manage design evaluations – SU2.opt.Project(). Here's how to set it up.

```
# load config
config = SU2.io.Config('config_naca0012.cfg')

# modify config
## writes iteration history to log files
config.CONSOLE      = 'CONCISE'
## number of processors for parallel solves
config.NUMBER_PART = 4

# set file state
state = SU2.io.State()
state.find_files(config)

# start project
project = SU2.opt.Project( config, state ,
                           folder=project_folder)
```

# Saving and Loading Projects

Projects save themselves with each call to a major component, like deformations, direct solutions, or adjoint solutions.

```
# try to load project
if os.path.exists('Project_Folder/project.pkl'):
    project = SU2.io.load_data('Project_Folder/project.pkl')
```

Loading them can save you the time of re-evaluating solutions.

To start over, delete the project.pkl file, or move it to an archive directory.

# Design of Experiments

To generate a surrogate model, we need to take a random sampling of the design space. This sampling we choose by a design of experiments. In this case we'll use Latin Hypercube Sampling.

```
# number of random samples
NS = 4

# bounds
XB = np.array([[-0.01,0.01]]*1)

# initial sample
X0 = np.zeros([1,1])

# generate sample locations with latin hypercube
XS = VyPy.sampling.lhc_uniform(XB, NS, X0)
```

# Running the Experiments

This loop evaluates SU2 at each design sample.

```
for i,x in enumerate(XS):

    # unpack design into a config
    print 'X_FFD:' , x
    konfig,_ = project.unpack_dvs(x)

    # Run SU2
    print 'EVALUATE SU2 DIRECT'
    f_drag = project.func('DRAG',konfig)
    f_lift = project.func('LIFT',konfig)

    print 'EVALUATE SU2 DRAG ADJOINT'
    df_drag = project.grad('DRAG','ADJOINT',konfig)

    print 'EVALUATE SU2 LIFT ADJOINT'
    df_lift = project.grad('LIFT','ADJOINT',konfig)
```

## Checking the Logs

While SU2 is Running, you can check the log files to make sure it's doing what you expected.

```
trent@ubuntu:NACA_Case$ cd projects/Test_Project/
trent@ubuntu:Test_Project$ cd DESIGNS/DSN_001/
trent@ubuntu:DSN_001$ cd DIRECT/
trent@ubuntu:DIRECT$ tail log_Direct.out
Iter  Time(s)   Res[Rho]   Res[RhoE]  Clift      Cdrag
  20  0.085426  -1.331934  4.097121   0.331850   0.018830
  21  0.085269  -1.364276  4.062276   0.329838   0.019734
  22  0.085333  -1.399437  4.026648   0.327799   0.020649
  23  0.085257  -1.434665  3.991820   0.326023   0.021490
  24  0.085094  -1.467139  3.958620   0.324635   0.022197
  25  0.084995  -1.497680  3.924778   0.323653   0.022728
trent@ubuntu:DIRECT$
```

# Data Exploration

You can also interact with the project through the python interpreter.

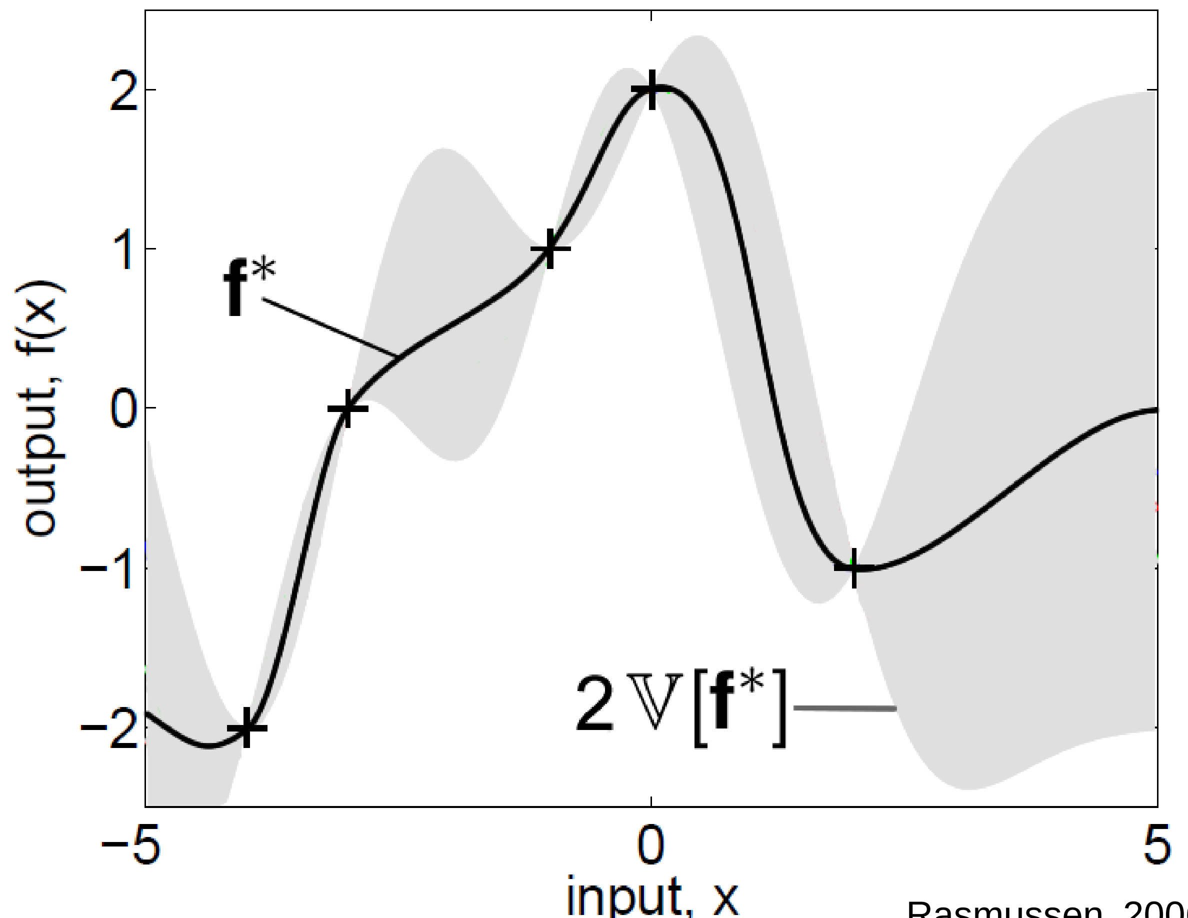
```
trent@ubuntu:NACA_Case$ cd projects/Test_Project/
trent@ubuntu:Test_Project$
trent@ubuntu:Test_Project$ python
Python 2.7.7 |Anaconda 2.0.1 (64-bit)
>>> import SU2
>>>
>>> project = SU2.io.load_data('project.pkl')
>>>
>>> project.results.keys()
['FUNCTIONS', 'GRADIENTS', 'VARIABLES', 'HISTORY']
>>>
>>> project.results.FUNCTIONS.keys()
['LIFT', 'DRAG', 'SIDEFORCE', 'MOMENT_X', 'MOMENT_Y',
'MOMENT_Z', 'FORCE_X', 'FORCE_Y', 'FORCE_Z', 'EFFICIENCY']
```

# Data Exploration

```
>>> print project.results.FUNCTIONS.LIFT
[0.3269416468, 0.2703001641, 0.3528436174, 0.3799183762,
   0.3040688057, 0.3269073365, 0.3203853104]
>>> print project.results.VARIABLES
[[0.0], [0.0089448155956601046],
[-0.0042716224182653443], [-0.0089275210543810577],
[0.0036615329193177178], [6.0326879239180899e-06],
[0.0011200741319185093]]
>>>
>>> project.results.HISTORY.DIRECT.keys()
['ITERATION', 'LIFT', 'DRAG', 'SIDEFORCE', 'MOMENT_X',
'MOMENT_Y', 'MOMENT_Z', 'FORCE_X', 'FORCE_Y', 'FORCE_Z',
'EFFICIENCY', 'Res_Flow[0]', 'Res_Flow[1]', 'Res_Flow[2]',
'Res_Flow[3]', 'Res_Flow[4]', 'Linear_Solver_Iterations',
'TIME']
```

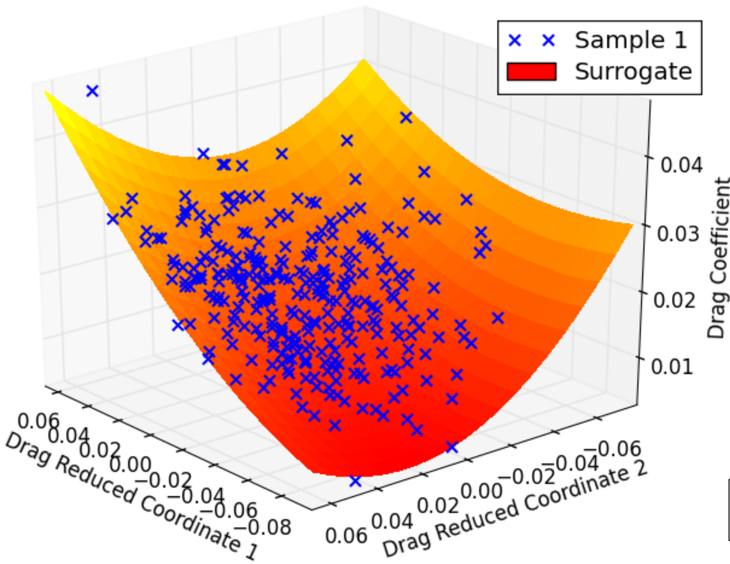
## Surrogate Modeling

With this data we'll build a surrogate model.




Rasmussen, 2006

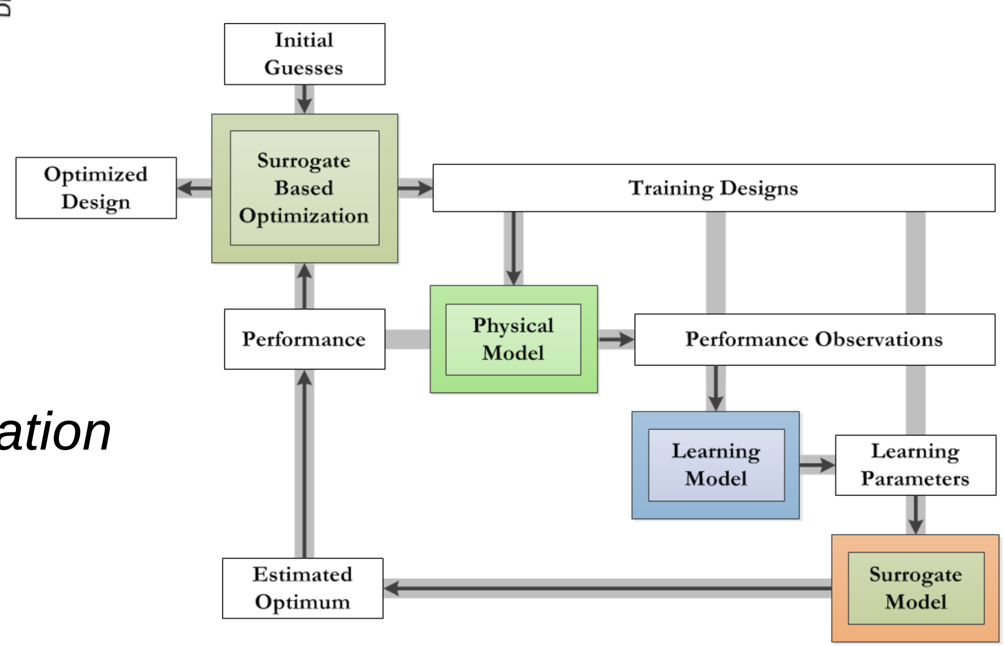
# Surrogate Modeling Tools




# VyPy

github.com/aerialhedgehog/VyPy

Google search: VyPy




*VyPy is a toolbox for optimization and surrogate modeling*

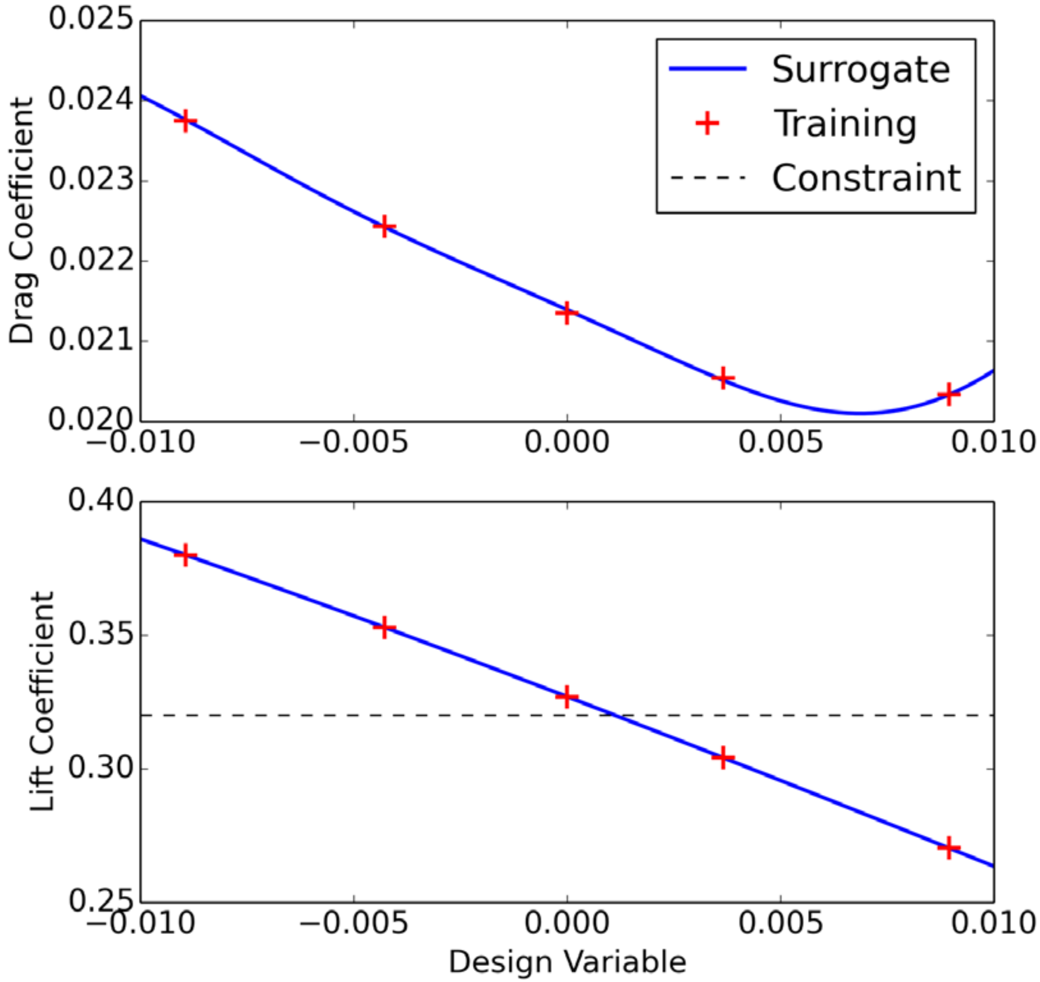
# Learning the Surrogate

These GPR surrogate models are trained on the sampled data locations, functions, and gradients.

```
drag_model = VyPy.regression.gpr.library.Gaussian(
    XS             ,
    F_drags        ,
    DF_drags       ,
    XB             ,
    sig_ny  = -4.0 , # noise guess of the objectives
    sig_ndy = -2.0 , # noise guess of the gradients
)

lift_model = VyPy.regression.gpr.library.Gaussian(
    XS             ,
    F_lifts        ,
    DF_lifts       ,
    XB             ,
    sig_ny  = -4.0 ,
    sig_ndy = -2.0 ,
)
```

# Visualizing the Surrogate Model

In this 1D case we can plot the surrogate model



# Surrogate Based Optimization

We can now interrogate the surrogate model, for example to estimate an optimum design. Your favorite optimization wrappers can work here. This is an example with VyPy's wrappers.

## Problem Setup

```
# the problem
problem = VyPy.optimize.Problem()

# variables
var = VyPy.optimize.Variable()
var.tag      = 'bump'
var.initial = np.array([[0.0] * ND])
var.bounds   = XB.T
var.scale    = 'bounds'
problem.variables.append(var)
```

# Objective and Constraint

```
# evaluator wrappers, VyPy passes dictionaries in and out
eval_drag = lambda (variables): \
     {'drag' : surrogates.drag.predict_YI( variables['bump'] ) }
eval_lift = lambda (variables): \
     {'lift' : surrogates.lift.predict_YI( variables['bump'] ) }

# the objective, drag
obj = VyPy.optimize.Objective()
obj.evaluator = eval_drag
obj.tag = 'drag'
obj.scale = 0.01
problem.objectives.append(obj)

# the constraint, lift > 0.3200
con = VyPy.optimize.Constraint()
con.evaluator = eval_lift
con.tag = 'lift'
con.edge = 0.3200
con.sense = '>'
con.scale = 1.0
problem.constraints.append(con)
```

# Optimization Wrapper

```
print "Local Optimization (SLSQP)"
driver = VyPy.optimize.drivers.scipy.SLSQP()
driver.verbose = False
result = driver.run(problem)
```

# Results

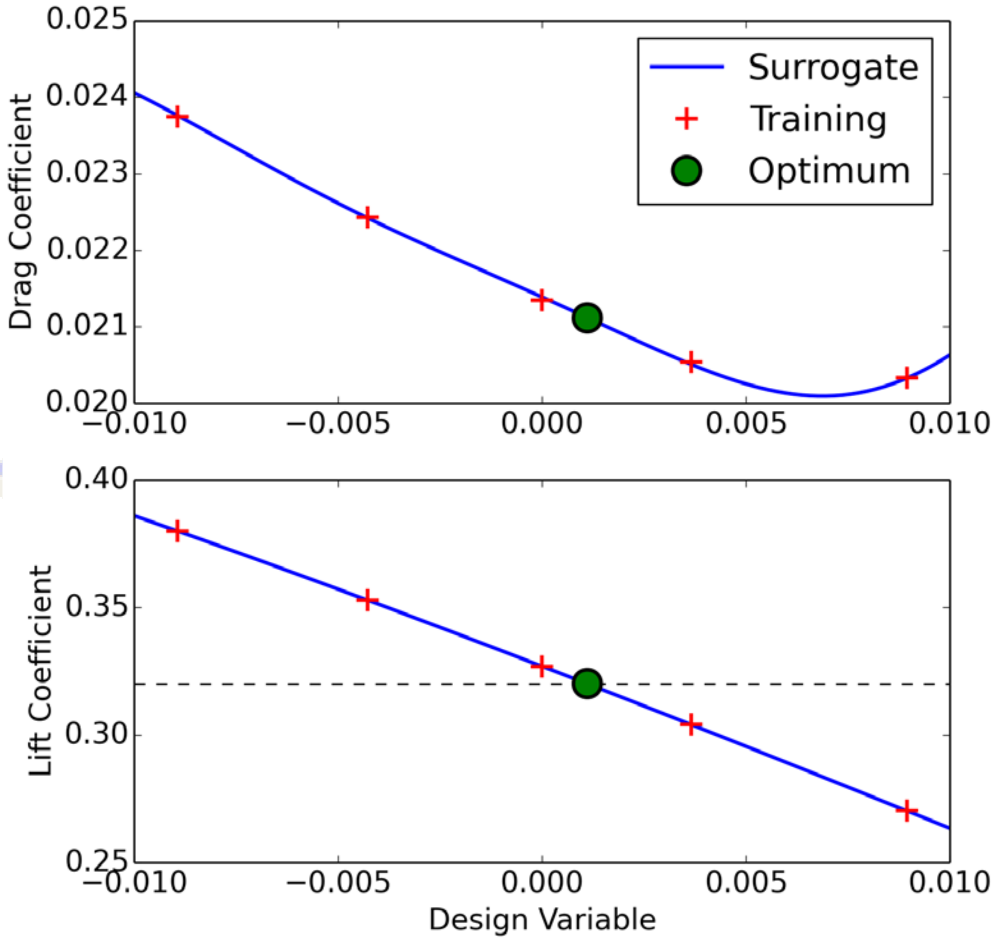
**Surrogate Optimum Prediction**

| | |
|---|---|
| Drag Coefficient | 0.0211 |
| Lift Coefficient | 0.3200 |

**SU2 Evaluation Check**, of the predicted design

| | |
|---|---|
| Drag Coefficient | 0.0211 |
| Lift Coefficient | 0.3204 |

# Optimization Results



# Topics Covered

- Scripting SU2
- Saving and loading data
- Interacting with the python interpreter
- Running a sample of experiments
- Surrogate based optimization