

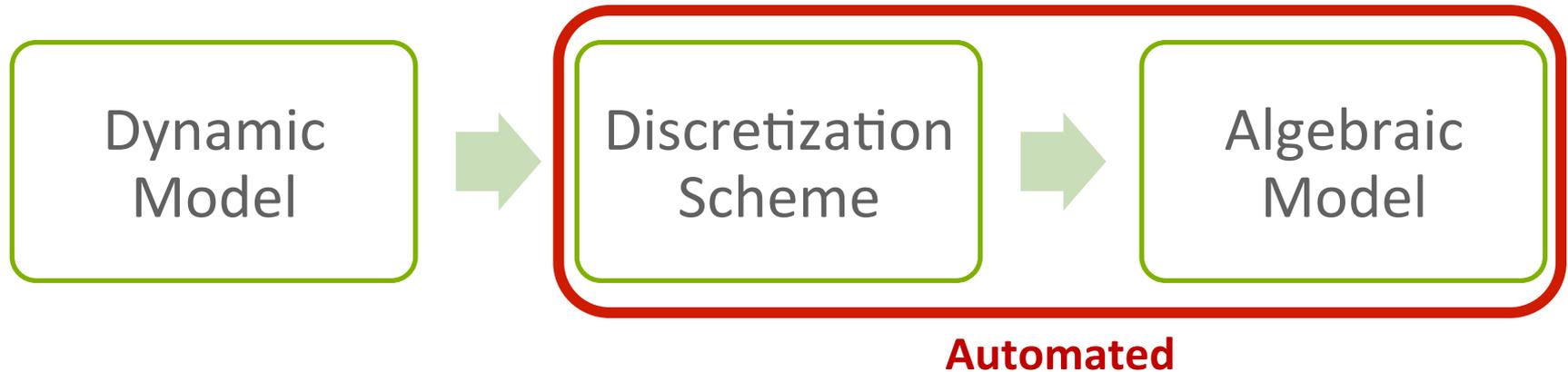
Automatic Discretization of ODE and PDE Systems Using Pyomo

Bethany Nicholson ~ Carnegie Mellon University

Victor Zavala ~ Argonne National Laboratory

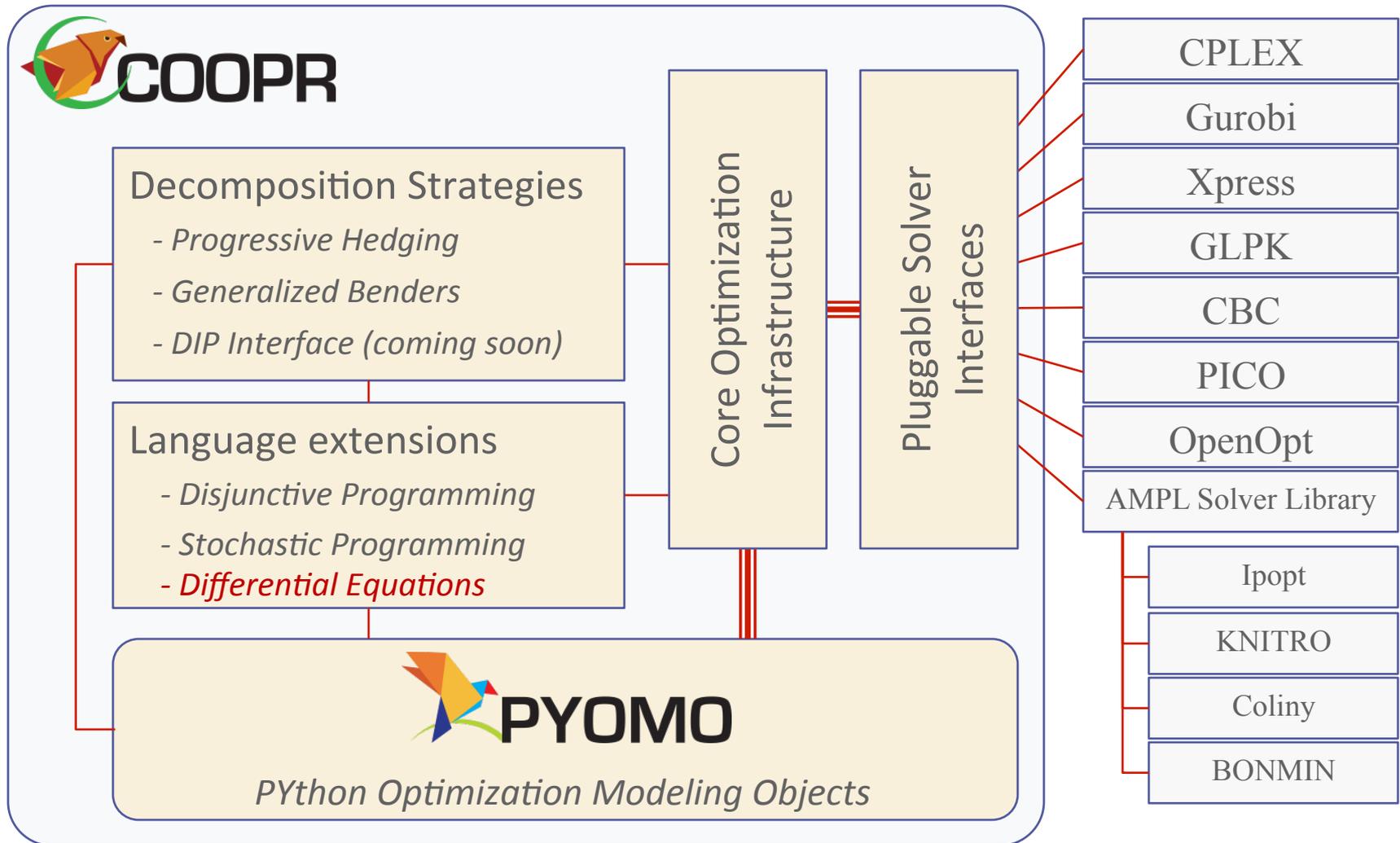
August 4, 2014

Automatic Discretization



- Popular Algebraic Modeling Languages
e.g., GAMS, AMPL, MOSEL
 - Can't represent differential equations
 - Syntax isn't easily extended
 - Limited scripting capability

Coopr: a COmmon Optimization Python Repository



Pyomo Overview



- Formulating optimization models natively within Python
 - Provide a natural syntax to describe mathematical models
 - Formulate large models with a concise syntax
- Highlights:
 - Python scripts provide a flexible context for exploring the structure of Pyomo models
 - Leverage high-quality third-party Python libraries, e.g., SciPy, NumPy, Matplotlib

```
from coopr.pyomo import *
m = ConcreteModel()
m.x1 = Var()
m.x2 = Var(bounds=(-1,1))
m.x3 = Var(bounds=(1,2))
m.obj = Objective(
    sense = minimize,
    expr = m.x1**2 + (m.x2*m.x3)**4 +
           m.x1*m.x3 + m.x2 +
           m.x2*sin(m.x1+m.x3) )
```

New Coopr Package: coopr.dae

- Extend Pyomo object model
 - ContinuousSet
 - StateVar
 - DerivativeVar
- General model transformations
 - Standardized framework for transforming dynamic system to (N)LP
 - Finite Difference Methods
 - Backward, Forward, and Central
 - Orthogonal Collocation
 - Radau and Legendre roots

PDE Example

- Matlab example problem

- PDE

$$\pi^2 \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right)$$

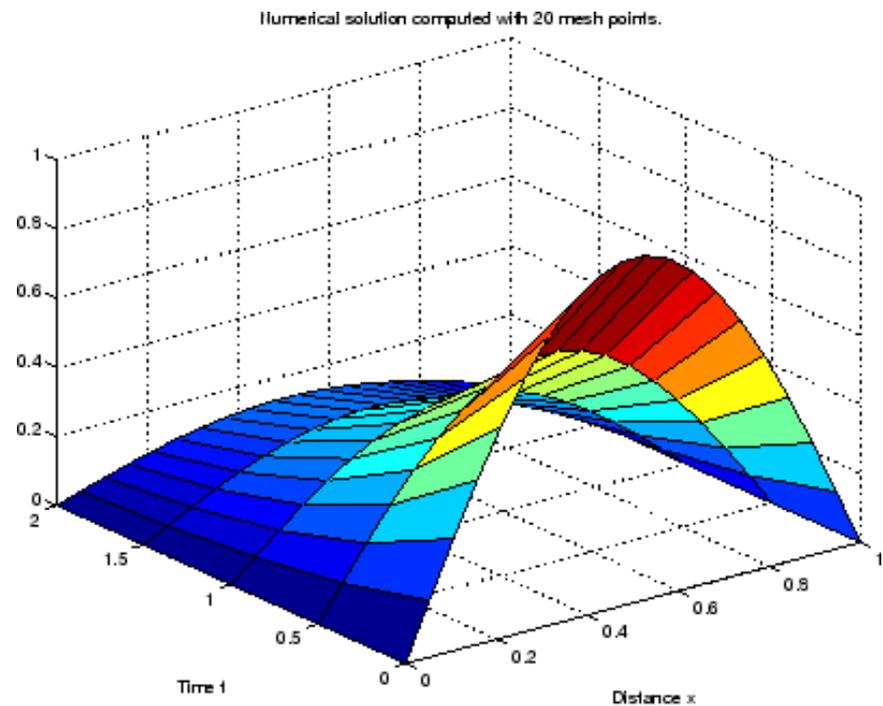
- Initial Condition

$$u(x,0) = \sin(\pi x)$$

- Boundary Conditions

$$u(0,t) = 0$$

$$\pi e^{-t} + \frac{\partial u}{\partial x}(1,t) = 0$$



PDE Example

```
from coopr.environ import *
from coopr.pyomo import *
from coopr.dae import *
from coopr.opt import SolverFactory
from coopr.dae.plugins.finitedifference import Finite_Difference_Transformation
from coopr.dae.plugins.colloc import Collocation_Discretization_Transformation
import math
```

```
m = ConcreteModel()
m.t = ContinuousSet(bounds=(0,2))
m.x = ContinuousSet(bounds=(0,1))
m.u = StateVar(m.x,m.t)

m.dudx = DerivativeVar(m.u,wrt=m.x)
m.dudx2 = DerivativeVar(m.u,wrt=(m.x,m.x))
m.dudt = DerivativeVar(m.u,wrt=m.t)
```

```
def _pde(m,i,j):
    if i == 0 or i == 1 or j == 0 :
        return Constraint.Skip
    return math.pi**2*m.dudt[i,j] == m.dudx2[i,j]
m.pde = Constraint(m.x,m.t,rule=_pde)

def _initcon(m,i):
    if i == 0 or i == 1:
        return Constraint.Skip
    return m.u[i,0] == sin(math.pi*i)
m.initcon = Constraint(m.x,rule=_initcon)

def _lowerbound(m,j):
    return m.u[0,j] == 0
m.lowerbound = Constraint(m.t,rule=_lowerbound)

def _upperbound(m,j):
    return math.pi*exp(-j)+m.dudx[1,j] == 0
m.upperbound = Constraint(m.t,rule=_upperbound)

m.obj = Objective(expr=1)
```

$$\pi^2 \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right)$$

$$u(x,0) = \sin(\pi x)$$

$$u(0,t) = 0$$

$$\pi e^{-t} + \frac{\partial u}{\partial x}(1,t) = 0$$

PDE Example

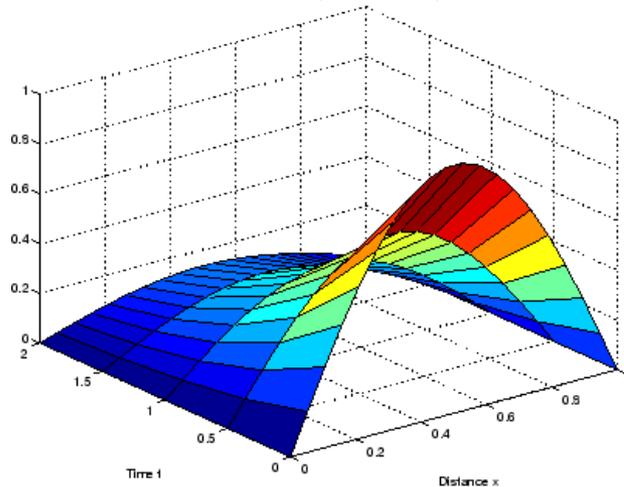
```
# Discretize using Finite Difference Method
discretize = Finite_Difference_Transformation()
disc = discretize.apply(m,nfe=25,wrt=m.x,scheme='BACKWARD')
disc = discretize.apply(disc,nfe=20,wrt=m.t,scheme='BACKWARD',clonemodel=False)

# Discretize using Orthogonal Collocation
#discretize2 = Collocation_Discretization_Transformation()
#disc = discretize2.apply(disc,nfe=10,ncp=3,wrt=m.x)
#disc = discretize2.apply(disc,nfe=20,ncp=3,wrt=m.t,clonemodel=False)

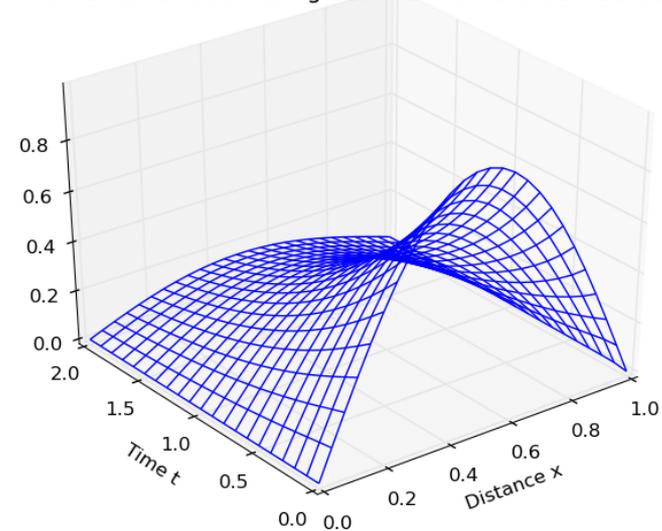
solver='ipopt'
opt=SolverFactory(solver)

results = opt.solve(disc,tee=True)
disc.load(results)
```

Numerical solution computed with 20 mesh points.



Numerical Solution Using Backward Difference Method



Other Work

- Additional Examples
 - Optimal Control
 - Parameter Estimation
 - Heat transfer in a building
 - Gas network
 - Distillation Column
- Integrals

Summary

- Created a flexible and concise way of representing arbitrary ordinary and partial differential equations
- Implemented several discretization schemes and developed a framework that is extensible to include others
- Future work
 - Finish implementing Integrals
 - Additional discretization schemes
 - Link coopr/pyomo to an integrator for doing model simulation or initialization
 - Develop frameworks for multigrid (multiscale) methods

Questions?

- Additional information:

<https://software.sandia.gov/trac/coopr>

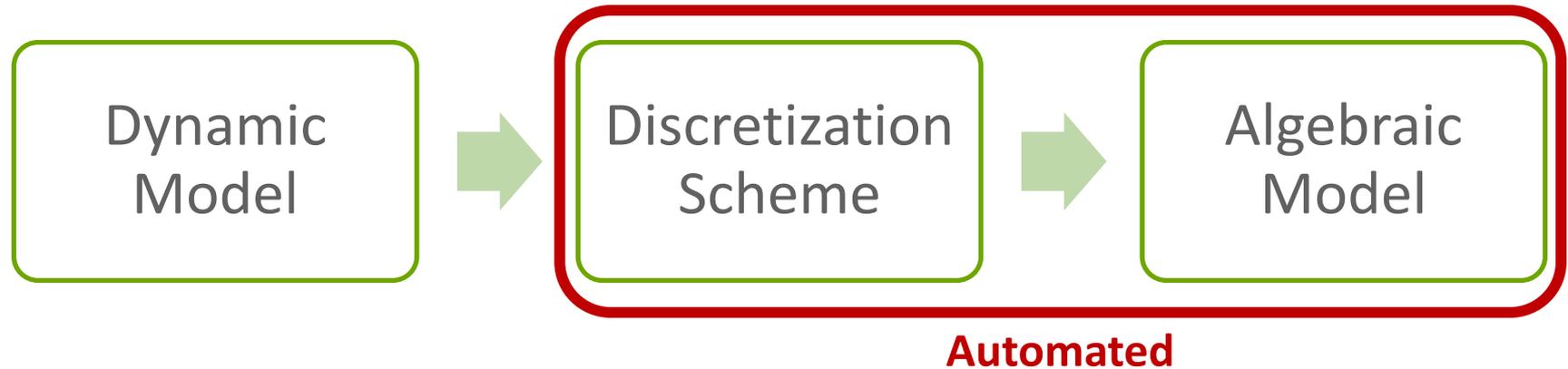
Automatic Discretization of ODE and PDE Systems Using Pyomo

Bethany Nicholson ~ Carnegie Mellon University

Victor Zavala ~ Argonne National Laboratory

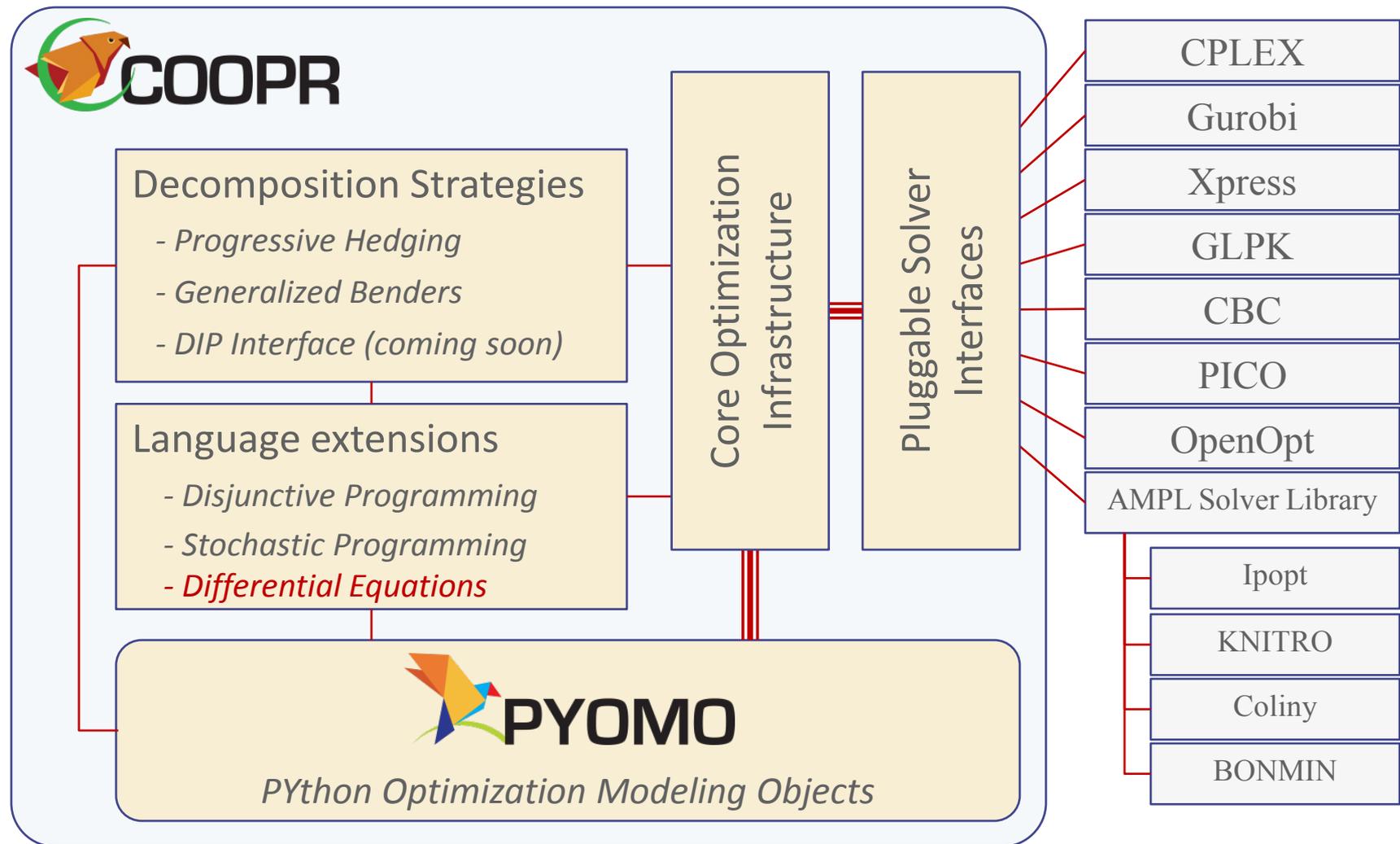
August 4, 2014

Automatic Discretization



- Popular Algebraic Modeling Languages
e.g., GAMS, AMPL, MOSEL
 - Can't represent differential equations
 - Syntax isn't easily extended
 - Limited scripting capability

Coopr: a COmmon Optimization Python Repository



Pyomo Overview



- Formulating optimization models natively within Python
 - Provide a natural syntax to describe mathematical models
 - Formulate large models with a concise syntax
- Highlights:
 - Python scripts provide a flexible context for exploring the structure of Pyomo models
 - Leverage high-quality third-party Python libraries, e.g., SciPy, NumPy, Matplotlib

```
from coopr.pyomo import *
m = ConcreteModel()
m.x1 = Var()
m.x2 = Var(bounds=(-1,1))
m.x3 = Var(bounds=(1,2))

m.obj = Objective(
    sense = minimize,
    expr = m.x1**2 + (m.x2*m.x3)**4 +
           m.x1*m.x3 + m.x2 +
           m.x2*sin(m.x1+m.x3) )
```



New Coopr Package: coopr.dae

- Extend Pyomo object model
 - ContinuousSet
 - StateVar
 - DerivativeVar

- General model transformations
 - Standardized framework for transforming dynamic system to (N)LP
 - Finite Difference Methods
 - Backward, Forward, and Central
 - Orthogonal Collocation
 - Radau and Legendre roots

PDE Example

- Matlab example problem

- PDE

$$\pi^2 \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right)$$

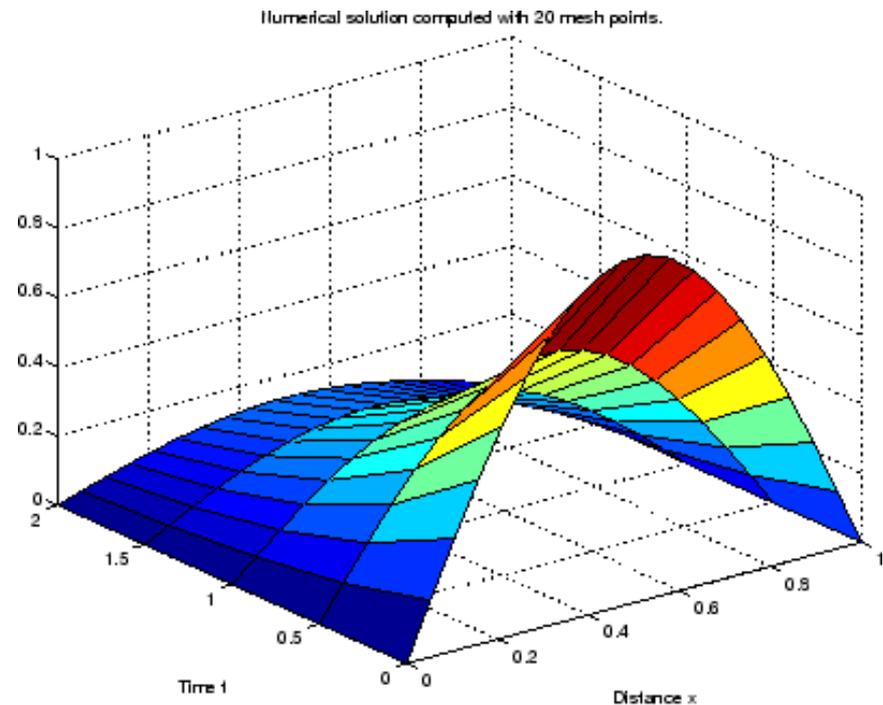
- Initial Condition

$$u(x, 0) = \sin(\pi x)$$

- Boundary Conditions

$$u(0, t) = 0$$

$$\pi e^{-t} + \frac{\partial u}{\partial x}(1, t) = 0$$



PDE Example

```
from coopr.environ import *
from coopr.pyomo import *
from coopr.dae import *
from coopr.opt import SolverFactory
from coopr.dae.plugins.finitedifference import Finite_Difference_Transformation
from coopr.dae.plugins.colloc import Collocation_Discretization_Transformation
import math
```

```
m = ConcreteModel()
m.t = ContinuousSet(bounds=(0,2))
m.x = ContinuousSet(bounds=(0,1))
m.u = StateVar(m.x,m.t)

m.dudx = DerivativeVar(m.u,wrt=m.x)
m.dudx2 = DerivativeVar(m.u,wrt=(m.x,m.x))
m.dudt = DerivativeVar(m.u,wrt=m.t)
```

$$\pi^2 \frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right)$$

$$u(x, 0) = \sin(\pi x)$$

$$u(0, t) = 0$$

$$\pi e^{-t} + \frac{\partial u}{\partial x}(1, t) = 0$$

```
def _pde(m,i,j):
    if i == 0 or i == 1 or j == 0 :
        return Constraint.Skip
    return math.pi**2*m.dudt[i,j] == m.dudx2[i,j]
m.pde = Constraint(m.x,m.t,rule=_pde)
```

```
def _initcon(m,i):
    if i == 0 or i == 1:
        return Constraint.Skip
    return m.u[i,0] == sin(math.pi*i)
m.initcon = Constraint(m.x,rule=_initcon)
```

```
def _lowerbound(m,j):
    return m.u[0,j] == 0
m.lowerbound = Constraint(m.t,rule=_lowerbound)
```

```
def _upperbound(m,j):
    return math.pi*exp(-j)+m.dudx[1,j] == 0
m.upperbound = Constraint(m.t,rule=_upperbound)
```

```
m.obj = Objective(expr=1)
```

PDE Example

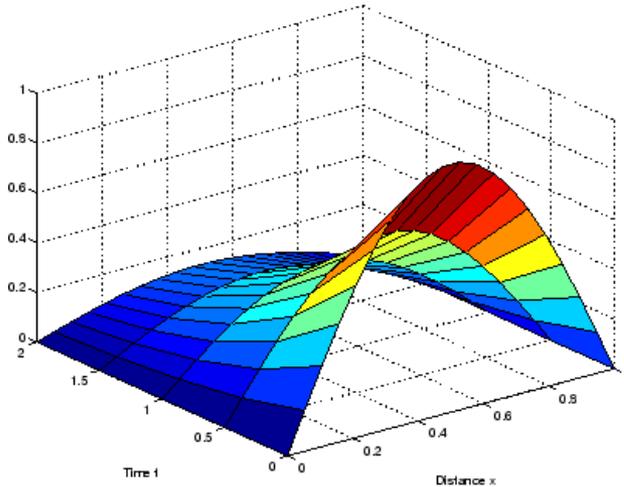
```
# Discretize using Finite Difference Method
discretize = Finite_Difference_Transformation()
disc = discretize.apply(m,nfe=25,wrt=m.x,scheme='BACKWARD')
disc = discretize.apply(disc,nfe=20,wrt=m.t,scheme='BACKWARD',clonemodel=False)

# Discretize using Orthogonal Collocation
#discretize2 = Collocation_Discretization_Transformation()
#disc = discretize2.apply(disc,nfe=10,ncp=3,wrt=m.x)
#disc = discretize2.apply(disc,nfe=20,ncp=3,wrt=m.t,clonemodel=False)

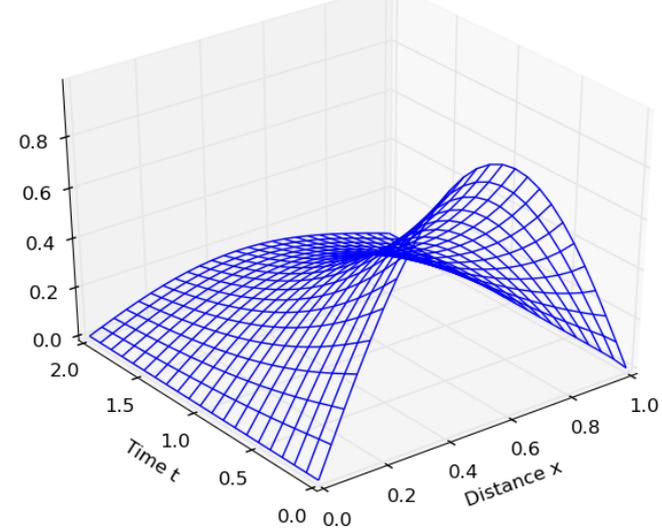
solver='ipopt'
opt=SolverFactory(solver)

results = opt.solve(disc,tee=True)
disc.load(results)
```

Numerical solution computed with 20 mesh points.



Numerical Solution Using Backward Difference Method



Other Work

- Additional Examples
 - Optimal Control
 - Parameter Estimation
 - Heat transfer in a building
 - Gas network
 - Distillation Column
- Integrals

Summary

- Created a flexible and concise way of representing arbitrary ordinary and partial differential equations
- Implemented several discretization schemes and developed a framework that is extensible to include others
- Future work
 - Finish implementing Integrals
 - Additional discretization schemes
 - Link coopr/pyomo to an integrator for doing model simulation or initialization
 - Develop frameworks for multigrid (multiscale) methods

Questions?

- Additional information:

<https://software.sandia.gov/trac/coopr>