ADiCape
in a large-scale industrial problem

Monika Petera, **Martin Bücker, Arno Rasch**
Institute for Scientific Computing
RWTH Aachen University

Optimization Problem

The Model:

A system of differential and algebraic equations (DAE):

$$M\dot{x} = F(x(p,t),t)$$

where:

x - set of state variables

p - subset of parameters from x

Goal:

To minimize the objective function: $\min_{p} \Phi(x(p,t), p, t_f)$

at the final time t_f

Modeling languages

A model can be written in a one of modeling languages, e.g. gPROMS, Modelica**,** and is usually represented as a system of **mathematical equations**

An equation oriented approach does not make any assumptions about how to solve a model or what quantities are considered known and unknown.

CapeML – A Common Model Exchange Language for Chemical Process Modeling designed for supporting the modeling process.

CapeML is a neutral model exchange representation based on the XML standard.

"CapeML – A Model Exchange Language for Chemical Process Modeling" - L.v.Wedel (TR May 2002)

A fragment of CapeML code

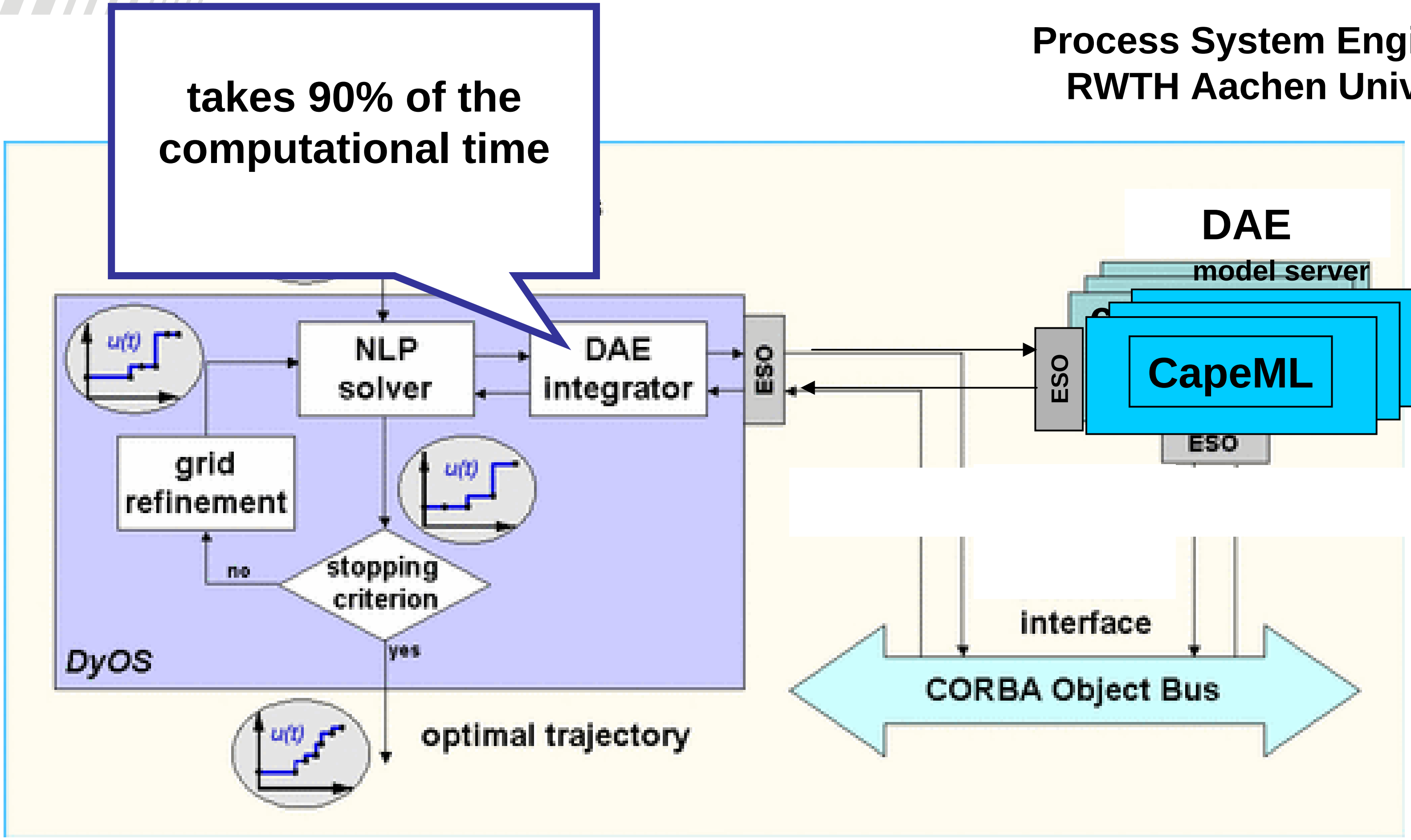
```
<Equation>
<BalancedEquation myID="V-0">
 <Expression>
  <Term>
   <Factor>
    <FunctionCall fcn.name=“sin">
     <Expression>
      <Term>
       <Factor>
        <VariableOccurrence
                definition="V-car-alpha“/>
       </Factor>
      </Term>
     </Expression>
    </FunctionCall>
   </Factor>
  </Term>
 </Expression>
```

$$\sin(alpha) = beta$$

```
<Expression>
 <Term>
  <Factor>
   <VariableOccurrence
             definition="V-car-beta“/>
  </Factor>
 </Term>
 </Expression>
</BalancedEquation>
</Equation>
```

DyOS- Dynamic Optimization Software

Process System Engineering
RWTH Aachen University



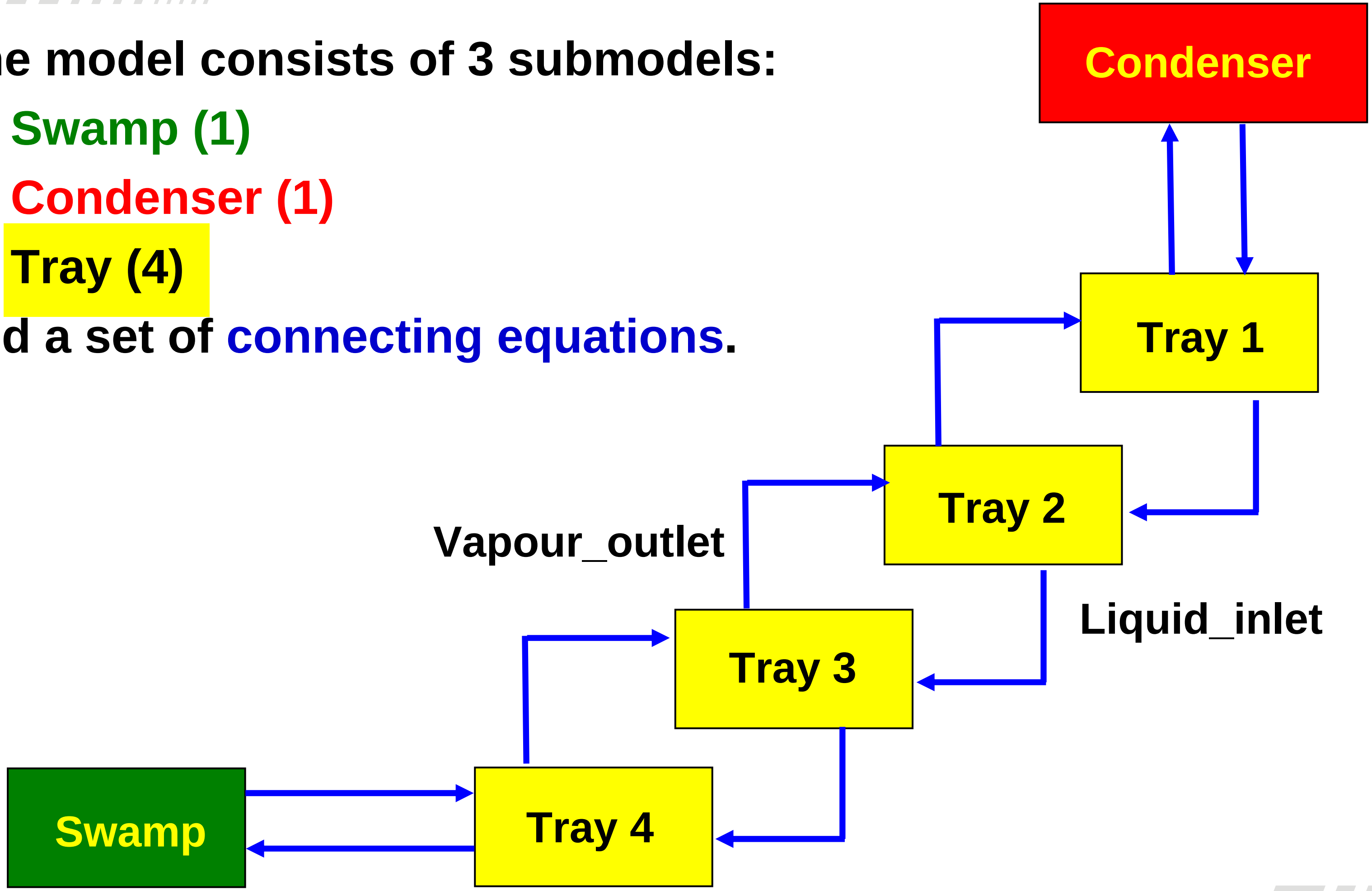
ESO (Equation Set Object) - an internal representation of the computation tree.

The model of the Distillation Column

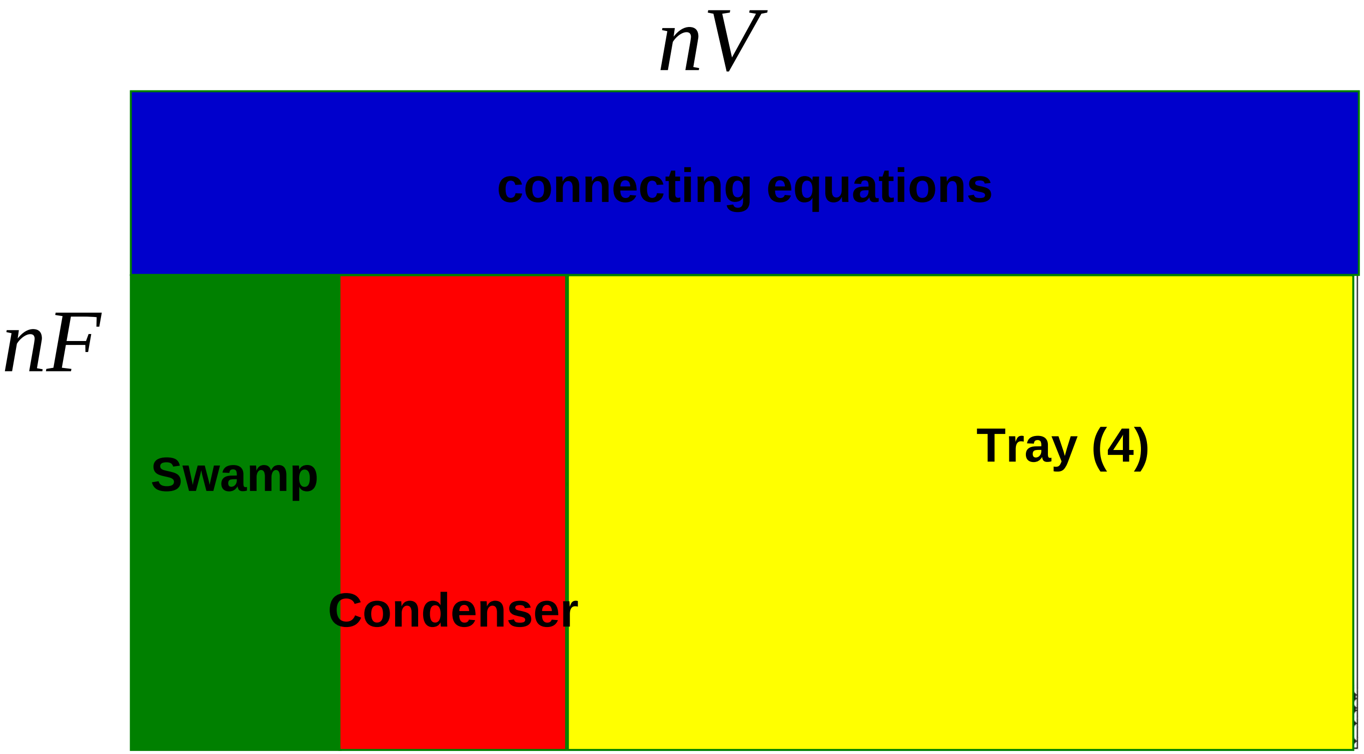
The model consists of 3 submodels:

- Swamp (1)
- Condenser (1)
- Tray (4)

and a set of connecting equations.



Jacobian sparsity structure



 # equations

$$nF = 206$$

 # variables

$$nV = 410$$

 # nonzeros

$$nnz = 919 \approx 1.1\%$$

Newton Step

state variables:

$$x_{k+1}(p) = x_k(p) - (LU)^{-1}\ F(x_k(p))$$

$$(LU)\cdot \Delta x_{k+1}(p) = F(x_k(p))$$

$$\Big\downarrow\ \frac{d}{dp}$$

derivatives:

$$(LU)\cdot \Delta s_{k+1}(p) = \frac{\partial F}{\partial x_k(p)} \frac{\partial x_k(p)}{\partial p}$$

$$s_{k+1}(p) = s_k(p) - (LU)^{-1}\left(\frac{\partial F}{\partial x_k(p)} \cdot s_k(p) \right)$$

LU- scaled system Jacobian matrix

$$LU = \frac{\partial F(x_0(p,t))}{\partial x} - \frac{M}{h_j}$$

$$\frac{d}{dp} x_k(p) = s_k(p)$$

Integration Algorithm

Process System Engineering
RWTH Aachen University

Compute $A_0 = \frac{\partial}{\partial x}(F(x_0, p))$

for j=1,...,j_{max} while convergence criterion not satisfied

$$h_j = H / n_j$$

$$LU = A_0 - \frac{M}{h_j}$$

for k=0,...,j-1

$$x_{k+1} = x_k - (LU)^{-1} F(x_k(p))$$

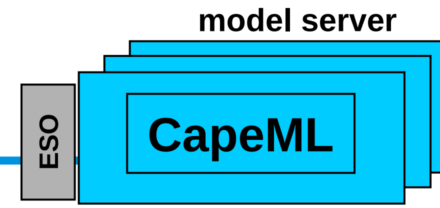
$$s_{k+1} = s_k - (LU)^{-1}\left(\frac{dF(x_k(p))}{dx_k} \cdot s_k\right)$$

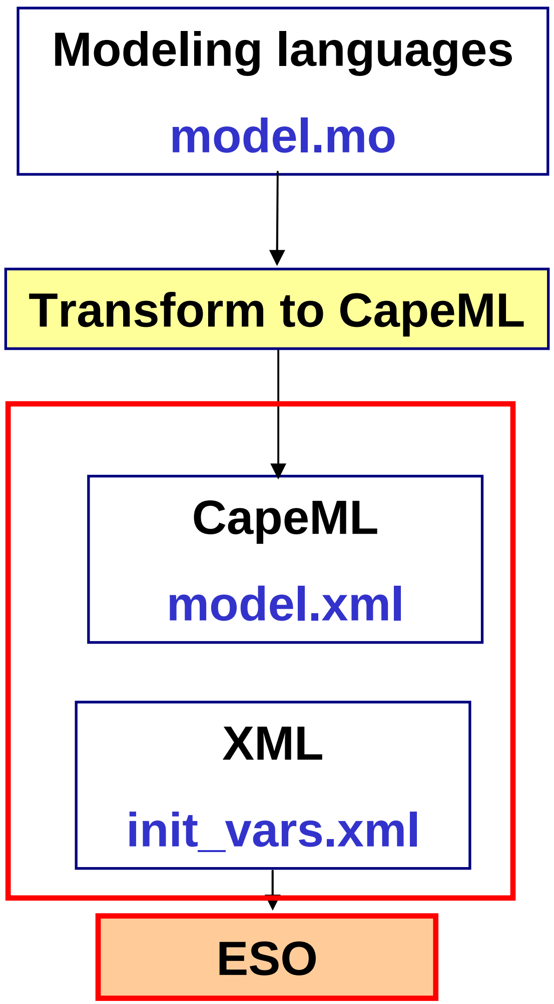
Reuse LU decomposition

AD seeding with S_k

M. Schlegel and W. Marquardt and R. Ehrig and U. Nowak: "Sensitivity Analysis of Linearly-implicit Differential-algebraic Systems by One-step Extrapolation" 2004

ESO of DAE System

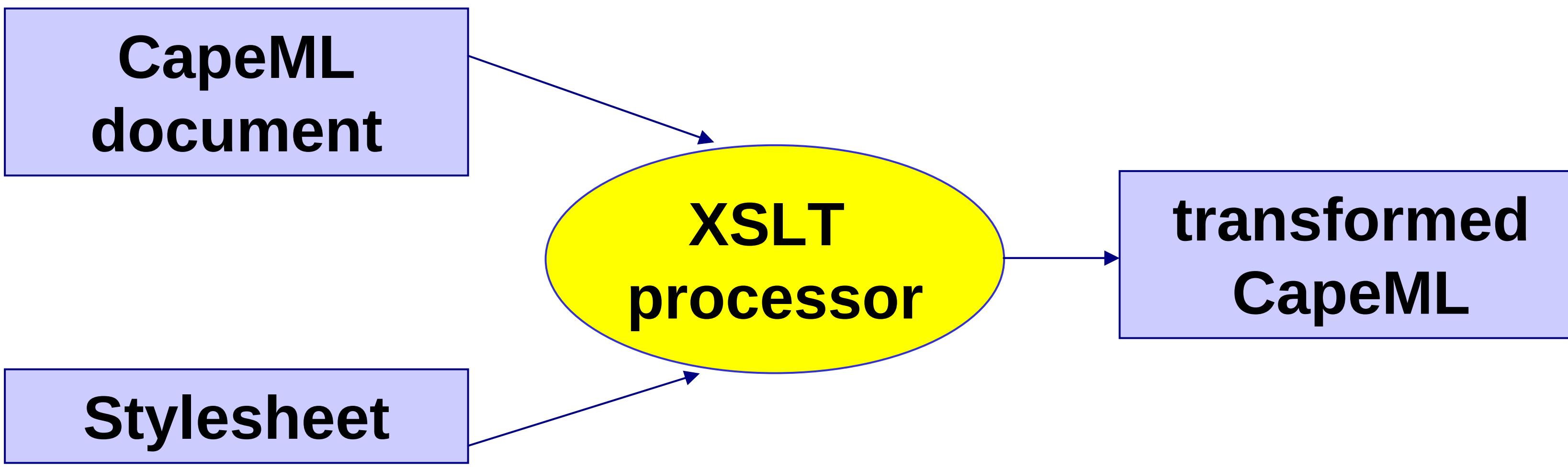




XSLT – Transform language

ADiCape is based on XSLT Transformation Language.

On any XML-based language document eg. in CapeML, an **XSLT stylesheet** is applied, to generate new document.



ADiCape consists of 2 XSLT Stylesheets (at the moment)

ADiCape_var.xsl

XML

init_vars.xml

$$V = \begin{bmatrix} v_1 \\ \dots \\ v_{22} = Temp \\ \dots \\ v_{410} \end{bmatrix}$$

gradients for the dF/dTemp

$$\nabla Temp = \begin{bmatrix} g\_Temp[1] = 0 \\ \dots \\ g\_Temp[22] = 1 \\ \dots \\ g\_Temp[410] = 0 \end{bmatrix}$$

name="S.Temp"

<**Variable** myID="V_m_S.Temp" **name="S.Temp"** varID="V_SWAMP_Temp" view="ovar">

<Experiment number="1" initial_value="85" result_value="0"/>

ADiCape_var.xsl

name="S.g_Temp"

dimension="410"

index="22"

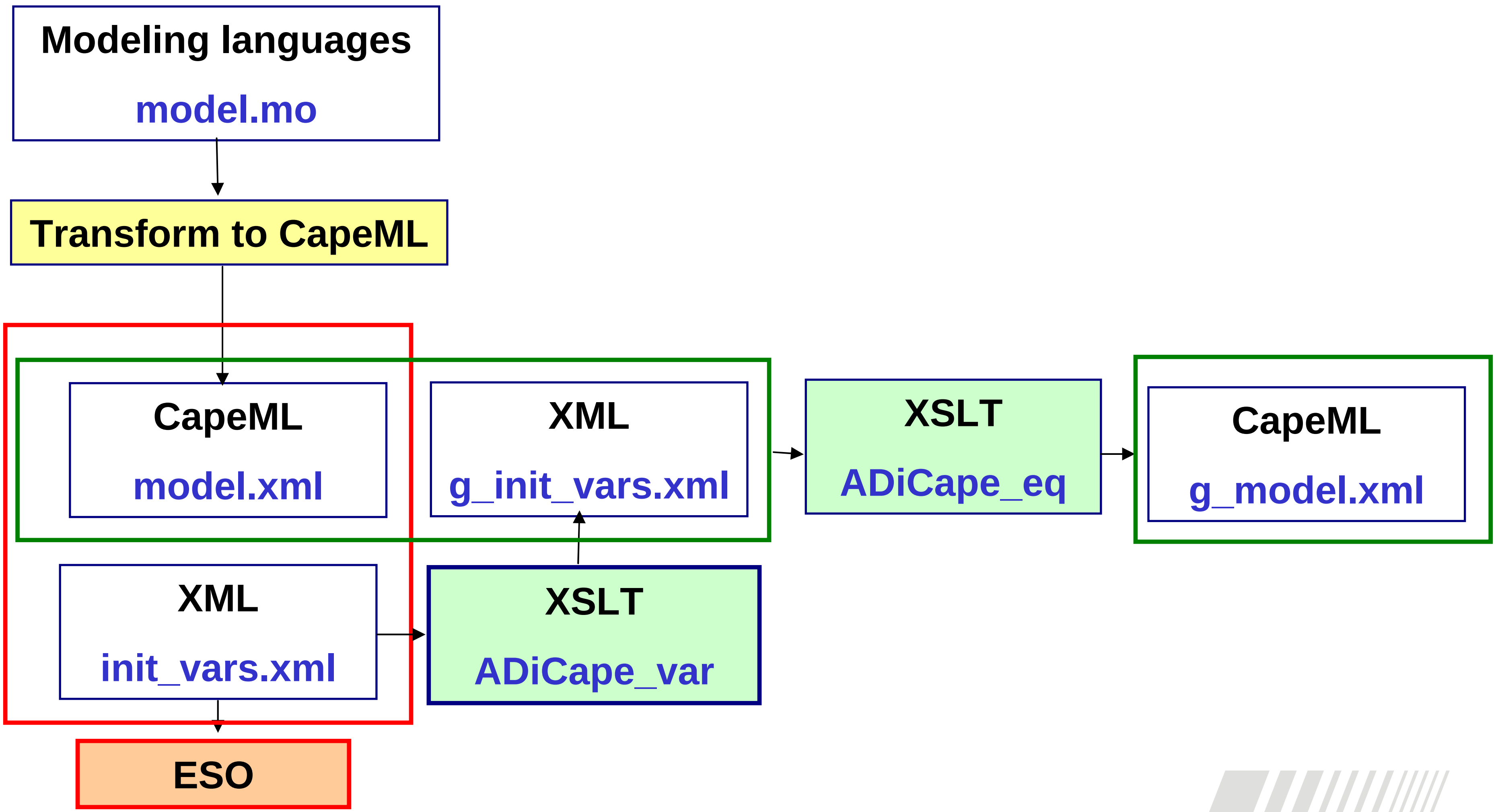
initial_value="1"

<**Vector** **name="S.g_Tem** ... **dimen** ...

<**VectorVariable** myID="V ... emp[22]" **index="22"**>

<Experiment number="1" **initial_value="1"** result_value="0"/>

ADiCape


Modeling languages
model.mo
Transform to CapeML
CapeML
model.xml
XML
g_init_vars.xml
XSLT
ADiCape_eq
CapeML
g_model.xml
XML
init_vars.xml
XSLT
ADiCape_var
ESO

ADiCape_eq.xsl

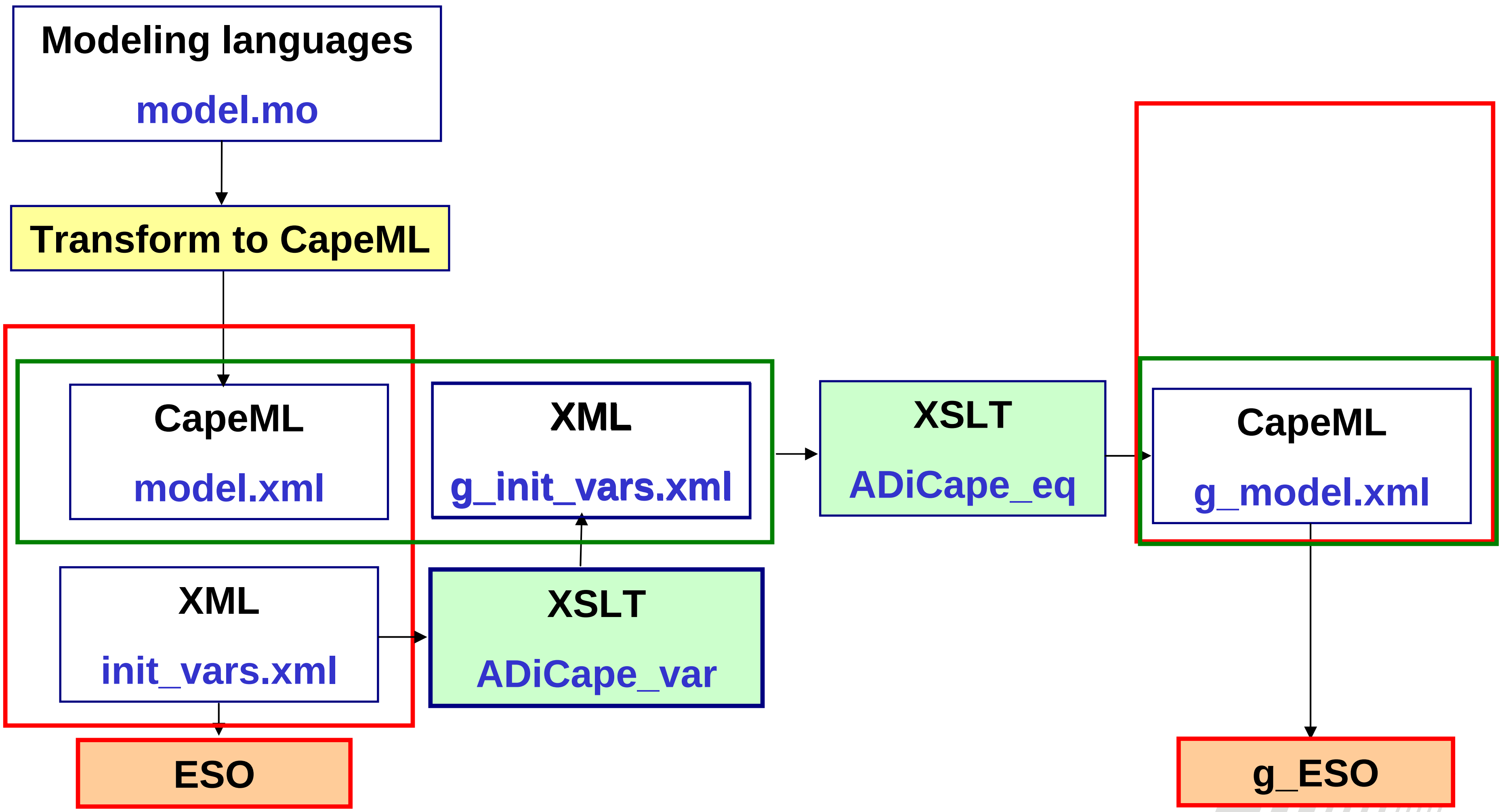
Templates:

```
<xsl:template match="Equation">
<xsl:template match="Expression">
<xsl:template match="FunctionCall">
<xsl:template match="VariableOccurence">
```

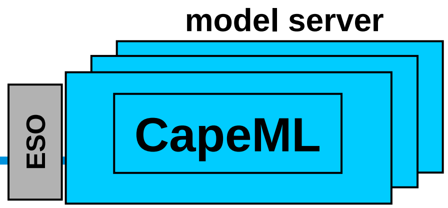
```
<Equation>
<BalancedEquation myID="V-0">
 <Expression>
  <Term>
   <Factor>
    <FunctionCall fcn.name="sin">
     <Expression>
      <Term>
       <Factor>
        <VariableOccurrence
                  definition="V-car-alpha"/>
       </Factor>
      </Term>
     </Expression>
    </FunctionCall>
   </Factor>
  </Term>
 </Expression>
...
```

```
<Equation>
<BalancedEquation myID="V-0">
                  <Distribution domain="V-g_i"/>
 <Expression>
  <Term>
   <Factor>
    <FunctionCall fcn.name="cos">
     <Expression>
      <Term>
       <Factor>
        <VariableOccurrence
                  definition="V-car-alpha"/>
       </Factor>
      </Term>
     </Expression>
    </FunctionCall>
   </Factor>
   <Factor mul.op="MUL">
      <VariableOccurrence
                  definition="V-car-g_alpha"/>
               <DomainOccurrence domain="V-g_i"/>
   </Factor>
  </Term>
 </Expression>
...
```

ESO for the differentiated DAE

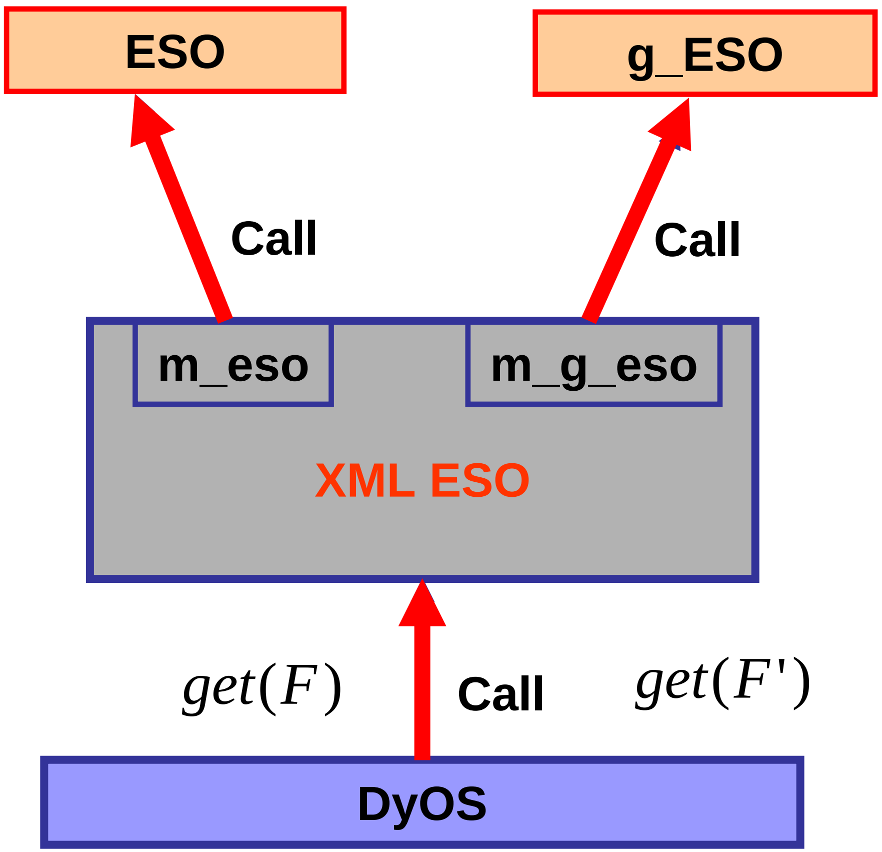


XML ESO



original DAE

First derivatives



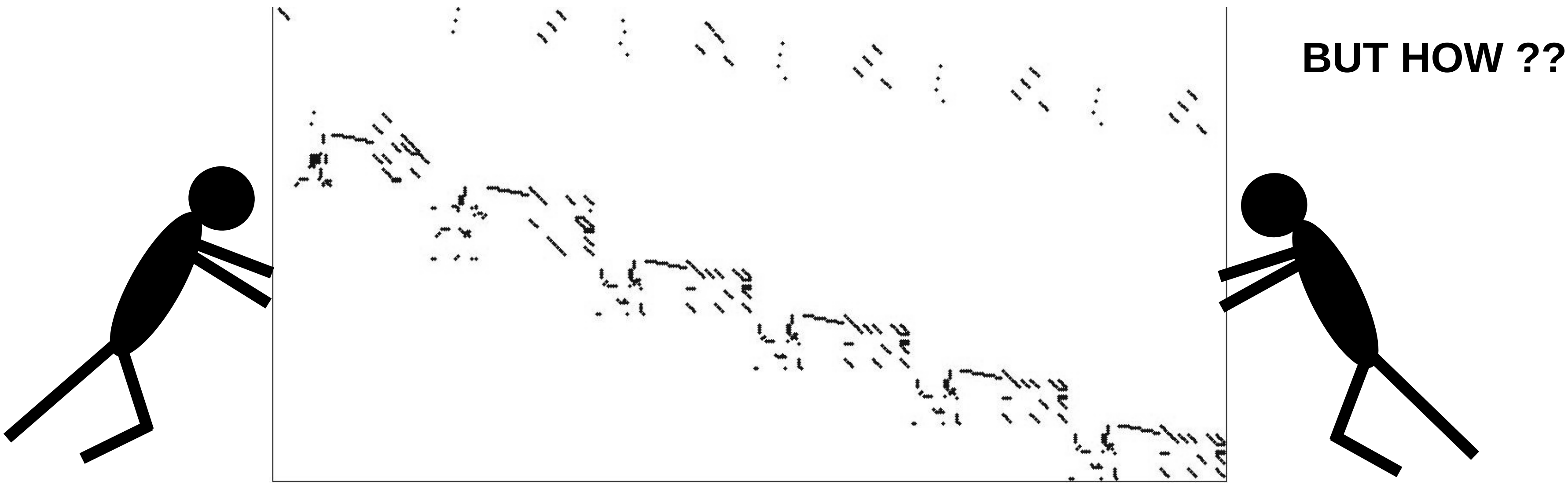
ADiCape:

Full Jacobian
Jacobian times Matrix

Sparse Jacobian matrices with known sparsity pattern:

- do not calculate “zeros”
- cut down number of operations
- save space

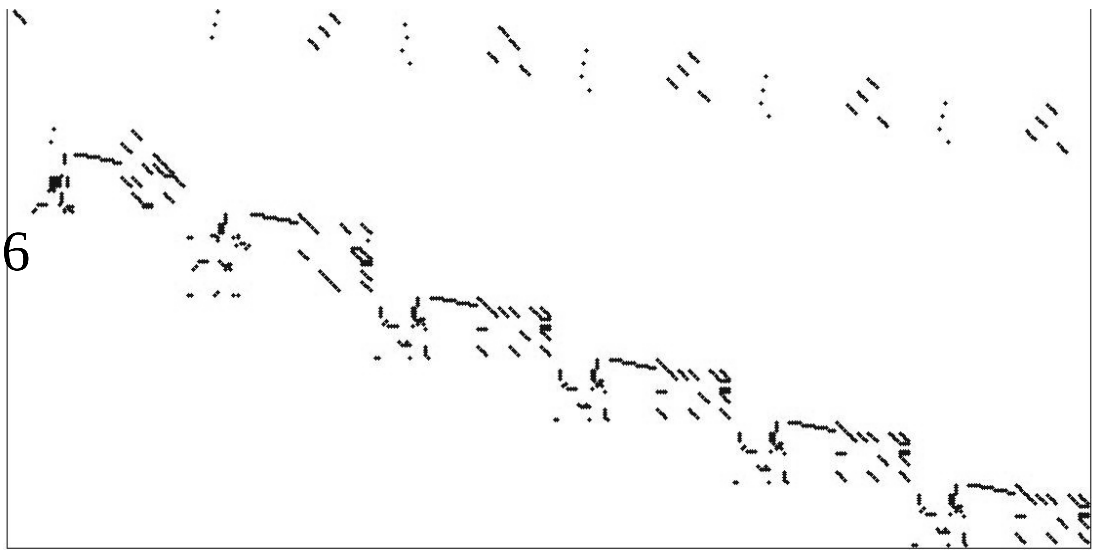
Matrix compression



Sparsity aware seeding with CPR – Curtis-Powell-Reid graph-coloring method!

Sparsity aware seeding

206



410

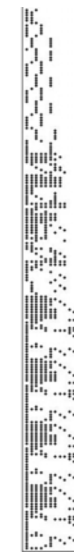
$$\frac{dF}{dx}$$

*

$seed$

13

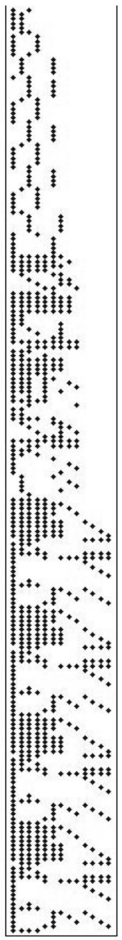
=

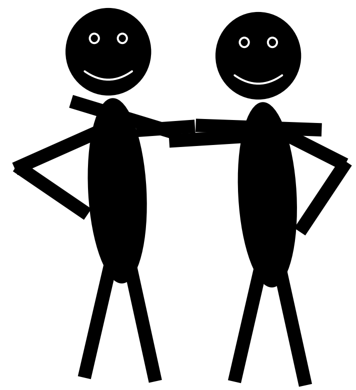


13

$$\frac{dF}{dx} \cdot seed$$

Compressed Full Jacobian





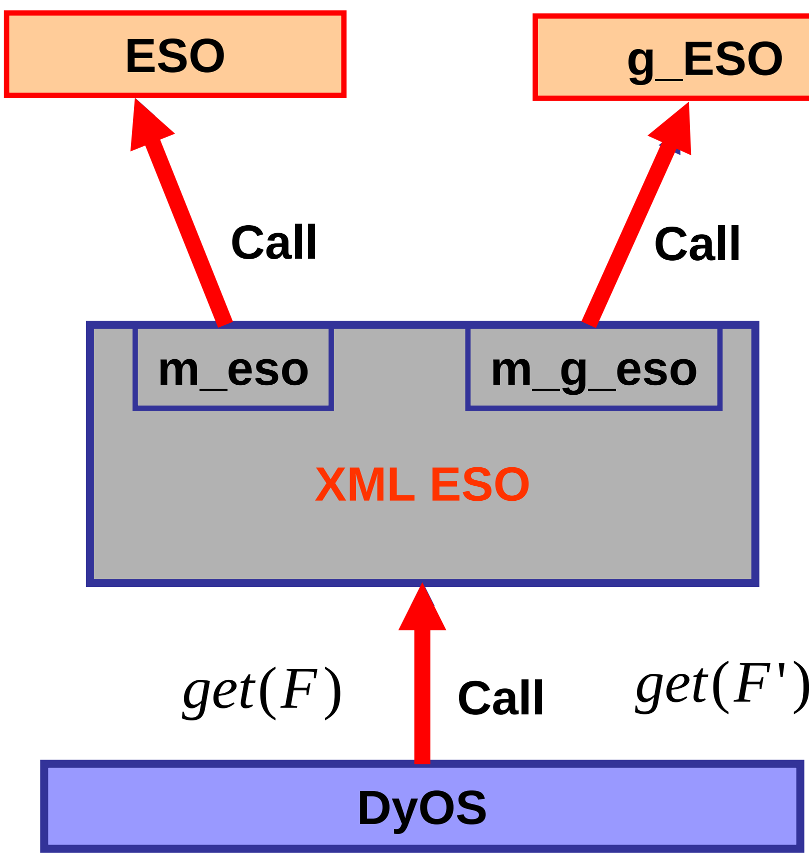
	size	time
full	206x410	55 min
compressed	206x13	3.4 s
factor	31.5	970 ?!

2.8 GHz Pentium 4
512MB cash

$$(31.5^2 \approx 970)$$

Future Work

original DAE | First derivatives | Second derivatives



h_ESO

change DyOS System:

- new Optimizer (IPOPT)* which uses 2nd derivatives
- change Integration Algorithm

* A. Wächter and L. T. Biegler: “On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming”, 2004

Second derivatives

$$x_{k+1}(p) = x_k(p) - (LU)^{-1}\; F(x_k(p))$$

$$(LU)\cdot \Delta x_{k+1}(p) = F(x_k(p))$$

$$J \cdot s$$

$$(LU)\cdot \Delta s_{k+1}(p) = \frac{\partial F}{\partial x_k(p)} s_k$$

$$J \cdot d + s^T \cdot H \cdot s$$

$$\frac{d}{dp}$$

$$(LU)\cdot \Delta d_{k+1}(p) = \frac{\partial F}{\partial x_k(p)} d_k + {s_k}^T \cdot \frac{\partial^2 F}{\partial {x_k}^2(p)} \cdot s_k$$

Summary and Conclusions

- ADiCape is a source transformation tool employing the forward mode of AD written for CapeML language
- with Curtis-Powell-Reid method, calculation of a compressed sparse Jacobian brings space savings and considerable speed-up
- We expect DyOS integrator performance to considerably improve with the use of AD-generated derivatives
- With the new optimizer and precise AD second derivatives we hope to considerably cut down the number of iterations of the Integrator.

THANK YOU

FOR YOUR ATTENTION !