OPENMDAO DEVELOPMENT UPDATE

Rob Falck
Development Team Lead
DEVELOPMENT TEAM

- John Jasa
- Tad Kollar
- Ken Moore
- Bret Naylor
- Kaushik Ponnapalli
- Steve Ryan
- Herb Schilling
ASSESSING OUR IMPACT
FOCUS OF RECENT DEVELOPMENT
LOWERING BARRIERS TO ENTRY

• Executable Documentation

• Practical MDO with OpenMDAO notebooks and videos

• build_pyoptsparse
Rewrote documentation using jupyter-book

Notebooks as documentation

Documentation can test itself as part of our CI process

Users can test OpenMDAO examples on Google Colab without the need to install anything locally.
• John Jasa has joined our team and has done an amazing job at producing tutorial notebooks and corresponding videos.

• These videos provide common lessons that we find ourselves teaching to users.
The MDOLab provides an amazing tool in the form of pyoptsparse.

We were finding it difficult to build pyoptsparse in a way that could provide IPOPT as an option for users.

Easier to build pyoptsparse with support for

- IPOPT
- ParOPT
- SNOPT (if source is available locally)

https://github.com/OpenMDAO/build_pyoptsparse
REDUCING USER PAIN

- Reports
- Visualization Tools
- Performance Improvements
I'm having trouble with my model.

Have you looked at the N2 diagram?

How do I do that again?

I'm having trouble with optimization.

How does your scaling look?

How can I tell?

REPORT GENERATION

• We provide information that users typically need.
  • N2 diagram for connectivity
  • Scaling report

1. Many users don’t know that these exist, let alone how to get them.
  • Why aren’t we just doing this all the time?

• Provide more rich feedback via HTML than standard output.
  • We shouldn’t limit ourselves to ascii (it’s not 1989 anymore)
  • Standard output often gets swamped by solver or optimizer output.
  • In the future we’ll be putting more of our standard output in reports.
## REPORT GENERATION

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Triggered by</th>
<th>During</th>
<th>Pre/Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>scaling*</td>
<td>Driver scaling report</td>
<td>Driver</td>
<td>_compute_totals</td>
<td>post</td>
</tr>
<tr>
<td>optimizer*</td>
<td>Optimization summary</td>
<td>Problem</td>
<td>run_driver</td>
<td>post</td>
</tr>
<tr>
<td>connections</td>
<td>Connections viewer</td>
<td>Problem</td>
<td>final_setup</td>
<td>post</td>
</tr>
<tr>
<td>total coloring*</td>
<td>Total coloring</td>
<td>Driver</td>
<td>get_coloring</td>
<td>post</td>
</tr>
<tr>
<td>n2*</td>
<td>N2</td>
<td>Problem</td>
<td>final_setup</td>
<td>post</td>
</tr>
<tr>
<td>checks</td>
<td>Config checks</td>
<td>Problem</td>
<td>final_setup</td>
<td>post</td>
</tr>
<tr>
<td>summary</td>
<td>Model summary</td>
<td>Problem</td>
<td>final_setup</td>
<td>post</td>
</tr>
</tbody>
</table>

* reports placed in the reports subdirectory by default.

* Users can design and implement their own reports.

* Users can choose individual reports, mute them all, and choose the destination directory.

* report generated automatically. Others need to be specified.
• This probably needs no introduction.
• It’s an extremely useful tool and now it’s just generated in the process of executing a model.
THE SCALING REPORT

• See at a glance the impact of scaling on variables and constraints in the optimizer’s context.
**THE OPTIMIZER REPORT**

**OpenMDAO Optimization Report for Problem brachistochrone (All values are in unscaled, physical units)**

- **Problem:** brachistochrone
- **Script:** /Users/falk/projects/openmdao/gp/test/brachistochrone.py
- **Optimizer:** SciPyOptimiz.SLSQP
- **Number of driver iterations:** 46
- **Number of objective calls:** 47
- **Number of derivative calls:** 44
- **Execution start time:** 2022-09-22 06:06:12
- **Wall clock run time:** 00 hours 00 minutes 00 seconds 041.7 milliseconds
- **Exit status:** SUCCESS

**Objectives**

<table>
<thead>
<tr>
<th>name</th>
<th>val</th>
<th>ref</th>
<th>ref0</th>
<th>adder</th>
<th>scaler</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>trn.phase0.time</td>
<td>1.0016e+00</td>
<td>1.0000e+00</td>
<td>0.0000e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Design Variables**

<table>
<thead>
<tr>
<th>name</th>
<th>atlas</th>
<th>size</th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>lower</th>
<th>upper</th>
<th>equals</th>
<th>ref</th>
<th>ref0</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>trn.phase0.dur</td>
<td>1</td>
<td>1.8</td>
<td>1.8</td>
<td>1.8</td>
<td>0.5</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
<td></td>
<td>10.0</td>
</tr>
<tr>
<td>trn.phase0.x</td>
<td>9</td>
<td>(0.0184, 0.21)</td>
<td>(0.92, 1.4e+21)</td>
<td>(1e+21)</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trn.phase0.y</td>
<td>9</td>
<td>(4.82, 9.84)</td>
<td>(7.11, 1e+21)</td>
<td>(1e+21)</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trn.phase0.v</td>
<td>10</td>
<td>(1.76, 10.0)</td>
<td>(1.22, 1e+21)</td>
<td>(1e+21)</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Summary of all optimization variables, and the current optimizer settings.
See the sparsity of your jacobian and how OpenMDAO is solving the derivatives.

Each "color" corresponds to a linear solve necessary to compute the total derivatives.
### The Inputs Report

<table>
<thead>
<tr>
<th>Promoted Name</th>
<th>Source Name</th>
<th>Source is IVC</th>
<th>Source is DV</th>
<th>Units</th>
<th>Shape</th>
<th>Tags</th>
<th>Val</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESIGN.comp:PR</td>
<td>_auto_iwc.v25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:PRmap</td>
<td>DESIGN.comp:map:PRmap</td>
<td></td>
<td></td>
<td>(1)</td>
<td>[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:eff</td>
<td>_auto_iwc.v26</td>
<td></td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:effMap</td>
<td>DESIGN.comp:map:effMap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[5.2]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:Wc</td>
<td>DESIGN.comp:corrinputs:Wc</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[0.789313]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:WcMap</td>
<td>DESIGN.comp:map:WcMap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[30]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:alphaMap</td>
<td>_auto_iwc.v22</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[0]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:NcMap</td>
<td>_auto_iwc.v23</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:SMN_map:RlineMap</td>
<td>DESIGN.comp:map:stall_R:RlineStall</td>
<td>✓</td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:SMW_map:NcMap</td>
<td>_auto_iwc.v22</td>
<td></td>
<td>rpm</td>
<td>(1)</td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:SMW_map:RlineMap</td>
<td>DESIGN.comp:map:stall_R:RlineStall</td>
<td>✓</td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:SMW_bal:Rnc:RlineStall</td>
<td>DESIGN.comp:map:stall_R:RlineStall</td>
<td>✓</td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:SMW_bal:Rnc:RlineStall</td>
<td>DESIGN.comp:map:SMW_bal:Rnc:RlineStall</td>
<td>✓</td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:SMW_bal:Rnc:RlineStall</td>
<td>DESIGN.comp:map:SMW_bal:Rnc:RlineStall</td>
<td>✓</td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:PRmap</td>
<td>DESIGN.comp:map:PRmap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[5.2]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:PRmap</td>
<td>DESIGN.comp:map:PRmap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[5.2]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:PRmap</td>
<td>DESIGN.comp:map:PRmap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[5.2]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:PRmap</td>
<td>DESIGN.comp:map:PRmap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[5.2]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:PRmap</td>
<td>DESIGN.comp:map:PRmap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[5.2]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:Wc</td>
<td>DESIGN.comp:corrinputs:Wc</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[0.789313]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:map:WcMap</td>
<td>DESIGN.comp:map:WcMap</td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>[30]</td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:Ft_Itec:PR</td>
<td>DESIGN.inlet:flow:flow:Ft_Itec:PR</td>
<td></td>
<td></td>
<td>(1)</td>
<td>[3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESIGN.comp:Ft_Itec:composition</td>
<td>DESIGN.inlet:flow:flow:Ft_Itec:composition</td>
<td></td>
<td></td>
<td>(1)</td>
<td>[1]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Similar to list_inputs in HTML format.
- Sortable, filterable columns to answer questions like
  - What inputs does the user need to be providing to this model?
  - Are all of these inputs connected to the same IVC?
PERFORMANCE IMPROVEMENTS

- Vectorized, fixed-dimension interpolants
- Coloring improvements
- Efficiency improvements to apply_linear calls under LinearBlockGS and LinearBlockJac
• Vectorized, fixed-dimension interpolation algorithms significantly increase speed.

• Discontinuing support for the Fortran-based MBI interpolation tool.
FUTURE DEVELOPMENT EFFORTS
AUTOMATIC DIFFERENTIATION

- **ExplicitFuncComp** and **ImplicitFuncComp** can use jax.
- Sympy source-code transformation
- Other possible paths?
OpenMDAO enables large, complex models.

Dealing with connections increases risk of user error.

- Forgotten connections, automatically assigned to AutoIVC unexpectedly.
- Inputs intended to be promoted to the same variable but forgotten.

Solutions?

- Better feedback – The inputs report.
- Some method of bundling connections?
  - Something like a Simulink bus?
- Development of best practices?
Several teams seem to be working on the best practice for providing input for a problem that hides code.

Let analysts and system engineers change select inputs without dealing with code or diving into many files.

Notebooks are one way.

Should we support a canonical way or let users handle it?
SUBPROBLEM INTERFACE

- OpenMDAO Problem where `run_model` and `compute_totals` are used within the `compute` and `compute_partials` methods of a component.

- Can improve performance by “hiding” inputs and outputs that are irrelevant to the outside problem.

- There are some challenges with rolling-your-own solution that might make a canonical OpenMDAO approach preferable.
Leveraging second derivatives can dramatically improve convergence.

Both IPOPT and newer versions of SNOPT can utilize Hessians.

Requires extension to MAUD upon which OpenMDAO is based.

Reliable, efficient AD is a prerequisite.
QUESTIONS?