Parallel Algorithmic Differentiation in OpenModelica

**Motivation**
- Electrical power generation shifting from large power plants to distributed small power producers.
- Small power producers together with conventional power plants form virtual power plants.
- Combined virtual power plants require very large heterogeneous models with many variables and states.
- For efficient and economic operation real-time optimization is needed.

**Previous work and resulting challenges**
- Modify generated code to obtain ADOL-C instrumented C++ code and enable computation of derivatives.
- For large models the C and C++ code generation may be very slow.
- Only few structural information of the model itself is available in the generated code.
- Parallel execution to speed up the simulation is not available out of the box.

**OpenModelica**
- Many large models can be described in the modeling language Modelica.
- Open source tool developed by the Open Source Modelica Consortium to simulate and optimize models using the Modelica language.
- Parses and compiles the models given and creates C and C++ code.
- Generated code can be run to obtain simulation results.

**Efficient implementation**
- ADOL-C uses execution of function with adouble type variables to generate internal representation of the function called the trace.
- Trace can be repeatedly used to calculate derivatives at different evaluation points.
- Additionally implemented functionality to read and write ASCII-based traces in branch ascii2tape.
  - `write_ascii_trace(FILENAME,tag)`
  - `read_ascii_trace(FILENAME,tag)`
- Traces can be generated while OpenModelica compiles the Modelica model.
- Allows to generate and evaluate traces without execution of C++ code generated by OpenModelica.
- Trace evaluation, and thus derivative calculation can now be called outside of C and C++ via Python or Julia.
- Pure trace evaluation is much faster than the execution of the generated C++ code due to less overhead.
- Structural information like sparsity can be used more efficiently.
- Parallelize Jacobian calculation by dividing seed matrix into partitions which can be handled individually.

**Calling ADOL-C drivers in julia**
- We wish to use high level programming language inside of OpenModelica.
- As this may lead to slower derivative calculation we tested calling jacobian by

  ```julia
  jac = zeros(Cdouble, n,n)
  refAr = [Ref(jac,i) for i=1:size(jac,1):length(jac)]
  ccall(:jacobian,"PATH/TD/libadolc.so","Cshort, \\
       Cint, Cint, Cint, Ptr(Cdouble),Ptr(Cptr(Cdouble)), \\
       tag,n,nargs,refAr]
  ```
- Tests show only small constant offset of approx. 0.3 sec.

**Acknowledgements**

The authors thank the German Federal Ministry of Education and Research (BMBF) for funding of the project PARADOM under the support code 01IH15002.

**References**