Mixed-language Automatic Differentiation

Valérie Pascual – Laurent Hascoët

AD 2016 Oxford - September 2016
Mixed Language AD - Outline

- Language standards and interoperability
- Automatic Differentiation algorithms for interoperability
- Performance discussion
- Implementation
- Further works
Mixed language Automatic Differentiation

Existing works for mixed-language software are categorized in two categories:
- Program understanding and analysis
- Refactoring (IDE)
Language standards and interoperability

• Compiler dependent – ad hoc mechanisms
different conventions to associate procedure names
between languages (*BIND_* C)
portability using macro definitions

• ISO Fortran 2003 interoperability standards for interaction
with C (*ISO_C_BINDING* module)
Support for interoperability

- Mixed language applications are compiled using Separate compilation and are differentiated using Global differentiation

- AD analyses must be global for accuracy

- Differentiation of the complete source code without using hand-written stubs
Support for interoperability

Language independent internal representation
Support for interoperability

Call graphs of Flow graphs

Nested Symbol Tables
Support for interoperability

• Interoperability of procedures:
  - Naming convention or *bind* attribute in Fortran 2003
  - Parameter-passing strategies: by reference, by value

• Propagation of data-flow information between a caller procedure and a callee using a “Translator”
Support for interoperability

• Each argument is split into elementary components
• For each elementary formal argument of the called procedure, the Translator associates
  - the corresponding elementary actual argument
  - a boolean copy / no copy upon return
• At entry from the called procedure: the data flow value of the passed argument is copied to the formal argument
• At return time, copy back to the actual parameter / no copy
Support for interoperability

• Fortran calls C:  
  
  REAL X
  
  void foo(float *a)
  
  CALL FOO(X)
  
  Passed argument is address of X
  
  Upon return, no copy takes place into &X

• C calls Fortran:  
  
  float *y;
  
  SUBROUTINE BAR(V)
  
  bar(y);
  
  REAL V
  
  Passed argument is *y
  
  Upon return, value of V is copied back into *y
Performance discussion

- CalculiX  http://www.calculix.de
  38000 lines of Fortran  +  160000 lines of C

- tapenade -b -head "linstatic(co)>(maxvm_pnorm)"

<table>
<thead>
<tr>
<th></th>
<th>analyses</th>
<th>total time</th>
<th># diff params</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.c + external</td>
<td>30 seconds</td>
<td>35 minutes</td>
<td>36</td>
</tr>
<tr>
<td>*.c *.f</td>
<td>80 seconds</td>
<td>9 minutes</td>
<td>30</td>
</tr>
</tbody>
</table>
TAPENADE
On-line Automatic Differentiation Engine

Given

- a source program,
- the name of the top routine to be differentiated,
- the dependent output variables whose derivatives are required,
- the independent input variables with respect to which it must differentiate,

this tool returns the forward (tangent) or reverse (adjoint) differentiated program.
If you want to be kept informed about new developments and releases of TAPENADE, subscribe to the tapenade-users mailing list.

Select the input language:
- from the files extensions
- Fortran 77
- Fortran 95
- C

Upload source and include files, repeatedly.
Type the file path in, or browse:

Choose File
No file chosen

and upload it

as a source
f90
as an include
program.c

Remove selected files
Retry with new files

Name of the top routine:
Original call graph

- External function PRINTF
- subroutine BAR
- subroutine FOO
  - Calls subroutine BAR

Differentiated call graph

- subroutine BAR_B
- subroutine FOO_B
  - Calls subroutine BAR_B

```c
/*
 * Generated by TAPENADE  (INRIA, Ecuador team)
 * Tapenade 3.11 (r6148) - 16 Aug 2016 14:18
 */
#include "GlobalDeclarations.c"
void foo(float *x, float *y) {
  bar(x, y);
}

void foo_b(float *x, float *xb, float *y, float *yb) {
  bar_b(x, xb, y, yb);
  yb = 0.0;
}
```

Tool: (DD999) Warning: mixed language differentiation not yet implemented
Command: Took procedure foo as default differentiation root
foo: Tool: (DD999) Warning: mixed language call is not yet implemented
main: [TC33] External I/O procedure printf is not declared

```fortran
! Generated by TAPENADE  (INRIA, Ecuador team)
! Tapenade 3.11 (r6148) - 16 Aug 2016 14:18
!
! Differentiation of bar in reverse (adjoint) mode:
! gradient of useful results: v
! with respect to varying inputs: u
SUBROUTINE BAR_B(u, ub0, v, vb) BIND(c)
  IMPLICIT NONE
  REAL, VALUE :: u
  REAL :: ub
  REAL :: v
  REAL :: vb
  REAL :: ub0
  u = 2*u
  v = u*u
END SUBROUTINE BAR
```
Further works

- Fortran 2003 “bind” statement for types, variables

- Generation of differentiated code following interoperability conventions and standards

- Separate differentiation for large mixed-language applications, with automatic generation of stub corresponding to a library