Measuring Uncertainty in Scientific Computations Using the Test Harness[1]

Brian T. Smith

Numerica 21 Inc.

August 3, 2011

IFIP WG 2.5 Working Conference 11

[1]The design of the test harness was supported in part by a DOE Phase 1 SBIR award DE-FG-02-04ER84028, July 14, 2004—April 12,2006

Introduction

- “A measured value is meaningless without a quantitative statement of its quality in the form of an uncertainty”
  - From A Framework for Uncertainty in Measurement, NPL, June 5, 2007
- This is just as true about scientific computation as it is about physical measurement
  - Software is useless unless the uncertainty in the results due to changes in its input are measured or analyzed and is demonstrated to be consistent with what is predicted from the characteristics of the problem being solved

This Presentation

- This presentation is about a tool to help provide assessment of uncertainty in scientific computation that
  - Depends on measured data
  - Depends on robustness of the algorithms
  - Depends on the correctness of the implementation

Software Development

- We write software to provide computations for some problem such as:
  - Modeling the physical phenomenon (weather)
  - Solving some mathematical problem (solve a linear system of equations)
  - ...
- In some cases, we have expectations (ideally mathematical models) of how the computations depend on the input data/parameters.

Software Devel. Continued

- In other cases, we have no expectations but want to discover what the uncertainty of the results are relative to the input data/parameters.
- What is presented is a general purpose tool TH
  - Helps with the measurement of uncertainty of results with respect to changes in input data or other changes of interest (different algorithm or different implementation).
  - It is designed and has been used to support testing of substantial packages of software (ASAP and BSSI)

Outline Of The Talk

- What is the test harness?
  - Its purpose and approach
  - Its design
  - Its installation in the application code
  - Its operation
  - Its support tools
- Case study
  - Measuring uncertainty in software to compute magnetic vector potentials around 3-D objects using a boundary element method

What Is The Test Harness?

- A change-detection tool to measure/diagnose changes made to code
  - The measures are user selected or user determined
  - Can be used to measure changes in results when the input is perturbed
- A tool package for large production code or libraries with components to:
  - Analyze code
  - Insert probes at desired locations
  - To monitor change in results
  - To measure change in intermediate values

Its Design

- Comparison of results from two "nearly equal" codes: Examples
  - Codes under development
    - one version is the "benchmark"
    - the other version is the "enhanced" version
  - Codes being optimized
    - one version is the non-optimized version
    - the other version is the optimized version
  - Codes being ported
    - one version is the development machine
    - the other version is the new machine

Design Continued

- Uncertainty testing
  - One version runs with base data
  - The other version runs the perturbed data
    - The difference in results measures the uncertainty of the solution to changed data or to whatever changed.

Design Continued

- Codes with the test harness installed in them run on one of two modes:
  - Generate mode:
    - Perform the computation
    - Write out data being monitored -- choices include:
      - Final results
      - Intermediate results
      - Initial data
      - At any specified in the code
- - Check mode
    - Perform the computation
    - Read saved data and compare with the data from the current computation

Design Continued

- Probes are user-determined -- either
  - Hand inserted (a single line)
  - Default inserted by the tools at each entry and exit point of specified procedures
- Probes specify:
  - Variable to be monitored
    - written out when in generate mode
    - read in and compared, when in check mode
  - Variable to be perturbed
    - perturbed in check mode
    - ignored in generate mode

Design Continued

- Tools perform the following operations:
  - Inserts probes that trace execution only
  - Inserts probes at default places (exit and entry of specified procedures)
  - Builds and inserts code to perform the data collection or perturbation
  - Builds and inserts code to perform the data reading and comparison

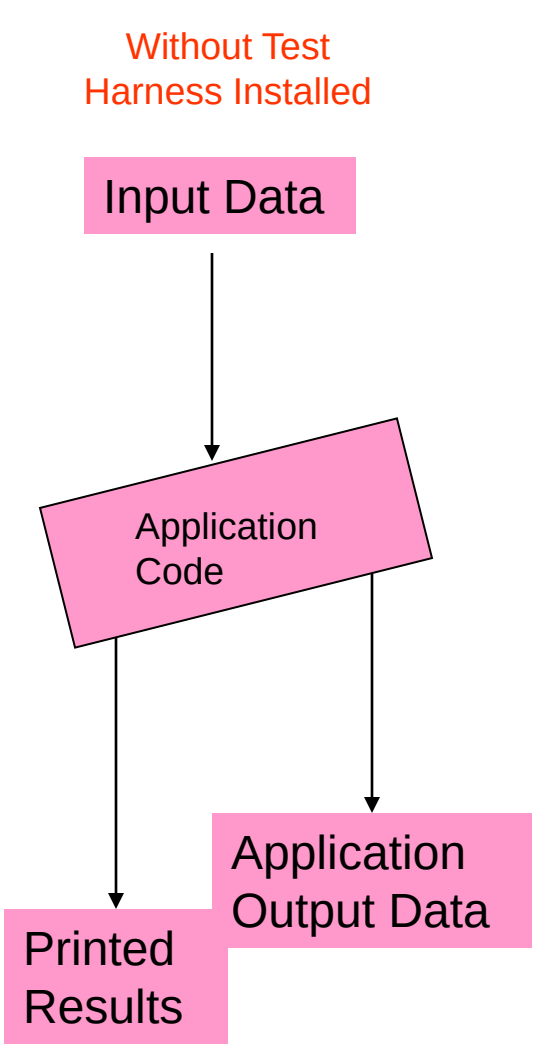
Design Continued

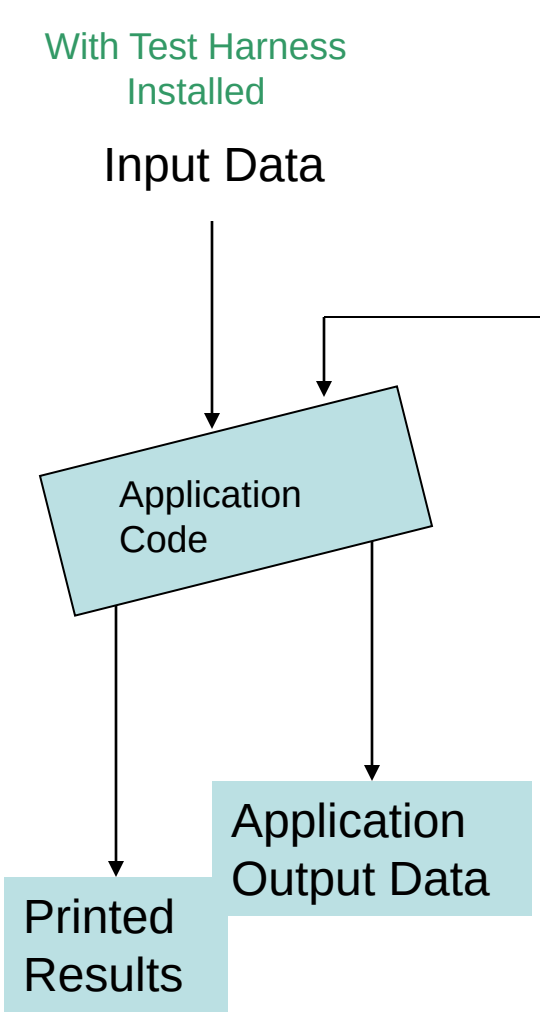
- Default procedures for perturbing and performing data comparisons are provided -- can select:
  - Identity comparison
  - Relative differences
  - Absolute differences
  - Thresholds above which the user is notified
  - Type of comparison for arrays
    - Norms
    - Relative to elements or norm

Design Continued

- Names of user-defined procedures can be specified to:
  - Perform the comparisons
  - Perturb the data
  - In both cases, the data environment at the probe point is accessible to these user-specified procedures via arguments

The Approach – The Application Code





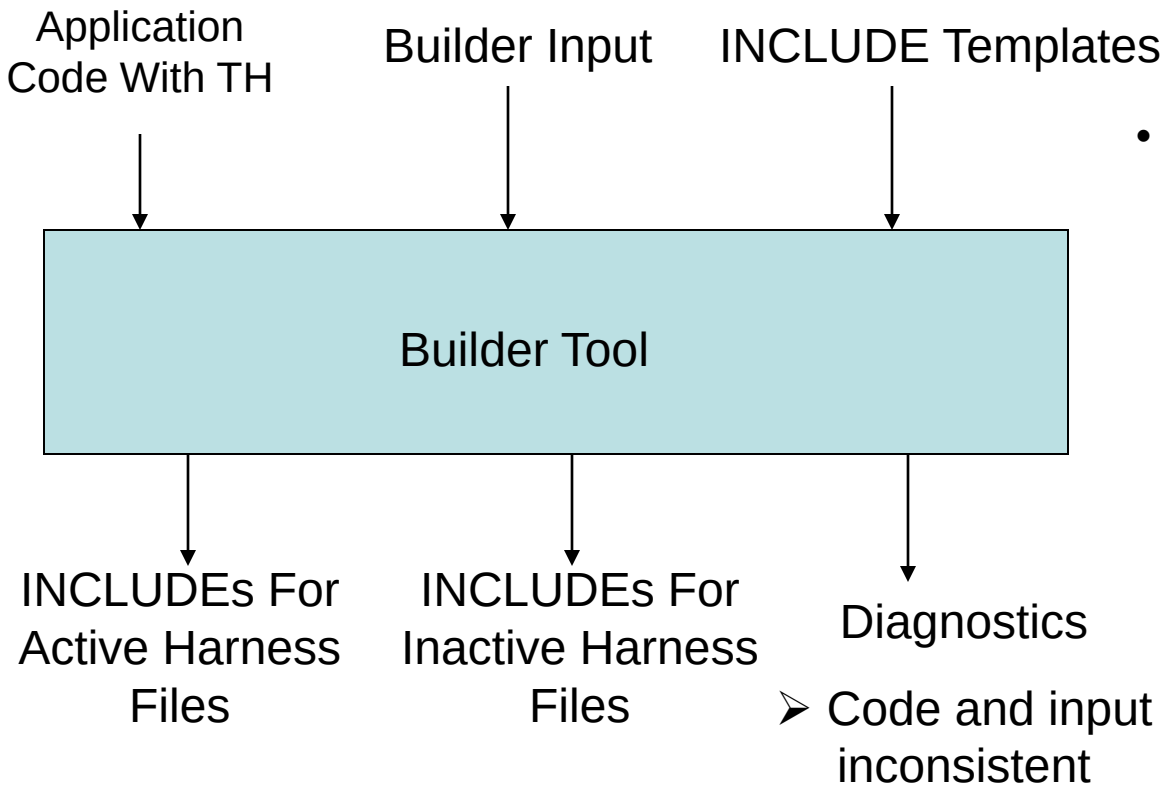
Insert INCLUDE lines into application code as follows:

- Specification part
- Initialization
- Input probes
- Output probes
- Specific probes
- Perturbing probes
- Finalization

Monitoring Required:
in main program

Optional:
in any selected subprogram

Building The Application With The Test Harness Installed

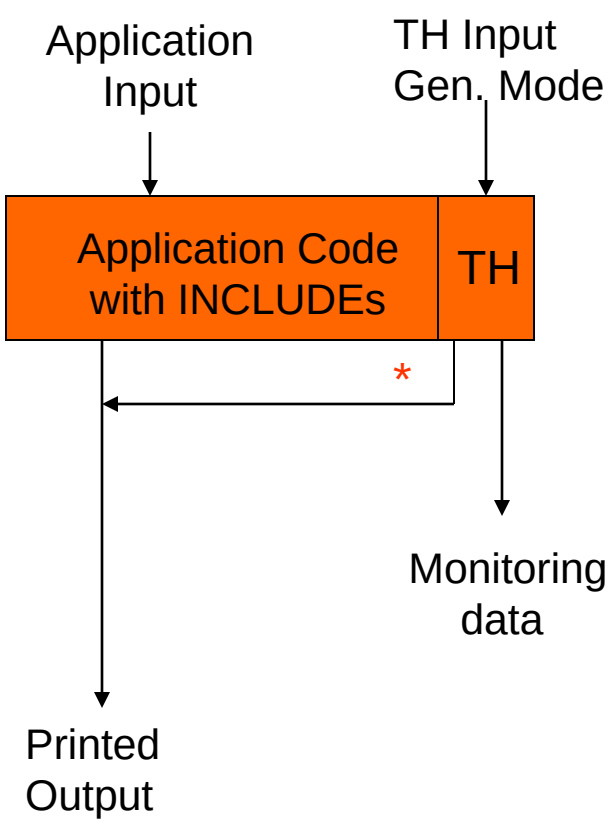


- Builder Input (user-supplied)
  - Variables To Monitor
    - Subprogram
    - Name
    - Declaration
    - Identity/closeness check strategy
    - Frequency
    - Debug Levels

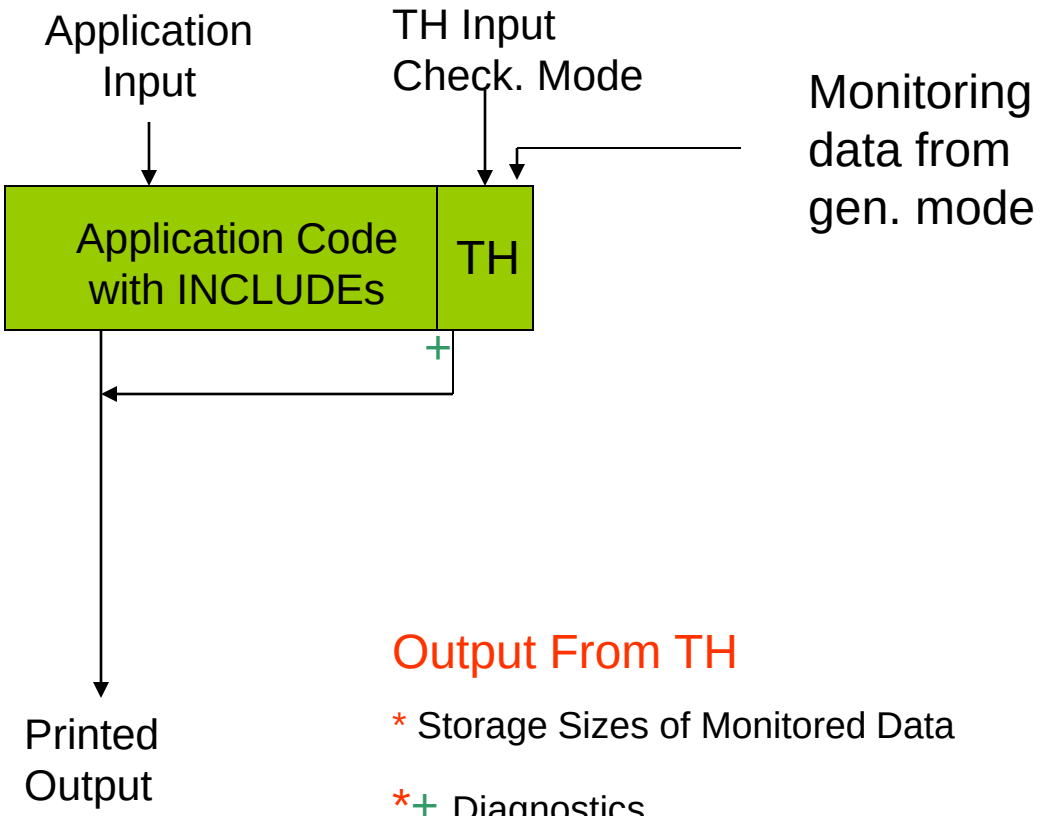
Operation With TH

Modes

Generate Mode



Check Mode



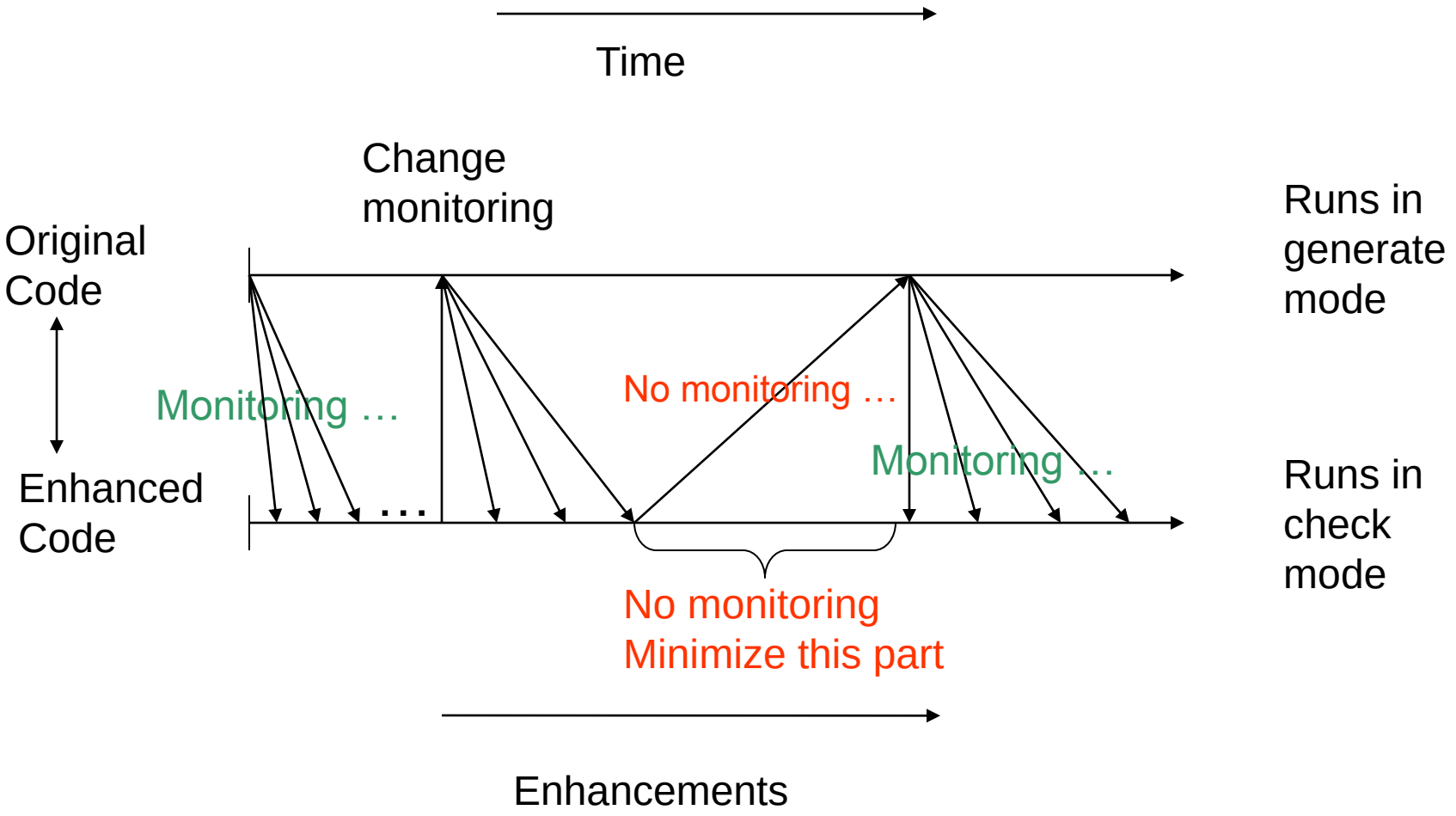
Output From TH

* Storage Sizes of Monitored Data

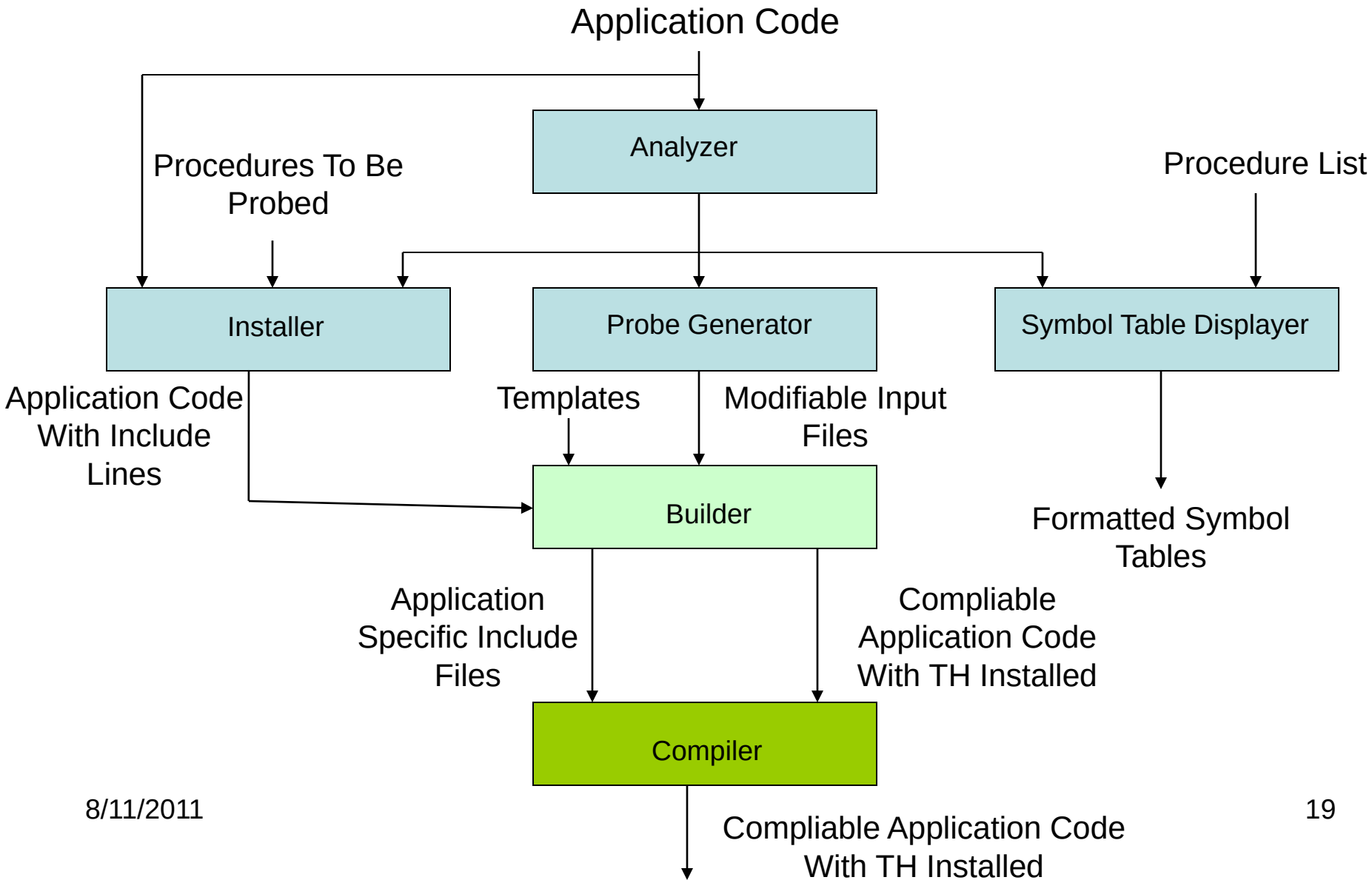
*+ Diagnostics

*+ Timing Performance per routine monitored

Usage Scenario



The TH Tools


Application Code
Analyzer
Procedures To Be Probed
Procedure List
Installer
Probe Generator
Symbol Table Displayer
Application Code With Include Lines
Templates
Modifiable Input Files
Builder
Formatted Symbol Tables
Application Specific Include Files
Compliable Application Code With TH Installed
Compiler
Compliable Application Code With TH Installed

The Case Study -- Outline

- Description of problem solved
- Description of software
  - Size in lines/procedures
  - Algorithms
    - Matrix assembly
    - Boundary integration accuracy
    - Linear equation solver
- Some simple examples of measuring uncertainty of results
    - Depending upon input boundary condition
    - Depending on internal values (quadrature points and weights)

Software Studied

- Examine a package of code to solve the 3-D vector Laplace equation
  - Application is magnetic potentials and fields surrounding 3-D objects
  - Computational expensive operation for "real" objects in "real" environments
  - The approach is to use a boundary element method where the solution on the boundary is completely determined from boundary conditions and that solution, using boundary integrals
  - The solution at any point in the interior of the boundary can be computed by evaluating boundary integrals using the boundary solution. (The boundary conditions can be flipped and the problem re-solved to provide data to compute the solution at an exterior point.)

Software Continued

- The package consists of:
  - Over 300 procedures
  - Standalone boundary node and element generators
  - Converters of surface boundaries and elements from PATRAN and SolidWorks formats to BEM package formats
  - Solvers (vector potential and fields)
  - Evaluators of the solution at exterior points of object
  - Visualization tools for the solution using VisIt Plot and TecPlot
  - Over 50K lines of code
  - All double precision although code is dependent on a single kind parameter that, when modified and the code recompiled, will change the precision

The Algorithm Examined

- The method of solution is to integrate singular Green's functions over the boundary using the boundary conditions to create a potentially large linear system of equations.

Uncertainties To Be Measured

- Uncertainty measures to be studied with the Test Harness
  - What is the uncertainty in the solution to changes in boundary conditions?
  - How sensitive are the results to changes in the quadrature points and weights?
  - The boundary conditions for real problems are likely only known to 3 digits, if that

Test Problems

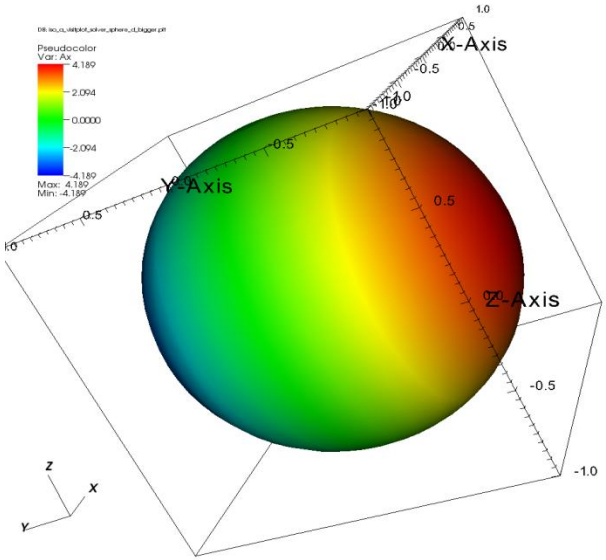
- Solve the vector potential Laplace equation (the A equation) for the following objects:
  - A sphere with Neumann boundary conditions (the solution is the magnetic potential on the boundary)
  - A annular cylinder with mixed boundary conditions (the solution is a mixture of the vector potential and the tangent of the magnetic field on the boundary)
  - A torus with Dirichlet boundary conditions (the solution is the tangent of the magnetic field on the boundary)

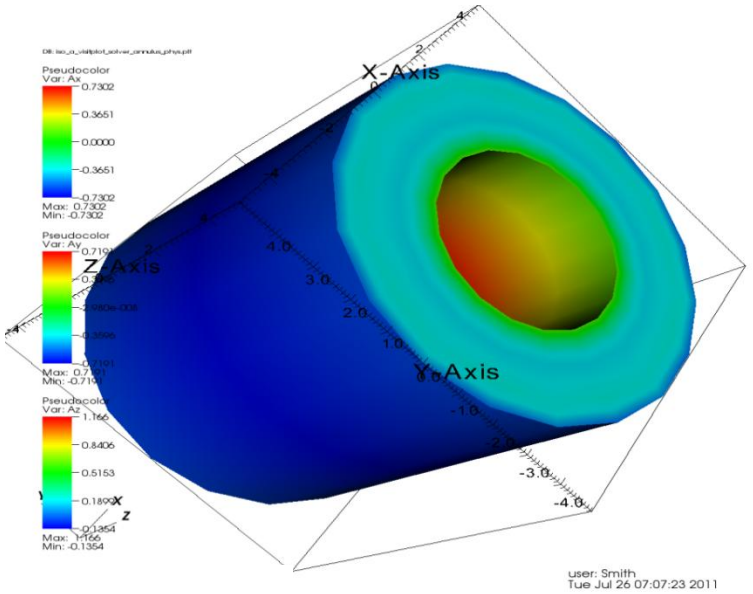
Perturbation Introduced

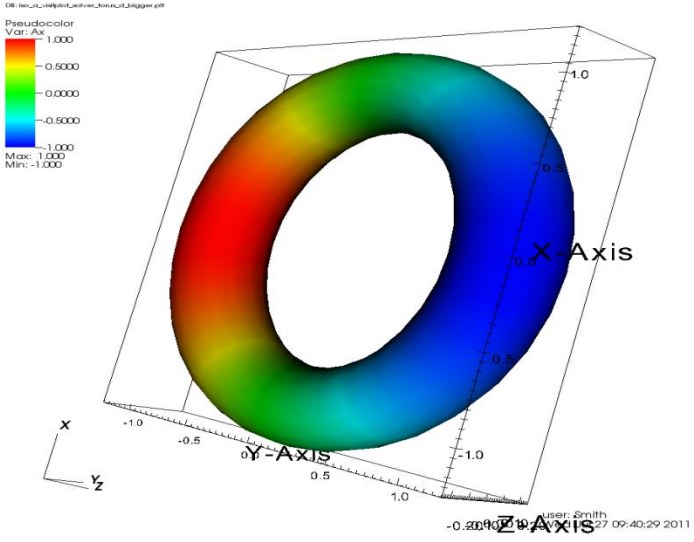
- Change the boundary conditions by introducing random perturbations relative to the boundary conditions themselves.
- The sizes of the changes:
  - 100 units in last place for double precision
  - 1 unit in the last place of single precision
  - 10,000 units in the last place of single precision
  - 100,000 units in the last place of single precision

Perturbations Tested

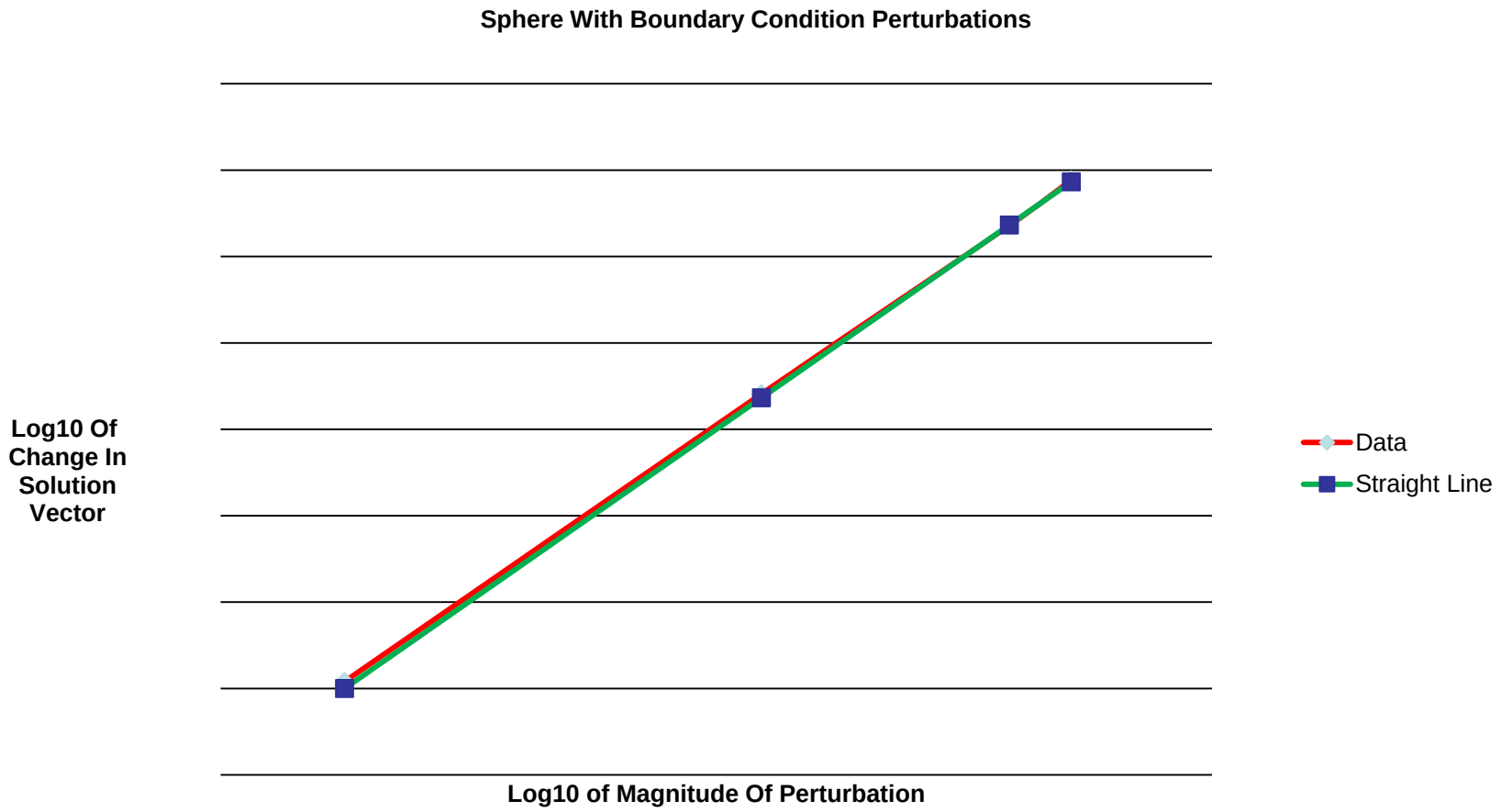
- Change the Gauss points and weights by introducing random perturbations of these values relative to themselves;
  - Use the same sizes as listed on previous slide
- Measure the maximum element norm of the difference relative to the maximum element in the solution (a TH supplied computation)


Pseudocolor
Var: Ax
4.189
2.094
0.0000
-2.094
-4.189
Max: 4.189
Min: -4.189
X-Axis
Y-Axis
Z-Axis

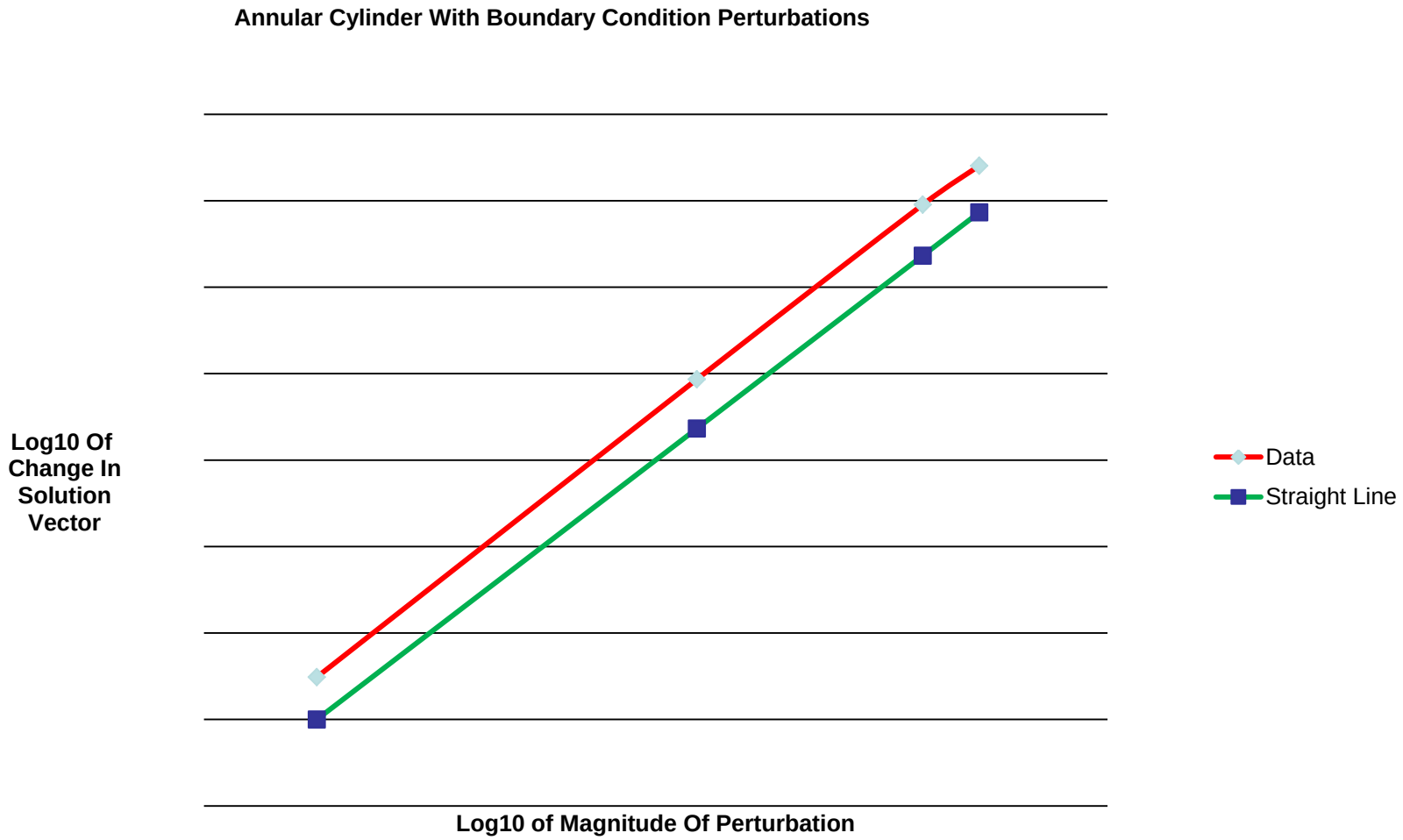

Pseudocolor
Var: Ax
0.7302
0.3651
0.0000
-0.3651
-0.7302
Max: 0.7302
Min: -0.7302
X-Axis
Z-Axis
Y-Axis
user: Smith
Tue Jul 26 07:07:23 2011


Pseudocolor
Var: Ax
1.000
0.5000
0.0000
-0.5000
-1.000
Max: 1.000
Min: -1.000
X-Axis
Y-Axis
Z-Axis
user: Smith

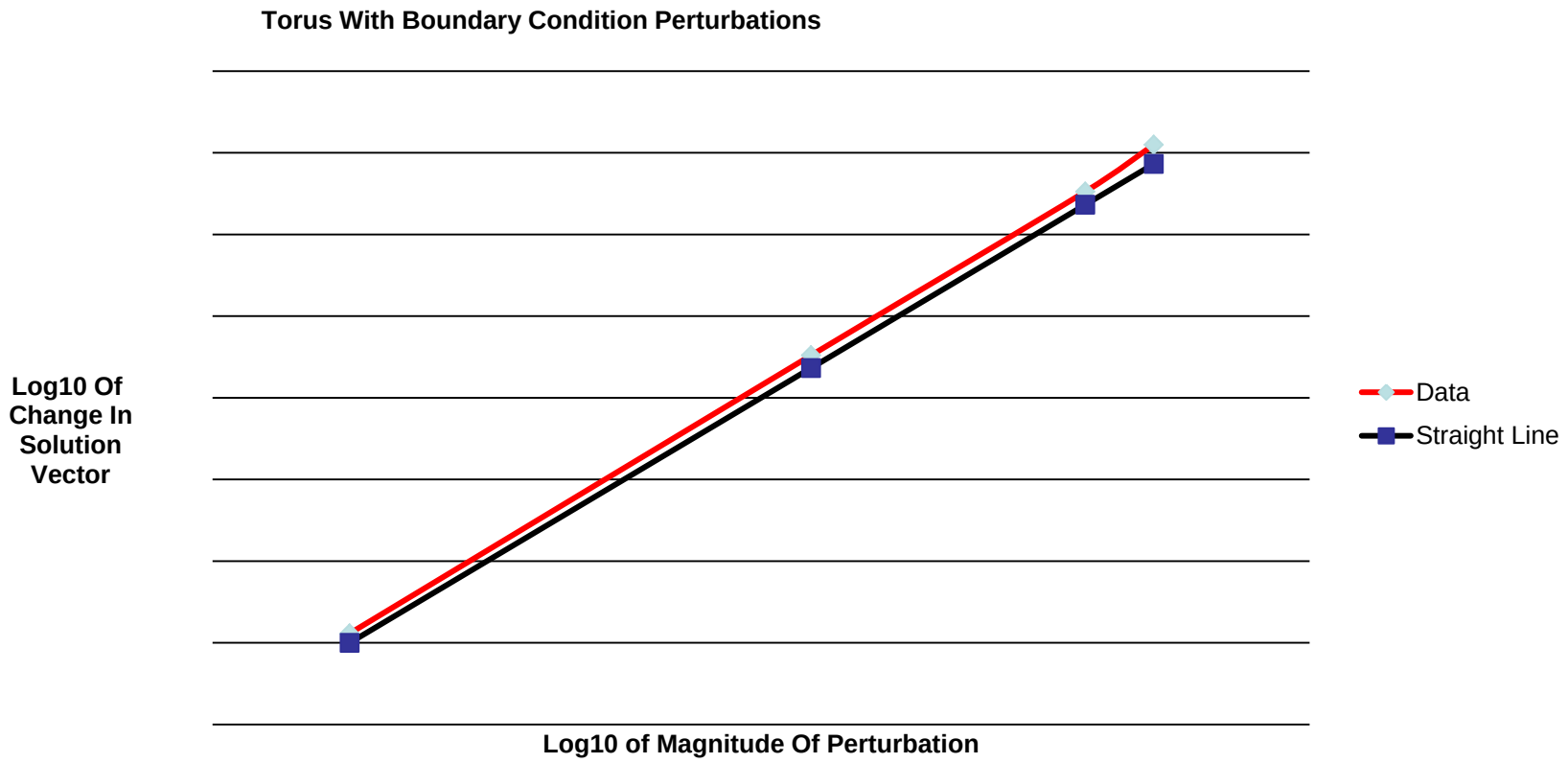
Results: Sphere BCs



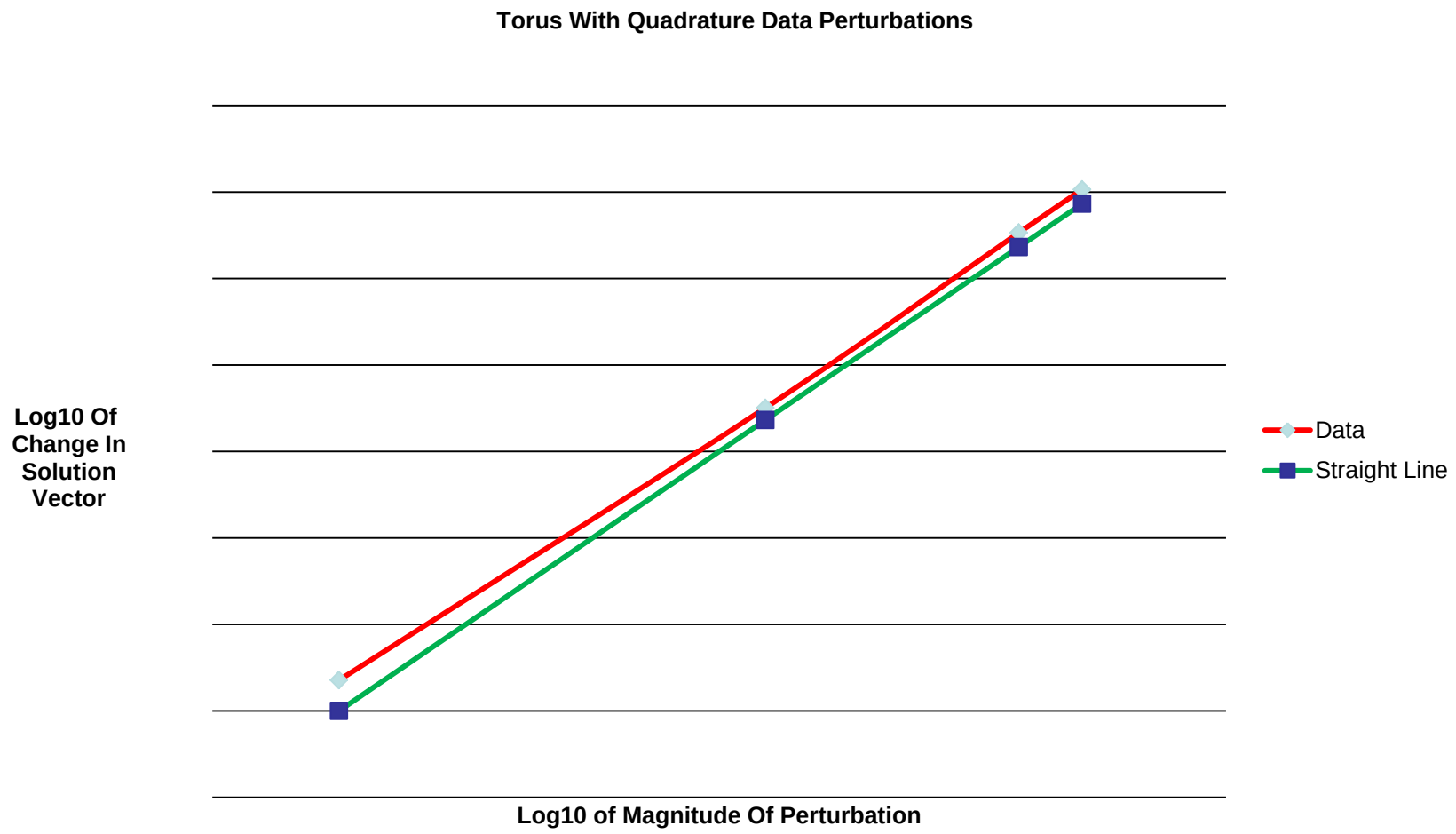
Results: Annular Cylinder BCs



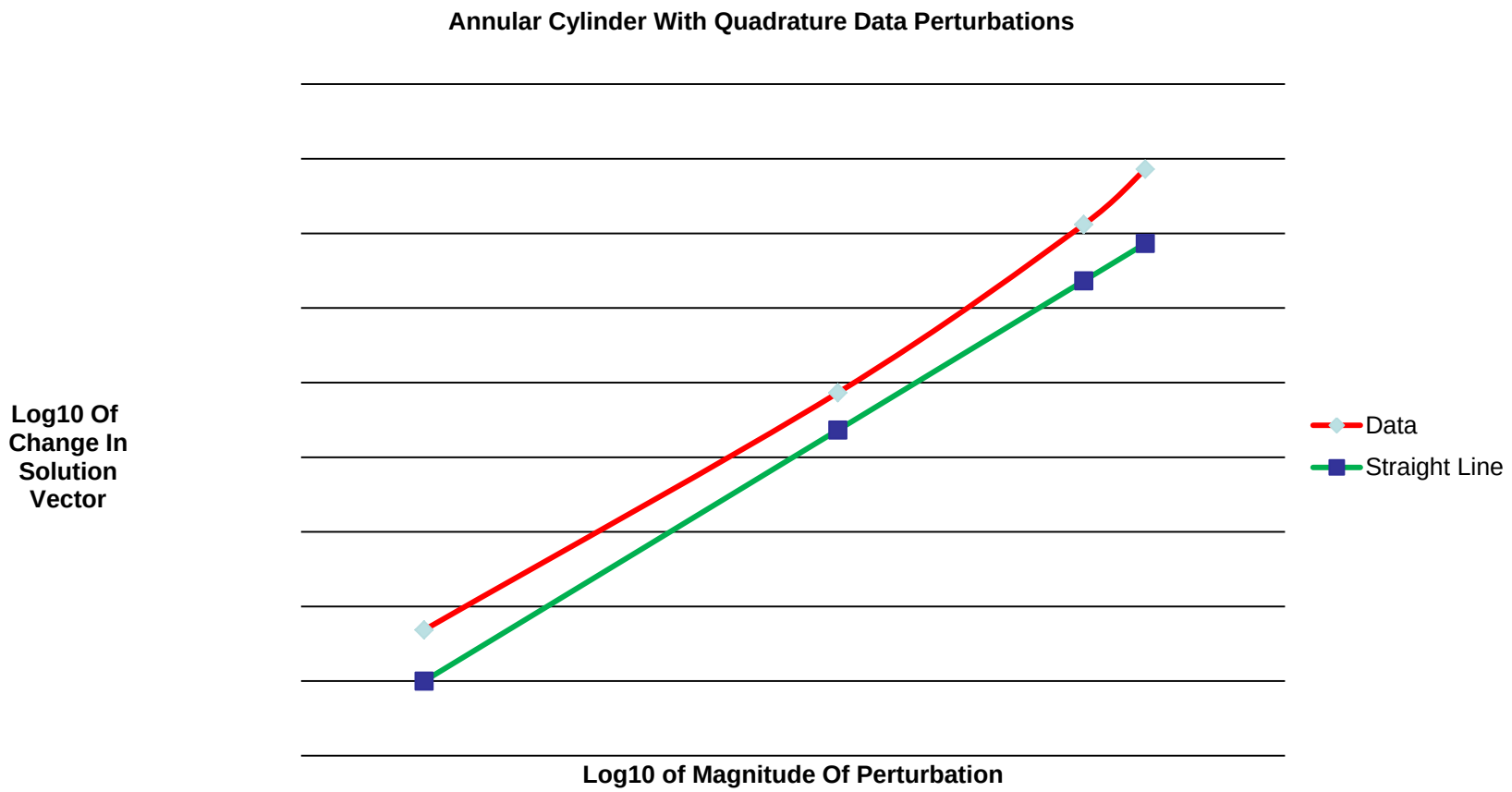
Results: Torus BCs



Results: Torus Quad



Results: Annular Cylinder Quad



Conclusions Of Case Study

- Analysis clearly shows the solutions will depend on the condition number of the generated linear system
  - For these problems, conditions numbers for sphere, torus, and annular cylinder are roughly 150, 200, 6000
- The case study confirms this behavior
  - The condition number is computed and printed
- Experiments with objects that are:
  - » More complicated topology
  - » Sharp edges and corners
  - Are needed

Summary

- The test harness has made it easy to measure the uncertainty in the solution under changes to various input and other critical values
  - The case study used problems with known solutions
  - These known solutions were not used in the measurements
- The purpose of the experiments was to formulate a computable measure to indicate the sensitivity of the solution to data uncertainty.
  - So far, the linear system condition number seems to be an indicator of this sensitivity
  - More study and analysis is needed

Thank You

- Brian T. Smith, N21 Inc.
  - 22 Crystal Mountain Rd
  - Angel Fire, NM 87710-1668
  - Email: carbess@swcp.com

Analyzer Information

- Information compiled by the Analyzer
  - Usual symbol table
    - All identifiers (also labels) and their attributes
    - Locates declaration/use points of all identifiers/labels
      - » Future: locate where names are actual arguments
  - Location of certain TH-related code points
    - All entry points into all procedures
    - All STOP actions (implicit or explicit statement) except run-time errors and exceptions
    - All RETURN actions (implicit or explicit statement)
    - All READ/WRITE/PRINT statements
      - » Separate list for input and output statements

Analyzer Information Cont'd

- Complete interfaces
  - For external/internal/module procedures
  - Creates reference/definition information for all dummy arguments that have no explicit intent
- Compares interfaces (explicit and generated) with actual references
  - Flags locations where interface and reference are inconsistent
    - Will implement an option to defeat this check by procedure name (e.g.: MPI references violate many Fortran rules, unless a tailored MPI module interface is provided)

Analyzer Information Cont'd

- Encodes all literals as identifiers
  - Locates where they are used to facilitate replacement with named constants
- Replaces labeled action statements (except labeled DO terminations) with unlabeled action statements
  - The replacement is a labeled CONTINUE statement followed by the original statement unlabeled
  - Necessary to insert TH probes at arbitrary points

Capabilities Of The New Tools

- A tool to create symbol tables for a Fortran prog.
- A tool to insert include lines in a Fortran application code that specify data probes in default places
- A tool to build that installs in a user-specified list of procedures default data monitoring for two cases:
  - No application code variables monitored but procedure trace and performance data generated
  - All application data “external” to a procedure monitored
    - “external” means:
      - » Dummy arguments
      - » Variables in common
      - » Variables accessed from a module
      - » (Future) data read or written from a file

TH Characteristics

- Allows the data monitoring to be selective
  - The user selects which variables in which routines are monitored
- Allows for “reduction tests” to avoid large “check-data” files
    - Examples “reduction” tests: array norms, key array elements, checking frequency
- Allows comparisons of past and current monitored values to be
    - Identity
    - “near-identity” with user-settable tolerances, both absolute and relative
    - A combination of the last two tolerances
    - By element or by norm
    - Differences completely ignored

Mechanism – Installing the Harness In Application Codes

- Required changes to application code files
  - Add “INCLUDE” lines, referencing the test harness module or compile the test harness module before any application code is modified
  - To the original source code
    - » Insert typically 3 or 4 “INCLUDE” lines per subprogram to be monitored
- Create an input file for the builder tool
  - For each monitored subprogram
    - – A list of variables to be monitored (possible to have none)

An Example – A Main Program

Original Application Code

```
Program main
   … Specifications

   … Executables

End program main
```

Application Code With TH Installed

```
Program mine
   INCLUDE "use_testing_harness_main"
   … Specifications
   INCLUDE "initialize_testing_harness"
   … Executables
   INCLUDE "write_output_and_finalize_all.mine"
      ….
   INCLUDE "perturb.bcs"
End program mine
```

Plus similar INCLUDE lines for each:

- STOP statement monitored
- Selected probe point monitored

Builder Input – An Example

```
Input.f90                    ! Input source file
Output.f90                   ! Output source file
HARNESS_TEMPLATES   ! Directory
APPL_TH_INCLUDES       ! Dir. Of Generated INCLUDE files
APPL_EM_INCLUDES       ! Dir of INCLUDE files for Inactive TH
#harness# th
  #hvar# debug_unit 6       ! Unit for debug output
  …
#main#  main
  #output#
     #mvars# a  “real(8)::” … abs_tol=10.0
     #ptbvars# a  “real(8)::” … value=10.0   how=rndm_rel     perturber=my_code
```

Types Of Application Subprograms

- Currently, any Fortran standard-compliant code (Fortran 77/90/95/03)
  - All inserted code is also F90-compliant
  - Dependencies on the language are modest but vary with the tool
- Plans to support C for:
  - Builder and includer tools using C template codes
    - Test harness templates can readily be rewritten in C
  - Analysis tools currently assume Fortran application code
    - Tools are designed to facilitate replacement of specific software pieces to support C or other languages

The TH Provides

- Storage size information of the data being monitored (per subprogram monitored)
  - Shows what routines are creating large amounts of data
- Provides timing performance per subprogram
  - Indicates code performance
- Provides performance information of the test harness itself
  - Performance cost to create the data
  - Performance cost to read past data and compare with current values with past values

Extensibility

- The generated “INCLUDE” files are visible
  - Can be modified easily by hand
- The data comparison and timer modules are visible
  - Data comparison module can be enhanced to:
    - Check identity/near-identity for the non-default, non-intrinsic types
    - Perform special data consistency tests on particular variables, like arrays and structures
  - Timer module can be modified or enhanced to
    - Provide different timers
    - To modify what is measured

Experience To Date

- The development and debugging of the builder tool
- Modernization of 5 application codes and checking the results
- Debugging a parallel MPI code using finite boundary element method
    - Decided first to serialize the code and debug the serial code with the test harness
        - » This worked and found problems
    - Work in progress to create a version of the test harness that works directly on parallel code

Code Maintenance And Development Over The Grid

- Test harness supports:
  - Relative and absolute comparison criteria with readily specified tolerances
  - Comparisons of aggregate structures like arrays and derived types
  - Frequency of comparisons specifiable
  - Test harness always installed in the application
    - The application code can be compiled, disabling the test harness, resulting in the original code
  - Easy replacement of comparison modules
    - Easy to change from looking for computational difference to evaluating the accuracy of the results

Vision Of Usage

- In the grid environment:
  - Easy retesting of the code on “new” machines
  - Easy comparisons of “new” code with “old” code
  - Designed to facilitate regression testing over the grid on a regular basis
  - Permits the use of new evaluation techniques that answer questions like:
    - Are the current results and past results “equivalent”?
      - » Do they solve the same problem?
      - » Are the results “numerically” indistinguishable?

Planned Enhancements

- Integration of HDF5 portable exchange format
- Modification to support SPMD MPI codes
- Specification of how to do this for C codes
  - Involves rewriting the include templates in C
  - Involves creating a version of the builder tool that creates C #INCLUDE files
  - C codes involve arrays of “unknown” extents
    - Use the same approach as used for Fortran assumed-size arrays – the array variable to be monitored may require its extents to be specified in checking procedures

Planned Enhancements

- Reimplementation of parts of the Analyzer and its related tools
  - Redesign to use an API to access symbol table and other related information
  - This will facilitate replacement of the reference version of the Analyzer with compiler-specific version using vendor-offered interfaces satisfying the API

Summary

- Test harness to monitor program enhancement
- Avoids tedious, error-prone operations of testing and checking
- Extensible and flexible tool
- Tracks enhancements in code
  - Useful for porting, code modernization
  - The tool supports application and library software development

Documentation

- Documentation:
  - 2-Page Summary
  - Getting Started instructions
  - Overview:
    - » Creating A Test Data Environment To Detect Errors In The Code Conversion Process, Version 0.6, November, 2005, approx. 14 pages, Brian T. Smith
  - User's Guide:
    - » The Test Harness User's Guide, Version 0.6, November, 2005, approx. 19 pages, Brian T. Smith

Implementations

- CD with installation tests and 3 sample applications of the test harness (code modernization)
- Installations fully tested and available for:
  - Linux X86 for Nag f95, g95, and pgf90
  - Linux EM64T for NAG f95 and g95
  - SUSE and AIX for xlf95
  - Cygwin for NAG f95, g95, CVF, Lahey
  - XP for CVF and Lahey/Fujitsu

Contact Information

- Brian T. Smith, N21 Inc.
  - 22 Crystal Mountain Rd
  - Angel Fire, NM 87710-1668
  - Email: carbess@swcp.com