Blurring the Inputs: A Natural Language Approach to Sensitivity Analysis

Bil Kleb,[*] Richard A. Thompson, and Christopher O. Johnston

To document model parameter uncertainties and to automate sensitivity analyses for numerical simulation codes, a natural-language-based method to specify tolerances has been developed. With this new method, uncertainties are expressed in a natural manner, i.e., as one would on an engineering drawing, namely, `5.25 +/- 0.01`. This approach is robust and readily adapted to various application domains because it does not rely on parsing the *particular* structure of input file formats. Instead, tolerances of a standard format are added to existing fields within an input file.

As a demonstration of the power of this simple, natural language approach, a Monte Carlo sensitivity analysis is performed for three disparate simulation codes: fluid dynamics (LAURA), radiation (HARA), and ablation (FIAT). Effort required to harness each code for sensitivity analysis was recorded to demonstrate the generality and flexibility of this new approach.

I. Introduction

The case studies in this paper use the Monte Carlo method[a] to find the sensitivity of various numerical simulation outputs with respect to model input parameters, but the key innovation is a very tiny Domain-Specific Language (DSL)[b] to express uncertainties, herein called a tolerance DSL.

The goals of this work were to develop a method of specifying uncertainties that could be readily used with a wide range of numerical simulation codes, and to create a persistent, in situ means to document model parameter uncertainties.

The following sections map out the approach of both the tolerance DSL and the Monte Carlo sensitivity analysis framework. These sections are followed by statistical convergence, correlations, and sensitivities for three disparate simulation applications.

II. Approach

The basic premise is to craft a "tolerance" and "tagging" mini-language that offers a natural, unobtrusive presentation and does not depend on parsing each type of input file format. Ruby, a dynamic programming language well suited to crafting DSLs,[c] fuels this approach, along with the software techniques of templating and regular expressions.[d]

To prepare for the Monte Carlo method, each input file is marked up with tolerances and associated tags that serve to label the input parameters (the tags) and their uncertainties (tolerances of the form `+/- 0.05`). Then, for each Monte Carlo point, these "fuzzy" files are sampled by replacing the marked-up parameters with random samples from a specified distribution, e.g., Normal or Uniform. These samples are recorded for subsequent data reduction, e.g. correlations between inputs and outputs.

The next subsection documents the design evolution of the tolerance mini-language, while the following subsection describes the details of the Monte Carlos framework that uses this mini-language.

[*]In the electronic version of this document, dark blue text indicates a hyperlink.

[a]For a general discussion of the Monte Carlo method, see wikipedia.org/wiki/Monte_Carlo_method, last accessed May 20th, 2007.

[b]See en.wikipedia.org/wiki/Domain-specific_programming_language, last accessed May 20th, 2007.

[c]Ruby comes installed on most platforms. For the remainder, packages or source can be obtained from ruby-lang.org free of charge.

[d]See wikipedia.org/wiki/Template_(software_engineering) and wikipedia.org/wiki/Regular_expression, last accessed May 21st, 2007.

A. A Tolerance Domain Specific Language

This section follows the evolution of the tolerance DSL developed for sensitivity analysis, which ultimately has the form `2.3 +/- 5% 'Name'`.

The initial design used Ruby's templating system, ERB,[e] where each input parameter of interest would be replaced by a specially marked field. These would be replaced by randomly sampled values during the Monte Carlos process.

For example, if an input file that had a line like,

```
parameter = 5.0
```

it would be templated like so,

```
parameter = <%= 5.0.pm 10, :percent %>
```

where the `pm 10, :percent` was to be read like a tolerance, i.e., "plus or minus 10 percent". Behind the scenes, `.pm` was a Ruby method call on the number (`5.0`) that took two arguments: a tolerance (`10`) and the type of tolerance (`:percent`).

Upon using this for only a couple of hours, however, all the dots, colons, percents, equals signs, and angle brackets were not only ugly but tedious to type. In addition, a tagging requirement appeared,[f] and adding yet another parameter to the `pm` method was just too much—this design had run out of steam.

After reflection and another design session,[g] Ruby's ERB templating system was abandoned in favor of a dedicated, natural tolerance language like that used on an engineering drawing. At this point, the DSL looked like,

```
1 +/- 0.5
5 +/- 10%
2 +/- 1o
```

where `%` signified percent and `o` signified order of magnitude. Tags necessary for tracking the input-output correlations could be added by placing a quotation-mark-delimited tag after the tolerance, e.g., `0.7 +/- 20% 'T_effective'`.

True to the nature of software development, another requirement appeared late in the game: tolerances might have different underlying distributions, e.g., Uniform or Normal. With the new design, which adhered to the Open-Closed Principle,[h] this change was trivial to accommodate by optionally appending a token that specifies the distribution type to use for the tolerance,[i]

```
0.7 +/- 0.2U
1.4 +/- 20%N
```

A default distribution can be specified, so `U` or `N` are only necessary for special cases.

Finally, when applying the approach to an older simulation code, yet another requirement was unearthed: some number fields have to be in a specified format. Originally, reverse engineering the existing field was attempted,[j] but edge cases[k] undermined the robustness of this approach.

Upon reflection and an impromptu brainstorming session,[l] extending the current grammar using `C`-style format specifiers was chosen. The progression went as follows, where `%4.1f` is a `C`-style format specifier:

```
1.35 +/- 10U 'tag' '%4.1f'
1.35%4.1f +/- 10U 'tag'
%4.1f 1.35 +/- 10U 'tag'
1.35:%4.1f +/- 10U 'tag'
1.35::%4.1f +/- 10U 'tag'
1.35 +/- 10U:%4.1f 'tag'
1.35 +/- 10U 4.1f 'tag'
1.35 +/- 10U_4.1f 'tag'
1.35 +/- 10U_f4.1 'tag'
1.35 +/- 10U_4.1f 'tag'
```

[e]ERB stands for Embedded Ruby—see www.ruby-doc.org/stdlib/libdoc/erb/rdoc/ for documentation, last accessed May 20th, 2007.

[f]Tagging is necessary for labeling input-output correlations.

[g]Bill Wood, NASA, personal communication.

[h]See Ref. 1 and wikipedia.org/wiki/Open/closed_principle, last accessed May 20th, 2007.

[i]See the Uncertainty Distributions section of the appendix on page 10 for a short discussion of choosing distributions.

[j]See blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/249911, last accessed May 20th, 2007.

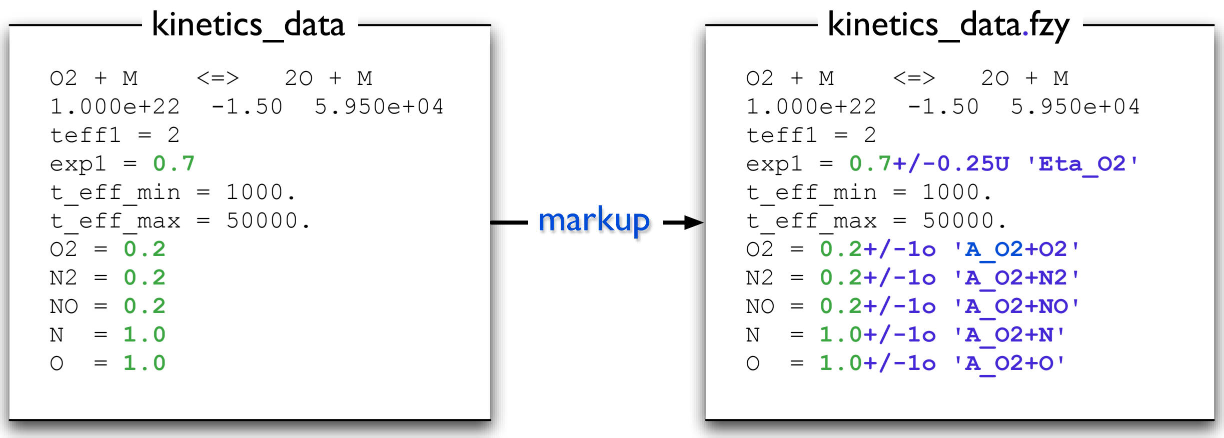
[k]See wikipedia.org/wiki/Edge_case, last accessed June 2nd, 2007.

[l]Bill Wood, NASA, personal communication.

Note the absence of the `%` character in the last 4 of the series. While necessary to be a valid `C`-style format specifier string, it is not necessary for the DSL—remember we get to make the rules in DSL-land. Also note the wobble toward the end: `4.1f` to `f4.1` and back. This came from a discussion about the choice between Fortran-style formats (`f4.1`) and `C`-style (`4.1f`) formats. Subtle, but the underlying Ruby code uses the `C`-style format, so choosing the latter avoids an error-prone translator and yet another configuration option.[m]

This tolerance DSL was implemented test-first[2] in 58 lines of Ruby via 165 lines of unit tests. The underlying probability distributions library is 57 lines of code via 90 lines of unit tests. Extending the DSL to accommodate format specifiers took 2 hours, a unit test, and 4 lines of code.

For example, consider figure 1, the markup of a portion of a chemical reaction input file per the parameter



```
O2 + M    <=>   2O + M
1.000e+22  -1.50  5.950e+04
teff1 = 2
exp1 = 0.7
t_eff_min = 1000.
t_eff_max = 50000.
O2 = 0.2
N2 = 0.2
NO = 0.2
N  = 1.0
O  = 1.0
```

```
O2 + M    <=>   2O + M
1.000e+22  -1.50  5.950e+04
teff1 = 2
exp1 = 0.7+/-0.25U 'Eta_O2'
t_eff_min = 1000.
t_eff_max = 50000.
O2 = 0.2+/-1o 'A_O2+O2'
N2 = 0.2+/-1o 'A_O2+N2'
NO = 0.2+/-1o 'A_O2+NO'
N  = 1.0+/-1o 'A_O2+N'
O  = 1.0+/-1o 'A_O2+O'
```

Figure 1. Sample tolerance and tagging markup.

tolerances listed in table 1 on page 5. Not only does the DSL provide a natural method of specifying tolerances, but it also servers as in situ documentation of model uncertainties.[n] Thanks to this tiny DSL, input files (or even code) are now easily marked up with tagged tolerance fields in a natural, readable way. And unlike prior art, a special parser for each input file is no longer required.

B. Monte Carlo Framework

The tolerance DSL provides the core mechanism to tag and specify parameter uncertainties and their distributions. The next step is to wrap these "fuzzy files" with a Monte Carlo process.

The basic steps of a Monte Carlo method are as follows:

1. Pick a simulation case to study (defines scope).

2. Assign uncertainties to model input parameters.

3. Randomly sample inputs, run, and repeat until the statistics have stabilized.

4. Compute input-output correlations, scatter plots, sensitivities, and other quantities of interest.

Each fuzzy file is randomly sampled to provide a given realization, the underlying code is run, and these steps are repeated until statistical convergence of the outputs is reached. The overall work flow is depicted in figure 2 on the following page.

This architecture was implemented with 120 lines of Ruby and configured with a small YAML file.[o] An example configuration file is shown in listing 1 on page 5 for the ablation application considered on page 9.

Apart from the configuration file, the user must provide a command that parses each output file specified and returns name-value pairs on standard output. For an example of an output file parser, see listing 2 on page 5.

[m]Value convention over configuration. For discussion of this tradeoff, see softwareengineering.vazexqi.com/files/pattern.html.

[n]This tolerance DSL comes with a utility to strip the tolerances and tags, providing a path to the original input data file.

[o]In this case, the underlying text-based database is YAML, a very simple data serialization format—see yaml.org for more information.

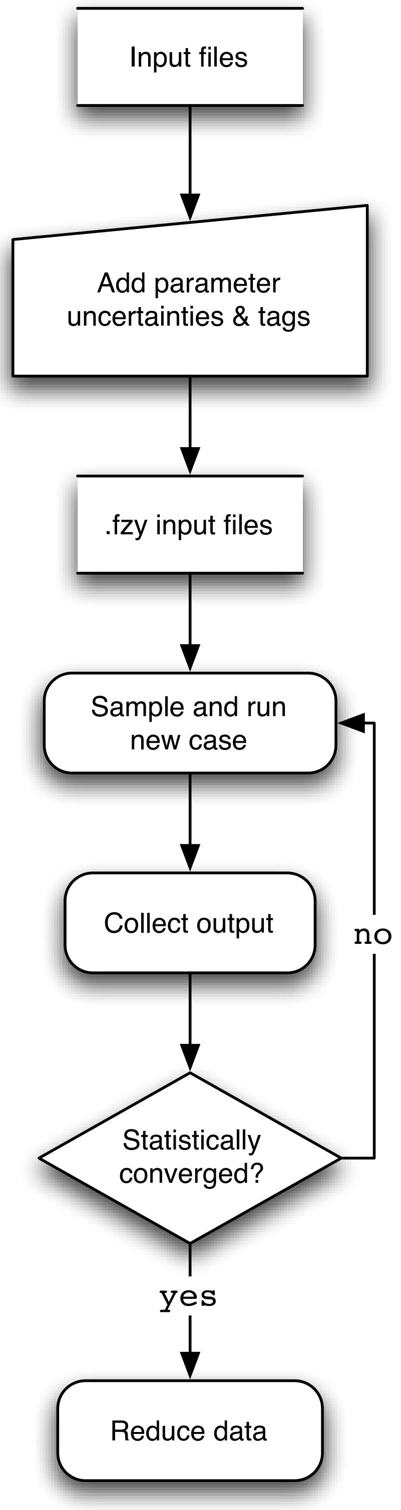


Figure 2. Simulation architecture.

Listing 1: Sample YAML configuration for the ablation application, page 9.

```
case_name   : fiat_pica_1
max_samples : 5000
nominal_dir : Nominal
require :
 - FIAT-v2.2.1
 - main.inp
 - envir.inp
execute : FIAT-v2.2.1 >& stderr.out
outputs :
 - surf.out : parse_surf.rb

cluster_available : true
node_prefix   : p1n
nodes         : !ruby/range 1..20 # or [ explicit array ]
jobs_per_node : 4
```

Listing 2: Sample output parser for the radiation application, page 7.

```
contents  = File.read('r1643.m')            # load output file
variables = contents.                       # store variable
              scan(/^\s*(\w+)\(/).flatten   # rootnames in array
node = 0                                    # inintialize node index
contents.split("\n").each do |line|         # begin line loop
  values = line.split.map{ |e| e.to_f }     # array of floats
  next unless values.size==variables.size   # skip extraneous lines
  node += 1                                 # increment node index
  variables.each_with_index do |v,i|        # begin variable loop
    name = "#{v}_#{node.to_s.rjust(3,'0')}" # add index to rootname
    puts "#{name} #{values[i]}"             # emit name-value pair
  end                                       # end variable loop
end                                         # end line loop
```

III. Applications

Initially, this system of obtaining sensitivities through the Monte Carlo method was focused on a single computational fluid dynamics code, but as the design evolved, the generality of this new approach was recognized. To discover the extent of this flexibility, the following sections document the application of the system to three disparate simulation codes spanning fluid dynamics, radiation, and ablation.

A. Computational Fluid Dynamics

LAURA,[p] configured for the laminar Reynolds-Averaged Navier-Stokes equations with Park's 5-species air model[q] was used to compute the stagnation point heating of a 3.5 m radius hemisphere flying at Mach 20.5 with a Reynolds number of 188,000.[r] Physical model parameters and estimated uncertainty ranges were taken from Palmer[7] and shown in table 1.

Table 1. Laura model parameter uncertainties.

Reaction rates	± 1 order
Vibration-dissociation	$0.7 \pm 0.25^*$
Thermal-relaxation	$\pm 10\%$
Collision integrals	$\pm 30\%$

* With Uniform distribution to assure samples fall in $[0, 1]$.

After generating a nominal solution to use as a restart and spending 4 hours marking up the input files for chemical kinetics, transport properties, and vibrational relaxation rates, 5,000 realizations were run over

[p]LAURA stands for Langley Upwind Relaxation Algorithm and is described in Refs. 3 and 4.
[q]See Ref. 5 plus updates from Ref. 6.
[r]This case represents the peak heating point of a generic International Space Station ballistic return trajectory.

the course of four days on a 4-node cluster of dual-CPU computers.[s]

Figure 3 shows stagnation-point heating-rate statistics as a function of samples. For this case, the median

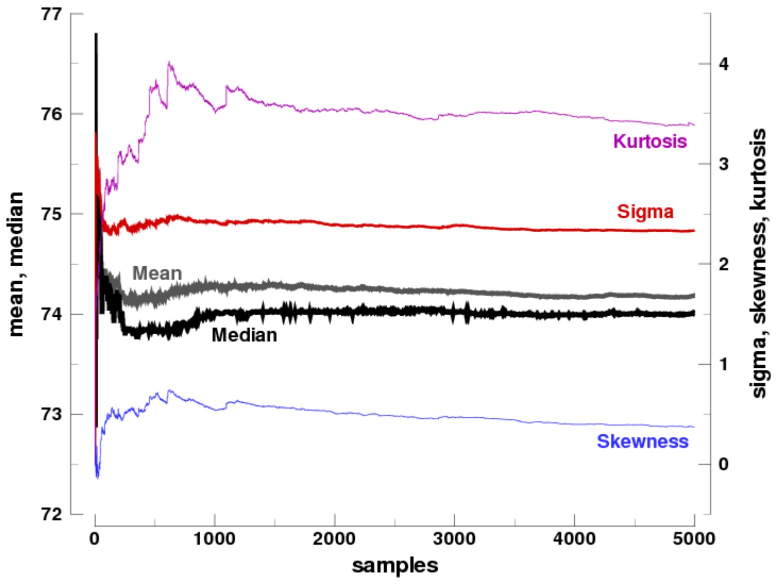


Figure 3. Stagnation point heating statistics as a function of samples.

stabilizes within the first 1,000 samples while the other moments are still gradually changing even at 5,000 samples. The output distribution not quite Normal as the Kurtosis is hovering above three and the skewness is non-zero. This could be due in part to the Uniform distribution applied to the thermal relaxation exponent to prevent samples from becoming non-physical.

The top 10 correlations between the stagnation point heating output and the model input parameters are shown in table 2. For this case, the tags of the form `Omega_ii_A:B` were given to collision cross-section

Table 2. Top 10 correlations for the Laura case.

Corr.	Tag
0.52	Omega_11_N2:O
0.32	Omega_11_N2:N
0.03	Omega_22_N2:O
0.03	Omega_11_N:O
0.02	Omega_11_N2:N2
0.05	Omega_22_N2:N2
0.01	Omega_11_O2:O
0.01	Omega_11_N2:O2

coefficients. The collision cross section of nitrogen molecules with respect to atomic oxygen and nitrogen are the largest, *linear* contributors to variations of stagnation point heating.[t]

Finally, the stagnation point heating sensitivities are shown as function of radial distance for the hemisphere in figure 4 on the following page. Note: the "error bars" are not symmetric because the output distribution is not Gaussian. The error bars are taken from the cumulative distribution 2.5 and 97.5 percentile points, giving the equivalent of a 2-sigma probability range, i.e., a 95% confidence interval.

[s]Automated statistical convergence detection had not yet been implemented.

[t]As Palmer argues for a similar case, these findings are expected because these collision cross sections control diffusion, viscosity, and conduction—all of which are important for this dissociated flow where heating is driven by diffusion-rate-limited recombination at the fully-catalytic wall. For more discussion, see Ref. 7.

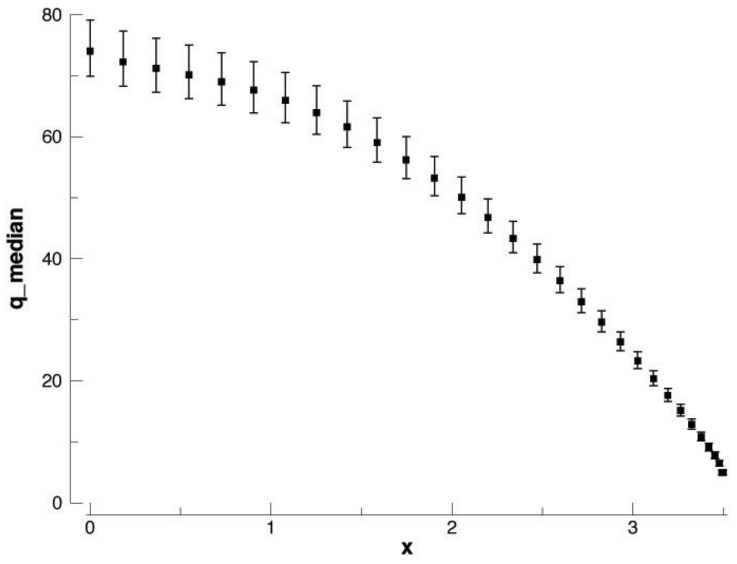


Figure 4. Two-sigma surface heating sensitivities for hemisphere.

B. Radiation

The Hara code[u] was used to compute the stagnation point radiative heat flux for the Fire II experiment at 1643 seconds.[9]

The input parameter uncertainties chosen for this demonstration were:

1. The atomic line oscillator strengths — uncertainties for each of these (over 300 considered) are published by NIST[v] and range from 3 to 25%.

2. The atomic and molecular electron-impact excitation rates (over 1,000 considered) for the non-Boltzman model were assigned order-of-magnitude uncertainties.

It took 8 hours to harness the Hara code, including marking up the input decks with uncertainties and tags, adapting the original system for re-use, and creating a small preprocessor used to mark up large matrices of input values with a single command.[w] 5,000 Hara samples took just under an hour to run on 20 dual-CPU cluster.

Figure 5 on the next page shows stagnation-point radiative heating-rate statistics as a function of samples. With final Kurtosis of three and a skewness of zero, the output distribution is Normal. This distribution was reached within 2,000 samples despite the large number of input variables sampled (1,500). This is due to the nature of this simulation code where many of the input parameters are not in play for a particular type of case.

The top 10 correlations for the Fire II 1643s case stagnation point radiative heating flux are shown in table 3 on the following page. The `N##` and `O##` each correspond to an atomic line as defined by Wiese et al.[10] The variability in `N46` is correlated with 18% of `q_m`'s variability, while `N69` is responsible for 12%, with other nitrogen lines contributing to a diminishingly smaller extent. Only the 3rd and 5th lines listed are located in the vacuum ultraviolet (VUV), which indicates a relatively small sensitivity to the VUV atomic lines.

The top 8 uncertainty contributors listed in table 3 are atomic-lines that Johnston had previously identified as the top contributors to the radiative flux for lunar-return shock layers.[8] They all have uncertainties of 10% (the strong lines that have only 3% uncertainties fall near the bottom of the list). Also, the non-Boltzmann rates contribute very little to the uncertainty (and do not appear in table 3). This is expected

[u]Hara stands for High-temperature Air Radiation and is described in Ref. 8.
[v]See physics.nist.gov.
[w]This preprocessor is shown in the Markup Helper section of the appendix, beginning on page 10.

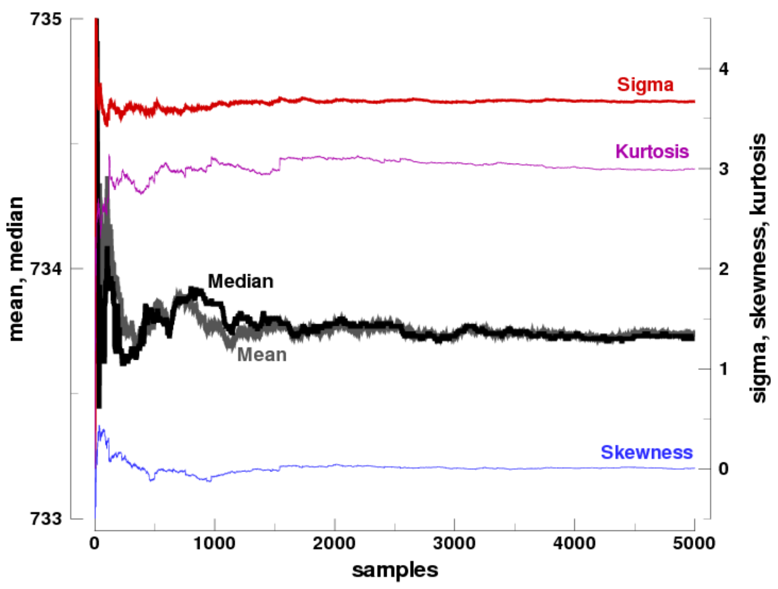


Figure 5. Stagnation point radiative heating statistics as a function of samples.

Table 3. Top 10 correlations for the Hara case.

Corr.	Tag
0.18	N46
0.12	N69
0.07	N15
0.07	N51
0.06	N29
0.03	N39
0.02	N52
0.02	N81
0.01	N12
0.01	O60

because the Fire II 1643s case is close to thermochemical equilibrium throughout most of the shock layer. The non-Boltzmann rates should be more significant for the relatively nonequilibrium Fire II 1636s case.

C. Ablation

The FIAT code[11] was used to compute the surface recession of a PICA tile subject to a 30-second arcjet pulse (one of the sample cases bundled with the code).

It took about 4 hours to harness the FIAT code, including extending the DSL to handle fixed-formatted fields and adding tolerances and tags to the input deck per table 4. Note: only materials involved with

Table 4. Fiat model parameter uncertainties.

Resin volume fraction	$\pm 5\%^*$
Decomposition rate	± 1 order
Virgin conductivities	$\pm 20\%$
Virgin emissivities	$\pm 0.05^*$
Char conductivities	$\pm 100\%$
Char emissivities	$\pm 0.10^*$
Pyrolysis enthalpies	$\pm 20\%$

* With Uniform distribution to assure samples fall in $[0, 1]$.

this case were marked up: PICA, RTV adhesive, and Al 2024. 5,000 FIAT cases required an hour and ten minutes to run on a 20 dual-CPU cluster.

Figure 6 shows statistics of the surface recession as a function of samples. While the output distribution

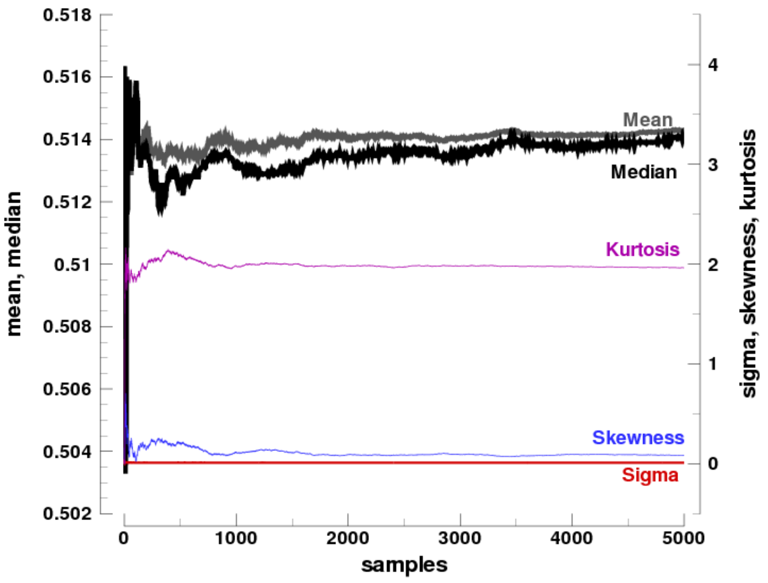


Figure 6. Surface recession statistics as a function of samples.

is only slightly skewed, it is less peaked than a Normal distribution, having a platykurtic Kurtosis of just under 2. This indicates that variability is characterized by frequent, modestly-sized deviations as opposed to infrequent, extreme variations. These modestly-sized deviations may be the reason that the mean and the median are still gradually rising even after 5,000 samples.

Table 5 on the next page shows the top 10 correlations for surface recession of the PICA sample. The variability in `PICA_(air)_EC13`, the char emissivity at 6,000 R, correlates with 90% of the variability in surface recession while everything else is largely uncorrelated. As shown in table 4, this quantity was was assigned an uncertainty of ± 0.10 with a Uniform distribution to prevent realizations outside the physically

Table 5. Top 10 correlations for the Fiat case.

Corr.	Tag
0.90	PICA_(air)_EC13
0.01	PICA_(air)_GAMA
0.01	PICA_(air)_EC12
0.01	PICA_(air)_KC12
0.004	PICA_(air)_KC11
0.002	PICA_(air)_KC13
0.002	PICA_(air)_KC09
0.002	RTV-560_KC08
0.001	RTV-560_KC22
0.001	PICA_(air)_BHp2_04

meaningful range of $[0, 1]$. This single, predominate correlation is plausible as the surface energy balance, i.e. how much energy gets into the TPS versus how much is re-radiated, is directly proportional to the emissivity.

IV. Concluding Remarks

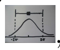
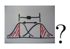
A lightweight, flexible sensitivity analysis capability has been demonstrated for three disparate simulation codes. Reactions from customers so far are positive and include, "This is a great step—shows the power of your DSL approach." and "That was fast! Sounds like you've developed a great system. This is very interesting and extremely useful." As shown by the demonstration section, the small, simple idea to standardize the format of the input parameter uncertainties as opposed to developing a new parser for each type of input file, allowed sensitivity analysis of entirely new classes of simulation codes in under a day with very few lines of code.

Contributing to the ability to rapidly adapt to new simulation codes was the extensive use of test-driven design. This technique, where no production code is written without a failing, automated unit test indicating production code needs to be written or changed, allowed confident, rapid evolution of the system design as new requirements were discovered. The battery of unit tests also serves as *executable* documentation of the system requirements, as opposed to code comments or static documentation that rapidly lose their efficacy when a system is subject to change.

Some areas of future work are clear: more thorough documentation, user testing, and error checking as well as a more formal system to facilitate tolerance markup, including a method to accommodate groups of inputs associated with table lookups. For the latter, applying a single tolerance to whole columns worth of inputs is one solution and could be accomplished by allowing duplicate tags.

Appendix

A. Uncertainty Distributions

What is meant by uncertainty, ? Typically this invokes thoughts of a normal distribution, . But should it be interpreted as a large-number statistics interpretation, , or a small-number statistics interpretation, ? If it is the later, then we need a new distribution function.

B. Markup Helper

To facilitate marking up tolerances for matrices of values used by the Hara code, the Ruby code shown in listing 3 on the next page converts `.mta` files to `.fzy` files. A `.mta` file recognizes order of magnitude tolerances of the form: `[i,j]+/-2o 'tag_#{i},{#j}'` where `#{i}` and `#{j}` are filled in with the appropriate indices for each value in the matrix of numbers.

Listing 3: Ruby code used to convert .mta to .fzy format.

```ruby
Dir['*.mta'].each do |file|
  File.open( file.sub(/mta$/,'fzy'), 'w' ) do |f|
    matrix_tolerance = false
    tol_string = ''
    i = j = 0
    IO.foreach(file) do |line|
      match = line.match(/\[i,j\]\+\/-\d+o\s*(['"]).*\1/)
      if match then
        matrix_tolerance = true
        tol_string = match.to_s[5..-1]
        next
      end
      if matrix_tolerance then
        fields = line.split
        if fields.empty? then
          matrix_tolerance = false
          tol_string = ''
          i = j = 0
          f.puts line
          next
        else
          i += 1
          fields.each_with_index do |e,m|
            j = m+1
            e = e << eval("\"#{tol_string}\"")
          end
          f.puts fields.join(' ')
        end
      else
        f.puts line
      end
    end
  end
end
```

Acknowledgments

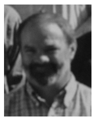
The authors would like to acknowledge: Grant Palmer of ELORET Corporation for samples of prior art and the following fellow NASA employees: Eric Walker for discussions concerning distributions in the absence of data, Michael Hemsch for remedial statistics lessons, Brian Hollis, Ron Merski, Mike Wright, the Vision for Space Exploration for a challenging problem and the freedom to solve it, Clyde Gumbert and James Moss for manuscript reviews, and Bill Wood for software design and implementation discussions.

About the Authors

Bil Kleb, a lifetime AIAA member, has worked in the area of computational aerothermodynamics at NASA's Langley Research Center for the past 17 years. During this time, he pioneered the first full-vehicle reentry simulation of the Shuttle Orbiter and earned an MBA from The College of William and Mary and a PhD of Aerospace Engineering from the University of Michigan.

Since 1999, Bil has been a part of the Agile software development community,[x] and for the past five years, a steward of the FUN3D software development team.[y]

Email: Bil.Kleb@NASA.gov



Richard Thompson, is a senior AIAA member with a degree in Aerospace Engineering from Virginia Tech. He has worked in NASA Langley Research Center's Aerothermodynamics Branch since 1983.

Email: R.A.Thompson@NASA.gov

Christopher Johnston, an AIAA member with a Ph.D. in Aerospace Engineering from Virginia Tech, has been a member of NASA Langley Research Center's Aerothermodynamics Branch since 2006.

Email: Christopher.O.Johnston@NASA.gov

[x]agilemanifesto.org, last accessed May 20th, 2007.
[y]fun3d.larc.nasa.gov, last accessed May 20th, 2007.

Colophon

This document was typeset in Computer Modern font with the LaTeX typesetting system using the `handout` option of the AIAA package,[z] version 3.8. Also employed were the `array`, `booktabs`, `color`, `fancyvrb`, `hyperref`, `svninfo`, `threeparttable`, `varioref`, and `wrapfig` packages.

References

[1]Robert C. Martin, *Agile Software Development, Principles, Patterns, and Practices*, Prentice Hall, 2002.

[2]Kent Beck, *Test Driven Development: By Example*, Addison-Wesley Professional, 2002.

[3]Peter A. Gnoffo, et al., *Conservation Equations and Physical Models for Hypersonic Air Flows in Thermal and Chemical Nonequilibrium*, NASA TP 2867, 1989.

[4]Peter A. Gnoffo, *An Upwind-Biased, Point-Implicit Relaxation Algorithm for Viscous, Compressible Perfect-Gas Flows*, NASA TP 2953, 1990.

[5]Chul Park, *Nonequilibrium Hypersonic Aerothermodynamics*, Wiley-Interscience, 1990.

[6]Chul Park, "Review of chemical-kinetic problems of future NASA missions. I—Earth entries", *AIAA Journal of Thermophysics and Heat Transfer*, 7(3), 1993, pp. 385–398.

[7]Grant Palmer, *Uncertainty Analysis of CEV LEO and Lunar Return Entries*, AIAA Paper 2007–4253, 2007.

[8]Christopher O. Johnston, *Nonequilibrium Shock-Layer Radiative Heating for Earth and Titan Entry*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 2006.

[9]Christopher O. Johnston, et al., *Nonequilibrium Stagnation-Line Radiative Heating for Fire II*, AIAA Paper 2007-3908, 2007.

[10]W. L. Wiese, et al., "Atomic Transition Probabilities of Carbon, Nitrogen, and Oxygen: A Critical Data Compilation", *Journal of Physical Chemistry Reference Data*, 1996.

[11]Y.-K. Chen and Frank S. Milos, "Ablation and Thermal Response Program for Spacecraft Heatshield Analysis", *Journal of Spacecraft & Rockets*, 36(3), 1999, pp. 475–483, also AIAA Paper 98–0273, Jan. 1998.

[12]Edward R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, 1983.

[13]Edward R. Tufte, *Envisioning Information*, Graphics Press, 1990.

[14]Edward R. Tufte, *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press, 1997.

[15]Edward R. Tufte, *Beautiful Evidence*, Graphics Press, 2006.

[16]Richard Phillips Fyenman, *Lectures in Physics*, Addison Wesley Longman, 1970.

[z]www.ctan.org, last accessed May 20th, 2007.