Response to Richard Tol's Comment on my Prague Lecture

William Nordhaus, Yale University
September 11, 2012

Richard Tol has proposed a correction to the slides that accompanied my keynote lecture at the Prague meetings of the EAREA in June 2012.[1] This note is written primarily so that readers can understand the context of the issues.

In my lecture, I devoted considerable time to the issue of computational complexity in integrated assessment models (IAMs). My concerns arise because of the increasing size and complexity of computerized modeling in environmental sciences and economics. Specialists in software architecture have studied the issues involved in developing large programs and emphasize the difficulties of ensuring that software is reliable and well-tested. A rule of thumb is that well-developed software contains in the order of 1 error per source line of code (SLOC). Since many computerized climate and integrated assessment models contain between 10,000 and 1 million SLOC, there is the prospect of many bugs contained in our code.[2]

This proposition is not just theoretical. There are many examples of catastrophically bad software, such as the errors that led to the crashing of a spacecraft because of insertion of a period instead of a comma in a FORTRAN statement; or inappropriate shutdown of five nuclear power reactors because of an incorrect formula programmed. Current luxury automobiles have millions of lines of code and probably contain untold thousands of bugs (travelers beware!).

---

[1] The slides as delivered are available at http://www.econ.yale.edu/~nordhaus/homepage/documents/Prague_June2012_v4_color.pdf.

[2] Useful references are Paul Clements et al., *Documenting Software Architectures: Views and Beyond,* Second Edition, Addison-Wesley, 2010. For a story on software in automobiles, see "Cars and Software Bugs," *The Economist,* http://www.economist.com/blogs/babbage/2010/05/techview_cars_and_software_bugs. For the bugs fixed in Windows 2000, see http://support.microsoft.com/kb/327194. For large climate models, see Kaitlin Alexander and Steve Easterbrook, "The Software Architecture of Global Climate Models," available at http://climatesight.files.wordpress.com/2011/08/poster.pdf. For catastrophic software errors, see Robert Glass, *Software Runaways* or *Computing Calamities,* or for more recent studies, W.E. Wong et al., "Recent Catastrophic Accidents: Investigating How Software Was Responsible," 2010 Fourth IEEE International Conference on Secure Software Integration and Reliability Improvement.

I noted that these problems are ones that are likely to arise for IAMs as well because they are large and complex. I drew on three examples in my discussion: one from the OECD Green model,[3] one from the FUND model, and one from coding errors in my own programming.

Tol is responding to my comments about the FUND model. This is one of the leading models used by researchers and governments to understand the economics of global warming. It has been used to calculate the social cost of carbon for the U.S., which calculation affects tens of billions of dollars of regulations.

The problem with the FUND model I discussed concerned a formula for one of the components of the damage function. The specification had agricultural damages, which were calculated with a formula having a normal variable in both the numerator and the denominator. This was pointed out in an article by Ackerman and Munitz, which made the following statement: "The manner in which the optimum temperature effect is modeled in FUND 3.5 could cause division by zero for a plausible value of a Monte Carlo parameter." [4] This led to a controversy about what "division by zero" means, and to Tol's response to their article and to my lecture.

It will be useful to examine the issue from a statistical point of view. The details are the following: In FUND 3.5, according to the model description, the damages for the level of temperature on agriculture have two terms. The first term of the damage component can be written as $y = az/(b-z)T$, where y = damages, T = temperature (an endogenous variable); a and b are parameters ; and z is a random variable, which in the FUND model is $(T-T^{opt,r})$. The variable $T^{opt,r}$ represents the optimal temperature in region r and is a normally distributed random variable, so y is the ratio of two normal variables. [5]

---

[3] This can be seen in greater detail in W. Nordhaus, "Integrated Assessment Models…" forthcoming, *Handbook of CGE Models,* available at cowles.econ.yale.edu/P/cd/d18a/d1839.pdf.

[4] See F. Ackerman and C. Munitz, "Climate damages in the *FUND model*: A disaggregated analysis," *Ecological Economics*. 2012.

[5] This was pointed out but not emphasized in Ackerman and Munitz. For the underlying statistics, see for example George Marsaglia, "Ratios of Normal Variables," *Journal of Statistical Software,* May 2006, Volume 16, Issue 4, pp. 1-10.

The ratio of two normal distributions with non-zero means is a non-central Cauchy distribution. A non-central Cauchy distribution has a standard Cauchy term and another complicated term, but we can focus on the Cauchy term. This distribution is "fat tailed" and has both infinite mean and infinite variance. So the level damage from agriculture in FUND 3.5 (from a statistical point of view) will dominate both the mean and dispersion of the estimated damages. Taken literally, the expected value of damages to agriculture are infinite at every temperature increase. This is subject to sampling error in finite samples of any size, but the sampling error is infinite since the moments do not exist, so any numerical calculations with finite samples are (infinitely) inaccurate. There is also a coding issue because it is not possible to get an accurate estimate of the distribution of a variable with infinite mean and variance in finite samples. The most troubling impact of this specification is the estimate of the distribution of outcomes (such as the social cost of carbon or SCC). If the damages are a fat tailed distribution, then the SCC is also fat- tailed. In finite samples, of course, all the moments are finite, but the estimates are unreliable or fragile and depend upon the sample.

Tol indicates that they do a check of the outcomes both by inspection and by trimming the extremes. As an analytical matter, a trimmed distribution is even more complicated than the Cauchy, but it still will have an infinite mean and variance.

I assume that this strange distribution was not intended, and in any case is easily corrected. My point was not to dwell on the shortcomings of our models. Rather, we need to recognize that most economists and environmental scientists are amateurs at software design and architecture. As computers get faster, as software packages get more capable, as our theories get more elaborate – there is a tendency to develop models that increase in parallel with the rapidly expanding frontier of computational abilities. This leads to increasingly large and complex models. We need also to ask, do we fully understand the implication of our assumptions? Is disaggregation really helping or hurting?[6]

There is another lesson here about uncertainty analyses. Deterministic IA models are already complex non-linear systems. Introducing uncertainty through a set of complicated functions of random variables adds yet another layer of complexity. Modelers need to be

[6] For important studies of the statistics of disaggregation, see H. Theil, *Linear Aggregation of Economic Relations* and Grunfeld and Griliches, "Is Aggregation Necessarily Bad?" *ReStat,* 1960.

especially careful that they have not changed the properties and outcomes of the models because of strange behavior or interactions of the added random variables. The properties of linear stochastic systems are moderately well-understood, but that is not the case for all non-linear stochastic systems.

My recommendations here are four: First, we modelers need to recognize the importance good software architecture. Second, we should restrain the urge to develop ever larger and more complex computational models unless there is a clear case that they will improve our understanding. Third, we need to undertake special scrutiny when we add random elements to non-linear dynamic models. Finally, we need to take the extra time and effort to examine, re-examine, and test our software to try to wring out the errors.