Parallel Collocation for Uncertainty Quantification on High Performance Systems

John Burkardt
Department of Scientific Computing
Florida State University
https://people.sc.fsu.edu/~jburkardt/presentations/...
burgers_2012_ornl.pdf

02 May 2012

Burger's Equation

The Burgers equation, occasionally called *the poor man's Navier Stokes equation* has some interesting features:

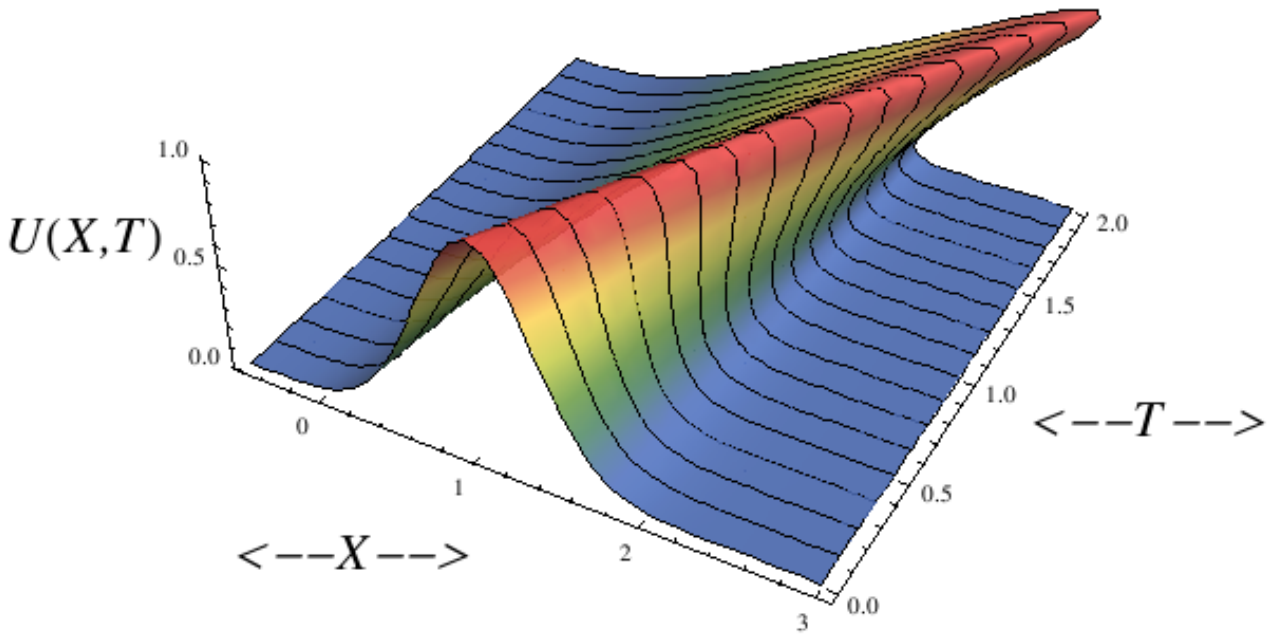
- includes a nonlinear term that can generate shocks and discontinuities;
- includes a smoothing term multiplied by a viscosity;
- definable as steady or time-dependent, viscid or inviscid.

The equation has the advantages that:

- it can be formulated in one spatial dimension x;
- it has only a single state variable $u(x)$ or $u(x,t)$.
- its simplicity makes it easy to define, solve, analyze, and plot.

Shock Waves for the Inviscid Time-Dependent Problem

Here is a computational (and nonphysical!) solution of an inviscid Burgers problem, in which the peak velocity overtakes the rest of the wave.



We'll look at viscous problems, in which this tendency is suppressed.

The Steady Viscous Burgers Equation

The steady viscous Burgers equation seeks a function $u(x)$ defined over an interval $[a, b]$, satisfying

$$u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

for which we might specify the Dirichlet boundary conditions

$$u(a) = \alpha; \quad u(b) = \beta.$$

The Conservation Form of Steady Burgers Equation

For technical reasons, it is preferable to rewrite the equation from its advection form to the conservation form:

$$\frac{1}{2}\frac{\partial u^2}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

While this doesn't change the mathematics at all, it does suggest a different discretization scheme.

Here, $0 < \nu$ is the viscosity. A high viscosity corresponds to a sticky fluid, suppressing shocks and discontinuities. As ν decreases, the fluid can support steep gradients. A discretized solution technique will need greater resolution to capture the behavior.

Problem Parameter Values

Typical problem data might be:

$$a = -1,\ \alpha = +1,\ b = +1,\ \beta = -1,\ \nu = 0.1$$

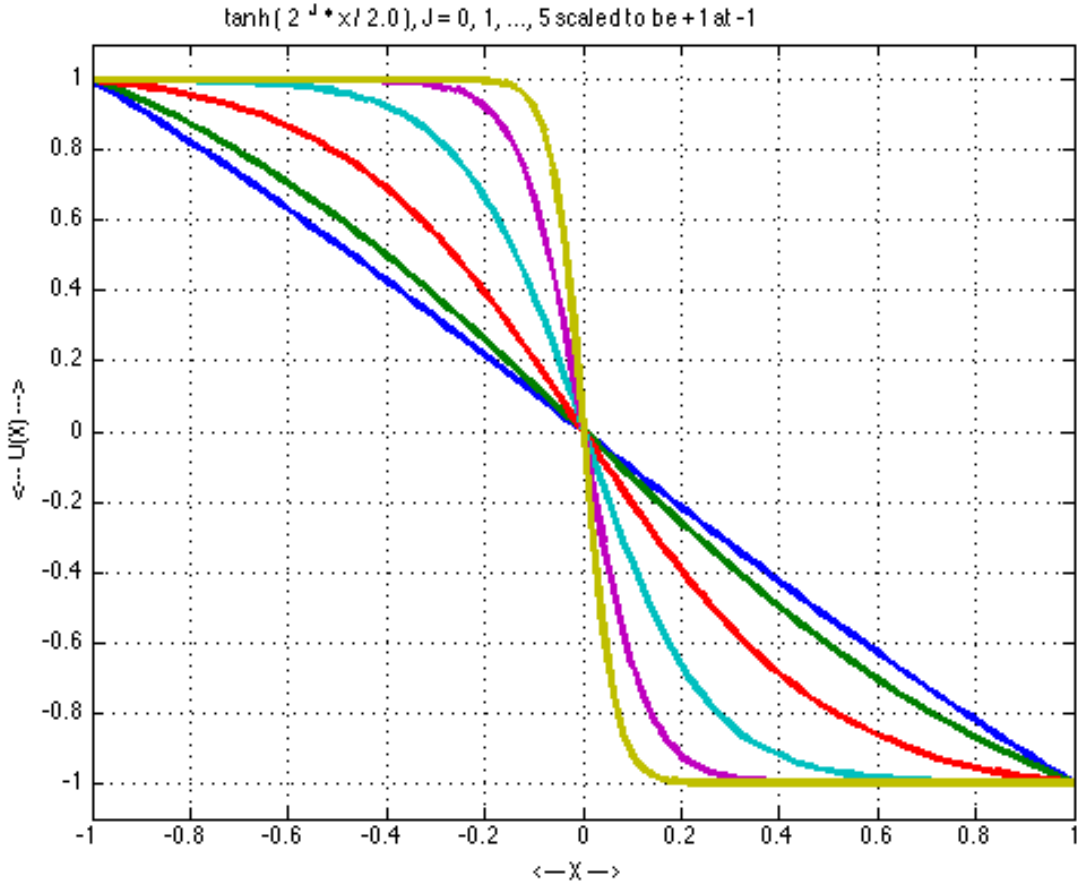
The specification of the values of these input parameters completes the definition of the analytic problem, and allows us to regard the solution $u(x)$ as a function of the parameters.

We will be interested in the relative importance of the influence of each parameter on the solution, and the effect of uncertainty in a parameter on the solution or on derived quantities of interest.

We'll use the values specified above as our "base data", and concentrate on solutions associated with relatively small perturbations of this data.

Solution Family For Varying Viscosity

For the given symmetric boundary conditions, the analytic solution depends on the viscosity, and has the shape of scaled **tanh(x)** function. Here is the kind of behavior we can expect from solutions to the exact equation for a variety of viscosity values.



Discretized Problem

A simple discretized version of the Burgers equation might use $m+1$ equally spaced nodes with spacing $dx = \frac{b-a}{m}$, with typical coordinate x_i, and discretized solution value u_i.

Since this is a nonlinear problem, we can construct system of equations $\vec{f}(\vec{u}) = 0$ that must be satisfied by the discrete solution; we can apply Newton's method to seek a solution.

At the first and last nodes, we impose the boundary conditions. At the interior nodes, we require the discrete solution to satisfy a discretized version of the Burgers equation.

The Steady Viscous Burgers Equation

Using $m+1$ evenly spaced points x_i, our discretized system is:

$$\begin{aligned} f_1 &= u_1 - \alpha \\ f_i &= \frac{1}{2}\frac{u_{i+1}^2 - u_{i-1}^2}{2dx} - \nu\frac{u_{i+1} - 2u_i + u_{i-1}}{dx^2}, \quad i = 2, \ldots, m \\ f_{m+1} &= u_{m+1} - \beta \end{aligned}$$

It is easy to write down the associated Jacobian matrix, and if we use as a starting point the linear interpolant to the two known boundary values, we can carry out the Newton procedure to obtain a solution.

The Quantity of Interest

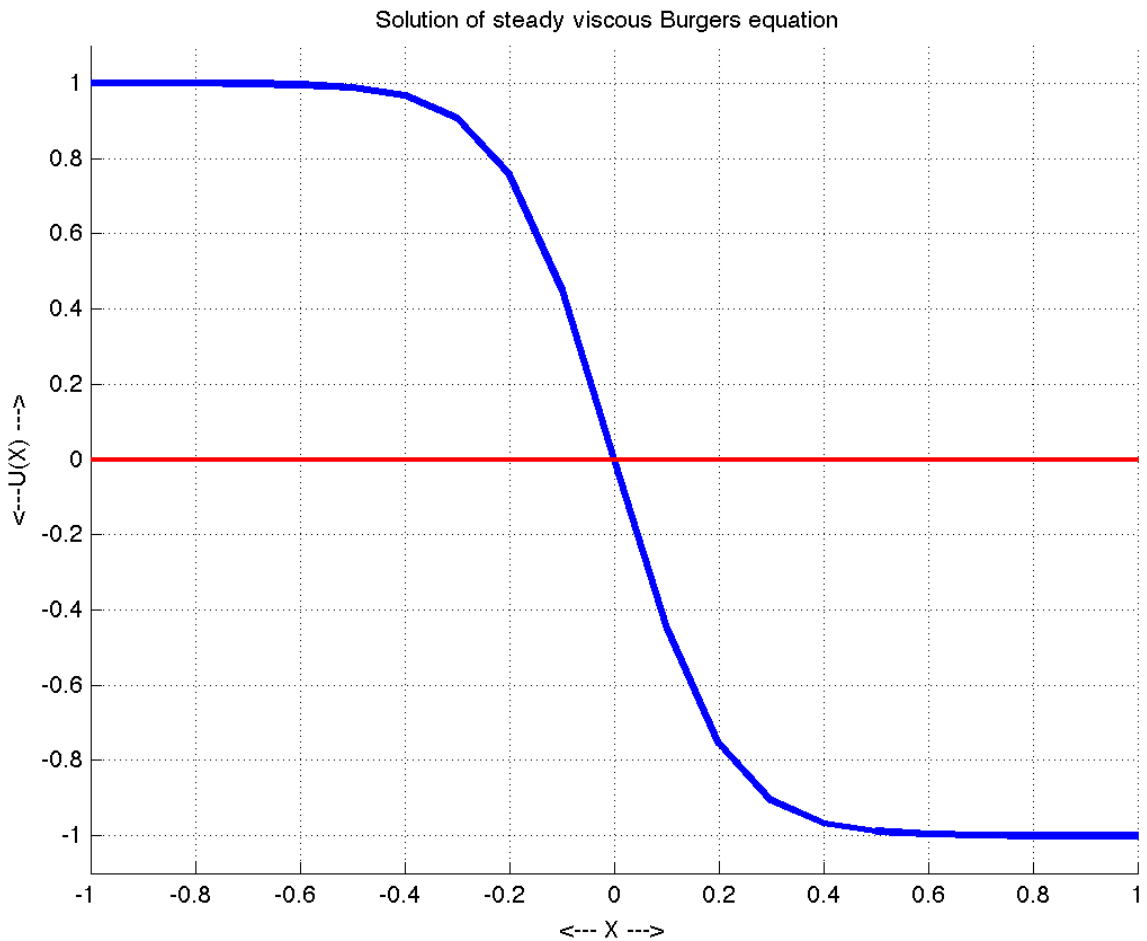
It's natural to focus on the solution function $u(x)$ as the most important object in the computation, but for many computations, one or more quantities of interest, derived from $u(x)$, might be the actual goal of the computation.

Such quantities can include the integral of the solution, the maximum deviation from some prescribed value, the lift or drag of an airfoil, the breaking point of a beam, or the total expenditure of fuel.

For this study, suppose the quantity of interest is the point x_0 where the solution changes sign. Since our solution is discretized, we'll use its linear interpolant to define x_0.

Our "base" solution

Here is our computed "base" solution for the parameter values $a = -1, \alpha = +1, b = +1, \beta = -1, \nu = 0.1$.
The value of the quantity of interest is $x_0 = 0$.



Initial Sensitivity Analysis

We have our solution, but it depends on the parameter values we chose. If we imagine there are errors or uncertainties in this data, our computed solution will differ from the actual one.

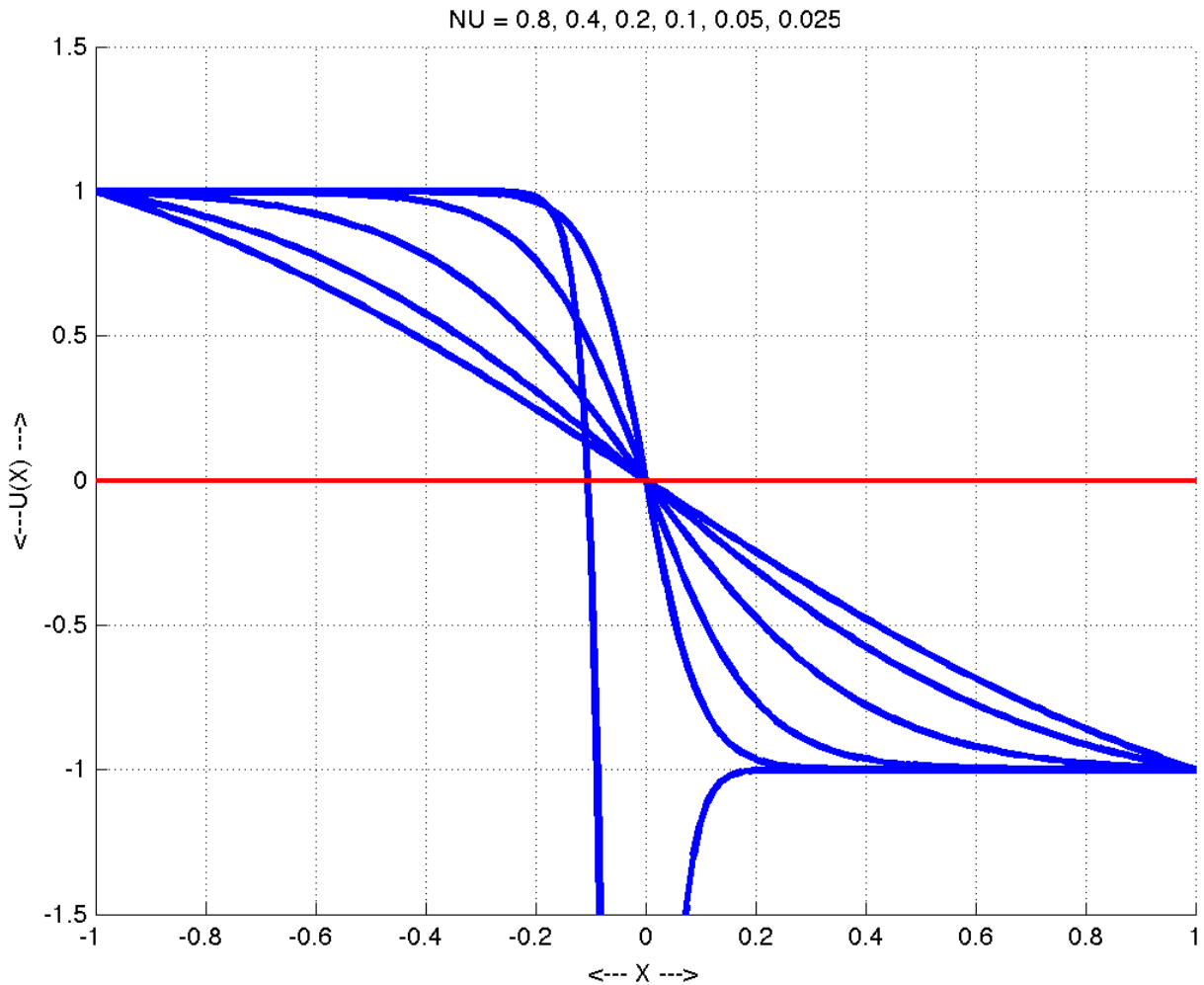
Can this effect be large for the types of errors we expect? Can we describe the types of errors we expect? Are some parameter errors more serious than others?

We can start with a crude sensitivity analysis, slightly modifying one base parameter at a time, and recomputing the solution. This suggests the strength of the dependence of u on each parameter, and thus the relative importance of each parameter.

This will suggest where our uncertainty investigation should focus.

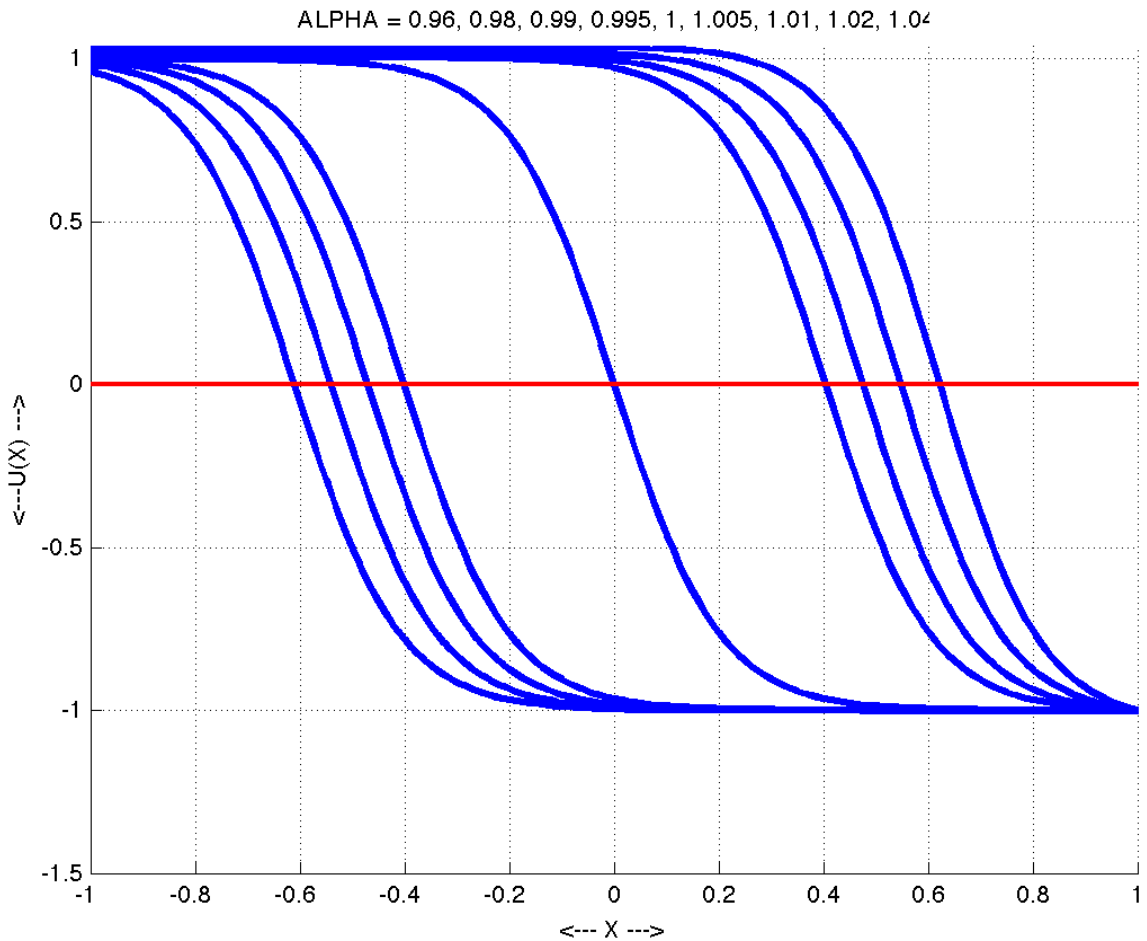
Sensitivity to ν, the Viscosity

Varying viscosity affects the flow . . . but not x_0!
The peculiar result for $\nu = 0.025$ is because of nonconvergence.



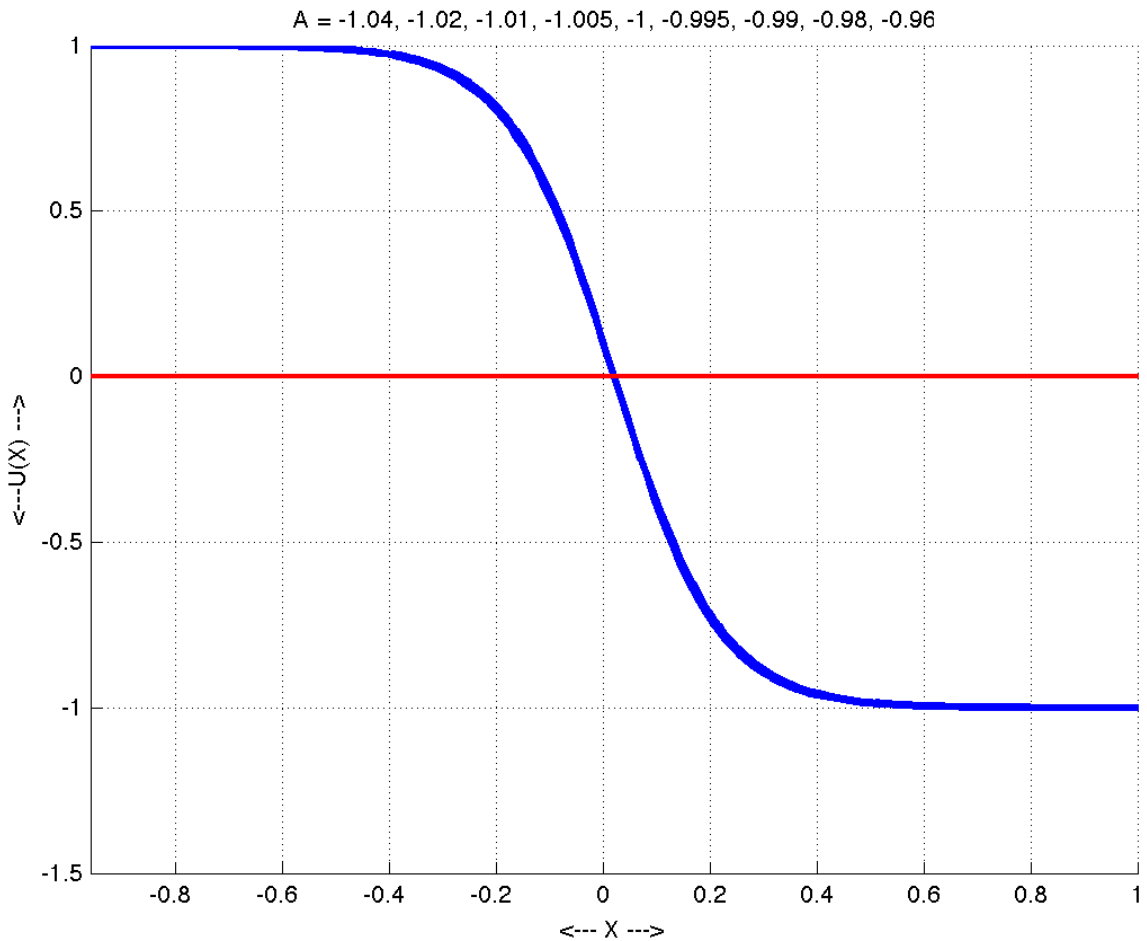
Sensitivity to α, the Left Hand Value

α determines the solution at the left endpoint a. The computed solution u is surprisingly sensitive to this quantity. Changing α from 1 to 1.005 is enough to make a startling jump in x_0.



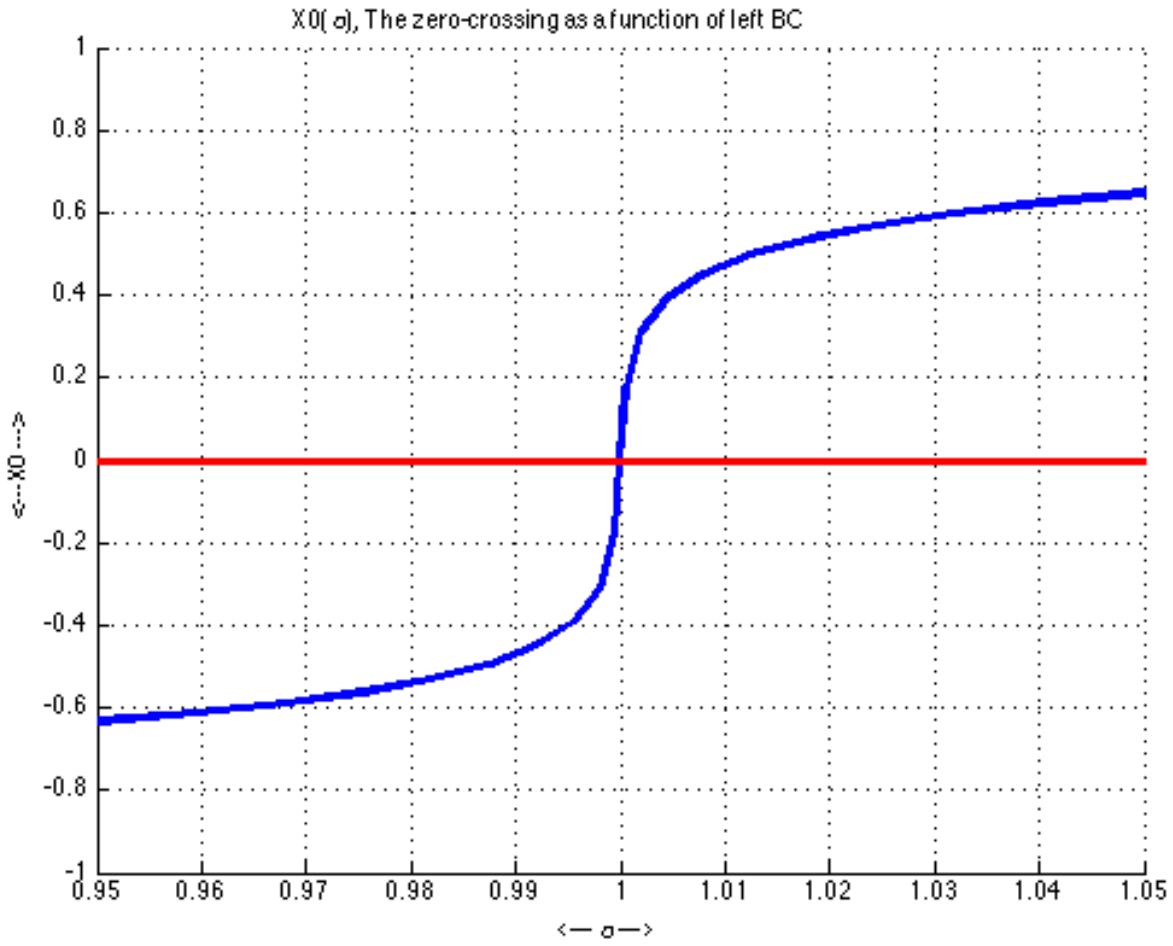
Sensitivity to a, the Left Endpoint

Our solution does not seem very sensitive to the value of a, the location of the left endpoint. For "reasonable" perturbations, $u(x)$ seems to change not at all!



Computing $X_0(\alpha)$

Here is a plot suggesting the behavior of x_0 as a function of α, verifying the extreme sensitivity near the base value.



Expected Value Estimate

If we suppose only α is uncertain, then the symmetry of our results suggests that $\mathbb{E}(x_0(\alpha))$ will be roughly 0, an uninteresting result. If we generate Gaussian deviates of α with mean 1 and standard deviation of 0.05, our estimates

M	E(X0(ALPHA)) estimate
16	-0.0262784
32	0.00325575
64	-0.0115093
128	0.00144016
256	0.0255031
512	0.0222146
1024	-0.0142951
2048	0.00458531
4096	-0.000209035

Variance Estimate

The variance estimate shows that uncertainty in α means we can expect crossing perturbation magnitudes of about 0.5, that is, halfway to the boundary.

M	Var(X0(ALPHA)) estimate
16	0.351927
32	0.343577
64	0.348422
128	0.335153
256	0.341199
512	0.346304
1024	0.341493
2048	0.348165
4096	0.346826

For this problem, the strong variance in x_0 persists even if we reduce the variance of α, or if we model α by a uniform deviate.

Discussion

The plot of $x_0(\alpha)$, and the computations of $\mathbb{E}(x_0(\alpha))$ and $\sigma^2(x_0(\alpha))$ were done by evaluating the full state solution at hundreds or thousands of values of α.

The resulting information is useful, but we really only investigated uncertainty with respect to a single parameter, on a simple 1D problem.

In practical problems, we expect that each state solution will be quite expensive. This alone might suggest using some kind of **interpolation scheme** to build a polynomial model of $x_0(\alpha)$ based on a much reduced number of sample evaluations.

However, practical problems are also likely to have tens (or even hundreds) of uncertainty parameters to investigate simultaneously, meaning our workload has the potential to explode.

The Time-Dependent Burgers Equation

The time dependent viscous Burgers equation is:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

Now we'll take periodic boundary conditions:

$$u(a,t) = u(b,t) \text{ for } t > 0;$$

and we specify an initial condition for $t = 0$:

$$u(x,0) = u_0(x).$$

and suppose that we will determine the solution up to time T.

Discretized Version

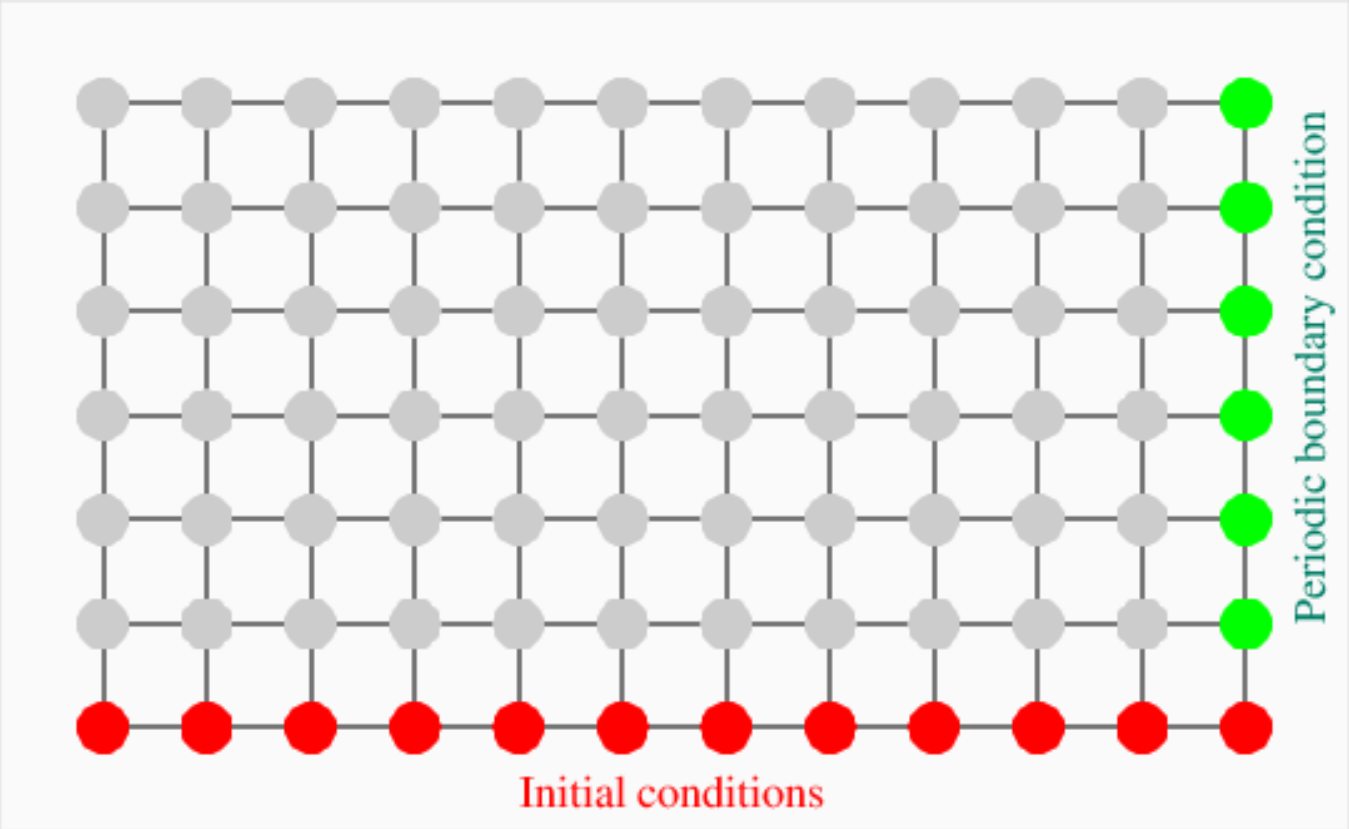
Our discretized geometry involves an $m+1$ by $n+1$ grid spaced equally in x and in t, with the solution stored as an array.

Column 0 holds our initial condition. Column $j+1$ is computed from column j; entry $(m, j+1)$ is set by periodicity.

```
|   u(0, 0)      ?       ?    ...    ?         ?
S   u(1, 0)      ?       ?    ...    ?         ?
P   u(2, 0)      ?       ?    ...    ?         ?
A   u(3, 0)      ?       ?    ...    ?         ?
C     ...      ...     ...    ...   ...       ...
E   u(m-1,0)    ?       ?    ...    ?         ?
|   u(m, 0)     ?       ?    ...    ?         ?
|
              -------TIME----->
```

Filling in an array

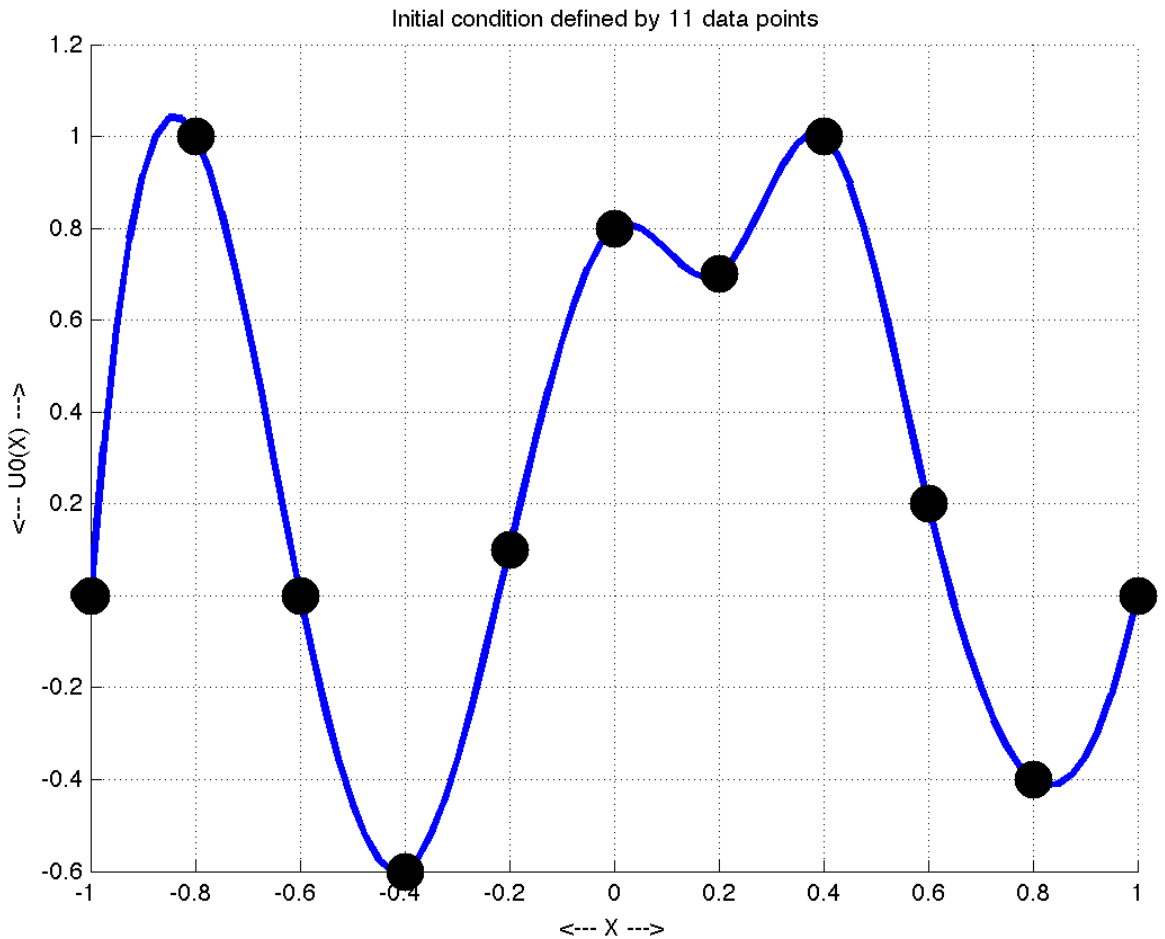
Geometrically, we imagine our problem with time as the y axis, so we are given the solution at the "bottom".



Each solution is more expensive than for our steady problem. If the initial data is uncertain, we also have more parameters to consider.

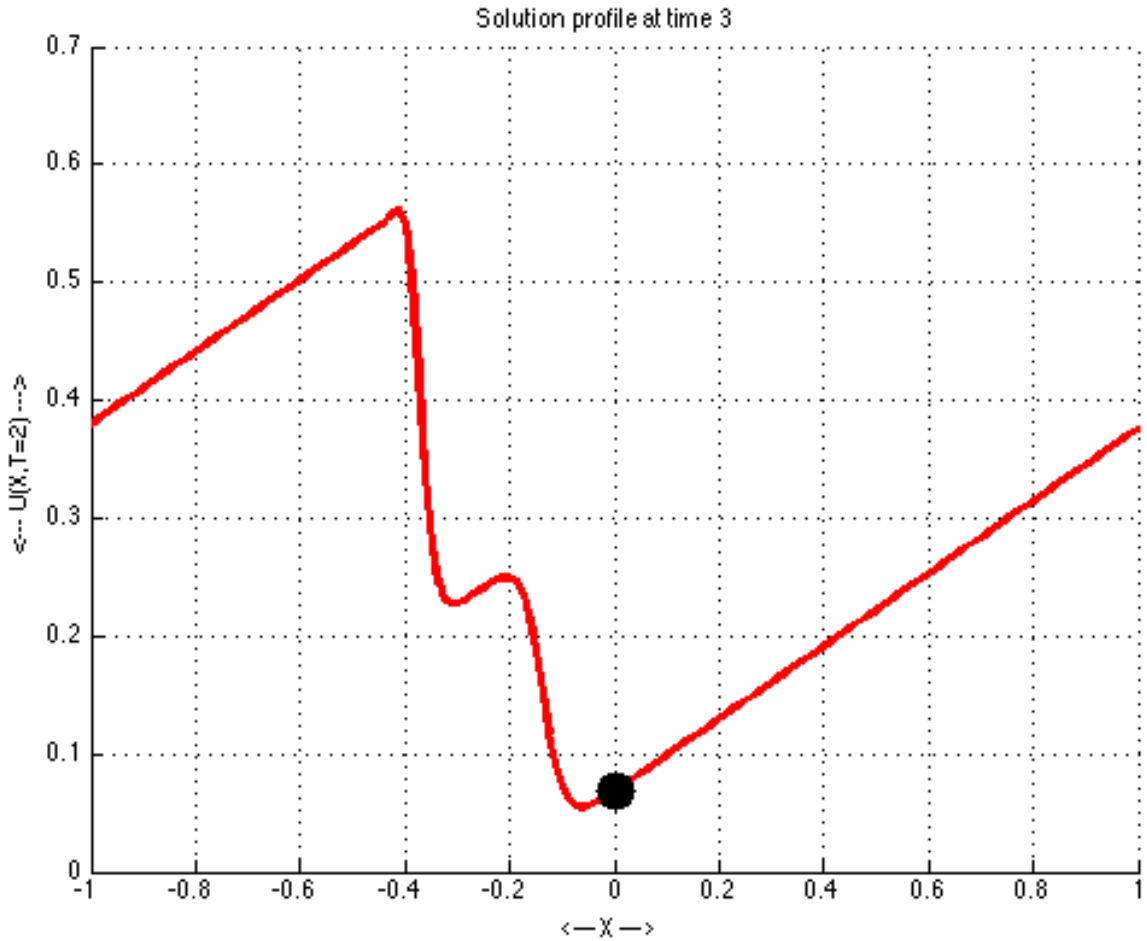
Data Defines the Initial Condition

We concentrate on the dependence of the solution on the initial condition, which we imagine is specified by some discrete set of data γ through which we pass a spline:



U(X=0,T=3) is our Quantity of Interest

The solution profile at $T = 3$ looks like this. Suppose our quantity of interest $q(\gamma)$ is simply the profile value at $x = 0$.



Quantifying the Uncertainty

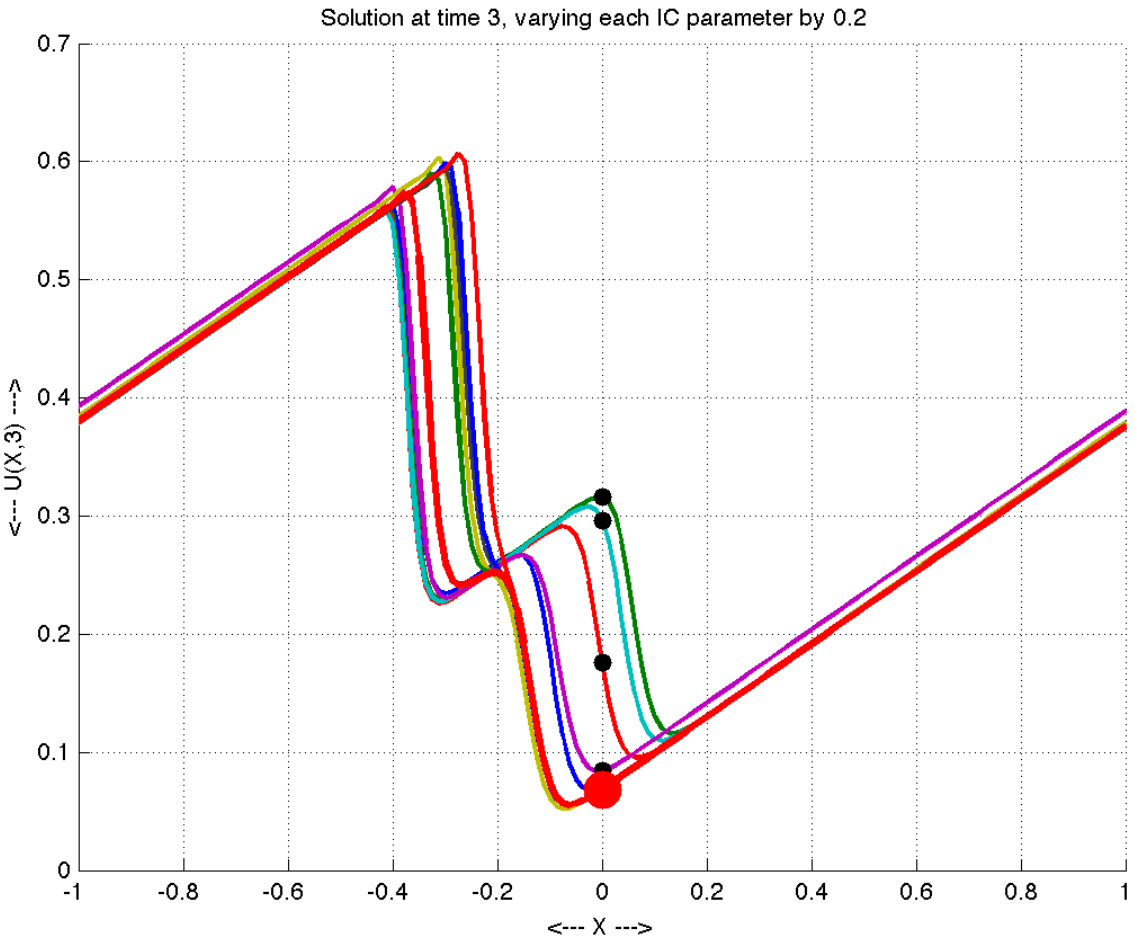
We want to estimate the uncertainty that our input data γ induces in our quantity of interest $q(\gamma)$.

To do this requires:

- asserting a model for the input data uncertainty;
- reducing the size of the input data set, if possible;
- sampling the space of input data intelligently;
- solving the state system for each input data set;
- combining the results to estimate $\mathbb{E}(q(\gamma))$ and $\sigma^2(q(\gamma))$.

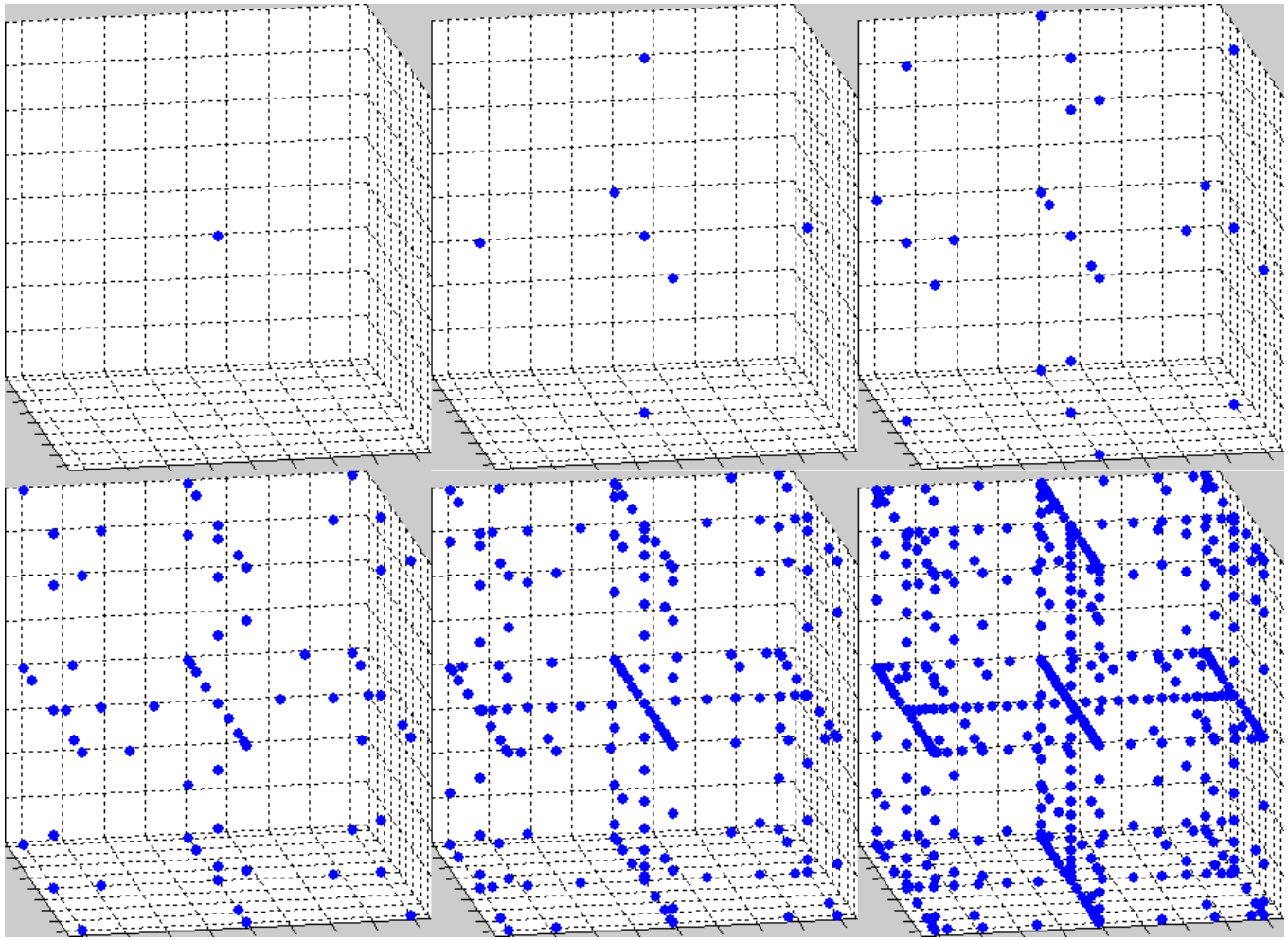
Modeling the Uncertainty

Assume uniform uncertainty in our initial condition parameters. Perturb each value by 0.2. Parameters 2 and 4 are strong, 3 moderate, and the rest have little influence.



Sampling the Input Space with a 3D Sparse Grid

Freeze all parameters but 2, 3 and 4. Let them vary uniformly ± 0.1 from their base values. This is the space we shall sample.



Sparse Grids

A sparse grid, like Monte Carlo or Quasi Monte Carlo, chooses many sets of data for input to the state system solver. It does not need to alter the internal features of the solver in any way.

Sparse grids differ from other sampling schemes because the sampling pattern produces a highly accurate polynomial model of the uncertainty influence if the state function depends smoothly on the input.

The sparse grid information can be used to estimate an integral, or to produce an interpolant function (that is, a "surrogate function") to the input/output data.

Attempt to Quantify Uncertainty

We have already computed the quantity of interest Q, the solution value U at $x = 0$ and time $t = 3$, for a "base" set of parameters.

We'll assume that parameters 2, 3, and 4 vary uniformly about their base values by ± 0.2, and we ask, assuming this uncertainty, what is the expected value of Q, written $\mathbb{E}(Q)$?

For this case, we're essentially asking for the average value of Q over the given range of possible parameter values.

We estimate $\mathbb{E}(Q)$ using Monte Carlo and Sparse Grid methods.

Monte Carlo Program Outline

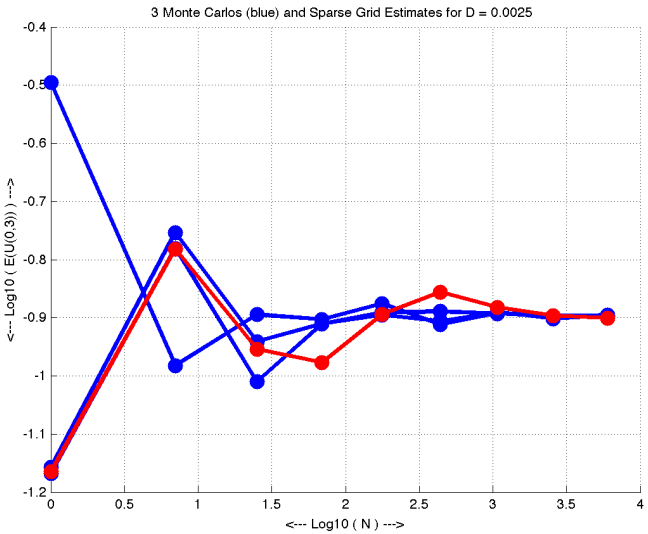
```
d = 3;                          uncertainty dimension
uk(1:9) = uk_base(1:9)          initial condition parameters
nu = ?                          viscosity

mcn = ?                         Free to choose any size
mcx = 2 * rand ( mcn, d ) - 1.0;   Sample [-1,+1]^3 uniformly
mcw = 1.0 / mcn;                The "weight" for MC

q = 0.0
for i = 1 : mcn
  uk(2:4) = uk_base(2:4) + sigma * mcx(i,1:3);
  U = burgers_solver ( uk, nu )
  q = q + mcw * U(nt,(nx+1)/2);
end
```

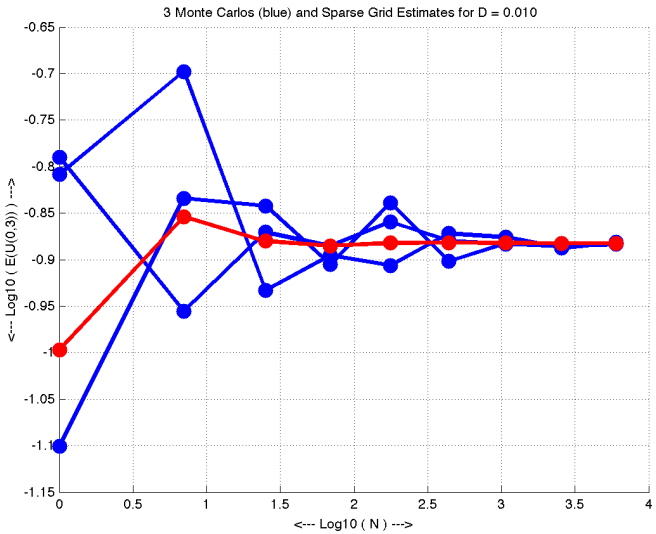
Sparse Grid Program Outline

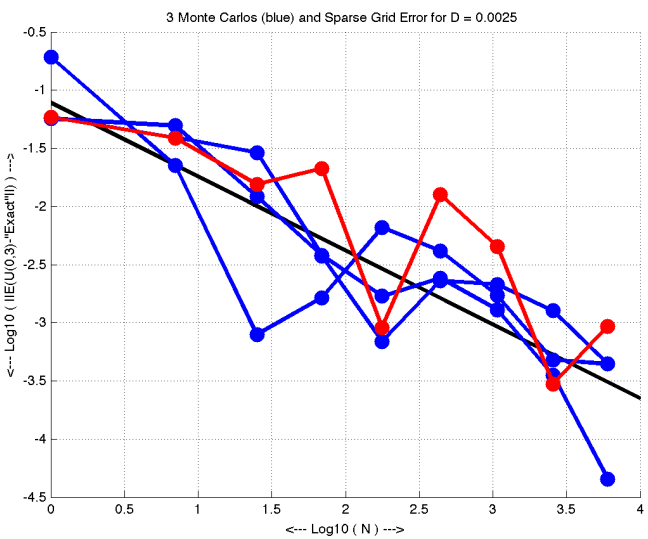
```
d = 3;                          uncertainty dimension
uk(1:9) = uk_base(1:9)          initial condition parameters
nu = ?                          viscosity

level = ?                       sparse grid level 0, 1, 2, ...
[ sgx, sgw ] = nwspgr ( 'ccu', d, level );
sgn = length ( sgw );

q = 0.0
for i = 1 : sgn
  uk(2:4) = uk_base(2:4) + sigma * sgx(i,1:3);
  U = burgers_solver ( uk, nu )
  q = q + sgw(i) * U(nt,(nx+1)/2);
end
```

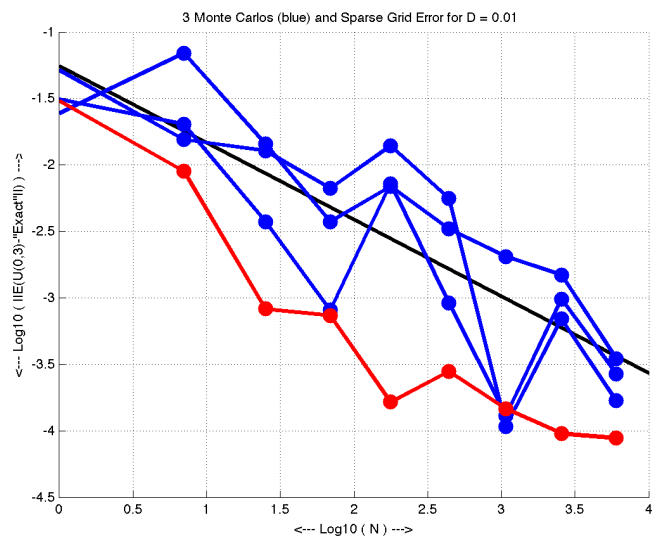
Convergence for Viscosity 0.0025 and 0.1

Q estimates in row 1, Q errors in row 2, SG (red), MC (blue). Sparse grids perform better when viscosity smooths the data.









Summary: Uncertainty Quantification

We have looked at two simple computational problems, and considered the influence of uncertainty in the input data upon an output quantity of interest.

Since the typical input data set can be large, it is important to try to identify those parameters that most strongly influence the output. A rough guess can be done by perturbations of the input; a more sophisticated approach computes a reduced order model.

A probability density function must be assigned to the input parameter space, reflecting our model of the uncertainty.

The sampling approach solves the system many times, and computes a quantity of interest Q by direct averaging (Monte Carlo) or evaluating an input/output model (sparse grid).

Summary: A Little More About Sparse Grids

A sparse grid approach may be more efficient that Monte Carlo sampling if the dependence of Q is smooth.

In this example, we found that some input parameters had very little influence. In our uncertainty model, we simply kept them fixed. But sparse grids can be designed that give most, but not all, attention to the dominant variables, while paying some attention to those with known weaker influence.

We used a simple uniform probability density, and in fact, the same one, for the three input parameters. Sparse grids can model normal variation, exponential variation, and other forms, and different densities can be used for different inputs.

Summary: High Performance Computing

In our simple case study, parallelism is available twice:

1) The sampling procedure provides, in advance, the input data sets at which the state system must be evaluated. Each of these evaluations can be carried out independently, and has only to return the final state solution, or some associated output data.

2) Because sampling procedures are non-intrusive, the state system evaluation does not need to be rewritten or adjusted in any way. Presumably, this pre-existing "deterministic" code has already been highly optimized and parallelized.

A single state solver might run efficiently on 200 nodes. An MPI approach could request 10,000 nodes, allowing us to compute 50 states simultaneously. If we have 5,000 sample inputs to process, each state solver runs through 100 sets, and the results will be collected and analyzed on a master process.

Summary: High Performance Computing

Another approach to parallelism divides the problem into independent tasks, to be executed in any order, and at any time. A master task divides up the problem, submits the tasks to a queue, collects results as they (unpredictably) are completed, and reports the final result.

Such a system is available even in MATLAB, as "task computing".

Such an approach takes advantage of a heavily scheduled computing cluster; instead of waiting for enough processors to load the entire set of tasks, tasks opportunistically seize processors as they become available.

Sampling approaches such as Monte Carlo and Sparse Grid can readily be implemented with such an approach.

Outline

Recipe for a UQ collocation on HPC:

- lay out the mathematical model;
- specify input parameters;
- for given inputs, develop solver (already available?);
- compute the basic solution u;
- identify influential parameters;
- reduce model (simply cut some parameters?);
- model input uncertainty;
- choose sampling approach (MC? Sparse grid?);
- implement on HPC;
- extract quantity of interest $q(u)$.

Remarks from the Audience:

Questions were posed at the end of this talk, and I here try to recall the heart of the questions, and a sort of response. I hope I have not misrepresented anyone's point, and of course, I have had time to rephrase my own responses to be a little more coherent than they may have been at the time!

Remarks from the Audience #1:

Your initial step of looking for input variables with a strong influence perturbs the reference values by a small amount. In the problems I work on, the parameters can "turn on and off", depending on the problem regime. So with your technique, I might think a variable is unimportant when it is only "sleeping".

I did not mean to suggest that my method of identifying the influential variables was universally applicable, or even reliable for this example problem. What I did want to indicate was that, given a problem with several input parameters, some may be much more important than others, and that it is sensible to try to determine this, so that you can study the important variables more carefully.

My approach used a simple perturbation. We could have computed sensitivities; we could have had information available from previous runs that suggested which variables to focus on.

Remarks from the Audience #2:

Your perturbation approach to identifying the influential inputs would miss higher order influences.

Yes, that's quite true. Assume that the base solution has two inputs, x and y, both set to zero, and that the quantity of interest is $q = x * y$. Perturbing x or y separately as I did would not reveal any influence on q.

In my presentation, I simply expelled from further consideration the input variables that I judged unimportant. A more sensible approach would simply assign them a lower weight in the analysis, with the option of increasing that weight if, as the calculation proceeded, we discovered that a variable had more influence than we had anticipated. Weighting the variables (rather than omitting some), and adaptively adjusting those weights, are approaches available in modifications of the sparse grid method, among others

Remarks from the Audience #3:

In your time-dependent example, you were very fortunate in your choice of quantity of interest, since ordinarily it would depend on the viscosity to a great extent.

I must apologize that I got so wrapped up in trying to analyze the effect of the multiple parameters in the initial condition that I unintentionally omitted the viscosity parameter from that analysis; indeed, the viscosity affects the quantity of interest, and should have been included. Since it was not, we essentially are asserting that there is no uncertainty in the viscosity value that we used in the computations.

Remarks from the Audience #4:

I am concerned that you start with a "poor-man's" version of the Navier-Stokes equation, then carry out a crude discretization of what is essentially the wrong equation.

I did not mean to pretend to be solving the Navier-Stokes equations. For whatever reason, let us suppose that my interest was the Burgers equation, pure and simple. If you will give me that, then we can proceed. My point was not to talk about how to solve the Navier-Stokes equations, but about how a person, who can already approximately solve a discretized system of equations, could try to frame an uncertainty quantification focused on the propagation of error from input parameters to an output quantity of interest.

Remarks from the Audience #5:

You essentially discard input parameters even though you know their effect is not actually zero. I am very strongly against ignoring any data.

I did omit input parameters, and my results paid a cost for ignoring the effects of those parameters on the quantity of interest. However, my point in that procedure was to claim that for a really large scale calculation, where there may be 50 to 100 parameters, (and I have talked to people who claim to consider thousands and tens of thousands of inputs), the resulting multidimensional probability integral becomes impossible to deal with.

One way to make the problem tractable is to look for some kind of reduced model with fewer variables; another is simply to try to identify and follow a small selection of important variables; another is to weight the variables and focus on the most important.

References:

Per Petterson, Gianluca Iaccarino, Jan Nordström, *Numerical analysis of the Burgers equation in the presence of uncertainty*, JCP, Volume 228, Number 22, p8394-8412, 2009.

Toby Driscoll, *Stochastic collocation for the Burgers equation*, http://www2.maths.ox.ac.uk/chebfun/examples/stats/html/StochasticCollocationBurgers.shtml.

Dongbin Xu, Numerical Methods for Stochastic Computations, Princeton University Press, 2010.