Solution of differential equations with Genetic Programming and the Stochastic Bernstein Interpolation

Daniel Howard
Biocomputing and Developmental Systems Group
University of Limerick, Ireland
DanielHoward@sunrisemalvern.freeserve.co.uk
Joseph Kolibal
Department of Mathematics
University of Southern Mississippi, USA
Joseph.Kolibal@usm.edu

BDS-TR-2005-001

June 19, 2005

Biocomputing-Developmental Systems Group
University of Limerick
Ireland

Abstract

This report introduces a method for the solution of the Convection-Diffusion equations (CDE) that combines Genetic Programming with Stochastic Bernstein Interpolation. Significantly, it is being used to solve a problem that has resisted analysis for a long time using other methods. Although the method in this report solves the one-dimensional CDE which has also been solved analytically and optimally, our strategy of combining the Stochastic Bernstein Interpolation method with GP allows for the method to extend to higher dimensions, and thus it shows how to construct GP based methods for solving a range of computational problems in multiple dimensions which have hitherto resisted numerical solution.

1 Introduction

The numerical solution of non-self-adjoint differential equations, e.g. the convection-diffusion equation (CDE), remains one of the greatest unsolved challenges in the field of Numerical Analysis (Süli et al, 1980) (Morton, 1996).

An interesting method for solution of differential equations with Genetic Programming was introduced and demonstrated in (section 10.7 in Koza, 1992). This method evolved the solution to differential equations. Satisfaction of the differential equation was measured at a number of test points, and satisfaction of the initial condition was also tested. Both of these became weighted components of the fitness measure.

An alternative Genetic Programming based method for solution of the two boundary value CDE was introduced in (Howard and Roberts, 2001). This used Genetic Programming to evolve a variable length vector of real numbers. These served as the constants of a monomial of increasing order. When the monomial was pre-multiplied by the factor $x(1-x)$ the CDE boundary conditions were satisfied a-priori. Since derivatives could now be obtained analytically from the evolved constants, the integral of the square of the differential equation could also be obtained analytically over the domain (0,1). Thus, eliminating the need for test points in the domain.

The method of (Howard and Roberts, 2001) however is compromised in more than one dimension because it is generally not possible to prescribe arbitrary Dirichlet boundary conditions. It is not always possible to modify the evolved polynomials to satisfy these boundary conditions.

The method that is introduced in this report borrows aspects of both of the aforementioned methods. It employs test points to sample the satisfaction of the CDE in the domain (Koza, 1992), and it also evolves a set of constants. The method uniquely combines Genetic Programming with Stochastic Bernstein Interpolation (SBI). SBI was recently developed by Joseph Kolibal and is a patent pending method of data interpolation.

Section 2 introduces the Stochastic Bernstein Interpolation. Section 3 gives the strategy for its evolutionary computation. Section 4 presents the combined GP-SBI interpolation method for numerically approximating the solution of CDE. Section 5 illustrates results and discovered properties of this method. Discussion and Conclusions follow with information about ideas for further work.

Appendix A describes in more detail the fundamental mathematical limitation of weighted residuals methods for solution of convection diffusion problems. This is followed by other appendices that provide more information on SBI.

2 Stochastic Bernstein Interpolation

The development of a unified approach to data regularization began with an observation on the area preserving properties of the Bernstein polynomials and a desire to construct more computable approximations which share this property. These approximations generalized to a broader set of methods built around the product of row and column sum one matrix products, which yield the Bernstein-style interpolation methods, connecting area preservation, stochasticity, approximation and interpolation. As most of the computational procedures that are described involve nothing more than the accumulation of sums in which order

does not matter, or inverting and multiplying matrices, the approach should be highly parallelizable using standard tools already developed for parallelizing matrix algebra. Furthermore, through the construction of Bernstein bases, the methods can be applied efficiently to large data sets in $O(n)$ operations.

The Bernstein functions which are introduced share many of the desirable approximation properties of the well known Bernstein polynomials. These functions provide a natural extension of the Bernstein polynomials to a continuum model, however they incorporate desirable features which make them usable where the application of Bernstein polynomials is untenable. These functions allow for flexible control over smoothing; improve computational efficiency for large n; and, permit the use of non-uniformly spaced data. The Bernstein functions can be used for constructing high frequency filters, error averaging, and function recovery. As such, they provide a convenient alternative to classical methods, e.g., least squares methods. The extension is accomplished by allowing the binomial probability distribution to become a Gaussian probability distribution, along with the appropriate change of variables. The new families of functions are characterized by a free parameter, equivalent to the thermal diffusivity in heat conduction problems, and allow for greater control of the function than is possible with polynomials. These functions can be applied to solve problems in:

1. *Data Approximation and Interpolation* – The ability to accurately interpolate or construct appropriate functional approximations between data points is crucial to many engineering and science applications with incomplete data sets. In the case of data approximation, the method provides for accurate approximation with very controllable smoothing and provides a uniform approximation to the data, with absolutely no spurious extrema being introduced (wiggles). In the case of interpolation, the original data points are included in all interpolated curves, and false extrema and spurious wiggles can be minimized (near monotonicity).
2. *Data Recovery* – Recovering the underlying characteristics of noisy data sets is imperative to understanding the behavior of many processes. The method is able to accomplish this task well even for very noisy data for which the errors, i.e. the noise, approaches zero net deviation from the data on arbitrary subsets of the domain (that is, the noise statistically has a central tendency; For cases where the noise is markedly skewed away from this central limit, the results will reflect the skewing of the noise. In these cases the technique must be applied with appropriate statistical models to remove the bias).
3. *Data Error Filtration* – Removing noise from data is important in regard to improving the quality of the data. The method can be used to filter different frequency components of noisy data, thereby improving the quality of the data. This approach is valid so long as the noise is of much higher frequency than the signal, and the range of validity of the approach can be extended by combining this with spectral shift methods.
4. *Data/Image Deconvolution* – Deconvolving data enables the recovery of features not readily seen or understood within the existing image or data set. This feature has a broad range of applications in numerous medical, industrial, scientific, and military applications. Similar to data recovery,

the proposed methodology can be employed to recover image details, effectively de-blurring an image.

Moreover, since these functions allow for robust representation of functions which are sampled at discrete points, they also allow for the robust determination of the derivatives of a function, which are computable as linear combinations of Gaussian terms.

The construction of a Bernstein interpolant is developed from the Bernstein function approximation through a deconvolution-reconvolution process. In turn, this can be recognized as being part of a broader approach to constructing interpolants relative to any probability density function, or mollifier which can be normalized to a probability density function. For interpolating particularly rough or noisy data, the Bernstein interpolation technique using Bernstein interpolation provides a robust mechanism for producing smooth interpolants which are relatively free of spurious artifacts. The construction gives rise to interpolants which are infinitely differentiable over the entire domain, not just twice differentiable as cubic splines, and they can be constructed to be ripple-free, unlike trigonometric interpolants.

Interpolation requires solving an associated deconvolution problem. Formally, this step involves solving a linear system of equations which constructs the set of pseudo-data points which, when approximated by a Bernstein function, maps to the original data. Thus, this two step construction interpolates the data; points between the original data are filled in with a smooth curve which joins the original data points to each other. Because it is possible to control the deconvolution and smoothing process locally, it is possible to construct nearly oscillation free interpolants to data which exhibit large changes in magnitude (e.g., step discontinuities).

In practice, it is more practicable to construct the interpolant using a Bernstein basis, consisting of Bernstein functions which interpolate known discrete data points. Using these and the fact that these functions form a basis for the vector space of interpolating Bernstein functions, allows the construction of the interpolant directly without the need to solve any large linear system in order to obtain the interpolant. The Bernstein function basis elements must be obtained by inverting the appropriate linear system, however these depend only on the mesh spacing (abscissa values) and not the data values (ordinates). Thus, for example for fixed grids, the basis needs only be constructed once, while for variable sized grids with variable mesh, the basis would have to be constructed for each grid used. The new interpolated or approximated data may consist of uniformly distributed points, or can be non-uniformly distributed in any given manner (the data need not be ordered).

2.1 The behavior of SBI and the free parameter σ

A parameter of the SBI controls the amount of turning or wiggle of the function between interpolation points. This is due to the intrinsic smoothing parameter that is contained in the definition of the Bernstein functions. the value of the free parameter σ at each x in the domain determines the rate at which solution of the heat equation is evolved. Large diffusion values give large change. Small diffusion values give small change.

While this result applies to stochastic interpolation using Bernstein functions as well as to stochastic approximation using Bernstein functions, the results are most conveniently illustrated using Bernstein approximation. In regard to Bernstein approximations, the choice of diffusion coefficient σ alters the convergence of the sequence of the Bernstein functions. Interesting choices for σ can be constructed using:

- Nodal point vanishing diffusion, $\sigma = \delta^2 \prod_{i \in I}(x_i - x)^2$, where $I \in \{0, 1, \ldots, n\}$. The Bernstein approximation will adhere to the points x_i, since no diffusion can occur at these points.

  The case of $I = \{0, n\}$ is the standard one, i.e., in constructing the Bernstein function from the Bernstein polynomial, a term of the form $sigma = 1/sqrtx(1-x)/2$ arises as an argument to the Gaussian function, and can be interpreted as a diffusion parameter. Since the diffusion parameter σ is greatest at $x = 1/2$, is monotonically increasing on $(0, 1/2)$, and is monotonically decreasing on $(1/2, 1)$, it is obvious that the greatest diffusion and hence greatest smoothing occurs at $x = 1/2$ with no smoothing at the endpoints, i.e., the function is interpolating there, as is the Bernstein polynomial.

- Alternatively, can set $I = \emptyset$ in which case we have constant damping throughout the interval and $\sigma = \delta^2$. In this case the smoothing is equal throughout the domain and at no points is the Bernstein function approximation interpolating.

The consequences of using different values of the diffusion coefficient on a uniformly spaced grid are illustrated for the function $f(x) = \cos(12\pi x)$ in Figs.1 and 2. Using a diffusion coefficient which is constant, as in Fig.2, avoids having the rate of convergence depend on x, and so is used in the construction of stochastic interpolants using Bernstein functions. Varying the value of σ from about 0.01 to 1.5 allows the construction of interpolants in which the amount of wiggle and ripple between interpolation points can be controlled. The smaller sigma is, the more prone the function is to develop a ripple, while the larger sigma is, the more prone the function is to develop larger overshoots, i.e., wiggles.

2.2 Pseudo-code for the method

Pseudo-Code (linear model):

1. Read data $\{(x_i, y_i)\}, i = 0, n-1$;
2. Convert data coordinates to lie on unit interval;
3. Construct convolution matrix A_{nn} (depends on x_i and smoothing parameter) using the generator of the row space (typically a Bernstein function).
4. Construct de-convolution matrix, A_{nn}^{-1};
5. Construct augmented matrix $\tilde{A_{mn}}$ using the same generator of the row space $m > n$;
6. Evaluate $\tilde{A}_{mn} A_{nn}^{-1} \mathbf{y}$ to obtain output data $\{y_i\}, i = 0, m-1$;

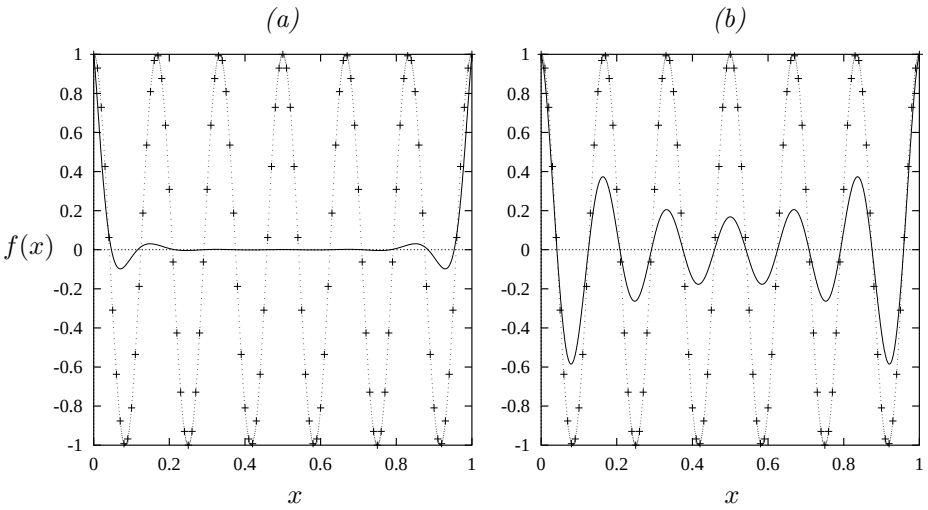

Figure 1: Varying the diffusion coefficient: (a) $\delta = 2.0$, (b) $\delta = 1.0$ (standard diffusion coefficient) base on sampling the function $f(x) = \cos(12\pi x)$ at 100 points with $\sigma = \delta^2 x(1-x)$.

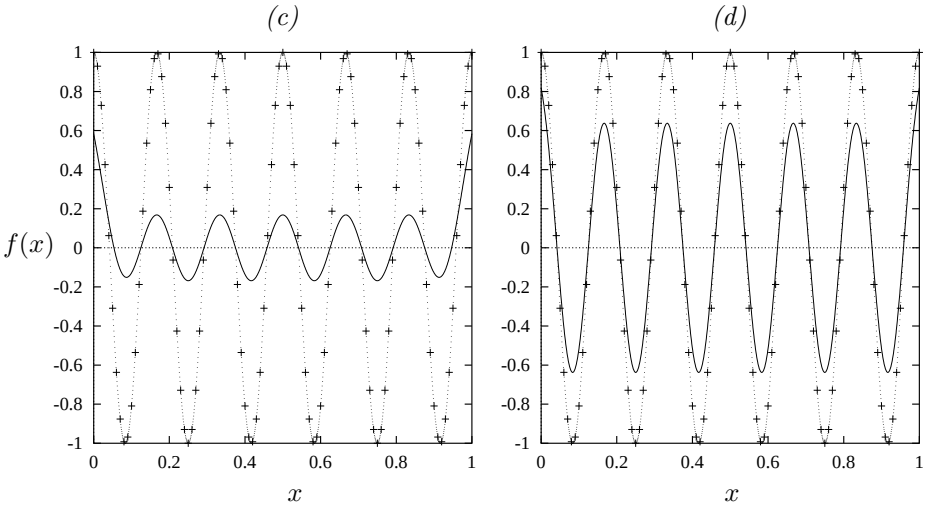

Figure 2: Varying the diffusion coefficient: (c) $\delta = 0.5$, and (d) $\delta = 0.25$, based on sampling $f(x) = \cos(12\pi x)$ at 100 points with $\sigma = \delta^2$.

7. Convert the output to the world coordinate system.

Note that in constructing A and $\tilde{A}$ the choice of smoothing parameters affects the results. This can be a single value for all x_i, or it can be a function.

2.3 Signal pre-conditioning

Signal pre-conditioning can be achieved using stochastic interpolation (deconvolution-convolution operators). Stochastic interpolation provides a mechanism for smoothing or filtering signals through a discrete convolution process, as well as deconvolving it to yield a pre-image which can be used to emphasize and enhance anomalies, detail, and signal fluctuations.

Stochastic interpolation using Bernstein functions uses a discrete convolution operator which is (nearly) area preserving and which yields infinitely differentiable interpolants which can be over and under relaxed to yield smooth families of curves which represent the underlying data. In particular, by over-relaxing the interpolation the technique can remove clutter from extremely noisy signals to achieve smooth function recovery.

If the signal is under-relaxed, the process instead produces smooth functions which are characterized by being highly oscillatory in the neighborhood of rapid deviations of the data; conversely, if the data is derived from smooth functions which contain no high frequency deviations, then under-relaxing the data yields curves which are nearly identical to the original curve, i.e., smooth data are eigenfunctions with eigenvalue 1 of the deconvolution operator. This may provide a very sensitive mechanism for characterizing regions of rapid data fluctuation. Thus, these methods can either directly provide effective noise removal, or can be used with other methods as a data pre-conditioner, particularly when attempting to solve difficult inverse problems.

2.4 Function interpolation

An advantage of function recovery using stochastic Bernstein methods is that it unifies traditional tasks such as function approximation, function smoothing, and function interpolation. Stochastic Bernstein interpolation results when the convolution and deconvolution steps are equal in measure. In all cases the result is a smooth (i.e., infinitely differentiable function) which fits the sampled data. The process yields excellent interpolants, even for very rough data.

Consider the function (line shown in green) in Fig. 3

$$f(z) = \cos(3z)\exp(\mathrm{abs}(\cos(\exp((z+5)/3))))$$

sampled at 1000 points, and compare it with a deconvolution of this data. Notice that the red line or graph of the pre-image of the function is the same as the data where the function is smooth, however it oscillates extensively at the three points where f has a discontinuous derivative. Thus deconvolution is a sensitive measure of function smoothness, and can be used to adaptively tune performance.

The function is next interpolated on 100 points (red x's) that are uniformly sampled along the interval, shown plotted in figure 4. The interpolant is now shown in green and the function is now plotted in blue. There is a very slight ripple around the second kink of f, however this can be tightened up as the method was applied in a straightforward manner without doing anything special to improve the results.

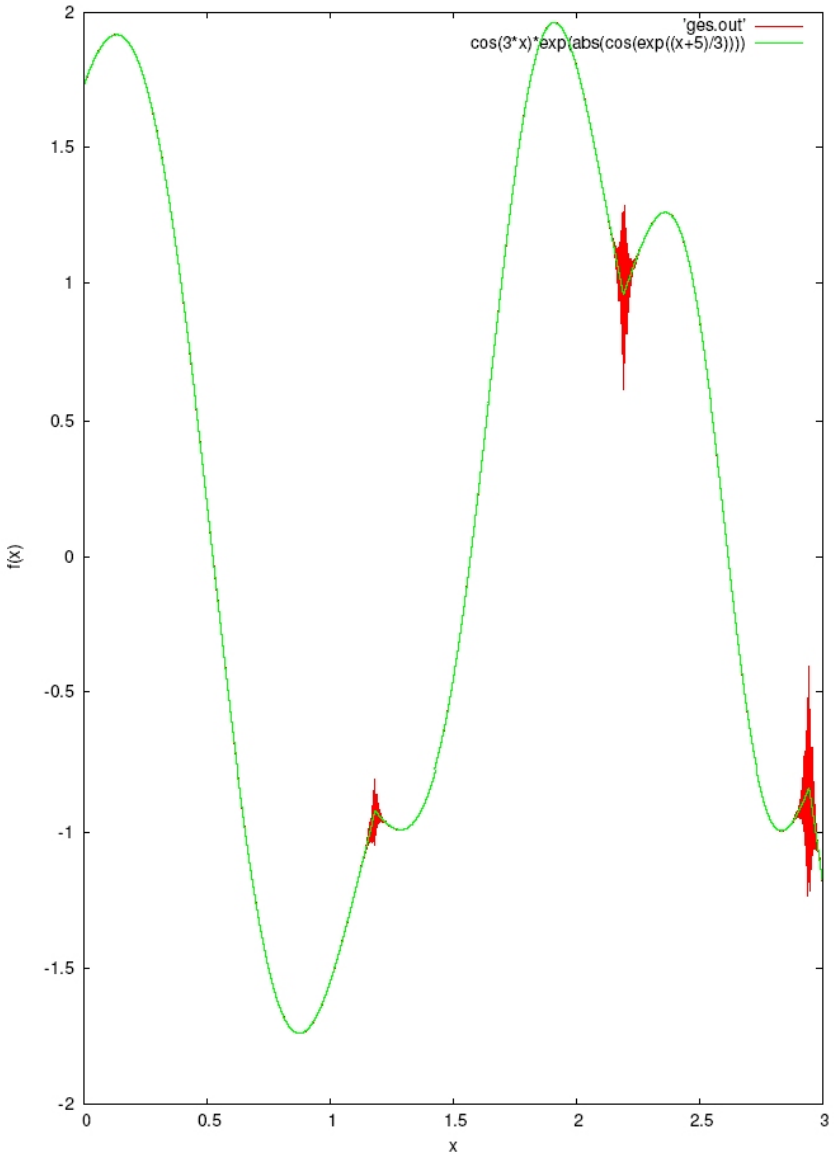

Figure 3: Function and its deconvolution. Green line is the function and the red line is the pre-image of the function (data ges.out)

3 Genetic Programming evolution of SBI

The idea is to evolve the data for the method. It involves design of a Genetic Programming method for arriving at the data vector. One advantage of the SBI is that it can work with an unstructured grid, i.e. both the points of the input values $\{(x_i, y_i)\}, i = 0, n-1$ and those of the output data $\{y_i\}, i = 0, m-1$ values need not be equally distributed.
The aforementioned pseudo-code now becomes:

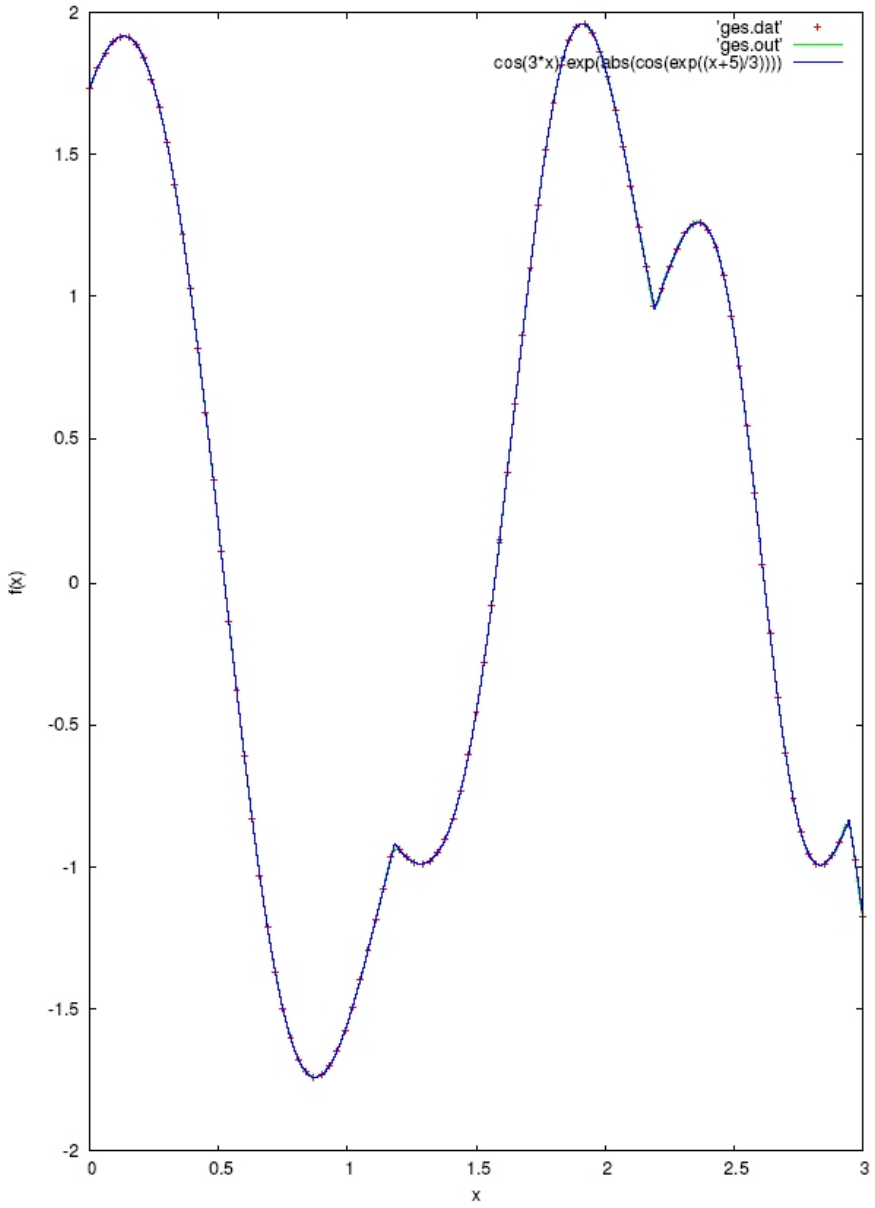

Figure 4: Interpolation of f using 100 interpolation points (red crosses data ges.dat). Function in blue, interpolant in green.

1. Genetic Programming evolution of $\{(x_i, y_i)\}, i = 0, n-1$;
2. Convert data coordinates to lie on unit interval;
3. Construct convolution matrix A_{nn} (depends on x_i and smoothing parameter) using the generator of the row space (typically a Bernstein function).
4. Construct de-convolution matrix, A_{nn}^{-1};

5. Construct augmented matrix $\tilde{A_{mn}}$ using the same generator of the row space $m > n$;
6. Evaluate $\tilde{A}_{mn}A_{nn}^{-1}\mathbf{y}$ to obtain output data $\{y_i\}, i = 0, m-1$;
7. Convert the output to the world coordinate system (solution to the CDE).
8. Derivatives: $\{dy_i/dx\}, i = 0, m-1$ and $\{d^2y_i/dx^2\}, i = 0, m-1$ are constructed by gradient versions of $\tilde{A}_{mn}$.
9. the GP fitness measure (satisfaction of the CDE) can now be obtained from the L2 norm of the pointwise values of the CDE squared: $\{(d^2y_i/dx^2 - Pdy_i/dx)^2\}$where P is the Peclet number of the CDE. The GP fitness measure is taken to be the negative of this L2 norm. GP tries to minimize this error by maximizing fitness.

4 Genetic Programming Solution of CDE

Without loss of generality a Genetic Programming implementation was constructed to tackle the one dimensional homogeneous version of the convection diffusion equation without source terms:

$$\frac{d^2T}{dx^2} - P\frac{dT}{dx} = 0$$
$$T = 1.0 \text{ at } x = 0.0$$
$$T = 0.0 \text{ at } x = 1.0$$

where T is the temperature and P is a constant known as the *Peclet Number* which measures the ratio of convection to diffusion. Recall that the SBI works in the domain (0,1) but that the solution can be mapped to a different functional domain. In two dimensions, for example, a 2D SBI could be constructed to attempt the same problem with boundary conditions on edges.

4.1 GP functions and terminals

The objective of the Genetic Programming implementation is to evolve the input vector to the SBI procedure. The GP formulation functions and terminals were chosen to be identical to the one that was presented in (Howard and Roberts, 2001) and summarized in table 1. Evaluation of a GP individual produces a variable size vector of real numbers. Arithmetic functions generate the real numbers by operating on the ephemeral random constant C_S. This is stored as one byte and can represent up to 256 values. Usually these are equally spaced in the range [0,1].

A set of record manipulation functions (of parity 2) interacts with the resultant vector. These manipulate two global pointers to the element position in the vector of resultant coefficients: pointer L for "last index" or tail position and pointer C for "current position". Both of these pointers are initially set to the first position. Function ADD writes its first argument to the vector element that is pointed to by L. ADD then increments L provided that $L < L_{MAX}$. ADD then sets $C = L$. Function $BACK$ decrements pointer C as long as $C > 0$. Function $WRITE$ overwrites the vector element at C with the value

Table 1: GP parameters

parameter	setting
functions	+, -, *, / $(x/0 = 1)$,
	ADD, $BACK$, $WRITE$,
	$Wm1$, $Wm2$, $Rm1$, $Rm2$
terminals	C_S
globals	$L_{MAX} = 50$
	variable length solution vector
	pointers L and C
	memories $m1$ and $m2$
version	steady state GP
kill tournament	size 2
breed tournament	size 4
regeneration	80% crossover (single child),
	20% injection
	10% mutation
max tree size	1000 nodes

that is held by one of its arguments. If $C < L_{MAX}$ it increments C but if $C > L$ it increments L. These functions return their other argument to the GP tree.

Moreover, the function set is enhanced with two memories $m1$ and $m2$ manipulated by functions again of two arguments: $Wm1$ returns its first argument to the GP tree and writes the content of its second argument to overwrite $m1$ and similarly for $Wm1$ and $m2$. Functions $Rm1$ and $Rm2$ simply return the contents of $m1$ and $m2$ respectively to the GP tree (ignoring the contents of their arguments).

The GP also implements a user defined parameter value that prevents solutions which produce a variable length vector smaller than a certain size (it severely punishes candidate solutions that violate this). Although this parameter helped to add variety to the runs, GP was able to overcome it by "nearly coalescing points" and thus it was not significant.

4.2 Non-uniform data vector and domain adjustment

One implementation is to set the y_i values in $\{(x_i, y_i)\}, i = 0, n-1$ to the variable length vector that is discovered by GP, and to spread the x_i values uniformly over the domain (0,1) while adding the two boundary conditions as additional points

The objective of Genetic Programming was to discover $\{(x_i, y_i)\}, i = 0, n-1$ and this involves discovery of both the x_i position and the y_i value of each data points $(0.0, 1.0)$ and $(1.0, 0.0)$ respectively.

However, it became apparent that Genetic Programming would value the freedom to position the points. Furthermore, upon numerical experimentation, it also became apparent that the Stochastic Bernstein Interpolation process would benefit from buffer zones *outside* of the solution region.

For this reasons the following modifications were implemented. The domain of solution was mapped not to $(0, 1)$ but to $(0.2, 0.8)$ and GP was allowed to

choose the position of the points anywhere (with restrictions as explained below) in the interval $(0, 1)$ and the algorithm to obtain this information from the variable length vector was implemented as follows:

1. After the new individual is evaluated check whether the size of the resultant vector of real numbers is odd. If it is even discard its last term;
2. the $x_0 = 0.0$ is fixed and y_0 is set to the first element of the resultant vector;
3. the $x_n = 1.0$ is fixed and y_n is set to the second element of the resultant vector;
4. then in turn take tuples of the vector v_i v_{i+1} corresponding to x_i and y_i of points in (0,1);
5. in doing this, however, x_i for these intermediate points is scaled, converted to a percentage and then matched to a number of discrete bins which span the(0,1) domain. These were implemented as 97 bins so that no $x_i - x_{i+1}$ gap could be less than a 0.01 gap.
6. the position of these bins take into account the presence of the boundary conditions which are prescribed exactly (not evolved) as $(0.2, 1.0)$ and $(0.8, 0.0)$ respectively.

The differential equation required a simple scaling of 0.6 to enable the solution to be mapped out from the (0.2,0.8) domain to the (0,1) domain:

$$\begin{aligned} 0.6\frac{d^2T}{dx^2} - P\frac{dT}{dx} &= 0 \\ T = 1.0 \text{ at } x &= 0.2 \\ T = 0.0 \text{ at } x &= 0.8 \end{aligned}$$

where T is the temperature and P is a constant known as the *Peclet Number* which measures the ratio of convection to diffusion.

4.3 Fitness measure

As described in the aforementioned SBI pseudo-code the fitness was calculated over m output data points $\{y_i\}, i = 0, m-1$. The location of these points can be user defined. A set of 241 points was probabilistically varied using a Latin sampling so that these would be proportionately distributed in the domain (0.2, 0.8).

There are a number of choices for the fitness measure and two were experimented with:

1. compute $0.6\frac{d^2y}{dx^2} - P\frac{dy}{dx}$ and square it at the location of each SBI output point. Average the resultant and take its square root. This is a measure of error and the negative of this measure was used as the measure of fitness.
2. when generating the random locations divide (0.2,0.8) into 60 regions. Each region has 4 uniform gaps and involves 5 output points and the square of $0.6\frac{d^2y}{dx^2} - P\frac{dy}{dx}$ can be integrated using the five point Boole's rule, for example, which has a good degree of accuracy.

3. although it is possible to obtain the analytical integral of the square of $0.6\frac{d^2y}{dx^2} - P\frac{dy}{dx}$, it was not attempted as it is an arduous task. Regardless, computing this analytical integral would still involve the value of the derivatives recovered at the location of the output points.

4.4 Derivatives

The derivatives of y are readily available from the Stochastic Bernstein Approximation by construction of derivatives of the augmented matrix $\tilde{A_{mn}}$. An alternative method would compute the derivatives by central, forward or backward differences in the output vector values y_i.

It was discovered that the SBI computed derivatives were smoother than those recovered with finite differences. There are good reasons for this. The SBI computed derivatives were used in nearly all experiments. The Genetic Programming approximation was also successful with finite difference recovered derivatives.

5 Experimental Results

As we are developing this new method under different fronts, this section is an exposé of results and interesting observations obtained under experimentation with different implementation variants. Our objectives are to use the model problem to gain further understanding of the behavior of the method with a view to its deployment in multi-dimensional systems.

5.1 Lower value of Peclet number runs

The value of σ in the SBI method was set at $\sigma = 1.0$ throughout all of the low Peclet number runs.

Figure 5 is a typical result of the output y_i values for a very low Peclet number (Pe = 1) numerical approximation with the GP-SBI method. This was obtained when the method used a random sample of output points and the integration region was over (0,1) - prior to the changes discussed in the previous section where the domain of the problem is embedded in (0.2,0.8).

Figure 6 illustrates typical results at a slightly higher Peclet number. These figures now also show the value of the absolute value of $\frac{d^2y}{dx^2} - P\frac{dy}{dx}$ at each of the output points.

In figure 7 and figure 8 at $P = 4$ the domain was not formally mapped to (0.2,0.8) but the boundary conditions brought inside the domain. However, for illustration the error was evaluated over the (0.2,0.8) range only. Distribution of values of the absolute value of $\frac{d^2y}{dx^2} - P\frac{dy}{dx}$ at each output point is shown in the inset graph. Note in this example that the boundary conditions were not chosen as $T = 1.0$ and as $T = 0.0$ at 0.2 and 0.8 respectively but other boundary condition values were selected. This aims to illustrate that any boundary conditions can be implemented with the method.

The SBI calculated derivatives at the output value locations are plotted in figure 8 together with the analytical solution for the derivatives. It is nearly always the case that the GP formulation approximates the first derivative closer to the analytical solution than it approximates the second derivative. Usually,

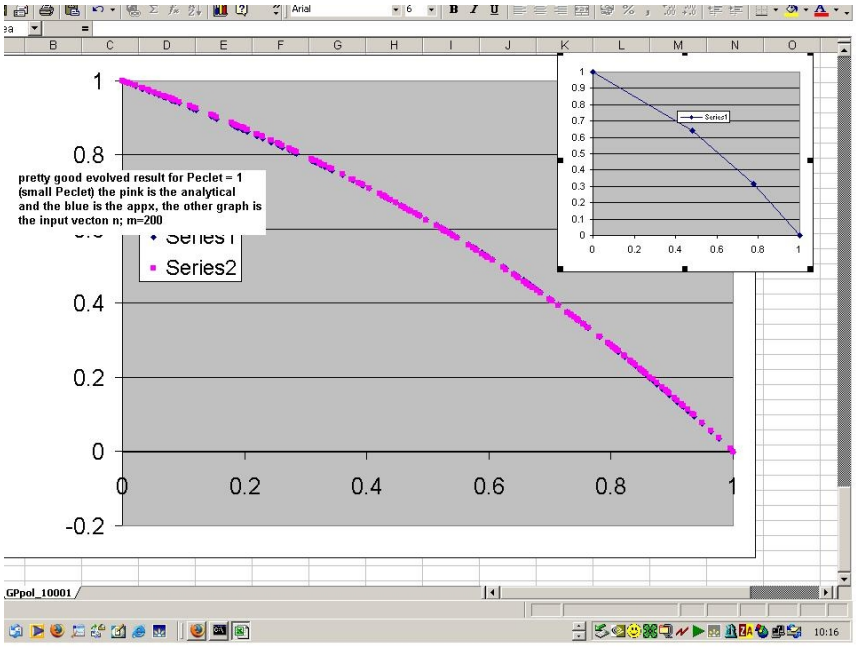

Figure 5: A result at a very low Peclet number. The magenta curve is the exact analytical solution and the blue line is the SBI approximation. The picture in the inset shows the position and value of the interpolation points: two internal points that were evolved by GP together with the boundary conditions. The output data was obtained at 200 random locations.

the error is more pronounced in the end regions. Figure 9 more clearly illustrates this phenomenon for a candidate solution at an earlier GP generation. This was one motivation for embedding the domain of interest in the (0.2,0.8) region. The other reason is described in the following section.

Figure 10 is from a run that accomplishes a similar level of error to that of figure 7 but now the GP solution evolved one fewer interpolation point.

5.2 FD derivatives versus SBI derivatives

It was interesting to note that if the derivatives are computed using finite differences they can experience some oscillation which translates as a source of difference to the fitness function. The SBI derivatives involve every point in the approximation and consequently are smoother than the very localized FD computed derivatives. Figure 11 illustrates how "clouds of dust" of points start to appear in the diagram. In practice this does not affect the GP evolution process which obtains the solution at these low Peclet numbers. The derivatives tend to get smoother at higher generations and reduce with reducing error of approximation.

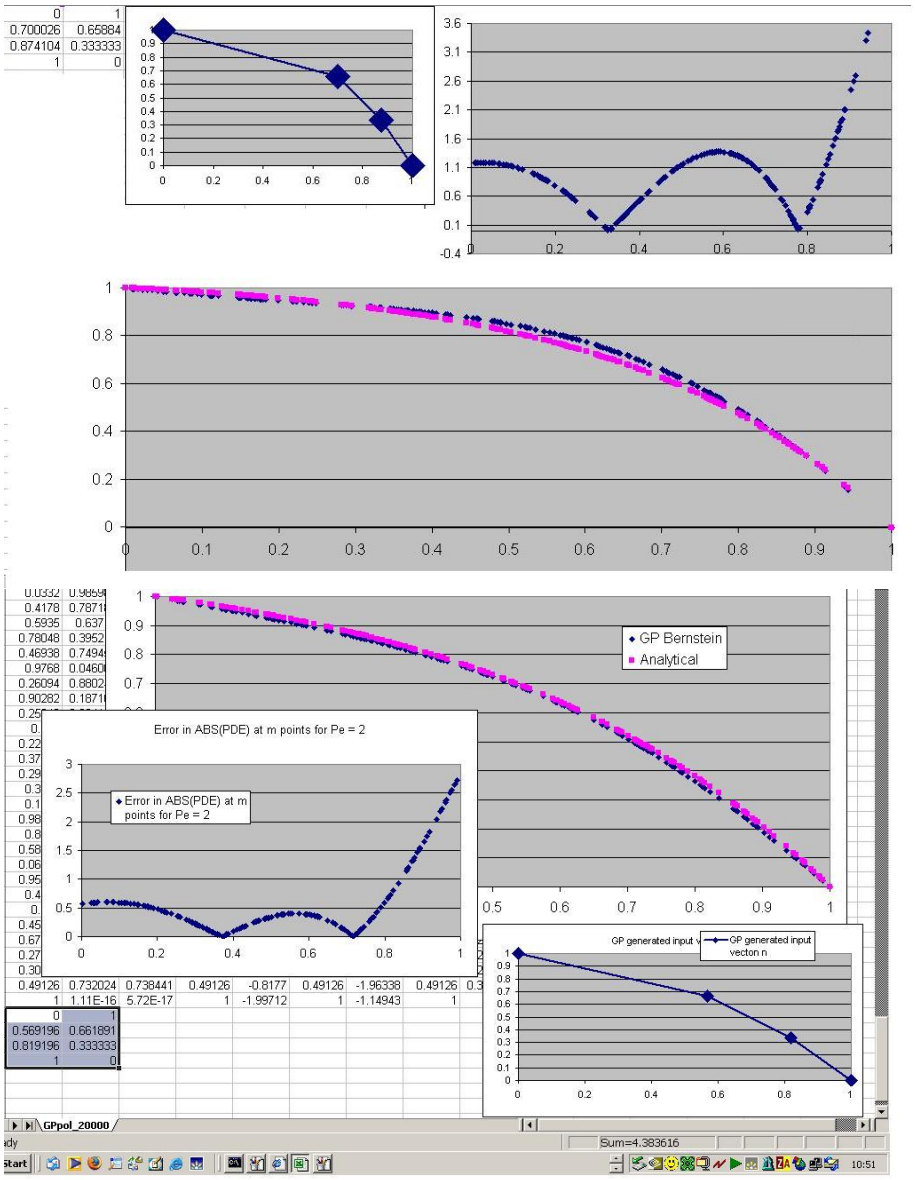

Figure 6: Two results at another low Peclet number $Pe = 2$ (the magenta curve is the exact analytical solution and the blue line is the SBI approximation) but now showing the distribution of the absolute error in the differential equation at the output points. The smaller pictures in the inset shows the position and value of the interpolation points and as in the previous figure, the output data was obtained at 200 random locations.

5.3 Higher value of Peclet number runs

As discussed previously the free parameter σ must be of equal value when building the matrix to compute the pre-image and when building the matrix to re-

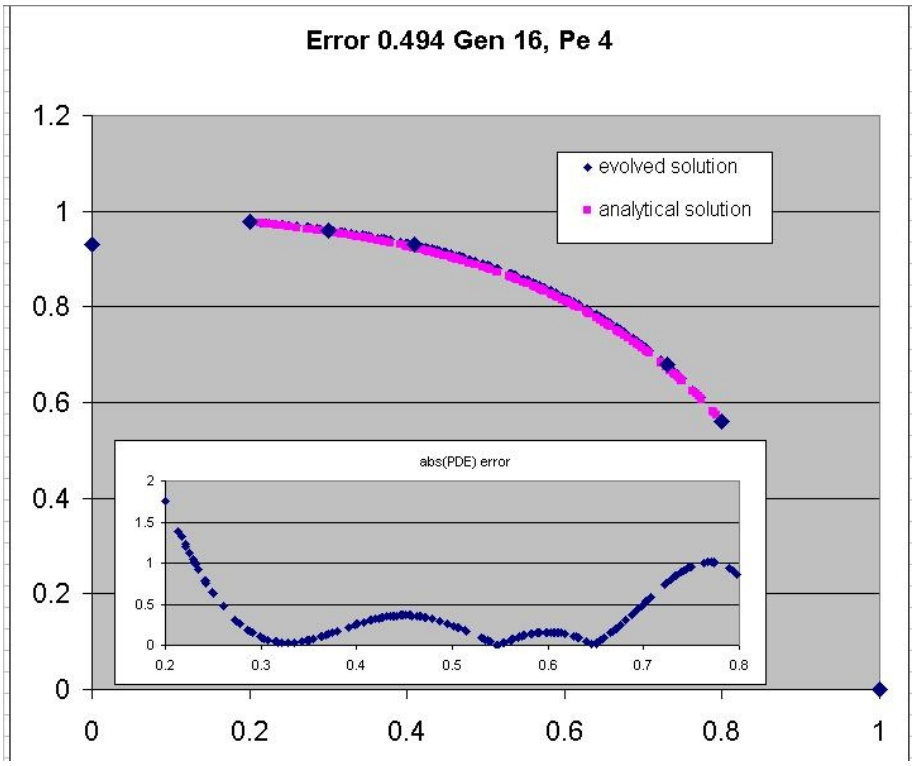

Figure 7: This result at $P = 4$ implemented the method of embedding the domain of evaluation into the (0.2-0.8). The image also illustrates the GP evolved interpolation points at $x = 0$ and $x = 1$ and two other points evolved inside the (0.2-0.8) region. Note that interpolation points corresponding to some boundary conditions are prescribed as constant values (not affected by evolution) at $x = 0.2$ and at $x = 0.8$.

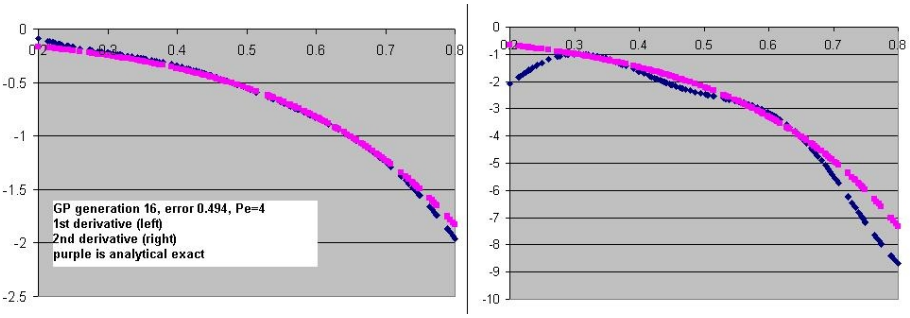

Figure 8: Graph of the derivatives calculated by the SBI approximation (in blue) as compared to the analytical solution for the derivatives (in magenta) at the output values. The graph on the left is the first derivative and the graph on the right is the second derivative.

covering the interpolation. If this is not the case then the points will not be

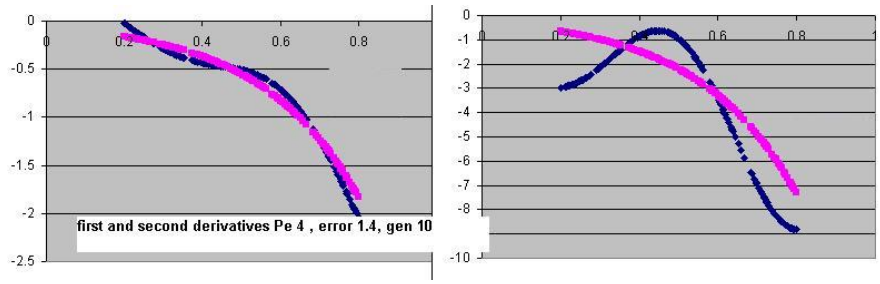

Figure 9: Candidate solution at an earlier generation. Graph of the derivatives calculated by the SBI approximation (in blue) as compared to the analytical solution for the derivatives (in magenta) at the output values. The graph on the left is the first derivative and the graph on the right is the second derivative. Better accuracy in the first derivative than in the second derivative is more clearly illustrated.

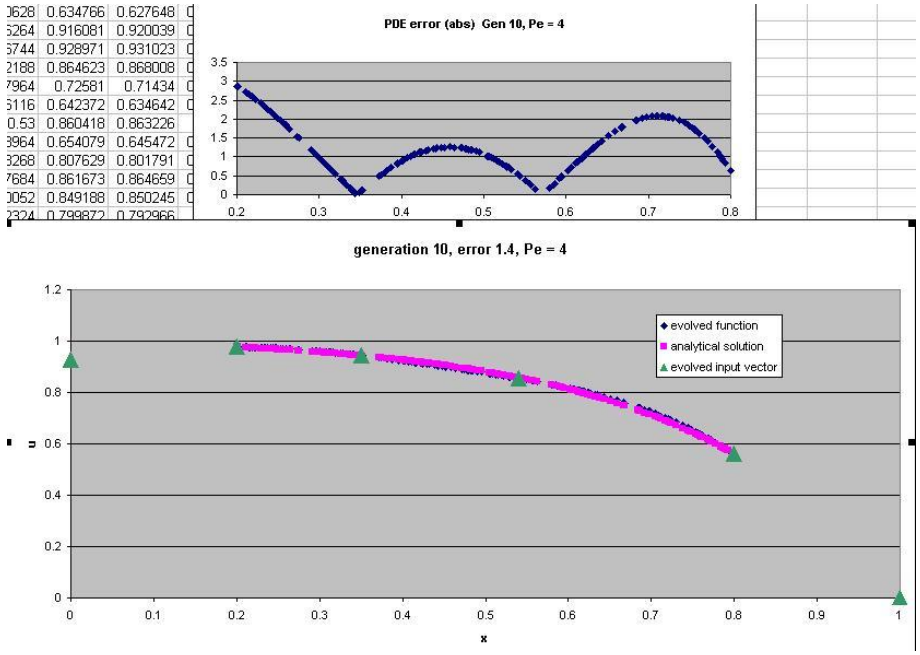

Figure 10: A result that used 6 interpolation points similar to the result of figure 7 that instead used 7 interpolation points

interpolated and the boundary conditions will not be satisfied. For this reason the value of σ was kept of equal value.

The value of σ in the SBI method was set at $\sigma = 1.0$ throughout all of the low Peclet number runs. However, for the higher Peclet number computations $P > 10$ it became necessary to drop the value of σ to lower values.

A value of $\sigma = 1.0$ does not result in good convergence properties at higher Peclet numbers. However, it was discovered that reducing the value of σ significantly could produce results which followed the right shape and were very close

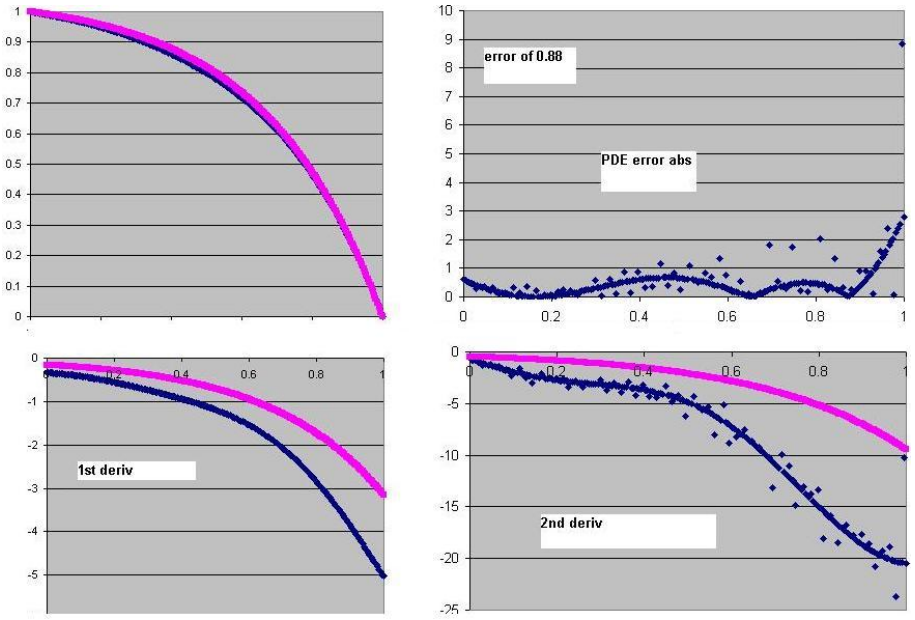

Figure 11: Illustrating derivatives computed using the finite difference measure: first derivative (left), second derivative (right).

to the analytical solution at high values of the Peclet number.

Figure 12 was a successful run at $P = 20$. These runs implement the embedding of the domain of computation in (0.2,0.8) with the appropriate scaling of the CDE so that the error as shown in the figure is the value of the absolute value of $0.6\frac{d^2y}{dx^2} - P\frac{dy}{dx}$. The scaling factor of 0.6 means that the Genetic Programming SBI approximation tackled a higher Peclet number (effectively 33).

Figure 12 illustrates two important observations:

1. consider the graph of the error and note how GP uses the buffer space (0.8,1.0) to turn the function upwards (so as to achieve a rapid lift in the function outside the area of interest);

2. note how GP has evolved more points than was required and simply clusters them (within the allowable gaps as explained previously) in the region near 1.0 effectively turning these into a single point.

We carried out only a few runs of the method on $Pe = 100$ using a GP population of $10,000$ running it at a "dangerously low value" of σ to $\sigma = 0.01$. Although the error remained quite high (L2 norm = 66.0 using 241 sampling points) the resulting values showed the correct shape for both the solution and its derivatives (figure 13). Note the "stair-casing effect" in the solution owing to the very low value of σ.

6 Discussion

The traditional view of numerical approximation is that an increasing number of approximation points is required to achieve a more accurate approximation (h

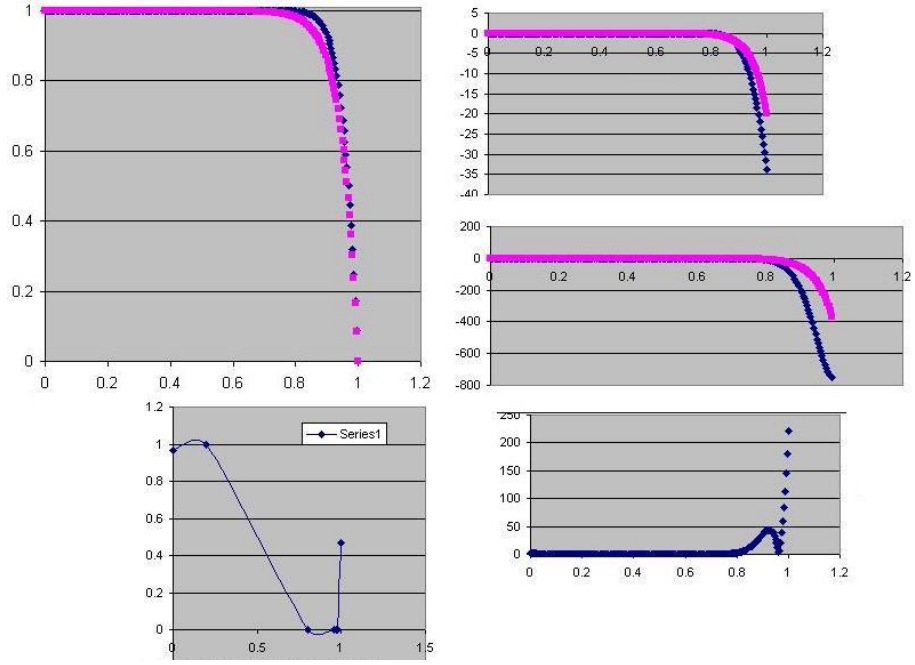

Figure 12: Results at higher Peclet numbers ($P = 20$) blue is the GP-SBI and magenta is the analytical solution. Clockwise from top left: solution; first derivative; second derivative; error; evolved and prescribed interpolation points (see text). Note that the first four figures have been rescaled to the (0,1) range from their original (0.2,0.8) range by the appropriate coordinate transformation.

refinement) and or a higher order of approximation is required (p refinement). However, Genetic Programming managed to obtain qualitatively good approximations at high Peclet number by manipulating the natural oscillations that are inherent in an approximation by manipulating the location of the sample points. It opportunistically evolved interpolation points outside of the domain of interest to turn the SBI appropriately. A low value of sigma assisted this process because it makes the resulting function more "stiff" as has been explained previously in this report.

7 Conclusions and further work

The method shows a lot of promise for a number of reasons:

1. It can be extended to multi-dimensions where different boundary conditions can be naturally imposed and the fitness measure only needs to concern itself with satisfaction of the differential equation as the boundary conditions are exactly interpolated.

2. Unlike previous methods it is impossible for the method to produce a large drop between points to return trivial solutions (a problem that had plagued earlier attempts at solving the problem with a regression GP)

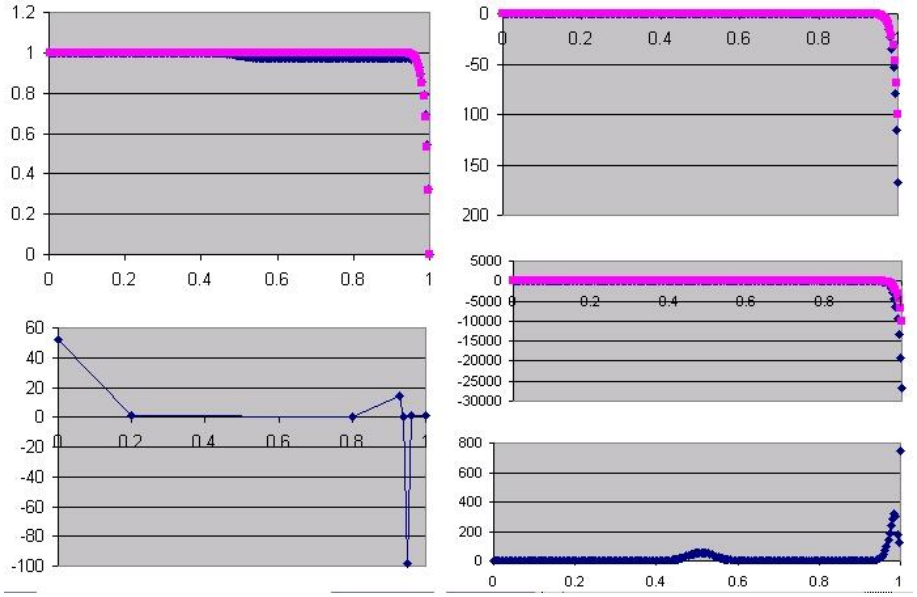

Figure 13: Results at higher Peclet numbers ($P = 100$) blue is the GP-SBI and magenta is the analytical solution. Clockwise from top left: solution; first derivative; second derivative; error; evolved and prescribed interpolation points. Note that the first four figures have been rescaled to the (0,1) range from their original (0.2,0.8) range by the appropriate coordinate transformation. GP has made a lot of effort to get the function to turn appropriately by manipulating the interpolation points outside of the domain of interest, i.e. in the regions (0.0,0.2) and (0.8,1.0).

3. It seems that only a few number of points need to be prescribed as interpolants. Consequently the Gauss elimination of the matrix system $A_{nn}^{-1}\mathbf{y}$ which needs to be performed at every fitness evaluation is currently inexpensive. However, we already have an excellent pseudo-inverse of this matrix that maintains the interpolation at the boundaries. We will be investigating this further.

4. Calculation of the derivatives using localized finite differences or some other technique can free up sigma. Currently, sigma needs to be set to a constant throughout to enable an easy computation of derivatives. If sigma could be varied by GP or prescribed for example at 2D boundaries to stiffen the points there, this could be a useful facility. Local calculation of derivatives may have its drawbacks.

5. Although the $Pe = 100$ solution of figure 13 exhibits a ramp like nature owing to the low sigma value, it is important to realize that sigma can be varied as a post processor to improve the gradients in the solution for the same set of fixed interpolation points.

6. We need to continue to investigate the behavior of the system and better to understand the search space for different Peclet numbers.

We also plan to work on the mathematics of the SBI:

- Improving the converge of the method at the edges of the domain. Presently, the method ranges from $O(h^4)$ to $O(h^6)$ over almost the entirely of the domain, however near the edges, this reduces to about $O(h^1)$, which is clearly a source of difficulty. In particular, estimates of the derivatives become increasingly difficult to obtain near either boundary point. The difficulty in part lies in the fact that in the integral formulation of the Bernstein function, the development of the approximation is done using a piecewise constant approximation to the data f_j. Increasing this, along with modifications of the Bernstein functions diffusion coefficient may lead to improved modelling of the interpolant and its derivatives near the edge of the domain.

- Extending stochastic methods to full multidimensional methods, i.e., not the one-dimensional interpolation done in each of the coordinate directions, has several advantages in dealing with the solution of multidimensional partial differential equations. The full development of this extension is quite feasible analytically, however carrying out the work remains to be done.

- Improving the computational speed of Bernstein Interpolation. Recent advances have reduced the cost of inverting the matrices to (n^2), which is achieved by modifying the structure of the deconvolution matrices so that they are Toeplitz. In addition, further advantage can be taken of block decomposition of the problem, or of using overlapping computational stencils, as is done for finite difference methods, however the goal is to achieve a recursive structure, thus leading to methods which are computationally on the order of the FFT to compute.

References

[1] K W Morton (1996) *Numerical Solution of Convection Diffusion Problems*, Chapman & Hall, 0-412-564440-8.

[2] E Süli, T Murdoch, K W Morton, (1980) *Optimal error estimation for Petrov-Galerkin methods in two dimensions*, Oxford University Numerical Analysis Report NA-90/22 (http://web.comlab.ox.ac.uk/oucl/publications/natr/na-90-22.html).

[3] J R Koza (1992) *Genetic Programming*, MIT Press.

[4] D Howard, S C Roberts (2001) *Genetic Programming Solution of the Convection Diffusion Equation*, GECCO 2001, pg. 34-41.

[5] J Kolibal, C Saltiel (2005) *Data Regularization Using Stochastic Methods* submitted to: SIAM Journal on Numerical Analysis. Paper ID is: Manuscript # 063083.

[6] K W Morton, I J Sobey (1991) *Discretization of a convection-diffusion equation*, Oxford University Numerical Analysis Research Report NA-91/4 (http://web.comlab.ox.ac.uk/oucl/publications/natr/na-91-4.html).

[7] P Gresho, R L Lee (1981) *Don't suppress the wiggles, they are telling you something.* Computers & Fluids 9, 223-253.

Appendix A : Current state of the art and its limitations

This section describes the limitations of any numerical approximation of non-self-adjoint partial differential equations (the heat equation, the Navier-Stokes equations, etc.) with the Weighted Residuals Methods (WRM).

Consider a simplified member of this class of differential equation: the 1D steady-state homogeneous convection diffusion equation (a two point boundary value problem):

$$\begin{aligned} \frac{d^2T}{dx^2} - P\frac{dT}{dx} &= 0 \\ T = 1.0 \text{ at } x &= 0.0 \\ T = 0.0 \text{ at } x &= 1.0 \end{aligned}$$

where T is the temperature unknown over the domain [0,1] and P is the Peclet number or the ratio of convection to diffusion.

The Weighted Residuals Method for numerical solution of this differential equation divides the domain [0,1] into a number of regions (called cells or elements) by means of nodes. Simple functions ϕ (usually linear or quadratic) are defined on a node and have a small local support over the adjacent two elements as illustrated in figure. At any point in (0,1) the temperature T is obtained by taking the sum of products of nodal temperatures T_i and nodal functions ϕ_i or $T = \sum_{i=1}^{n} \phi_i T_i$, where n is the number of such nodal functions and temperatures.

Consider the self-adjoint version of the convection diffusion equation when $P = 0$ and the differential equation at any point with the nodal functions and unknowns:

$$\sum_{i=1}^{n} \frac{d^2\phi_i}{dx^2} T_i = 0$$

The Weighted Residuals Method writes a nodal differential equation by pre-multiplying this expression with another function, the weighting function w_i at node i and defined similarly to the ϕ_i. Furthermore it takes the expression under the integral sign:

$$\sum_{j=1}^{n} \sum_{i=1}^{n} \int w_j \frac{d^2\phi_i}{dx^2} T_i = 0$$

Integration by parts is required to reduce differentiability requirements on ϕ_i (weak formulation) resulting in:

$$\sum_{j=1}^{n}\sum_{i=1}^{n}\int \frac{dw_j}{dx}\frac{d\phi_i}{dx}T_i = \sum_{j=1}^{n}\int_{\Omega} w_j\frac{d\phi_i}{dx}T_i$$

where Ω is the boundary integral (it has lower dimensionality: in the 1D case it is a nodal value; in the 2D case it is an edge, and in the 3D case it is a surface).

This is now a matrix system of equations of the form:

$$\begin{aligned} Au &= b \\ a_{ji}u_i &= b_j \\ a_{ji} &= \int \frac{dw_j}{dx}\frac{d\phi_i}{dx} \\ b_j &= \int_{\Omega} w_j\frac{d\phi_i}{dx}T_i \end{aligned}$$

where A is the square matrix of size nxn u is the vector of nodal unknowns, i.e. T, and b is the right hand side vector both of size n. In order to obtain the vector of nodal unknown matrix A needs to be inverted, i.e. $u = A^{-1}b$. Usually, the system of equations is solved by Gauss elimination. Note that the system is usually banded because functions ϕ_i and w_i have local support.

An important assumption is the Galerkin weighting. This replaces w_i by ϕ_i to make matrix A symmetric and positive definite:

$$a_{ji} = \int \frac{d\phi_j}{dx}\frac{d\phi_i}{dx}$$

As stated, this matrix system coincides with the Ritz Method. A functional $F(u)$ is a landscape that must be searched. However, this landscape is defined by a quadratic functional and therefore the minimizer is immediately provided by setting its first derivative with respect to u to zero. The resulting $\hat{u}$ is a minimizer provided the matrix A is positive definite.

$$\begin{aligned} F(u) &= \frac{1}{2}u^T Au - b^T u + c \\ \frac{dF}{du} &= Au - b \\ \frac{d^2F}{du^2} &= A \\ \frac{dF}{du} &= 0 \text{ or } A\hat{u} = b \end{aligned}$$

The solution of the matrix system of equations defined by a Galerkin weighted residual method for a self-adjoint problem ($P = 0$) is a "direct" search method (one step) of search over the landscape $F(u)$.

Consider what happens when $P > 0$. The first effect is to make the matrix A an unsymmetrical matrix. More importantly, for a given P and width of each element (the spacing between nodal points in [0,1]) the matrix A is no

longer positive definite. When this occurs then the equivalence between the Ritz method and the Galerkin WRM is gone. Solution of the matrix system no longer provides the minimization of the functional $F(u)$ because this is no longer quadratic. The solution of the matrix equations is highly oscillatory and has no resemblance to the true solution.

Over the past thirty five years, engineers and mathematicians have tried to address this problem by restoring the Ritz equivalence through the manipulation of the weighting functions w_i. Methods which use a different weighting function to ϕ_i are popularly known as Petrov-Galerkin methods. C. J. Hemker, a Dutch mathematician, introduced a function for w_i which perfectly restores the equivalence to the Ritz method for the 1D case (the case presented in this section). This function is in turn a function of P and of the difference between the points. The finite difference method has introduced equivalent schemes to those of the finite element method in one dimension (Morton and Sobey, 1991) called "upwind differences". However, in all cases these Petrov-Galerkin functions cannot easily constructed (they are certainly not the cross product of 1D Hemker functions) for: (a) multi-dimensional differential equations (partial differential equations); (b) non-linear differential equations (such as Navier-Stokes equations) other than in the most trivial of cases.

In partial differential equations and in non-linear differential equations, such techniques as: upwind differences (finite differences method); artificial viscosity (finite volume method); streamline upwinding or SUPG (finite element method); can provide deceptively smooth result which are solutions to a different set of differential equations to the one intended (Gresho and Lee, 1981). Over the past 35 years this problem has been addressed but never solved. The reasons is that properly restoring the equivalence to the Ritz method for such differential equations is a formidable challenge which is problem and geometry dependent. Even in simple cases when it is known how to it is just too difficult to do so, in other cases the required function is unknown.

Appendix B - Additional information on SBI

The new method has excellent representational fidelity (appendix D) and a good candidate for function recovery from experimental measurements which contain clutter. Its other advantages are:

1. This technique is robust, i.e., it always converges numerically;
2. It is adaptable to any data ordering in any dimension;
3. The method is hierarchical, i.e., data can be added at any time to improve the results.
4. The method is not harmonic, but it is spectral, i.e., it is possible to construct a sequence of components whose sum converges to the same result.
5. The method can be made adaptive through:
   (a) spectral decomposition,
   (b) windowing,

(c) tuning the convolution coefficients (these can be functions of the data), and

(d) choosing the stochastic components using statistical theory.

SBI never throws away spectral information, it simply attenuates it. For example, all recovered functions are infinitely differentiable.

Another difference to other spectral methods concerns the approximation in L1. Figures 14 to 16 illustrate equivalent approximations in L1. The recovered function is similarly close to the data in all cases (in terms of the evolution of the function). The method is very flexible because different results can resemble analysis by a spectral method and are obtained by adjusting a free deconvolution parameter.

Figure 17 illustrates this feature extraction in the region from $x = 1.7$ to $x = 2.3$ (approximately). It is the result of the following procedure:

1. Take the data from the function recovery that used 1,000 as the deconvolution parameter (right plot of figure 15).

2. Extract the smooth function on a dense set (about 1,000 points) in the region of interest.

3. Re-approximate using the previous approximation.

The resulting curve in figure 17 does a much better job of approximating the function in this region. The procedure amounts to cloning this segment of the curve, intensifying its attributes and then extracting the smooth underlying feature from this curve. This may be compared with the response to function recovery using 1,000 as the deconvolution parameter (right plot of figure 15) from $x = 1.7$ to $x = 2.3$. It illustrates "windowing properties" (design of CWT and window Fourier transforms were motivated by this need).

That is not quite multi-resolution analysis the way wavelets do it, but it is comparable. An advantage of this novel approach is that it easily extends to multiple dimensions (interpolation is a bit more difficult to do, but approximation is not).

Appendix C: SBI and Data Approximation

Consider Fig. 18. Starting with an underlying smooth function, sampling the function and adding noise at each sample location, the Bernstein function recovery obtains a smooth representative function for the data based on the noisy data. The following questions and answers are offered with respect to figure 18:

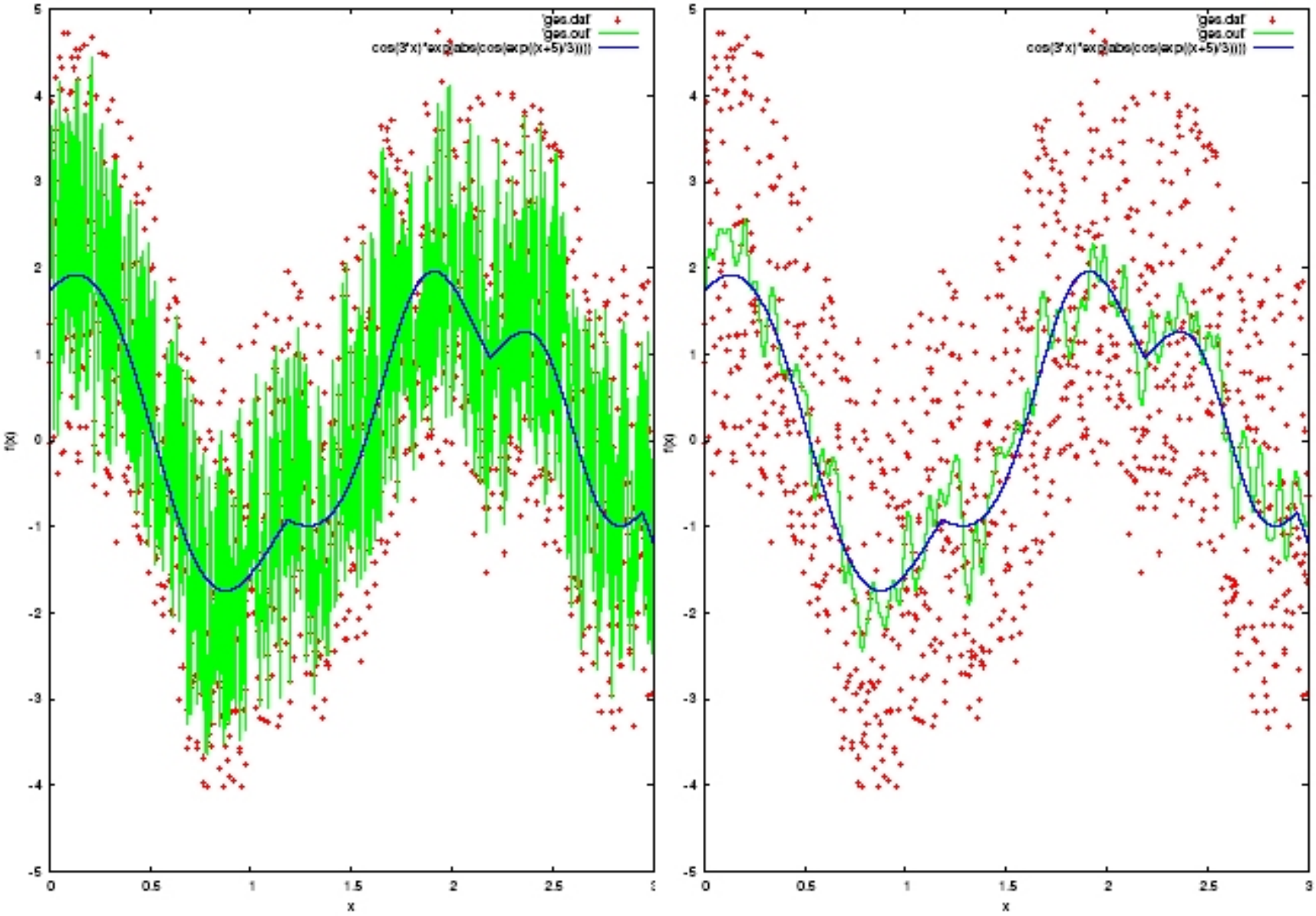

Figure 14: Function recovery with deconvolution parameter value equal to 1 (left) and equal to 10 (right).

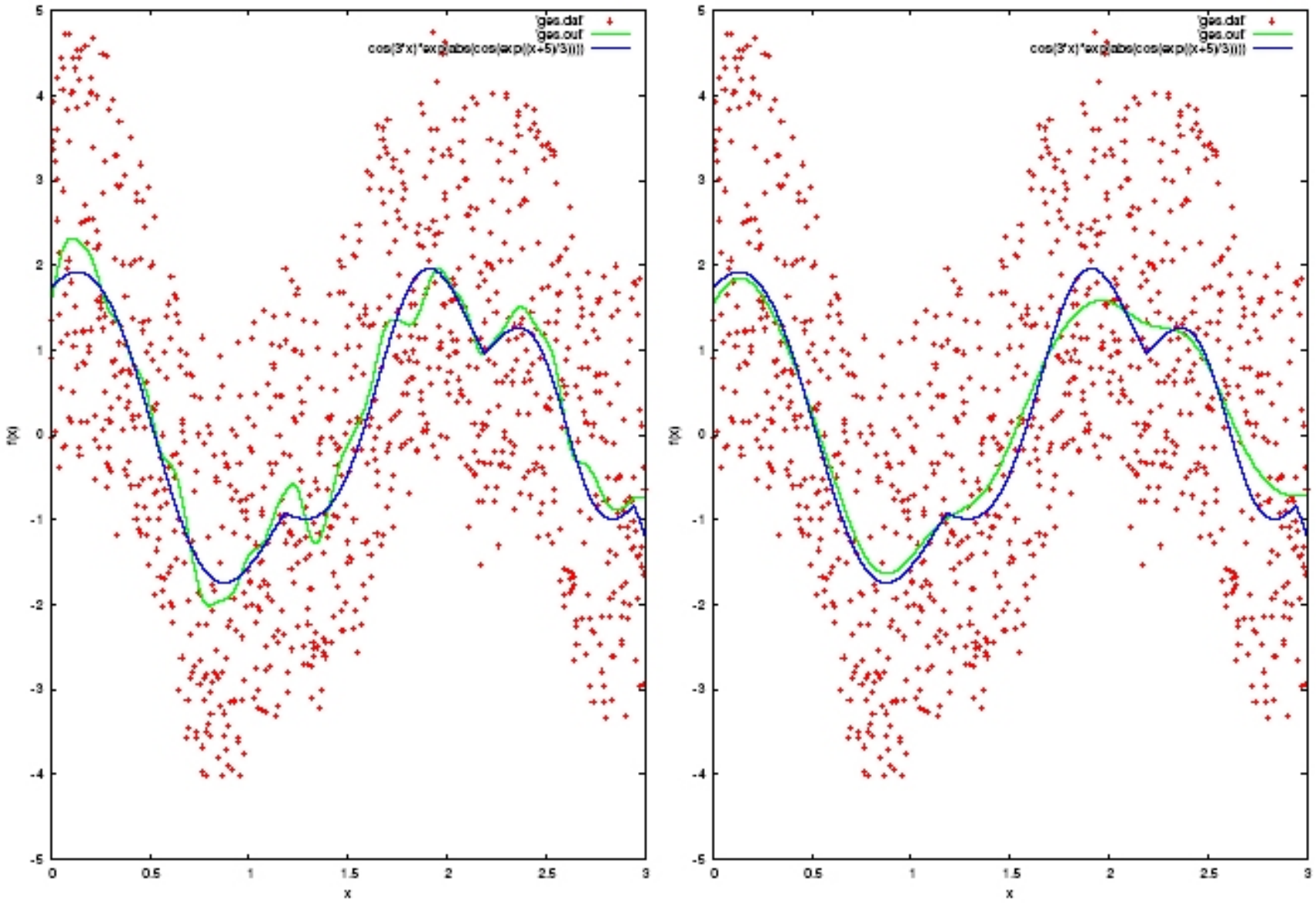

Figure 15: Function recovery with deconvolution parameter value equal to 100 (left) and equal to 1,000 (right).

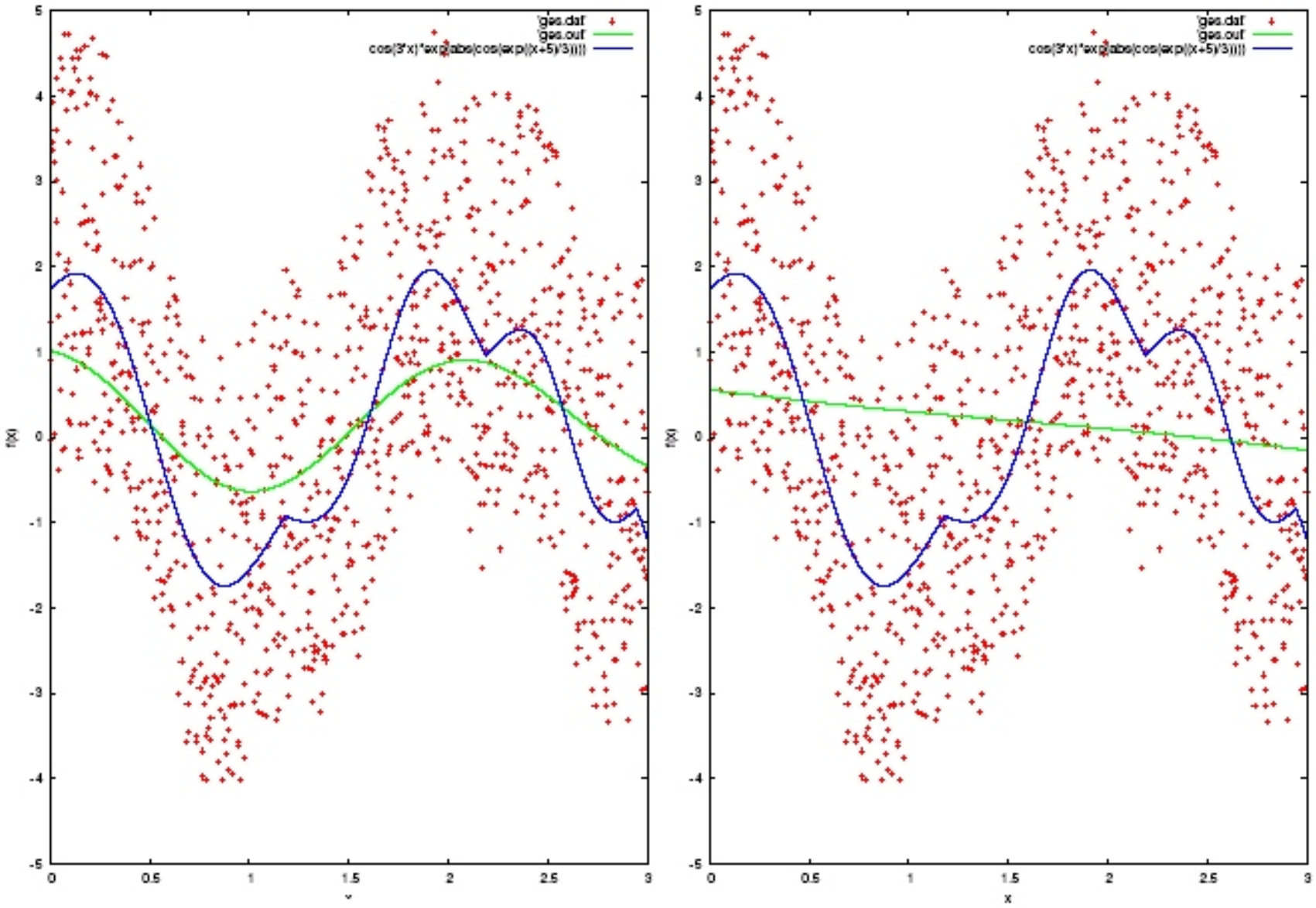

Figure 16: Function recovery with deconvolution parameter value equal to 10,000 (left) and equal to 100,000 (right).

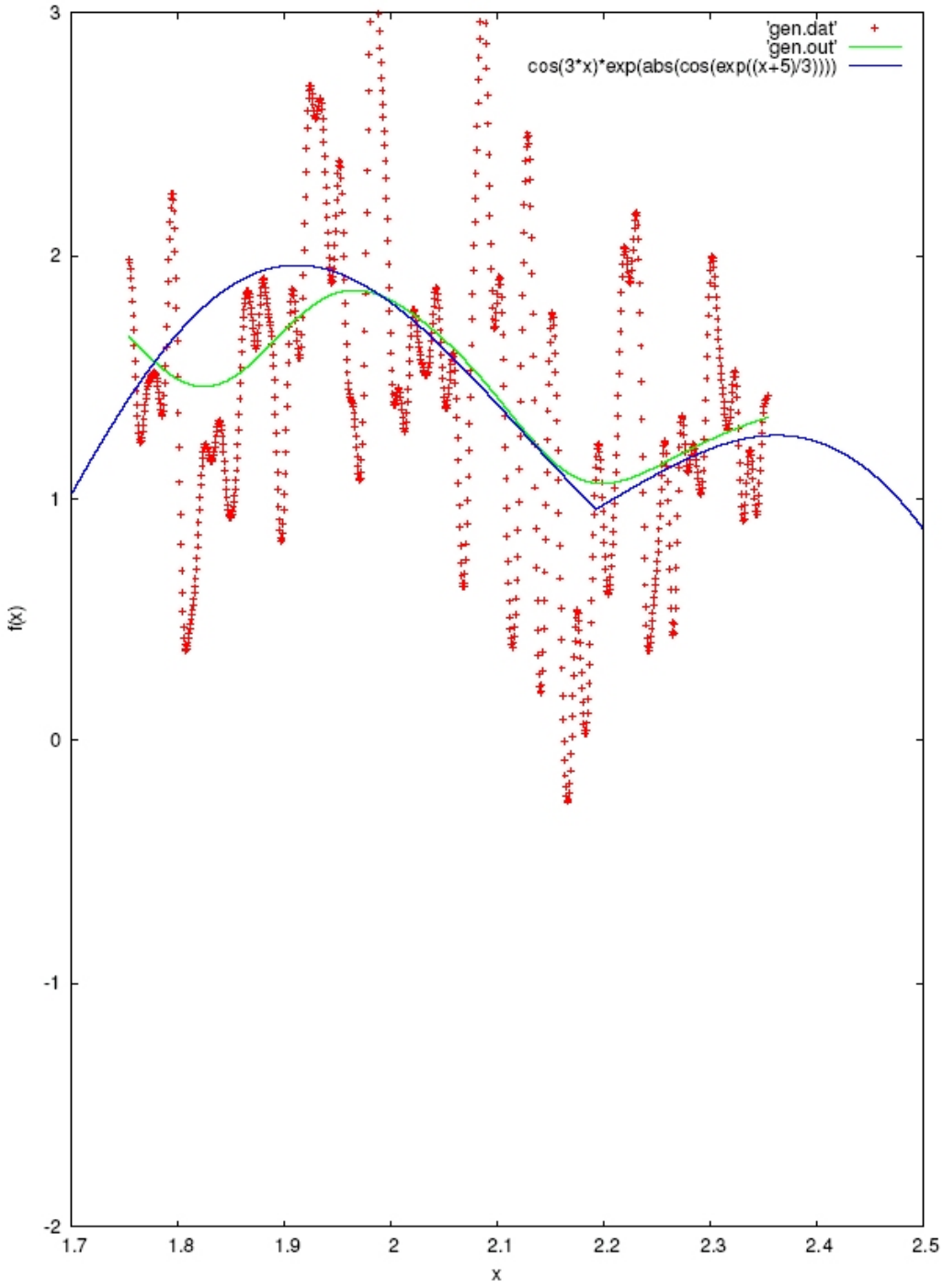

Figure 17: Feature extraction.

Comment/Question	**Answer**
It looks a reasonable fit.	A score of methods may give similar results. The method is useful because it gives good results in many cases without tweaking it.
Firstly, about the data: 1. Is x uniformly sampled on [0,1]? 2. Is the variance constant across [0,1]?	 It was uniformly sampled, however it need not be, i.e. the method works on any arbitrary grid. Were the random perturbations generated of the data? If that is the question, then the answer is yes. Other models are possible but that was the simplest.
Secondly, about the fitted curve: 1. What are the boundary conditions? The fit near x=0 looks good, but it is not supported by the data in that region. Some standard methods (e.g. a standard kernel-based method) may perform poorly at the boundaries, but there are modifications to handle that. You could probably get a similar fit with splines with appropriate boundary conditions. 2. How do you determine model order? The wiggle towards x=1 seems odd, but who is to say that it is not supported by the data?	 There are no boundary conditions, results are from the standard linear model, i.e. no attempt at any corrections using feedback from any additional information (though doing so could have enhanced the results). The SBI method of convolution does not come from solution of a differential equation. The model does not have any order since it is non-polynomial. Instead there is a measure (locally) of the distance that the curve is away from the original data (i.e., it is a convolution with a mollifier, and can move it further away or closer to the original curve).
Thirdly about the method: 1. Can the method be used to obtain confidence bounds on the curve?	 The model by itself has no statistical components, and since it is not an L2 fit, there are no best fits, only families of functions which will be appropriate fits.

In considering these answers readers should note that:

1. In Mathematics, a mollifier is a smooth function $f\colon \mathbb{R}^n \to \mathbb{R}$ such that: $f(x) \geq 0$ for all $x \in \mathbb{R}^n$; f is normalized such that $\int_{\mathbb{R}^n} f(x)dx = 1$; $f(0) = 1$. The process requires a mollifier when it is desirable to construct smooth approximations or interpolations to the data. This is usually desirable although this is not necessary. Use of any other normalized function to construct the row space of the stochastic matrix will construct approximations that rapidly vary from point to point. Using a mollifier means that the information that is filled in between the know points also varies smoothly and predictably between these points.

2. Why is the method "non-polynomial"? The representative functions are close to polynomials on a regular grid but they are not polynomials. The

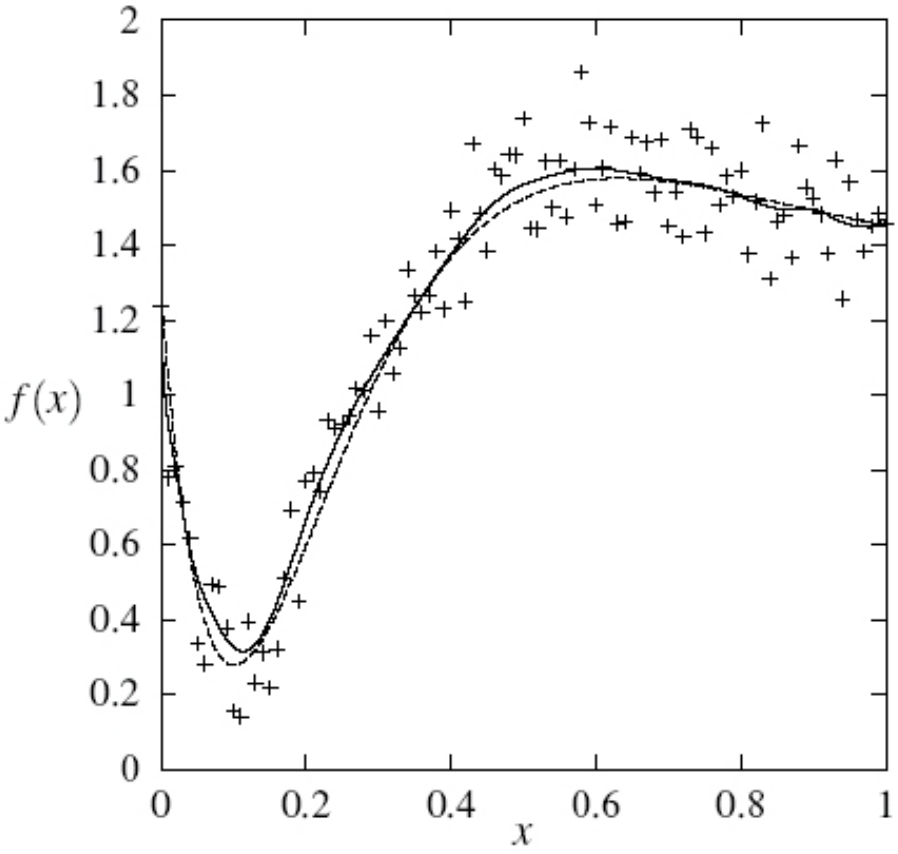

Figure 18: Illustrating function recovery using Bernstein functions, K_n, based on sampling the function $f(x) = u(x) + \epsilon(x)$ (crosses). The smooth function $u(x) = 4.26(e^{-3x} - 4e^{-6x} + 3e^{-9x}) + 1.28$ is shown using a dotted line and a solid line shows K_n recovered from the data, $\{(x_k,f_k)\}$. The perturbations are obtained using a normal distribution $N(0,1)$ scaled by 0.2 to perturb $u(x)$.

polynomials they are close to are the Bernstein polynomials. For the approximations, the functions are composed of convolutions of error functions, and for the interpolant they are more complicated.

Appendix D: Fidelity of SBI

An important consideration of any method which aims to represent functions is how well it can do that task. Ultimately, what matters is the representational fidelity of the method. When it gets that right, everything else follows (multi-resolution analysis, scalability, and everything else).

This appendix revisits the case used in figures 14 to 16 in which the underlying function:

$$cos(3 * x) * exp(abs(cos(exp((x + 5)/3))))$$

is perturbed, but this time to interpolate the data along with the noise.

The first experiment takes the 1,000 points and interpolates these to 10,000 points uniformly spaced on the interval [0:3]. This gives a richer characterization of the data, finding a representative smooth function which fits the data and the noise.

This is a lot of data. Thus, figure 19 shows the data plotted in the interval [1.7:2.3], and figure 20 gets a closer look in the interval [1.7:1.8].

From these figures it is noticeable that the interpolant has peaks and valleys where the 1,000 data set did not. In effect, the method is predicting that the representative smoothing to the noisy data would be shaped as shown in these two figures.

Notice in the graph on the interval [1.7:1.8] that the curve is smooth.

Instead of the analysis of figures 14 to 16 which used the original coarse data, a further experiment convolves the interpolated data containing these 10,000 points, which were obtained from the data by interpolation. Figure 21 displays the results that speak for themselves. The method finds all of the features of the underlying curve, except for the sudden drop-off at $x = 3$. However, there is not enough information at this point to discern that from the data. It slightly overshoots and smooths out the features, but given the level of noise, and the fact that this was a biased perturbation, it has done credibly well at feature extraction. It has correctly predicted all of the features of the underlying smooth function.

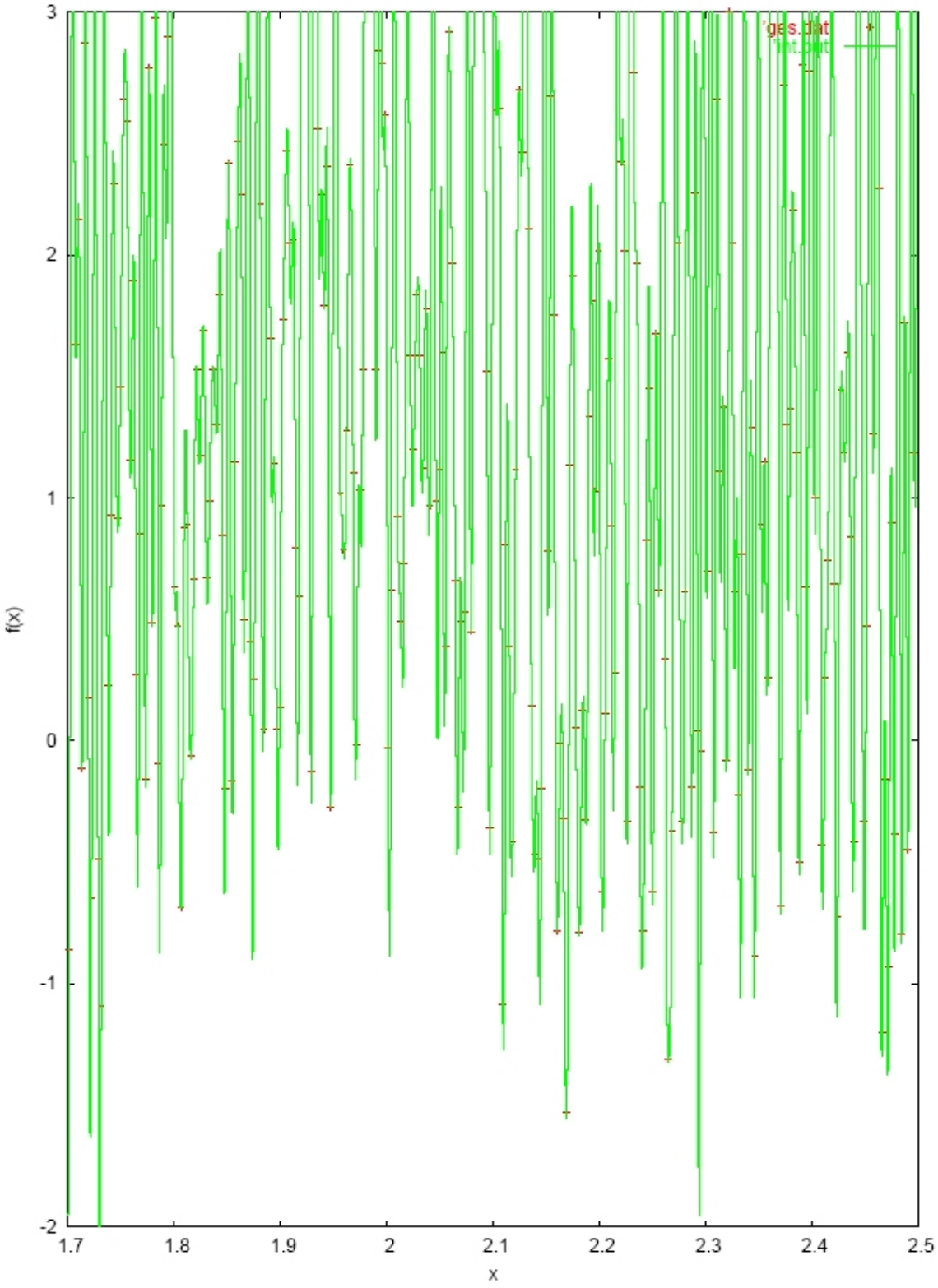

Figure 19: Data plotted in the interval [1.7:2.3]

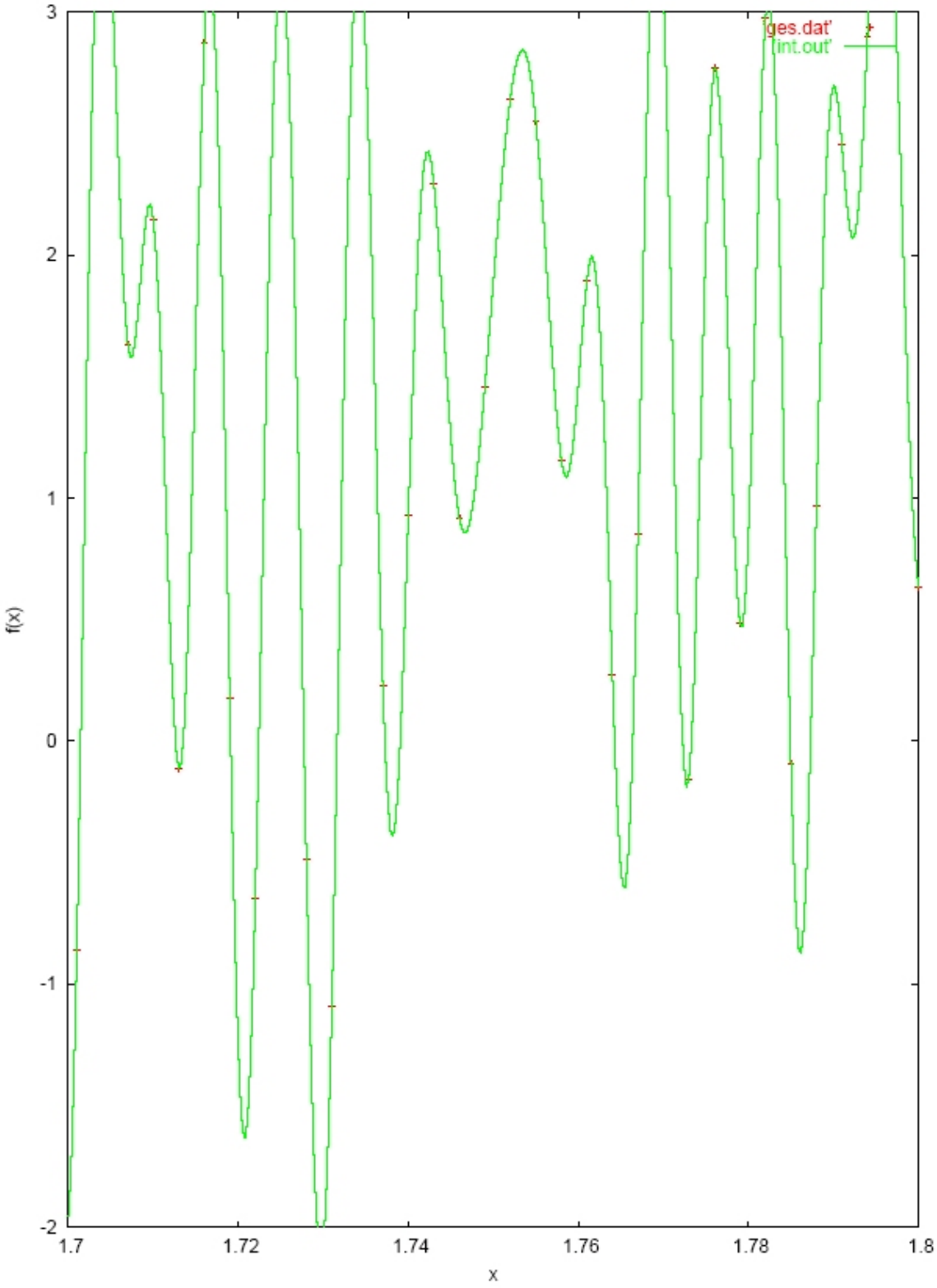

Figure 20: Closer look in the interval [1.7:1.8]

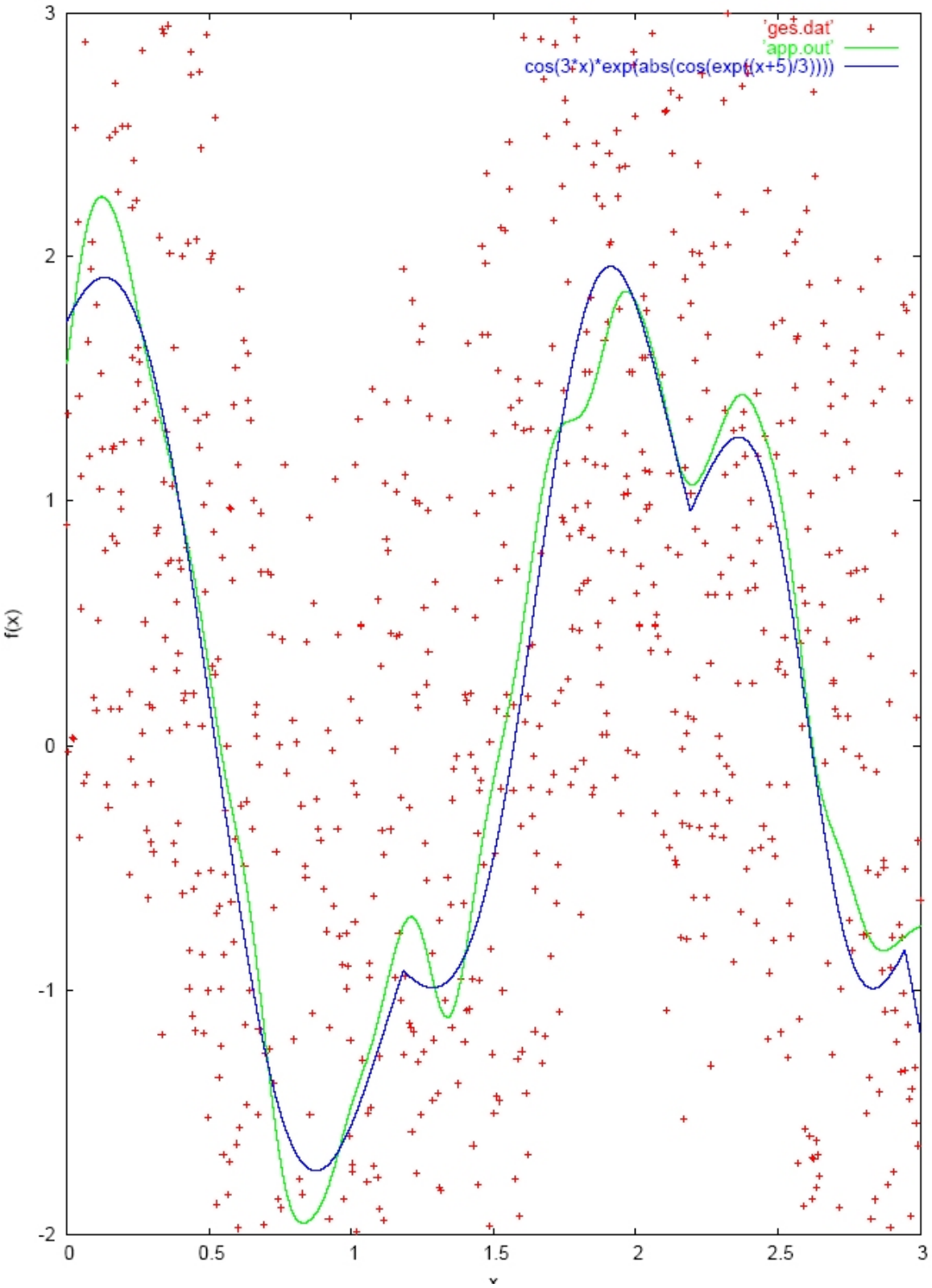

Figure 21: The method finds all of the features of the underlying curve.