Towards a Bell-Curve Calculus and its Application to e-Science

Towards a Bell-Curve Calculus and its Application to e-Science

Lin Yang[1], Alan Bundy[1], Dave Berry[2], and Conrad Hughes[1]
l.yang@ed.ac.uk bundy@inf.ed.ac.uk daveb@nesc.ac.uk conrad@nesc.ac.uk

When we analyse some quality of service (QoS) properties, such as run time, accuracy and reliability, errors need to be taken into account. Although interval arithmetic can deal with worst-case analysis; for average-case analysis, we need some other methods. In this paper we will introduce the importance and methodology of developing a bell-curve calculus. We will explain the status of our current research and perform analysis on problems we have encountered.

1 Introduction

Grid computing has an almost ten-year history since it was derived, from an analogy to the power Grid, to denote a proposed distributed computing infrastructure for advanced science and engineering collaborations [1]. It is strongly required by consumers, scientists, engineers, enterprises, countries, and even the whole world to share resources, services and knowledge [2]. This sharing is supported and implemented by web services, software systems designed to support interoperable machine-to-machine interaction over a network. These services can be composed to form workflows in all kinds of structures. It is very helpful to measure the performance of the services because their quality affects the quality of the Grid directly.

In scientific workflows, experimental data is generated and propagated from one service to another. It would be useful to get rough estimates of various QoS properties, e.g. reliability, accuracy, run time, etc. so that e-Scientists could do analysis and evaluations on either services or data produced. We have previously thought about the use of interval arithmetic to calculate error bounds on such estimates. The idea is to extend a numeric value to a number interval, e.g. we use the interval [41, 43] to represent the possible value of 42. Extended numeric analysis is used as the way of interval propagation in workflows. The simplest example is for a unary and monotonically increasing function f(x), the extended function $f^*([a, b]) = [f(a), f(b)]$ [3]. Using interval arithmetic and propagating error bounds will get the biggest accumulative error during workflow executions, so it is a good method for doing a worst-case analysis.

However, in more common situations, e-Scientists may want to know the likelihood of each data value. So for average-case analysis, we propose to use normal distributions (bell curves) to add the concept of probability to differentiate the likely from the unlikely values of QoS properties. That is, if we associate the probability density function (pdf) of bell curves with the estimate, then some values in the interval have a higher probability than others. Figure 1 shows the pdf and cumulative density function (cdf) of a standard bell curve.

So now the questions are:

(1) Can we use bell-curve calculus to describe QoS properties and what are the advantages and disadvantages?

(2) How can we define a bell-curve calculus?

We will discuss this further in the next two sections.

2 Why A Bell-Curve Calculus

Initial experimental evidence from DIGS[4] suggests that bell curves are a possible approximation to the probabilistic behaviour of a number of QoS properties used in e-Science workflows, including the reliability of services, considered as their mean time to failure; the accuracy of numerical calculations in workflows; and the run time of services. Moreover, the Central Limit Theorem also gives us some theoretical support by concluding that:

> *"The distribution of an average tends to be Normal, even when the distribution from which the average is computed is decidedly non-Normal*[5]*."*

Furthermore, from the mathematical description of bell curves, we can see that the biggest advantage of using bell-curve is that its

[1] School of Informatics, University of Edinburgh

[2] National e-Science Centre

[3] Personal communication with Prof. Alan Bundy (BBN 1416)

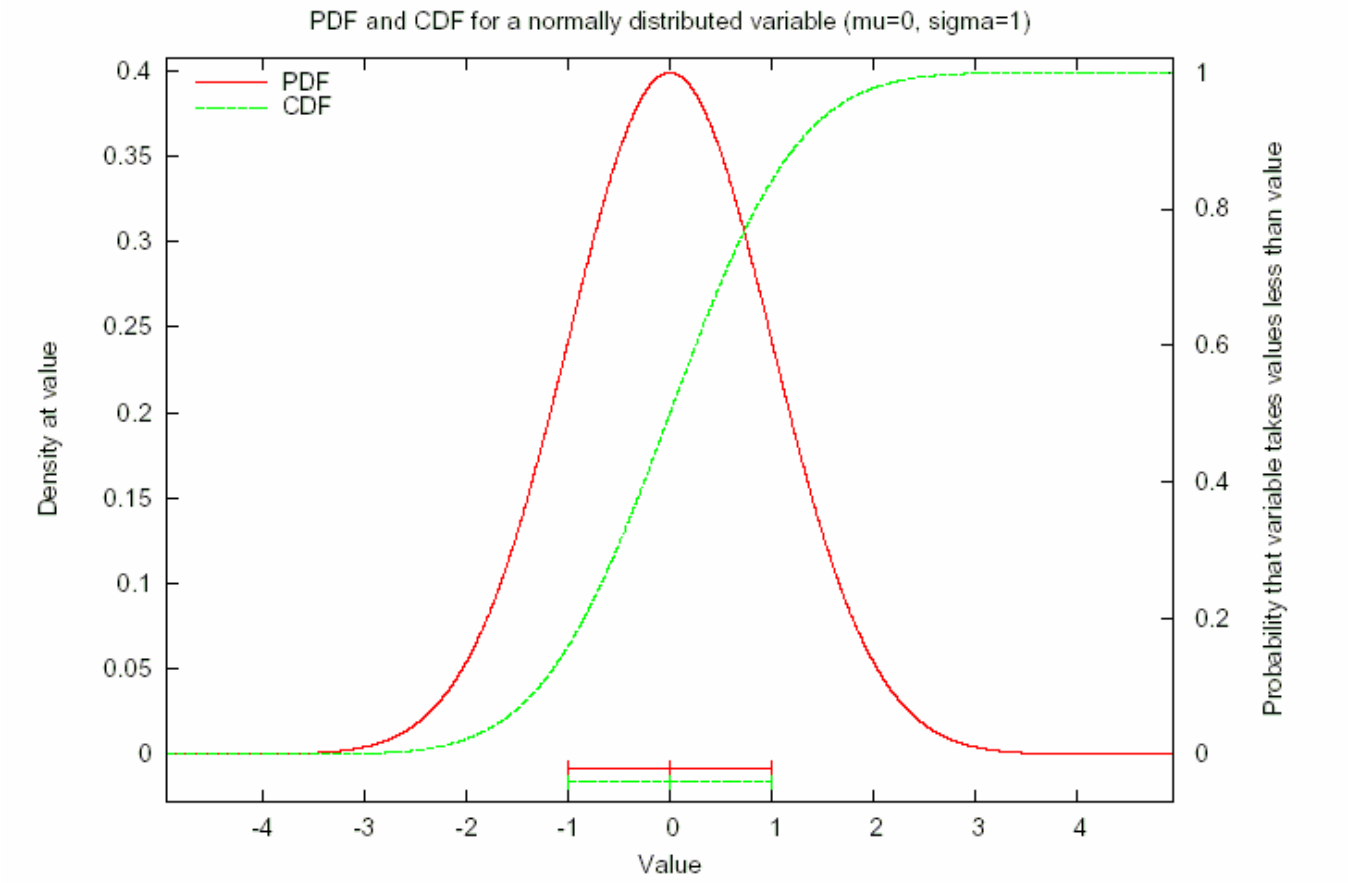
[4] DIGS (Dependability Infrastructure for Grid Services) is an EPSRC-funded project, to investigate in fault-tolerance system and other quality of service issues in service-oriented architectures

[5] http://www.statisticalengineering.com/central_limit_theorem.htm

probability density function (pdf) $p(x) = \frac{1}{\sqrt{2p}\,s} e^{-\frac{(x-m)^2}{2s^2}}$ has only two parameters: mean value m and standard deviation s, where m decides the location of a bell curve and s decides the shape of a bell curve. While evaluating the performance of a workflow, we need to gather all the possible data values of the QoS properties we analyse from all the input services is needed. We do calculations and analysis using the information and pass the results through the whole workflow. It will be a big burden if we transfer and calculate all the possible data values one by one. Now using bell curves which have only two parameters, the job becomes easier. All we need to do is to store and propagate the two parameters in workflows and a bell curve can be constructed at any time from m and s.

Then we will see if we can calculate the QoS properties of a whole workflow from the corresponding properties of its component services, namely if we can define some inference rules to derive the QoS properties of composite services from the correlative properties of their components.

FIGURE 1
THE PDF AND CDF CURVES OF A STANDARD BELL CURVE



The graph shows a standard bell curve with parameters – mean value m =0 and standard deviation s =1. The red curve is the pdf (probability density function) curve, indicating the probability of each possible value of variable x. It can be generally presented as $p(x) = \frac{1}{\sqrt{2p}\,s} e^{-\frac{(x-m)^2}{2s^2}}$. The green curve is the cdf (cumulative density function) curve, integrated from its pdf. It gives the probability that a normally distributed variable will output a value $\leq$ x.

We consider four fundamental methods to combine Grid services (we use services S_1 and S_2 to represent two arbitrary services)[6].

[6] Personal communication with Prof. Alan Bundy (BBN 1509)

Sequential: S_2 is invoked after S_1's invocation and the input of S_2 is the output of S_1.

Parallel_All: S_1 and S_2 are invoked simultaneously and the outputs are both passed to the next service.

Parallel_First: The output of whichever of S_1 and S_2 first succeeds is passed to the next service.

Conditional: S_1 is invoked first. If it succeeds, its output is the output of the workflow; if it fails, S_2 is invoked and the output of S_2 is the output of the whole workflow.

In terms of the three QoS properties and four combination methods, we have twelve fundamental combination functions (see Table 1). For instance, the combination function of run time in sequential services is the sum of the run times of the component services.

TABLE 1
THE TWELVE FUNDAMENTAL COMBINATION FUNCTIONS

	Seq	Para_All	Para_Fir	Cond
run time	sum	max	min	cond1
accuracy	mult	combine1	varies?	cond2
reliability	mult	combine2	varies?	cond3

The table shows the twelve fundamental combination functions in terms of three QoS properties and the four basic combination methods. Sum, max, min and mult represent respectively taking the sum, maximum, minimum and multiplication of the input bell-curves. Varies means the functions have not been defined and may have any form according to various situations. Cond1-3 are three different conditional functions and their calculation depends on the succeeding results.

Here we convert the formula of bell curve to a function in terms of m and s, then get the bell curve function as

$$bc(m,s) = l\,x.\frac{e^{-\frac{(x-m)^2}{2s^2}}}{\sqrt{2p}\,s}.$$

Our job is to define different instantiations of the combination functions applying to different QoS properties and different workflow structures.

3 Methodology

Suppose we have two bell curves corresponding to two services. We present them using a bell curve function defined in Section2 as $bc(m_1, s_1)$ and $bc(m_2, s_2)$ and the combination function as $F(bc(m_1, s_1), bc(m_2, s_2))$, which is actually a function in terms of four parameters -- m_1, m_2, s_1 and s_2.

Therefore we have two main tasks:

(1) Can we find a satisfying instantiation of $F(bc(m_1, s_1), bc(m_2, s_2))$ for every situation we are investigating?

(2) How good will our approximations be? 'Good' here means accurate and efficient.

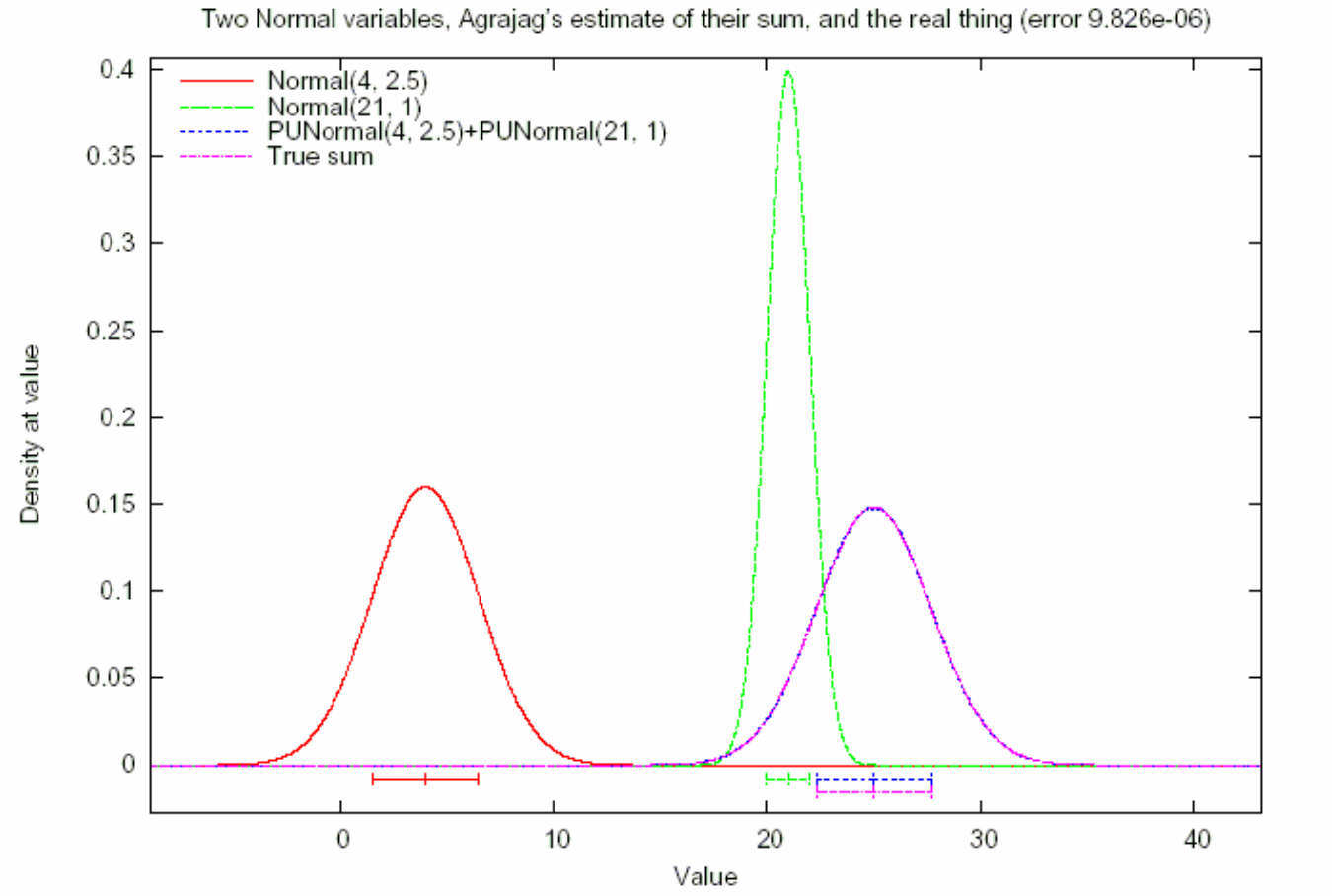
For example, for the property run time in sequential services, we can use $m_0 = m_1 + m_2$ and $s_0 = \sqrt{s_1^2 + s_2^2}$ to do the approximation -- the relational equations are proved true in mathematics[7].

Our experiments are based on a system called Agrajag. Agrajag is a framework written in Perl and C, developed by Conrad

[7] http://mathworld.wolfram.com/NormalSumDistribution.html

Hughes, to implement some operations and measurements on some basic models of stochastic distributions. Using Agrajag, we got a satisfying match (the error is generated by the limited calculation in the approximation method in Agrajag) of the piecewise uniform approximation curve (blue curve) and our estimate curve (mauve curve) (see Figure 2).

FIGURE 2
THE SUM OF TWO BELL CURVES AND ITS APPROXIMATION



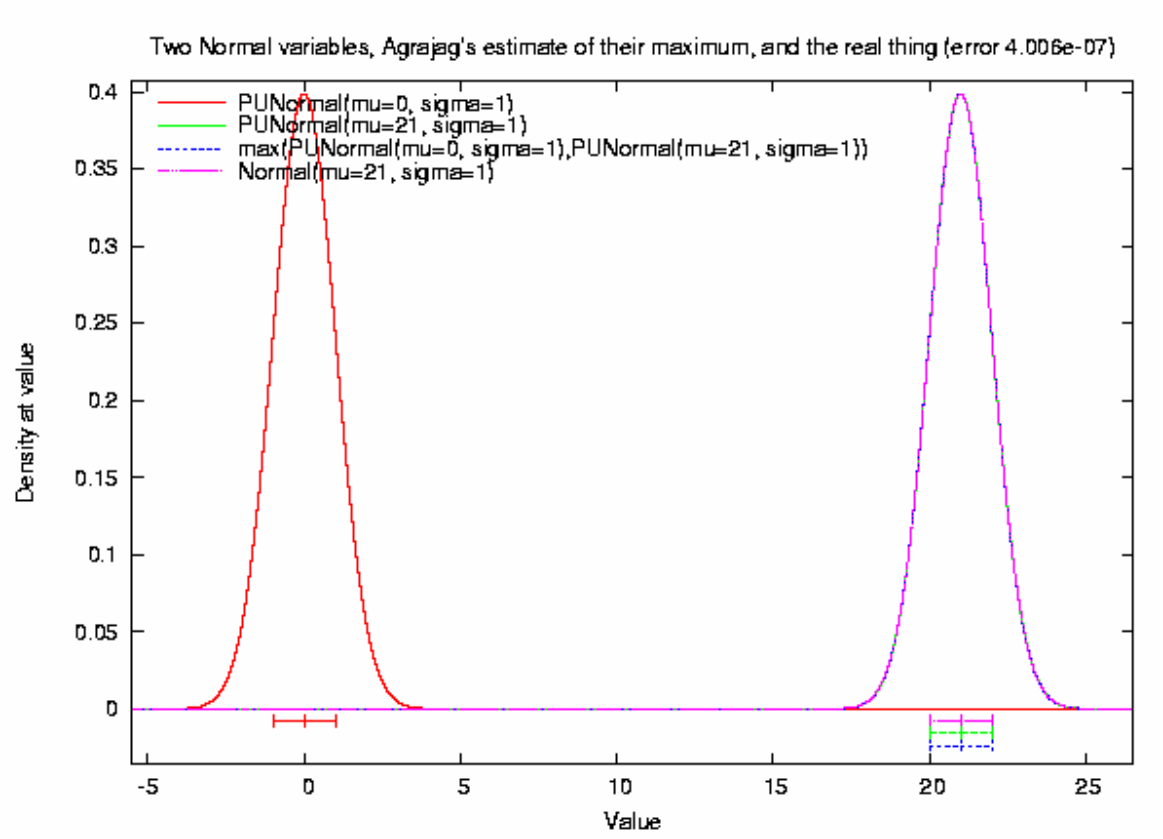
The graph shows the sum of two bell curves (red curve and green curve). It can be used to model the run time of sequential combinations. Here we use a mathematically proved method: $m_0 = m_1 + m_2$ and $s_0 = \sqrt{s_1^2 + s_2^2}$ to estimate the piecewise uniform curve (blue curve) produced by Agrajag. The mauve curve is our approximation curve, which almost coincides with the blue curve. We can see there is still a tiny error shown at the title of the graph. It is caused by the approximation using piecewise uniform functions. In the ideal situation (the resolution values which divide a curve to locally constant and connected segments $\to +\infty$), the error is zero.

Some of our combination functions have been defined by ourselves and tested in Agrajag. For example, for runtime in parallel_all structure, we need to get the maximum of two bell curves. Figure 3 shows the situation of the maximum of two bell curves using the combination method: $m_0 = \max(m_1, m_2)$ and $s_0 = \max(s_1, s_2)$. In this graph, we can see that our estimate achieved a good result – the error is very small. But does it always work like this? When we choose two closer bell curves as the inputs, the error became comparatively large (see Figure 4). This inconsistency decided one aspect worth investigation: through systematic experimentation using Agrajag, we need to explore in a wide range of data to find various error status in different input situations.

Another investigation aspect is to define and compare all sorts of combination methods to get, say, the maximum of bell curves. For example, through testing in Agrajag, we discovered that in most common situations, to get the maximum of two bell curves, the effect of approximating the output curve using $s_0 = \max(s_1, s_2)$ is better than that using $s_0 = \sqrt{s_1^2 + s_2^2}$ or that using $s_0 = 1/(1/s_1 + 1/s_2)$. Our main goals are to make comparisons of all sorts of approximation methods and find the best one based

on different situations.

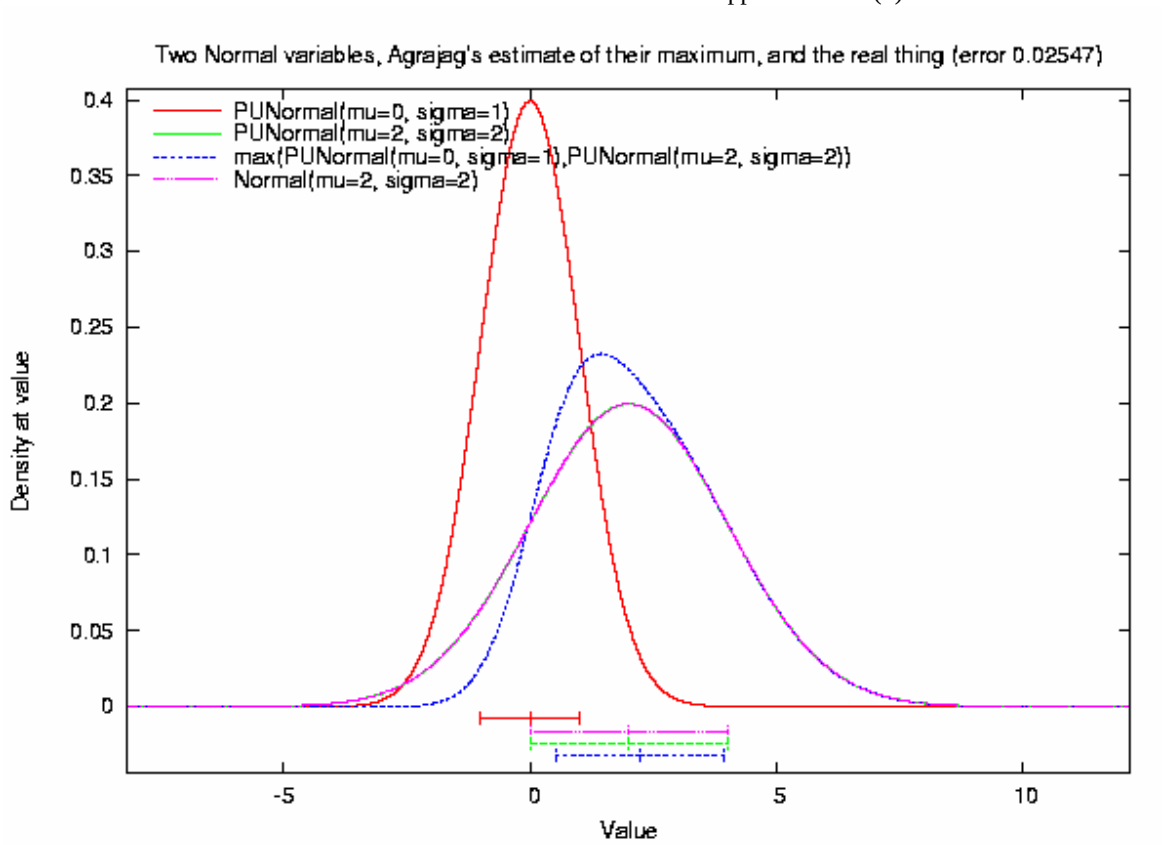
FIGURE 3
The Maximum of Two Bell Curves and Its Approximation (1)



This graph shows an ideal situation of getting the maximum of two bell curves. The red curve and the green curve are the two inputs. In this case, using the method $m_0 = \max(m_1, m_2)$ and $s_0 = \max(s_1, s_2)$, the green curve is the piecewise uniform form of our approximation, the mauve curve. The blue curve is Agrajag estimate. Since the green curve, the blue curve and the mauve curve almost coincide with each other, they are hardly distinguished in the figure.

This raises the question: since Agrajag can perform piecewise uniform approximation of bell-curve combinations, why do we still need a bell-curve calculus? Why don't we just use Agrajag to produce a bell-curve approximation to a workflow using the data from its component services? The answer is efficiency. Agrajag's calculations do well in small workflow calculations, but the more common scenario is that workflows sometimes are composed of thousands of services. To take all the inputs and get an approximation requires huge calculation capacity, which will reduce the speed of Agrajag heavily. While using bell-curve calculus, we just need to do calculations among the parameters, which will make the calculation procedure more efficient.

FIGURE 4
The Maximum of Two Bell Curves and Its Approximation (2)



In this graph, we use the same combination method as that in Figure 3, but taking two much closer bell curves as the inputs. This time, there is a distinct difference between Agrajag's estimate (the blue curve) and our approximation (the mauve curve). We can see that in this case, the mauve curve and the green curve are the same curve, so they coincide with each other.

4 Current Outcomes And Analysis

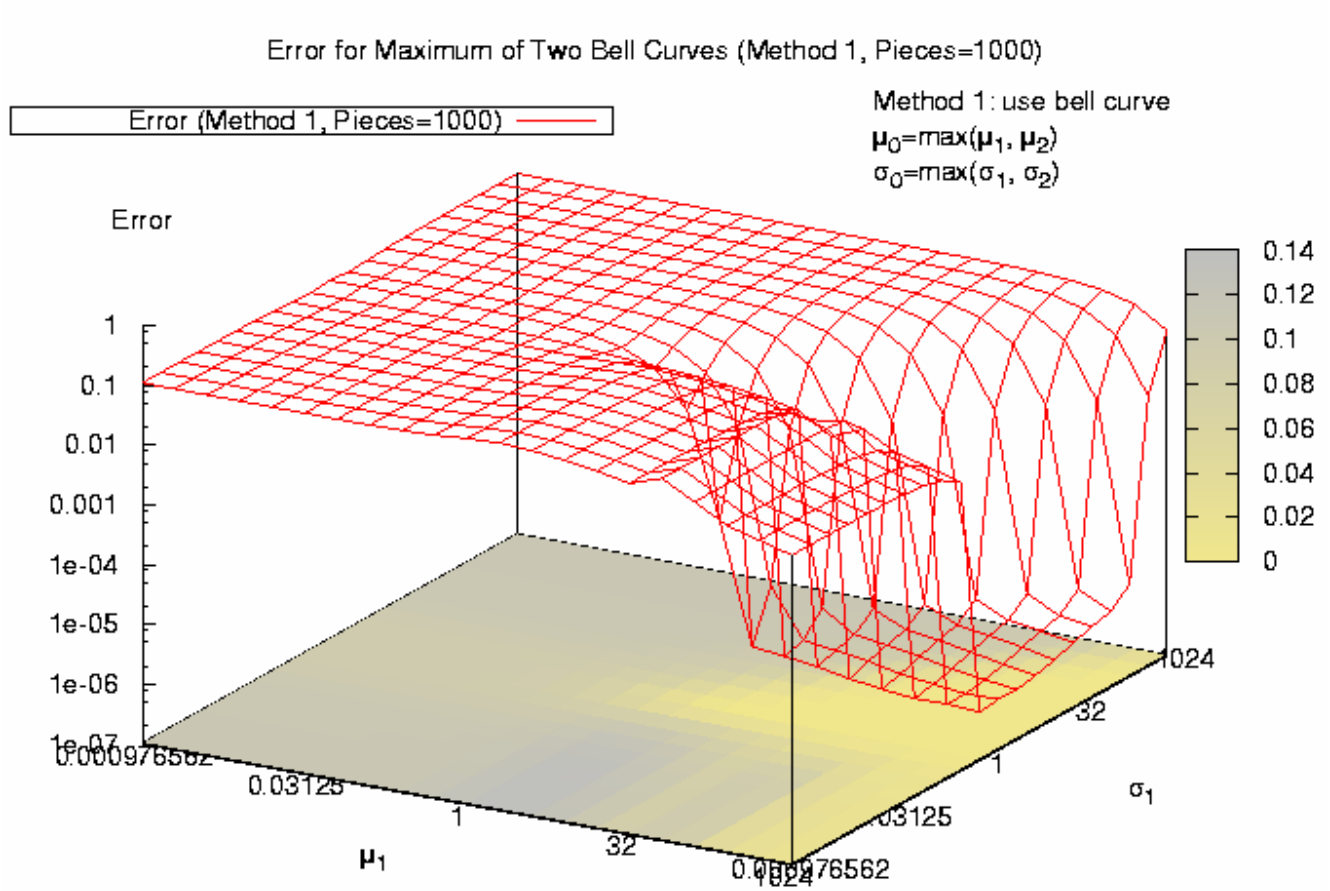
In this section, we will give some experimental results and analysis according to the methodology we have described in Section 2 and Section 3. Since the combination function of sum of two bell curves is exact, we make our first attempt on the method of the getting maximum of two bell curves, which does not have a known simple mathematical combination.

To get more intuitive results, we use Gnuplot[8] to draw 3D graphs. Without loss of generality, we fix one of the input bell curve to the standard bell curve (m_2=0 and s_2=1). Then the three dimensions are set as m_1, s_1 and the deviation between the piecewise uniform estimation and our approximation using our combination methods. To ensure that common situations are considered, we generate $bc(m_1, s_1)$ from a range of logarithmic-scaled integers, e.g., $2^{-10} \le m_1 \le 2^{10}$ and $2^{-10} \le s_1 \le 2^{10}$.

Figure 5 shows the experimental results using a combination method (Method 1): $m_0 = \max(m_1, m_2)$ and $s_0 = \max(s_1, s_2)$. From this graph we can see how the value of error changes. Especially in the area of $7 \le m_1 \le 9$ and $1 \le s_1 \le 1.6$, the errors are near 1e-06, which is a quite satisfying number.

[8] Gnuplot is a portable command-line driven interactive data and function plotting utility for many operating systems. It can plot either 2D or 3D graphs.

FIGURE 5
THE MAX OF TWO BELL CURVES AND ITS APPROXIMATION – METHOD 1



The graph shows the error distribution for $2^{-10} \le m_1 \le 2^{10}$ and $2^{-10} \le s_1 \le 2^{10}$. The X-axis is the value of m_1, the Y-axis is the value of s_1 and the Z-axis is the value of deviation between the piecewise uniform estimate (the resolution values is 1000 for this case) and bell-curve calculus estimate using the method $m_0 = \max(m_1, m_2)$ and $s_0 = \max(s_1, s_2)$. A lower value for the Z-axis shows a better fit. The values of x, y and z are discrete, but we set it drawn with the parameter 'by steps', which allows Gnuplot to connect every two points and give us a clearer figure. The colourful platform map at the bottom of the coordinates indicates the various values of error. From the label on the right, we can see that from grey to yellow, the value of error decreases.

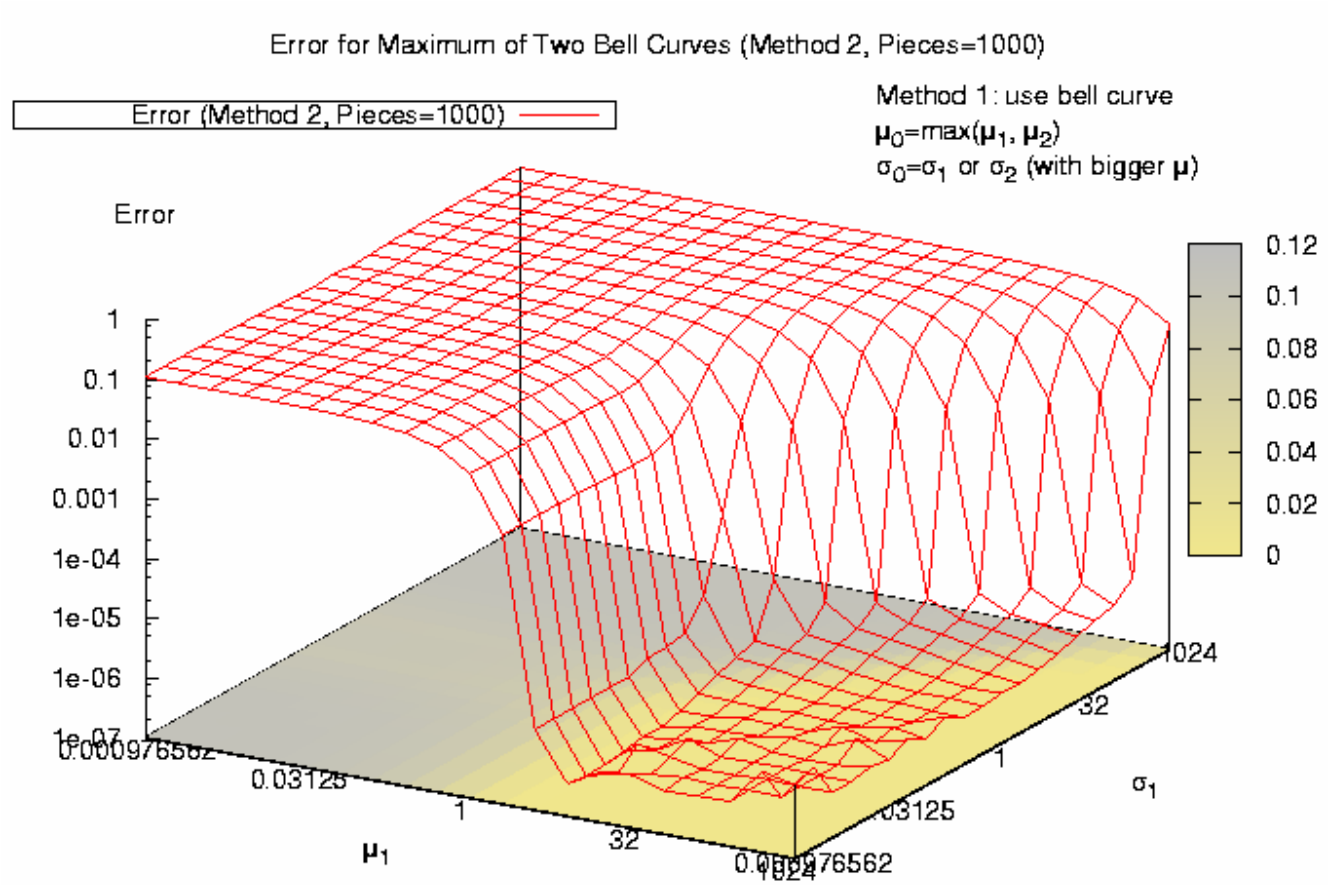
Does the method shown in Figure 5 achieve the best result? We tested on another method (Method 2): $m_0 = \max(m_1, m_2)$ and $s_0 = s_1$ or s_2 (with bigger m) (see Figure 6). In Figure 6, we can see that the area of tiny errors is extended, compared to Figure 5. To make a clearer comparison, we put the two graphs produced separately by Method 1 and Method 2 together in Figure 7. In most areas, the two surfaces coincide with each other. It is always true when $s_1 \ge 1$ because $s_2 \equiv 1$ and the both methods will take $s_0 = s_1$. Whereas in the area $m_1 \ge 4$ and $s_1 < 1$, the green surface (Method 2) is much lower than the red one (Method 1). But the two methods are still the best two among all the methods we tried. Table 2 shows all the combination methods we had tried to get the maximum of two bell curves and their average errors. For all the methods we used, $m_0 = \max(m_1, m_2)$.

TABLE 2
THE COMBINATION METHODS OF GETTING MAXIMUM OF TWO BELL CURVES

Method	Average error	Method	Average error
$s_0 = \max(s_1, s_2)$	0.0563955	$s_0 = s_1$ or s_2 (with bigger m)	0.0550807
$s_0 = \sqrt{s_1^2 + s_2^2}$	0.0782595	$s_0 = 1/(1/s_1 + 1/s_2)$	0.0799015
$s_0 = s_1 + s_2$	0.0862074	$s_0 = 1/s_1 + 1/s_2$	0.1160483
$s_0 = 0.8 \times s_1 + 0.2 \times s_2$ (with bigger m)	0.0564251	$s_0 = \sqrt{(s_1^2 + s_2^2)/2}$	0.0676253
$s_0 = s_1 \times s_2$	0.0550807	$s_0 = \sqrt{s_1 \times s_2}$	0.0648305
$s_0 = (s_1 + s_2)/2$	0.0650388	$s_0 = \lvert s_1 - s_2 \rvert \times s_1 / s_2 + \lvert s_1 - s_2 \rvert \times s_2 / s_1$	0.1194134

The table shows the situation when we use different combination methods to get the maximum of two bell curves at the same resolution value. We set $m_0 = \max(m_1, m_2)$ in all the methods. m_1 and s_1 both take values from 2^{-5} to 2^5. Since we use the standard bell curve as one input and $m_1 > 0$, $s_0 = s_1$ or s_2 (with bigger m) and $s_0 = s_1 \times s_2$ are the same method in this case. Despite this, the first two combination methods are the best two methods we got. The combination methods we choose are rough hypotheses based on Figure 4. We estimate the output parameters according to the location and shape of the Agrajag approximation curve. We calculated the average error of each method to compare how good these combination methods are.

FIGURE 6
THE MAX OF TWO BELL CURVES AND ITS APPROXIMATION – METHOD 2

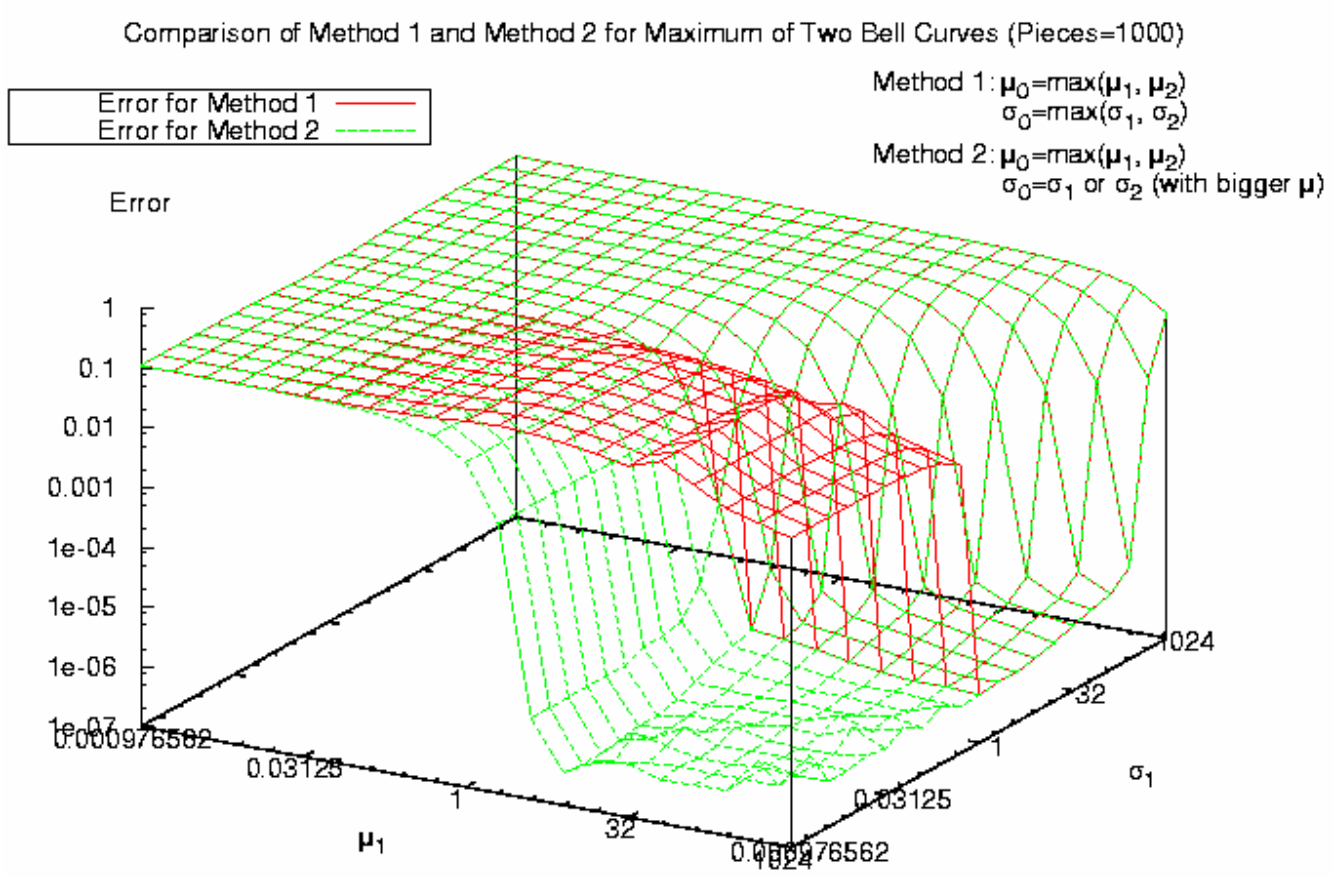


The graph shows the error distribution using the method $m_0 = \max(m_1, m_2)$ and $s_0 = s_1$ or s_2 (with bigger m). Please note that the yellow areas in the platform map do not imply that all the values of the error are zero, but rather the errors have too small difference on values to distinguish.

When we observe the above three figures, we can see that the errors produced by both methods stay stable at a comparatively high value in some areas. For instance, in Figure 6, there are some areas with correspondingly high error values and a sharp descend on error values at $m_1 \approx 4$. Why is there a distinct difference among the values? We did an experiment using method 2 to get the answer.

We set the numbers of pieces of piecewise uniform functions as 10, 100 and 1000 and got the three piecewise uniform estimates of the maximum of two bell curves. Then we used method 2 to derive our approximation of the maximum and obtained three error distributions. We drew the three distributions in one graph (Figure 8). We can see that the high-error areas of the three distributions coincided with each other. But since the three distributions used different resolution values, which means that the precisions of the three calculations are different and there should be a minimum difference on the error values with different numbers of pieces. While in some areas, the value is almost unchanged, which means the method we used did not get a correct result in these areas. We tested all the methods we used in our experiment and could not find a satisfying method.

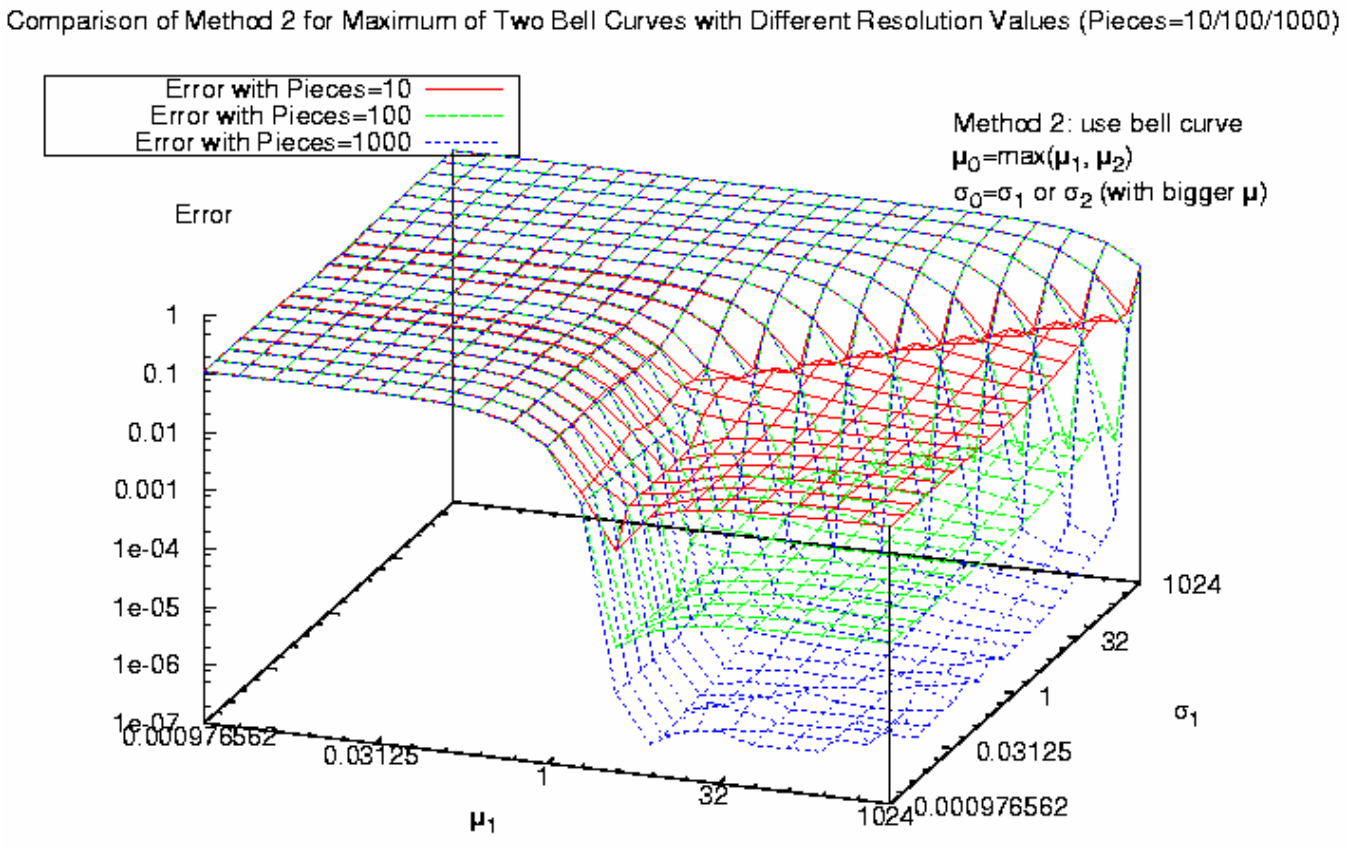
FIGURE 7
THE COMPARISON OF METHOD 1 AND 2 FOR MAX OF TWO BELL CURVES



The graph made a comparison between two methods to approximate the max of two bell curves. Method 1 is $m_0 = \max(m_1, m_2)$ and $s_0 = \max(s_1, s_2)$; while method 2 is $m_0 = \max(m_1, m_2)$ and $s_0 = s_1$ or s_2 (with bigger m). In certain areas, method 2 is much better.

Besides getting the maximum of two bell curves, we also tried to define the combination method to approximate the minimum and the multiplication of two bell curves. The approximation of minimum has the similar graph as the maximum ones. Figure 9 shows the situation of getting the multiplication of two bell curves using the method: $m_0 = m_1 * m_2$ and $s_0 = s_1 + s_2$. In some areas the error value is satisfactorily small, while in other areas, a high flat platform appears again, indicating the method is wrong in those areas. Furthermore, we use linear combinations to compound different combination methods to get maximum of two bell curves. Some achieved improved results, but in some areas, we still cannot get a satisfying result.

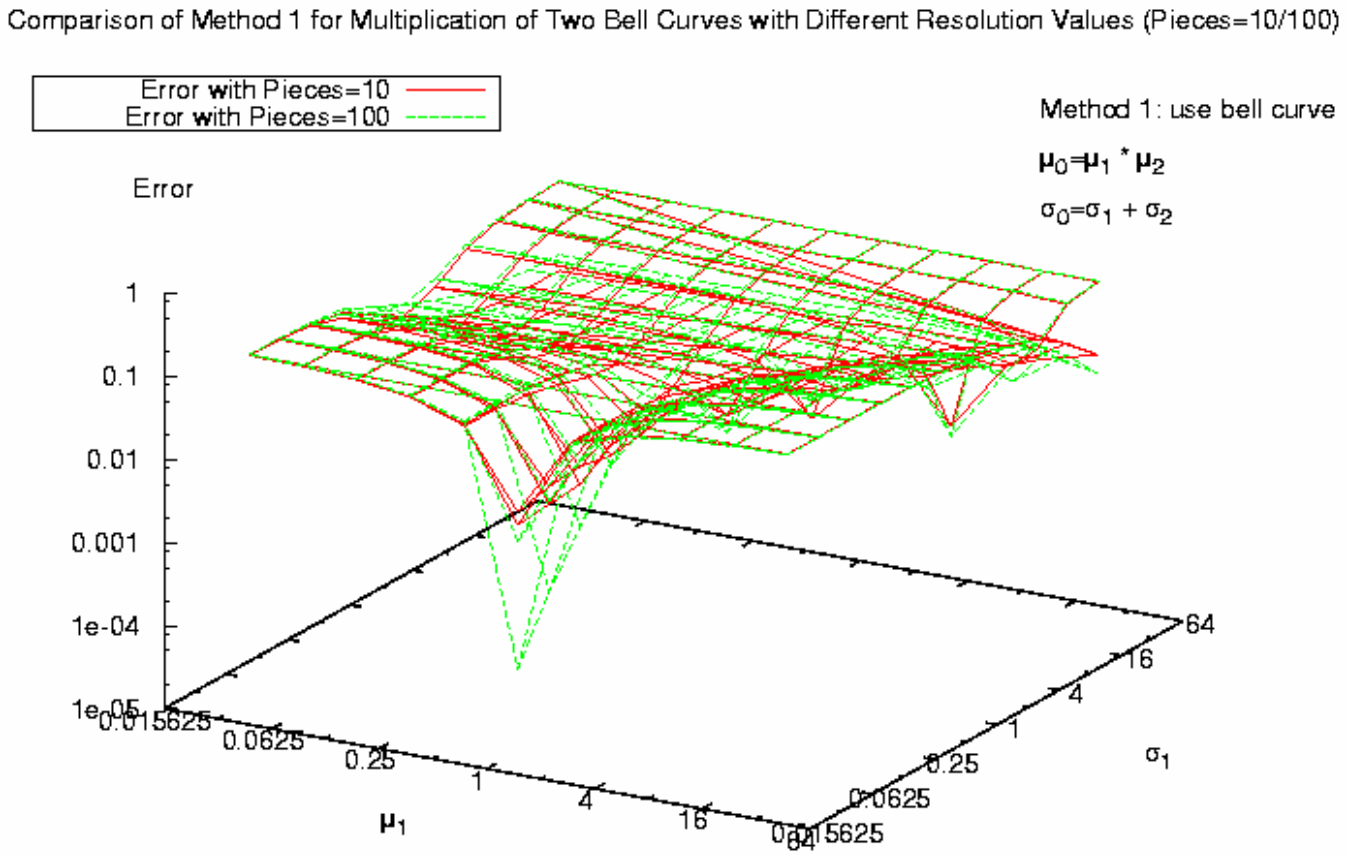
FIGURE 8
THE COMPARISON OF COMBINATION METHOD 2 WITH DIFFERENT RESOLUTION VALUES



The graph provides us the comparison of error distributions when we separately take the number of a pieces in piecewise uniform estimate as 10, 100 and 1000. Here the method is $m_0 = \max(m_1, m_2)$ and $s_0 = s_1$ or s_2 (with bigger m).

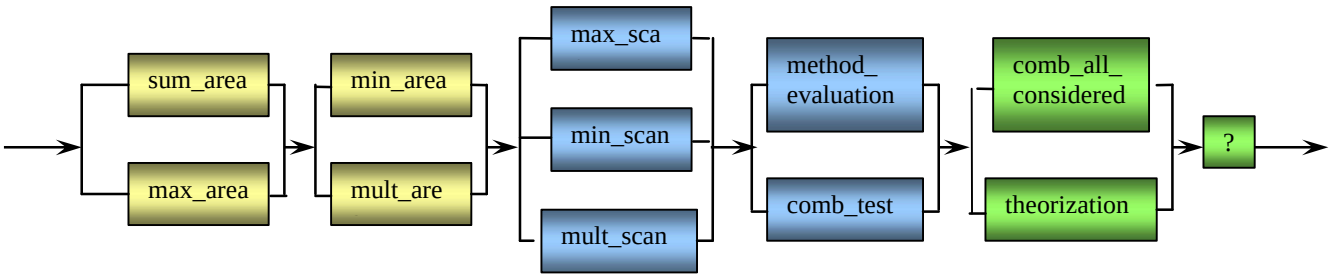
Figure 10 shows a short-term schedule of doing experiments using our bell-curve calculus. There are four element combination methods – sum, max, min and mult, which we are investigating using Agrajag. Comb is considered on the basis of the results of the four elements methods. During the procedure, we will develop some ways to evaluate our results. We will also summarise all the results we get and draw a conclusion on the feasibility of the bell-curve calculus. If it is proved a suitable way to describe and evaluate QoS properties, further steps, such as the design of an updated bell-curve calculus and seeking suitable infrastructure or framework which can support our bell-curve calculus, may be explored.

FIGURE 9
THE COMPARISON OF COMBINATION METHOD FOR GETTING MINIMUM WITH DIFFERENT NUMBERS OF PIECES



The graph gives the comparison of using the same combination method ($m_0 = m_1 * m_2$ and $s_0 = s_1 + s_2$) to approximate the multiplication of two bell curves with different resolution values (pieces are 10 and 100) in the estimates of piecewise uniform functions. In this figure, we can still find the problem similar to that in Figure 8. That is, in some areas, the error values stay high and unchanged when we changed the resolution value.

FIGURE 10
THE SHORT-TERM SCHEDULE OF BELL-CURVE CALCULUS EXPERIMENTS



In this figure, the four khaki modules represent finished work; the blue modules represent ongoing work; and the green modules represent future work. Sum, max, min and mult separately expressed the four fundamental combination methods -- summation, maximum, minimum and multiplication of bell curves. Comb is another fundamental combination method – combination of bell curves, which can be any combination form of sum, max, min and mult. Area shows that we apply different kinds of method (for instance, Method 1~Method 12 for maximum of bell curves defined in earlier paragraphs in this section) to a large area, say $2^{-10} \le m_1 \le 2^{10}$ and $2^{-10} \le s_1 \le 2^{10}$, to see if the method is satisfying everywhere. Scan means that we do not define any method, but use a set of continuous numbers as the approximated parameters m_0 and s_0 to see if there is suitable parameter set. In the stage of method_evaluation, we use some ways, for example, calculating average error, to do evaluation on all the methods we are investigating. We will consider all the possible situations of combination method 'comb' after doing some tests on it in previous

phase. Then we will theorize the results we get from the pre-steps. Any next step will be decided after the theorization.

5 Future Work

In Section 3 and 4, we described the methodology and some experimental results of our bell-curve calculus. The results tentatively proved that bell curve is a possible way to estimate the combination of QoS properties. However, as we indicated at the end of Section 4, we still cannot find a suitable method to approximate maximum or minimum of bell curves in some areas. There are three choices:

1. Continue investigating the combination methods using bell curve, in aid of some theoretical analysis, such as to find out relations of m s and s s by drawing graphs of satisfying values against certain form of m and s .
2. Define some other calculus to do the approximation. For instance, in the unsatisfying areas, we may consider using log-normal calculus, which uses log-normal distribution functions to describe the combination of QoS properties. If it works, we may also combine the two kinds of calculus if the bell-curve one does better than log-normal one in some areas.
3. Carry a third parameter which estimates how good our approximation is.

We will try the three ways in parallel. Also as stated in Figure 10, we will do some case study to see if there are more combination functions beyond the 12 fundamental ones introduced in Section 2. After defining all the necessary basic combination methods, we will apply them to deal with complex workflows and do some complexity analysis as well. We will also decide if the bell-curve calculus is a possible way to use in the representation and evaluation of QoS properties, especially the properties we are investigating. More steps will be designed if the bell-curve calculus is proved feasible.

6 Conclusion

In the above five sections, we have given background information including the definition of the Grid, web services, grid services and interval arithmetic. We have also discussed the importance and necessity of developing bell-curve calculus to deal with QoS properties. We have defined QoS properties and their fundamental structure concerned in our research. The methodology we are using to implement the bell-curve calculus has been described. Moreover, current experimental results and problems have been presented and the possible solutions have been provided as well.

In our research, we will try to prove our hypothesis:

The bell-curve calculus is a good estimate to Quality of Service properties in a wide range.

By now, according to the experimental results we have got, the bell-curve calculus is a suitable method to do approximation to the QoS properties we are studying in certain areas. In the next steps, we will focus on solving the existed problems and continue accomplishing our bell-curve calculus.

REFERENCES

[1] Foster, I., Kesselman, C., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, 1999

[2] Smarr, L., Chapter 1. Grids in Context, in The Grid 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003