A New Technique for Monte Carlo Generation of Monotonic Functions

A. C. Cem Say

Department of Computer Engineering
Boğaziçi University
Bebek 80815, İstanbul, Turkey
say@boun.edu.tr

Abstract

We propose using a different representation in current techniques of random generation of monotonic functions for Monte Carlo simulation of incompletely specified differential equations. Our method extends the scope of the technique to cover cases in which no envelopes have been specified for the function under consideration. Furthermore, the new representation does not entail unjustified implicit assumptions about the shape of the function. The general problem of "fair coverage" of the space of monotonic functions between two points is examined.

Introduction

One interesting alternative to semiquantitative simulation of incompletely known differential equations (Kuipers 1994) is the use of the Monte Carlo technique (Kalos and Whitlock 1986). In this approach, a large number of ODE's matching the given QDE are randomly generated making use of the available semiquantitative information, each such ODE is numerically integrated, and the family of results is treated as a representative sample of the infinite family of behaviors that would be entailed by the QDE. This technique has been used for verifying the stability of closed-loop nonlinear systems (Gazi *et al.* 1997) and assigning probabilities to alternative semiquantitative simulation predictions (Brajnik 1997). We propose using a different representation for the randomly generated monotonic functions matching the original incomplete specifications in the numerical integration phase of these techniques. Our method extends the scope of the technique to cover cases in which no envelopes have been specified for the function under consideration. Furthermore, our representation does not entail unjustified implicit assumptions about the shape of the function. The general problem of "fair coverage" of the space of monotonic functions between two points is examined.

Semiquantitative Reasoning with the Monte Carlo Approach

Semiquantitative simulation methods (Kuipers 1994, Kay 1996, Berleant and Kuipers 1998) take QDE's augmented with two kinds of quantitative information as input: A numerical interval can be specified for each landmark value, and a pair of quantitative envelope functions can be given for each monotonic (M^+ or M^-) relationship. The output is a list of qualitative behavior predictions in which numerical intervals have been associated with the landmark and time-point values. Semiquantitative simulation inherits the soundness guarantee and the incompleteness problem from qualitative simulation. Practical application of this method to complicated systems is problematic, usually because of the tendency of the underlying pure qualitative simulation algorithm to produce behavior trees which branch intractably.

Another method of solving such incompletely specified differential equations employs the Monte Carlo approach. This technique (Gazi *et al.* 1997) consists of the following steps:

```
Algorithm: MONTE-CARLO
1. Let N (the number of ODE's to be generated) be a
   sufficiently large integer (numbers like 10000 are
   common in the literature)
2. Do the following N times:
   2.1 Create an initial value problem from the input as
       follows:
       2.1.1 Create a vector of numerical values for the
             system variables at $t_0$ by randomly choosing
             a number for each of them from the
             corresponding user-specified interval
       2.1.2 For each monotonic functional relationship in
             the QDE, randomly generate a matching
             concrete monotonic function between the
             given quantitative envelopes
   2.2 Solve the initial value problem created in Step 2.1 by
       numerical integration; this is an iterative process
       whose cost and correctness depend on step size and
       number of steps
   2.3 Record the relevant results of the integration
```

Unlike qualitative simulation, the Monte Carlo approach has no soundness guarantee; its coverage depends on the computational budget one is willing to spend. If the number N is increased, the confidence that we would have about not missing a rare type of behavior would correspondingly increase.

One other important factor on which the usefulness of the Monte Carlo technique depends is whether the underlying space is covered "fairly" or not. For instance, if a variable's initial value is given to be in the interval [0,10], but more than 75% of our simulations start with a value less than 5 for this variable, there is something unfair going on. (We will be examining another sort of unfairness related to monotonic function generation later in this paper.) Fair coverage enhances the reliability of probability assignments for qualitative behaviors (Brajnik 1997) as well as minimizing the risk of missing a qualitatively distinct type of behavior.

Let us examine Step 2.1.2 in the algorithm above in greater detail. Gazi, Seider and Ungar (1997) approach the problem of randomly generating a monotonic increasing function $y=f(x)$ such that $f_l(x) \leq f(x) \leq f_u(x)$, where $f_l(x)$ and $f_u(x)$ are the given quantitative envelope functions, (Figure 1) as follows:

```
Algorithm: GENERATE-FUNCTION
1. Let G (the number of grid points to be selected in $x$) be a
   sufficiently large integer
2. Let $x_1$ and $x_G$ be the smallest and greatest possible $x$
   values to be considered, respectively. The interval
   $[x_1, x_G]$ will be divided by the grid points to G-1
   intervals of equal length
3. Choose $y_1$ randomly in $[f_l(x_1), f_u(x_1)]$ from some
   distribution
4. Choose $y_G$ randomly in $[\max(f_l(x_G), y_1), f_u(x_G)]$ from
   some distribution
5. Do the following for all available $(x_i, y_i)$, $(x_j, y_j)$ pairs
   with remaining untouched grid points between them:
   5.1 Select a midpoint $x_k$ in this interval
   5.2 Randomly choose $y_k$ in $[\max(f_l(x_k),y_i),\min(f_u(x_k),y_j)]$
6. Connect the points $(x_k, y_k)$, $k$=1,…,G, with straight lines
```

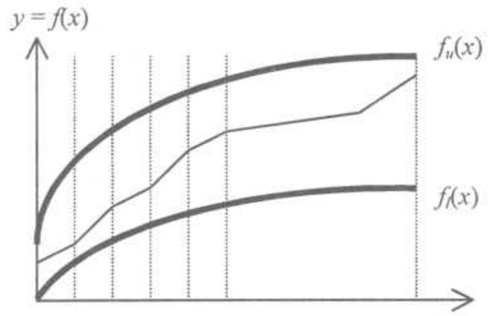

Figure 1: Generation of $f(x)$

(Decreasing functions are treated in an analogous manner.)

The probability distribution to be used in Step 5.2 of the above algorithm to choose a y_k for a given x_k is of crucial importance from the point of view of fair coverage. Gazi, Seider and Ungar (1997) provide a careful analysis of this problem and show that a uniform distribution is not always the best answer. When one already has an (x_{k-1}, y_{k-1}) pair, and the "next" point (x_k, y_k) is to be determined by choosing y_k, the shape of the envelopes delimiting these choices play a crucial role in whether the coverage is fair or not. If the y intervals to be used in these consecutive choices do not overlap, as seen in Figure 2, making the selection using a uniform distribution is nonproblematic. On the other hand, if these intervals overlap as in Figure 3, a uniform distribution can be proven to cause the family of resulting functions to be unfairly "attracted" towards the upper envelope. Gazi *et al.* solve this problem by using a distribution formula which is biased to favor points nearer the lower envelope. In this formula, a parameter has to be selected by the user depending on the shape of the envelope and the number G. An essential feature of this approach is the assumption that the number of grid points (equaling the number of random choices for y values that will be made) is fixed at the beginning of the algorithm and is a factor in the determination of the probability formula used for the random choices. High values of G are good for the "quality" of the generated functions, but they cause the computation of the probability formula to be impractical.

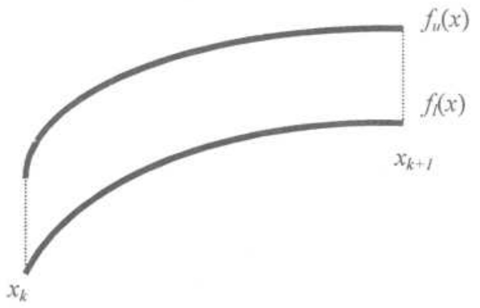

Figure 2: y intervals do not overlap.

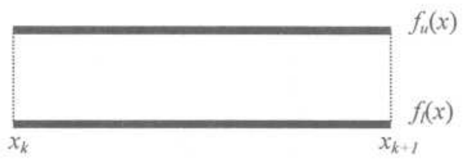

Figure 3: y intervals overlap completely.

Step 2.2 of the MONTE-CARLO algorithm performs numerical integration on the ODE prepared by the previous steps. There are several alternative methods (Press *et al.* 1988) that can be employed for this process. All of these accept the ODE as a set of first order differential equations for the phase variables v_i, each having the form

$$\frac{dv_i(t)}{dt} = F_i(v_1, v_2, \ldots, v_P),$$

where P is the order of the model. If we restrict ourselves to systems that can be handled by QSIM, the F_i are supposed to be defined only in terms of addition, multiplication, negation, and functions of continuous and strictly nonzero derivative (Kuipers 1994). It is these monotonic functions f among the QSIM variables (corresponding to QSIM's M^+ and M^- constraints) that we are interested in. At every time-step t_x of the numerical integration, the functions F_i are evaluated using the v_j values at that time point. During that computation, each monotonic function $f(x)$ appearing in F_i will have to be evaluated to yield $y=f(x(t_x))$. (Note that, if a relationship represented by a QSIM $M^+(x, y)$ constraint appears as a function f in the ODE, neither one of the QSIM variables x and y need to be phase variables of the ODE. For instance, the constraints $add(x, y, z)$, $M^+(z, u)$, $d/dt(w, u)$ translate to

$$\frac{dw}{dt} = f\left(x + y\right),$$

where only x, y, and w are the phase variables. We do know the numerical values of all QSIM variables, including the "non-phase" ones, at the start of the integration.)

Brajnik (1997) presents an interesting application of this approach to assign probability expectations to behavioral predictions outputted by semiquantitative simulation: The incomplete differential equation is first solved by a semi-quantitative simulator, and the list of predictions is obtained. The equation is then solved with the Monte Carlo approach of (Gazi *et al.* 1997). The numbers of numerical solutions which match the individual qualitative predictions are recorded, yielding a frequency distribution, which is an estimate of the probability expectation of each behavior. One bonus of this method is that it provides a possible way of showing that some qualitative predictions are not spurious: If even one numerical solution uniquely matches a prediction, we can trust that it is a genuine possibility.

Problems

We identify three problems with the method of monotonic function generation described in the previous section:

1. The method imposes an unjustified shape restriction (piecewise linearity) on all the generated functions. If the results of the Monte Carlo runs will be matched to QSIM behaviors as in (Brajnik 1997), this creates a theoretical problem because the QSIM monotonic functions are supposed to be differentiable throughout their domains, and so none of the ODE's considered by the Monte Carlo method actually correspond to the original QDE. Even if a differentiable spline is fitted to the generated points (as mentioned by Brajnik (1997),) having to choose y values corresponding to non-grid-point x values from the fitted curve leads to an unfair selection: Consider the function segment depicted in black in Figure 4. x_k and x_{k+1} are two consecutive grid points, and y_k and y_{k+1} are the corresponding function values, as selected by Step 5.2 of the GENERATE-FUNCTION algorithm. For simplicity, we assume that the envelope functions are separated widely enough in this interval so that they do not enter the picture. Assume that the numerical integration algorithm requires several y values for x's within this interval. Although a huge selection of points within this rectangle are consistent with monotonicity and our knowledge of the endpoints, we are forced to use only the points which are on the curve. Note that these intermediate y values would not be "random" in the sense that the y_k's are selected randomly in Step 5.2. Since we have no justification that the real function in question has such a form, and since (of course) only a finite amount of Monte Carlo runs and a finite resolution for the grid are allowed, this can be considered as a violation of the principle of fair coverage.

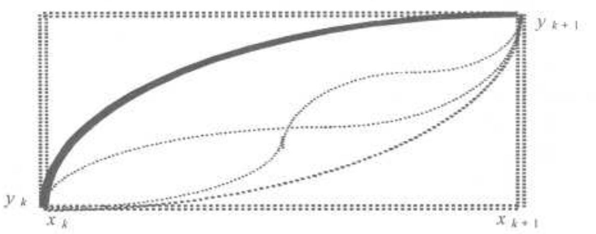

Figure 4: All y values corresponding to x's in the interval have to be on the dark curve. A few of the unconsidered curves are also shown.

2. The method may waste resources to generate parts of the function that will turn out to be unnecessary in the numerical integration phase. Suppose that the variable x has a legal range of [0, 1000]. The GENERATE-FUNCTION algorithm outputs a function over this entire domain. If a particular numerical integration from a particular set of initial values produces a behavior in which x does not get out of the interval [30, 40], most of the resources (which could have been used to provide a finer resolution within this interval) will have been wasted.
3. The method requires information about envelope functions to begin with. If we have no quantitative clue about the shape of a relationship except for the fact that it is monotonic, the algorithms described in the previous section would not work.

The Point-List Representation

Our proposed solution to the problems described in the previous section is based on generating the monotonic functions point by point as the need arises during integration, and representing them as lists of these points, without committing to any shape about the intervals in between. In our approach, the functions are not pregenerated to exclusively cover a fixed interval between the legal range limits of the variables. If a variable crosses its legal range limit during integration, it is the simulator's responsibility to detect the problem and stop the integration.

The next subsection explains the changes to the current algorithms necessitated by our approach.

Changes to the Algorithm

- Step 2.1.2 of the MONTE-CARLO algorithm will be modified as follows:

2.1.2 For each monotonic functional relationship $y=f(x)$ in the QDE, initialize the corresponding point-list to the single-element list $\{(x(t_0),y(t_0))\}$, using the initial values generated in Step 2.1.1, and set the variables *leftmost(f)* and *rightmost(f)* to the point value $(x(t_0),y(t_0))$.

- In the zeroth time-step of numerical integration, there is no need for evaluating any monotonic function, since all system variables have already been assigned initial values by Step 2.1.1 of MONTE-CARLO. So, for any invocation $f(x(t_0))$, where $f(x)=y$ is a monotonic function, simply plug in the available $y(t_0)$ value.
- In subsequent time-steps of numerical integration, do the following whenever $f(x)$ is supposed to be evaluated for $x=r$:

```
IF the point $(r,y)$ is a member of $f$'s point-list
  THEN RETURN $y$,
ELSE IF $r > rightmost(f)$ OR $r < leftmost(f)$
    THEN EXTRAPOLATE
    ELSE INTERPOLATE
```

- The routines EXTRAPOLATE and INTERPOLATE, as described below, will be incorporated to the algorithm.

We present the algorithm for calculating f for an x beyond the presently covered interval only for the case where the current x value r is greater than *rightmost(f)*, and f is increasing. The other cases are handled in an analogous manner.

```
Algorithm: EXTRAPOLATE
1. $(maxx,maxy) := rightmost(f)$
2. $\Delta x := r - maxx$
3. IF $f_l(r) > maxy$ AND $f_u(r) < \infty$
     THEN Pick $f(r)$ randomly from $(f_l(r), f_u(r))$ using the
            uniform distribution
     ELSE  IF $f_l(r) > maxy$
              THEN $\beta := \arctan((f_l(r) - maxy) / \Delta x)$
              ELSE $\beta := 0$
           IF $f_u(r) = \infty$
              THEN $\theta := \pi/2$
              ELSE $\theta := \arctan((f_u(r) - maxy) / \Delta x)$
           Pick the angle $\alpha$ randomly from $(\beta, \theta)$ using
              the uniform distribution
           $f(r) := maxy + \Delta x.\tan(\alpha)$
4. Append $(r, f(r))$ to the end of $f$'s point-list
5. $rightmost(f) := (r, f(r))$
6. RETURN $f(r)$
```

Figures 5, 6 and 7 depict different cases that may arise during the extrapolation. Figure 5 corresponds to the "no-overlap / two-envelopes" case where the y-coordinate of the current maximum point in the point-list is outside the finite y interval available for $f(r)$. In this case, we choose $f(r)$ uniformly from $(f_l(r),f_u(r))$, as recommended by Gazi *et al.* In the other cases, we use the following idea: A straight line between the last known point of f (that is, $(maxx,maxy)$) and the point that is to be generated now (that is, $(r, f(r))$) will have the slope $\tan(\alpha)$, where α is an angle that can take its values from the interval (β, θ). If we have an upper envelope at point r, the greatest possible slope will be caused by picking $f(r) = f_u(r)$. If no upper envelope exists, α can be arbitrarily close to $\pi/2$. The algorithm covers all these possibilities. See the next subsection for a further discussion of the effect of this distribution on fair coverage.

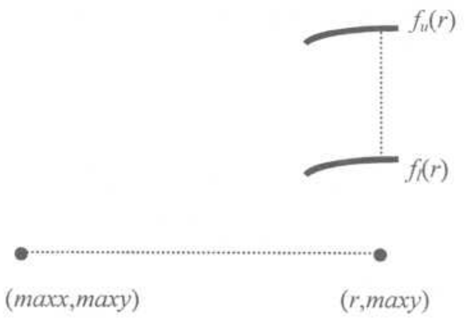

Figure 5: The no-overlap / two-envelopes case

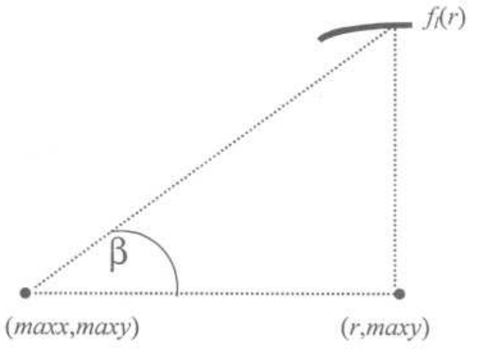

Figure 6: $\beta > 0$ if $f_l(r) > maxy$

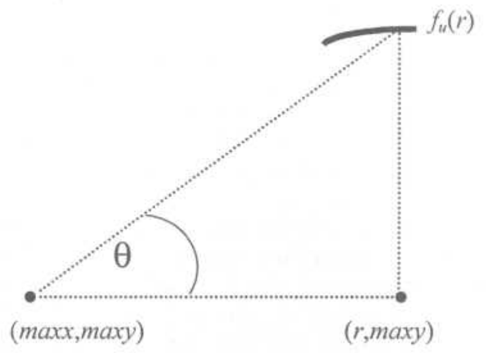

Figure 7: $\theta < \pi/2$ only if there is an upper envelope.

When a new function point is to be created between two points that are currently in the point-list, we adopt a modified version of the method of (Gazi *et al.* 1997):

```
Algorithm: INTERPOLATE
1. Find the two points in f's point-list between which the
   point $(r, f(r))$ that will be generated now will be in-
   serted; let these points be called $(lowx, lowy)$ and
   $(highx, highy)$, where $lowx < r < highx$
2. $k := \mathrm{round}((r - lowx) / (highx - lowx) * G)$
3. Use the formula in (Gazi et al. 1997) for $y_k$, plugging in
   $k$ to find $f(r)$
4. Insert $(r, f(r))$ in the place found in Step 1 in the point-
   list
5. RETURN $f(r)$
```

The G value used in Step 2 and in writing the probability distribution function for $f(r)$ in Step 3 is predetermined depending on the computational budget.

Discussion

Functions delimited by infinite (that is, nonexistent) envelopes are handled by our technique, which enables us to perform semiquantitative reasoning with a lower level of input information than allowed by the state-of-the-art techniques.

We claim that our technique provides a better coverage than that of Gazi *et al.* when performing interpolations between two already obtained points of the function, since we use the same distribution as Gazi *et al.*, but without any unnecessary commitment to function shape and with a "free" random selection for each necessary f value.

When performing "extrapolations," that is, when extending the covered domain of the function beyond a present maximum, there are three cases to examine:

- If there is no envelope pair delimiting the function, a performance comparison with other techniques is not possible, since they do not handle this case.
- If the y-coordinate of the present maximum is outside of the interval from which the new function value will be selected, we use a uniform distribution like Gazi *et al.*, so the coverage performance would be the same.
- In the remaining case, that is, when the new function value has to be selected from an interval between the present maximum y-value and the upper envelope, the nonexistence of a fixed grid prevents us from using Gazi *et al.*'s distribution. To try to avoid the envelope attraction effect, which could arise if we picked the function value from the available y-interval with a uniform distribution, we use a uniform distribution on the angle of the line between the two points with respect to the x-axis. Table 1 shows that the angle picking method produces points which are statistically further away from the upper envelope compared to the distance picking method, based on the situation in Figure 8, where Δx is 1.

A discontinuity is noticed when one examines the median of the generated $f(r)$ values in the no-overlap case (Figure 5) as the upper envelope is pushed to infinity. A huge finite value, say, H, of $f_u(r)$ causes a median which is $(H-f_l(r))/2$ units away from the lower envelope, whereas that distance drops to the much smaller value

$(r-maxx).\tan(((\pi/2)-\arctan((f_l(r)-maxy)/(r-maxx)))/2)-f_l(r)$

when $f_u(r)=\infty$. When the y-interval becomes infinite, the only remaining interval that can be sampled fairly is the angle interval. The angle interval is not used in the finite-envelope / no-overlap case, since it would cause an unfair attraction to the *lower* envelope.

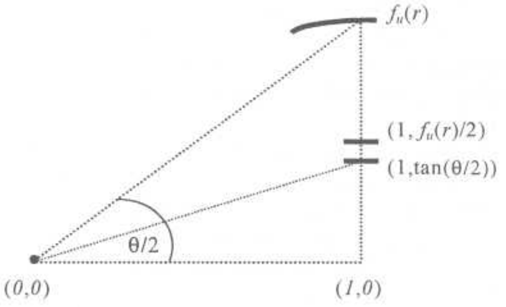

Figure 8: f values chosen uniformly from $(0, f_u(r))$ have a greater median than those produced by choosing the angle from $(0, \theta)$.

θ	median of uniform distance picking ($f_u(r)/2 = \tan(\theta)/2$)	median of uniform angle picking ($\tan(\theta/2)$)
0.1	0.05017	0.05004
0.2	0.10136	0.10034
0.3	0.15467	0.15114
0.4	0.21140	0.20271
0.5	0.27315	0.25534
0.6	0.34207	0.30934
0.7	0.42114	0.36503
0.8	0.51482	0.42279
0.9	0.63008	0.48306
1.0	0.77870	0.54630
1.1	0.98238	0.61311
1.2	1.28608	0.68414
1.3	1.80105	0.76020
1.4	2.89894	0.84229
1.5	7.05071	0.93160

TABLE 1. Median f-values produced by the two methods for different θ values in Figure 8

Compared with Gazi *et al.*'s method, our technique clearly spends more time and memory for each monotonic function; the requirements increase linearly with the number of numerical simulation steps when a new point has to be created for each step and no search needs to be conducted in the point-list to find the insertion place of the new point. In the example given in (Gazi *et al.* 1997), G (the number of generated points) is set to be only 11, whereas our method may result in as many points as the number of integration time-steps. When forced to generate an equal number of points as ours, their technique would waste some of this effort for x ranges that would not be visited during the simulation, and a "less random" family of curves would be used for the ranges that *are* visited, as explained above.

Distributions of Monotonic Functions in a Bounding Rectangle

In this section, we explore an alternative way of designing a probability distribution for $f(r)$, where the new point is to be between two already known points of the monotonic function. Figure 9 illustrates the idea: If we start in the lower left corner square, and are supposed to either move up or right one square at a time, how many different paths ending at the upper right corner are there? It is clear that, if we have i rows and j columns in our grid, the answer is

$$\binom{xdiff + ydiff}{xdiff},$$

where *xdiff* is the number of "right" moves that we are supposed to make, that is, j-1, and *ydiff* is i-1.

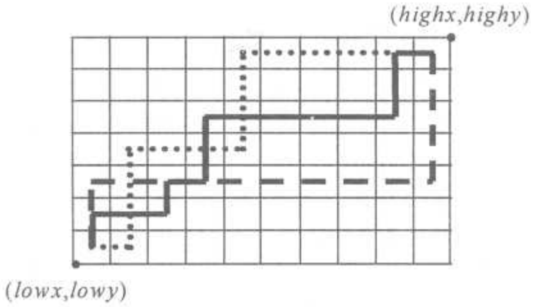

Figure 9: Some possible "monotonic" paths between the two corners

The probability distribution that we are looking for is supposed to tell us which $f(r)$ value is more likely for a given r, which is, in the context of Figure 9, almost the same question as asking which squares in the column corresponding to r have more paths passing through them. (See the end of this section for a discussion.) Using the combination formula given above, the number of paths passing through a square S is the product of the number of paths from the lower left square to S and the number of paths from S to the upper right square. Figure 10 shows a grid with i=3 and j=5 for which this calculation has been made for each square. It can be seen that the formula

$$\frac{\text{number of paths through the square at } (r, y)}{\text{number of all paths between the corners}}$$

is an approximation to the probability distribution we are looking for.

Now consider increasing the resolution of the grid to infinity, that is, taking the limit of the formula above as *xdiff* approaches infinity. (Note that we know the finite value that *xdiff* / *ydiff* is supposed to have, it is simply (*highx* - *lowx*) / (*highy* - *lowy*).) We can then use the obtained formula to calculate the probability that a given Δy interval contains $f(r)$ for given r.

1	3	6	10	15
5	8	9	8	5
15	10	6	3	1

Figure 10: "Path counts" of each square

We conclude this section by listing the problems about this approach: First of all, the simple combination formula for the number of paths that we used in our example cannot be used in the derivation of the distribution formula, since (allowing, as it does, paths with segments having a 90° angle with the horizontal,) it causes the same path to contribute to the path counts of more than one square in the same column. As the careful reader may have noticed, the sum of the numbers in a column of Figure 10 does not equal the number of paths between the corner squares, as given by the combination formula. An alternative (and less elegant) formula counting paths formed by line segments with angles in (0°, 90°) that connect the nodes of the grid, as shown in Figure 11, would have to be employed. Furthermore, when the envelope structure causes the shape of the area to be covered to be different than a rectangle, the formula gets even less elegant and harder to calculate.

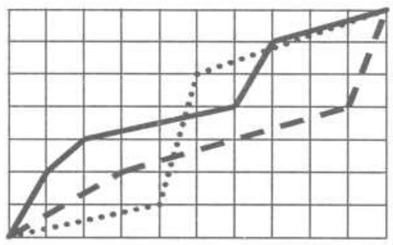

Figure 11: Some monotonic paths composed of non-perpendicular segments between the two corners

Conclusion

We presented a new technique for randomly generating monotonic functions required in the Monte Carlo simulation of incompletely specified differential equations. Unlike other techniques, our method works even when envelopes have not been specified for the function under consideration. Furthermore, the new representation does not entail unjustified implicit assumptions about the shape of the function.

Ortega *et al.* (1999) present a semiquantitative Monte Carlo reasoner in which the monotonicity constraint is not imposed on the underlying functions. Our point-by-point generation approach can be incorporated to their framework as well.

We are currently working on a full implementation of the techniques described here. Software products and other results of this work will be put on the WWW at the URL http://www.cmpe.boun.edu.tr/~say/files/montecarlo as they become available.

Acknowledgements

I thank Nadir Yücel, Can Özturan, Zafer Barutçuoğlu, Kutsi Nircan, Elton Ballhysa, Çiğdem Gündüz, Özlem Çetinoğlu, and Halim Levent Bağcıoğlu for their help in various stages of this work. I also thank the anonymous referees for their helpful comments. This research was partially supported by the Boğaziçi University Research Fund. (Grant no: 00A102)

References

Berleant, D., and Kuipers, B. 1998. Qualitative and Quantitative Simulation: Bridging the Gap. *Artificial Intelligence* 95: 215-255.

Brajnik, G. 1997. Statistical Properties of Qualitative Behaviors. In *Proc. Eleventh Int. Workshop on Qualitative Reasoning*, Cortona, Italy. 233-240.

Gazi, E., Seider, W. D., and Ungar, L. H. 1997. A Nonparametric Monte Carlo Technique for Controller Verification. *Automatica* 33: 901-906.

Kalos, M. H., and Whitlock, P. A.. 1986. *Monte Carlo Methods* Vol. 1. New York, NY: John Wiley & Sons.

Kay, H. 1996. Refining Imprecise Models and Their Behaviors. Ph. D. Dissertation, The University of Texas at Austin.

Kuipers, B. J. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. Cambridge, Mass.: The MIT Press.

Ortega, J. A., Gasca R. M., and Toro, M. 1999. A Semi-quantitative Methodology for Reasoning about Dynamic Systems. In *Proc. Thirteenth Int. Workshop on Qualitative Reasoning*, Loch Awe, Scotland. 169-177.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. 1988. *Numerical Recipes in C*. New York, NY: Cambridge University Press.