Statistical properties of qualitative behaviors

Giorgio Brajnik
Dip. di Matematica e Informatica
Università di Udine, Udine — Italy
giorgio@dimi.uniud.it

Abstract

This paper presents a Monte Carlo method for analyzing the space of trajectories entailed by a qualitative differential equation and deriving statistical properties of qualitative behaviors. Estimates of the occurrence probability of behaviors and estimates of parameters defined over numerical trajectories can be computed and used in several ways. For example, behavior probability can be used to rank behaviors and discard those below a significance threshold. Parameter estimation can be used to determine the average maximum response of a controller to a given perturbation. An example of a second–order oscillatory system is discussed.

Introduction

Formalisms, methods and tools developed by the Qualitative Reasoning community have the potential to represent and deal with levels of uncertainty that are unpaired by those developed within other communities. For example a Qualitative Differential Equation (QDE) and the QSIM simulator (Kuipers, 1994) can represent and deal not only with a specific (linear or non–linear) dynamical system, but can handle a family of systems characterized by uncertainty affecting both parameter values (*parametric uncertainty*) and functional relationships between two or more parameters (*functional uncertainty*).

While several methods (e.g. linearization, sensitivity analysis, stochastic differential equations) can be used when dealing with parametric uncertainty (since functional forms are known), very few methods exist that can deal with functional uncertainty. Not only qualitative simulation can be used in such cases to generate some prediction, but it predicts *all* the possible outcomes of *all* the systems belonging to the family denoted by the QDE. This constitutes a solid foundation for tackling robustness analysis problems, *i.e.* proving that certain properties — like stability — hold for an entire family of systems. These problems arise often and in many tasks: in diagnosis, to test a fault hypothesis; in monitoring, to predict a risky situation; in design, to determine the dynamics of a partially–specified device.

But a free lunch is a very rare opportunity, and this case is no exception. In fact, the uncertainty present in the problem may lead to results affected by ambiguity hindering their practical usefulness. This is due to qualitative simulation being very conservative, yielding extremely accurate results (in the sense that they are valid representations of reality). Often, however, such a level of accuracy is not required by practical applications (as definitely shown by the broad and effective application of approximate methods, that are precise but inaccurate). We could then trade some accuracy for some precision.

Stochastic differential equations and stochastic simulation methods (*e.g.* (Doyle and Sacks, 1989)) can be used to achieve similar results. However, they require as input a level of knowledge about the dynamical system that is higher than that required by a QDE. In fact, probability distributions for system variables have to be given in the former case, and transition probabilities between states in the latter.

Alternatively, as shown by Gazi and colleagues (Gazi *et al.*, 1996; Gazi *et al.*, 1997), Monte Carlo methods can be used to sample the trajectory space entailed by a QDE and derive conclusions for the whole family of systems without reducing the uncertainty level.

In this paper we describe a Monte Carlo method for analyzing such a space and deriving statistical properties of qualitative behaviors. In this way we get an estimate of the occurrence probability of qualitative behaviors and estimates of parameters of the numerical trajectory. This information can be used in several ways. For example, behavior probability can be used to rank behaviors and discard those having a very low probability. Parameter estimation can be used, for example, to determine the average maximum response of a controller to a given perturbation.

This work extends the previous work (Gazi *et al.*, 1996; Gazi *et al.*, 1997) centered on comparing numerical trajectories against temporal logic statements for verification purposes. We compare trajectories against qualitative behaviors, achieving the following three results. First, getting the a–priori proba-

bility of occurrence of a set of exhaustive behaviors sheds considerable light on the qualitative features of trajectories that are practically plausible. Second, qualitative behaviors usually contain information that is not available in numerical trajectories (*e.g.* quiescence, stability, equality between two maxima) that can be used to extract data of interest (*e.g.* average time to reach quiescence). Finally, concrete examples of systems exhibiting a qualitative behavior can be generated and examined in detail.

Outline of the method and its usage

Within a QDE parametric uncertainty is represented by symbolic landmarks denoting unknown real numbers, possibly bounded by numeric ranges; functional uncertainty by monotonic (or variations like S– or U–shaped) relationships, possibly bounded by a pair of functions.

Monte Carlo methods solve mathematical problems via simulation based on random numbers (Rubinstein, 1981). In our case the user defines a QDE, provides numerical envelopes bounding the monotonic relationships used in the QDE, and specifies some control parameters for the Monte Carlo process. Then random numbers are used to generate instances of monotonic relationships. These instances are inserted into the QDE which is numerically integrated yielding a trajectory that is compared to qualitative behaviors obtained from qualitative simulation of the QDE.

This comparison yields a boolean answer: the trajectory is an instance of a behavior, or it is not. By computing the proportion of instances that satisfy a behavior, standard statistical techniques can be used to estimate the *probability of occurrence* of a qualitative behavior (*i.e.* the probability that the trajectory of a random instance of the QDE matches the behavior). In addition, if the value of some parameter is sought among the instances of a behavior (like average settling time for an overdamped oscillatory system), then it can be determined by collecting its value from each instance of the behavior.

In general, this information can be used for a number of decision making processes.

Occurrence probability can be used for ranking the behaviors. In this way most probable ones, or those with the highest risk (cost times probability, where the cost is given by the user) can be given the highest priority. For example, in analyzing the outcomes of a lake–dam system, the behavior where the lake level crosses a watch–out threshold presumably has a high cost. Even with a very low probability, such a behavior may be worth of further analysis. In a design context the occurrence probability of each behavior can be used to perform a robustness analysis of the effects of some change in the model (the value of the controller gain, for example). The change will affect the probabilities of behaviors. This method could be used, for example, to determine a value for the parameter that maximizes the occurrence probability of a selected behavior.

Occurrence probability can also be embedded within hypothesis testing. For example, to find out which behaviors have an occurrence probability lower than a given threshold. These behaviors can then be regarded as either impossible (*i.e.* spurious) or practically impossible (like the behavior of the "classical" filling bathtub model where the level asymptotically reaches the top of the tub and settle there: highly unlikely for any height). In a monitoring context such an information can be used to neglect predictions that would lead to costly resource utilization. In a design task, to determine for example that the only behaviors violating some performance requirements are practically impossible.

Statistical estimations of parameters defined over numerical trajectories can be used in several ways. These values may distinguish a faulty from a normal situation, or may give important design insights. A simple thing is to look within the instances of a behavior to find out the numerical values corresponding to symbolic landmarks used in the QDE. For example, to determine the average time needed by a variable to reach a specific value.

Parameters are not limited to landmarks. In fact, any parameter that can be defined on a trajectory can be obtained. For example, one could determine the average frequency of a family of oscillatory systems; or the average of the maximum response of a controller to a perturbation.

An example

We present now an example based on a second order oscillatory system: while easy to understand, it is representative of many interesting real–world systems and sufficiently complex to pinpoint many issues raised by the method.

Suppose a designer wants to analyze the effect that the damping factor has on the global dynamics of a very poorly known system. In particular the designer is interested in determining the likelihood for the partially known system to reach the equilibrium point after a given number of cycles and in knowing what is the period and settling time for such a system (*i.e.* the time needed by x to enter the region $\pm 1.0^{-6}$).

The model is $\ddot{x} = F(x) + G(\dot{x})$ where $F(x)$ denotes the "force" exerted towards the equilibrium point when the "position" is x, whereas $G(\dot{x})$ is the "friction" force due to "velocity" $\dot{x}$. F and G denote two families of functions $\{f\}$ and $\{g\}$ that are monotonically decreasing, passing through the origin, and bounded by $\underline{f}(x) \leq f(x) \leq \overline{f}(x)$ and $\underline{g}(x) \leq g(x) \leq \overline{g}(x)$. In this specific case bounds are defined using pairs of linear functions specifying an error band around the (nominal) value of the parameter. That is

$F(x) = -a(1 \pm e)x \pm d$, $G(\dot{x}) = -b(1 \pm e)\dot{x} \pm d$, where $e = 25\%$, $a = 10.0$, $b = 2.0$ and $d = 5.0$. Notice that F and G include non-linear functions in a rather large uncertainty band ($50\% + 10$ for both functions).

The QDE can be defined straightforwardly from such specifications. A qualitative simulation from an initial state where $x \in [0.9, 1.1], \dot{x} = 0$ (with a limit on the number of states being examined) leads to 6 qualitative behaviors (figure 1): in all except the last one the system follows a spiraling trajectory of zero, half or more cycles and then approaches the equilibrium point.

Such a prediction cannot be used to infer that behaviors 1 to 5 are highly unlikely nor to get sufficiently precise values for the sought parameters. In fact, when facing a significant uncertainty semi-quantitative simulation techniques are unable to discriminate between alternative qualitative behaviors. While each qualitative behavior is possible for some (unknown) instance of the QDE, the numerical information added to the QDE induces a probability distribution among these behaviors. Our method statistically estimates such a distribution.

The Monte Carlo simulation method, where 100 random samples $\widehat{f}$ and $\widehat{g}$ from F and G are generated and simulated with random initial states in $[0.9, 1.1] \times [0, 0]$, leads to 100 trajectories all matching only behavior 6, where equilibrium is not reached before 3 cycles. In addition, the average oscillation period is 0.450 ± 0.018 (at 95% confidence level) which is quite precise compared to the value computed on the basis of ranges predicted by QSIM.

Results change when $b = 5.0$. In this case the probability of occurrence of each behavior is given in table 1. It can be seen that the chances of approaching equilibrium before 3 cycles is getting higher. In addition, the average settling time for behaviors 2 to 5 is 5.667 ± 0.114, while the period for behavior 6 becomes 5.726 ± 0.106.

Behavior	Probability
1	$0.070 \pm 0.050 = [0.020, 0.120]$
2	$0.160 \pm 0.072 = [0.088, 0.232]$
3	$0.230 \pm 0.082 = [0.148, 0.312]$
4	$0.260 \pm 0.086 = [0.174, 0.346]$
5	$0.270 \pm 0.087 = [0.183, 0.357]$
6	$0.280 \pm 0.088 = [0.192, 0.368]$

This table gives an estimate of the proportion of 100 trajectories that match a given behavior. The 95% confidence intervals give the likely variation of such a proportion when different samples are explored.

Table 1: Occurrence probabilities when $b = 5.0$

Finally, setting $b = 5.5$ and running another experiment yields the table (2). Settling in behavior 1 occurs in an average of 5.52 ± 0.16, while for behaviors 2 to 5 in 6.87 ± 0.534. The average period for behavior 6 is 5.96 ± 0.292.

Furthermore for behaviors 5 and 6 the hypothesis "probability < 0.10" is supported by the data with a 95% significance level. If 0.10 is the threshold below which a behavior is deemed negligible, then both can be discarded with only 5% chances of error.

These results have been obtained from running 100 Monte Carlo trials which took around 3 minutes elapsed time[1].

Behavior	Probability
1	$0.210 \pm 0.080 = [0.130, 0.290]$
2	$0.260 \pm 0.086 = [0.174, 0.346]$
3	$0.280 \pm 0.088 = [0.192, 0.368]$
4	$0.200 \pm 0.078 = [0.122, 0.278]$
5	$0.050 \pm 0.043 = [0.007, 0.093]$
6	$0.050 \pm 0.043 = [0.007, 0.093]$

It is very likely that the system will not exhibit 2 or more cycles (*i.e.* behaviors 5 and 6).

Table 2: Occurrence probabilities when $b = 5.5$

The simulation method: main issues and adopted solutions

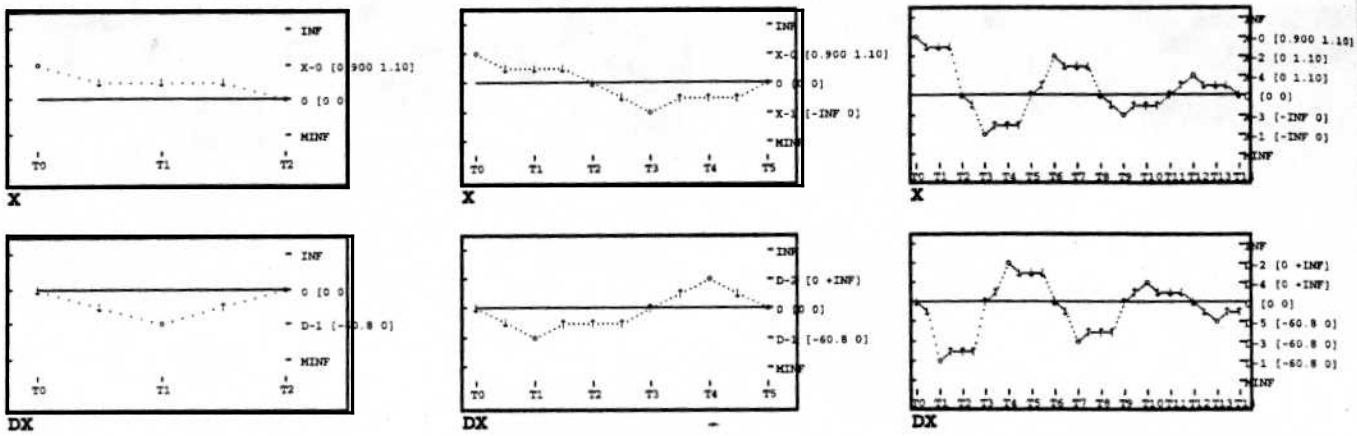
The Monte Carlo simulation method is based on generation of random instances and classification of their trajectories.

Generation of random QDE instances

The problem of generating random instances of QDE's has been discussed in detail by Gazi and colleagues (Gazi *et al.*, 1997). The basic idea is to generate a random ordinary differential equation by replacing each monotonic relationship of the QDE with a random instance of it. A random function satisfying the monotonicity and envelope constraints is constructed by generating a sequence of monotonic random points satisfying the envelope $(\underline{f}, \overline{f})$ and then interpolating these points (with straight lines or splines).

The problem of generating these points with a uniform probability distribution, however, is not as simple as it might seem. One of the simplest algorithms consists of generating k random points $x_1, x_2, ..., x_k$ within the domain of the function and then, for each x_i, selecting a random y_i from the interval $[\underline{f}(x_i), \overline{f}(x_i)]$. The problem with this algorithm is that when two successive intervals overlap (i.e. $[\underline{f}(x_{i-1}), \overline{f}(x_{i-1})]$ and $[\underline{f}(x_i), \overline{f}(x_i)]$ are not disjoint), then (assuming the function has to be increasing) the selection of y_i occurs within the top part

[1] On a Sun-20 running Lucid Lisp; the bulk of the time is due to numerical integration.



Six qualitative behaviors are predicted by QSIM for the oscillator; only behaviors 1, 2 and 6 are shown. Behavior 6 is not complete. DX denotes $\dot{x}$.

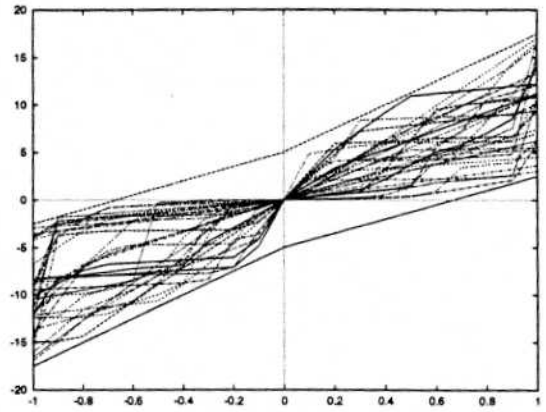
Figure 1: Qualitative behaviors for the oscillator

of $[\underline{f}(x_i), \overline{f}(x_i)]$, precisely in $[y_{i-1}, \overline{f}(x_i)]$. In other terms, the random selection performed at stage i is dependent on the value selected at stage $i-1$.

As shown in (Gazi *et al.*, 1997) the potential overlap of the envelope at two consecutive points has the effect that the resulting generated points (and the corresponding interpolated function) cluster towards $\overline{f}(\cdot)$. The authors suggest another algorithm that gives more uniformly distributed points. The algorithm we used is derived from that one. At the endpoints x_0 and x_{h+1} of the domain two values y_0 and y_{h+1} within the envelope are randomly generated; then the following steps are repeated h times: (i) random selection of a pair of consecutive points $(x_i, y_i), (x_{i+1}, y_{i+1})$, (ii) splitting of the interval (x_i, x_{i+1}) with a random point x^*, and (iii) random selection of a point y^* such that $y_i < y^* < y_{i+1}$ and $\underline{f}(x^*) \leq y^* \leq \overline{f}(x^*)$. Resulting piece–wise linear interpolated functions are quite uniformly distributed and have widely changing shape (see figure 2).

Depending on the curvature of the functions $(\underline{f}, \overline{f})$ an invalid interpolated function $\widehat{f}$ can be generated if $\widehat{f}$ crosses either bound. To overcome this, the test $\underline{f}(x) \leq \widehat{f}(x) \leq \overline{f}(x)$ is performed each time the function is evaluated. If it fails then the instance of the QDE is discarded. So far we found that the chances of failing such a test are sufficiently low to justify the use of such a run–time check. In the situation described by the example the problem cannot occur, since envelopes are linear.

At this point the differential equation can be numerically integrated from a randomly chosen initial state satisfying the qualitative specification (up to a predefined time limit and using a given integration time–step; our implementation uses a Runge–Kutta method), yielding a trajectory for each variable. The



Fifteen instances of the monotonic relationship $-G$ used in the previous example, restricted on the interval [-1, 1], passing through the origin and generated by interpolating ten randomly chosen points.

Figure 2: Sampled instances for function G(x)

trajectory can then be compared to qualitative behaviors.

Comparison of trajectories with behaviors

The aim of this process is to determine if a numerical trajectory (up to time limit t^*) is an instance of a qualitative behavior (up to qualitative time T^*). This is achieved by comparing the trajectory of each variable, separately, with the sequence of qualitative values prescribed for that variable in the behavior and then constructing a global interpretation of the behavior against the system trajectory.

It is worth noting the differences between the information present in the two descriptions of the dynamics. A trajectory of variable v is a sequence of

numerical events $<t_i, v_i>$, where t_i is a time point, v_i is the numerical value of $v(t_i)$, t_i's are increasing and $t_{i+1} - t_i$ is the integration time–step. Conversely, the behavior of a variable consists of: (i) a sequence of alternating time–point and (open) time–intervals *qualitative events* $<T_i, <qmag_i, qdir_i>>$, where T_i is either a time landmark or a pair of adjacent landmarks, $qmag_i$ is a landmark (or pair of landmarks) denoting the value $v(T_i)$ (or an open interval containing it), $qdir_i \in \{\texttt{dec}, \texttt{std}, \texttt{inc}\}$ denotes the sign of $\dot{v}(T_i)$; (ii) numeric ranges (possibly $[-\infty, +\infty]$) assigned to each landmark; and (iii) a labeling of T_i (saying whether the corresponding state is *quiescent*, *stable* or if it triggers a *transition* to a new operating region).

Time–point qualitative events may correspond to: (a) the variable reaching one of its landmarks, (b) its derivative changing sign, or (c) some other variable being involved in situations (a) or (b) (like the event at T1 for variable x in behavior 1 of figure 1).

The comparison process is based upon two steps: *event extraction* from each variable trajectory and a *behavior interpretation* against extracted events.

Event extraction

The aim is to extract from the trajectory a sequence of *qualitative events* that can be later compared to those included in the behavior. Event extraction should be *correct*, *i.e.* all relevant qualitative events corresponding to situations (a) and (b) discussed above must be generated as well as intermediate interval–time events. It should also detect *asymptotic trends* towards limit points (like when the oscillator position approaches its equilibrium value).

The extraction mechanism is based on the following principles.

Each numerical event $<t_i, v_i>$ is analyzed together with its neighbors and those of the derivative of v. If $\dot{v}$ has not been computed during numerical integration, it is estimated using the formula $\frac{1}{12h}[v_{i-2} - 8v_{i-1} + 8v_{i+1} - v_{i+2}]$ (h is the time step) whose error is $O(h^4)$ (Comincioli, 1992). Whenever $\dot{v}$ reaches or crosses 0 an event is generated, as well as when v reaches or crosses 0 or any other interesting landmark value given as input.

"Holes" between numerical events (*i.e.* $v_i - v_{i-1}$ being finite) and roundoff may hamper the detection of relevant events (that is, the event defined by $v = \alpha$ may be missed if no v_i is ever equal to α). For this reason the extraction mechanism detects not only if a variable *reaches* a given value, but also if it *crosses* the value (*i.e.* if $v_i - \alpha$ changes sign). Similarly when detecting changes in the *qdir* of a variable. When a variable v crosses a value α between t_{i-1} and t_i then the detected event abstracts the time instead of approximating it: the resulting event states that there exists $t \in [t_{i-1}, t_i]$ such that $v(t) = \alpha$, without committing to a precise value of t.

To cope with floating point roundoff errors, all comparisons between values are performed with a tolerance ε (for the example discussed in previous section $\varepsilon = 1.0^{-6}$). When $|v_i - \alpha| < \varepsilon$ an event stating that $v = \alpha$ is created. This solution enables detection of asimptotic trends towards limit values (for example to detect that the level of the bathtub is approaching asimptotically an unknown limit).

The value of ε is the threshold below which any change in a variable is neglected. As a consequence there may be a multiple detection of the event $v = \alpha$ even if v is monotonic: consider the case where v_{i-1} and v_i are both in an ε–neighborhood of α. For asimptotic trends there is actually an indefinite number of v_i's that are all within the neighborhood of the limit. To ensure conciseness these events have to be merged into a single one, by extending the time range of the event.

If there is a time point beyond which all the state variables of the system stay within the ε–neiborhood of some value and their derivatives stay within the ε–neighborhood of 0, then the system is approaching quiescence. An appropriate *quiescence event* is created to represent such a situation.

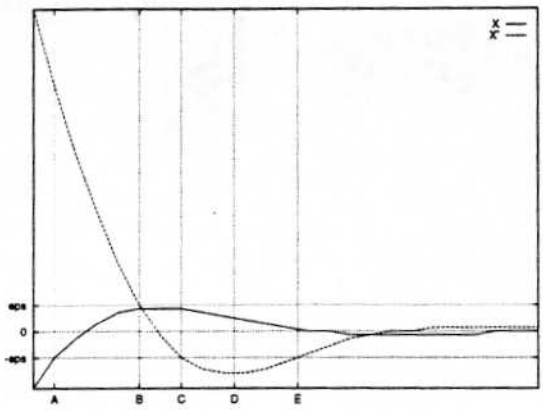
Finally, notice that while a variable may enter the neighborhood of some value at some point, its derivative may still be outside the neighborhood of 0. The variable's *qdir* may therefore change even though its magnitude doesn't. Since $qdir \neq std$ implies a change in the magnitude such a situation is inconsistent and such events could never match a QSIM behavior. To overcome this an *insignificant change event* is created indicating that the variables may change an indefinite number of times but always within a neighborhood of some value. This is the situation illustrated in figure 3.

Behavior interpretation

This process determines whether a trajectory is an instance of a behavior and, if so, yields a numerical interpretation of landmarks of the behavior.

Since the trajectory and the behaviors cover a finite amount of time[2] they may include not enough information to support the conclusion that the trajectory is an instance of the behavior. We adopt a conservative approach: unless the contrary can be proven, the trajectory is considered instance of the behavior. Of course, the longer is the numerical simulation and/or the qualitative behavior, the greater is the amount of information that can be used to get a definite answer. Thus one way to increase the resolution power of the

[2]Not necessarily the same since qualitative time rarely is sufficiently precise to be unambiguously compared to values of the real line.



When x enters the neighborhood of 0 ($t \geq A$), its derivative x' is still changing significantly (*i.e.* $x' \notin [-\varepsilon, \varepsilon]$). This portion of trajectory yields the following events:

time	description
$\geq A$	insignificant change for $x = 0$
$[B, C]$	$x' = 0$, dec
D	x' at a minimum, std
$\geq E$	x and x' quiescent at 0

Figure 3: Insignificant change events

method is to extend the temporal horizon(s), at the cost of an increased complexity.

Construction of behavior interpretation is based on the following steps. First, to each extracted event (for variable v) is assigned a sequence of qualitative states of the behavior (called *segment*) consisting of those consecutive states that qualitatively correspond to the event (*i.e.* the *qmag* of the state has the same sign as that in the event and *qdir*'s are the same). Additionally, insignificant–change events match any state and quiescence events match only quiescent states. In this way situations like those depicted in figure 3 are correctly matched with behaviors showing synchronous events of variables x and $\dot{x}$. If at the end of the analysis an event has an empty segment then the match fails.

Second, numerical information in an event is used to restrict ranges of variables' magnitudes in states included in the event's segment. By assigning values to variable TIME a global consistency among different variables is achieved. If a landmark results with an empty range the match fails.

Finally, total order between adjacent landmarks of a variable is enforced by requiring that they differ for at least ε (unless they are involved in insignificant–change events, where by definition the variable magnitude is constant).

The complexity of event extraction and behavior interpretation is $O(k(n+m))$, where k is the number of variables being analysed, n is the number of numerical events of the variable trajectory, and m is the number of qualitative states in the behavior.

Statistical analysis

Statistical experiment are based on generating N instances of the QDE, simulating them against K initial states and comparing these NK trajectories against M behaviors[3]. The aim is to obtain an estimation of (i) the *occurrence probability* of a behavior b: the probability that a random instance of the QDE simulated against a random initial state satisfying the qualitative initial state generates a trajectory τ that is an instance of b; and of (ii) the average value of some function $\mu(\tau)$ of the trajectory.

To obtain the first kind of information the problem could be in principle formulated as a multinomial random variable X, whose values $<x_1, x_2, ..., x_M>$ give the number of trajectories that match behavior 1,2, ..., M, with $\sum x_i = NK$. The choice of such a r.v. is justified by the fact that the M behaviors are exhaustive and mutually exclusive and the samples are independent (since we're drawing them from an infinite population and one instance is independent from the preceding ones).

However this is not possible because of *multiple classifications*. In fact, due to insufficient information, a trajectory can match more than one behavior, invalidating thus the choice of such a r.v.

To get around this obstacle we define M binomial r.v. X_1, ..., X_M, where each X_i gives the number of trajectories that match behavior i. In this way M different experiments are run using the same sample of trajectories. Notice that in this way each behavior is treated as being independent from the others.

Standard statistical tests can then be run to get the required information and an estimate of its accuracy. In particular, the proportion p of sampled trajectories that match a behavior is an unbiased estimator of the occurrence probability for the behavior and a large–sample confidence interval or hypothesis test can be easily derived.

Though not shown in the previous example, one could define other binomial r.v. that cover sets of behaviors, and in this way overcome the limitation due to the inability to use a multinomial r.v. For example, to estimate the probability of occurrence of behaviors 1 to 5 but not behavior 6 for the oscillatory system one could define the r.v. X_{1-5}: "number of trajectories matching behavior 1 or behavior 2 or ... behavior 5 and not behavior 6". By appropriately collecting the samples that satisfy such a specification and running the same tests described above, the desired answer can be determined.

To estimate the average value of some parameter μ it is necessary first to collect the sampled trajecto-

[3] In the example presented above $K = 1$.

ries that satisfy some requirements (for example those exhibiting an overdamped dynamics, *i.e.* matching behavior 1 of our example) and then extract from them samples of μ. If the sample size is sufficienty large the average value of μ can be assumed to be normally distributed (central limit theorem) and from the sample average and deviation a confidence interval can be derived for μ.

Discussion and related work

The Monte Carlo method just described is relatively independent from the features used to generate the qualitative behaviors. It can also be used with TeQSIM, a simulator based on QSIM that generates only behaviors that satisfy trajectory constraints specified in temporal logic (Brajnik and Clancy, 1996). TeQSIM can be employed as a means to reduce the number of behaviors used for classification. Only those satisfying certain requirements will be compared to sampled trajectories, with substantial resource saving. Of course, behaviors will not be exhaustive any more, and some trajectory will not be instance of any behavior.

The method can also be used with purely qualitative behaviors (*i.e.* without using QSIM to generate absolute bounds for landmarks). Since the matching process is based on qualitative features of trajectories and behavior states, the numeric information present in behaviors plays a minor role.

Finally, this method could also be applied to classify trajectories with respect to temporal logic statements. In fact, a behavior can be represented as a (quite complex) conjunction of logical statements describing each state. With a conceptually simple extension the method could then determine the probability that a temporal logic statement is true for members of the QDE by checking that events extracted from trajectories satisfy the temporal logic formulæ. Except for statistical processing, this is similar to the work that is being carried on by Gazi, Ungar and colleagues cited above. They sample instances of a QDE in order to verify properties of the dynamics of those instances without performing a qualitative simulation. In particular they extract from trajectories minima and maxima events and check them against temporal logic statements (using propositional linear–time temporal logic) to verify invariant properties of controllers for chemical plants. While sharing the same techniques for sampling the QDE, our method:

- refers to qualitative behaviors and properties identified by QSIM (like quiescence, stability). Since trajectories are matched to behaviors these properties are then associated to matching trajectories as well. Using only numerical information this can be difficult or impossible;
- does a detailed analysis of trajectories to extract events that support behavior matching; more detailed than just extracting minima and maxima;
- provides a statistical characterization of behaviors and of parameters of trajectories/behaviors;
- provides actual concrete instances of behaviors and of differential equations generating them. In addition, precise and sound numerical interpretation of symbolic landmarks is derived.

Conclusions

The Monte Carlo method presented above derives statistical data about qualitative behaviors. It is conceptually and algorithmically simple and quite effective. We believe it represents a viable way to couple qualitative representations of trajectories with numerical ones in a way that relaxes absolute accuracy to get statistically significant bounds without requiring strong assumptions on uncertainty.

The program implementing the method has been tested on several first– and second–order systems with satisfactory results. We are currently using it in controller design problems to study global changes of dynamics caused by changes in the controller.

The main limitation of the method, as for all Monte Carlo techniques, is the computational effort necessary to get statististically significant results. However, for off–line analysis of incompletely known dynamical systems (like controller design) the tool is viable and practically useful. It can be used whenever QSIM can, and being independent from the latter, it can take advantage of all the enhancements added to QSIM.

Acknowledgements

I'd like to thank Bert Kay for his program `ODESIM` that is being used for numerical integration and Giacomo Della Riccia and Laura Pagani for helpful discussions.

References

Brajnik, G. and Clancy, D. J. 1996. Temporal constraints on trajectories in qualitative simulation. In Clancey, B. and Weld, D., editors 1996, *Proc. of the Tenth National Conference on Artificial Intelligence.* AAAI Press. 979–984.

Comincioli, V. 1992. *Metodi numerici e statistici per le scienze applicate.* Casa Editrice Ambrosiana, Milan, Italy.

Doyle, J. and Sacks, E. 1989. Stochastic analysis of qualitative dynamics. In *11th IJCAI.*

Gazi, E.; Seider, W.D.; and Ungar, L.H. 1996. Verification of controllers in the presence of uncertainty: application to styrene polymerization. *Industrial & Engineering Chemistry Research* 35(7):2277–2287.

Gazi, E.; Seider, W.D.; and Ungar, L.H. 1997. A non–parametric Monte–Carlo technique for controller verification. *Automatica*. To appear.

Kuipers, B.J. 1994. ***Qualitative Reasoning: modeling and simulation with incomplete knowledge.*** MIT Press, Cambridge, Massachusetts.

Rubinstein, R.Y. 1981. *Simulation and the Monte Carlo methods.* John Wiley and Sons, New York.