Toward efficient parameter sweeping and optimisation of fluid performance models by Taylor-model based symbolic numerical integration

Richard A. Hayden
Imperial College London
London, United Kingdom
rh@doc.ic.ac.uk

Fluid-approximation or *mean-field* techniques are currently very popular approaches to the efficient analysis of massively-parallel Markov models. In this paper, we exploit the ODE-representation of these approximations to develop efficient techniques for simultaneous partially-symbolic numerical integration over parameter ranges. In particular, we borrow the *Taylor model* data structure of Makino and Berz [14] from the field of verified numerical integration. We use this to compute, in an efficient manner, tight bounds on the range of the ODE solutions over time with parameters lying in large intervals. This has applications to fast parameter sweeping, sensitivity analysis and global optimisation of model parameters in perfomance models amenable to fluid approximation.

1 Introduction

Fluid-approximation or *mean-field* techniques have become very popular recently as a means of addressing the state-space explosion problem for Markov chains consisting of large populations of interacting components [e.g. 2, 3, 5, 8, 10, 26]. This kind of approach is based on the construction of ordinary differential equations (ODEs) from a Markovian model which approximate the evolution of the number of components in each of their local states over time. Such an approach is very powerful since it results in an exponential reduction in the analysis complexity with respect to traditional explicit-state approaches.

The availability of these techniques has made it possible to analyse fairly large parameter spaces in reasonable periods of time by adopting the simple strategy of iterating one by one over each parameter combination and solving the corresponding set of ODEs [e.g. 6, 7, 9, 23, 24]. However such a brute-force approach will of course still become costly in the presence of large or multi-dimensional parameter ranges, even given the relatively low cost of the numerical integration of each of the individual ODE systems. A further disadvantage is the necessary discretisation of continuous parameter ranges and thus the potential accidental omission of important model configurations.

In a more ideal world, we would be able to keep track of the entire possible solution space over continuous ranges of parameters by performing only one numerical integration operation. In this paper we make tentative initial steps in this direction by leveraging devices developed in the field of verified numerical integration.

Verified numerical integration is the rigorous numerical computation of guaranteed enclosures of solutions to differential equations [e.g. 4, 12, 16, 21]. This is in contrast to traditional approximate solution methods such as the Runge–Kutta approaches which compute only an approximate solution up to some *local* error tolerance. A verified solver usually proceeds at discrete time steps much like traditional approaches. However, at each time step, it does not yield a single point solution, but rather a range guaranteed to contain the formal exact solution at that point. This is necessary in order to capture the uncertainty arising from the discrete integration at each time step and also that due to floating point round-off errors. An immediate requirement of such an approach is therefore the ability to propagate solution *ranges* from one time step to the next, as opposed to just propagating *points*. To do this efficiently

Submitted to:
PASTA 2011

and to avoid rapid growth in the bounds requires inexpensive and compact representations for the solution range at each time point.

For the purposes of solving systems of ODEs arising from fluid analysis of performance models, the traditional approximate solution approaches appear to be sufficient. Indeed, verified numerical integration would appear to be an unnecessary additional computational overhead given the fact that fluid analysis is already a (normally unquantified) approximation and, further, that model parameters are rarely specified with quantitative certainty. The purpose of this paper is not to attempt to apply verified numerical integration techniques to fluid analysis. Rather, we wish to borrow from this field, devices and techniques, which will allow us to achieve our goal of simultaneously performing a traditional approximate numerical integration over parameter ranges. This would seem like a promising approach since the kernel of the verified numerical integration problem — the efficient propagation of solution ranges — is, as we will see, directly related to the problem we have set out to address.

In the next section we will set out the problem of interest formally. We will then introduce the required technical devices borrowed from the field of verified numerical integration, specifically, *interval methods* and *Taylor models*. Finally, we will exhibit our approach on a simple peer-to-peer software update model.

2 Bounding of numerical solutions to initial value problems over parameter ranges

In this section we will set out formally the problem of interest. For simplicity we will consider here a straightforward fixed step-size Euler method as our basic numerical integration algorithm. However, we do not see why these ideas could not be extended to more efficient iterative solvers such as explicit Runge–Kutta methods with adaptive step size. We also consider here only time-autonomous systems, but again, this is just for the sake of brevity.

Let $\dot{\mathbf{y}}(t,\mathbf{p}) := \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y}(t,\mathbf{p}) = \mathbf{f}(\mathbf{y}(t,\mathbf{p}),\mathbf{p})$ be the initial value problem (IVP) of interest where for $t \in \mathbb{R}_+$, $\mathbf{y}(t,\mathbf{p}) \in \mathbb{R}^n$ is the solution vector, $\mathbf{p} \in \mathbb{R}^k$ is the vector of parameters and $\mathbf{y}(0,\mathbf{p}) = \mathbf{g}(\mathbf{p})$ is the vector of initial conditions.

Given some step size $h \in \mathbb{R}_+$ and number of steps $m \in \mathbb{Z}_+$, inducing a sequence of times $t_j := jh$ for $0 \leq j \leq m$, we then define $\tilde{\mathbf{y}}(t_j,\mathbf{p})$ to be the approximate numerical solution of this IVP computed, as mentioned above, in this case by applying the Euler method in the standard manner:

$$\tilde{\mathbf{y}}(t_j,\mathbf{p}) := \tilde{\mathbf{y}}(t_{j-1},\mathbf{p}) + h\mathbf{f}(\tilde{\mathbf{y}}(t_{j-1},\mathbf{p}),\mathbf{p}) \tag{1}$$

for $0 < j \leq m$ and taking $\tilde{\mathbf{y}}(t_0,\mathbf{p}) := \mathbf{g}(\mathbf{p})$.

Let $\mathbf{p}^L, \mathbf{p}^H \in \mathbb{R}^k$ specify an interval box of parameters $\mathbf{P} = [\mathbf{p}^L, \mathbf{p}^H]$ by:

$$[\mathbf{p}^L, \mathbf{p}^H] := \{\mathbf{p} \in \mathbb{R}^k \ : \ p_i \in [p_i^L, p_i^H] \text{ for } i = 1,\ldots,k\}$$

Our goal here is to compute time-varying bounds $\tilde{\mathbf{y}}^L(t_j,\mathbf{P})$ and $\tilde{\mathbf{y}}^H(t_j,\mathbf{P})$ efficiently such that for all $0 \leq j \leq m$ and all $\mathbf{p} \in P$:

$$\tilde{\mathbf{y}}^L(t_j,\mathbf{P}) \leq \tilde{\mathbf{y}}(t_j,\mathbf{p}) \leq \tilde{\mathbf{y}}^H(t_j,\mathbf{P}) \tag{2}$$

where vector inequalities are taken component wise.

3 Interval arithmetic

Interval arithmetic [e.g. 1, 11] extends standard arithmetic operations to operate directly on intervals. An interval $X = [x^L, x^H]$ for $x^L, x^H \in \mathbb{R}$ with $x^L \leq x^H$ is defined in the standard way as the set of all points $x \in \mathbb{R}$ such that $x^L \leq x \leq x^H$. We identify points $x \in \mathbb{R}$ with the interval $[x,x]$. The *midpoint* and *width* of an interval X are denoted by $m(X) := (x^L + x^H)/2$ and $w(X) := x^H - x^L$, respectively. An interval vector

$\mathbf{X} = [\mathbf{x}^L, \mathbf{x}^H]$ for $\mathbf{x}^L, \mathbf{x}^H \in \mathbb{R}^n$ is defined, as in the last section, by the Cartesian product of the constituent component intervals $X_i = [x_i^L, x_i^H]$.

Basic arithmetic operations $\odot \in \{+, -\times, \div\}$ are defined for intervals by:

$$X \odot Y := \{x \odot y \ : \ x \in X, y \in Y\}$$

so that, for example, $[x^L, x^H] + [y^L, y^H] = [x^L + y^L, x^H + y^H]$ and similarly for the other operations. Interval versions of other elementary functions can be defined similarly. For an arbitrary real function $h(x)$ and interval X, an *interval extension* $H(X)$ is an interval containing $h(x)$ for each $x \in X$, that is, $H(X) \supseteq \{h(x) \ : \ x \in X\}$. Interval extensions are often computed straightforwardly by substituting the interval X into $h(x)$ and evaluating the expression using interval arithmetic. However, the resulting interval is often wider than the actual function range. Consider for example, the function $g(x) := x/(x-1)$ evaluated over the interval $X := [3,4]$, so that the natural interval extension is $G([3,4]) = [3,4]/([3,4] - 1) = [3,4]/[2,3] = [1,2]$. However the actual range of $g(x)$ over X is $[4/3, 3/2]$. The reason for this overestimation is the so-called *dependency problem*; the fact that x occurs twice in the definition of $g(x)$ is not recognised by the natural interval extension and it is evaluated as if each occurrence was a different variable. In fact, in this case, the definition of $g(x)$ can be rewritten as $1 + 1/(x-1)$, so that the variable x occurs only once. Then the natural interval extension is the exact range $[4/3, 3/2]$.

The situation gets worse when considering interval vectors. For example, if $X := [-1,1]$ and we wish to enclose the vector function $\mathbf{g}(x) := (x,x)^T$ over X, performing the natural interval extension on each component of the vector yields the box $[-1,1] \times [-1,1]$. The true range, however, is just the line segment between $(-1,-1)^T$ and $(1,1)^T$. The problem of overestimation worsens here dramatically because the true range cannot be enclosed exactly as an interval vector. This phenomenon is known as the *wrapping effect* since non-interval ranges are wrapped by interval enclosures.

A naïve approach to computing bounds $\tilde{\mathbf{y}}^L(t_j, \mathbf{P})$ and $\tilde{\mathbf{y}}^H(t_j, \mathbf{P})$ as specified in the previous section would then simply be to evaluate $\mathbf{y}(0, \mathbf{p})$ and Eq. (1) recursively using interval arithmetic. Unfortunately, such a direct approach is likely to lead quickly to massive overestimation due to the aforementioned dependency problem and the wrapping effect. This approach is the unverified equivalent of Moore's *direct interval method* [20] for verified numerical integration. Various refinements to ameliorate the wrapping effect do exist, such as the *parallelepiped method* [19, 20], where, at each time step, the solution enclosure is represented by parallelepipeds instead of rectangular boxes. The *QR method* [13] was developed to stabilise this approach numerically by orthogonalisation of the associated transformation matrices. We believe that it would be fairly straightforward to benefit from these ideas in the unverified case developed in this paper. However, for parameter ranges which are not extremely small, we would still expect the results to be poor since the dependency problem remains largely untempered. Furthermore, any approach based just on interval methods is restricted by the fact that interval-based enclosures must be convex.

Since the results are usually fairly poor, and for the sake of brevity, we do not give any examples of purely interval-based bound computations here. Instead, we will proceed to introduce a much more advanced representation for the ODE solution at each time point, the *Taylor model*. The discussion of this section is not wasted however since a Taylor model also includes an interval component.

4 Taylor model arithmetic

Taylor models were first introduced by Makino and Berz [14, 15] in order to address both the dependency problem and the wrapping effect in the bounding of function ranges. A Taylor model representation of a function consists of a Taylor polynomial of some given degree and an interval *remainder bound*.

Specifically, let $f(\mathbf{p}) : \mathbf{P} \to \mathbb{R}$ be a function on some interval vector $\mathbf{P} \subseteq \mathbb{R}^k$. Let $\mathbf{p}^0 \in \mathbf{P}$ and $q_f : \mathbb{R}^k \to \mathbb{R}$ be an rth-degree k-variate polynomial in the variables $(p_i - p_i^0)$ for $1 \leq i \leq k$. Furthermore let

R_f be an interval. Then $T_f := q_f + R_f$ is an rth-degree Taylor model of f on $\mathbf{P}$ about $\mathbf{p}^0$ if:

$$f(\mathbf{p}) \in T_f(\mathbf{p}) = q_f(\mathbf{p} - \mathbf{p}^0) + R_f$$

for all $\mathbf{p} \in \mathbf{P}$. Taylor models of functions may be computed very naturally by using Taylor's theorem to truncate Taylor expansions and bound the remainder interval. Starting from existing Taylor models $T_f = q_f + R_f$ and $T_g = q_g + R_g$ on $\mathbf{P}$ about $\mathbf{p}^0$ of f and g, respectively, we may construct Taylor models for combinations of these functions by employing Taylor model arithmetic [14]. For example, we have:

$$f \pm g \in T_{f\pm g} := T_f \pm T_g := (q_f \pm q_g) + (R_f \pm R_g)$$

where arithmetic on intervals is performed as in the previous section. In the case of the product $f \times g$, we have:

$$f \times g \in q_f \times q_g + q_f \times R_g + q_g \times R_f + R_f \times R_g$$

Note that $q_f \times q_g$ is a polynomial of degree $2r$. We do not wish for Taylor models to grow exponentially in size with the number of multiplications performed on them or they will not remain a compact and efficient representation for very long. For this reason we split $q_f \times q_g =: q_{f\times g} + q_e$ where $q_{f\times g}$ are the parts of $q_f \times q_g$ of degree r and below and q_e contains the terms of degree higher than r. For the Taylor model of $f \times g$, we will take $q_{f\times g}$ as the polynomial component and absorb the polynomial q_e into the remainder bound along with the terms $q_f \times R_g$, $q_g \times R_f$ and $R_f \times R_g$. Specifically, we define:

$$f \times g \in T_{f\times g} := T_f \times T_g := q_{f\times g} + (R(q_e) + R(q_f) \times R_g + R(q_g) \times R_f + R_f \times R_g)$$

where the function $R(q)$ takes a polynomial q and returns an interval enclosing its range on $\mathbf{P}$.

There are many methods described in the literature to compute $R(q)$ [17]. In fact, the exact range is itself an interval but computing it is NP hard [17] so various overestimating procedures are usually employed. Most approaches focus on exact bounding of the first- and second-degree parts of q and then bound the rest directly by substituting in $\mathbf{P}$ and performing interval arithmetic. In fact, even exact bounding of a degree two polynomial can be very expensive. For this reason, we adopt the compromise approach given by Lin and Stadtherr [12], in which only the first-degree and diagonal second-degree terms are bounded exactly. That is, we compute:

$$R(q) := \sum_{i=1}^{k} \left[a_i \left(P_i - p_i^0 + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right] + R$$

where the first-degree and diagonal second-degree terms of q are $\sum_{i=1}^{k} \left(a_i(p_i - p_i^0)^2 + b_i(p_i - p_i^0) \right)$ and R is the interval obtained by evaluating all of the other terms directly using interval arithmetic.[1] For a Taylor model $T = q + R$, by $R(T)$ we mean $R(q) + R$, that is, an interval including the entire range of the Taylor model.

The key benefit of Taylor models over plain interval methods are that, wherever possible, computations are performed *symbolically* in terms of the parameter variable $\mathbf{p}$. In this manner, we alleviate greatly both the dependency problem and the wrapping effect. We defer only to the interval remainder bound to 'mop up' the higher-degree (and thus hopefully less significant) parts of the Taylor model as the computation proceeds.

We return now to the central problem of this paper, the efficient computation of the bounds $\tilde{\mathbf{y}}^L(t_j, \mathbf{P})$ and $\tilde{\mathbf{y}}^H(t_j, \mathbf{P})$ of Eq. (2). Our approach will be a fairly straightforward recursive application of Taylor

[1] For numerical reasons, if $|a_i| < \varepsilon$ for some small positive ε we evaluate the whole of q directly using interval arithmetic.

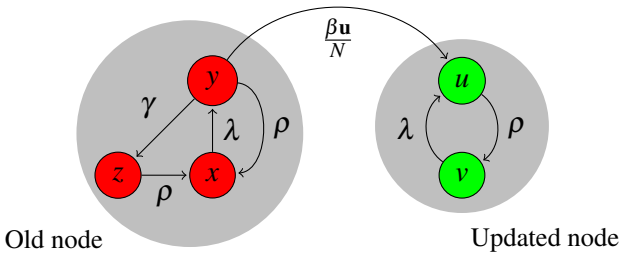


Figure 1: *State representation of the behaviour of a single node in the software update process model.*

model arithmetic. Specifically, for each time point t_j, we will construct a Taylor model $T_{\tilde{\mathbf{y}}(t_j)}$ of $\tilde{\mathbf{y}}(t_j, \mathbf{p})$ on $\mathbf{P}$ about the (component-wise) midpoint $\mathbf{p}^0 := m(\mathbf{P})$.[2] Given a Taylor model $T_{\tilde{\mathbf{y}}(t_{j-1})}$ of $\tilde{\mathbf{y}}(t_{j-1}, \mathbf{p})$ for $j > 0$, a Taylor model $T_{\tilde{\mathbf{y}}(t_j)}$ of $\tilde{\mathbf{y}}(t_j, \mathbf{p})$ can be computed according to the rules of Taylor model arithmetic directly from Eq. (1). An initial Taylor model of $\tilde{\mathbf{y}}(t_0, \mathbf{p})$ is given by computing one of $\mathbf{g}(\mathbf{p})$. The bounds we seek are then obtained by computing the interval range $R(T_{\tilde{\mathbf{y}}(t_j)})$ of $T_{\tilde{\mathbf{y}}(t_j)}$ at each time point t_j. In the next section we illustrate this procedure and the resulting bounds explicitly for the fluid approximation ODEs derived from a simple Markovian model.

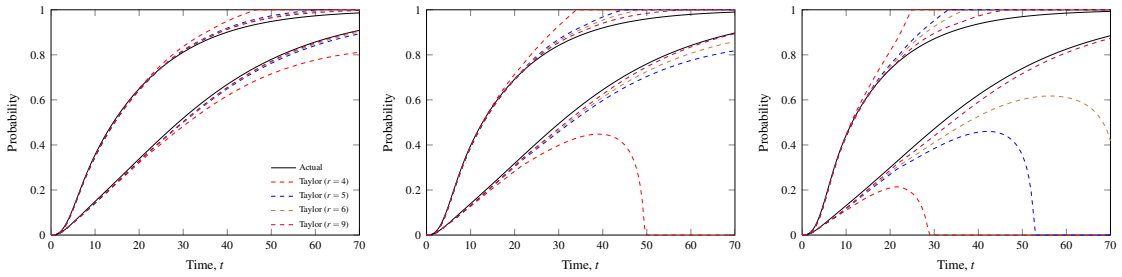
5 Taylor-model derived bounds for a peer-to-peer software update model

We consider a very simple continuous-time Markov chain (CTMC) model of a peer-to-peer software update process. There are two general classes of nodes in this system which we term *old* and *updated*. Old nodes are those running an old software version and new nodes are those which have been updated to a new version. Both types of nodes switch between being *on* and *off* (at rates λ and ρ, respectively). When an updated node is on, an old node may locate it and subsequently update itself in a peer-to-peer fashion. Whenever an old node comes on, it polls the network for new nodes (so it can be updated) before giving up if it does not find one after an exponential delay at rate γ. Updated nodes have two states which are just on and off, which we write as u and v, respectively. Old nodes have three states: on (y), off (x) and a state representing an old node which is on but has given up seeking updates (z). We follow a usual methodology for the stochastic modelling of epidemics in that the chance of an old node finding an updated one in a small period of time is $\frac{\beta \mathbf{u}}{N}$, where N is the total component population and $\mathbf{u}$ is the number of nodes in state u, that is, the rate is proportional to the number of available updated nodes. The transitions are given in Figure 1.

Let $\mathbf{x}_t$ be the number of nodes in state x at time t, rescaled by N, and similarly for the other states. Then $\boldsymbol{S}_t = (\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t, \mathbf{u}_t, \mathbf{v}_t)^T$ is the rescaled discrete aggregated state vector. We will write the corresponding continuous fluid approximation as $(\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t), \mathbf{u}(t), \mathbf{v}(t))^T$ given in the usual [e.g. 8] manner by the system of ODEs:

$$
\begin{aligned}
\dot{\mathbf{x}}(t) &= -\lambda \mathbf{x}(t) + \rho \mathbf{z}(t) + \rho \mathbf{y}(t) \\
\dot{\mathbf{y}}(t) &= -\beta \mathbf{y}(t) \mathbf{u}(t) - \rho \mathbf{y}(t) - \gamma \mathbf{y}(t) + \lambda \mathbf{x}(t) \\
\dot{\mathbf{z}}(t) &= -\rho \mathbf{z}(t) + \gamma \mathbf{y}(t) \\
\dot{\mathbf{u}}(t) &= -\rho \mathbf{u}(t) + \lambda \mathbf{v}(t) + \beta \mathbf{y}(t) \mathbf{u}(t) \\
\dot{\mathbf{v}}(t) &= -\lambda \mathbf{v}(t) + \rho \mathbf{u}(t)
\end{aligned}
$$

[2] Vectors of Taylor models are handled simply in a component-wise fashion.



(a) $\Gamma = 0.6 + [-0.25, 0.25]$ **(b)** $\Gamma = 0.6 + [-0.3, 0.3]$ **(c)** $\Gamma = 0.6 + [-0.35, 0.35]$

Figure 2: *Taylor-model computed bounds for the cumulative distribution function of the time for an old node to become updated in the software update model. Also shown are 'actual' bounds computed by discretising the parameter range with a very small step.*

In this section, we keep things simple by considering a univariate Taylor model, that is, on a single parameter. The parameter we will consider varying is γ. Write then $(\tilde{\mathbf{x}}(t_j,\gamma), \tilde{\mathbf{y}}(t_j,\gamma), \tilde{\mathbf{z}}(t_j,\gamma), \tilde{\mathbf{u}}(t_j,\gamma), \tilde{\mathbf{v}}(t_j,\gamma))^T$ for the Euler numerical integration of these ODEs (according to Eq. (1)) over some sequence of time steps $t_j = jh$, for a variable value of γ, fixed initial condition $(0.9, 0, 0, 0, 0.1)^T$ and fixed values of the other parameters: $\lambda = 0.2$, $\beta = 2.0$ and $\rho = 0.1$. Let Γ be an interval range of interest for the model parameter γ, then we may compute the vector of Taylor models $(T_{\tilde{\mathbf{x}}(t_j)}, T_{\tilde{\mathbf{y}}(t_j)}, T_{\tilde{\mathbf{z}}(t_j)}, T_{\tilde{\mathbf{u}}(t_j)}, T_{\tilde{\mathbf{v}}(t_j)})^T$ about $\gamma^0 := m(\Gamma)$ according to the last section. Specifically, given for example $(T_{\tilde{\mathbf{x}}(t_{j-1})}, T_{\tilde{\mathbf{y}}(t_{j-1})}, T_{\tilde{\mathbf{z}}(t_{j-1})}, T_{\tilde{\mathbf{u}}(t_{j-1})}, T_{\tilde{\mathbf{v}}(t_{j-1})})^T$ for $j > 0$, we can compute:

$$T_{\tilde{\mathbf{y}}(t_j)} = T_{\tilde{\mathbf{y}}(t_{j-1})} + h\left(-\beta T_{\tilde{\mathbf{y}}(t_{j-1})} \times T_{\tilde{\mathbf{u}}(t_{j-1})} - \rho T_{\tilde{\mathbf{y}}(t_{j-1})} - T_\gamma T_{\tilde{\mathbf{y}}(t_{j-1})} + \lambda T_{\tilde{\mathbf{x}}(t_{j-1})}\right) \tag{3}$$

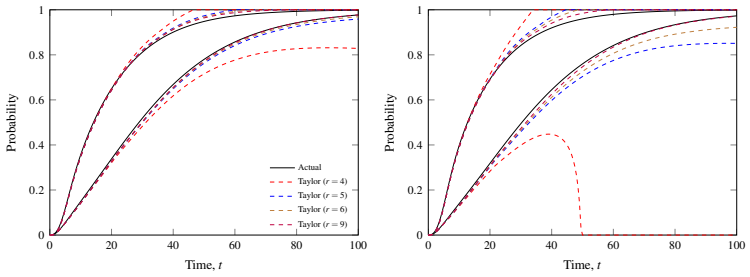
and similarly for the other components, where $T_\gamma := \gamma^0 + (\gamma - \gamma^0)$ is the Taylor model of the variable γ. In fact, this computation is actually somewhat naïve and can exhibit rapid growth in the remainder interval, thus causing the resulting range bound to blow up after only a small period of time. Indeed, since the width of an interval sum is equal to the sum of the widths of the intervals, it is clear from the form of Eq. (4) that the width of the interval remainder bound in the Taylor model is non-decreasing with j. In cases where the ODE solutions contract, we would hope that the remainder bound might also be able to contract. Rewriting the update rule by collecting terms as follows (and similarly for the other components) yields:

$$T_{\tilde{\mathbf{y}}(t_j)} = T_{\tilde{\mathbf{y}}(t_{j-1})} \times \left(1 + h(-\beta T_{\tilde{\mathbf{u}}(t_{j-1})} - \rho - T_\gamma)\right) + h\lambda T_{\tilde{\mathbf{x}}(t_{j-1})} \tag{4}$$

This improves the situation massively. In particular, it is clear that it is now theoretically possible for the interval remainder bound to shrink as the integration progresses. This issue is also directly related to the way in which rearranging the order of interval computations can significantly improve the tightness of the resulting interval, as discussed earlier.

Figures 2 and 3 give the results of applying this scheme to compute bounds on the fluid approximation of the cumulative distribution function of the time for an old node to become updated in the software update model, that is, the quantity $\frac{\tilde{\mathbf{u}}(t,\gamma)+\tilde{\mathbf{v}}(t,\gamma)-0.1}{0.9}$. We consider different parameter ranges for $\gamma \in \Gamma$, different degrees r for the Taylor models and use a step size for the numerical integration of $h = 0.5$.

We observe that for both large parameter and time ranges, we can obtain impressively tight Taylor-model derived bounds for relatively small degree of Taylor polynomial. We believe that we may be



(a) $\Gamma = 0.6 + [-0.25, 0.25]$ **(b)** $\Gamma = 0.6 + [-0.3, 0.3]$

Figure 3: *Taylor-model computed bounds for the cumulative distribution function of the time for an old node to become updated in the software update model over a larger time range. Also shown are 'actual' bounds computed by discretising the parameter range with a very small step.*

able to significantly reduce the degree of polynomial required to maintain tight bounds for time ranges of interest by considering alternative representations for the remainder bound. Specifically, in the context of verified numerical integration, Lin and Stadtherr [12] define a new type of Taylor model, which, instead of an interval remainder bound, uses a parallelepiped representation for the remainder. This extension is analogous to the parallelepiped techniques mentioned earlier for purely interval-based techniques. As long as the degree of Taylor polynomial remains reasonably small (as in the cases considered in this paper), this approach is extremely efficient and not a lot more expensive than a single numerical integration.

Bounds computed by Taylor model techniques can form the basis of verified branch and bound global optimisation algorithms [22]. We intend to develop similar global optimisation algorithms in our non-verified context. These could be used, for example, to determine, in an efficient manner, optimal parameters minimising some reward function (such as energy consumption) subject to the satisfaction of a given service level agreement.

Finally, we hope to integrate the kind of techniques described in this paper in our *Grouped PEPA Analyser (GPA)* [25] tool, making use of the efficient implementation of Taylor model arithmetic provided by the *COSY Infinity* [18] software package.

6 Conclusion

In this paper, we have shown that Taylor-model techniques taken from the field of verified numerical integration can be used to compute bounds efficiently on the numerical solution of systems of ODEs over large parameter ranges. For Markovian performance models which are amenable to ODE-based fluid approximation, these techniques are directly useful for sensitivity analysis or for coping with uncertainty in parameters. Furthermore, they are intended to form the future basis of efficient global optimisation algorithms for stochastic models amenable to fluid approximation.

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computation*. Academic Press, 1983.

[2] R. Bakhshi, L. Cloth, W. Fokkink, and B. R. Haverkort. Mean-field framework for performance evaluation of push-pull gossip protocols. *Performance Evaluation*, 68(2):157–179, Feb. 2011. ISSN 01665316. doi: 10.1016/j.peva.2010.08.025. URL `http://dx.doi.org/10.1016/j.peva.2010.08.025`.

[3] M. Benaïm and J.-Y. Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 65(11-12):823–838, Nov. 2008. ISSN 01665316. doi: 10.1016/j.peva.2008.03.005. URL `http://dx.doi.org/10.1016/j.peva.2008.03.005`.

[4] M. Berz and K. Makino. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. *Reliable Computing*, 4(4):361–369, Nov. 1998. ISSN 1385-3139. doi: 10.1023/A:1024467732637. URL http://www.springerlink.com/content/up50314qw3073l17/.

[5] A. Bobbio, M. Gribaudo, and M. Telek. Analysis of Large Scale Interacting Systems by Mean Field Method. In *Fifth International Conference on Quantitative Evaluation of Systems*, pages 215–224. IEEE, Sept. 2008. ISBN 978-0-7695-3360-5. doi: 10.1109/QEST.2008.47. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4634974.

[6] A. Clark, S. T. Gilmore, and M. Tribastone. Scalable Analysis of Scalable Systems. In M. Chechik and M. Wirsing, editors, *Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering*, volume 5503 of *Lecture Notes in Computer Science*, pages 1–17, Berlin, Heidelberg, Mar. 2009. Springer Berlin Heidelberg. ISBN 978-3-642-00592-3. doi: 10.1007/978-3-642-00593-0. URL http://portal.acm.org/citation.cfm?id=1533013.1533015.

[7] A. Clark, S. T. Gilmore, and M. Tribastone. Quantitative Analysis of Web Services Using SRMC. In M. Bernardo, L. Padovani, and G. Zavattaro, editors, *Formal Methods for Web Services*, volume 5569 of *Lecture Notes in Computer Science*, pages 296–339, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-01917-3. doi: 10.1007/978-3-642-01918-0. URL http://www.springerlink.com/content/124751nlq5575328/.

[8] R. A. Hayden and J. T. Bradley. A fluid analysis framework for a Markovian process algebra. *Theoretical Computer Science*, 411(22-24):2260–2297, May 2010. ISSN 03043975. doi: 10.1016/j.tcs.2010.02.001. URL http://dx.doi.org/10.1016/j.tcs.2010.02.001.

[9] R. A. Hayden, A. Stefanek, and J. T. Bradley. Fluid computation of passage time distributions in large Markov models. *To appear in Theoretical Computer Science*, 2011. doi: 10.1016/j.tcs.2011.07.017. URL http://aesop.doc.ic.ac.uk/pubs/fluid-passage-time/.

[10] J. Hillston. Fluid flow approximation of PEPA models. In *Second International Conference on the Quantitative Evaluation of Systems*, pages 33–42. IEEE, Sept. 2005. ISBN 0-7695-2427-3. doi: 10.1109/QEST.2005.12. URL http://www.computer.org/portal/web/csdl/doi/10.1109/QEST.2005.12.

[11] L. Jaulin, K. Michel, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, 2001.

[12] Y. Lin and M. A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Applied Numerical Mathematics*, 57(10):1145–1162, Oct. 2007. ISSN 01689274. doi: 10.1016/j.apnum.2006.10.006. URL http://portal.acm.org/citation.cfm?id=1280292.1280494.

[13] R. Lohner. *Einschliessung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben und Anwendungen*. Ph. D., Universität Karlsruhe, 1988.

[14] K. Makino and M. Berz. Remainder Differential Algebras and Their Applications. In *Computational Differentiation: Techniques, Applications, and Tools*, pages 63–74. SIAM, 1996.

[15] K. Makino and M. Berz. Efficient Control of the Dependency Problem Based on Taylor Model Methods. *Reliable Computing*, 5(1):3–12, 1999. ISSN 1385-3139. doi: 10.1023/A:1026485406803. URL http://www.springerlink.com/content/qv3028612g1vl25l/.

[16] K. Makino and M. Berz. Suppression of the wrapping effect by Taylor model-based validated integrators. Technical report, Michigan State University, 2003. URL http://bt.pa.msu.edu/cgi-bin/display.pl?name=VIRC03.

[17] K. Makino and M. Berz. Taylor model range bounding schemes. In *Third International Workshop on Taylor Methods*, 2004.

[18] K. Makino and M. Berz. COSY INFINITY Version 9. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 558(1):346–350, Mar. 2006. ISSN 01689002. doi: 10.1016/j.nima.2005.11.109. URL http://dx.doi.org/10.1016/j.nima.2005.11.109.

[19] R. E. Moore. Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solutions of ordinary differential equations. *Error in Digital Computation*, 2:103–140, 1965.

[20] R. E. Moore. *Interval analysis*. Prentice Hall, 1966.

[21] M. Neher, K. R. Jackson, and N. S. Nedialkov. On taylor model based integration of ODEs. *SIAM Journal on Numerical Analysis*, 45 (1):236–262, 2007.

[22] A. M. Sahlodin and B. Chachuat. Convex/concave relaxations of parametric ODEs using Taylor models. *Computers & Chemical Engineering*, 35(5):857–844, Jan. 2011. ISSN 00981354. doi: 10.1016/j.compchemeng.2011.01.031. URL http://dx.doi.org/10.1016/j.compchemeng.2011.01.031.

[23] A. Stefanek, R. A. Hayden, and J. T. Bradley. Fluid analysis of energy consumption using rewards in massively parallel Markov models. In *2nd ACM/SPEC International Conference on Performance Engineering (ICPE)*, pages 121–132, Karlsruhe, 2011. doi: http://dx.doi.org/10.1145/1958746.1958767.

[24] A. Stefanek, R. A. Hayden, and J. T. Bradley. Fluid computation of the performance-energy trade-off in large scale Markov models. *To appear in SIGMETRICS Performance Evaluation Review*, 2011. URL http://aesop.doc.ic.ac.uk/pubs/fluid-performance-energy.

[25] A. Stefanek, R. A. Hayden, and J. T. Bradley. GPA - A tool for fluid scalability analysis of massively parallel systems. In *To appear in 8th International Conference on Quantitative Evaluation of Systems (QEST)*, 2011. URL http://aesop.doc.ic.ac.uk/pubs/gpanalyser.

[26] M. Tribastone, S. T. Gilmore, and J. Hillston. Scalable Differential Analysis of Process Algebra Models. *IEEE Transactions on Software Engineering*, PP(99):1, 2010. ISSN 0098-5589. doi: 10.1109/TSE.2010.82. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5567115.