Bayesian Probabilistic Numerical Methods for Differential Equations

Junyang Wang
[1]Newcastle University UK

February 22, 2021

Differential equations provide an important mathematical framework for modelling the behaviour of quantities that evolve in continuous time and space, often within a complex system or process.

Many physical and scientific phenomena, including fundamental laws such as Newtons laws of motion, are formulated as differential equations.

However, most useful differential equations lack a closed form solution expressible in terms of established functions, and so in practice numerical (computational) methods are required to obtain a discrete approximation to quantities of interest.

Classical numerical methods approximate quantities of interest by taking a finite number of evaluations from some known and computationally tractable quantity, such as the gradient field, and use these within an algorithm to construct an approximation.

This is in principle similar to statistics or machine learning, where a finite number of observations of some unknown, underlying process are used to infer the process itself.

In this view, numerical algorithms can be interpreted as estimators, and statistical considerations can be brought to bear. Going further, on can consider probabilistic numerical methods, which output a probability distribution over the quantity of interest. This is analogous to the Bayesian version of algorithms for solving differential equations.

We propose a probabilistic numerical method for solving nonlinear, partial differential equations, in particular initial value problems.

We give a review of the proposed algorithm in the case of linear PDEs, and present computational results for a linear example and a nonlinear example.

Consider the initial value problem on 1D spatial domain

$$Du(t,x) = f(t,x), \qquad x \in \Omega,\ t \in [0,T] \tag{1}$$

$$u(0,x) = g(x), \qquad x \in \Omega. \tag{2}$$

Define a GP prior $GP(m,K)$ on u. The idea is then to update this prior sequentially via evaluations of the gradient field $f(t,x)$. Because D is an linear operator, the updated distribution is still a Gaussian Process.

We want to estimate the solution u on a grid of points

$$[0 = t_0 < t_1 < \cdots < t_{n-1} = T] \times [x_{min} = x_0 < x_1 < \cdots < x_m = x_{max}]$$

We now have a grid of points $v_{ij} = (t_i, x_j)$. This is somewhat difficult to work with because the kernel is a function of 4 variables (each point has a time and position coordinate), so a kernel 'matrix' would be 4 dimensional.

To get around this, we order the points lexicographically.

Let

$$\boldsymbol{\rho_i} = \{\boldsymbol{v_0}, \boldsymbol{v_1}, \ldots, \boldsymbol{v_i}\}$$

where

$$\begin{aligned}\boldsymbol{v_i} = \{v_{i1}, v_{i2}, \ldots, v_{im}\} &= \{w_{i\times m+1}, w_{i\times m+2}, \ldots, w_{(i+1)\times m}\} \\ &= \{(t_i, x_1), (t_i, x_2), \ldots, (t_i, x_m)\}\end{aligned}$$

where the index of w denotes the lexicographical order, starting from 1.

Consider a space-time kernel $\Sigma((t,x),(t',x')) = K(t,t')C(x,x')$.
Then for example on $\boldsymbol{\rho_i}$ we have

$$\Sigma(\boldsymbol{\rho_i},\boldsymbol{\rho_i}) = \begin{bmatrix} \Sigma(w_1,w_1) & \dots & \Sigma(w_1,w_{i\times m}) \\ \vdots & \vdots & \vdots \\ \Sigma(w_{i\times m},w_1) & \dots & \Sigma(w_{i\times m},w_{i\times m}) \end{bmatrix}$$
$$= \begin{bmatrix} K(t_0,t_0)C(x_1,x_1) & \dots & K(t_0,t_i)C(x_1,x_m) \\ \vdots & \vdots & \vdots \\ K(t_i,t_0)C(x_m,x_1) & \dots & K(t_i,t_i)C(x_m,x_m) \end{bmatrix}$$

Now that we have defined the kernel matrix, we can update the Gaussian Process via the usual conjugate formulae.

$$m^{i+1}(r) = m(r) + [\Sigma(r, \boldsymbol{\rho_0}), \Sigma_{\bar{D}}(r, \boldsymbol{\rho_i})] M_{i+1}^{-1} \begin{bmatrix} u_0 - m(\rho_0) \\ f_{1:i} - m_D(\boldsymbol{\rho_i}) \end{bmatrix}$$

and

$$\Sigma^{i+1}(r, r') = \Sigma(r, r') - [\Sigma(r, \boldsymbol{\rho_0}), \Sigma_{\bar{D}}(r, \boldsymbol{\rho_i})] M_{i+1}^{-1} \begin{bmatrix} \Sigma(\boldsymbol{\rho_0}, r') \\ \Sigma_D(\boldsymbol{\rho_i}, r') \end{bmatrix}$$

where

$$f_{1:i} = \{f(\boldsymbol{v_0}), f(\boldsymbol{v_1}), \ldots, f(\boldsymbol{v_i})\}$$

$$f(\boldsymbol{v_i}) = \{f(t_i, x_1), f(t_i, x_2), \ldots, f(t_i, x_m)\}$$

$$M_{i+1} = \begin{bmatrix} \Sigma(\boldsymbol{\rho_0}, \boldsymbol{\rho_0}) & \Sigma_{\bar{D}}(\boldsymbol{\rho_0}, \boldsymbol{\rho_i}) \\ \Sigma_D(\boldsymbol{\rho_i}, \boldsymbol{\rho_0}) & \Sigma_{D\bar{D}}(\boldsymbol{\rho_i}, \boldsymbol{\rho_i}) \end{bmatrix}$$

Inverting the M_{i+1} matrix is expensive, as it is a $(i+1)m$ by $(i+1)m$ matrix, to get around this, it can be shown via induction that:

$$m^{i+1}(r) = m^i(r) + \Sigma^i_{\bar{D}}(r, \boldsymbol{v_i})\Sigma^i_{D\bar{D}}(\boldsymbol{v_i}, \boldsymbol{v_i})^{-1}(f_i - m^i_D(\boldsymbol{v_i}))$$

$$\Sigma^{i+1}(r, r') = \Sigma^i(r, r') - \Sigma^i_{\bar{D}}(r, \boldsymbol{v_i})\Sigma^i_{D\bar{D}}(\boldsymbol{v_i}, \boldsymbol{v_i})^{-1}\Sigma^i_D(\boldsymbol{v_i}, r')$$

With these sequential updating formulae, one only has to invert $\Sigma^i_{D\bar{D}}(\boldsymbol{v_i}, \boldsymbol{v_i})^{-1}$, which is only m by m. Note in the case there's only one independent variable (meaning an ordinary differential equation), $\Sigma^i_{D\bar{D}}(\boldsymbol{v_i}, \boldsymbol{v_i})^{-1}$ is just a scalar so there's no matrix inversion at all.

Experimental example: Consider the 1D heat equation on $x \in [0, 1]$.

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0 \tag{3}$$

With initial and boundary conditions:

$$u(0, x) = \sin(\pi x)$$

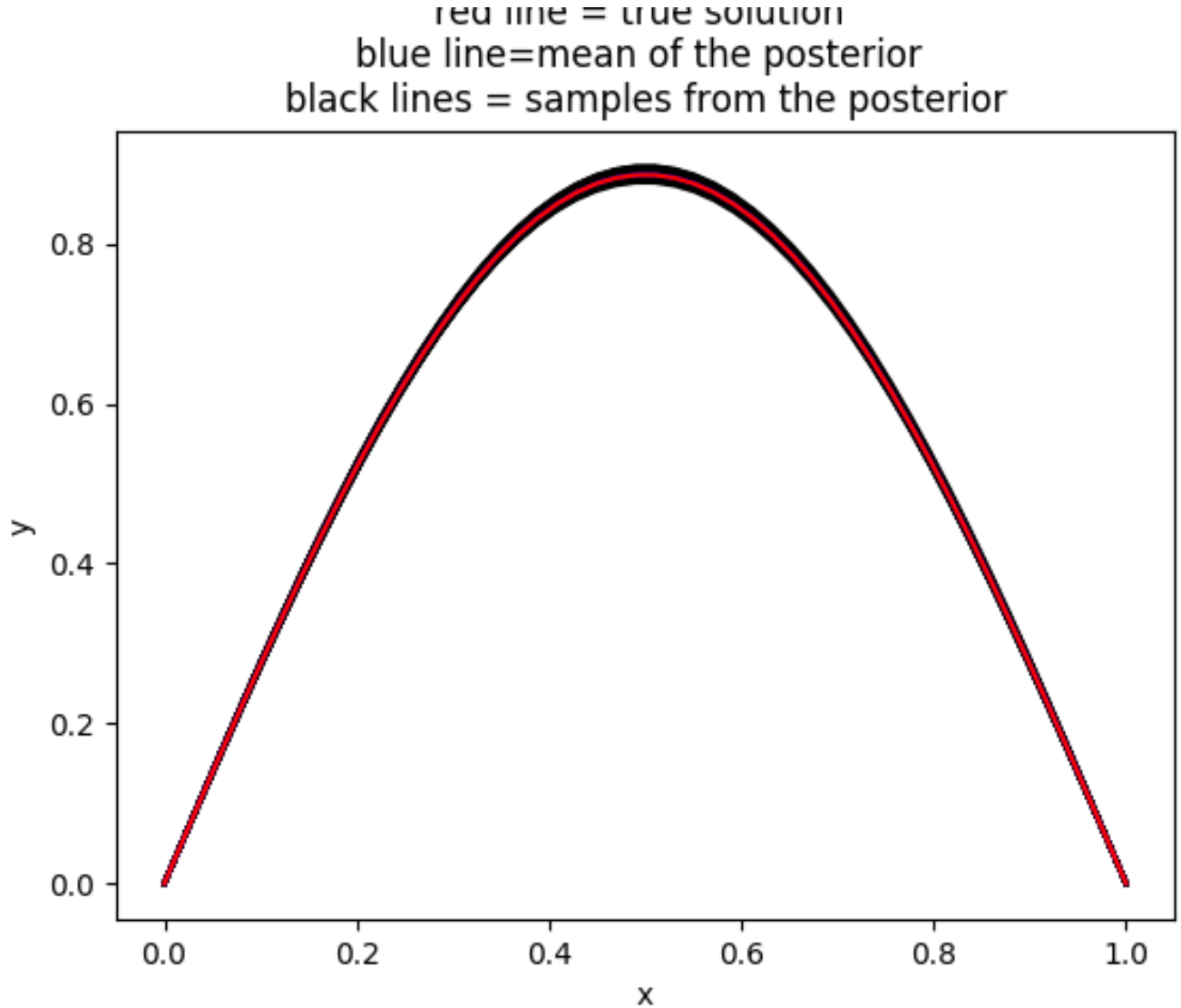
$$u(t, 0) = 0$$

$$u(t, 1) = 0$$

$$\alpha = 0.02$$

Via separation of variables (i.e. assume solution is of the form $u(t, x) = T(t)X(x)$, you can obtain the analytical solution

$$u^*(t, x) = \exp(-\alpha \pi^2 t)\sin(\pi x)$$



This is a snapshot of the solution at a single timepoint. The graph shows that the mean of the solution (blue line) estimated by our algorithm coincides basically exactly with the true solution (red line).

A separable kernel in the form $\Sigma((t,x),(t',x')) = K(t,t')C(x,x')$ was used. The time kernel was chosen to be Matern $3/2$ and the position kernel Matern $5/2$.

To access the performance of the algorithm, we look at two statistics. The first is the maximum error ('L infinity error') of the posterior mean,

$$\sup_{\substack{0 \leq t \leq T \\ x_{min} \leq x \leq x_{max}}} |\mathbb{E}[u(t,x)] - u^*(t,x)|$$

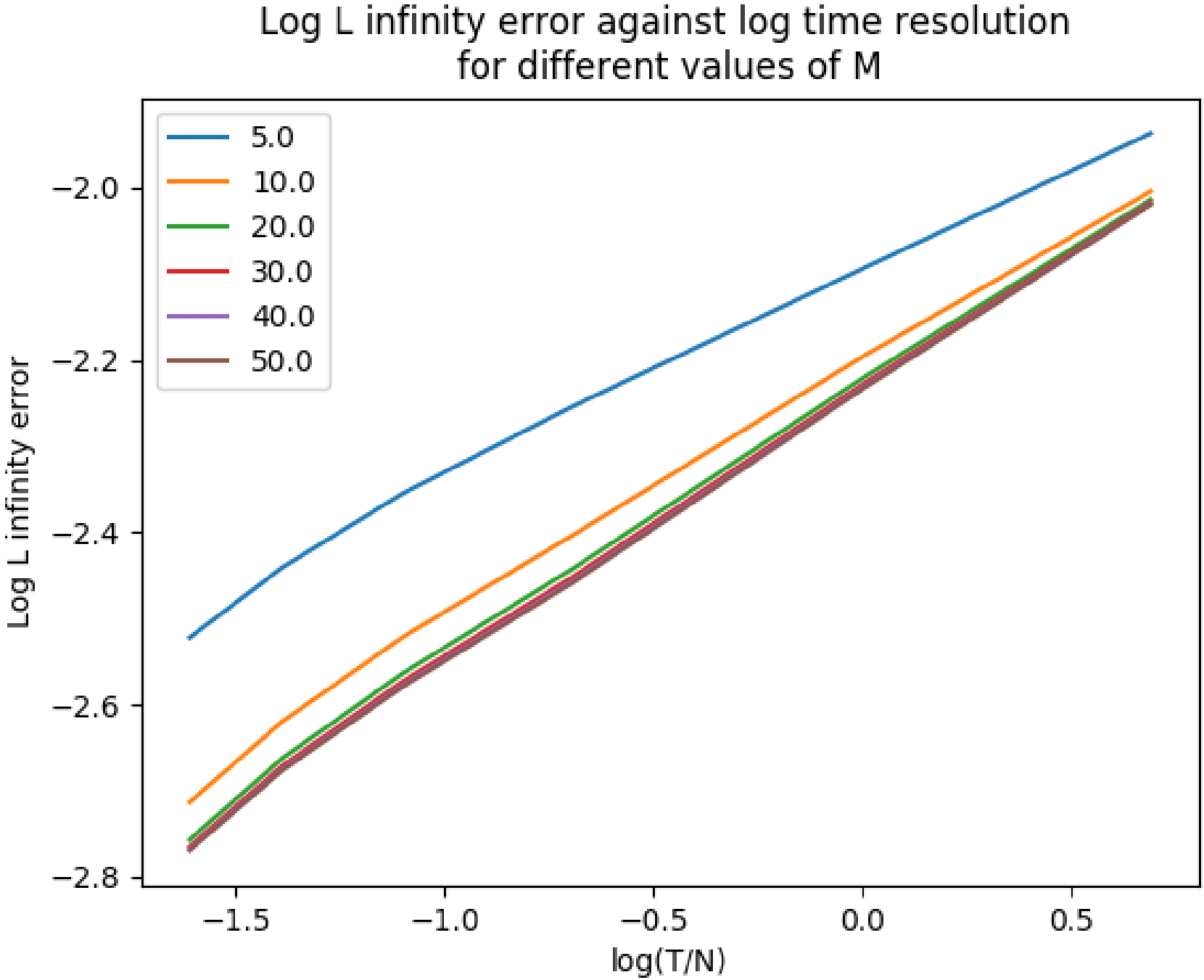
which is approximated by taking the maximum over the discrete $N \times M$ grid. This measures how accurate point estimation of the solution is.

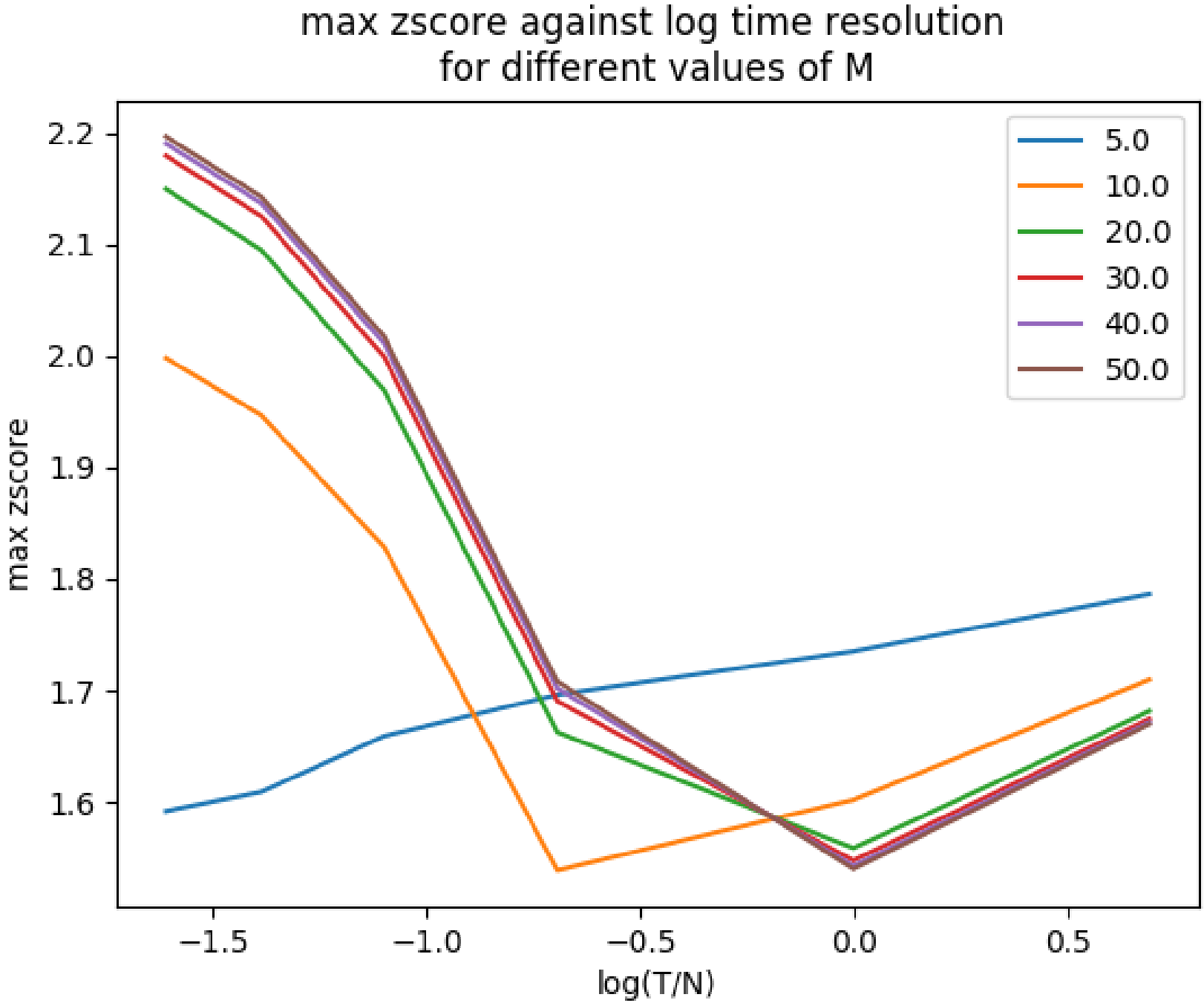
The second statistic, the 'z-score'

$$\sup_{\substack{0\leq t\leq T \\ x_{min}\leq x\leq x_{max}}} \frac{|\mathbb{E}[u(t,x)] - u^*(t,x)|}{\sqrt{\mathbb{V}[u(t,x)]}}$$

assesses the trustworthiness of the posterior spread as an estimate for the error in the posterior mean. This again is approximated by taking the maximum over the discrete $N \times M$ grid.

The subsequent slides will show for the earlier example as well as a nonlinear example, that the error decreases as the log resolution of time decreases (meaning the number of data points used to train the solution increases), and the z scores remain relatively constant at small values (meaning the posterior is sensible).


Log L infinity error against log time resolution
for different values of M
5.0
10.0
20.0
30.0
40.0
50.0
−2.0
−2.2
−2.4
−2.6
−2.8
Log L infinity error
−1.5
−1.0
−0.5
0.0
0.5
log(T/N)


max zscore against log time resolution
for different values of M
max zscore
log(T/N)
2.2
2.1
2.0
1.9
1.8
1.7
1.6
−1.5
−1.0
−0.5
0.0
0.5
5.0
10.0
20.0
30.0
40.0
50.0

Experimental nonlinear example:
Consider Burger's equation on $x \in [0, 1]$.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \alpha\frac{\partial^2 u}{\partial x^2} = 0 \tag{4}$$

With initial and boundary conditions:

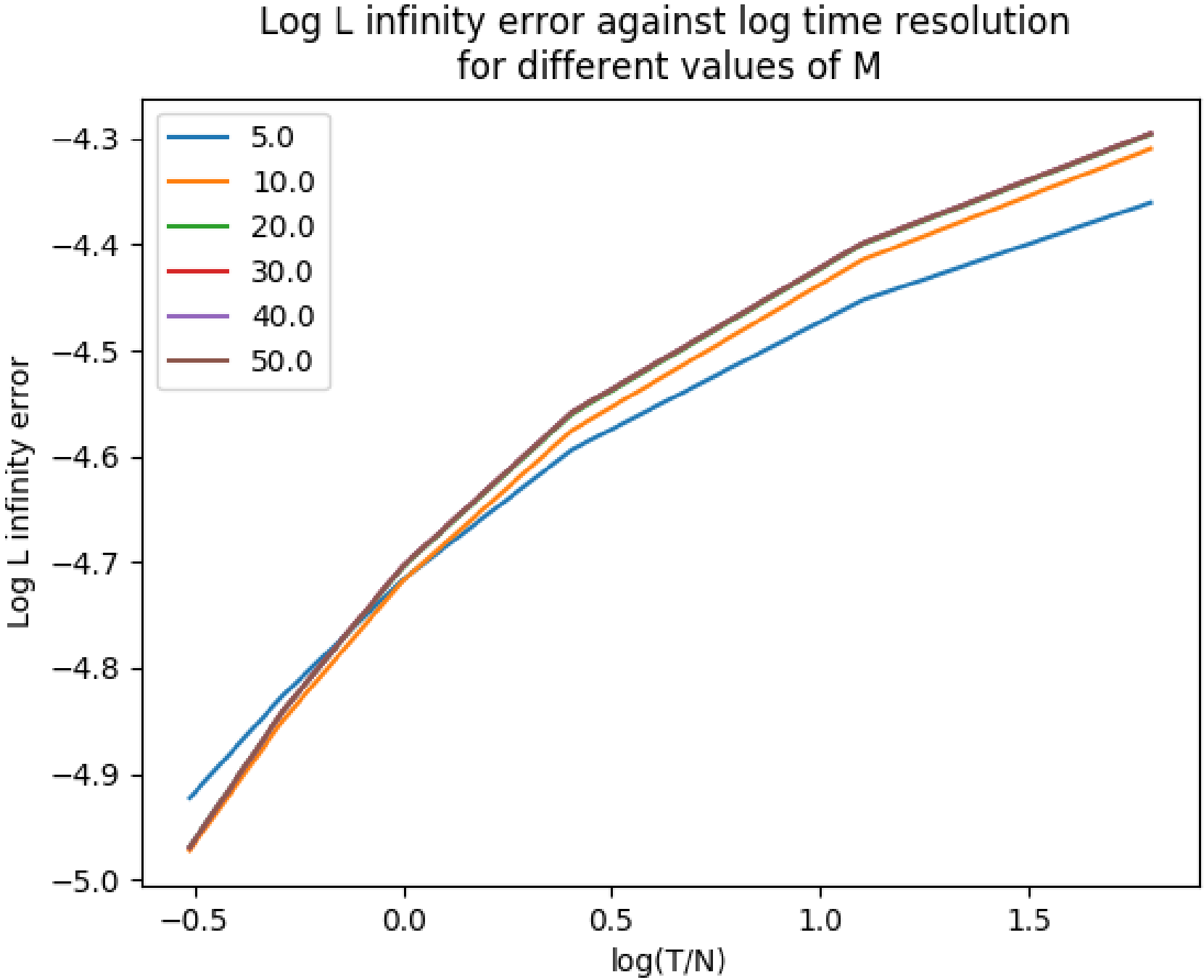
$$u(0, x) = 2\frac{\alpha \mathrm{aksin}(kx)}{\mathrm{b} + \mathrm{acos}(\mathrm{k}x)}$$
$$u(t, 0) = 0$$
$$u(t, 1) = 0$$

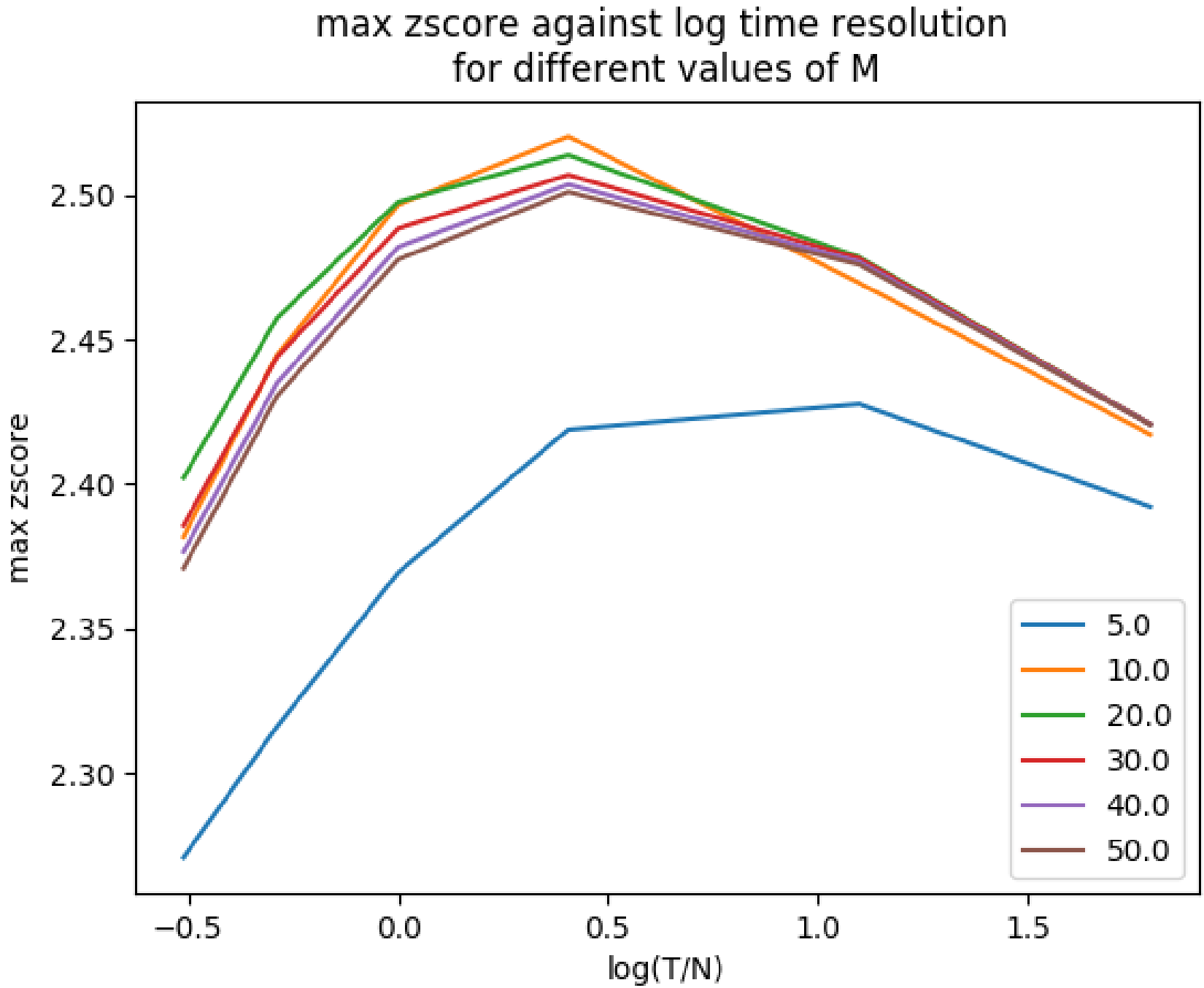
We take the parameters

$$\alpha = 0.02, \mathrm{a} = 1, \mathrm{b} = 2, k = 1$$

It can be shown the analytical solution is

$$u^*(t, x) = 2\frac{\alpha \mathrm{ak} \exp(-\alpha k^2 t)\sin(kx)}{\mathrm{b} + \mathrm{a}\exp(-\alpha k^2 t)\cos(\mathrm{k}x)}$$


Log L infinity error against log time resolution
for different values of M
5.0
10.0
20.0
30.0
40.0
50.0
−4.3
−4.4
−4.5
−4.6
−4.7
−4.8
−4.9
−5.0
Log L infinity error
−0.5
0.0
0.5
1.0
1.5
log(T/N)


max zscore against log time resolution
for different values of M
max zscore
2.50
2.45
2.40
2.35
2.30
−0.5
0.0
0.5
1.0
1.5
log(T/N)
5.0
10.0
20.0
30.0
40.0
50.0

References I