Tolerance vs. Error Results for a Class of Error Control B-spline Gaussian Collocation PDE Solvers

Paul Muir and Jack Pew

Abstract B-spline Gaussian collocation software for the numerical solution of partial differential equations has been widely used for several decades. BACOL and BACOLI are recently developed packages of this class that provide control of estimates of the temporal and spatial errors of the numerical solution through the use of adaptive time-stepping/method order selection and adaptive spatial mesh refinement. Several previous studies have investigated the performance of the BACOL and BACOLI packages with respect to work-accuracy, i.e., efficiency, measures. In this report, we investigate the reliability of the BACOL and BACOLI packages, focusing on the relationship between the requested tolerance and the accuracy achieved. In particular, we consider the effect, on the reliability of the software, of (i) the degree of the piecewise polynomials employed in the representation of the spatial dependence of the approximate solution, (ii) the type of spatial error control employed, and (iii) the type of spatial error estimate computed.

1 Introduction/Background

B-spline Gaussian collocation software for the numerical solution of ordinary and partial differential equations has been widely used for several decades. A characterizing feature of such software is the representation of the numerical solution as a linear combination of known B-spline [5] basis functions (piecewise polynomials of a given degree p) with unknown coefficients. The unknown B-spline coefficients are determined by solving collocation equations, obtained by requiring the approximate solution to satisfy (i) the differential equations at the images of the set of $p-1$

Paul Muir
Saint Mary's University, Halifax, NS, Canada, B3H 3C3 e-mail: muir@smu.ca

Jack Pew
Saint Mary's University, Halifax, NS, Canada, B3H 3C3 e-mail: jack.pew@gmail.com

Gauss points on each subinterval of a mesh that partitions the problem domain, and (ii) additional equations that depend on the boundary conditions.

A well known software package of this class for ODEs is the boundary value ODE solver COLSYS [3], which appeared about 35 years ago. The COLSYS package featured estimation and control of an estimate of the global error of the collocation solution; the adaptive control of the global error estimate with respect to a given user tolerance was obtained through the use of mesh refinement based on equidistribution of the error estimates for each mesh subinterval.

We next consider B-spline Gaussian collocation software for the numerical solution of PDEs. While there has been some development of software of this type for time-dependent PDEs in two spatial dimensions, see, e.g., [17] and [10] and references within, software of this class for time-dependent PDEs in one spatial dimension (1D) is at a more advanced state, and it is this case upon which we will focus in this report.

For 1D PDE systems, we assume that there is a spatial mesh, $\{x_i\}, i = 0, \ldots, NINT$, that partitions the spatial domain $[a,b]$, and then the approximate solution has the form

$$\underline{U}(x,t) = \sum_{i=1}^{NC} \underline{y}_i(t) B_i(x), \tag{1}$$

where $\underline{y}_i(t)$ is the time-dependent (vector) coefficient of the i-th B-spline basis function, $B_i(x)$, and $NC = NINT(p-1)+2$. The B-spline basis is chosen to have C^1-continuity and we recall that p is the user specified degree of the B-spline basis functions. Assuming a system of $NPDE$ PDEs of the form

$$\underline{u}_t(x,t) = \underline{f}\left(t,x,\underline{u}(x,t),\underline{u}_x(x,t),\underline{u}_{xx}(x,t)\right), \qquad a \le x \le b, \quad t \ge t_0, \tag{2}$$

with initial conditions

$$\underline{u}(x,t_0) = \underline{u}_0(x), \qquad a \le x \le b, \tag{3}$$

and boundary conditions

$$\underline{b}_L\left(t,\underline{u}(a,t),\underline{u}_x(a,t)\right) = \underline{0}, \qquad \underline{b}_R\left(t,\underline{u}(b,t),\underline{u}_x(b,t)\right) = \underline{0}, \quad t \ge t_0, \tag{4}$$

the collocation conditions are represented by a system of ODEs the form

$$\underline{U}_t(\xi_l,t) = \underline{f}\left(t,\xi_l,\underline{U}(\xi_l,t),\underline{U}_x(\xi_l,t),\underline{U}_{xx}(\xi_l,t)\right), \tag{5}$$

for $l = 2, \ldots, NC-1$. The collocation points are $\xi_l = x_{i-1} + h_i\rho_j, l = 1+(i-1)(p-1)+j, i = 1, \ldots, NINT, j = 1, \ldots, p-1$, $\{\rho_i\}_{i=1}^{p-1}$, are the images of the $p-1$ Gauss points on $[0,1]$ and $h_i = x_i - x_{i-1}$. (Thus the number of collocation points per subinterval, $kcol = p-1$.) The points $\xi_1 = a$ and $\xi_{NC_p} = b$ are associated with requiring $\underline{U}(x,t)$ to satisfy the boundary conditions:

$$\underline{b}_L\left(t,\underline{U}(a,t),\underline{U}_x(a,t)\right) = 0, \qquad \underline{b}_R\left(t,\underline{U}(b,t),\underline{U}_x(b,t)\right) = 0. \tag{6}$$

About 35 years ago, the B-spline Gaussian collocation PDE solver, PDECOL [11] appeared. The B-spline basis was implemented using the de Boor B-spline package [5]. The boundary conditions were differentiated with respect to time in order to obtain ODEs that could be coupled with the ODEs representing the collocation conditions (5). The ODE solver, GEARIB [7], was used to treat this ODE system, and thus temporal error control was provided by this solver. PDECOL used a fixed spatial mesh which meant that no control of the spatial error was provided. The major computational cost associated with PDECOL was the numerical solution of the linear systems that arose during the computation. The coefficient matrices associated with these linear systems had non-zeros clustered near the main diagonal and in PDECOL these systems were treated with a band matrix solver.

About a decade later, a modification of PDECOL called EPDCOL [9] was released; EPDCOL replaced the band matrix solver employed by PDECOL with an almost block diagonal (ABD) solver, COLROW [6], that was able to take into account in an efficient manner the structure of the linear systems arising from the B-spline collocation process. Savings in execution time of about 50% were obtained. Both PDECOL and EPDCOL have been widely used in the numerical solution of 1D PDEs.

The next update to this family of PDE solvers, BACOL [19], released a little over a decade later, was a new implementation of the B-spline Gaussian collocation algorithm (rather than a modification of PDECOL or EPDCOL) with several additional features. BACOL employed a Differential-Algebraic Equation (DAE) solver, DASSL [4], so that the ODEs (5) could be solved together with the algebraic equations representing the boundary conditions (4). (That is, the boundary conditions could be treated directly.) Temporal error control was provided by DASSL. The most significant new feature of BACOL was the implementation of spatial error estimation and control. The spatial error estimates were obtained by computing a second (higher spatial order) collocation solution, $\underline{\bar{U}}(x,t)$, based on B-splines of degree $p+1$, which was then used together with $\underline{U}(x,t)$ to compute a set of $NPDE$ spatial error estimates of the form,

$$E_s(t) = \sqrt{\int_a^b \left(\frac{U_s(x,t) - \bar{U}_s(x,t)}{ATOL_s + RTOL_s |U_s(x,t)|} \right)^2 dx}, \quad s = 1, \ldots, NPDE, \tag{7}$$

where $U_s(x,t)$ is the sth component of $\underline{U}(x,t)$, $\bar{U}_s(x,t)$ is the sth component of $\underline{\bar{U}}(x,t)$, and $ATOL_s$ and $RTOL_s$ are absolute and relative tolerances corresponding to the sth component of the solution. At the end of *each* time step, t, the collocation solution, $\underline{U}(x,t)$ was accepted if

$$\max_{1 \le s \le NPDE} E_s(t) \le 1. \tag{8}$$

Otherwise, the time step was rejected and BACOL computed a second set of spatial error estimates of the form,

$$\hat{E}_i(t) = \sqrt{\sum_{s=1}^{NPDE} \int_{x_{i-1}}^{x_i} \left(\frac{U_s(x,t) - \bar{U}_s(x,t)}{ATOL_s + RTOL_s |U_s(x,t)|} \right)^2 dx}, \quad i = 1, \ldots, NINT, \quad (9)$$

which were used within an adaptive mesh refinement (AMR) process. (We note that the controlled error estimates, (7), are slightly different from the error estimates, (9), that drive the adaptation process.) The goal of the AMR process was to determine a spatial mesh for the current step such that the estimated spatial errors were (i) approximately equidistributed over the mesh subintervals and (ii) satisfied the user tolerance requirement (8); see [21] for further details. BACOL was shown to have superior performance compared to a number of comparable packages, especially for problems with sharp moving layers and higher accuracy requirements [20]. We note however that the computation of the higher order collocation solution, $\underline{\bar{U}}(x,t)$, represented a substantial cost within BACOL, essentially doubling the cost of the computation.

In BACOL, the spatial error estimate was used to control the error in $\underline{U}(x,t)$, the approximate solution returned by the code. We refer to this as *Standard (ST) Error Control.* A minor modification of BACOL would allow it to return the higher order collocation solution, $\underline{\bar{U}}(x,t)$, and in this case, the computation of the returned solution would be controlled based on an error estimate for a collocation solution that is of one lower order than the returned solution. We refer to this as *Local Extrapolation (LE) error control.* (These error control modes are typically associated with software for initial value ODEs, and in particular codes based on Runge-Kutta formula pairs; see, e.g., [8].)

The most recent update to this family of solvers is a modification of BACOL known as BACOLI [12]. The key feature of BACOLI is the avoidance of the expensive computation of the second collocation solution, $\underline{\bar{U}}(x,t)$. BACOLI instead computes only one collocation solution and one low cost interpolant which is then used in the computation of a spatial error estimate. There are two options available for the interpolant. One involves a superconvergent interpolant (SCI) [1] which is based on the presence of certain points within the spatial domain where $\underline{U}(x,t)$ is of at least one higher order of accuracy, i.e., superconvergent. The SCI is of the same order of accuracy as $\underline{\bar{U}}(x,t)$ and leads to a spatial error estimate for $\underline{U}(x,t)$, similar to the situation in BACOL. This option therefore provides ST Error Control. The other interpolant option involves a lower order interpolant (LOI) [2], of one order of accuracy lower than $\underline{U}(x,t)$. The LOI is based on interpolation of $\underline{U}(x,t)$ at certain points such that the interpolation error of the LOI agrees asymptotically with the error of a collocation solution of one lower order than $\underline{U}(x,t)$. This option therefore provides LE Error Control. BACOLI, using either error estimation/control option, has been shown to be about twice as fast as BACOL [12].

When considering the effectiveness of a numerical software package, the most common analysis involves work-precision (i.e., work-accuracy) diagrams that show, for a given solver applied to a given test problem, the relationship between the CPU time and the accuracy for that solver. A number of such studies involving BACOL and BACOLI have been performed; see, [20], [12].

However, another important measure of the quality of a software package is its reliability, and a key component of this measure involves assessing the relationship between the tolerance requested and the accuracy achieved by the solver on a given test problem. This depends on the quality of the error estimates and the algorithms that adapt the computation to attempt to control the error estimates with respect to the user specified tolerance.

The purpose of this report is to present an experimental analysis of the reliability of the error control PDE solvers BACOL and BACOLI applied to some standard test problems chosen from the literature. (Some preliminary results of this type for BACOL were reported in [18].) In particular, we are interested in how the choice of the degree of the B-spline basis and the choice of error control mode (ST or LE) impact on the reliability of the solvers. We also examine the effectiveness of the spatial error estimates provided by the SCI and LOI schemes in BACOLI, compared with the spatial error estimation scheme used in BACOL.

In the next section, we identify the test problems to be considered and then provide numerical results that allow us to explore the relationship between the requested tolerance and the achieved error for the BACOL and BACOLI packages. The final section provides our conclusions and suggestions for future work.

2 Numerical Results

In this section we present numerical results for the four codes identified in the previous section:

- BACOL with ST control: **BAC/ST**
- BACOL with LE error control: **BAC/LE**
- BACOLI with the SCI scheme/ST error control: **SCI/ST**
- BACOLI with the LOI scheme/LE error control: **LOI/LE**

We consider the One Layer Burgers Equation (**OLBE**) with $\varepsilon = 10^{-3}$ and 10^{-4}:

$$u_t = \varepsilon u_{xx} - uu_x, \tag{10}$$

with boundary conditions at $x = 0$ and $x = 1$ ($t > 0$) and an initial condition at $t = 0$ ($0 \le x \le 1$) taken from the exact solution is

$$u(x,t) = \frac{1}{2} - \frac{1}{2}\tanh\left(\frac{x - \frac{t}{2} - \frac{1}{4}}{4\varepsilon}\right).$$

The solution has a sharp layer region around $x = 0.25$ when $t = 0$. As t goes from 0 to 1, the layer moves to the right and is located around $x = 0.75$ when $t = 1$. A plot of the solution for $\varepsilon = 10^{-4}$ is given in [12].

We also consider the Two Layer Burgers Equation (**TLBE**) with $\varepsilon = 10^{-3}$ and 10^{-4}; this problem has the same PDE (10) but the boundary conditions at $x = 0$ and

$x = 1$ ($t > 0$) and an initial condition at $t = 0$ ($0 \le x \le 1$) are taken from the exact solution is

$$u(x,t) = \frac{0.1e^{-A} + 0.5e^{-B} + e^{-C}}{e^{-A} + e^{-B} + e^{-C}},$$

where,

$$A = \frac{0.05}{\varepsilon}(x - 0.5 + 4.95t), \quad B = \frac{0.25}{\varepsilon}(x - 0.5 + 0.75t), \quad C = \frac{0.5}{\varepsilon}(x - 0.375).$$

When $t = 0$, the solution has two sharp layers around $x = 0.25$ and $x = 0.5$. As t increases, these layers move to the right and merge, forming a single layer around $x = 0.7$ when $t \approx 0.5$. As time increases further, the single layer continues to move to the right, and is located around $x = 0.9$ when $t = 1$. A plot of the solution for $\varepsilon = 10^{-4}$ is given in [12].

We apply each of the four codes identified above to the OLBE and TLBE, with $\varepsilon = 10^{-3}$ and 10^{-4}, for a range of *kcol* values and for 91 tolerance values (with $ATOL_1 = RTOL_1$) uniformly distributed from 10^{-1} to 10^{-10}. (That is the tolerance values are $10^{-1}, 10^{-1.1}, 10^{-1.2}, \ldots, 10^{-2}, 10^{-2.1}, 10^{-2.2}, \ldots, 10^{-3}, \ldots, 10^{-4}, \ldots, 10^{-10}$.) For each experiment, we report the L^2-norm of the error at the final time, $T_{out} = 1$. This error has the form

$$\sqrt{\int_a^b (U_s(x,t) - V_s(x,t))^2 \, dx}, \quad s = 1, \ldots, NPDE, \tag{11}$$

where $U_s(x,t)$ is the sth component of the collocation solution, $\underline{U}(x,t)$, and $V_s(x,t)$ is the sth component of the exact solution, $\underline{V}(x,t)$. This integral is computed using a high order quadrature rule applied to each subinterval of the spatial mesh that is used on the final time step. (We note that the error (11) that is assessed is different in form from either of the error estimates, (7), (9).)

2.1 Reliability by Code

In this subsection we consider the reliability of each code (with respect to tolerance vs. error) over the four test problems identified above, for $kcol = 3, 5, 7, 9$, and over the range of tolerances identified above. The results for the BAC/ST code are given in Figures 1(a) - 16(a), and show good correlation between the requested tolerance and the achieved accuracy. The points are clustered around the reference line corresponding to equal values of the tolerance and error. A closer look at the relationship between the error and the tolerance can be obtained if we plot tolerance vs. ETR, where ETR is the error tolerance ratio, i.e., error divided by the tolerance. Figures 1(b) - 16(b) gives plots of the tolerance vs. the ETR. (We plot a reference line (in green) which is the horizontal line at $10^0 = 1$.) From these figures we see that the ETR varies over the range of tolerance values, with some errors greater than

the tolerance and some less than the tolerance. In all cases the maximum ETR is within an order of magnitude of the tolerance. The minimum ETR is also typically within an order of magnitude of the tolerance, although in some cases it is smaller by as much as two orders of magnitude. The second horizontal line (in red) in these plots corresponds to the mean ETR value over the range of tolerances. For the less difficult problems with $\varepsilon = 10^{-3}$, the mean ETR value is always less than the tolerance. For the more challenging instances of the two problems, with $\varepsilon = 10^{-4}$, we see that in all cases the mean is within a small multiple of the tolerance, no larger than about 3.

We note also that it is generally at coarse tolerances that we see the largest ETR values for a given problem and *kcol* value.

An examination of Figures 17 - 32 for the BAC/LE code, Figures 33 - 48 for the SCI/ST code, and Figures 49 - 64 for the LOI/LE code show generally similar results with the exception of Figures 37 and 45, corresponding to the SCI/ST code with $kcol = 3$ applied to the more challenging instances of the problems with $\varepsilon = 10^{-4}$. For those cases, we see that the SCI/ST code performs poorly; the maximum and mean ETR values are both substantially larger than the tolerance. We also note that in these plots there gaps where no tolerance vs. ETR values are plotted; these gaps correspond to cases where the code has failed. (The assessment of the failures of the codes is considered in more detail in [12].)

Table 1 gives a summary of the ETR results for the four codes over all problem, ε, and *kcol* values (with the exclusion of the $kcol = 3$, $\varepsilon = 10^{-4}$ results for the SCI/ST code.)

Code	Maximum	Mean	Minimum
BAC/ST	7.95	1.09	0.01
BAC/LE	7.73	1.01	0.01
SCI/ST	6.85	0.87	0.01
LOI/LE	5.99	0.74	0.01

Table 1: *Overall maximum, mean, and minimum ETR values for each code over both test problems, both ε values, and for $kcol = 3,5,7,9$. Results for SCI/ST with $kcol = 3$ for the instances of the problems that employ $\varepsilon = 10^{-4}$ are excluded.*

2.2 Reliability vs. *kcol*

In this subsection, we consider, for the each of the codes, the dependence of the relationship between the error and the tolerance on the value of *kcol*. From Figures 1 - 16, for the BAC/ST code, we can observe that as *kcol* increases there is a decrease in the number of instances for which the error is greater than the tolerance. The mean ETR values consistently decrease as *kcol* increases.

Similar observations can be made from Figures 17 - 32 for the BAC/LE code, Figures 33 - 48 for the SCI/ST code, and Figures 49 - 64 for the LOI/LE code.

We can conclude therefore that the reliability of the codes tends to be better for larger kcol values.

Table 2 gives a summary of the ETR results for the four *kcol* values over all problem, ε, and code combinations (with the exclusion of the $kcol = 3$, $\varepsilon = 10^{-4}$ results for the SCI/ST code.)

kcol	Maximum	Mean	Minimum
3	7.95	1.31	0.01
5	6.85	1.04	0.01
7	6.60	0.77	0.01
9	6.75	0.67	0.01

Table 2: *Overall maximum, mean, and minimum ETR values for each kcol value over both test problems, both ε values, and the four codes. Results for SCI/ST with $kcol = 3$ for the instances of the problems that employ $\varepsilon = 10^{-4}$ are excluded.*

2.3 Reliability vs. Spatial Error Control Mode

In this subsection we consider the BAC/ST and BAC/LE codes since the error control mode (ST vs. LE) is the only difference between these codes. Comparing Figures 1 - 16 for BAC/ST with the corresponding Figures 17 - 32 for BAC/LE we can see that the two codes appear to have quite similar performance. Comparing the maximum, mean, and minimum ETR values for the BAC/ST and BAC/LE codes in Table 1, we see that the two codes give very similar results.

We can thus conclude that the error control mode does not have a significant influence on the reliability.

2.4 Reliability vs. Spatial Error Estimation Scheme

In this section we compare BAC/ST with SCI/ST and BAC/LE with LOI/LE. The first pair of codes both employ the ST error control mode but differ in the type of error estimate they compute; the latter pair of codes both employ the LE error control mode but again differ in the type of error estimate they use. From a comparison of the figures for the BAC/ST and SCI/ST codes (excluding the $kcol = 3$ cases mentioned previously where the SCI/ST code exhibits inferior performance), we see that the codes have comparable reliability. Similarly a comparison of the figures for the BAC/LE and LOI/LE codes shows that these codes also have comparable reliabil-

ity. Comparing the maximum, mean, and minimum ETR values for the BAC/ST and SCI/ST codes and the corresponding values for the BAC/LE and LOI/LE codes in Table 1, we see that the pairs of codes give very similar results.

This suggests that the spatial error estimates computed by the SCI and LOI schemes are generally of comparable quality to the original spatial error estimates computed by the BACOL code.

3 Discussion/Conclusions/Future Work

Several conclusions can be drawn from the results presented in the previous section:

- The codes considered in this report appear to be reliable; the error is generally well correlated with the requested tolerance. We see that in almost all cases the error is within the correct order of magnitude of the requested tolerance. *Furthermore the error is, on average, a small multiple of the requested tolerance.* (The only exception is for the SCI/ST, $kcol = 3$ case, where issues associated with the quality of its error estimates have already been observed - see [12].)
- Sometimes the error is actually as much as two orders of magnitude less than the tolerance. While it is desirable for the error to be about the same size as or even slightly less than the tolerance, if the error is too much less than the tolerance this can lead to inefficiency in the computation.
- The degree p of the B-spline basis as specified by the choice of $kcol$ ($p = kcol + 1$) *does* appear to have an impact on the reliability. A comparison of the results for the $kcol = 3, 5, 7,$ and 9 cases, shows that as $kcol$ is increased, the reliability of the codes tends to increase; that is, the error is, on average, closer to and in some cases even less than the tolerance for the larger $kcol$ values.
- The relationship between the error and tolerance for each of the two versions of BACOL that employ different error control modes (ST vs. LE) is quite similar. This suggests that the choice of error control mode does not have a strong impact on the reliability of the solvers.
- The results show that the new SCI and LOI spatial error estimation schemes are of comparable quality to the original spatial error estimation scheme employed by BACOL.

While the entire family of software packages discussed in this report (PDECOL, EPDCOL, BACOL, and BACOLI) leave $kcol$ as a parameter that must be chosen by the user, it is in fact not straightforward for the user to make an appropriate choice of this parameter. The correlation between higher $kcol$ values and improved reliability that can be seen from the results presented in this report, as well as earlier results investigating the relationship between the choice of $kcol$ and the efficiency of the solvers [12], will provide a basis for further analysis that will allow us to develop a new release of BACOLI in which the code itself will choose $kcol$ appropriately to improve the efficiency and reliability of the computations performed by BACOLI.

Two other issues that need to be examined further are (i) the presence of a substantial number of cases where larger errors are associated with coarser tolerances and (ii) the significant number of results where the error is more than an order of magnitude less than the tolerance.

The results presented in this report consider the error across the spatial domain but only at the end of the computation when $t = T_{out}$. Thus an extension of the analysis considered in this report would be to consider some measure of the error over the entire computation across both time and space. This would give a more extensive assessment of the error over the entire computation. Also, as noted in several places in this report, another extension of the analysis would be to consider modifications to the code and the error computations that would have the form of the error used to assess the codes be the same as the form of the error estimates computed by the codes. It might also be worthwhile to modify the codes to have the same error estimates used for control and spatial mesh adaptation.

Since BACOL and BACOLI employ DASSL to perform the computations associated with determining the time-dependent B-spline coefficients, the overall reliability of the computations performed by these codes must be dependent on the reliability of DASSL (unless one were to employ DASSL with a much smaller tolerance than the user specified tolerance, leading to a significant degradation in efficiency). An example of the relationship between the tolerance and the error for DASSL can be seen in Figure 1 of [16]; it shows generally good correlation between the error and the tolerance, but with variations in the ETR of approximately one order of magnitude, either above or below the tolerance, similar to the results reported here for BACOL and BACOLI.

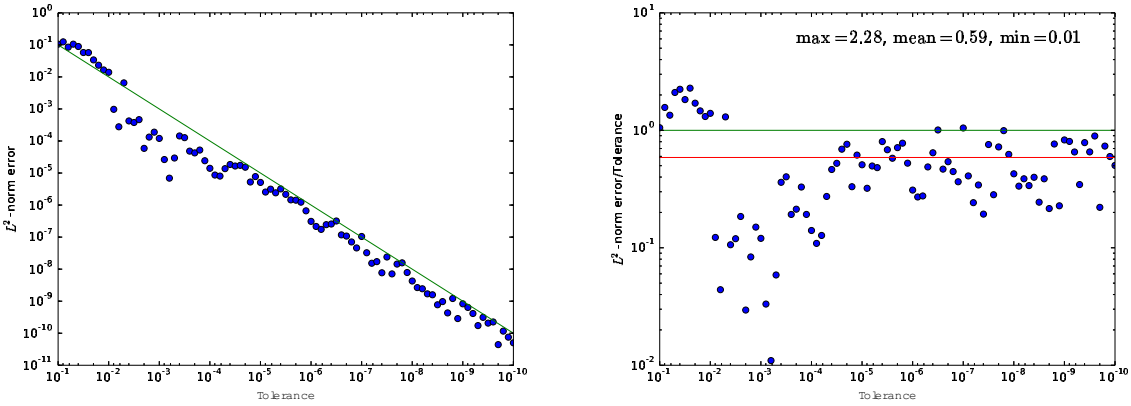
For DASSL and the codes considered in this report, the relationship between the tolerance and the error is not particularly smooth. That is, while the correlation between a given tolerance and the corresponding error is generally good (i.e., the error is, on average, within a small multiple of the tolerance), a small change in the tolerance does not necessarily lead to a similar small change in the error. This is related to what is referred to as the "conditioning" or "stability" of the software [16]. The papers [14], [15], and [13] describe how control theoretic techniques can be employed within ODE software to provide an improved condition number for the software. We plan to introduce these type of techniques into the BACOL and BACOLI packages.

Although the results considered here have focused on B-spline Gaussian collocation software for 1D PDEs, the approach for the 2D case considered in [10] very much builds on the 1D case, and we therefore expect the results of this report to be relevant for the numerical treatment of 2D PDEs.

References

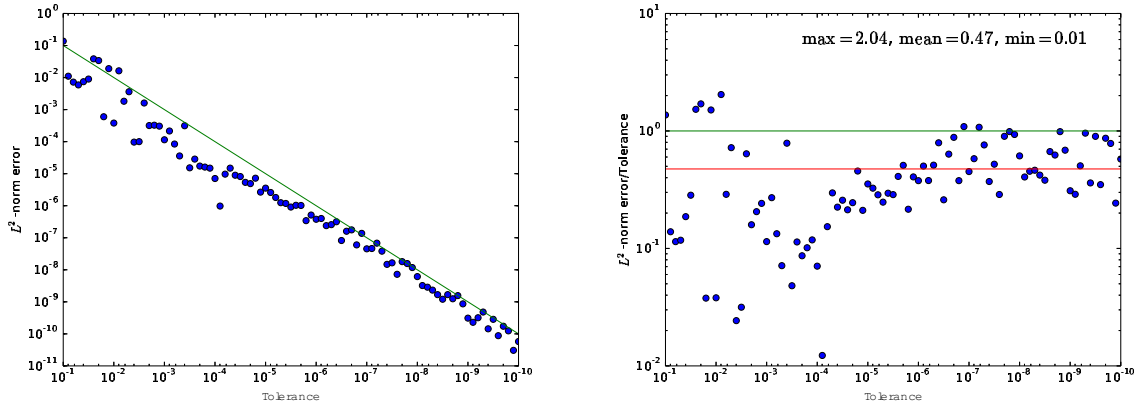
1. T. Arsenault, T. Smith, and P.H. Muir. Superconvergent interpolants for efficient spatial error estimation in 1D PDE collocation solvers. *Can. Appl. Math. Q.*, 17:409–431, 2009.

2. T. Arsenault, T. Smith, P.H. Muir, and J. Pew. Asymptotically correct interpolation-based spatial error estimation for 1D PDE solvers. *Can. Appl. Math. Q.*, 20:307–328, 2012.
3. U.M. Ascher, J. Christiansen, and R.D. Russell. Collocation software for boundary value odes. *ACM Trans. Math. Softw.*, 7:209–222, 1981.
4. K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1989.
5. C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1978.
6. J.C. Díaz, G. Fairweather, and P. Keast. Algorithm 603. COLROW and ARCECO: FORTRAN packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination. *ACM Trans. Math. Software*, 9(3):376–380, 1983.
7. C. William Gear. *Numerical initial value problems in ordinary differential equations*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1971.
8. E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.
9. P. Keast and P.H. Muir. Algorithm 688: EPDCOL: a more efficient PDECOL code. *ACM Trans. Math. Softw.*, 17(2):153–166, 1991.
10. Z. Li and P.H. Muir. B-spline Gaussian collocation software for two-dimensional parabolic PDEs. *Adv. Appl. Math. Mech.*, 5:528–547, 2013.
11. N.K. Madsen and R.F. Sincovec. Algorithm 540: PDECOL, general collocation software for partial differential equations. *ACM Trans. Math. Softw.*, 5(3):326–351, 1979.
12. J. Pew, Z. Li, and P.H. Muir. A computational study of the efficiency of collocation software for 1D parabolic PDEs with interpolation-based spatial error estimation. *Saint Mary's University, Dept. of Mathematics and Computing Science Technical Report Series,http://cs.smu.ca/tech_reports*, 2013.
13. Gernot Pulverer, Gustaf Söderlind, and Ewa Weinmüller. Automatic grid control in adaptive BVP solvers. *Numer. Algorithms*, 56(1):61–92, 2011.
14. Gustaf Söderlind. Digital filters in adaptive time-stepping. *ACM Trans. Math. Software*, 29(1):1–26, 2003.
15. Gustaf Söderlind and Lina Wang. Adaptive time-stepping and computational stability. *J. Comput. Appl. Math.*, 185(2):225–243, 2006.
16. Gustaf Söderlind and Lina Wang. Evaluating numerical ODE/DAE methods, algorithms and software. *J. Comput. Appl. Math.*, 185(2):244–260, 2006.
17. W. Sun, *B-spline collocation methods for elasticity problems*, in Scientific Computing and Applications, Adv. Comput. Theory Pract., 7, Kananaskis, 2000, 133–141.
18. R. Wang. High order adaptive collocation software for 1-D parabolic PDEs. *Ph.D. Thesis, Dalhousie University*, 2002.
19. R. Wang, P. Keast, and P.H. Muir. BACOL: B-spline Adaptive COLlocation software for 1D parabolic PDEs. *ACM Trans. Math. Software*, 30(4):454–470, 2004.
20. R. Wang, P. Keast, and P.H. Muir. A comparison of adaptive software for 1D parabolic PDEs. *J. Comput. Appl. Math.*, 169(1):127–150, 2004.
21. R. Wang, P. Keast, and P.H. Muir. A high-order global spatially adaptive collocation method for 1-D parabolic PDEs. *Appl. Numer. Math.*, 50(2):239–260, 2004.



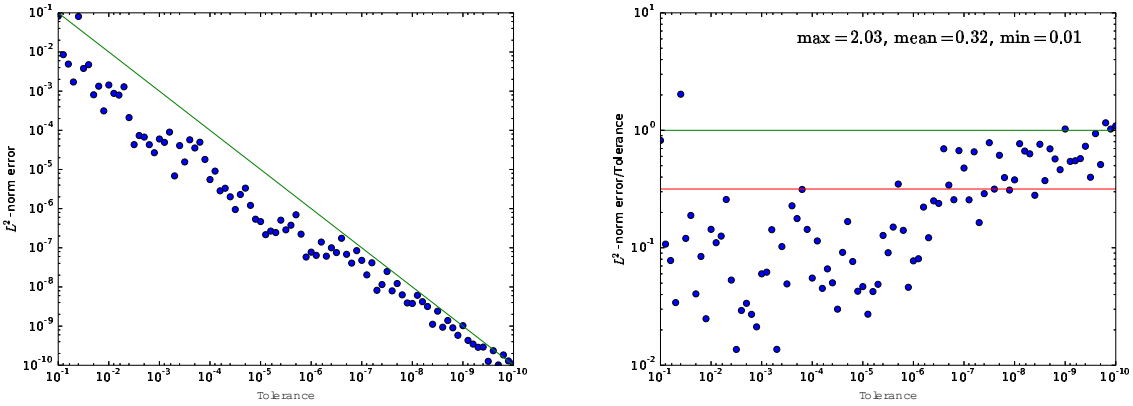
(a) (b)

Fig. 1: BAC/ST, $kcol = 3$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



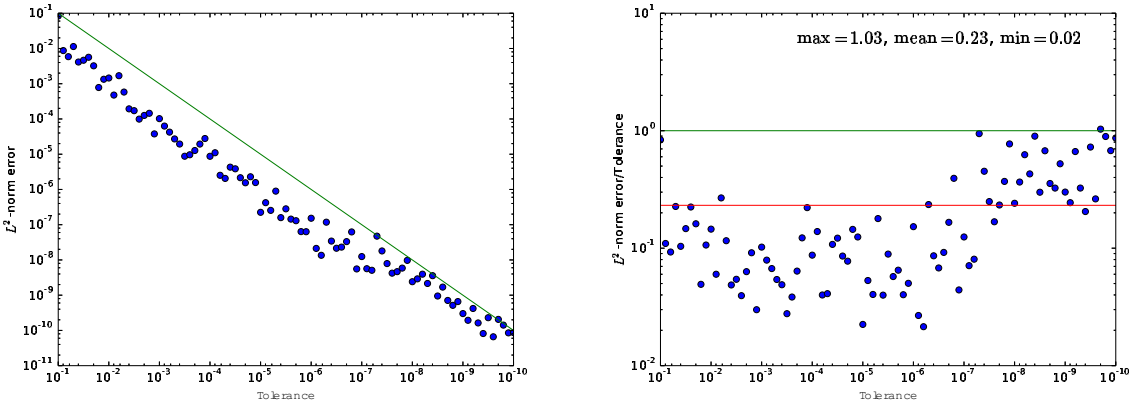
(a) (b)

Fig. 2: BAC/ST, $kcol = 5$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



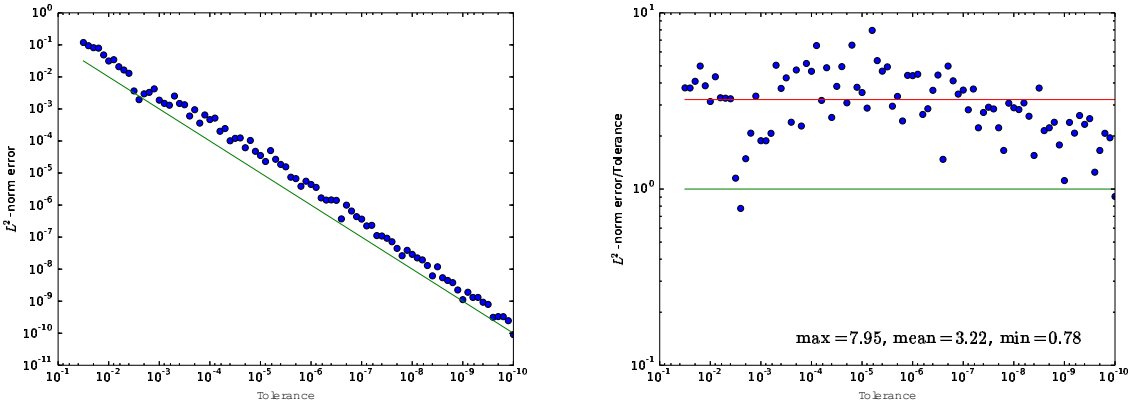
(a) (b)

Fig. 3: BAC/ST, $kcol = 7$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



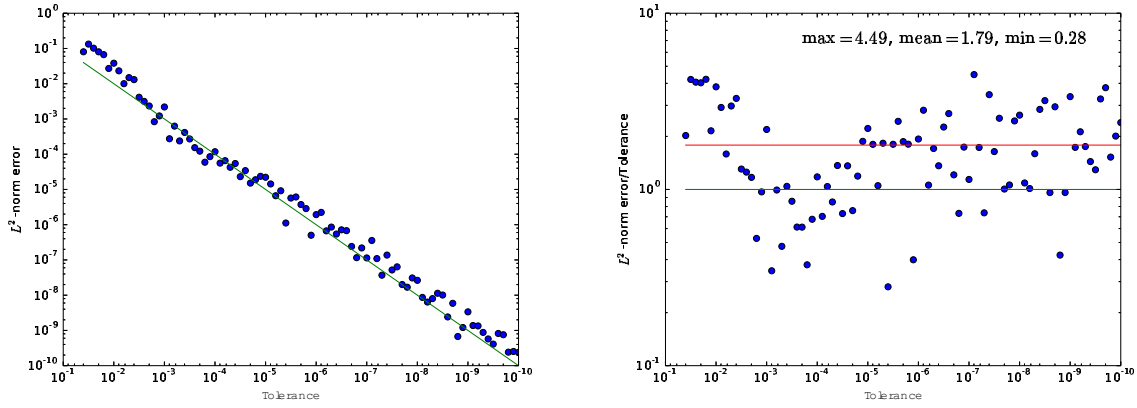
(a) (b)

Fig. 4: BAC/ST, $kcol = 9$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



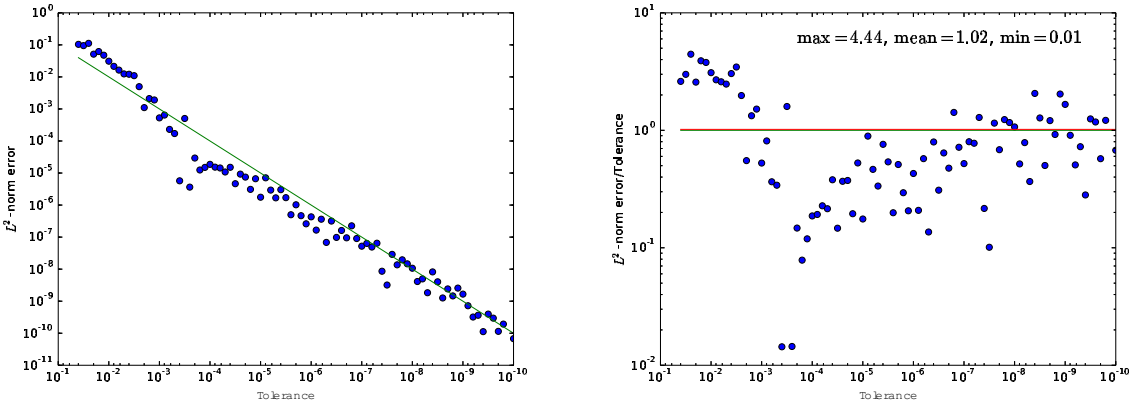
(a) (b)

Fig. 5: BAC/ST, $kcol = 3$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



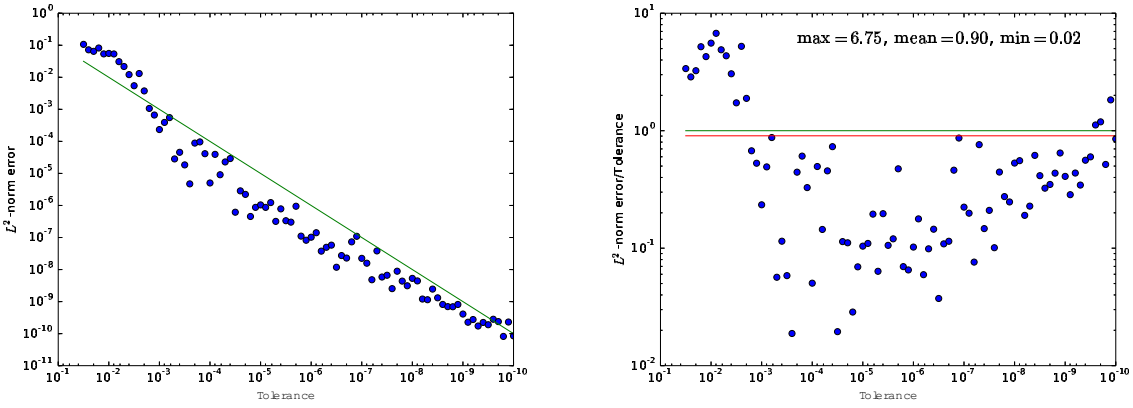
(a) (b)

Fig. 6: BAC/ST, $kcol = 5$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



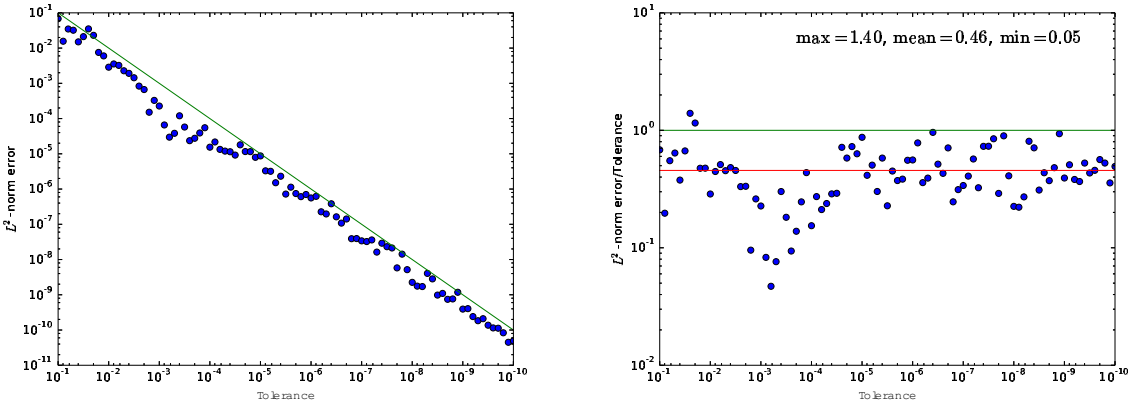
(a) (b)

Fig. 7: BAC/ST, $kcol = 7$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



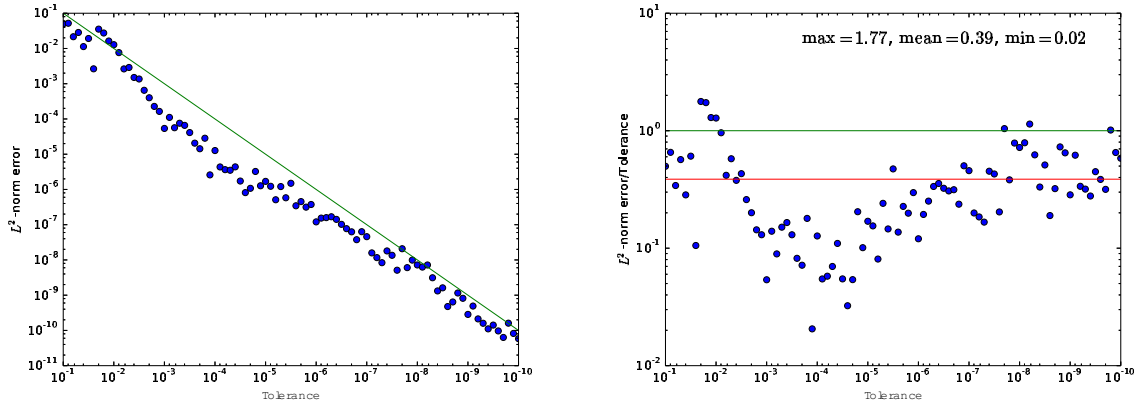
(a) (b)

Fig. 8: BAC/ST, $kcol = 9$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



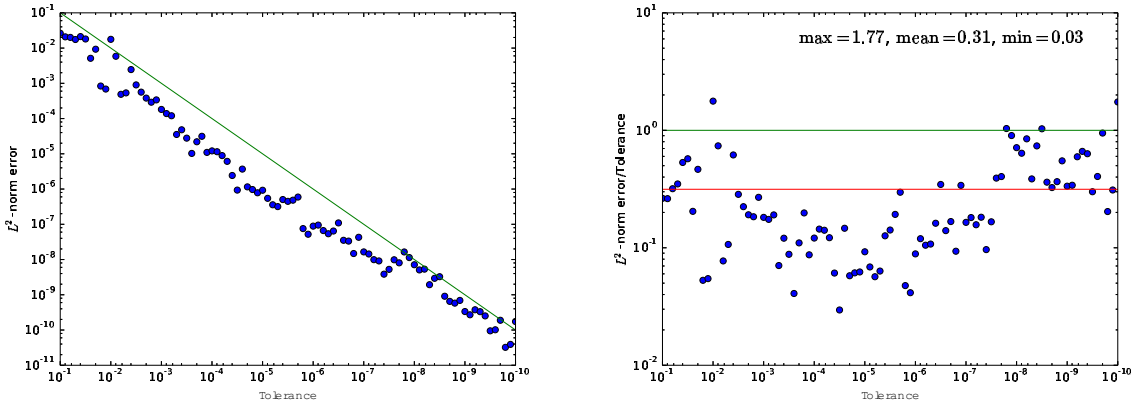
(a) (b)

Fig. 9: BAC/ST, $kcol = 3$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



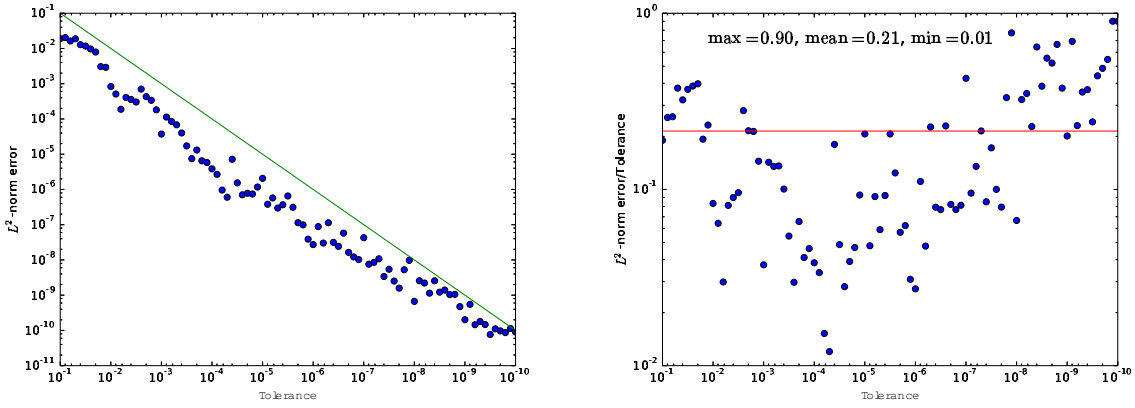
(a) (b)

Fig. 10: BAC/ST, $kcol = 5$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



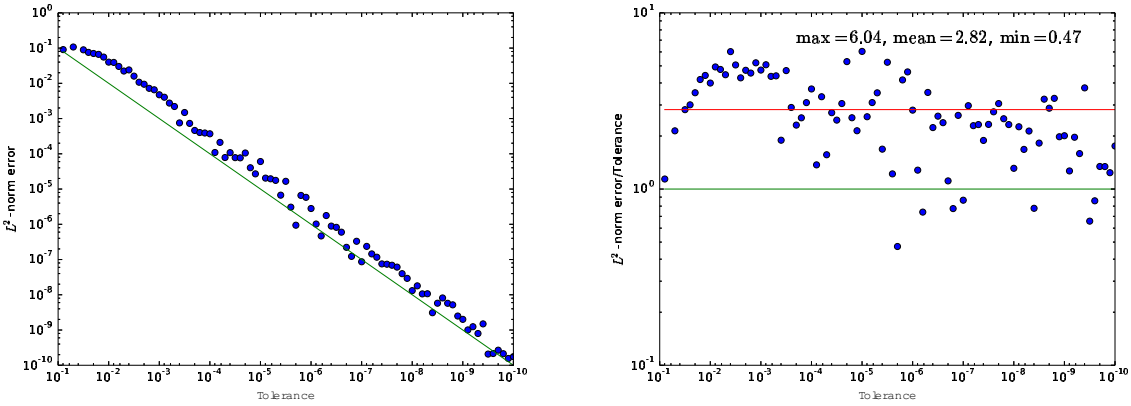
(a) (b)

Fig. 11: BAC/ST, $kcol = 7$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



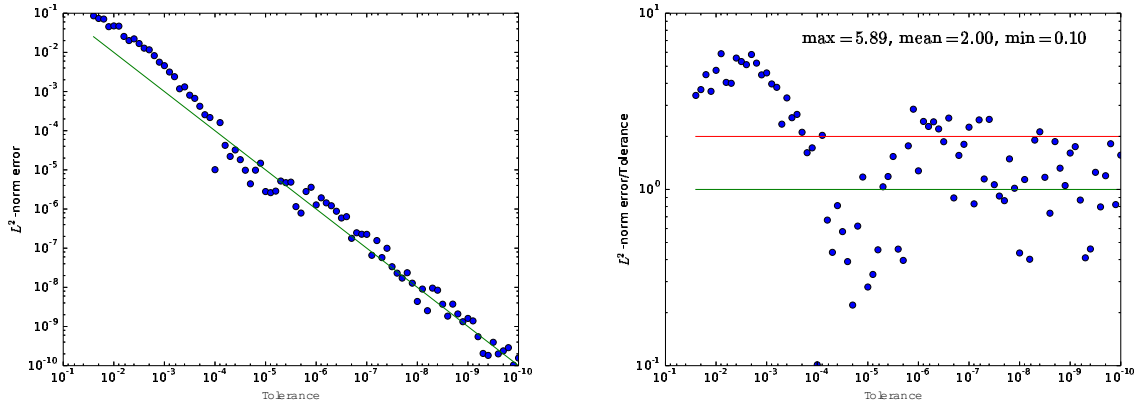
(a) (b)

Fig. 12: BAC/ST, $kcol = 9$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



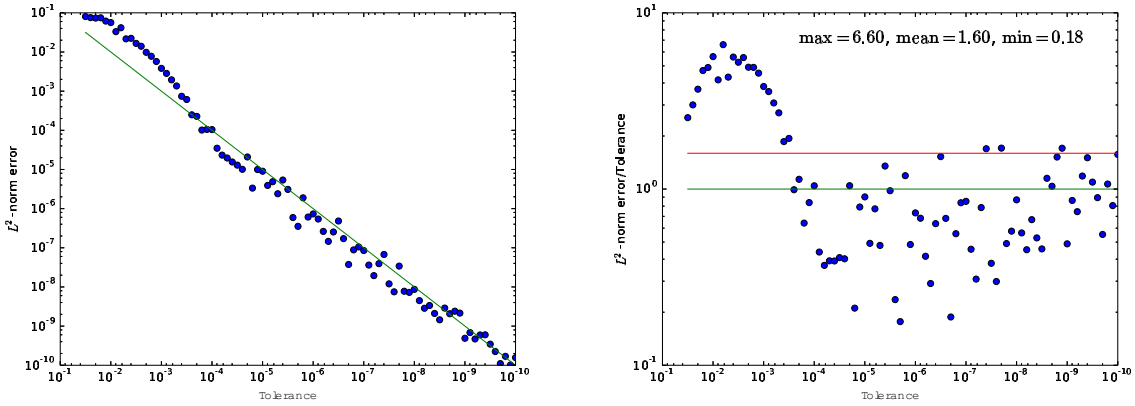
(a) (b)

Fig. 13: BAC/ST, $kcol = 3$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



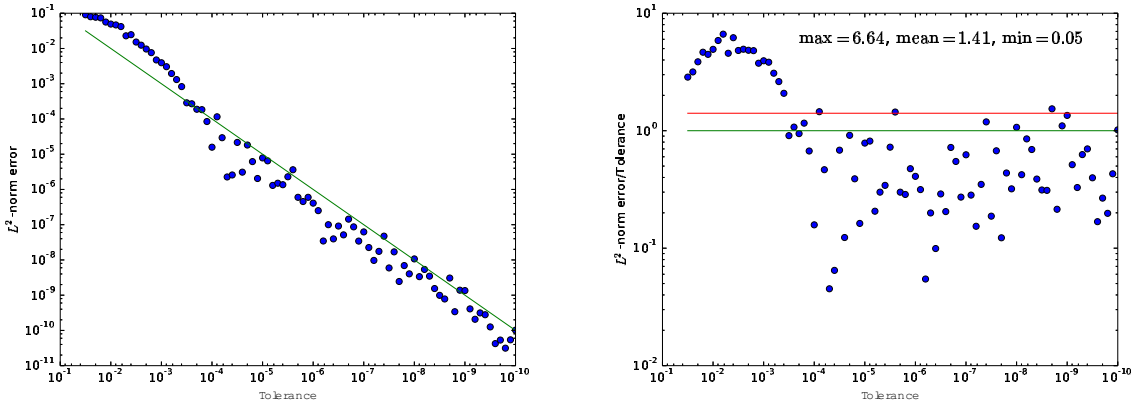
(a) (b)

Fig. 14: BAC/ST, $kcol = 5$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



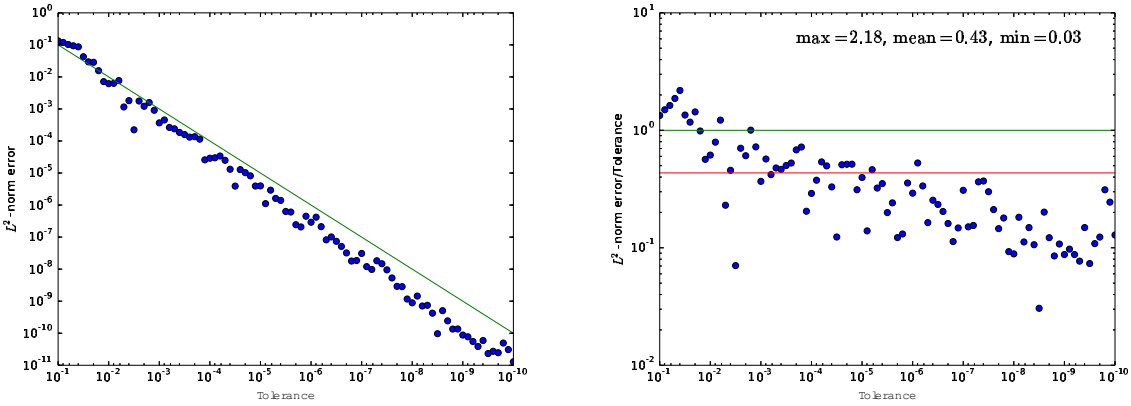
(a) (b)

Fig. 15: BAC/ST, $kcol = 7$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



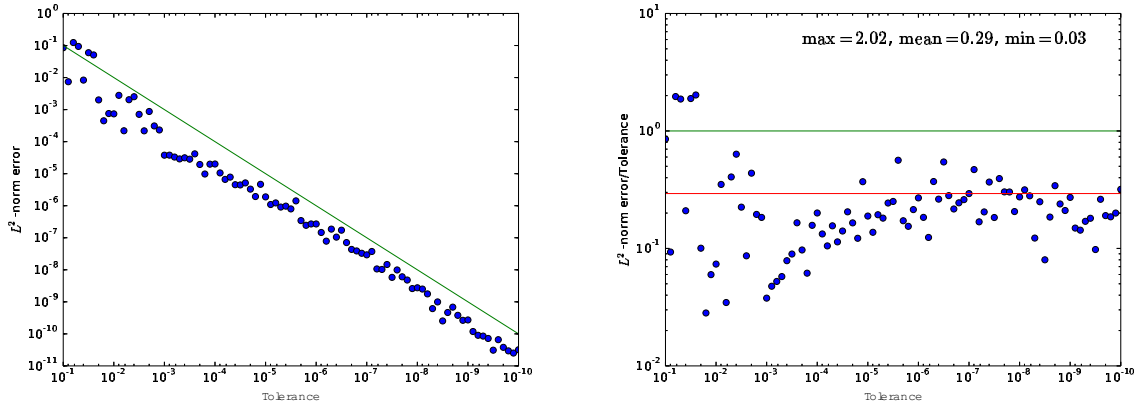
(a) (b)

Fig. 16: BAC/ST, $kcol = 9$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



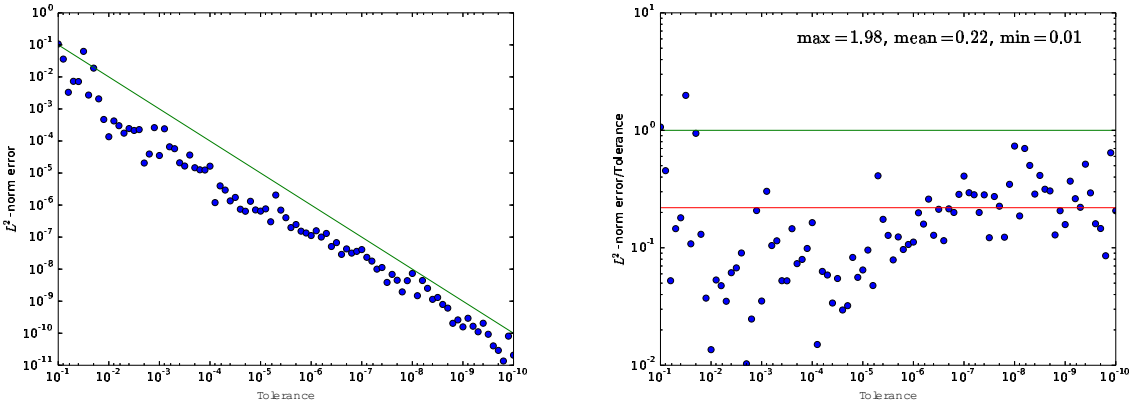
(a) (b)

Fig. 17: BLE/LE, $kcol = 3$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



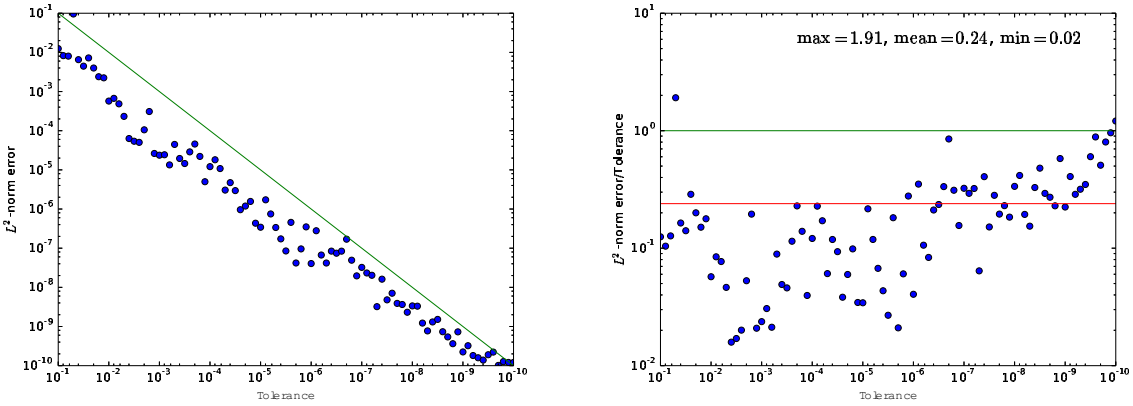
(a) (b)

Fig. 18: BLE/LE, $kcol = 5$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



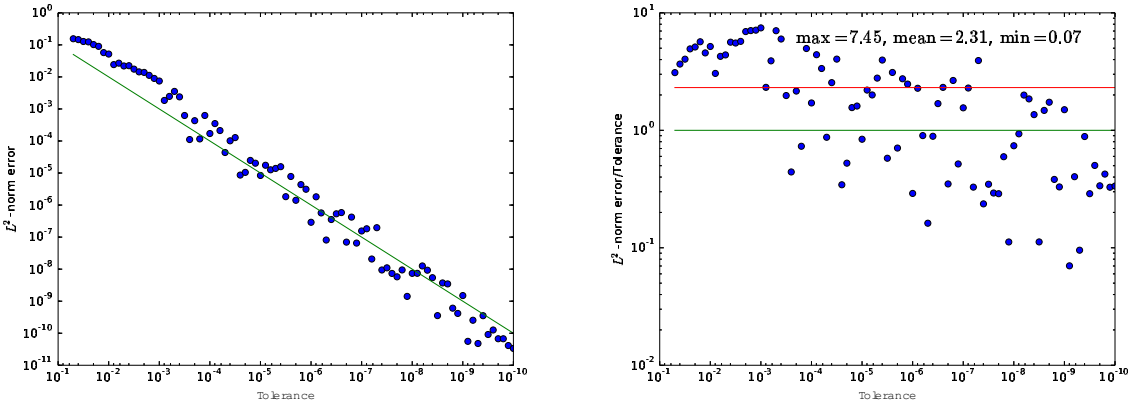
(a) (b)

Fig. 19: BLE/LE, $kcol = 7$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



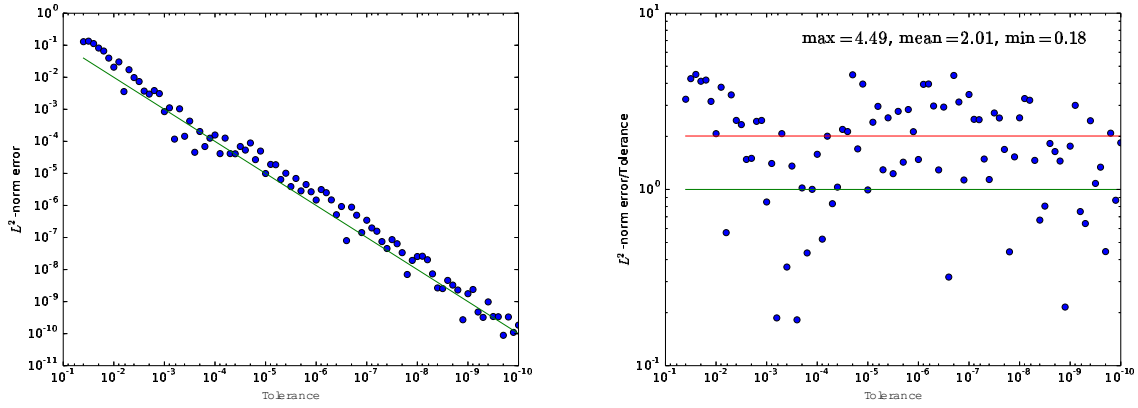
(a) (b)

Fig. 20: BLE/LE, $kcol = 9$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



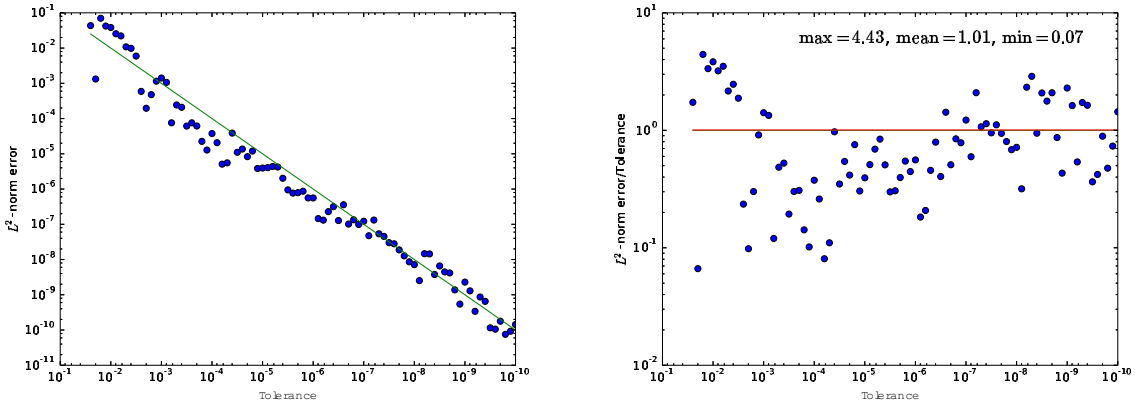
(a) (b)

Fig. 21: BLE/LE, $kcol = 3$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



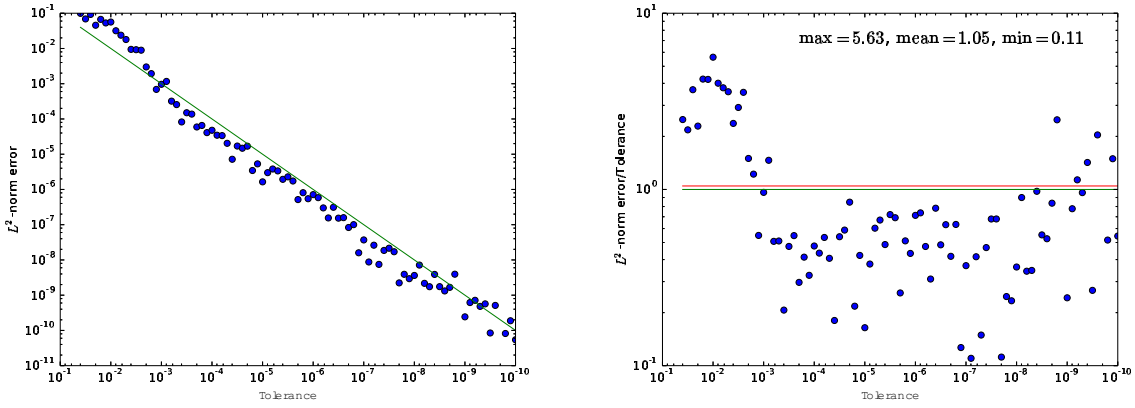
(a) (b)

Fig. 22: BLE/LE, $kcol = 5$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



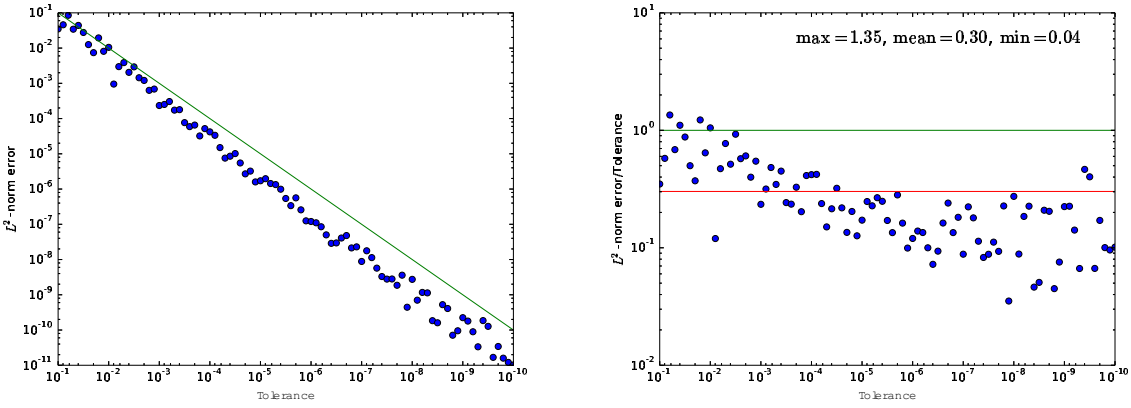
(a) (b)

Fig. 23: BLE/LE, $kcol = 7$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



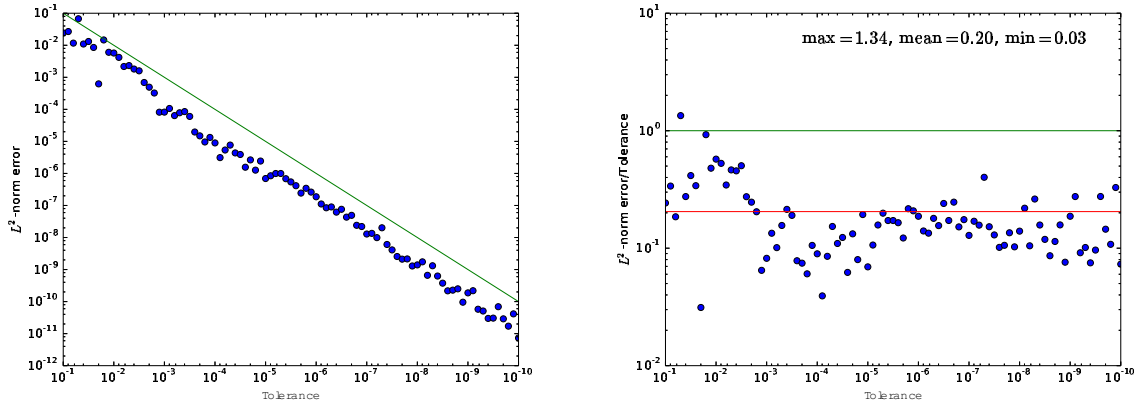
(a) (b)

Fig. 24: BLE/LE, $kcol = 9$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



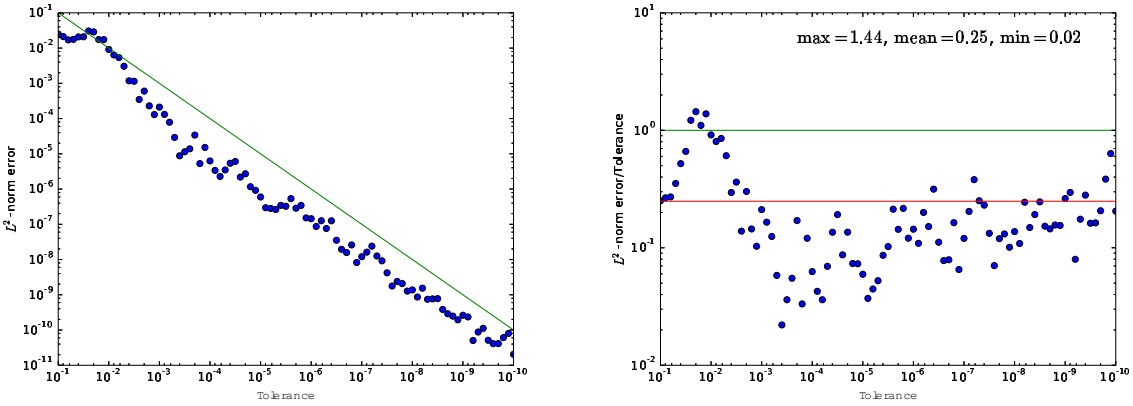
(a) (b)

Fig. 25: BLE/LE, $kcol = 3$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



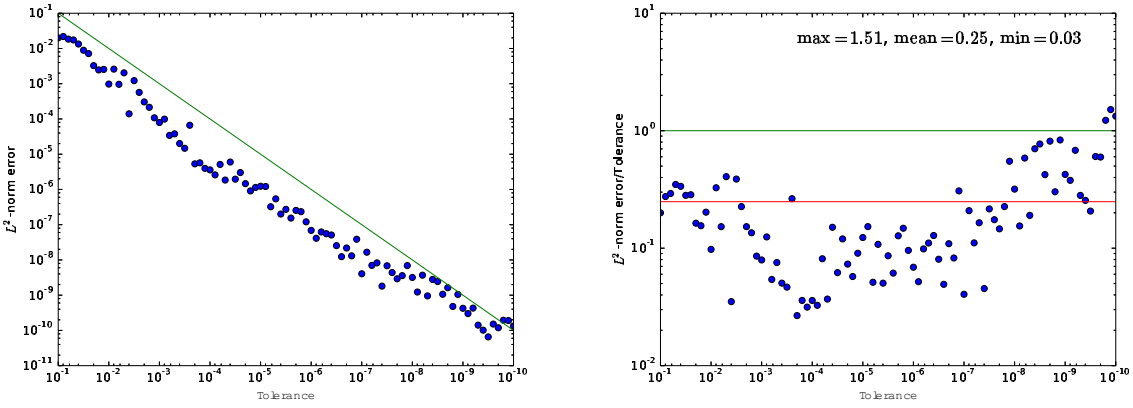
(a) (b)

Fig. 26: BLE/LE, $kcol = 5$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



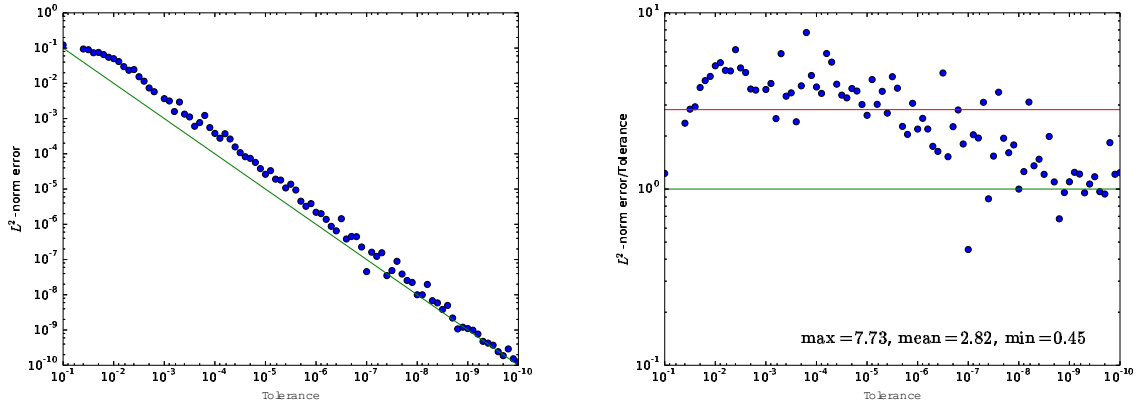
(a) (b)

Fig. 27: BLE/LE, $kcol = 7$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



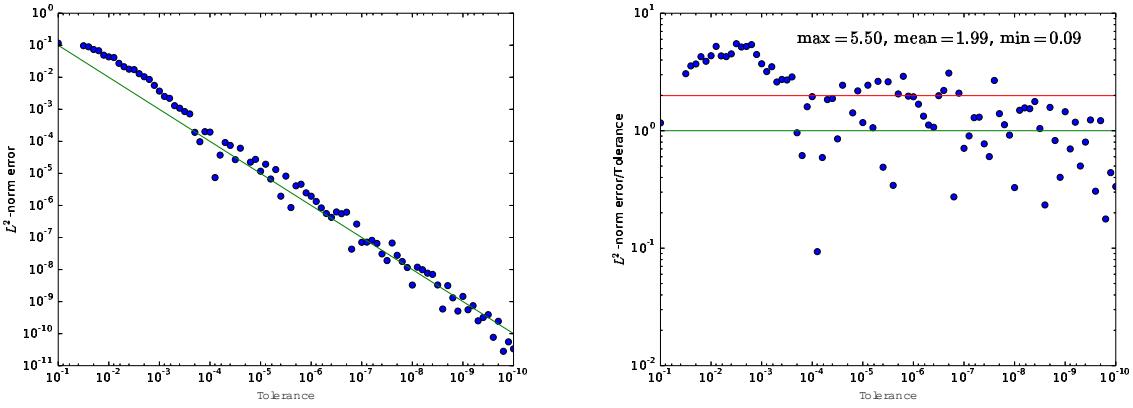
(a) (b)

Fig. 28: BLE/LE, $kcol = 9$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



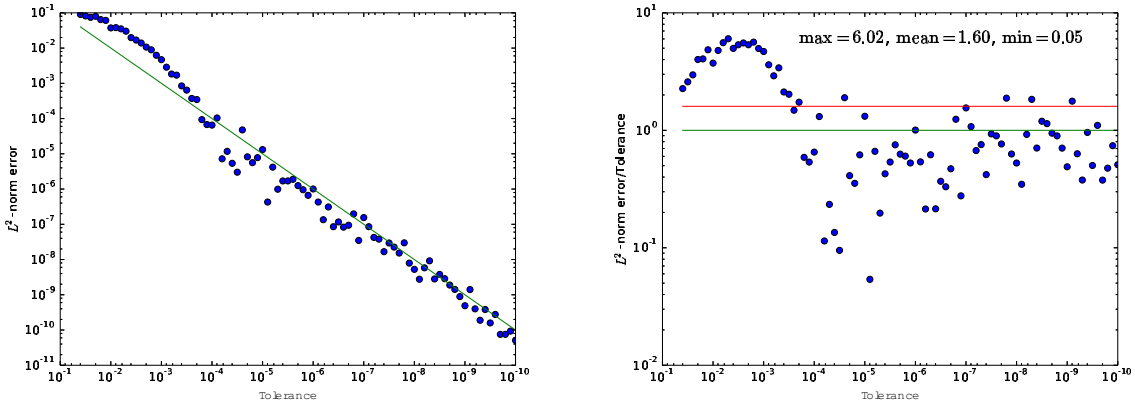
(a) (b)

Fig. 29: BLE/LE, $kcol = 3$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



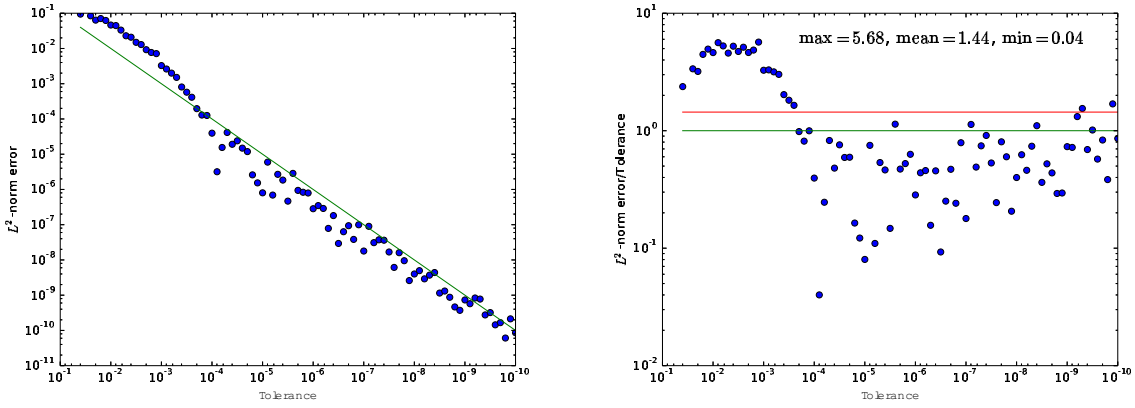
(a) (b)

Fig. 30: BLE/LE, $kcol = 5$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



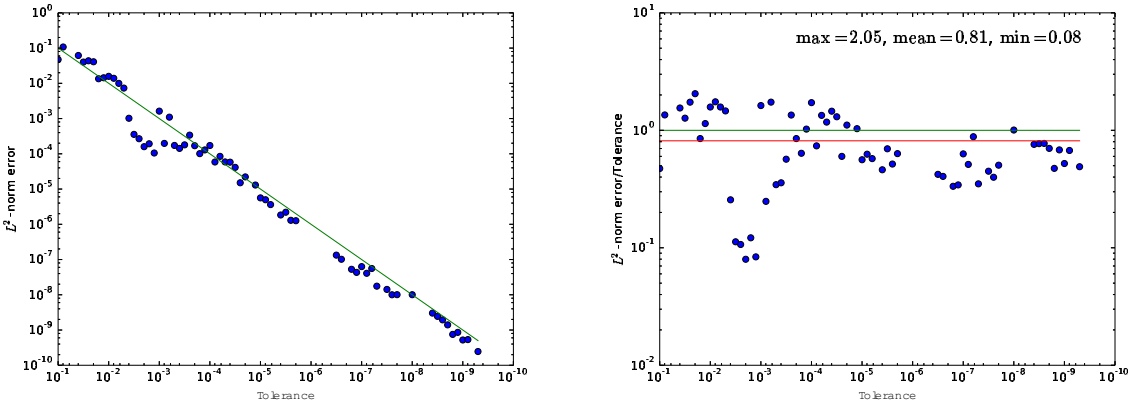
(a) (b)

Fig. 31: BLE/LE, $kcol = 7$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



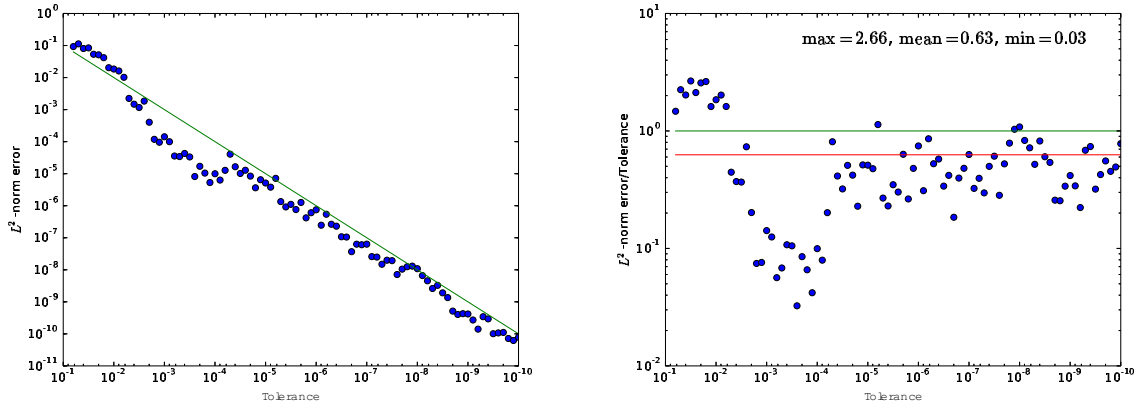
(a) (b)

Fig. 32: BLE/LE, $kcol = 9$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



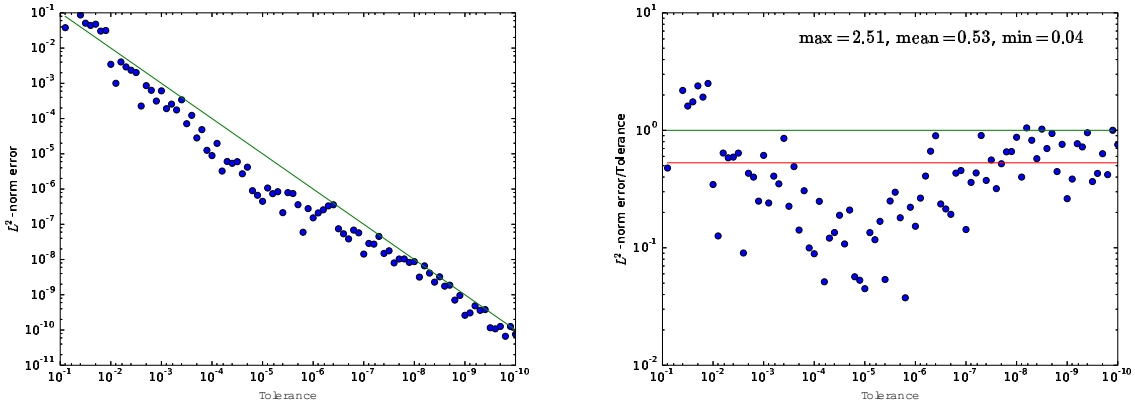
(a) (b)

Fig. 33: SCI/ST, $kcol = 3$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



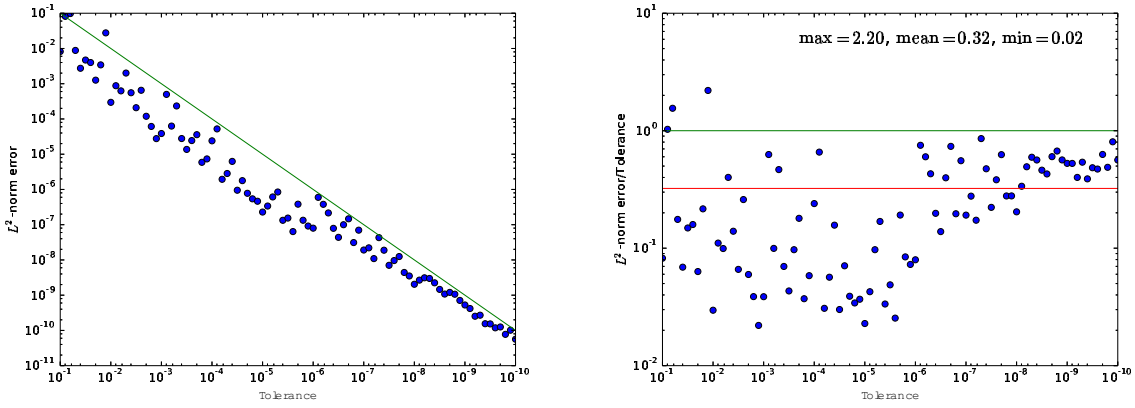
(a) (b)

Fig. 34: SCI/ST, $kcol = 5$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



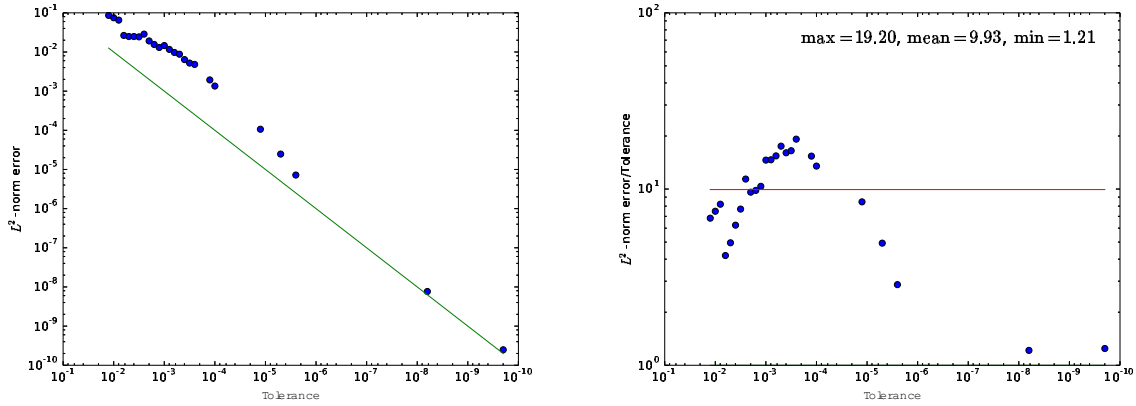
(a) (b)

Fig. 35: SCI/ST, $kcol = 7$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



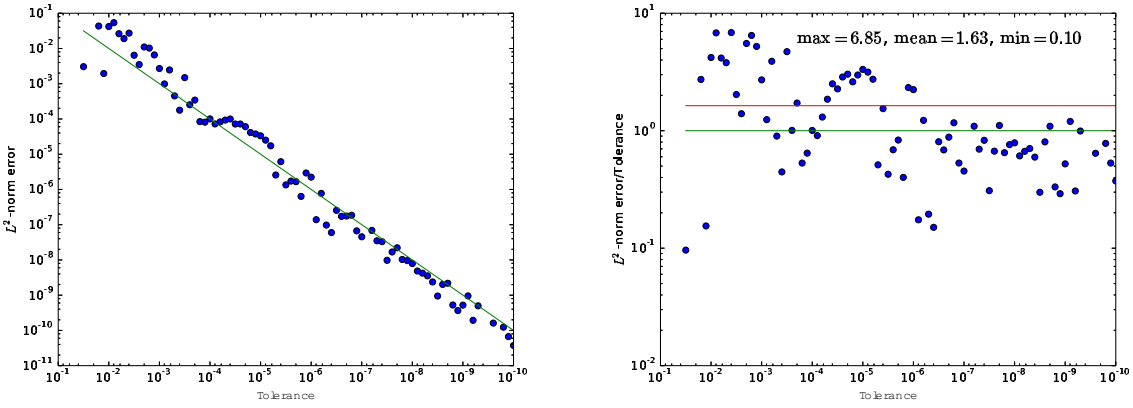
(a) (b)

Fig. 36: SCI/ST, $kcol = 9$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



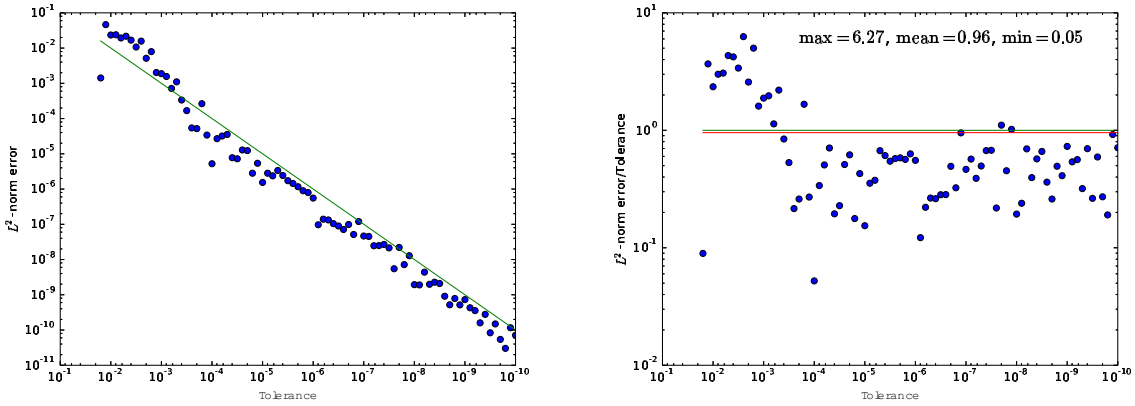
(a) (b)

Fig. 37: SCI/ST, $kcol = 3$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



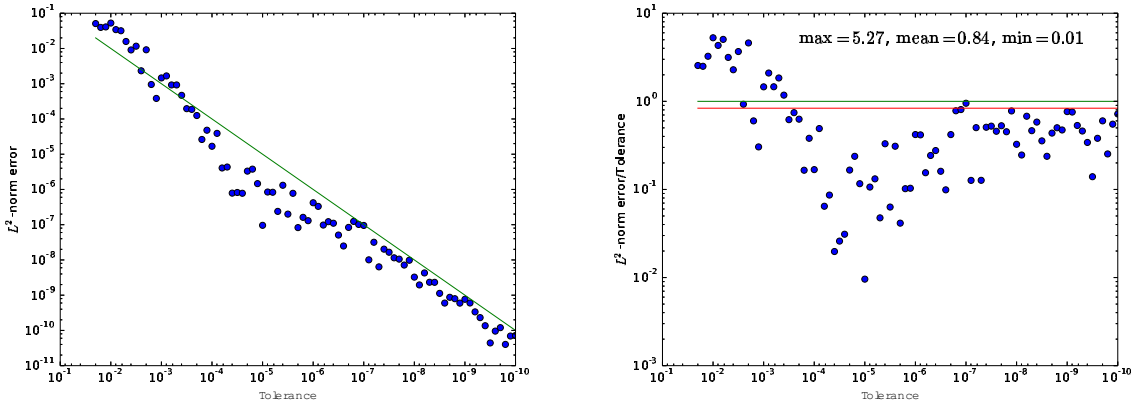
(a) (b)

Fig. 38: SCI/ST, $kcol = 5$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



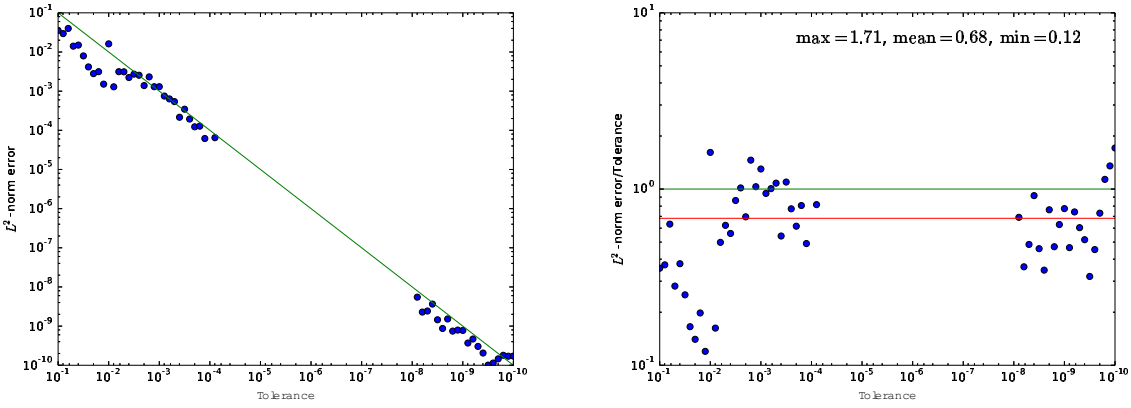
(a) (b)

Fig. 39: SCI/ST, $kcol = 7$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



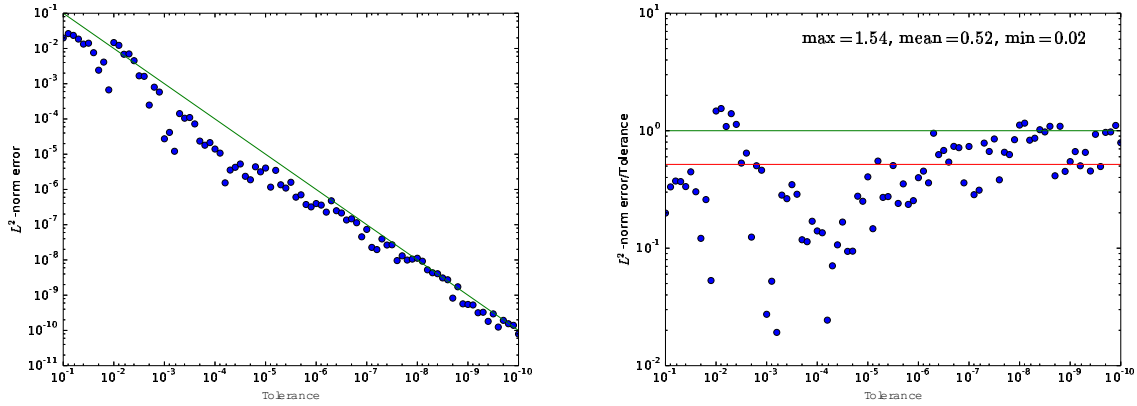
(a) (b)

Fig. 40: SCI/ST, $kcol = 9$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



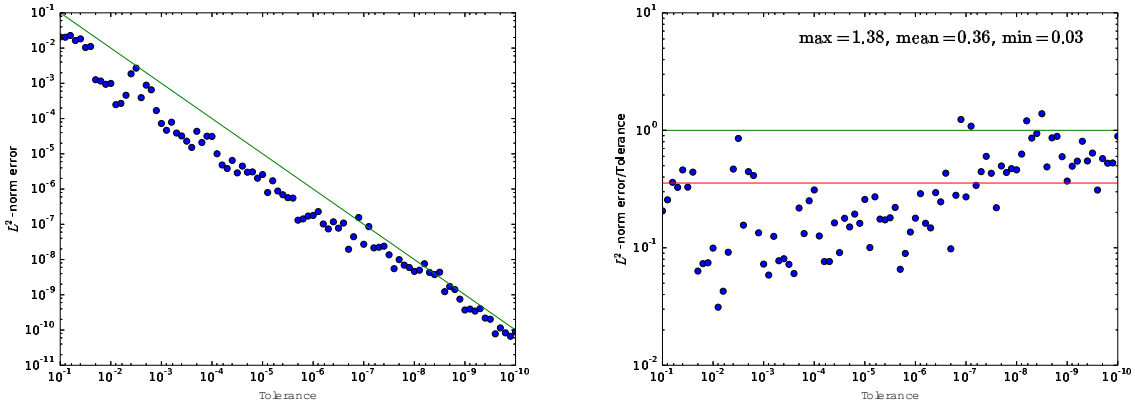
(a) (b)

Fig. 41: SCI/ST, $kcol = 3$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



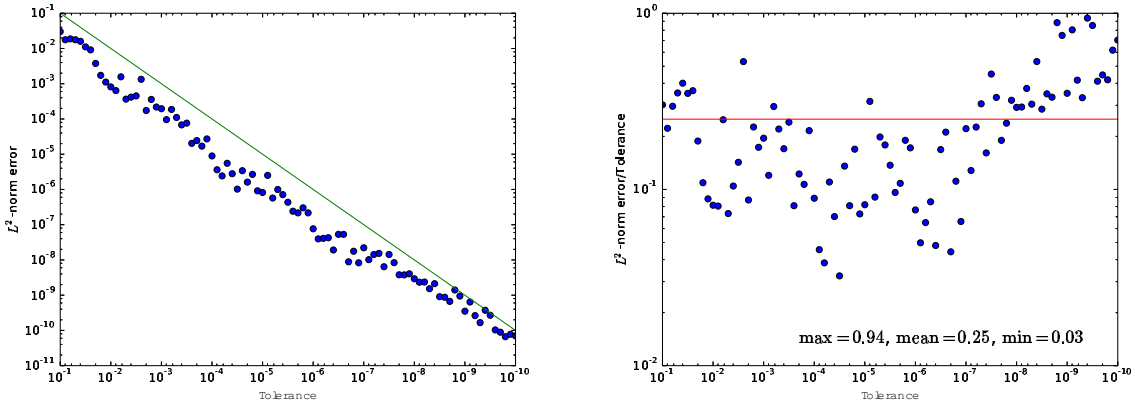
(a) (b)

Fig. 42: SCI/ST, $kcol = 5$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



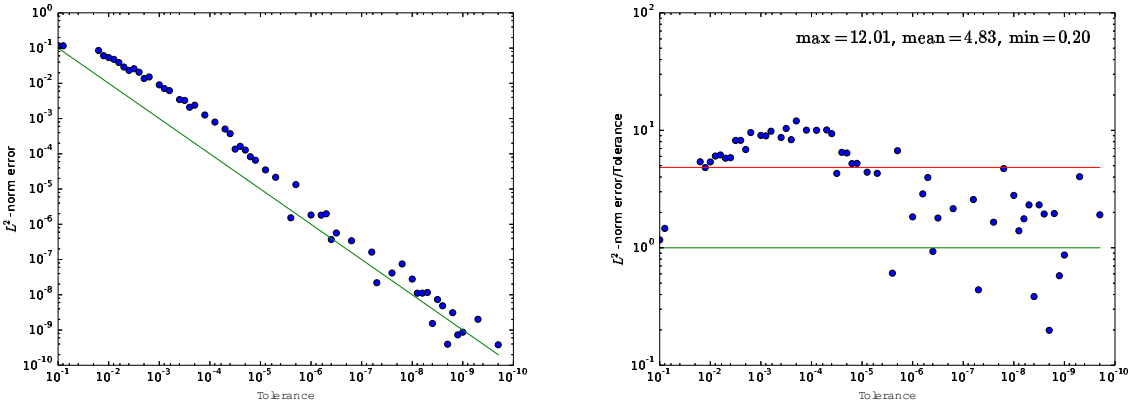
(a) (b)

Fig. 43: SCI/ST, $kcol = 7$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



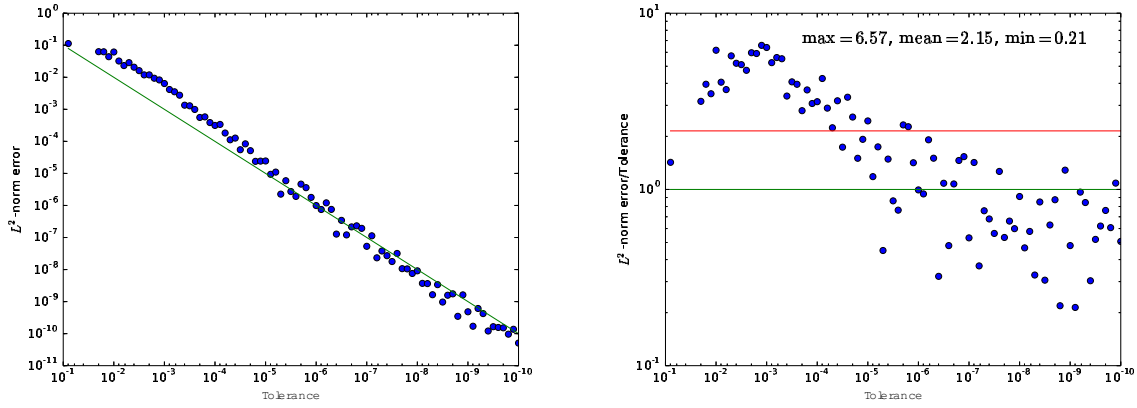
(a) (b)

Fig. 44: SCI/ST, $kcol = 9$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



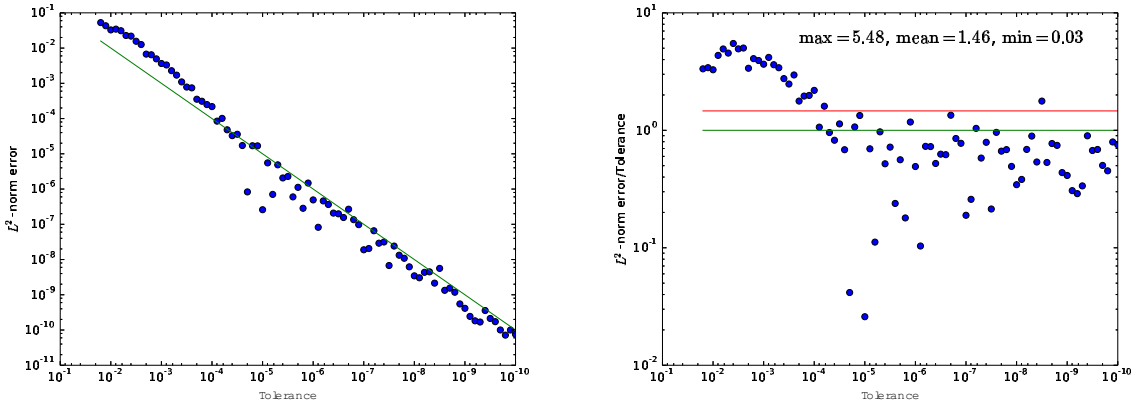
(a) (b)

Fig. 45: SCI/ST, $kcol = 3$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



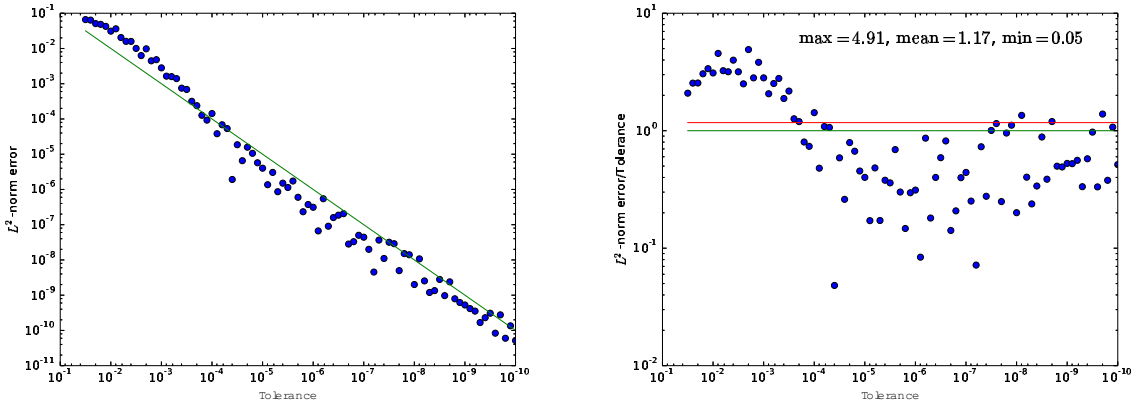
(a) (b)

Fig. 46: SCI/ST, $kcol = 5$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



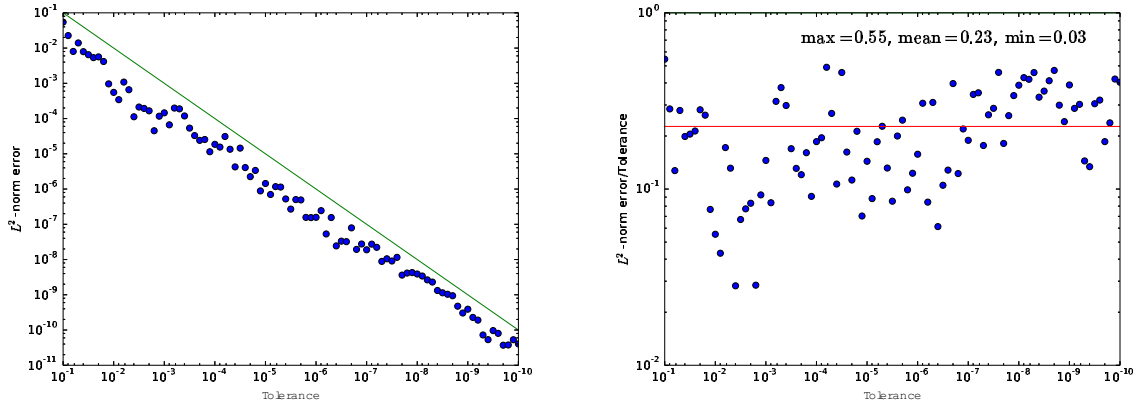
(a) (b)

Fig. 47: SCI/ST, $kcol = 7$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



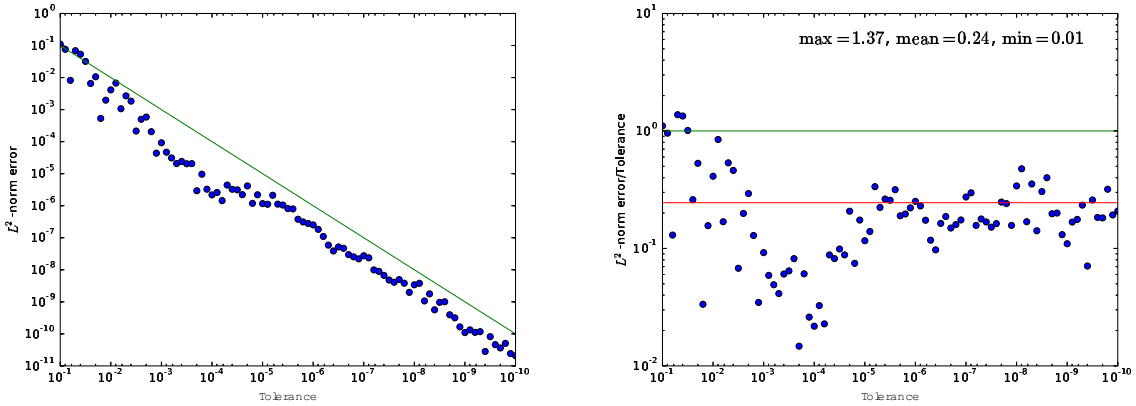
(a) (b)

Fig. 48: SCI/ST, $kcol = 9$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



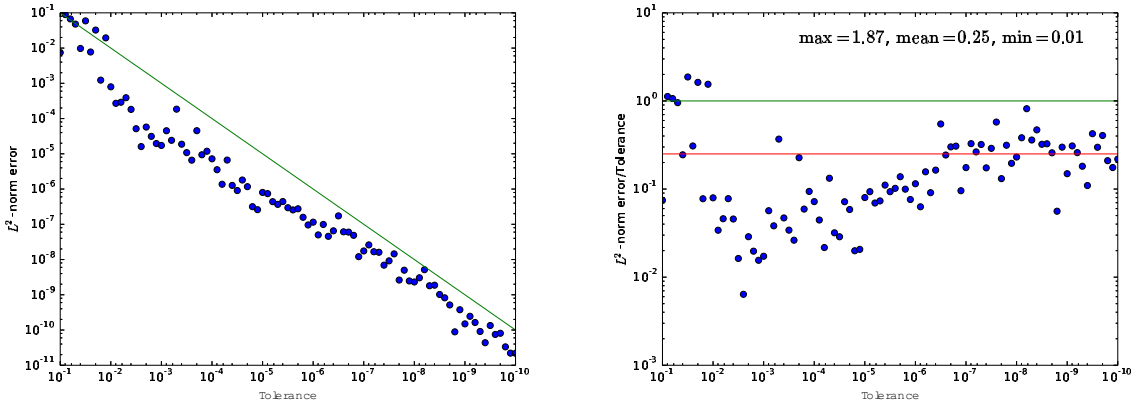
(a) (b)

Fig. 49: LOI/LE, $kcol = 3$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



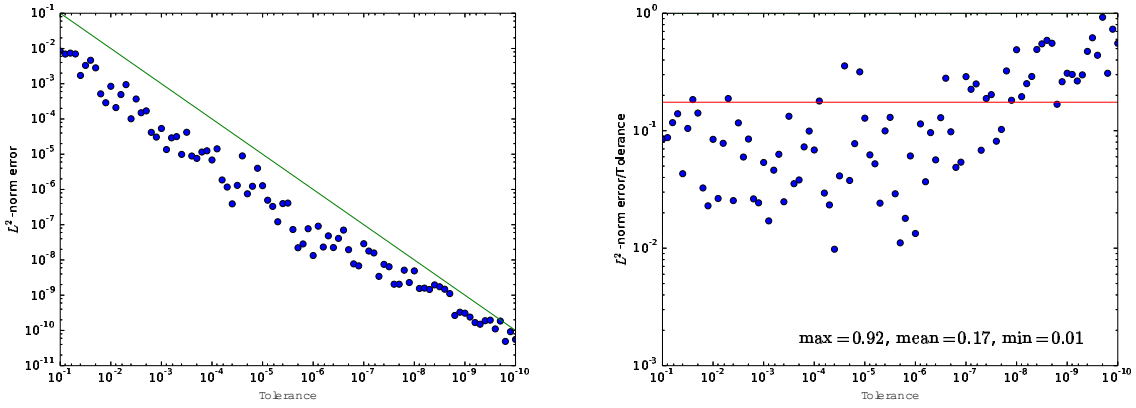
(a) (b)

Fig. 50: LOI/LE, $kcol = 5$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



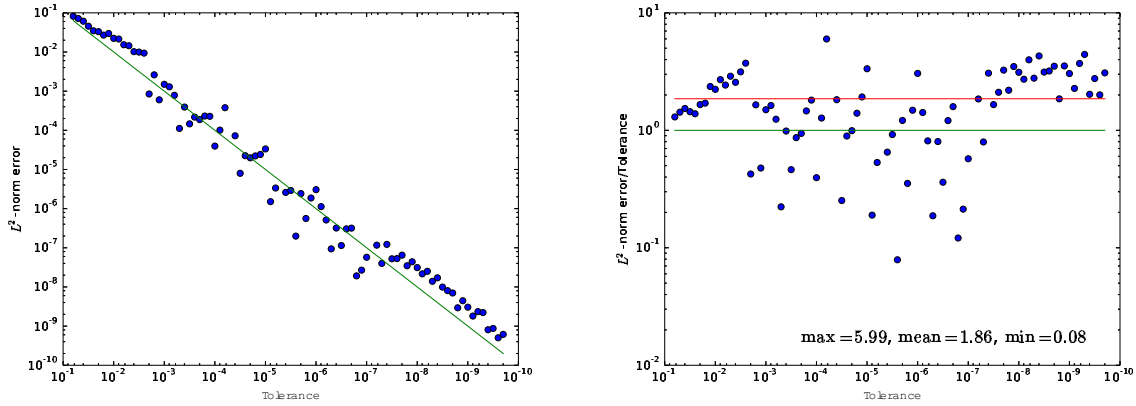
(a) (b)

Fig. 51: LOI/LE, $kcol = 7$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



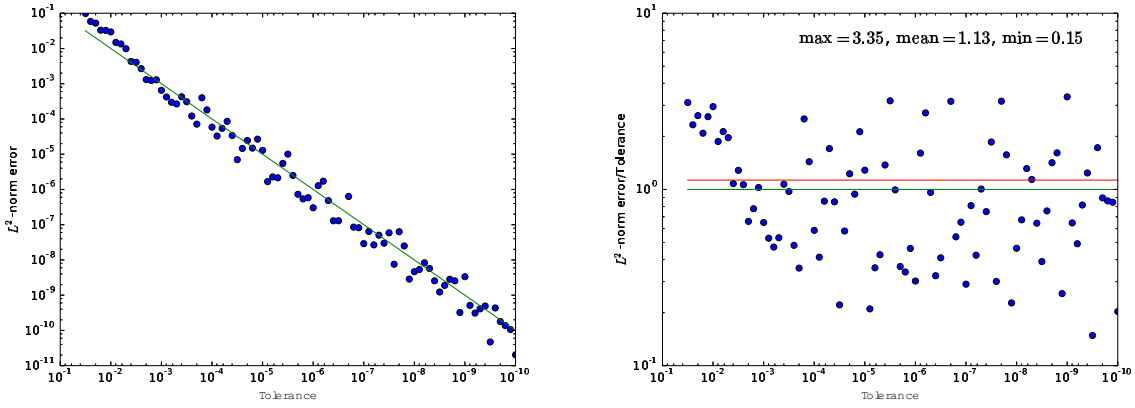
(a) (b)

Fig. 52: LOI/LE, $kcol = 9$, OLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



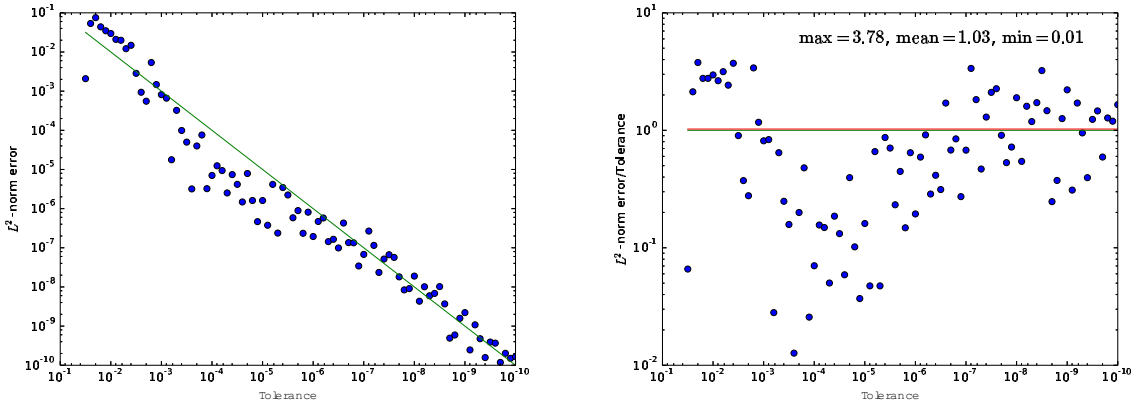
(a) (b)

Fig. 53: LOI/LE, $kcol = 3$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



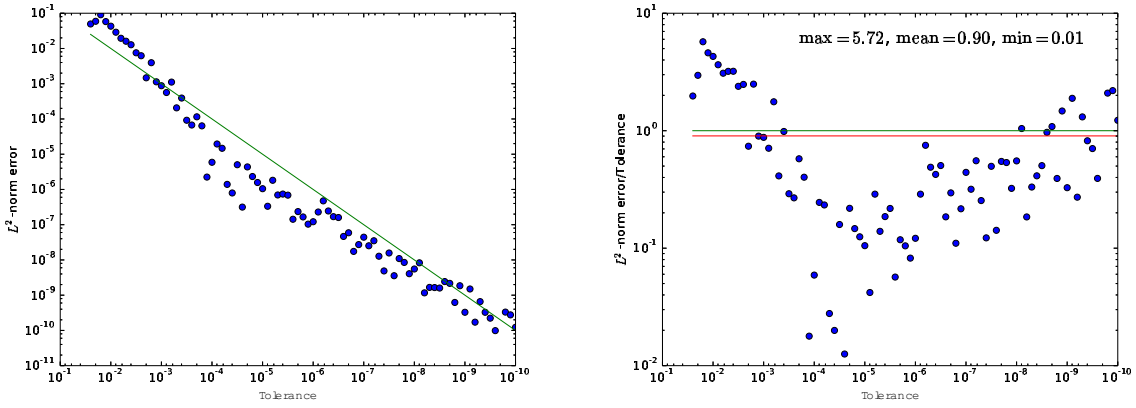
(a) (b)

Fig. 54: LOI/LE, $kcol = 5$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



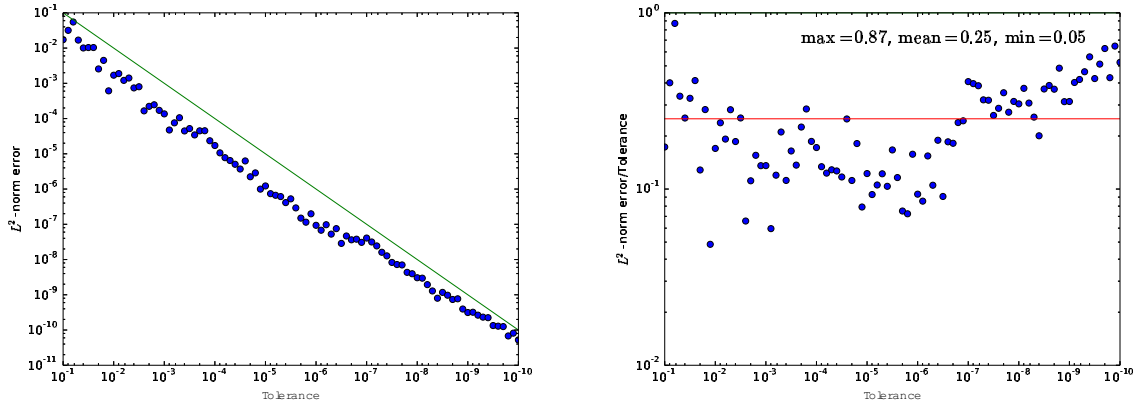
(a) (b)

Fig. 55: LOI/LE, $kcol = 7$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



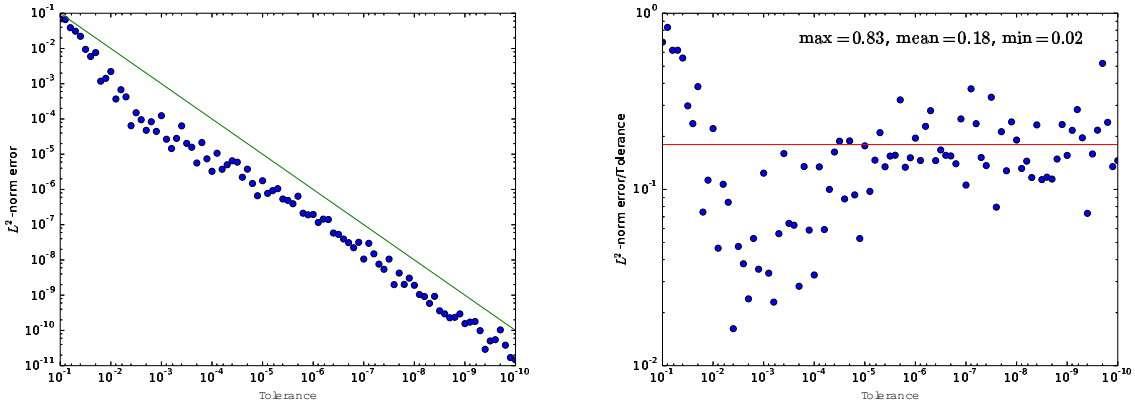
(a) (b)

Fig. 56: LOI/LE, $kcol = 9$, OLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



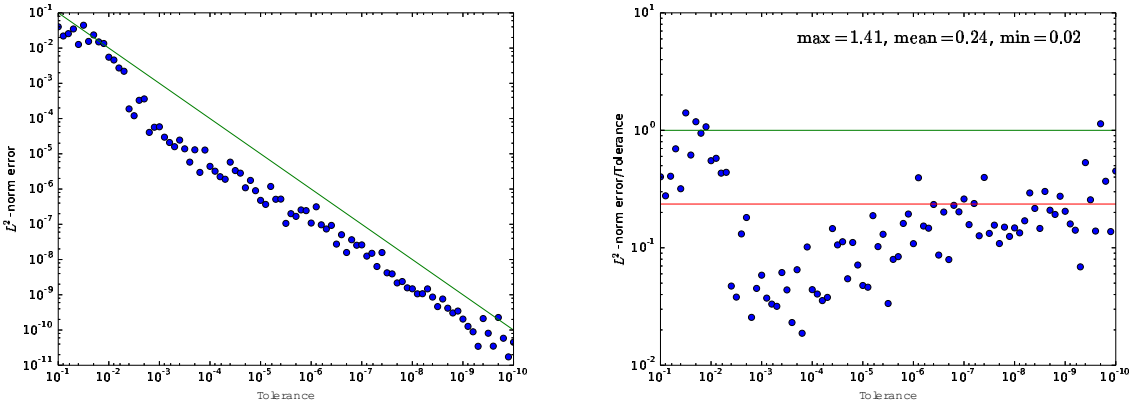
(a) (b)

Fig. 57: LOI/LE, $kcol = 3$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



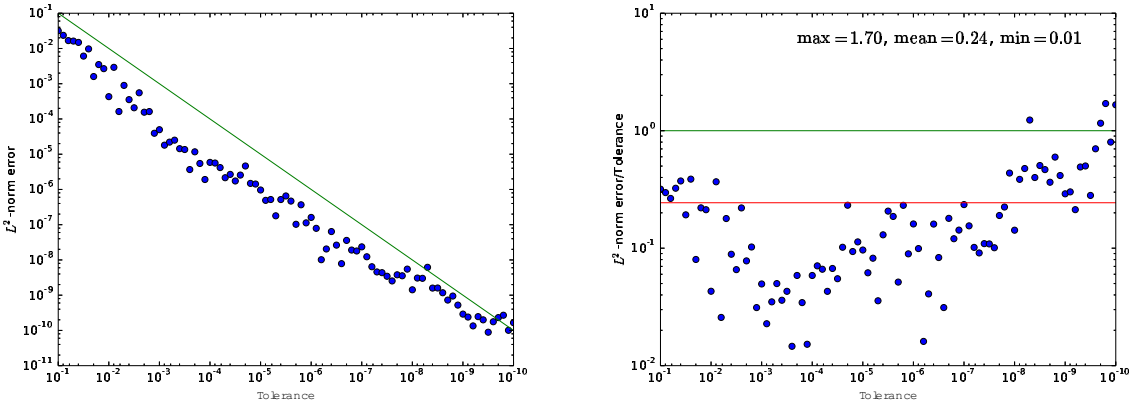
(a) (b)

Fig. 58: LOI/LE, $kcol = 5$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



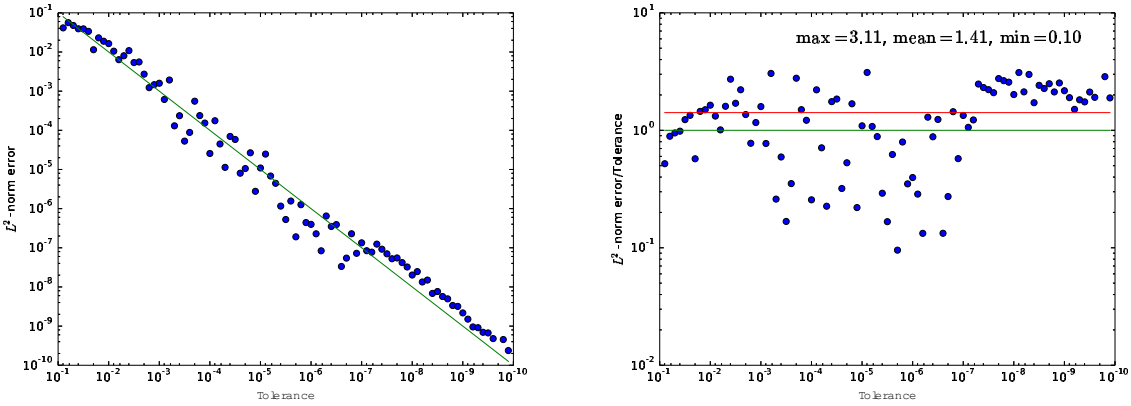
(a) (b)

Fig. 59: LOI/LE, $kcol = 7$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



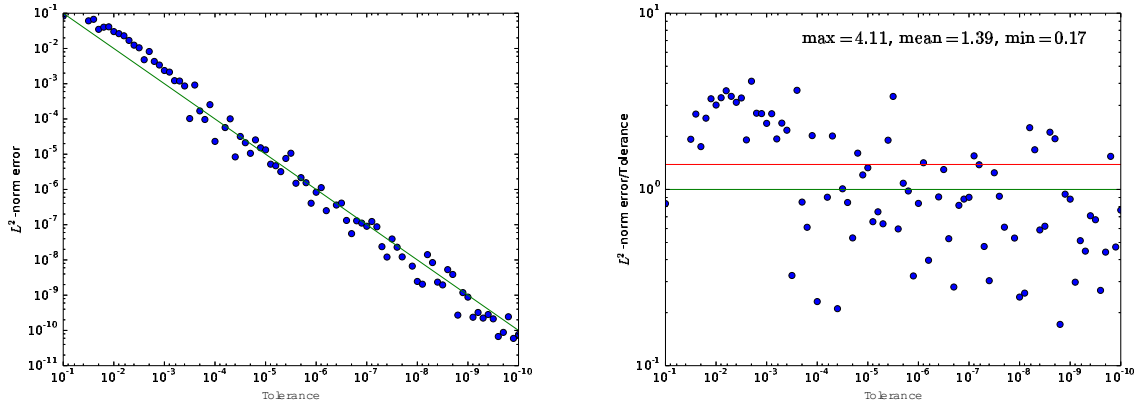
(a) (b)

Fig. 60: LOI/LE, $kcol = 9$, TLBE, $\varepsilon = 10^{-3}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



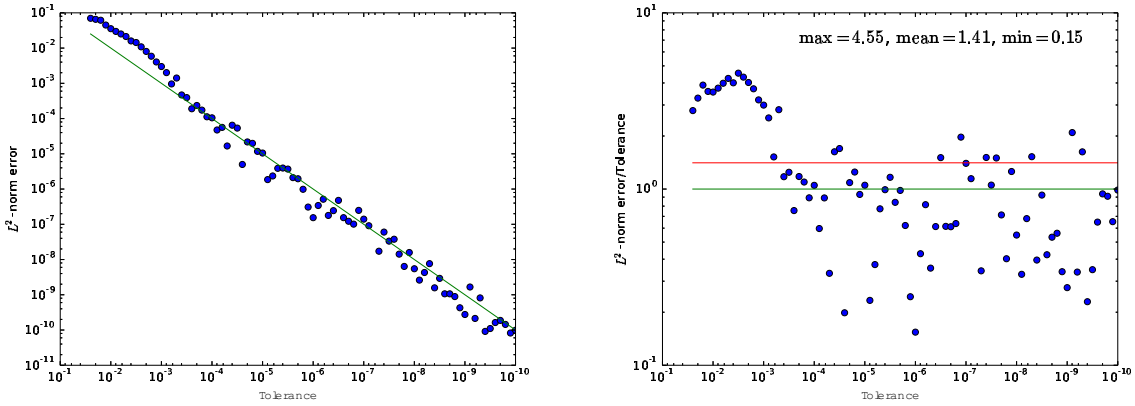
(a) (b)

Fig. 61: LOI/LE, $kcol = 3$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



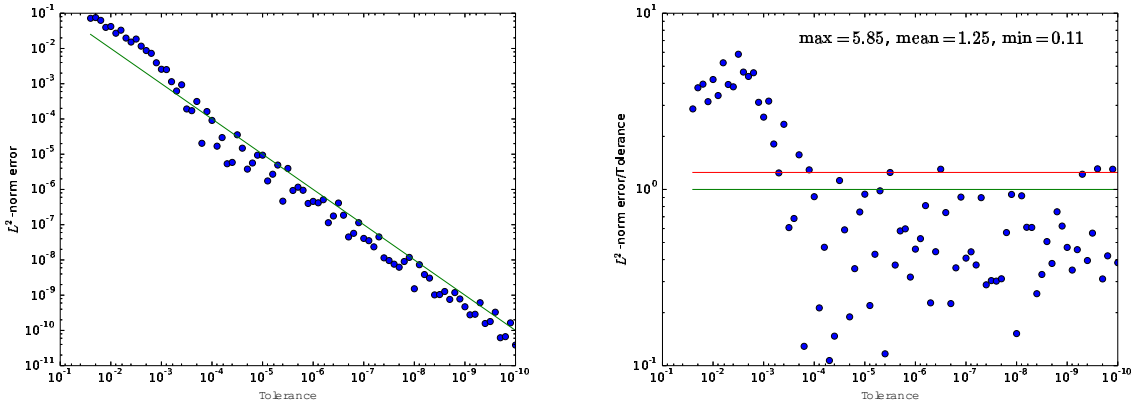
(a) (b)

Fig. 62: LOI/LE, $kcol = 5$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



(a) (b)

Fig. 63: LOI/LE, $kcol = 7$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR



(a) (b)

Fig. 64: LOI/LE, $kcol = 9$, TLBE, $\varepsilon = 10^{-4}$, (a) Tolerance vs. Error (b) Tolerance vs. ETR