Performance of collocation software for singular BVPs

Winfried Auzinger
Othmar Koch
Dirk Praetorius
Gernot Pulverer
Ewa Weinmüller

Technical Report

ANUM Preprint No. 4/04

Technische Universität Wien

Vienna University of Technology

Institute for Analysis and Scientific Computing

Abstract

In this study we investigate the performance of collocation codes applied to approximate the solution of systems of ordinary differential equations of the first order,

$$z'(t) = f(t, z(t)),\ t \in (a, b),$$
$$B(z(a), z(b)) = 0.$$

The right-hand side of the differential equation may contain a singularity of the first kind, which means that it can be of the form

$$f(t, z(t)) = \frac{1}{t} M(t) z(t) + g(t, z(t)),\ t \in (0, 1),$$

where $M(t)$ is a matrix that depends continuously on t and $g(t, z(t))$ is a smooth vector-valued function.

The main purpose of this work is to develop the second version SBVP 2.0 of our first MATLAB Code SBVP 1.0 published in 2002, see [4]. Therefore various tests are executed to find out about the performance of the code with different settings of some of the parameters.

In Chapter 2 the improved version of the Newton method is discussed. Here the old version of the code SBVP 1.0 is used. In Chapter 3 the test runs for SBVP 2.0 are recorded. The parameters are first set as in the beta-version of SBVP and then updated in accordance to the results of the tests. We first investigate the performance of the code for different parameter settings in context of the Jacobian update, further tests cover the questions dealing with the number of meshpoints on the starting grid, the setting for the monitor function or some possibilities of mesh distribution by varying the ratio between the largest and the smallest stepsize.

After the completion of this main phase of this work, in Chapter 4 we compare the new code to other collocation solvers. These other codes are SBVP 1.0, the old version of our code as well as the MATLAB standard code BVP4C and the FORTRAN 90 routine COLNEW.

Main criterion for most of the observations is the runtime required to solve the boundary value problem. As COLNEW is not a MATLAB solver, this criterion is to be exchanged with the fcounts and dfcounts and the number of meshpoints on the final grid.

The boundary value problems that are used as test models throughout this study can be found in Chapter 5.

Contents

Chapter 1

Introduction

1.1 Preliminaries

At the beginning of this study the research group consisting of Ewa B. Weinmüller, Winfried Auzinger, Othmar Koch and Günter Kneisl aimed at an enhancement of their collocation code Sbvp 1.0 for boundary value problems with a singularity of the first kind. In the development process, mainly carried out by Günter Kneisl, the main interest was to provide the knowledge on how changes in some of the codes' basic strategies influence the performance of the program.

Therefore different parameters were in need of optimization. The paradigms that form the basis for the decisions in later chapters were

- stability with respect to the runtime and to the solution of a wide range of problems,
- a focus on difficult basic conditions to ensure best results for hard to solve problems, and
- ingenuousness to necessary major changes that could be possible at any time during the test phase.

1.2 Test Conditions

Most of the tests in this study are carried out with the beta-version of Sbvp 2.0. Only for the tests from Chapter 2 the old version, Sbvp 1.0, is used. The parameters are tested one by one. After one parameter has been fixed at its final value the tests for the next one is executed.

At the end of all tests aiming at parameter optimization the performance of Sbvp 2.0 is compared to Sbvp 1.0, the Matlab standard solver Bvp4c and the Fortran routine Colnew.

Since Sbvp is a Matlab routine, the test environment is a Windows version of Matlab. We use Matlab 6.5.0.180913a (R13). In Section 4.4.1, the test environment is changed to a Unix system with an equivalent Matlab version.

For the Windows operating system, the underlying processor is an AMD Athlon 1400 with 512 MB DDRAM and a swap file set to 1 GB. The Unix system is the stat-server of the Institute of Analysis and Scientific Computing, at the Vienna University of Technology.

1.3 Problem Class

In this study we consider boundary value problems (BVPS) for systems of ordinary differential equations (ODEs) of the form,

$$z'(t) = f(t, z(t)),\ t \in (a, b),$$
$$B(z(a), z(b)) = 0.$$

The right-hand side f may contain a singularity of the first kind, which means

$$f(t, z(t)) = \frac{1}{t} M(t) z(t) + g(t, z(t)),\ t \in (0, 1),$$

where $M(t)$ is a matrix that depends continuously on t and $g(t, z(t))$ is a smooth vector-valued function.

Models for BVPS used in the present work can be found in Section 5.1.

1.4 SBVP 1.0

As the main purpose of this work is to guide and facilitate the development of SBVP 2.0, the basis of this study is SBVP 1.0, a MATLAB code for the numerical solution of boundary value problems for systems of ODEs with a singularity of the first kind. The code SBVP 1.0 is based on collocation at either equidistant or Gaussian collocation points. For more information on SBVP 1.0 refer to [5].

1.5 Test Approach

The core of the tests are the BVP-files. These are MATLAB m-files in which the BVPS are encoded so that they can be used by SBVP. The results of the routine are saved in an array of logstructs which form a simple database in the MATLAB environment. All necessary data is saved there and during the test some thousands of single datasets are produced. Unfortunately all routines different from SBVP 2.0 do not support such a load of information for the solution process. So the data needs to be extracted using global variables or slight hacks in the code. Where such modifications are necessary, the code under consideration is tested using known BVPS to provide the highest safety.

After the datasets are produced and the database is complete, the data is extracted via self written evaluation-routines which are developed with respect to the advantages and special structure of the database. The extracted results are the basis for the decisions in the following chapters.

Chapter 2

Evolution from SBVP 1.0 to SBVP 2.0

In this chapter the relevant program parameters are optimized so as to yield highest stability at the quickest possible runtime. While Section 2.1 contains test results for a parameter taken directly from `sbvpcol.m`, the collocation solver of SBVP 1.0, all the other sections in this chapter are based on a beta-version of SBVP 2.0.

2.1 Newton SBVP 1.0

In the subroutine `sbvpcol.m`, the collocation solver of SBVP 1.0, the constraint for the monotonicity test

$$\|\tilde{\Delta}\| \geq \alpha \|\Delta\|, \tag{2.1}$$

has to be optimized with respect to α. Here, $\Delta = U\backslash(L\backslash(-F))$ is the Newton correction, where $[L, U]$ is the LU-decomposition of the Jacobian DF and F is the residual calculated using an approximation A_n, whereas $\tilde{\Delta} = U\backslash(L\backslash(-\tilde{F}))$ is the Newton correction where $\tilde{F}$ is the new residual calculated using the new approximation $A_{n+1} = A_n + \lambda\Delta$.

If the scaled residual does not improve by at least α, the Jacobian of the problem is updated. Apart from α the parameter ZfSensitive controls whether the Jacobian is updated in every single step. If ZfSensitive is set to 1, the Jacobian is always updated, whereas if it is set to 0, constraint (2.1) decides about updating.

2.1.1 Initial Test for Jacobian Update

First, α from inequality (2.1) is set to $\frac{1}{10}$, $\frac{1}{2}$ and $\frac{3}{4}$ and the results are compared with the case where the Jacobian is always updated. In this case, the parameter ZfSensitive is set to 1.

As this test is aiming at an improvement of the implemented Newton iteration instead of testing the whole package, the mesh-size has to be set manually. In order to see what influence the parameter α has on the runtime, the number of meshpoints was set to 50 and 200 points.

For these tests thirteen of the test problems described in Section 5 are selected. The following tables (2.1 - 2.13) show the test results in terms of the runtime. Runtime is measured using the MATLAB task `cputime`. With this recommended method the runtime is set before the routine starts and when the routine stops it is set again less the prior value. Other test characteristics like the number of fcounts and dfcounts turned out to be very intricately related to the performance of the code and their interrelation was too involved to admit reliable conclusions. Therefore the gained insights are mainly based on runtimes.

In the tests of this chapter, when we talk about tolerance, the Newton tolerance is meant. In all the other chapters tolerance means the global error tolerance that needs to be satisfied by the computed solution.

In most of the cases the solution is reached in the shortest runtime when α is set to $\frac{1}{10}$, as can be seen from the tables, but this is not the case for every single test configuration. So for BVP 21a (cf. Table 2.3) $\alpha = \frac{1}{10}$ is the best choice especially for the configuration with 200 points at a tolerance of 10^{-13} with only 50 % of the runtime for $\alpha = \frac{1}{2}$ or $\alpha = \frac{3}{4}$. For BVP 1a (cf. Table 2.2) the other two settings for α lead to better results. A quite ambivalent result can be seen for BVP 56 (cf. Table 2.6). Here the runtimes for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are noticeably higher than those for $\alpha = \frac{1}{10}$, at least for the two configurations with the tolerance set to 10^{-6} and the configuration with the tolerance at 10^{-13} and 50 points. For 200 points and the stricter tolerances the values $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ lead to much better results.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing0026	10^{-6}	50	0.1	0	0.223
funsing0026	10^{-6}	50	0.5	0	0.198
funsing0026	10^{-6}	50	0.75	0	0.201
funsing0026	10^{-6}	50	—	1	0.215
funsing0026	10^{-6}	200	0.1	0	0.500
funsing0026	10^{-6}	200	0.5	0	0.547
funsing0026	10^{-6}	200	0.75	0	0.546
funsing0026	10^{-6}	200	—	1	0.645
funsing0026	10^{-13}	50	0.1	0	0.215
funsing0026	10^{-13}	50	0.5	0	0.229
funsing0026	10^{-13}	50	0.75	0	0.230
funsing0026	10^{-13}	50	—	1	0.242
funsing0026	10^{-13}	200	0.1	0	0.557
funsing0026	10^{-13}	200	0.5	0	0.679
funsing0026	10^{-13}	200	0.75	0	0.671
funsing0026	10^{-13}	200	—	1	0.739

Table 2.1: Runtimes for BVP 0026 when testing α. Slight advantages for $\alpha = \frac{1}{10}$ can be seen but they are quite small except for the test with 200 points and the tolerance set to 10^{-13}.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing1a	10^{-6}	50	0.1	0	0.197
funsing1a	10^{-6}	50	0.5	0	0.194
funsing1a	10^{-6}	50	0.75	0	0.204
funsing1a	10^{-6}	50	—	1	0.214
funsing1a	10^{-6}	200	0.1	0	0.600
funsing1a	10^{-6}	200	0.5	0	0.476
funsing1a	10^{-6}	200	0.75	0	0.474
funsing1a	10^{-6}	200	—	1	0.746
funsing1a	10^{-13}	50	0.1	0	0.232
funsing1a	10^{-13}	50	0.5	0	0.220
funsing1a	10^{-13}	50	0.75	0	0.223
funsing1a	10^{-13}	50	—	1	0.252
funsing1a	10^{-13}	200	0.1	0	0.606
funsing1a	10^{-13}	200	0.5	0	0.543
funsing1a	10^{-13}	200	0.75	0	0.522
funsing1a	10^{-13}	200	—	1	0.832

Table 2.2: Runtimes for BVP 1a when testing α. When ZfSensitive is set to 1, the runtime is higher in every single test, whereas the other options are quite comparable. $\alpha = \frac{1}{10}$ is slow for 10^{-13} and 200 points.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing21a	10^{-6}	50	0.1	0	0.247
funsing21a	10^{-6}	50	0.5	0	0.359
funsing21a	10^{-6}	50	0.75	0	0.381
funsing21a	10^{-6}	50	—	1	0.265
funsing21a	10^{-6}	200	0.1	0	0.774
funsing21a	10^{-6}	200	0.5	0	1.504
funsing21a	10^{-6}	200	0.75	0	1.487
funsing21a	10^{-6}	200	—	1	0.886
funsing21a	10^{-13}	50	0.1	0	0.253
funsing21a	10^{-13}	50	0.5	0	0.378
funsing21a	10^{-13}	50	0.75	0	0.380
funsing21a	10^{-13}	50	—	1	0.290
funsing21a	10^{-13}	200	0.1	0	0.781
funsing21a	10^{-13}	200	0.5	0	1.502
funsing21a	10^{-13}	200	0.75	0	1.500
funsing21a	10^{-13}	200	—	1	0.980

Table 2.3: Runtimes for BVP 21a when testing α. $\alpha = \frac{1}{10}$ is fastest for every configuration, whereas $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ nearly need twice the runtime to finish.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing37c	10^{-6}	50	0.1	0	0.205
funsing37c	10^{-6}	50	0.5	0	0.219
funsing37c	10^{-6}	50	0.75	0	0.202
funsing37c	10^{-6}	50	—	1	0.234
funsing37c	10^{-6}	200	0.1	0	0.586
funsing37c	10^{-6}	200	0.5	0	0.526
funsing37c	10^{-6}	200	0.75	0	0.528
funsing37c	10^{-6}	200	—	1	0.745
funsing37c	10^{-13}	50	0.1	0	0.214
funsing37c	10^{-13}	50	0.5	0	0.230
funsing37c	10^{-13}	50	0.75	0	0.236
funsing37c	10^{-13}	50	—	1	0.245
funsing37c	10^{-13}	200	0.1	0	0.627
funsing37c	10^{-13}	200	0.5	0	0.573
funsing37c	10^{-13}	200	0.75	0	0.554
funsing37c	10^{-13}	200	—	1	0.834

Table 2.4: Runtimes for BVP 37c when testing α. With α set to $\frac{1}{2}$ or $\frac{3}{4}$ the best results are achieved here. In the most difficult configuration $\alpha = \frac{1}{10}$ is about 10 % slower than these two.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing54	10^{-6}	50	0.1	0	0.209
funsing54	10^{-6}	50	0.5	0	0.198
funsing54	10^{-6}	50	0.75	0	0.206
funsing54	10^{-6}	50	—	1	0.257
funsing54	10^{-6}	200	0.1	0	0.556
funsing54	10^{-6}	200	0.5	0	0.547
funsing54	10^{-6}	200	0.75	0	0.573
funsing54	10^{-6}	200	—	1	0.854
funsing54	10^{-13}	50	0.1	0	0.227
funsing54	10^{-13}	50	0.5	0	0.225
funsing54	10^{-13}	50	0.75	0	0.225
funsing54	10^{-13}	50	—	1	0.298
funsing54	10^{-13}	200	0.1	0	0.575
funsing54	10^{-13}	200	0.5	0	0.632
funsing54	10^{-13}	200	0.75	0	0.642
funsing54	10^{-13}	200	—	1	0.956

Table 2.5: Runtimes for BVP 54 when testing α. Unsurprisingly the results for ZfSensitive = 1 are much less favorable than the others.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing56	10^{-6}	50	0.1	0	0.486
funsing56	10^{-6}	50	0.5	0	0.684
funsing56	10^{-6}	50	0.75	0	0.703
funsing56	10^{-6}	50	—	1	0.561
funsing56	10^{-6}	200	0.1	0	2.364
funsing56	10^{-6}	200	0.5	0	2.772
funsing56	10^{-6}	200	0.75	0	2.748
funsing56	10^{-6}	200	—	1	2.866
funsing56	10^{-13}	50	0.1	0	0.485
funsing56	10^{-13}	50	0.5	0	0.689
funsing56	10^{-13}	50	0.75	0	0.682
funsing56	10^{-13}	50	—	1	0.655
funsing56	10^{-13}	200	0.1	0	4.553
funsing56	10^{-13}	200	0.5	0	2.788
funsing56	10^{-13}	200	0.75	0	2.748
funsing56	10^{-13}	200	—	1	3.476

Table 2.6: Runtimes for BVP 56 when testing α. As this BVP has quite high runtimes compared to the others, the results are especially relevant. $\alpha = \frac{1}{10}$ seems to be superior but just in the hardest configuration it nearly takes twice the time compared to $\alpha = \frac{1}{2}$.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing6001	10^{-6}	50	0.1	0	0.274
funsing6001	10^{-6}	50	0.5	0	0.384
funsing6001	10^{-6}	50	0.75	0	0.370
funsing6001	10^{-6}	50	—	1	0.308
funsing6001	10^{-6}	200	0.1	0	0.929
funsing6001	10^{-6}	200	0.5	0	1.204
funsing6001	10^{-6}	200	0.75	0	1.203
funsing6001	10^{-6}	200	—	1	1.082
funsing6001	10^{-13}	50	0.1	0	0.308
funsing6001	10^{-13}	50	0.5	0	0.376
funsing6001	10^{-13}	50	0.75	0	0.371
funsing6001	10^{-13}	50	—	1	0.346
funsing6001	10^{-13}	200	0.1	0	0.963
funsing6001	10^{-13}	200	0.5	0	1.214
funsing6001	10^{-13}	200	0.75	0	1.207
funsing6001	10^{-13}	200	—	1	1.266

Table 2.7: Runtimes for BVP 6001 when testing α. Here the shortest runtimes are achieved for $\alpha = \frac{1}{10}$. ZfSensitive = 1 can take the second place three times, but it is slowest in the hardest configuration.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing6002	10^{-6}	50	0.1	0	0.194
funsing6002	10^{-6}	50	0.5	0	0.175
funsing6002	10^{-6}	50	0.75	0	0.177
funsing6002	10^{-6}	50	—	1	0.223
funsing6002	10^{-6}	200	0.1	0	0.516
funsing6002	10^{-6}	200	0.5	0	0.453
funsing6002	10^{-6}	200	0.75	0	0.465
funsing6002	10^{-6}	200	—	1	0.664
funsing6002	10^{-13}	50	0.1	0	0.212
funsing6002	10^{-13}	50	0.5	0	0.215
funsing6002	10^{-13}	50	0.75	0	0.221
funsing6002	10^{-13}	50	—	1	0.238
funsing6002	10^{-13}	200	0.1	0	0.550
funsing6002	10^{-13}	200	0.5	0	0.564
funsing6002	10^{-13}	200	0.75	0	0.581
funsing6002	10^{-13}	200	—	1	0.771

Table 2.8: Runtimes for BVP 6002 when testing α. $\alpha = \frac{1}{10}$ and $\alpha = \frac{1}{2}$ show almost the same results with slight advantages for $\frac{1}{2}$. The difference between $\frac{1}{2}$ and $\frac{3}{4}$ is negligible.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing7001b	10^{-6}	50	0.1	0	0.205
funsing7001b	10^{-6}	50	0.5	0	0.198
funsing7001b	10^{-6}	50	0.75	0	0.201
funsing7001b	10^{-6}	50	—	1	0.229
funsing7001b	10^{-6}	200	0.1	0	0.565
funsing7001b	10^{-6}	200	0.5	0	0.514
funsing7001b	10^{-6}	200	0.75	0	0.498
funsing7001b	10^{-6}	200	—	1	0.732
funsing7001b	10^{-13}	50	0.1	0	0.214
funsing7001b	10^{-13}	50	0.5	0	0.266
funsing7001b	10^{-13}	50	0.75	0	0.263
funsing7001b	10^{-13}	50	—	1	0.246
funsing7001b	10^{-13}	200	0.1	0	0.593
funsing7001b	10^{-13}	200	0.5	0	0.701
funsing7001b	10^{-13}	200	0.75	0	0.705
funsing7001b	10^{-13}	200	—	1	0.833

Table 2.9: Runtimes for BVP 7001b when testing α. While the difference between $\frac{1}{2}$ and $\frac{3}{4}$ is beyond the test's accuracy, $\alpha = \frac{1}{10}$ shows better runtimes for the stricter tolerance.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing71	10^{-6}	50	0.1	0	0.207
funsing71	10^{-6}	50	0.5	0	0.224
funsing71	10^{-6}	50	0.75	0	0.165
funsing71	10^{-6}	50	—	1	0.252
funsing71	10^{-6}	200	0.1	0	0.635
funsing71	10^{-6}	200	0.5	0	0.603
funsing71	10^{-6}	200	0.75	0	0.840
funsing71	10^{-6}	200	—	1	0.779
funsing71	10^{-13}	50	0.1	0	0.229
funsing71	10^{-13}	50	0.5	0	0.235
funsing71	10^{-13}	50	0.75	0	0.261
funsing71	10^{-13}	50	—	1	0.263
funsing71	10^{-13}	200	0.1	0	0.681
funsing71	10^{-13}	200	0.5	0	0.704
funsing71	10^{-13}	200	0.75	0	0.935
funsing71	10^{-13}	200	—	1	0.836

Table 2.10: Runtimes for BVP 71 when testing α. The results show to be a head to head for $\alpha = \frac{1}{10}$ and $\alpha = \frac{1}{2}$.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing73	10^{-6}	50	0.1	0	0.970
funsing73	10^{-6}	50	0.5	0	0.923
funsing73	10^{-6}	50	0.75	0	0.949
funsing73	10^{-6}	50	—	1	1.058
funsing73	10^{-6}	200	0.1	0	5.609
funsing73	10^{-6}	200	0.5	0	5.187
funsing73	10^{-6}	200	0.75	0	4.990
funsing73	10^{-6}	200	—	1	6.385
funsing73	10^{-13}	50	0.1	0	0.985
funsing73	10^{-13}	50	0.5	0	0.930
funsing73	10^{-13}	50	0.75	0	0.976
funsing73	10^{-13}	50	—	1	1.128
funsing73	10^{-13}	200	0.1	0	5.655
funsing73	10^{-13}	200	0.5	0	5.185
funsing73	10^{-13}	200	0.75	0	4.993
funsing73	10^{-13}	200	—	1	6.532

Table 2.11: Runtimes for BVP 73 when testing α. The runtimes are higher than in the average tests and again $\frac{1}{2}$ beats $\frac{1}{10}$.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing8001	10^{-6}	50	0.1	0	0.227
funsing8001	10^{-6}	50	0.5	0	0.346
funsing8001	10^{-6}	50	0.75	0	0.332
funsing8001	10^{-6}	50	—	1	0.249
funsing8001	10^{-6}	200	0.1	0	0.711
funsing8001	10^{-6}	200	0.5	0	1.093
funsing8001	10^{-6}	200	0.75	0	1.096
funsing8001	10^{-6}	200	—	1	0.949
funsing8001	10^{-13}	50	0.1	0	0.248
funsing8001	10^{-13}	50	0.5	0	0.351
funsing8001	10^{-13}	50	0.75	0	0.350
funsing8001	10^{-13}	50	—	1	0.284
funsing8001	10^{-13}	200	0.1	0	0.767
funsing8001	10^{-13}	200	0.5	0	1.072
funsing8001	10^{-13}	200	0.75	0	1.086
funsing8001	10^{-13}	200	—	1	1.039

Table 2.12: Runtimes for BVP 8001 when testing α. This test shows obviously better results for $\alpha = \frac{1}{10}$. Here, even ZfSensitive = 1 is competitive.

BVP	Tol	MeshPts.	α	ZfS	Runtime
funsing8002	10^{-6}	50	0.1	0	0.233
funsing8002	10^{-6}	50	0.5	0	0.365
funsing8002	10^{-6}	50	0.75	0	0.348
funsing8002	10^{-6}	50	—	1	0.243
funsing8002	10^{-6}	200	0.1	0	0.627
funsing8002	10^{-6}	200	0.5	0	1.124
funsing8002	10^{-6}	200	0.75	0	1.104
funsing8002	10^{-6}	200	—	1	0.853
funsing8002	10^{-13}	50	0.1	0	0.234
funsing8002	10^{-13}	50	0.5	0	0.362
funsing8002	10^{-13}	50	0.75	0	0.356
funsing8002	10^{-13}	50	—	1	0.275
funsing8002	10^{-13}	200	0.1	0	0.702
funsing8002	10^{-13}	200	0.5	0	1.112
funsing8002	10^{-13}	200	0.75	0	1.135
funsing8002	10^{-13}	200	—	1	0.962

Table 2.13: Runtimes for BVP 8002 when testing α. As for the related BVP 8001 $\alpha = \frac{1}{10}$ is fastest and ZfSensitive = 1 is second.

At this point two major insights can be concluded from the tests:

- $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ produce highly comparable results.
- ZfSensitive = 1 does not lead to the quickest runtime for any BVP.

A universal conclusion regarding the optimal value for α cannot be drawn from these tests. So the next step is tightening the test conditions. The number of meshpoints is increased to 500 and 1000, the tolerance is fixed at 10^{-13}, the degree of collocation is varied and finally all the test configurations are tested for the vectorized and the nonvectorized case, where vectorization means that the BVP is given in a vectorized form. For further information refer to [5].

The first thing apparent from the following tables (2.14 - 2.91) is that the above two insights are confirmed. When ZfSensitive is set to 1 the runtime is slowest in nearly all cases and never is fastest, and the runtimes for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are nearly the same for most of the BVPS. There are only two BVPS with a noticeable gap between the runtimes of these settings for α. BVP 73 is much better for $\alpha = \frac{3}{4}$ (cf. Tables 2.44 - 2.46 and 2.83 - 2.85). The other one is BVP 71 (cf. Tables 2.41 - 2.43 and 2.80 - 2.82) with $\alpha = \frac{1}{2}$ yielding better results. As both BVPS are regular, they are not a main target of SBVP 2.0 which are singular BVPS, cf. Section 1.4.

As the runtime for $\alpha = \frac{1}{2}$ is about 50 % the runtime for $\alpha = \frac{3}{4}$ for BVP 71 and otherwise the runtime for $\alpha = \frac{3}{4}$ is about 75 % - 90 % the runtime for $\alpha = \frac{1}{2}$ the first decision was to prefer $\alpha = \frac{1}{2}$, especially since both values are quite comparable in all the other configurations.

The results for the non-vectorized case show an advantage for α set to $\frac{1}{10}$. The biggest gain is observed for collocation order 4, and it is getting lower for higher collocation order. Also with 500 meshpoints the results for $\alpha = \frac{1}{10}$ are relatively better than with 1000 meshpoints. So it seems that the harder the test configuration the better the tests for $\alpha = \frac{1}{2}$. But even the configuration with 1000 meshpoints and collocation order 8 yields faster runtimes for $\alpha = \frac{1}{10}$ in some cases.

For the vectorized case the picture is quite different. Firstly, the vectorized implementation is preferable due to higher efficiency. So the runtimes are very short compared to the non-vectorized case. Here, $\alpha = \frac{1}{2}$ is better than $\alpha = \frac{1}{10}$ in most of the tests.

Of course there are also counterexamples where $\alpha = \frac{1}{2}$ does not yield the best results (e. g. Table 2.22 or Table 2.69) but the overall impression is as discussed above. Finally, the choice between two basic strategies had to be made:

1. Quick solutions in the (recommended) vectorized case favor $\alpha = \frac{1}{2}$. On the other hand the quite slow runtimes for the non-vectorized case – which is easier to encode – are not very user-friendly.
2. Quicker solutions in the non-vectorized case and therefore slower solutions in the vectorized case. This can be achieved with $\alpha = \frac{1}{10}$. The runtimes in both cases would be balanced but the user is deprived of the possibility to get high performance for recommended usage.

Figure 2.1 is helpful for this difficult decision. It can be seen that the vectorized case is much quicker than the non-vectorized case in every single configuration. And because it is

recommended in the documentation of SBVP 1.0 the decision is to support the vectorized case and therefore α is set to $\frac{1}{2}$.

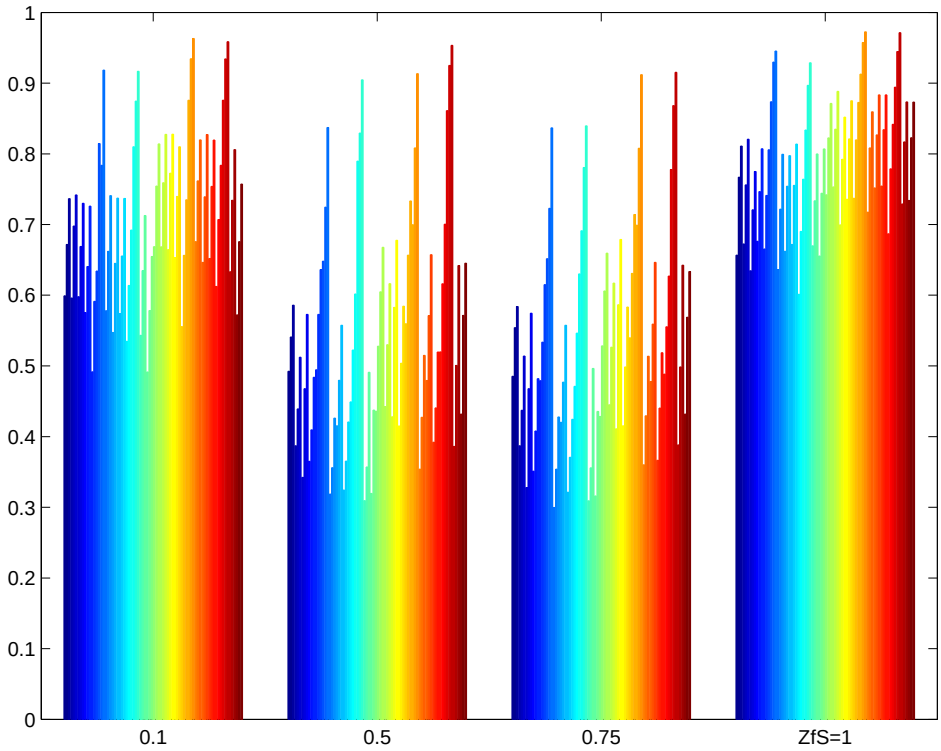


Figure 2.1: This figure shows the relative runtime of the vectorized BVP compared to the non-vectorized BVP ($\frac{t_{vec}}{t_{nonvec}}$). It covers all configurations from Tables 2.14 - 2.91 and it can be seen – as all bars are lower than 1 – that the vectorized case is quicker in every configuration.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing0026	10^{-13}	500	0	4	0.1	0	2.574
funsing0026	10^{-13}	500	0	4	0.5	0	3.985
funsing0026	10^{-13}	500	0	4	0.75	0	4.045
funsing0026	10^{-13}	500	0	4	—	1	3.355
funsing0026	10^{-13}	500	1	4	0.1	0	1.542
funsing0026	10^{-13}	500	1	4	0.5	0	1.962
funsing0026	10^{-13}	500	1	4	0.75	0	1.963
funsing0026	10^{-13}	500	1	4	—	1	2.203

Table 2.14: Runtimes for BVP 0026 when testing α with 500 MP and collocation order 4. The quickest runtime is achieved for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing0026	10^{-13}	500	0	6	0.1	0	4.667
funsing0026	10^{-13}	500	0	6	0.5	0	6.740
funsing0026	10^{-13}	500	0	6	0.75	0	6.579
funsing0026	10^{-13}	500	0	6	—	1	6.359
funsing0026	10^{-13}	500	1	6	0.1	0	3.135
funsing0026	10^{-13}	500	1	6	0.5	0	3.646
funsing0026	10^{-13}	500	1	6	0.75	0	3.646
funsing0026	10^{-13}	500	1	6	—	1	4.877

Table 2.15: Runtimes for BVP 0026 when testing α with 500 MP and collocation order 6. The quickest time is achieved by $\alpha = \frac{1}{10}$ in both cases, the vectorized and the non-vectorized one.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing0026	10^{-13}	500	0	8	0.1	0	7.681
funsing0026	10^{-13}	500	0	8	0.5	0	10.015
funsing0026	10^{-13}	500	0	8	0.75	0	10.014
funsing0026	10^{-13}	500	0	8	—	1	10.966
funsing0026	10^{-13}	500	1	8	0.1	0	5.658
funsing0026	10^{-13}	500	1	8	0.5	0	5.869
funsing0026	10^{-13}	500	1	8	0.75	0	5.848
funsing0026	10^{-13}	500	1	8	—	1	8.893

Table 2.16: Runtimes for BVP 0026 when testing α with 500 MP and collocation order 8. While $\alpha = \frac{1}{10}$ is much quicker in the non-vectorized case, the runtimes for the vectorized case are quite similar. ZfSensitive = 1 is slowest.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing1a	10^{-13}	500	0	4	0.1	0	3.075
funsing1a	10^{-13}	500	0	4	0.5	0	3.725
funsing1a	10^{-13}	500	0	4	0.75	0	3.725
funsing1a	10^{-13}	500	0	4	—	1	4.005
funsing1a	10^{-13}	500	1	4	0.1	0	1.833
funsing1a	10^{-13}	500	1	4	0.5	0	1.442
funsing1a	10^{-13}	500	1	4	0.75	0	1.442
funsing1a	10^{-13}	500	1	4	—	1	2.694

Table 2.17: Runtimes for BVP 1a when testing α with 500 MP and collocation order 4. $\alpha = \frac{1}{10}$ is best in the non-vectorized case, $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are best for the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing1a	10^{-13}	500	0	6	0.1	0	5.769
funsing1a	10^{-13}	500	0	6	0.5	0	6.089
funsing1a	10^{-13}	500	0	6	0.75	0	6.089
funsing1a	10^{-13}	500	0	6	—	1	7.882
funsing1a	10^{-13}	500	1	6	0.1	0	4.026
funsing1a	10^{-13}	500	1	6	0.5	0	2.674
funsing1a	10^{-13}	500	1	6	0.75	0	2.663
funsing1a	10^{-13}	500	1	6	—	1	5.959

Table 2.18: Runtimes for BVP 1a when testing α with 500 MP and collocation order 6. While the runtime is slightly higher in the non-vectorized case for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$, these two configurations are much quicker in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing1a	10^{-13}	500	0	8	0.1	0	9.825
funsing1a	10^{-13}	500	0	8	0.5	0	9.304
funsing1a	10^{-13}	500	0	8	0.75	0	9.293
funsing1a	10^{-13}	500	0	8	—	1	14.070
funsing1a	10^{-13}	500	1	8	0.1	0	7.290
funsing1a	10^{-13}	500	1	8	0.5	0	4.767
funsing1a	10^{-13}	500	1	8	0.75	0	4.777
funsing1a	10^{-13}	500	1	8	—	1	11.546

Table 2.19: Runtimes for BVP 1a when testing α with 500 MP and collocation order 8. $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are quickest for these configurations.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing21a	10^{-13}	500	0	4	0.1	0	3.835
funsing21a	10^{-13}	500	0	4	0.5	0	8.863
funsing21a	10^{-13}	500	0	4	0.75	0	8.673
funsing21a	10^{-13}	500	0	4	—	1	4.877
funsing21a	10^{-13}	500	1	4	0.1	0	2.293
funsing21a	10^{-13}	500	1	4	0.5	0	3.035
funsing21a	10^{-13}	500	1	4	0.75	0	2.844
funsing21a	10^{-13}	500	1	4	—	1	3.095

Table 2.20: Runtimes for BVP 21a when testing α with 500 MP and collocation order 4. This BVP is solved in shortest time for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing21a	10^{-13}	500	0	6	0.1	0	7.080
funsing21a	10^{-13}	500	0	6	0.5	0	17.766
funsing21a	10^{-13}	500	0	6	0.75	0	17.765
funsing21a	10^{-13}	500	0	6	—	1	9.183
funsing21a	10^{-13}	500	1	6	0.1	0	4.737
funsing21a	10^{-13}	500	1	6	0.5	0	8.312
funsing21a	10^{-13}	500	1	6	0.75	0	8.312
funsing21a	10^{-13}	500	1	6	—	1	6.619

Table 2.21: Runtimes for BVP 21a when testing α with 500 MP and collocation order 6. Best results are achieved for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing21a	10^{-13}	500	0	8	0.1	0	12.207
funsing21a	10^{-13}	500	0	8	0.5	0	31.486
funsing21a	10^{-13}	500	0	8	0.75	0	31.466
funsing21a	10^{-13}	500	0	8	—	1	15.702
funsing21a	10^{-13}	500	1	8	0.1	0	8.913
funsing21a	10^{-13}	500	1	8	0.5	0	18.035
funsing21a	10^{-13}	500	1	8	0.75	0	18.076
funsing21a	10^{-13}	500	1	8	—	1	12.168

Table 2.22: Runtimes for BVP 21a when testing α with 500 MP and collocation order 8. As in Tables 2.20 and 2.21 this BVP is solved most efficiently for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing37c	10^{-13}	500	0	4	0.1	0	3.255
funsing37c	10^{-13}	500	0	4	0.5	0	4.086
funsing37c	10^{-13}	500	0	4	0.75	0	4.106
funsing37c	10^{-13}	500	0	4	—	1	4.116
funsing37c	10^{-13}	500	1	4	0.1	0	1.873
funsing37c	10^{-13}	500	1	4	0.5	0	1.492
funsing37c	10^{-13}	500	1	4	0.75	0	1.442
funsing37c	10^{-13}	500	1	4	—	1	2.784

Table 2.23: Runtimes for BVP 37c when testing α with 500 MP and collocation order 4. The vectorized case shows advantages for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing37c	10^{-13}	500	0	6	0.1	0	6.019
funsing37c	10^{-13}	500	0	6	0.5	0	6.650
funsing37c	10^{-13}	500	0	6	0.75	0	6.650
funsing37c	10^{-13}	500	0	6	—	1	8.062
funsing37c	10^{-13}	500	1	6	0.1	0	3.856
funsing37c	10^{-13}	500	1	6	0.5	0	2.724
funsing37c	10^{-13}	500	1	6	0.75	0	2.713
funsing37c	10^{-13}	500	1	6	—	1	6.019

Table 2.24: Runtimes for BVP 37c when testing α with 500 MP and collocation order 6. $\alpha = \frac{1}{10}$ is better for the non-vectorized case but $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are better for the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing37c	10^{-13}	500	0	8	0.1	0	10.124
funsing37c	10^{-13}	500	0	8	0.5	0	10.014
funsing37c	10^{-13}	500	0	8	0.75	0	10.035
funsing37c	10^{-13}	500	0	8	—	1	14.390
funsing37c	10^{-13}	500	1	8	0.1	0	7.351
funsing37c	10^{-13}	500	1	8	0.5	0	4.847
funsing37c	10^{-13}	500	1	8	0.75	0	4.837
funsing37c	10^{-13}	500	1	8	—	1	11.617

Table 2.25: Runtimes for BVP 37c when testing α with 500 MP and collocation order 8. Here $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are even better for the non-vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing54	10^{-13}	500	0	4	0.1	0	3.565
funsing54	10^{-13}	500	0	4	0.5	0	3.565
funsing54	10^{-13}	500	0	4	0.75	0	3.825
funsing54	10^{-13}	500	0	4	—	1	4.546
funsing54	10^{-13}	500	1	4	0.1	0	1.753
funsing54	10^{-13}	500	1	4	0.5	0	1.763
funsing54	10^{-13}	500	1	4	0.75	0	1.833
funsing54	10^{-13}	500	1	4	—	1	3.025

Table 2.26: Runtimes for BVP 54 when testing α with 500 MP and collocation order 4. ZfSensitive = 1 shows the worst results and $\alpha = \frac{3}{4}$ is noticeably slower than $\alpha = \frac{1}{2}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing54	10^{-13}	500	0	6	0.1	0	6.199
funsing54	10^{-13}	500	0	6	0.5	0	6.118
funsing54	10^{-13}	500	0	6	0.75	0	6.719
funsing54	10^{-13}	500	0	6	—	1	8.783
funsing54	10^{-13}	500	1	6	0.1	0	3.666
funsing54	10^{-13}	500	1	6	0.5	0	3.505
funsing54	10^{-13}	500	1	6	0.75	0	3.585
funsing54	10^{-13}	500	1	6	—	1	6.510

Table 2.27: Runtimes for BVP 54 when testing α with 500 MP and collocation order 6. $\alpha = \frac{1}{10}$ and $\alpha = \frac{1}{2}$ show quite similar results.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing54	10^{-13}	500	0	8	0.1	0	9.884
funsing54	10^{-13}	500	0	8	0.5	0	9.914
funsing54	10^{-13}	500	0	8	0.75	0	10.455
funsing54	10^{-13}	500	0	8	—	1	15.152
funsing54	10^{-13}	500	1	8	0.1	0	6.269
funsing54	10^{-13}	500	1	8	0.5	0	6.309
funsing54	10^{-13}	500	1	8	0.75	0	6.429
funsing54	10^{-13}	500	1	8	—	1	12.208

Table 2.28: Runtimes for BVP 54 when testing α with 500 MP and collocation order 8. Typically for BVP 54 the results for $\alpha = \frac{1}{10}$ and $\alpha = \frac{1}{2}$ are comparable.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing56	10^{-13}	500	0	4	0.1	0	13.549
funsing56	10^{-13}	500	0	4	0.5	0	19.218
funsing56	10^{-13}	500	0	4	0.75	0	19.178
funsing56	10^{-13}	500	0	4	—	1	16.644
funsing56	10^{-13}	500	1	4	0.1	0	11.036
funsing56	10^{-13}	500	1	4	0.5	0	12.458
funsing56	10^{-13}	500	1	4	0.75	0	12.498
funsing56	10^{-13}	500	1	4	—	1	14.541

Table 2.29: Runtimes for BVP 56 when testing α with 500 MP and collocation order 4. The results for $\alpha = \frac{1}{10}$ are best, but in the vectorized situation the gap between the different settings is smaller.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing56	10^{-13}	500	0	6	0.1	0	49.230
funsing56	10^{-13}	500	0	6	0.5	0	35.601
funsing56	10^{-13}	500	0	6	0.75	0	35.751
funsing56	10^{-13}	500	0	6	—	1	39.296
funsing56	10^{-13}	500	1	6	0.1	0	38.585
funsing56	10^{-13}	500	1	6	0.5	0	25.797
funsing56	10^{-13}	500	1	6	0.75	0	25.847
funsing56	10^{-13}	500	1	6	—	1	36.542

Table 2.30: Runtimes for BVP 56 when testing α with 500 MP and collocation order 6. Surprisingly the results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are much better than for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing56	10^{-13}	500	0	8	0.1	0	53.998
funsing56	10^{-13}	500	0	8	0.5	0	84.381
funsing56	10^{-13}	500	0	8	0.75	0	84.341
funsing56	10^{-13}	500	0	8	—	1	86.074
funsing56	10^{-13}	500	1	8	0.1	0	49.601
funsing56	10^{-13}	500	1	8	0.5	0	70.641
funsing56	10^{-13}	500	1	8	0.75	0	70.581
funsing56	10^{-13}	500	1	8	—	1	81.387

Table 2.31: Runtimes for BVP 56 when testing α with 500 MP and collocation order 8. Here, $\alpha = \frac{1}{10}$ is the best choice.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6001	10^{-13}	500	0	4	0.1	0	5.127
funsing6001	10^{-13}	500	0	4	0.5	0	9.303
funsing6001	10^{-13}	500	0	4	0.75	0	9.324
funsing6001	10^{-13}	500	0	4	—	1	6.720
funsing6001	10^{-13}	500	1	4	0.1	0	2.964
funsing6001	10^{-13}	500	1	4	0.5	0	2.965
funsing6001	10^{-13}	500	1	4	0.75	0	2.794
funsing6001	10^{-13}	500	1	4	—	1	4.276

Table 2.32: Runtimes for BVP 6001 when testing α with 500 MP and collocation order 4. The good results for $\alpha = \frac{1}{10}$ are striking in the non-vectorized case but in the vectorized case the results are balanced, except for ZfSensitive = 1.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6001	10^{-13}	500	0	6	0.1	0	9.724
funsing6001	10^{-13}	500	0	6	0.5	0	14.991
funsing6001	10^{-13}	500	0	6	0.75	0	14.992
funsing6001	10^{-13}	500	0	6	—	1	13.169
funsing6001	10^{-13}	500	1	6	0.1	0	6.439
funsing6001	10^{-13}	500	1	6	0.5	0	5.337
funsing6001	10^{-13}	500	1	6	0.75	0	5.308
funsing6001	10^{-13}	500	1	6	—	1	9.504

Table 2.33: Runtimes for BVP 6001 when testing α with 500 MP and collocation order 6. At the higher collocation order $\alpha = \frac{1}{10}$ loses ground in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6001	10^{-13}	500	0	8	0.1	0	16.734
funsing6001	10^{-13}	500	0	8	0.5	0	22.422
funsing6001	10^{-13}	500	0	8	0.75	0	22.422
funsing6001	10^{-13}	500	0	8	—	1	23.404
funsing6001	10^{-13}	500	1	8	0.1	0	12.398
funsing6001	10^{-13}	500	1	8	0.5	0	9.554
funsing6001	10^{-13}	500	1	8	0.75	0	9.594
funsing6001	10^{-13}	500	1	8	—	1	18.707

Table 2.34: Runtimes for BVP 6001 when testing α with 500 MP and collocation order 8. Compared to the results of Table 2.33 for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ in the vectorized case, the results are even better here.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6002	10^{-13}	500	0	4	0.1	0	2.744
funsing6002	10^{-13}	500	0	4	0.5	0	3.665
funsing6002	10^{-13}	500	0	4	0.75	0	3.645
funsing6002	10^{-13}	500	0	4	—	1	3.646
funsing6002	10^{-13}	500	1	4	0.1	0	1.502
funsing6002	10^{-13}	500	1	4	0.5	0	1.522
funsing6002	10^{-13}	500	1	4	0.75	0	1.532
funsing6002	10^{-13}	500	1	4	—	1	2.414

Table 2.35: Runtimes for BVP 6002 when testing α with 500 MP and collocation order 4. The vectorized runtimes are quite balanced but again $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are much slower in the non-vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6002	10^{-13}	500	0	6	0.1	0	5.028
funsing6002	10^{-13}	500	0	6	0.5	0	6.139
funsing6002	10^{-13}	500	0	6	0.75	0	6.109
funsing6002	10^{-13}	500	0	6	—	1	7.211
funsing6002	10^{-13}	500	1	6	0.1	0	3.244
funsing6002	10^{-13}	500	1	6	0.5	0	2.944
funsing6002	10^{-13}	500	1	6	0.75	0	2.914
funsing6002	10^{-13}	500	1	6	—	1	5.438

Table 2.36: Runtimes for BVP 6002 when testing α with 500 MP and collocation order 6. $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are best for the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6002	10^{-13}	500	0	8	0.1	0	8.462
funsing6002	10^{-13}	500	0	8	0.5	0	9.594
funsing6002	10^{-13}	500	0	8	0.75	0	9.594
funsing6002	10^{-13}	500	0	8	—	1	13.158
funsing6002	10^{-13}	500	1	8	0.1	0	6.239
funsing6002	10^{-13}	500	1	8	0.5	0	5.348
funsing6002	10^{-13}	500	1	8	0.75	0	5.348
funsing6002	10^{-13}	500	1	8	—	1	10.495

Table 2.37: Runtimes for BVP 6002 when testing α with 500 MP and collocation order 8. The higher the collocation order the better the runtimes for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing7001b	10^{-13}	500	0	4	0.1	0	3.034
funsing7001b	10^{-13}	500	0	4	0.5	0	5.498
funsing7001b	10^{-13}	500	0	4	0.75	0	5.488
funsing7001b	10^{-13}	500	0	4	—	1	3.965
funsing7001b	10^{-13}	500	1	4	0.1	0	1.742
funsing7001b	10^{-13}	500	1	4	0.5	0	1.783
funsing7001b	10^{-13}	500	1	4	0.75	0	1.763
funsing7001b	10^{-13}	500	1	4	—	1	2.664

Table 2.38: Runtimes for BVP 7001b when testing α with 500 MP and collocation order 4. $\alpha = \frac{1}{10}$ is better for the non-vectorized case whereas $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are competitive in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing7001b	10^{-13}	500	0	6	0.1	0	5.728
funsing7001b	10^{-13}	500	0	6	0.5	0	8.773
funsing7001b	10^{-13}	500	0	6	0.75	0	8.752
funsing7001b	10^{-13}	500	0	6	—	1	7.862
funsing7001b	10^{-13}	500	1	6	0.1	0	3.755
funsing7001b	10^{-13}	500	1	6	0.5	0	3.204
funsing7001b	10^{-13}	500	1	6	0.75	0	3.245
funsing7001b	10^{-13}	500	1	6	—	1	5.938

Table 2.39: Runtimes for BVP 7001b when testing α with 500 MP and collocation order 6. Compared to the test with collocation order 4 the results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ have improved here.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing7001b	10^{-13}	500	0	8	0.1	0	9.744
funsing7001b	10^{-13}	500	0	8	0.5	0	12.898
funsing7001b	10^{-13}	500	0	8	0.75	0	12.928
funsing7001b	10^{-13}	500	0	8	—	1	14.151
funsing7001b	10^{-13}	500	1	8	0.1	0	7.180
funsing7001b	10^{-13}	500	1	8	0.5	0	5.427
funsing7001b	10^{-13}	500	1	8	0.75	0	5.488
funsing7001b	10^{-13}	500	1	8	—	1	11.517

Table 2.40: Runtimes for BVP 7001b when testing α with 500 MP and collocation order 8. For the high collocation order, $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are favorable in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing71	10^{-13}	500	0	4	0.1	0	3.746
funsing71	10^{-13}	500	0	4	0.5	0	4.416
funsing71	10^{-13}	500	0	4	0.75	0	6.930
funsing71	10^{-13}	500	0	4	—	1	4.397
funsing71	10^{-13}	500	1	4	0.1	0	2.003
funsing71	10^{-13}	500	1	4	0.5	0	1.983
funsing71	10^{-13}	500	1	4	0.75	0	3.265
funsing71	10^{-13}	500	1	4	—	1	2.644

Table 2.41: Runtimes for BVP 71 when testing α with 500 MP and collocation order 4. This regular BVP is the first one where $\alpha = \frac{3}{4}$ is much slower than $\alpha = \frac{1}{2}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing71	10^{-13}	500	0	6	0.1	0	6.770
funsing71	10^{-13}	500	0	6	0.5	0	7.591
funsing71	10^{-13}	500	0	6	0.75	0	12.077
funsing71	10^{-13}	500	0	6	—	1	8.312
funsing71	10^{-13}	500	1	6	0.1	0	4.156
funsing71	10^{-13}	500	1	6	0.5	0	3.966
funsing71	10^{-13}	500	1	6	0.75	0	6.600
funsing71	10^{-13}	500	1	6	—	1	5.738

Table 2.42: Runtimes for BVP 71 when testing α with 500 MP and collocation order 6. The very slow runtimes for $\alpha = \frac{3}{4}$ are striking.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing71	10^{-13}	500	0	8	0.1	0	11.356
funsing71	10^{-13}	500	0	8	0.5	0	12.248
funsing71	10^{-13}	500	0	8	0.75	0	19.769
funsing71	10^{-13}	500	0	8	—	1	14.431
funsing71	10^{-13}	500	1	8	0.1	0	7.861
funsing71	10^{-13}	500	1	8	0.5	0	7.371
funsing71	10^{-13}	500	1	8	0.75	0	12.458
funsing71	10^{-13}	500	1	8	—	1	11.026

Table 2.43: Runtimes for BVP 71 when testing α with 500 MP and collocation order 8. $\alpha = \frac{3}{4}$ shows the worst results. And again $\alpha = \frac{1}{2}$ is quickest for the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing73	10^{-13}	500	0	4	0.1	0	25.737
funsing73	10^{-13}	500	0	4	0.5	0	25.176
funsing73	10^{-13}	500	0	4	0.75	0	23.784
funsing73	10^{-13}	500	0	4	—	1	29.753
funsing73	10^{-13}	500	1	4	0.1	0	20.850
funsing73	10^{-13}	500	1	4	0.5	0	19.879
funsing73	10^{-13}	500	1	4	0.75	0	16.443
funsing73	10^{-13}	500	1	4	—	1	24.806

Table 2.44: Runtimes for BVP 73 when testing α with 500 MP and collocation order 4. This four-dimensional regular BVP shows a reversed picture from BVP 71.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing73	10^{-13}	500	0	6	0.1	0	59.646
funsing73	10^{-13}	500	0	6	0.5	0	60.056
funsing73	10^{-13}	500	0	6	0.75	0	52.165
funsing73	10^{-13}	500	0	6	—	1	69.901
funsing73	10^{-13}	500	1	6	0.1	0	52.175
funsing73	10^{-13}	500	1	6	0.5	0	49.812
funsing73	10^{-13}	500	1	6	0.75	0	40.728
funsing73	10^{-13}	500	1	6	—	1	62.690

Table 2.45: Runtimes for BVP 73 when testing α with 500 MP and collocation order 6. $\alpha = \frac{3}{4}$ is best for both cases.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing73	10^{-13}	500	0	8	0.1	0	122.005
funsing73	10^{-13}	500	0	8	0.5	0	116.557
funsing73	10^{-13}	500	0	8	0.75	0	96.309
funsing73	10^{-13}	500	0	8	—	1	146.791
funsing73	10^{-13}	500	1	8	0.1	0	111.841
funsing73	10^{-13}	500	1	8	0.5	0	105.452
funsing73	10^{-13}	500	1	8	0.75	0	80.876
funsing73	10^{-13}	500	1	8	—	1	136.336

Table 2.46: Runtimes for BVP 73 when testing α with 500 MP and collocation order 8. Again the big advantage for $\alpha = \frac{3}{4}$ is striking.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8001	10^{-13}	500	0	4	0.1	0	4.146
funsing8001	10^{-13}	500	0	4	0.5	0	9.063
funsing8001	10^{-13}	500	0	4	0.75	0	9.133
funsing8001	10^{-13}	500	0	4	—	1	5.337
funsing8001	10^{-13}	500	1	4	0.1	0	2.253
funsing8001	10^{-13}	500	1	4	0.5	0	2.804
funsing8001	10^{-13}	500	1	4	0.75	0	2.824
funsing8001	10^{-13}	500	1	4	—	1	3.575

Table 2.47: Runtimes for BVP 8001 when testing α with 500 MP and collocation order 4. Obviously the high runtimes in the non-vectorized case are favoring $\alpha = \frac{1}{10}$, but the differences are smaller in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8001	10^{-13}	500	0	6	0.1	0	7.681
funsing8001	10^{-13}	500	0	6	0.5	0	14.581
funsing8001	10^{-13}	500	0	6	0.75	0	14.661
funsing8001	10^{-13}	500	0	6	—	1	10.635
funsing8001	10^{-13}	500	1	6	0.1	0	4.877
funsing8001	10^{-13}	500	1	6	0.5	0	5.207
funsing8001	10^{-13}	500	1	6	0.75	0	5.217
funsing8001	10^{-13}	500	1	6	—	1	7.801

Table 2.48: Runtimes for BVP 8001 when testing α with 500 MP and collocation order 6. Nearly twice the time is needed in the non-vectorized case for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8001	10^{-13}	500	0	8	0.1	0	13.109
funsing8001	10^{-13}	500	0	8	0.5	0	25.337
funsing8001	10^{-13}	500	0	8	0.75	0	25.616
funsing8001	10^{-13}	500	0	8	—	1	19.017
funsing8001	10^{-13}	500	1	8	0.1	0	9.343
funsing8001	10^{-13}	500	1	8	0.5	0	12.438
funsing8001	10^{-13}	500	1	8	0.75	0	12.719
funsing8001	10^{-13}	500	1	8	—	1	15.212

Table 2.49: Runtimes for BVP 8001 when testing α with 500 MP and collocation order 8. Appallingly the runtimes for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are about 30 % worse in the vectorized case, whereas they only were about 10 % worse for collocation order 6 as can be seen in Table 2.48.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8002	10^{-13}	500	0	4	0.1	0	4.156
funsing8002	10^{-13}	500	0	4	0.5	0	9.053
funsing8002	10^{-13}	500	0	4	0.75	0	9.063
funsing8002	10^{-13}	500	0	4	—	1	4.797
funsing8002	10^{-13}	500	1	4	0.1	0	2.043
funsing8002	10^{-13}	500	1	4	0.5	0	2.894
funsing8002	10^{-13}	500	1	4	0.75	0	2.865
funsing8002	10^{-13}	500	1	4	—	1	3.144

Table 2.50: Runtimes for BVP 8002 when testing α with 500 MP and collocation order 4. As BVP 8002 is closely related to BVP 8001 the results are quite similar to those in Table 2.47.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8002	10^{-13}	500	0	6	0.1	0	7.481
funsing8002	10^{-13}	500	0	6	0.5	0	17.445
funsing8002	10^{-13}	500	0	6	0.75	0	17.475
funsing8002	10^{-13}	500	0	6	—	1	9.473
funsing8002	10^{-13}	500	1	6	0.1	0	4.326
funsing8002	10^{-13}	500	1	6	0.5	0	7.631
funsing8002	10^{-13}	500	1	6	0.75	0	7.611
funsing8002	10^{-13}	500	1	6	—	1	7.050

Table 2.51: Runtimes for BVP 8002 when testing α with 500 MP and collocation order 6. Even ZfSensitive = 1 is faster in the vectorized case than $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8002	10^{-13}	500	0	8	0.1	0	12.328
funsing8002	10^{-13}	500	0	8	0.5	0	21.901
funsing8002	10^{-13}	500	0	8	0.75	0	22.132
funsing8002	10^{-13}	500	0	8	—	1	16.914
funsing8002	10^{-13}	500	1	8	0.1	0	8.071
funsing8002	10^{-13}	500	1	8	0.5	0	9.544
funsing8002	10^{-13}	500	1	8	0.75	0	9.484
funsing8002	10^{-13}	500	1	8	—	1	13.650

Table 2.52: Runtimes for BVP 8002 when testing α with 500 MP and collocation order 8. Again $\alpha = \frac{1}{10}$ is the quickest option.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing0026	10^{-13}	1000	0	4	0.1	0	6.079
funsing0026	10^{-13}	1000	0	4	0.5	0	8.823
funsing0026	10^{-13}	1000	0	4	0.75	0	8.813
funsing0026	10^{-13}	1000	0	4	—	1	8.272
funsing0026	10^{-13}	1000	1	4	0.1	0	4.066
funsing0026	10^{-13}	1000	1	4	0.5	0	4.657
funsing0026	10^{-13}	1000	1	4	0.75	0	4.657
funsing0026	10^{-13}	1000	1	4	—	1	6.139

Table 2.53: Runtimes for BVP 0026 when testing α with 1000 MP and collocation order 4. The best times are achieved for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing0026	10^{-13}	1000	0	6	0.1	0	12.197
funsing0026	10^{-13}	1000	0	6	0.5	0	15.793
funsing0026	10^{-13}	1000	0	6	0.75	0	15.833
funsing0026	10^{-13}	1000	0	6	—	1	17.626
funsing0026	10^{-13}	1000	1	6	0.1	0	9.203
funsing0026	10^{-13}	1000	1	6	0.5	0	9.554
funsing0026	10^{-13}	1000	1	6	0.75	0	9.594
funsing0026	10^{-13}	1000	1	6	—	1	14.500

Table 2.54: Runtimes for BVP 0026 when testing α with 1000 MP and collocation order 6. Again the best times are achieved for $\alpha = \frac{1}{10}$, but the gap in the vectorized case is smaller.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing0026	10^{-13}	1000	0	8	0.1	0	21.771
funsing0026	10^{-13}	1000	0	8	0.5	0	24.996
funsing0026	10^{-13}	1000	0	8	0.75	0	25.297
funsing0026	10^{-13}	1000	0	8	—	1	32.938
funsing0026	10^{-13}	1000	1	8	0.1	0	17.725
funsing0026	10^{-13}	1000	1	8	0.5	0	16.694
funsing0026	10^{-13}	1000	1	8	0.75	0	16.684
funsing0026	10^{-13}	1000	1	8	—	1	28.701

Table 2.55: Runtimes for BVP 0026 when testing α with 1000 MP and collocation order 8. $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are best in the vectorized case. That differs from the equivalent test with 500 meshpoints, cf. Table 2.16.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing1a	10^{-13}	1000	0	4	0.1	0	7.610
funsing1a	10^{-13}	1000	0	4	0.5	0	8.192
funsing1a	10^{-13}	1000	0	4	0.75	0	8.212
funsing1a	10^{-13}	1000	0	4	—	1	10.395
funsing1a	10^{-13}	1000	1	4	0.1	0	5.088
funsing1a	10^{-13}	1000	1	4	0.5	0	3.625
funsing1a	10^{-13}	1000	1	4	0.75	0	3.655
funsing1a	10^{-13}	1000	1	4	—	1	7.831

Table 2.56: Runtimes for BVP 1a when testing α with 1000 MP and collocation order 4. The best times in the vectorized case are achieved for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing1a	10^{-13}	1000	0	6	0.1	0	15.723
funsing1a	10^{-13}	1000	0	6	0.5	0	14.451
funsing1a	10^{-13}	1000	0	6	0.75	0	14.491
funsing1a	10^{-13}	1000	0	6	—	1	23.033
funsing1a	10^{-13}	1000	1	6	0.1	0	11.937
funsing1a	10^{-13}	1000	1	6	0.5	0	7.661
funsing1a	10^{-13}	1000	1	6	0.75	0	7.630
funsing1a	10^{-13}	1000	1	6	—	1	19.228

Table 2.57: Runtimes for BVP 1a when testing α with 1000 MP and collocation order 6. The higher the collocation order the better the results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing1a	10^{-13}	1000	0	8	0.1	0	29.062
funsing1a	10^{-13}	1000	0	8	0.5	0	23.714
funsing1a	10^{-13}	1000	0	8	0.75	0	23.644
funsing1a	10^{-13}	1000	0	8	—	1	44.404
funsing1a	10^{-13}	1000	1	8	0.1	0	24.054
funsing1a	10^{-13}	1000	1	8	0.5	0	14.621
funsing1a	10^{-13}	1000	1	8	0.75	0	14.591
funsing1a	10^{-13}	1000	1	8	—	1	39.447

Table 2.58: Runtimes for BVP 1a when testing α with 1000 MP and collocation order 8. This test shows an outstanding performance for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing21a	10^{-13}	1000	0	4	0.1	0	9.343
funsing21a	10^{-13}	1000	0	4	0.5	0	20.800
funsing21a	10^{-13}	1000	0	4	0.75	0	20.680
funsing21a	10^{-13}	1000	0	4	—	1	12.038
funsing21a	10^{-13}	1000	1	4	0.1	0	6.209
funsing21a	10^{-13}	1000	1	4	0.5	0	8.903
funsing21a	10^{-13}	1000	1	4	0.75	0	8.502
funsing21a	10^{-13}	1000	1	4	—	1	8.422

Table 2.59: Runtimes for BVP 21a when testing α with 1000 MP and collocation order 4. Similar to Table 2.20, the results are very good for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing21a	10^{-13}	1000	0	6	0.1	0	19.038
funsing21a	10^{-13}	1000	0	6	0.5	0	48.680
funsing21a	10^{-13}	1000	0	6	0.75	0	48.560
funsing21a	10^{-13}	1000	0	6	—	1	24.986
funsing21a	10^{-13}	1000	1	6	0.1	0	14.712
funsing21a	10^{-13}	1000	1	6	0.5	0	28.371
funsing21a	10^{-13}	1000	1	6	0.75	0	28.481
funsing21a	10^{-13}	1000	1	6	—	1	19.799

Table 2.60: Runtimes for BVP 21a when testing α with 1000 MP and collocation order 6. $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ perform very badly for this BVP.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing21a	10^{-13}	1000	0	8	0.1	0	35.341
funsing21a	10^{-13}	1000	0	8	0.5	0	86.585
funsing21a	10^{-13}	1000	0	8	0.75	0	86.414
funsing21a	10^{-13}	1000	0	8	—	1	46.777
funsing21a	10^{-13}	1000	1	8	0.1	0	29.252
funsing21a	10^{-13}	1000	1	8	0.5	0	58.684
funsing21a	10^{-13}	1000	1	8	0.75	0	58.674
funsing21a	10^{-13}	1000	1	8	—	1	39.857

Table 2.61: Runtimes for BVP 21a when testing α with 1000 MP and collocation order 8. The runtimes of $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are very slow, even in the vectorized case they take twice as long as for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing37c	10^{-13}	1000	0	4	0.1	0	7.941
funsing37c	10^{-13}	1000	0	4	0.5	0	8.902
funsing37c	10^{-13}	1000	0	4	0.75	0	8.923
funsing37c	10^{-13}	1000	0	4	—	1	10.645
funsing37c	10^{-13}	1000	1	4	0.1	0	5.188
funsing37c	10^{-13}	1000	1	4	0.5	0	3.695
funsing37c	10^{-13}	1000	1	4	0.75	0	3.705
funsing37c	10^{-13}	1000	1	4	—	1	7.831

Table 2.62: Runtimes for BVP 37c when testing α with 1000 MP and collocation order 4. This BVP shows a well-known behavior. $\alpha = \frac{1}{10}$ for the non-vectorized case and $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ for the vectorized case are the best choices here.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing37c	10^{-13}	1000	0	6	0.1	0	16.273
funsing37c	10^{-13}	1000	0	6	0.5	0	15.562
funsing37c	10^{-13}	1000	0	6	0.75	0	15.563
funsing37c	10^{-13}	1000	0	6	—	1	23.474
funsing37c	10^{-13}	1000	1	6	0.1	0	12.038
funsing37c	10^{-13}	1000	1	6	0.5	0	7.841
funsing37c	10^{-13}	1000	1	6	0.75	0	7.761
funsing37c	10^{-13}	1000	1	6	—	1	19.277

Table 2.63: Runtimes for BVP 37c when testing α with 1000 MP and collocation order 6. Here $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are even better in the non-vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing37c	10^{-13}	1000	0	8	0.1	0	29.783
funsing37c	10^{-13}	1000	0	8	0.5	0	25.206
funsing37c	10^{-13}	1000	0	8	0.75	0	25.186
funsing37c	10^{-13}	1000	0	8	—	1	45.085
funsing37c	10^{-13}	1000	1	8	0.1	0	24.124
funsing37c	10^{-13}	1000	1	8	0.5	0	14.731
funsing37c	10^{-13}	1000	1	8	0.75	0	14.691
funsing37c	10^{-13}	1000	1	8	—	1	39.457

Table 2.64: Runtimes for BVP 37c when testing α with 1000 MP and collocation order 8. Again a very good result is achieved for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing54	10^{-13}	1000	0	4	0.1	0	8.201
funsing54	10^{-13}	1000	0	4	0.5	0	8.161
funsing54	10^{-13}	1000	0	4	0.75	0	8.702
funsing54	10^{-13}	1000	0	4	—	1	11.386
funsing54	10^{-13}	1000	1	4	0.1	0	4.557
funsing54	10^{-13}	1000	1	4	0.5	0	4.567
funsing54	10^{-13}	1000	1	4	0.75	0	4.696
funsing54	10^{-13}	1000	1	4	—	1	8.392

Table 2.65: Runtimes for BVP 54 when testing α with 1000 MP and collocation order 4. Comparable results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{1}{10}$ and a slight disadvantage for $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing54	10^{-13}	1000	0	6	0.1	0	15.502
funsing54	10^{-13}	1000	0	6	0.5	0	15.492
funsing54	10^{-13}	1000	0	6	0.75	0	16.383
funsing54	10^{-13}	1000	0	6	—	1	24.555
funsing54	10^{-13}	1000	1	6	0.1	0	10.175
funsing54	10^{-13}	1000	1	6	0.5	0	10.175
funsing54	10^{-13}	1000	1	6	0.75	0	10.335
funsing54	10^{-13}	1000	1	6	—	1	20.129

Table 2.66: Runtimes for BVP 54 when testing α with 1000 MP and collocation order 6. The test shows the typical behavior for this BVP, see also Tables 2.26, 2.27, 2.28 and 2.65.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing54	10^{-13}	1000	0	8	0.1	0	26.938
funsing54	10^{-13}	1000	0	8	0.5	0	26.938
funsing54	10^{-13}	1000	0	8	0.75	0	28.01
funsing54	10^{-13}	1000	0	8	—	1	46.587
funsing54	10^{-13}	1000	1	8	0.1	0	19.798
funsing54	10^{-13}	1000	1	8	0.5	0	19.758
funsing54	10^{-13}	1000	1	8	0.75	0	20.008
funsing54	10^{-13}	1000	1	8	—	1	40.659

Table 2.67: Runtimes for BVP 54 when testing α with 1000 MP and collocation order 8. Again the runtimes are related in the same way.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing56	10^{-13}	1000	0	4	0.1	0	36.833
funsing56	10^{-13}	1000	0	4	0.5	0	42.911
funsing56	10^{-13}	1000	0	4	0.75	0	43.021
funsing56	10^{-13}	1000	0	4	—	1	52.996
funsing56	10^{-13}	1000	1	4	0.1	0	32.257
funsing56	10^{-13}	1000	1	4	0.5	0	30.024
funsing56	10^{-13}	1000	1	4	0.75	0	30.074
funsing56	10^{-13}	1000	1	4	—	1	48.380

Table 2.68: Runtimes for BVP 56 when testing α with 1000 MP and collocation order 4. The vectorized case is better for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ and the non-vectorized case for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing56	10^{-13}	1000	0	6	0.1	0	115.807
funsing56	10^{-13}	1000	0	6	0.5	0	99.784
funsing56	10^{-13}	1000	0	6	0.75	0	99.894
funsing56	10^{-13}	1000	0	6	—	1	152.128
funsing56	10^{-13}	1000	1	6	0.1	0	108.226
funsing56	10^{-13}	1000	1	6	0.5	0	80.676
funsing56	10^{-13}	1000	1	6	0.75	0	80.696
funsing56	10^{-13}	1000	1	6	—	1	145.679

Table 2.69: Runtimes for BVP 56 when testing α with 1000 MP and collocation order 6. The best times are achieved for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing56	10^{-13}	1000	0	8	0.1	0	280.483
funsing56	10^{-13}	1000	0	8	0.5	0	314.542
funsing56	10^{-13}	1000	0	8	0.75	0	314.652
funsing56	10^{-13}	1000	0	8	—	1	337.986
funsing56	10^{-13}	1000	1	8	0.1	0	270.148
funsing56	10^{-13}	1000	1	8	0.5	0	287.303
funsing56	10^{-13}	1000	1	8	0.75	0	287.022
funsing56	10^{-13}	1000	1	8	—	1	328.713

Table 2.70: Runtimes for BVP 56 when testing α with 1000 MP and collocation order 8. Surprisingly the results concerning the different settings for α are different from the cases with collocation order 4 and collocation order 6. But this is similar to the results with 500 meshpoints, see Tables 2.29 - 2.31.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6001	10^{-13}	1000	0	4	0.1	0	12.788
funsing6001	10^{-13}	1000	0	4	0.5	0	20.109
funsing6001	10^{-13}	1000	0	4	0.75	0	19.998
funsing6001	10^{-13}	1000	0	4	—	1	17.486
funsing6001	10^{-13}	1000	1	4	0.1	0	8.643
funsing6001	10^{-13}	1000	1	4	0.5	0	7.120
funsing6001	10^{-13}	1000	1	4	0.75	0	7.201
funsing6001	10^{-13}	1000	1	4	—	1	12.548

Table 2.71: Runtimes for BVP 6001 when testing α with 1000 MP and collocation order 4. $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are best in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6001	10^{-13}	1000	0	6	0.1	0	27.039
funsing6001	10^{-13}	1000	0	6	0.5	0	33.999
funsing6001	10^{-13}	1000	0	6	0.75	0	34.179
funsing6001	10^{-13}	1000	0	6	—	1	38.005
funsing6001	10^{-13}	1000	1	6	0.1	0	20.600
funsing6001	10^{-13}	1000	1	6	0.5	0	14.531
funsing6001	10^{-13}	1000	1	6	0.75	0	14.681
funsing6001	10^{-13}	1000	1	6	—	1	30.724

Table 2.72: Runtimes for BVP 6001 when testing α with 1000 MP and collocation order 6. We observe good results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6001	10^{-13}	1000	0	8	0.1	0	50.843
funsing6001	10^{-13}	1000	0	8	0.5	0	53.106
funsing6001	10^{-13}	1000	0	8	0.75	0	53.247
funsing6001	10^{-13}	1000	0	8	—	1	73.365
funsing6001	10^{-13}	1000	1	8	0.1	0	41.680
funsing6001	10^{-13}	1000	1	8	0.5	0	27.349
funsing6001	10^{-13}	1000	1	8	0.75	0	27.350
funsing6001	10^{-13}	1000	1	8	—	1	63.060

Table 2.73: Runtimes for BVP 6001 when testing α with 1000 MP and collocation order 6. The higher the collocation order the better the results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$, a fact that can be observed throughout the tables.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6002	10^{-13}	1000	0	4	0.1	0	6.660
funsing6002	10^{-13}	1000	0	4	0.5	0	8.212
funsing6002	10^{-13}	1000	0	4	0.75	0	8.172
funsing6002	10^{-13}	1000	0	4	—	1	9.484
funsing6002	10^{-13}	1000	1	4	0.1	0	4.306
funsing6002	10^{-13}	1000	1	4	0.5	0	3.936
funsing6002	10^{-13}	1000	1	4	0.75	0	3.906
funsing6002	10^{-13}	1000	1	4	—	1	7.130

Table 2.74: Runtimes for BVP 6002 when testing α with 1000 MP and collocation order 4. Once more the results for the vectorized case are better for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6002	10^{-13}	1000	0	6	0.1	0	13.600
funsing6002	10^{-13}	1000	0	6	0.5	0	14.952
funsing6002	10^{-13}	1000	0	6	0.75	0	15.292
funsing6002	10^{-13}	1000	0	6	—	1	21.060
funsing6002	10^{-13}	1000	1	6	0.1	0	10.054
funsing6002	10^{-13}	1000	1	6	0.5	0	8.542
funsing6002	10^{-13}	1000	1	6	0.75	0	8.552
funsing6002	10^{-13}	1000	1	6	—	1	17.415

Table 2.75: Runtimes for BVP 6002 when testing α with 1000 MP and collocation order 6. Similar to Table 2.74 the vectorized case is better for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing6002	10^{-13}	1000	0	8	0.1	0	24.825
funsing6002	10^{-13}	1000	0	8	0.5	0	24.935
funsing6002	10^{-13}	1000	0	8	0.75	0	25.386
funsing6002	10^{-13}	1000	0	8	—	1	40.478
funsing6002	10^{-13}	1000	1	8	0.1	0	20.539
funsing6002	10^{-13}	1000	1	8	0.5	0	16.394
funsing6002	10^{-13}	1000	1	8	0.75	0	16.414
funsing6002	10^{-13}	1000	1	8	—	1	35.761

Table 2.76: Runtimes for BVP 6002 when testing α with 1000 MP and collocation order 8. The high collocation order and the vectorized realization are leading to the best results for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing7001b	10^{-13}	1000	0	4	0.1	0	7.511
funsing7001b	10^{-13}	1000	0	4	0.5	0	11.727
funsing7001b	10^{-13}	1000	0	4	0.75	0	11.807
funsing7001b	10^{-13}	1000	0	4	—	1	10.275
funsing7001b	10^{-13}	1000	1	4	0.1	0	4.897
funsing7001b	10^{-13}	1000	1	4	0.5	0	4.597
funsing7001b	10^{-13}	1000	1	4	0.75	0	4.326
funsing7001b	10^{-13}	1000	1	4	—	1	7.751

Table 2.77: Runtimes for BVP 7001b when testing α with 1000 MP and collocation order 4. The results are similar to Table 2.38 but the increased number of meshpoints seems to be a slight advantage for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing7001b	10^{-13}	1000	0	6	0.1	0	15.512
funsing7001b	10^{-13}	1000	0	6	0.5	0	19.928
funsing7001b	10^{-13}	1000	0	6	0.75	0	19.969
funsing7001b	10^{-13}	1000	0	6	—	1	23.183
funsing7001b	10^{-13}	1000	1	6	0.1	0	11.697
funsing7001b	10^{-13}	1000	1	6	0.5	0	8.782
funsing7001b	10^{-13}	1000	1	6	0.75	0	8.793
funsing7001b	10^{-13}	1000	1	6	—	1	19.338

Table 2.78: Runtimes for BVP 7001b when testing α with 1000 MP and collocation order 6. The same behavior as for collocation order 4 but with a noticeable margin to $\alpha = \frac{1}{10}$ in the vectorized case.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing7001b	10^{-13}	1000	0	8	0.1	0	28.521
funsing7001b	10^{-13}	1000	0	8	0.5	0	30.944
funsing7001b	10^{-13}	1000	0	8	0.75	0	30.964
funsing7001b	10^{-13}	1000	0	8	—	1	44.174
funsing7001b	10^{-13}	1000	1	8	0.1	0	23.374
funsing7001b	10^{-13}	1000	1	8	0.5	0	16.083
funsing7001b	10^{-13}	1000	1	8	0.75	0	16.063
funsing7001b	10^{-13}	1000	1	8	—	1	39.026

Table 2.79: Runtimes for BVP 7001b when testing α with 1000 MP and collocation order 8. Nearly comparable times for the non-vectorized case and a big margin for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ in the vectorized case. ZfSensitive = 1 is always slowest.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing71	10^{-13}	1000	0	4	0.1	0	8.912
funsing71	10^{-13}	1000	0	4	0.5	0	10.114
funsing71	10^{-13}	1000	0	4	0.75	0	23.624
funsing71	10^{-13}	1000	0	4	—	1	10.876
funsing71	10^{-13}	1000	1	4	0.1	0	5.458
funsing71	10^{-13}	1000	1	4	0.5	0	5.257
funsing71	10^{-13}	1000	1	4	0.75	0	11.536
funsing71	10^{-13}	1000	1	4	—	1	7.471

Table 2.80: Runtimes for BVP 71 when testing α with 1000 MP and collocation order 4. As seen in Table 2.41 the results for $\alpha = \frac{3}{4}$ are much slower than for $\alpha = \frac{1}{2}$. Moreover, the behavior in both cases is similar to most of the previous results.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing71	10^{-13}	1000	0	6	0.1	0	17.815
funsing71	10^{-13}	1000	0	6	0.5	0	18.917
funsing71	10^{-13}	1000	0	6	0.75	0	39.527
funsing71	10^{-13}	1000	0	6	—	1	22.833
funsing71	10^{-13}	1000	1	6	0.1	0	12.598
funsing71	10^{-13}	1000	1	6	0.5	0	11.656
funsing71	10^{-13}	1000	1	6	0.75	0	21.951
funsing71	10^{-13}	1000	1	6	—	1	17.785

Table 2.81: Runtimes for BVP 71 when testing α with 1000 MP and collocation order 6. This BVP gives very bad results for $\alpha = \frac{3}{4}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing71	10^{-13}	1000	0	8	0.1	0	32.056
funsing71	10^{-13}	1000	0	8	0.5	0	32.447
funsing71	10^{-13}	1000	0	8	0.75	0	61.909
funsing71	10^{-13}	1000	0	8	—	1	42.882
funsing71	10^{-13}	1000	1	8	0.1	0	25.126
funsing71	10^{-13}	1000	1	8	0.5	0	22.732
funsing71	10^{-13}	1000	1	8	0.75	0	38.826
funsing71	10^{-13}	1000	1	8	—	1	36.102

Table 2.82: Runtimes for BVP 71 when testing α with 1000 MP and collocation order 8. $\alpha = \frac{1}{2}$ is the best choice for this BVP.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing73	10^{-13}	1000	0	4	0.1	0	79.224
funsing73	10^{-13}	1000	0	4	0.5	0	76.190
funsing73	10^{-13}	1000	0	4	0.75	0	65.624
funsing73	10^{-13}	1000	0	4	—	1	93.455
funsing73	10^{-13}	1000	1	4	0.1	0	69.380
funsing73	10^{-13}	1000	1	4	0.5	0	65.605
funsing73	10^{-13}	1000	1	4	0.75	0	51.053
funsing73	10^{-13}	1000	1	4	—	1	83.570

Table 2.83: Runtimes for BVP 73 when testing α with 1000 MP and collocation order 4. The best runtimes are achieved by $\alpha = \frac{3}{4}$, but even $\alpha = \frac{1}{2}$ is faster than $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing73	10^{-13}	1000	0	6	0.1	0	222.841
funsing73	10^{-13}	1000	0	6	0.5	0	220.778
funsing73	10^{-13}	1000	0	6	0.75	0	174.791
funsing73	10^{-13}	1000	0	6	—	1	267.074
funsing73	10^{-13}	1000	1	6	0.1	0	208.209
funsing73	10^{-13}	1000	1	6	0.5	0	204.244
funsing73	10^{-13}	1000	1	6	0.75	0	151.778
funsing73	10^{-13}	1000	1	6	—	1	252.323

Table 2.84: Runtimes for BVP 73 when testing α with 1000 MP and collocation order 6. This BVP unexpectedly favors $\alpha = \frac{3}{4}$. Moreover, $\alpha = \frac{1}{2}$ is better than $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing73	10^{-13}	1000	0	8	0.1	0	509.012
funsing73	10^{-13}	1000	0	8	0.5	0	478.768
funsing73	10^{-13}	1000	0	8	0.75	0	363.552
funsing73	10^{-13}	1000	0	8	—	1	587.044
funsing73	10^{-13}	1000	1	8	0.1	0	487.982
funsing73	10^{-13}	1000	1	8	0.5	0	456.547
funsing73	10^{-13}	1000	1	8	0.75	0	332.798
funsing73	10^{-13}	1000	1	8	—	1	570.240

Table 2.85: Runtimes for BVP 73 when testing α with 1000 MP and collocation order 8. At the high collocation order, $\alpha = \frac{3}{4}$ is best again.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8001	10^{-13}	1000	0	4	0.1	0	10.185
funsing8001	10^{-13}	1000	0	4	0.5	0	20.740
funsing8001	10^{-13}	1000	0	4	0.75	0	20.709
funsing8001	10^{-13}	1000	0	4	—	1	13.981
funsing8001	10^{-13}	1000	1	4	0.1	0	6.450
funsing8001	10^{-13}	1000	1	4	0.5	0	8.011
funsing8001	10^{-13}	1000	1	4	0.75	0	8.041
funsing8001	10^{-13}	1000	1	4	—	1	10.195

Table 2.86: Runtimes for BVP 8001 when testing α with 1000 MP and collocation order 4. As it can be seen in Tables 2.47 - 2.49 too, this BVP is best solved for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8001	10^{-13}	1000	0	6	0.1	0	20.960
funsing8001	10^{-13}	1000	0	6	0.5	0	38.896
funsing8001	10^{-13}	1000	0	6	0.75	0	39.166
funsing8001	10^{-13}	1000	0	6	—	1	30.884
funsing8001	10^{-13}	1000	1	6	0.1	0	15.392
funsing8001	10^{-13}	1000	1	6	0.5	0	19.478
funsing8001	10^{-13}	1000	1	6	0.75	0	19.528
funsing8001	10^{-13}	1000	1	6	—	1	25.227

Table 2.87: Runtimes for BVP 8001 when testing α with 1000 MP and collocation order 6. The results are best for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8001	10^{-13}	1000	0	8	0.1	0	38.586
funsing8001	10^{-13}	1000	0	8	0.5	0	74.927
funsing8001	10^{-13}	1000	0	8	0.75	0	74.968
funsing8001	10^{-13}	1000	0	8	—	1	59.506
funsing8001	10^{-13}	1000	1	8	0.1	0	31.105
funsing8001	10^{-13}	1000	1	8	0.5	0	48.109
funsing8001	10^{-13}	1000	1	8	0.75	0	48.159
funsing8001	10^{-13}	1000	1	8	—	1	51.975

Table 2.88: Runtimes for BVP 8001 when testing α with 1000 MP and collocation order 8. BVP 8001 typically favors $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8002	10^{-13}	1000	0	4	0.1	0	9.835
funsing8002	10^{-13}	1000	0	4	0.5	0	23.003
funsing8002	10^{-13}	1000	0	4	0.75	0	23.054
funsing8002	10^{-13}	1000	0	4	—	1	12.428
funsing8002	10^{-13}	1000	1	4	0.1	0	5.628
funsing8002	10^{-13}	1000	1	4	0.5	0	9.935
funsing8002	10^{-13}	1000	1	4	0.75	0	9.954
funsing8002	10^{-13}	1000	1	4	—	1	9.123

Table 2.89: Runtimes for BVP 8002 when testing α with 1000 MP and collocation order 4. The absolutely best results are achieved for $\alpha = \frac{1}{10}$.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8002	10^{-13}	1000	0	6	0.1	0	19.528
funsing8002	10^{-13}	1000	0	6	0.5	0	47.398
funsing8002	10^{-13}	1000	0	6	0.75	0	47.468
funsing8002	10^{-13}	1000	0	6	—	1	27.49
funsing8002	10^{-13}	1000	1	6	0.1	0	13.199
funsing8002	10^{-13}	1000	1	6	0.5	0	27.089
funsing8002	10^{-13}	1000	1	6	0.75	0	27.009
funsing8002	10^{-13}	1000	1	6	—	1	22.622

Table 2.90: Runtimes for BVP 8002 when testing α with 1000 MP and collocation order 6. The relations between the runtimes are about the same as in Table 2.89.

BVP	Tol	MeshPts.	Vect.	Ord.	α	ZfS	Runtime
funsing8002	10^{-13}	1000	0	8	0.1	0	34.860
funsing8002	10^{-13}	1000	0	8	0.5	0	74.808
funsing8002	10^{-13}	1000	0	8	0.75	0	76.109
funsing8002	10^{-13}	1000	0	8	—	1	53.147
funsing8002	10^{-13}	1000	1	8	0.1	0	26.398
funsing8002	10^{-13}	1000	1	8	0.5	0	48.269
funsing8002	10^{-13}	1000	1	8	0.75	0	48.210
funsing8002	10^{-13}	1000	1	8	—	1	46.416

Table 2.91: Runtimes for BVP 8002 when testing α with 1000 MP and collocation order 8. Though the runtimes for $\alpha = \frac{1}{2}$ and $\alpha = \frac{3}{4}$ are close to ZfSensitive = 1, they are nearly twice as long as those for $\alpha = \frac{1}{10}$.

Chapter 3

A New Version: SBVP 2.0

After this first decision for the Newton solver had been made, a beta-version of SBVP 2.0 had to be optimized with respect to some parameters. All further tests in this study are based on this beta-code of SBVP 2.0.

In addition to some basic improvements, the main issue in the development of SBVP 2.0 was the modification of the zerofinder. In SBVP 1.0, the zerofinder consists of a Newton zerofinder and the standard MATLAB zerofinder `fsolve`. The new procedure for the zerofinder consists of 3 different algorithms:

- 'Fast Frozen Newton' is a cheap routine with a small domain of convergence.
- 'Predictor-Corrector Linesearch' is more expensive but has a large domain of convergence.
- The 'Trust Region Method' is the most expensive routine but it has the largest domain of convergence.

While the first two levels, the Fast Frozen Newton and the Predictor-Corrector Linesearch, are enabled by default, the third one, the Trust Region Method, can be activated only by a private parameter in the code. This restriction was necessary since some tests showed that the performance of the Trust Region Method usually was not favorable.

Other important issues in the development of SBVP 2.0 were the implementation of a revised mesh selection strategy, the efficient calculation of the modified defect and last but not least the implementation of a logging structure that contains information on the solution process.

This logging structure was used to retrieve all necessary data from the tests this work is based on. It is an easy, user friendly way to analyze the history of the solution process of the BVP and it was a great help throughout this study. In practice, however, the logging structure should be disabled to enhance the performance of the code.

3.1 Mesh-Distribution and Risk-Premium

The next parameters that are optimized are Mesh-Distribution and Risk-Premium. These private parameters are set in the `sbvp20.m`-file of the package.

The first possible setting for Mesh-Distribution is All which aims at the equidistribution of the global error of the numerical solution. The following theoretical considerations can be found in [23].

Consider a mesh

$$\Delta := (a := \tau_0, \tau_1, \ldots, \tau_{N-1}, b := \tau_N),$$

and $J_i := [\tau_i, \tau_{i+1}],\ h_i := \tau_{i+1} - \tau_i,\ i = 0, \ldots, N-1,\ h_{\max} = \max_i h_i,\ h_{\min} = \min_i h_i$. On this mesh, we compute the numerical solution using polynomial collocation $p(t),\ t \in [a, b]$, for a BVP in ODEs posed on the interval $[a, b]$, see Section 1.3.

Moreover, we assume that an asymptotically correct estimate for the global error of p is available. Denote by $e(t)$ the absolute value of the global error of the numerical solution, $e(t) = |y(t) - p(t)|$, and by $\varepsilon(t)$ the absolute value of the error estimate.

Since we have assumed the estimate to be asymptotically correct, we may write

$$\begin{aligned} e(t) &= h_i^m \frac{\varepsilon(t)}{h_i^m} + O(h^{m+1}) \\ &= h_i^m \left(\frac{\Theta(t)}{h_i} \right)^m + O(h^{m+1}), \quad t \in J_i, \end{aligned}$$

where

$$\Theta(t) := \sqrt[m]{\varepsilon(t)}, \quad t \in [a, b],$$

and m is the order of the numerical method. Note that actually we use a smoothed version of the error estimate. Now, define

$$\begin{aligned} &\rho(t) := g'(t) := \frac{1}{N h_i}, \quad t \in J_i,\ i = 0, \ldots, N-1, \\ &\tilde{\Theta}(t) := \frac{\Theta(t)}{\sqrt[m]{\min_{\tau \in [a,b]} (T_{abs} + T_{rel} |p(\tau)|)}}, \\ &\widehat{\rho\Theta}(t) := \max \left\{ \rho(t)\tilde{\Theta}(t), \frac{\max_{\tau \in [a,b]} \rho(\tau)\tilde{\Theta}(\tau)}{K} \right\}, \\ &I := \int_a^b \widehat{\rho\Theta}(t)\, dt. \end{aligned}$$

The first of these four equations implies that

$$\int_a^b \rho(t)\, dt = 1.$$

In the second equation, T_{abs} and T_{rel} denote absolute and relative tolerance parameters used for the mixed tolerances.

The mesh redistribution is organized as follows:

$$\rho(t) \mapsto \frac{1}{I}\widehat{\rho\Theta}(t) =: \tilde{\rho}(t),$$
$$N \mapsto \max\{1.5N, (1+\delta)IN\} =: \tilde{N},$$

where δ is a safety margin which we set to $\delta := 0.1$. Note that the first equation implies

$$\int_a^b \tilde{\rho}(t)\,dt = 1.$$

Now, the new grid is computed according to

$$\tilde{g}'(t) := \tilde{\rho}(t),$$
$$\tilde{\tau}_j := \tilde{g}^{-1}\left(\frac{j}{\tilde{N}}\right), \quad j = 0, \ldots, \tilde{N}.$$

Alternatively, if Mesh-Distribution is set to TolOpt, we have

$$\tilde{\Theta}(t) := \frac{\Theta(t)}{\sqrt[m]{T_{abs} + T_{rel}|p(t)|}}.$$

Here, instead of the equidistribution of the global error, TolOpt aims at a distribution of the error which attempts to satisfy mixed tolerance requirements most efficiently.

The third option eTol is a modification of the formula for TolOpt:

$$\tilde{\Theta}(t) := \frac{\Theta(t)}{\sqrt[m]{T_{abs} + T_{rel}|p(t_j)|}}.$$

Here, t_j comes from

$$\max_i \frac{\Theta(t_i)}{\sqrt[m]{T_{abs} + T_{rel}|p(t_i)|}} = \frac{\Theta(t_j)}{\sqrt[m]{T_{abs} + T_{rel}|p(t_j)|}}.$$

In general there is a greater risk to fail satisfying the tolerances when redistributing a mesh than by using coherent refinement. But as the efficiency concerning the number of meshpoints of coherent refinement is not necessarily better than at a redistributed mesh, an option is needed to decide when to 'risk' redistribution instead of coherent refinement. This option is Risk-Premium.

Risk-Premium is the factor by which the number of mesh points predicted to satisfy the tolerances must be smaller for redistribution than for coherent mesh refinement. For the tests, Risk-Premium was set to

- $\frac{1}{10}$, i. e. redistribution is chosen if it needs 10 % fewer meshpoints,
- $\frac{1}{4}$, i. e. redistribution is chosen if it needs 25 % fewer meshpoints,
- $\frac{1}{3}$, i. e. redistribution is chosen if it needs 33 % fewer meshpoints,
- $\frac{1}{2}$, i. e. redistribution is chosen if it needs 50 % fewer meshpoints.

While $\frac{1}{4}$ and $\frac{1}{3}$ are most interesting because they seem to be moderate and reasonable, $\frac{1}{10}$ and $\frac{1}{2}$ are mainly chosen to confirm that extreme values of this parameters are usually not favorable for the performance.

Because Risk-Premium and Mesh-Distribution control closely related parts of the program, separate tests for each of the parameters would be misleading. So they are tested in every combination. That means 12 different options for each test configuration, leading to a total of 900 datasets because 25 BVPS were tested and three tolerances used.

3.1.1 Test Results

For the tests the relative and the absolute tolerance are chosen to be equal. If not stated otherwise, default options are used. The chosen tolerances should show the performance for rather fast results (10^{-3}), rather accurate solutions (10^{-8}) and a value in between those two to see some kind of development (10^{-6}).

The main characteristic for the tests is the runtime. Some results are discussed in a more detailed manner including figures of the grid evolution during the solution process. These figures show the different grids in each step of refinement, the number of meshpoints in the grid, a loose picture of the distribution and the tolerance factor, which tells about the ratio of the calculated solution and the estimated error:

$$tolfactor = max\left(\frac{\|\epsilon\|}{T_{abs} + \|y_c\| * T_{rel}}\right)$$

where ϵ is the estimated error of the solution approximation y_c. When the tolerance factor is smaller than 1, the routine assumes that the tolerances are satisfied. In Tables 3.1 - 3.75 the results of the tests are given.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing0026	10^{-8}	All	$\frac{1}{10}$	10	0.110
funsing0026	10^{-8}	All	$\frac{1}{4}$	10	0.091
funsing0026	10^{-8}	All	$\frac{1}{3}$	10	0.100
funsing0026	10^{-8}	All	$\frac{1}{2}$	10	0.110
funsing0026	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.120
funsing0026	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.091
funsing0026	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.110
funsing0026	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.120
funsing0026	10^{-8}	eTol	$\frac{1}{10}$	10	0.101
funsing0026	10^{-8}	eTol	$\frac{1}{4}$	10	0.100
funsing0026	10^{-8}	eTol	$\frac{1}{3}$	10	0.110
funsing0026	10^{-8}	eTol	$\frac{1}{2}$	10	0.110

Table 3.1: Results for the combined test for Risk-Premium and Mesh-Distribution. Best results are achieved for $\frac{1}{4}$.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing0026	10^{-6}	All	$\frac{1}{10}$	10	0.090
funsing0026	10^{-6}	All	$\frac{1}{4}$	10	0.080
funsing0026	10^{-6}	All	$\frac{1}{3}$	10	0.101
funsing0026	10^{-6}	All	$\frac{1}{2}$	10	0.081
funsing0026	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.101
funsing0026	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.080
funsing0026	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.090
funsing0026	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.080
funsing0026	10^{-6}	eTol	$\frac{1}{10}$	10	0.080
funsing0026	10^{-6}	eTol	$\frac{1}{4}$	10	0.100
funsing0026	10^{-6}	eTol	$\frac{1}{3}$	10	0.101
funsing0026	10^{-6}	eTol	$\frac{1}{2}$	10	0.101

Table 3.2: Results for the combined test for Risk-Premium and Mesh-Distribution. Very similar runtimes are achieved for each configuration.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing0026	10^{-3}	All	$\frac{1}{10}$	5	0.061
funsing0026	10^{-3}	All	$\frac{1}{4}$	5	0.061
funsing0026	10^{-3}	All	$\frac{1}{3}$	5	0.080
funsing0026	10^{-3}	All	$\frac{1}{2}$	5	0.060
funsing0026	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.050
funsing0026	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.060
funsing0026	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.060
funsing0026	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.081
funsing0026	10^{-3}	eTol	$\frac{1}{10}$	5	0.060
funsing0026	10^{-3}	eTol	$\frac{1}{4}$	5	0.061
funsing0026	10^{-3}	eTol	$\frac{1}{3}$	5	0.070
funsing0026	10^{-3}	eTol	$\frac{1}{2}$	5	0.060

Table 3.3: Results for the combined test for Risk-Premium and Mesh-Distribution. These very quick runtimes do not allow any statement concerning quality.

Bvp 0026 is one of the demofiles that have been released with the first version of Sbvp. Therefore no problems were to be expected and the results show exactly this picture. All configurations are solving these problems very quickly, not allowing any statement of the performance of the solver. Only for the strict tolerance Risk-Premium at $\frac{1}{4}$ shows a slight advantage, especially for TolOpt and All.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing015	10^{-8}	All	$\frac{1}{10}$	26	0.130
funsing015	10^{-8}	All	$\frac{1}{4}$	26	0.130
funsing015	10^{-8}	All	$\frac{1}{3}$	26	0.150
funsing015	10^{-8}	All	$\frac{1}{2}$	26	0.151
funsing015	10^{-8}	TolOpt	$\frac{1}{10}$	26	0.150
funsing015	10^{-8}	TolOpt	$\frac{1}{4}$	26	0.130
funsing015	10^{-8}	TolOpt	$\frac{1}{3}$	26	0.131
funsing015	10^{-8}	TolOpt	$\frac{1}{2}$	26	0.160
funsing015	10^{-8}	eTol	$\frac{1}{10}$	26	0.150
funsing015	10^{-8}	eTol	$\frac{1}{4}$	26	0.150
funsing015	10^{-8}	eTol	$\frac{1}{3}$	26	0.140
funsing015	10^{-8}	eTol	$\frac{1}{2}$	26	0.160

Table 3.4: Results for the combined test for Risk-Premium and Mesh-Distribution.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing015	10^{-6}	All	$\frac{1}{10}$	33	0.120
funsing015	10^{-6}	All	$\frac{1}{4}$	33	0.131
funsing015	10^{-6}	All	$\frac{1}{3}$	33	0.140
funsing015	10^{-6}	All	$\frac{1}{2}$	33	0.131
funsing015	10^{-6}	TolOpt	$\frac{1}{10}$	26	0.110
funsing015	10^{-6}	TolOpt	$\frac{1}{4}$	26	0.110
funsing015	10^{-6}	TolOpt	$\frac{1}{3}$	33	0.120
funsing015	10^{-6}	TolOpt	$\frac{1}{2}$	33	0.120
funsing015	10^{-6}	eTol	$\frac{1}{10}$	33	0.140
funsing015	10^{-6}	eTol	$\frac{1}{4}$	33	0.140
funsing015	10^{-6}	eTol	$\frac{1}{3}$	33	0.141
funsing015	10^{-6}	eTol	$\frac{1}{2}$	33	0.140

Table 3.5: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt shows the best behavior and 26 meshpoints are enough for the lower Risk-Premiums.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing015	10^{-3}	All	$\frac{1}{10}$	20	0.070
funsing015	10^{-3}	All	$\frac{1}{4}$	20	0.060
funsing015	10^{-3}	All	$\frac{1}{3}$	20	0.060
funsing015	10^{-3}	All	$\frac{1}{2}$	20	0.070
funsing015	10^{-3}	TolOpt	$\frac{1}{10}$	47	0.240
funsing015	10^{-3}	TolOpt	$\frac{1}{4}$	20	0.050
funsing015	10^{-3}	TolOpt	$\frac{1}{3}$	20	0.070
funsing015	10^{-3}	TolOpt	$\frac{1}{2}$	20	0.070
funsing015	10^{-3}	eTol	$\frac{1}{10}$	20	0.060
funsing015	10^{-3}	eTol	$\frac{1}{4}$	20	0.070
funsing015	10^{-3}	eTol	$\frac{1}{3}$	20	0.080
funsing015	10^{-3}	eTol	$\frac{1}{2}$	20	0.070

Table 3.6: Results for the combined test for Risk-Premium and Mesh-Distribution. A little instability shows for TolOpt at $\frac{1}{10}$. The rest is very fast.

For Bvp 015 TolOpt performs best for tolerance 10^{-6} and eTol is slowest. However, the difference is not too high. In Figure 3.1 the mesh adaptation process for this Bvp at tolerance 10^{-6} and Risk-Premium $\frac{1}{10}$ is shown for all different settings of Mesh-Distribution. The grid is divided into ten equally spaced sections. The numbers under the sections indicate the number of meshpoints in each section. Note that for the first grid the meshpoints coincide with the endpoints of the sections. The main purpose of the numbers is to help to recognize and distinguish between higher densities. Especially on dense grids this is quite difficult to do, by simply looking at the grid.

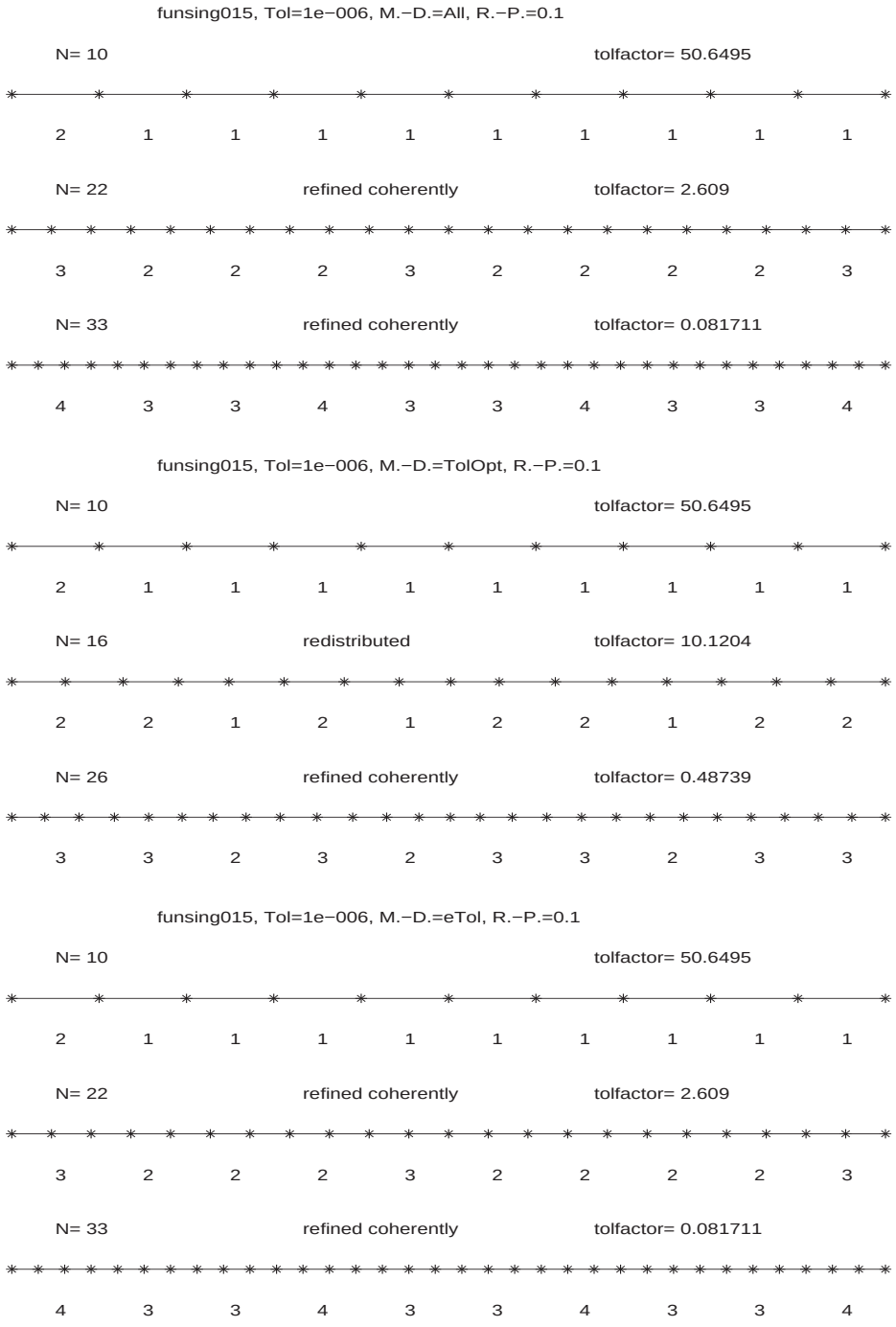


Figure 3.1: This figure shows the grid evolution of BVP 015 for the tolerance set to 10^{-6} and Risk-Premium at $\frac{1}{10}$. First, the grid evolution for Mesh-Distribution All, then for TolOpt, and finally for eTol is given.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1001	10^{-8}	All	$\frac{1}{10}$	58	0.211
funsing1001	10^{-8}	All	$\frac{1}{4}$	58	0.200
funsing1001	10^{-8}	All	$\frac{1}{3}$	58	0.211
funsing1001	10^{-8}	All	$\frac{1}{2}$	58	0.230
funsing1001	10^{-8}	TolOpt	$\frac{1}{10}$	78	0.391
funsing1001	10^{-8}	TolOpt	$\frac{1}{4}$	58	0.190
funsing1001	10^{-8}	TolOpt	$\frac{1}{3}$	58	0.200
funsing1001	10^{-8}	TolOpt	$\frac{1}{2}$	58	0.211
funsing1001	10^{-8}	eTol	$\frac{1}{10}$	58	0.200
funsing1001	10^{-8}	eTol	$\frac{1}{4}$	58	0.220
funsing1001	10^{-8}	eTol	$\frac{1}{3}$	58	0.211
funsing1001	10^{-8}	eTol	$\frac{1}{2}$	58	0.200

Table 3.7: Results for the combined test for Risk-Premium and Mesh-Distribution. Again TolOpt with $\frac{1}{10}$ seems to be a bit unstable.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1001	10^{-6}	All	$\frac{1}{10}$	60	0.220
funsing1001	10^{-6}	All	$\frac{1}{4}$	60	0.201
funsing1001	10^{-6}	All	$\frac{1}{3}$	60	0.230
funsing1001	10^{-6}	All	$\frac{1}{2}$	60	0.220
funsing1001	10^{-6}	TolOpt	$\frac{1}{10}$	60	0.210
funsing1001	10^{-6}	TolOpt	$\frac{1}{4}$	60	0.201
funsing1001	10^{-6}	TolOpt	$\frac{1}{3}$	60	0.211
funsing1001	10^{-6}	TolOpt	$\frac{1}{2}$	60	0.210
funsing1001	10^{-6}	eTol	$\frac{1}{10}$	60	0.211
funsing1001	10^{-6}	eTol	$\frac{1}{4}$	60	0.221
funsing1001	10^{-6}	eTol	$\frac{1}{3}$	60	0.210
funsing1001	10^{-6}	eTol	$\frac{1}{2}$	60	0.220

Table 3.8: Results for the combined test for Risk-Premium and Mesh-Distribution. Very similar runtimes show for Bvp 1001 at the tolerance set to 10^{-6}.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1001	10^{-3}	All	$\frac{1}{10}$	40	0.110
funsing1001	10^{-3}	All	$\frac{1}{4}$	40	0.100
funsing1001	10^{-3}	All	$\frac{1}{3}$	40	0.110
funsing1001	10^{-3}	All	$\frac{1}{2}$	40	0.100
funsing1001	10^{-3}	TolOpt	$\frac{1}{10}$	40	0.110
funsing1001	10^{-3}	TolOpt	$\frac{1}{4}$	40	0.100
funsing1001	10^{-3}	TolOpt	$\frac{1}{3}$	40	0.110
funsing1001	10^{-3}	TolOpt	$\frac{1}{2}$	40	0.110
funsing1001	10^{-3}	eTol	$\frac{1}{10}$	40	0.100
funsing1001	10^{-3}	eTol	$\frac{1}{4}$	40	0.110
funsing1001	10^{-3}	eTol	$\frac{1}{3}$	40	0.120
funsing1001	10^{-3}	eTol	$\frac{1}{2}$	40	0.120

Table 3.9: Results for the combined test for Risk-Premium and Mesh-Distribution.

BVP 1001 is a very good example to demonstrate the ambivalence of Mesh-Distribution TolOpt with Risk-Premium $\frac{1}{10}$. At the strict tolerance the number of meshpoints for this configuration increases by about 20 and the runtime is twice the runtime of all the other configurations. On the other hand TolOpt shows no problem for any other settings of Risk-Premium or tolerance.
As seen in Figure 3.2 TolOpt nearly finishes with 52 meshpoints (tolfactor=1.0689) but one more refinement is needed, whereas All is satisfied with 58 meshpoints. The higher number of meshpoints was sufficient to satisfy the tolerances and the lower number in the configuration with TolOpt was penalized with one further, expensive mesh refinement.

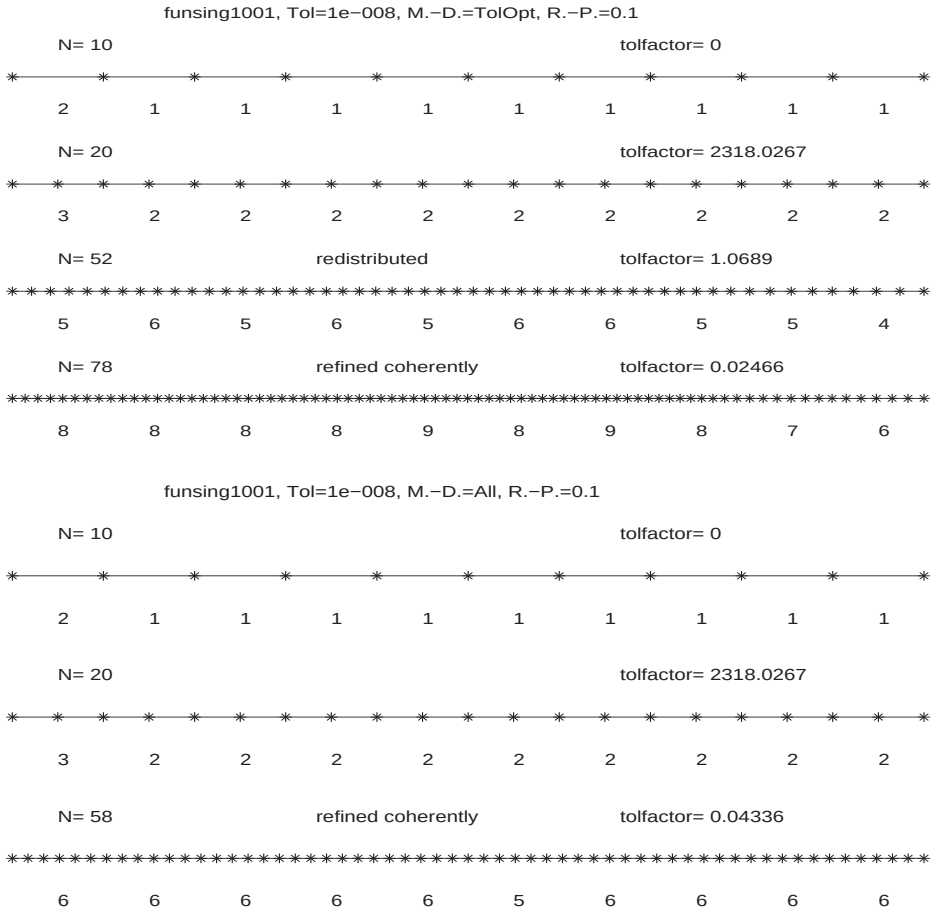


Figure 3.2: This figure shows the grid evolution of Bvp 1001 for the tolerance 10^{-8} and Risk-Premium $\frac{1}{10}$ for TolOpt and All.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1002	10^{-8}	All	$\frac{1}{10}$	22	0.110
funsing1002	10^{-8}	All	$\frac{1}{4}$	22	0.091
funsing1002	10^{-8}	All	$\frac{1}{3}$	22	0.090
funsing1002	10^{-8}	All	$\frac{1}{2}$	22	0.090
funsing1002	10^{-8}	TolOpt	$\frac{1}{10}$	32	0.150
funsing1002	10^{-8}	TolOpt	$\frac{1}{4}$	32	0.150
funsing1002	10^{-8}	TolOpt	$\frac{1}{3}$	32	0.160
funsing1002	10^{-8}	TolOpt	$\frac{1}{2}$	32	0.161
funsing1002	10^{-8}	eTol	$\frac{1}{10}$	22	0.090
funsing1002	10^{-8}	eTol	$\frac{1}{4}$	22	0.110
funsing1002	10^{-8}	eTol	$\frac{1}{3}$	22	0.100
funsing1002	10^{-8}	eTol	$\frac{1}{2}$	22	0.120

Table 3.10: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt is very slow compared to All and eTol, which results from the 10 additional meshpoints required.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1002	10^{-6}	All	$\frac{1}{10}$	23	0.090
funsing1002	10^{-6}	All	$\frac{1}{4}$	23	0.080
funsing1002	10^{-6}	All	$\frac{1}{3}$	23	0.090
funsing1002	10^{-6}	All	$\frac{1}{2}$	23	0.100
funsing1002	10^{-6}	TolOpt	$\frac{1}{10}$	22	0.090
funsing1002	10^{-6}	TolOpt	$\frac{1}{4}$	22	0.070
funsing1002	10^{-6}	TolOpt	$\frac{1}{3}$	22	0.091
funsing1002	10^{-6}	TolOpt	$\frac{1}{2}$	22	0.090
funsing1002	10^{-6}	eTol	$\frac{1}{10}$	23	0.080
funsing1002	10^{-6}	eTol	$\frac{1}{4}$	23	0.100
funsing1002	10^{-6}	eTol	$\frac{1}{3}$	23	0.090
funsing1002	10^{-6}	eTol	$\frac{1}{2}$	23	0.110

Table 3.11: Results for the combined test for Risk-Premium and Mesh-Distribution. There are no problems for any configuration but a slight advantage of one meshpoint for TolOpt.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1002	10^{-3}	All	$\frac{1}{10}$	23	0.091
funsing1002	10^{-3}	All	$\frac{1}{4}$	23	0.101
funsing1002	10^{-3}	All	$\frac{1}{3}$	23	0.110
funsing1002	10^{-3}	All	$\frac{1}{2}$	42	0.111
funsing1002	10^{-3}	TolOpt	$\frac{1}{10}$	23	0.100
funsing1002	10^{-3}	TolOpt	$\frac{1}{4}$	23	0.080
funsing1002	10^{-3}	TolOpt	$\frac{1}{3}$	31	0.110
funsing1002	10^{-3}	TolOpt	$\frac{1}{2}$	31	0.100
funsing1002	10^{-3}	eTol	$\frac{1}{10}$	23	0.090
funsing1002	10^{-3}	eTol	$\frac{1}{4}$	23	0.110
funsing1002	10^{-3}	eTol	$\frac{1}{3}$	23	0.110
funsing1002	10^{-3}	eTol	$\frac{1}{2}$	42	0.121

Table 3.12: Results for the combined test for Risk-Premium and Mesh-Distribution. $\frac{1}{2}$ needs most of the meshpoints in every setting for Mesh-Distribution. Hence the lower Risk-Premiums are the choice here.

BVP 1002 is a very interesting model. For the strict tolerance, All performs best and especially TolOpt is lagging behind. For Mesh-Distribution at All and eTol, the first mesh is redistributed to 22 meshpoints and the tolerances are satisfied. With TolOpt only 21 meshpoints are used and the tolerances are missed by a narrow margin, so another step of refinement is needed to finally satisfy the tolerances, cf. Figure 3.3.

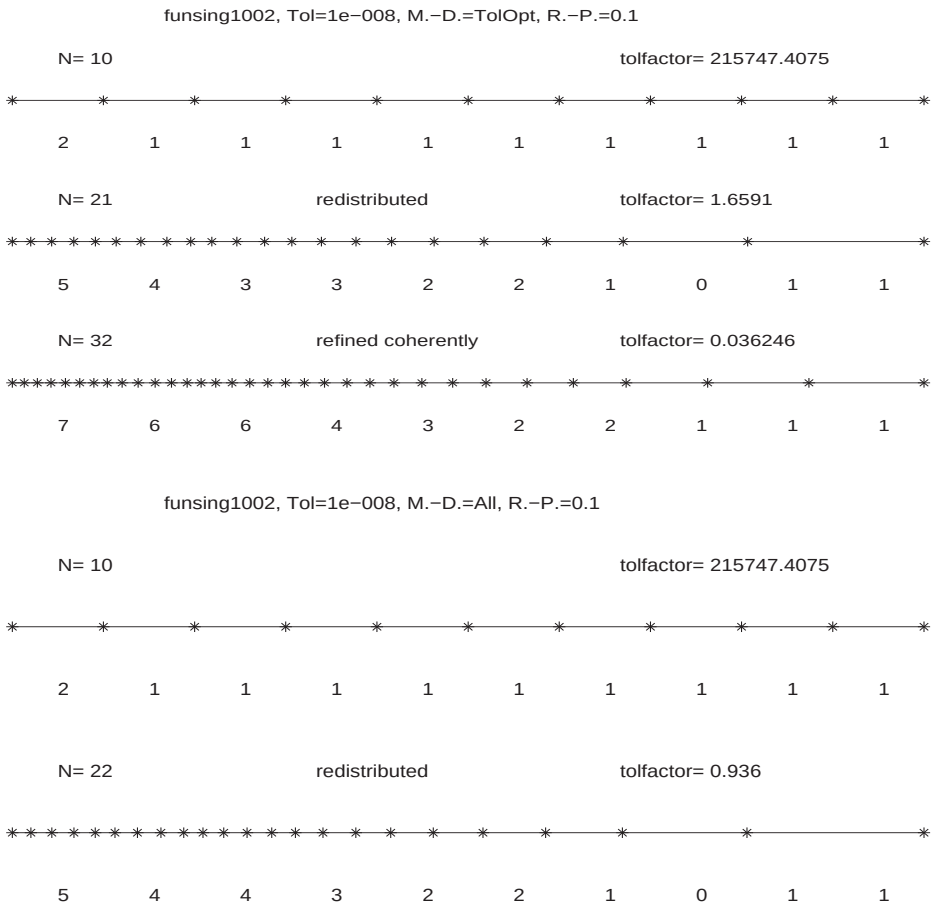


Figure 3.3: This figure shows the grid evolution of Bvp 1002 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$ for TolOpt and All.

For the tolerance 10^{-6} TolOpt shows a more favorable behavior. In fact it needs one point less to satisfy the tolerances, but as this one point does not mean that the number of refinements was different the other two options do well too. Finally for the mild tolerance 10^{-3} the higher Risk-Premiums yield worse results. This means that redistribution is the better way here.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1004	10^{-8}	All	$\frac{1}{10}$	15	0.090
funsing1004	10^{-8}	All	$\frac{1}{4}$	15	0.090
funsing1004	10^{-8}	All	$\frac{1}{3}$	15	0.090
funsing1004	10^{-8}	All	$\frac{1}{2}$	15	0.090
funsing1004	10^{-8}	TolOpt	$\frac{1}{10}$	15	0.090
funsing1004	10^{-8}	TolOpt	$\frac{1}{4}$	15	0.080
funsing1004	10^{-8}	TolOpt	$\frac{1}{3}$	15	0.091
funsing1004	10^{-8}	TolOpt	$\frac{1}{2}$	15	0.091
funsing1004	10^{-8}	eTol	$\frac{1}{10}$	15	0.080
funsing1004	10^{-8}	eTol	$\frac{1}{4}$	15	0.100
funsing1004	10^{-8}	eTol	$\frac{1}{3}$	15	0.100
funsing1004	10^{-8}	eTol	$\frac{1}{2}$	15	0.100

Table 3.13: Results for the combined test for Risk-Premium and Mesh-Distribution. Identical results are achieved for every configuration.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1004	10^{-6}	All	$\frac{1}{10}$	15	0.080
funsing1004	10^{-6}	All	$\frac{1}{4}$	15	0.070
funsing1004	10^{-6}	All	$\frac{1}{3}$	15	0.090
funsing1004	10^{-6}	All	$\frac{1}{2}$	15	0.080
funsing1004	10^{-6}	TolOpt	$\frac{1}{10}$	15	0.080
funsing1004	10^{-6}	TolOpt	$\frac{1}{4}$	15	0.071
funsing1004	10^{-6}	TolOpt	$\frac{1}{3}$	15	0.080
funsing1004	10^{-6}	TolOpt	$\frac{1}{2}$	15	0.090
funsing1004	10^{-6}	eTol	$\frac{1}{10}$	15	0.070
funsing1004	10^{-6}	eTol	$\frac{1}{4}$	15	0.080
funsing1004	10^{-6}	eTol	$\frac{1}{3}$	15	0.090
funsing1004	10^{-6}	eTol	$\frac{1}{2}$	15	0.081

Table 3.14: Results for the combined test for Risk-Premium and Mesh-Distribution. Identical results are achieved for every configuration.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1004	10^{-3}	All	$\frac{1}{10}$	9	0.060
funsing1004	10^{-3}	All	$\frac{1}{4}$	11	0.050
funsing1004	10^{-3}	All	$\frac{1}{3}$	11	0.060
funsing1004	10^{-3}	All	$\frac{1}{2}$	11	0.071
funsing1004	10^{-3}	TolOpt	$\frac{1}{10}$	8	0.040
funsing1004	10^{-3}	TolOpt	$\frac{1}{4}$	8	0.040
funsing1004	10^{-3}	TolOpt	$\frac{1}{3}$	11	0.050
funsing1004	10^{-3}	TolOpt	$\frac{1}{2}$	11	0.060
funsing1004	10^{-3}	eTol	$\frac{1}{10}$	8	0.050
funsing1004	10^{-3}	eTol	$\frac{1}{4}$	8	0.070
funsing1004	10^{-3}	eTol	$\frac{1}{3}$	11	0.070
funsing1004	10^{-3}	eTol	$\frac{1}{2}$	11	0.070

Table 3.15: Results for the combined test for Risk-Premium and Mesh-Distribution. The lower Risk-Premiums yield a lower number of meshpoints, though the overall runtimes are too short to be considered representative.

The observations above are approved: For tolerance 10^{-3} it is better to redistribute instead of refining coherently, even if the number of meshpoints for redistribution is only a bit smaller. Due to the very low runtimes no further conclusion can be drawn from Bvp 1004.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1005	10^{-8}	All	$\frac{1}{10}$	140	0.650
funsing1005	10^{-8}	All	$\frac{1}{4}$	140	0.671
funsing1005	10^{-8}	All	$\frac{1}{3}$	140	0.681
funsing1005	10^{-8}	All	$\frac{1}{2}$	160	0.561
funsing1005	10^{-8}	TolOpt	$\frac{1}{10}$	137	0.982
funsing1005	10^{-8}	TolOpt	$\frac{1}{4}$	137	0.630
funsing1005	10^{-8}	TolOpt	$\frac{1}{3}$	137	0.631
funsing1005	10^{-8}	TolOpt	$\frac{1}{2}$	175	0.580
funsing1005	10^{-8}	eTol	$\frac{1}{10}$	140	0.671
funsing1005	10^{-8}	eTol	$\frac{1}{4}$	140	0.691
funsing1005	10^{-8}	eTol	$\frac{1}{3}$	140	0.701
funsing1005	10^{-8}	eTol	$\frac{1}{2}$	159	0.551

Table 3.16: Results for the combined test for Risk-Premium and Mesh-Distribution. Strikingly good runtimes are achieved for $\frac{1}{2}$ though here the most meshpoints are required. TolOpt with $\frac{1}{10}$ is rather slow.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1005	10^{-6}	All	$\frac{1}{10}$	120	0.400
funsing1005	10^{-6}	All	$\frac{1}{4}$	120	0.431
funsing1005	10^{-6}	All	$\frac{1}{3}$	120	0.441
funsing1005	10^{-6}	All	$\frac{1}{2}$	120	0.420
funsing1005	10^{-6}	TolOpt	$\frac{1}{10}$	120	0.411
funsing1005	10^{-6}	TolOpt	$\frac{1}{4}$	120	0.401
funsing1005	10^{-6}	TolOpt	$\frac{1}{3}$	120	0.410
funsing1005	10^{-6}	TolOpt	$\frac{1}{2}$	120	0.411
funsing1005	10^{-6}	eTol	$\frac{1}{10}$	120	0.410
funsing1005	10^{-6}	eTol	$\frac{1}{4}$	120	0.431
funsing1005	10^{-6}	eTol	$\frac{1}{3}$	120	0.441
funsing1005	10^{-6}	eTol	$\frac{1}{2}$	120	0.440

Table 3.17: Results for the combined test for Risk-Premium and Mesh-Distribution. There is hardly a difference in the results above.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing1005	10^{-3}	All	$\frac{1}{10}$	240	0.561
funsing1005	10^{-3}	All	$\frac{1}{4}$	240	0.581
funsing1005	10^{-3}	All	$\frac{1}{3}$	240	0.571
funsing1005	10^{-3}	All	$\frac{1}{2}$	473	0.902
funsing1005	10^{-3}	TolOpt	$\frac{1}{10}$	240	0.631
funsing1005	10^{-3}	TolOpt	$\frac{1}{4}$	240	0.550
funsing1005	10^{-3}	TolOpt	$\frac{1}{3}$	240	0.550
funsing1005	10^{-3}	TolOpt	$\frac{1}{2}$	473	0.931
funsing1005	10^{-3}	eTol	$\frac{1}{10}$	240	0.581
funsing1005	10^{-3}	eTol	$\frac{1}{4}$	240	0.591
funsing1005	10^{-3}	eTol	$\frac{1}{3}$	240	0.621
funsing1005	10^{-3}	eTol	$\frac{1}{2}$	473	0.921

Table 3.18: Results for the combined test for Risk-Premium and Mesh-Distribution. The number of meshpoints and the runtime for $\frac{1}{2}$ is bad, whereas $\frac{1}{10}$ performs best, with the greatest advantage for Mesh-Distribution set to All.

The very first thing to see is that Risk-Premium at $\frac{1}{2}$ falls far behind for tolerance 10^{-3}. But even for the strict one the number of meshpoints is noticeably higher though the runtimes decrease. TolOpt has problems with $\frac{1}{10}$ whereas this is the best setting for the Risk-Premium for All and eTol when $\frac{1}{2}$ is left aside.

As seen in Figures 3.4 - 3.6 the configurations with Risk-Premium at $\frac{1}{10}$ are redistributing the mesh a second time but they fail the tolerances and coherent refinement is needed as the last step. The coherent refinement when Risk-Premium is set to $\frac{1}{2}$ is more effective than the redistribution and so the effort required for the calculations on the grid with $N = 160$ is

smaller than the effort for an additional step of refinement.

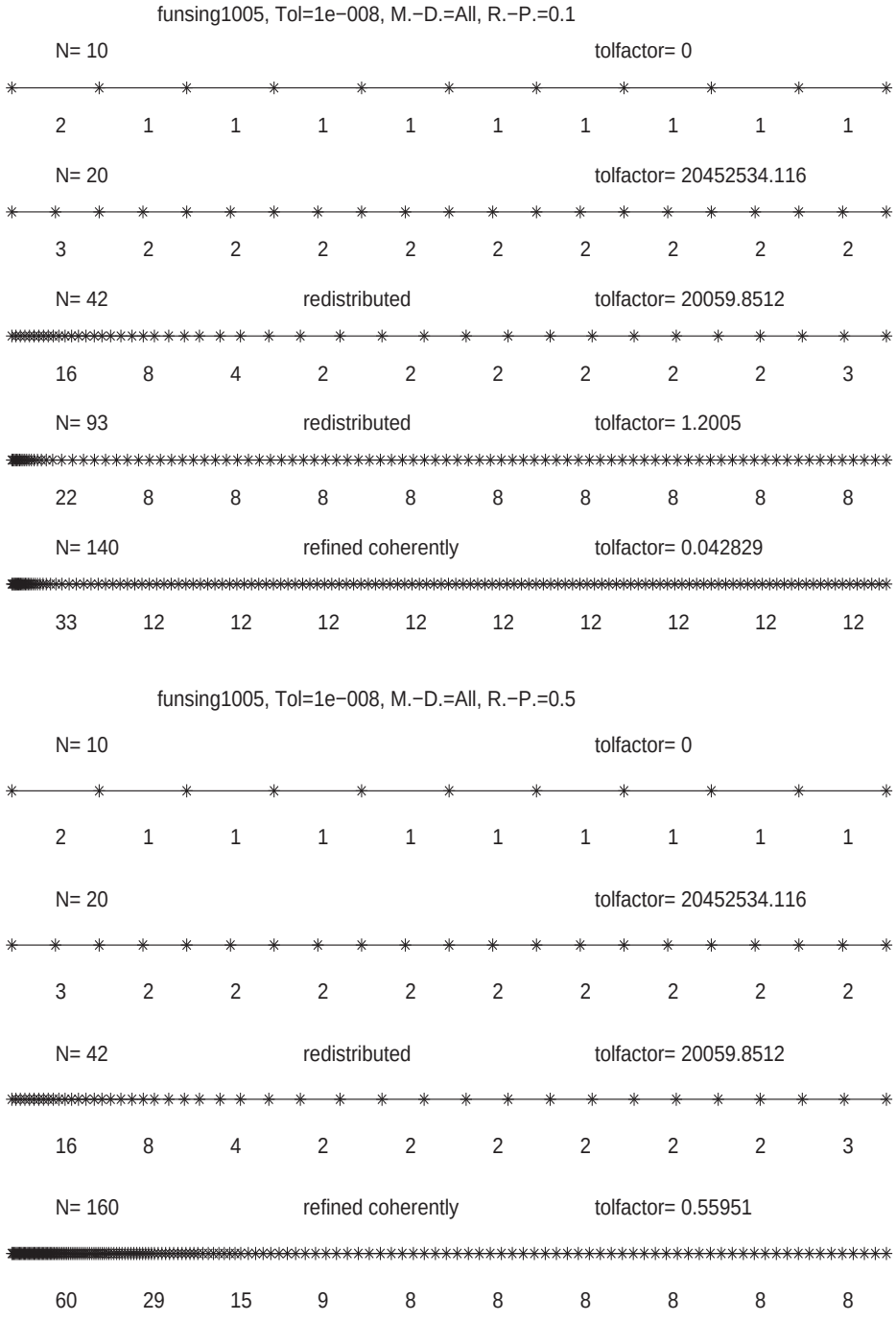


Figure 3.4: This figure shows the grid evolution of Bvp 1005 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$ and $\frac{1}{2}$ for All.

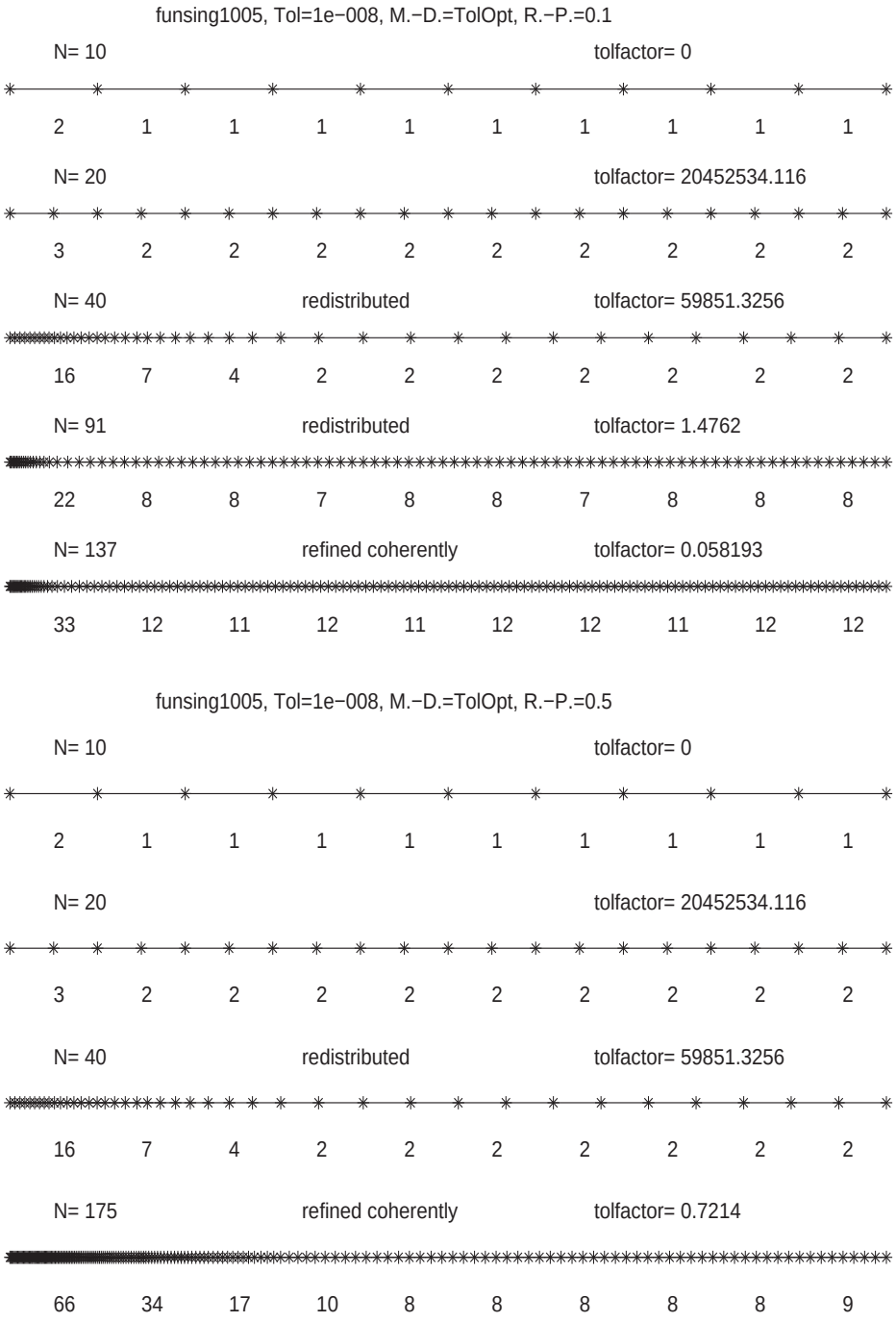


Figure 3.5: This figure shows the grid evolution of BVP 1005 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$ and $\frac{1}{2}$ for TolOpt.

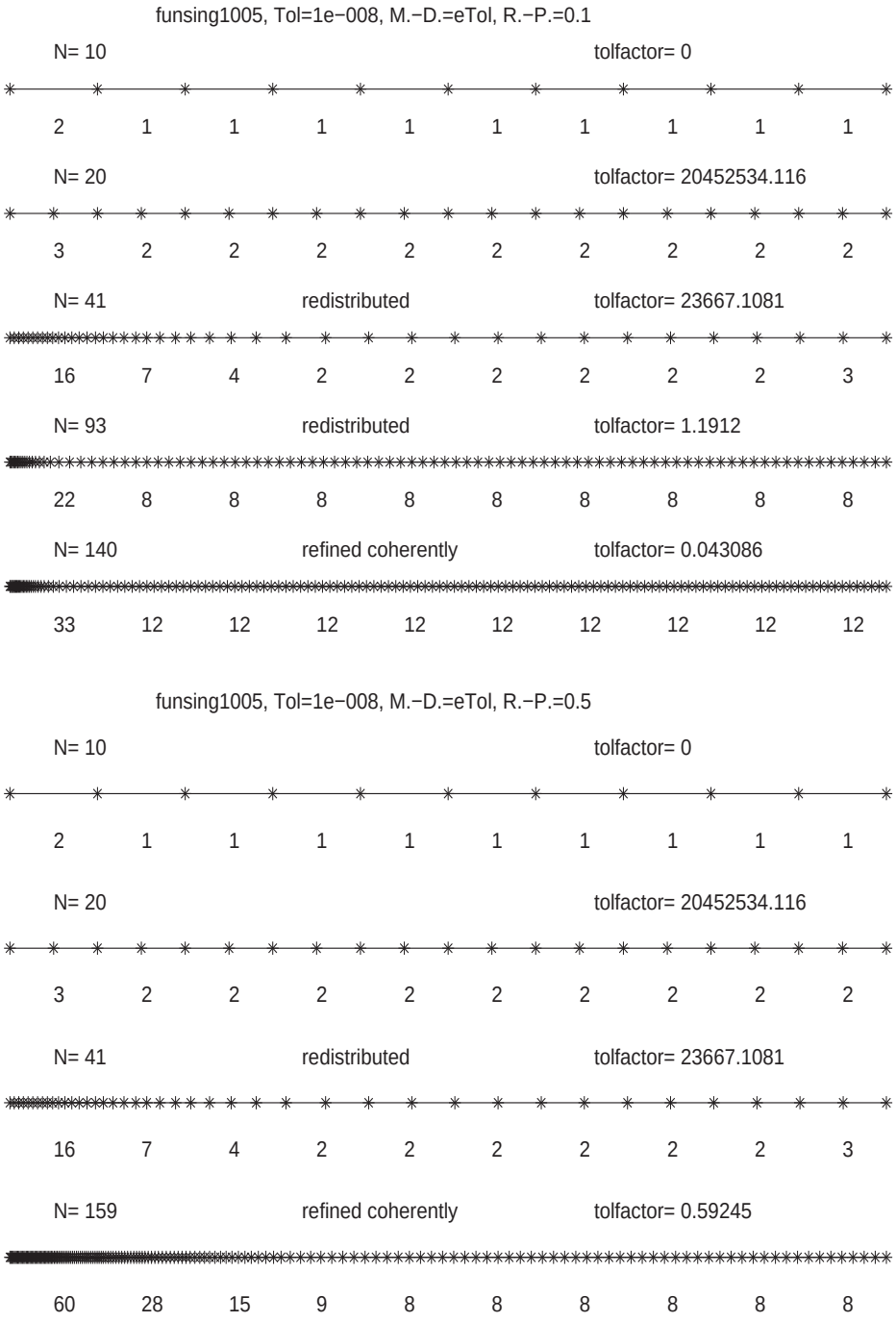


Figure 3.6: This figure shows the grid evolution of Bvp 1005 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$ and $\frac{1}{2}$ for eTol.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing15	10^{-8}	All	$\frac{1}{10}$	26	0.140
funsing15	10^{-8}	All	$\frac{1}{4}$	26	0.141
funsing15	10^{-8}	All	$\frac{1}{3}$	26	0.150
funsing15	10^{-8}	All	$\frac{1}{2}$	26	0.150
funsing15	10^{-8}	TolOpt	$\frac{1}{10}$	23	0.120
funsing15	10^{-8}	TolOpt	$\frac{1}{4}$	26	0.140
funsing15	10^{-8}	TolOpt	$\frac{1}{3}$	26	0.130
funsing15	10^{-8}	TolOpt	$\frac{1}{2}$	26	0.140
funsing15	10^{-8}	eTol	$\frac{1}{10}$	23	0.120
funsing15	10^{-8}	eTol	$\frac{1}{4}$	26	0.140
funsing15	10^{-8}	eTol	$\frac{1}{3}$	26	0.150
funsing15	10^{-8}	eTol	$\frac{1}{2}$	26	0.160

Table 3.19: Results for the combined test for Risk-Premium and Mesh-Distribution. $\frac{1}{10}$ is the best choice here.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing15	10^{-6}	All	$\frac{1}{10}$	33	0.120
funsing15	10^{-6}	All	$\frac{1}{4}$	33	0.120
funsing15	10^{-6}	All	$\frac{1}{3}$	33	0.140
funsing15	10^{-6}	All	$\frac{1}{2}$	33	0.130
funsing15	10^{-6}	TolOpt	$\frac{1}{10}$	27	0.110
funsing15	10^{-6}	TolOpt	$\frac{1}{4}$	33	0.120
funsing15	10^{-6}	TolOpt	$\frac{1}{3}$	33	0.120
funsing15	10^{-6}	TolOpt	$\frac{1}{2}$	33	0.130
funsing15	10^{-6}	eTol	$\frac{1}{10}$	28	0.130
funsing15	10^{-6}	eTol	$\frac{1}{4}$	33	0.140
funsing15	10^{-6}	eTol	$\frac{1}{3}$	33	0.120
funsing15	10^{-6}	eTol	$\frac{1}{2}$	33	0.141

Table 3.20: Results for the combined test for Risk-Premium and Mesh-Distribution. Again $\frac{1}{10}$ shows the best behavior, especially when looking at the number of meshpoints.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing15	10^{-3}	All	$\frac{1}{10}$	33	0.130
funsing15	10^{-3}	All	$\frac{1}{4}$	33	0.120
funsing15	10^{-3}	All	$\frac{1}{3}$	33	0.130
funsing15	10^{-3}	All	$\frac{1}{2}$	33	0.130
funsing15	10^{-3}	TolOpt	$\frac{1}{10}$	41	0.170
funsing15	10^{-3}	TolOpt	$\frac{1}{4}$	33	0.110
funsing15	10^{-3}	TolOpt	$\frac{1}{3}$	33	0.110
funsing15	10^{-3}	TolOpt	$\frac{1}{2}$	33	0.130
funsing15	10^{-3}	eTol	$\frac{1}{10}$	26	0.110
funsing15	10^{-3}	eTol	$\frac{1}{4}$	33	0.120
funsing15	10^{-3}	eTol	$\frac{1}{3}$	33	0.120
funsing15	10^{-3}	eTol	$\frac{1}{2}$	33	0.131

Table 3.21: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt and $\frac{1}{10}$ is quite risky, eTol shows best results with this setting of Risk-Premium.

We observe similar results for Bvp 15. For the mild tolerance TolOpt shows worst performance with $\frac{1}{10}$, eTol works in a best way.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing2002	10^{-8}	All	$\frac{1}{10}$	54	0.210
funsing2002	10^{-8}	All	$\frac{1}{4}$	54	0.230
funsing2002	10^{-8}	All	$\frac{1}{3}$	54	0.231
funsing2002	10^{-8}	All	$\frac{1}{2}$	60	0.170
funsing2002	10^{-8}	TolOpt	$\frac{1}{10}$	45	0.180
funsing2002	10^{-8}	TolOpt	$\frac{1}{4}$	45	0.200
funsing2002	10^{-8}	TolOpt	$\frac{1}{3}$	45	0.191
funsing2002	10^{-8}	TolOpt	$\frac{1}{2}$	60	0.150
funsing2002	10^{-8}	eTol	$\frac{1}{10}$	51	0.221
funsing2002	10^{-8}	eTol	$\frac{1}{4}$	51	0.220
funsing2002	10^{-8}	eTol	$\frac{1}{3}$	51	0.220
funsing2002	10^{-8}	eTol	$\frac{1}{2}$	60	0.170

Table 3.22: Results for the combined test for Risk-Premium and Mesh-Distribution. Though most meshpoints are required for $\frac{1}{2}$ the runtimes are very short. TolOpt is the best choice for this Bvp.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing2002	10^{-6}	All	$\frac{1}{10}$	75	0.211
funsing2002	10^{-6}	All	$\frac{1}{4}$	75	0.230
funsing2002	10^{-6}	All	$\frac{1}{3}$	75	0.210
funsing2002	10^{-6}	All	$\frac{1}{2}$	90	0.180
funsing2002	10^{-6}	TolOpt	$\frac{1}{10}$	59	0.181
funsing2002	10^{-6}	TolOpt	$\frac{1}{4}$	59	0.170
funsing2002	10^{-6}	TolOpt	$\frac{1}{3}$	59	0.170
funsing2002	10^{-6}	TolOpt	$\frac{1}{2}$	59	0.190
funsing2002	10^{-6}	eTol	$\frac{1}{10}$	68	0.200
funsing2002	10^{-6}	eTol	$\frac{1}{4}$	68	0.201
funsing2002	10^{-6}	eTol	$\frac{1}{3}$	68	0.211
funsing2002	10^{-6}	eTol	$\frac{1}{2}$	90	0.160

Table 3.23: Results for the combined test for Risk-Premium and Mesh-Distribution. The fewest number of meshpoints is needed for TolOpt and the slowest runtime is observed for eTol and $\frac{1}{2}$.

Figures 3.7 - 3.8 show the reason of the good runtimes at higher numbers of meshpoints for All and eTol. While only one refinement is needed with the higher Risk-Premium, one additional is needed for all the other choices. Interestingly with the Mesh-Distribution at TolOpt quite short runtimes are achieved even with two refinements. This is caused by the lowest number of meshpoints required to satisfy the tolerance.

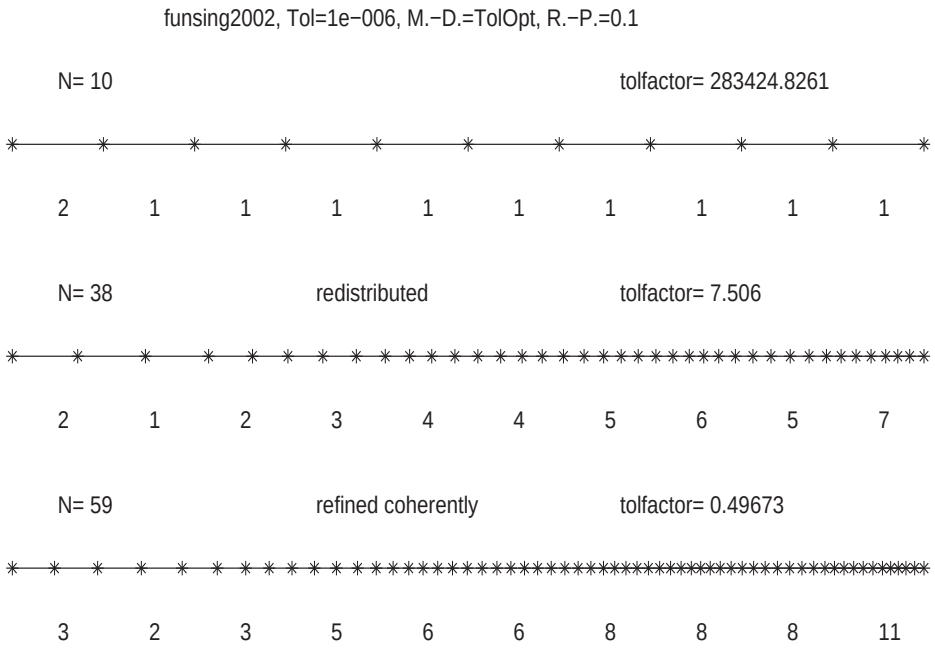


Figure 3.7: This figure shows the grid evolution of BVP 2002 for the tolerance set to 10^{-6} and Risk-Premium at $\frac{1}{10}$ for TolOpt.

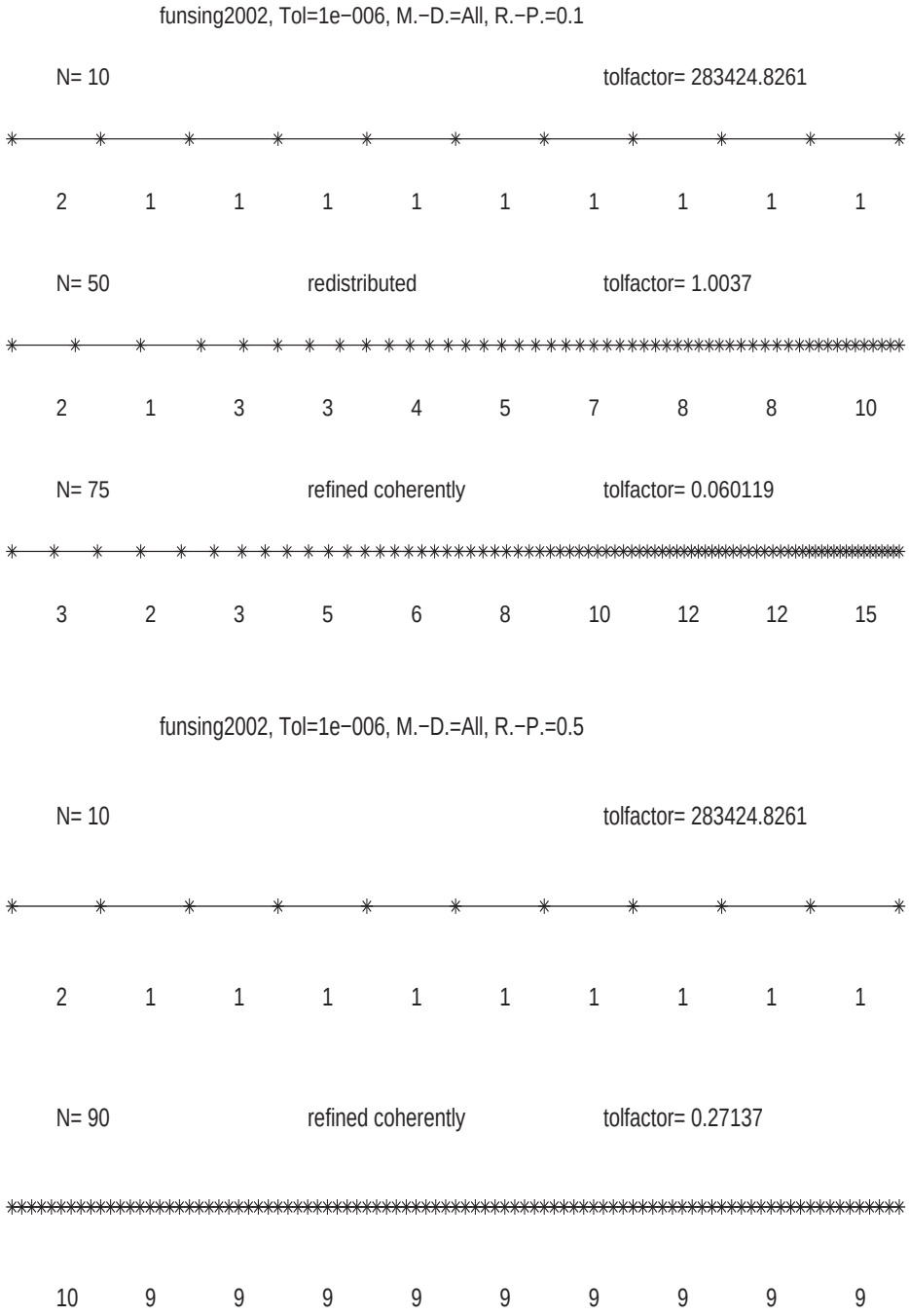


Figure 3.8: This figure shows the grid evolution of Bvp 2002 for the tolerance set to 10^{-6} and Risk-Premium at $\frac{1}{10}$ and $\frac{1}{2}$ for All.

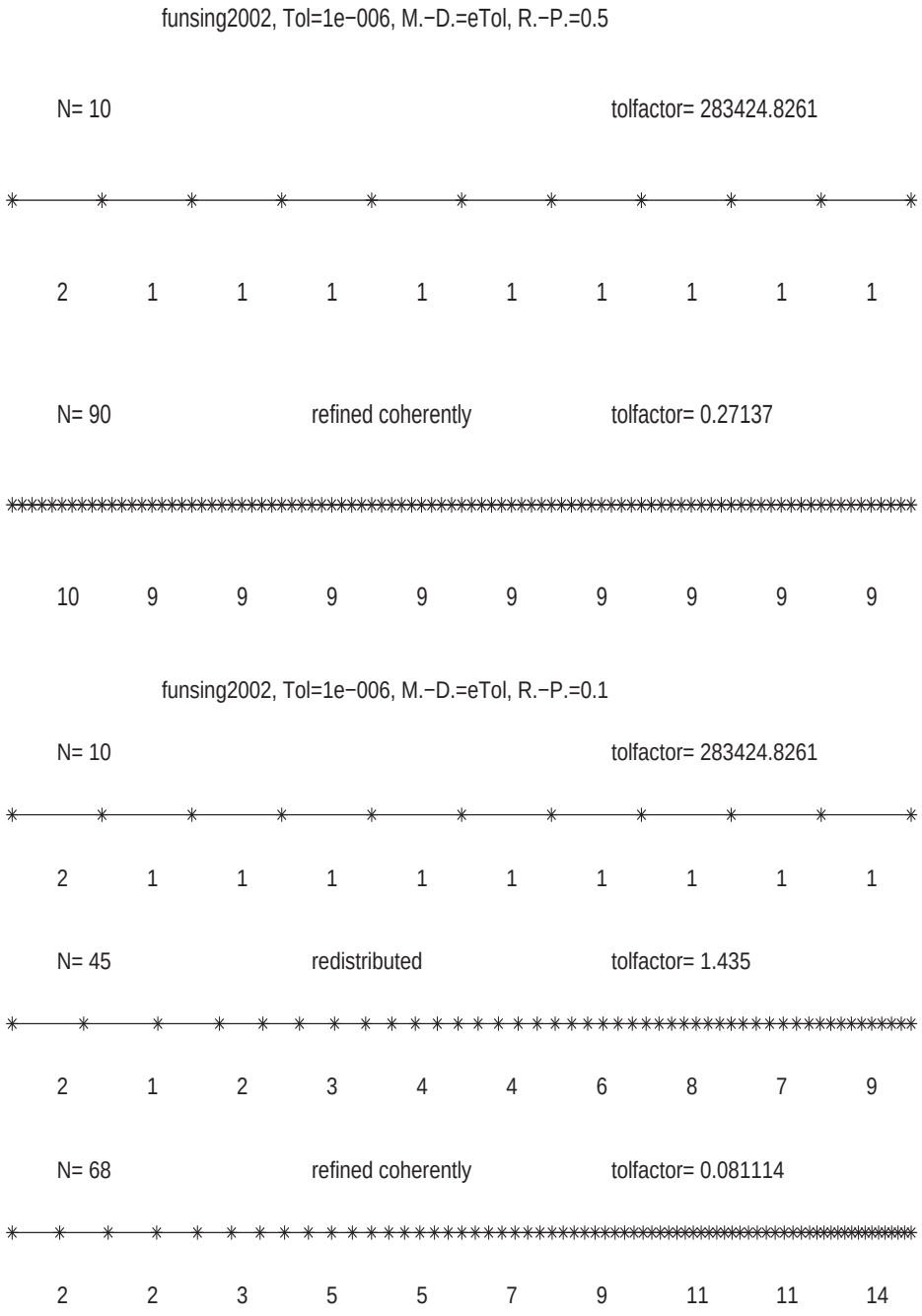


Figure 3.9: This figure shows the grid evolution of BVP 2002 for the tolerance set to 10^{-6} and Risk-Premium at $\frac{1}{10}$ and $\frac{1}{2}$ for eTol.

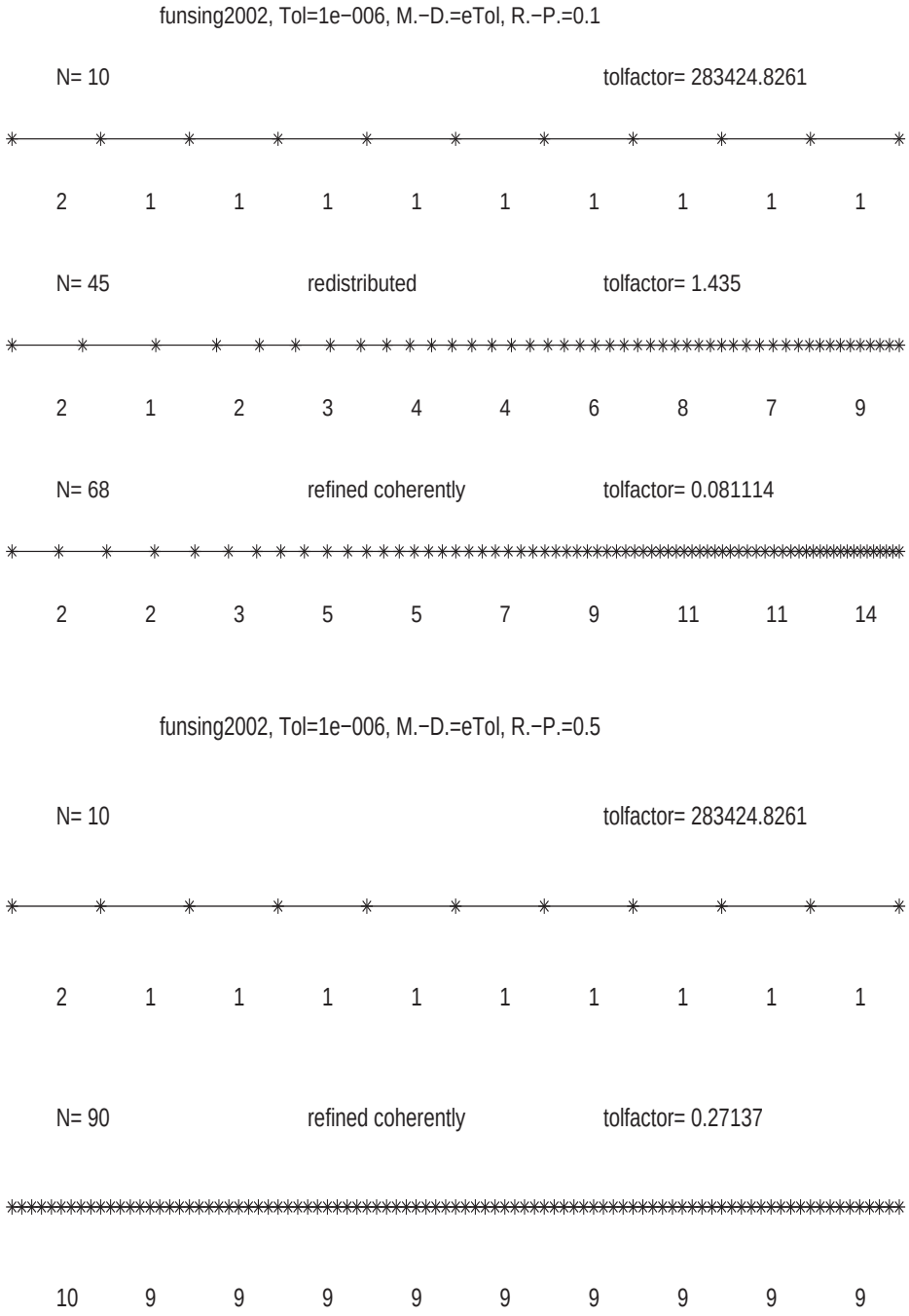


Figure 3.10: This figure shows the grid evolution of Bvp 2002 for the tolerance set to 10^{-6} and Risk-Premium at $\frac{1}{10}$ and $\frac{1}{2}$ for eTol.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing2002	10^{-3}	All	$\frac{1}{10}$	34	0.091
funsing2002	10^{-3}	All	$\frac{1}{4}$	34	0.100
funsing2002	10^{-3}	All	$\frac{1}{3}$	71	0.180
funsing2002	10^{-3}	All	$\frac{1}{2}$	71	0.180
funsing2002	10^{-3}	TolOpt	$\frac{1}{10}$	45	0.140
funsing2002	10^{-3}	TolOpt	$\frac{1}{4}$	45	0.130
funsing2002	10^{-3}	TolOpt	$\frac{1}{3}$	45	0.130
funsing2002	10^{-3}	TolOpt	$\frac{1}{2}$	71	0.171
funsing2002	10^{-3}	eTol	$\frac{1}{10}$	45	0.131
funsing2002	10^{-3}	eTol	$\frac{1}{4}$	45	0.150
funsing2002	10^{-3}	eTol	$\frac{1}{3}$	45	0.150
funsing2002	10^{-3}	eTol	$\frac{1}{2}$	71	0.190

Table 3.24: Results for the combined test for Risk-Premium and Mesh-Distribution. All shows best performance for the lower Risk-Premiums. $\frac{1}{2}$ needs the highest number of meshpoints to satisfy the tolerance.

TolOpt wins for the strict tolerance and All performs very well for the mild one, at least for the lower Risk-Premiums. eTol is stable but cannot compete with the fastest times overall. Again when Risk-Premium is set to $\frac{1}{2}$ the results are worst for tolerance 10^{-3} but for the stricter tolerances this setting does best with any setting of Mesh-Distribution though more meshpoints are required.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing2008	10^{-8}	All	$\frac{1}{10}$	133	0.381
funsing2008	10^{-8}	All	$\frac{1}{4}$	133	0.401
funsing2008	10^{-8}	All	$\frac{1}{3}$	180	0.551
funsing2008	10^{-8}	All	$\frac{1}{2}$	180	0.530
funsing2008	10^{-8}	TolOpt	$\frac{1}{10}$	153	0.651
funsing2008	10^{-8}	TolOpt	$\frac{1}{4}$	153	0.651
funsing2008	10^{-8}	TolOpt	$\frac{1}{3}$	153	0.651
funsing2008	10^{-8}	TolOpt	$\frac{1}{2}$	180	0.541
funsing2008	10^{-8}	eTol	$\frac{1}{10}$	118	0.340
funsing2008	10^{-8}	eTol	$\frac{1}{4}$	118	0.330
funsing2008	10^{-8}	eTol	$\frac{1}{3}$	118	0.360
funsing2008	10^{-8}	eTol	$\frac{1}{2}$	180	0.551

Table 3.25: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt is very slow, eTol is quite fast but All with Risk-Premium set to $\frac{1}{10}$ is the most competitive.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing2008	10^{-6}	All	$\frac{1}{10}$	159	0.520
funsing2008	10^{-6}	All	$\frac{1}{4}$	159	0.521
funsing2008	10^{-6}	All	$\frac{1}{3}$	159	0.530
funsing2008	10^{-6}	All	$\frac{1}{2}$	267	0.891
funsing2008	10^{-6}	TolOpt	$\frac{1}{10}$	158	0.621
funsing2008	10^{-6}	TolOpt	$\frac{1}{4}$	158	0.611
funsing2008	10^{-6}	TolOpt	$\frac{1}{3}$	147	0.461
funsing2008	10^{-6}	TolOpt	$\frac{1}{2}$	147	0.431
funsing2008	10^{-6}	eTol	$\frac{1}{10}$	138	0.450
funsing2008	10^{-6}	eTol	$\frac{1}{4}$	138	0.450
funsing2008	10^{-6}	eTol	$\frac{1}{3}$	138	0.491
funsing2008	10^{-6}	eTol	$\frac{1}{2}$	267	0.901

Table 3.26: Results for the combined test for Risk-Premium and Mesh-Distribution. Mesh-Distribution set to eTol combined with the lower Risk-Premiums has the best behavior for this Bvp. TolOpt surprises with better results at the higher Risk-Premiums and All is a bit slower but comparable to eTol.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing2008	10^{-3}	All	$\frac{1}{10}$	90	0.170
funsing2008	10^{-3}	All	$\frac{1}{4}$	176	0.401
funsing2008	10^{-3}	All	$\frac{1}{3}$	176	0.390
funsing2008	10^{-3}	All	$\frac{1}{2}$	176	0.380
funsing2008	10^{-3}	TolOpt	$\frac{1}{10}$	135	0.351
funsing2008	10^{-3}	TolOpt	$\frac{1}{4}$	118	0.250
funsing2008	10^{-3}	TolOpt	$\frac{1}{3}$	118	0.261
funsing2008	10^{-3}	TolOpt	$\frac{1}{2}$	176	0.360
funsing2008	10^{-3}	eTol	$\frac{1}{10}$	114	0.260
funsing2008	10^{-3}	eTol	$\frac{1}{4}$	114	0.280
funsing2008	10^{-3}	eTol	$\frac{1}{3}$	114	0.281
funsing2008	10^{-3}	eTol	$\frac{1}{2}$	176	0.431

Table 3.27: Results for the combined test for Risk-Premium and Mesh-Distribution. All with $\frac{1}{10}$ is the only configuration that needs less than 100 mesh-points and has the best runtime overall. TolOpt has the best runtimes for the middle Risk-Premiums. Risk-Premium set to $\frac{1}{2}$ is far away from the best results for each setting of Mesh-Distribution.

Very good results are achieved with All and $\frac{1}{10}$ but even eTol is excellent for the lower Risk-Premiums. TolOpt is worst for the strict tolerance and can only compete at the higher Risk-Premiums for the other tolerances.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing21a	10^{-8}	All	$\frac{1}{10}$	10	0.110
funsing21a	10^{-8}	All	$\frac{1}{4}$	10	0.110
funsing21a	10^{-8}	All	$\frac{1}{3}$	10	0.110
funsing21a	10^{-8}	All	$\frac{1}{2}$	10	0.130
funsing21a	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.090
funsing21a	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.121
funsing21a	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.100
funsing21a	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.120
funsing21a	10^{-8}	eTol	$\frac{1}{10}$	10	0.101
funsing21a	10^{-8}	eTol	$\frac{1}{4}$	10	0.120
funsing21a	10^{-8}	eTol	$\frac{1}{3}$	10	0.120
funsing21a	10^{-8}	eTol	$\frac{1}{2}$	10	0.110

Table 3.28: Results for the combined test for Risk-Premium and Mesh-Distribution. All configurations are fast and stable.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing21a	10^{-6}	All	$\frac{1}{10}$	10	0.090
funsing21a	10^{-6}	All	$\frac{1}{4}$	10	0.090
funsing21a	10^{-6}	All	$\frac{1}{3}$	10	0.101
funsing21a	10^{-6}	All	$\frac{1}{2}$	10	0.101
funsing21a	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.080
funsing21a	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.100
funsing21a	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.090
funsing21a	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.100
funsing21a	10^{-6}	eTol	$\frac{1}{10}$	10	0.090
funsing21a	10^{-6}	eTol	$\frac{1}{4}$	10	0.080
funsing21a	10^{-6}	eTol	$\frac{1}{3}$	10	0.110
funsing21a	10^{-6}	eTol	$\frac{1}{2}$	10	0.100

Table 3.29: Results for the combined test for Risk-Premium and Mesh-Distribution. Every configuration yields a result very quickly. All grids are the same.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing21a	10^{-3}	All	$\frac{1}{10}$	5	0.070
funsing21a	10^{-3}	All	$\frac{1}{4}$	5	0.080
funsing21a	10^{-3}	All	$\frac{1}{3}$	5	0.060
funsing21a	10^{-3}	All	$\frac{1}{2}$	5	0.070
funsing21a	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.060
funsing21a	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.080
funsing21a	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.070
funsing21a	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.080
funsing21a	10^{-3}	eTol	$\frac{1}{10}$	5	0.071
funsing21a	10^{-3}	eTol	$\frac{1}{4}$	5	0.060
funsing21a	10^{-3}	eTol	$\frac{1}{3}$	5	0.070
funsing21a	10^{-3}	eTol	$\frac{1}{2}$	5	0.070

Table 3.30: Results for the combined test for Risk-Premium and Mesh-Distribution. No problems are observed for any configuration.

The results for Bvp 21a are so close and the runtimes so short that the only observation one can make is that all configurations are fast and stable.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing400	10^{-8}	All	$\frac{1}{10}$	10	0.070
funsing400	10^{-8}	All	$\frac{1}{4}$	10	0.050
funsing400	10^{-8}	All	$\frac{1}{3}$	10	0.060
funsing400	10^{-8}	All	$\frac{1}{2}$	10	0.050
funsing400	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.040
funsing400	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.050
funsing400	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.051
funsing400	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.060
funsing400	10^{-8}	eTol	$\frac{1}{10}$	10	0.061
funsing400	10^{-8}	eTol	$\frac{1}{4}$	10	0.050
funsing400	10^{-8}	eTol	$\frac{1}{3}$	10	0.070
funsing400	10^{-8}	eTol	$\frac{1}{2}$	10	0.050

Table 3.31: Results for the combined test for Risk-Premium and Mesh-Distribution. No problems are observed for any configuration.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing400	10^{-6}	All	$\frac{1}{10}$	10	0.060
funsing400	10^{-6}	All	$\frac{1}{4}$	10	0.040
funsing400	10^{-6}	All	$\frac{1}{3}$	10	0.060
funsing400	10^{-6}	All	$\frac{1}{2}$	10	0.070
funsing400	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.060
funsing400	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.060
funsing400	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.050
funsing400	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.050
funsing400	10^{-6}	eTol	$\frac{1}{10}$	10	0.060
funsing400	10^{-6}	eTol	$\frac{1}{4}$	10	0.050
funsing400	10^{-6}	eTol	$\frac{1}{3}$	10	0.040
funsing400	10^{-6}	eTol	$\frac{1}{2}$	10	0.060

Table 3.32: Results for the combined test for Risk-Premium and Mesh-Distribution. No problem is observed for any configuration.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing400	10^{-3}	All	$\frac{1}{10}$	5	0.050
funsing400	10^{-3}	All	$\frac{1}{4}$	5	0.060
funsing400	10^{-3}	All	$\frac{1}{3}$	5	0.060
funsing400	10^{-3}	All	$\frac{1}{2}$	5	0.040
funsing400	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.050
funsing400	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.050
funsing400	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.040
funsing400	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.050
funsing400	10^{-3}	eTol	$\frac{1}{10}$	5	0.040
funsing400	10^{-3}	eTol	$\frac{1}{4}$	5	0.050
funsing400	10^{-3}	eTol	$\frac{1}{3}$	5	0.050
funsing400	10^{-3}	eTol	$\frac{1}{2}$	5	0.051

Table 3.33: Results for the combined test for Risk-Premium and Mesh-Distribution. No problem is observed for any configuration.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing500	10^{-8}	All	$\frac{1}{10}$	10	0.040
funsing500	10^{-8}	All	$\frac{1}{4}$	10	0.040
funsing500	10^{-8}	All	$\frac{1}{3}$	10	0.040
funsing500	10^{-8}	All	$\frac{1}{2}$	10	0.051
funsing500	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.060
funsing500	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.040
funsing500	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.040
funsing500	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.050
funsing500	10^{-8}	eTol	$\frac{1}{10}$	10	0.051
funsing500	10^{-8}	eTol	$\frac{1}{4}$	10	0.060
funsing500	10^{-8}	eTol	$\frac{1}{3}$	10	0.040
funsing500	10^{-8}	eTol	$\frac{1}{2}$	10	0.040

Table 3.34: Results for the combined test for Risk-Premium and Mesh-Distribution. Like Bvp 400 this Bvp is rather easy to solve. So the routine does not have any problems.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing500	10^{-6}	All	$\frac{1}{10}$	10	0.040
funsing500	10^{-6}	All	$\frac{1}{4}$	10	0.030
funsing500	10^{-6}	All	$\frac{1}{3}$	10	0.060
funsing500	10^{-6}	All	$\frac{1}{2}$	10	0.050
funsing500	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.050
funsing500	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.040
funsing500	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.040
funsing500	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.030
funsing500	10^{-6}	eTol	$\frac{1}{10}$	10	0.040
funsing500	10^{-6}	eTol	$\frac{1}{4}$	10	0.060
funsing500	10^{-6}	eTol	$\frac{1}{3}$	10	0.060
funsing500	10^{-6}	eTol	$\frac{1}{2}$	10	0.040

Table 3.35: Results for the combined test for Risk-Premium and Mesh-Distribution. Quick results and no problems are observed here.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing500	10^{-3}	All	$\frac{1}{10}$	5	0.040
funsing500	10^{-3}	All	$\frac{1}{4}$	5	0.030
funsing500	10^{-3}	All	$\frac{1}{3}$	5	0.030
funsing500	10^{-3}	All	$\frac{1}{2}$	5	0.030
funsing500	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.040
funsing500	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.050
funsing500	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.030
funsing500	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.050
funsing500	10^{-3}	eTol	$\frac{1}{10}$	5	0.050
funsing500	10^{-3}	eTol	$\frac{1}{4}$	5	0.040
funsing500	10^{-3}	eTol	$\frac{1}{3}$	5	0.060
funsing500	10^{-3}	eTol	$\frac{1}{2}$	5	0.041

Table 3.36: Results for the combined test for Risk-Premium and Mesh-Distribution. No problems for this BVP.

The results for BVPS 400 and 500 are very fast and do not show any problems for any configuration. This is because both BVPS are solved exactly. The solution is a polynomial of degree 5.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing54	10^{-8}	All	$\frac{1}{10}$	1566	80.736
funsing54	10^{-8}	TolOpt	$\frac{1}{10}$	1416	66.155
funsing54	10^{-8}	eTol	$\frac{1}{10}$	849	26.368
funsing54	10^{-8}	eTol	$\frac{1}{4}$	1850	111.781

Table 3.37: Results for the combined test for Risk-Premium and Mesh-Distribution. The one thing to see here is that $\frac{1}{10}$ is stable.

Table 3.37 shows the most interesting results of all. The strict tolerance and the quite difficult BVP are a serious problem for SBVP 2.0. Only when Risk-Premium is set to $\frac{1}{10}$ all three settings for Mesh-Distribution deliver results. This does not mean that the routine does not solve it at all for the other Risk-Premiums, it only means that more than 3500 meshpoints are required, which was the upper limit for the tests reported here.

The conclusion drawn from this problem is that Risk-Premium at $\frac{1}{10}$ is the one to be chosen if there is not another configuration that is better for most of the other problems. Remarkably, eTol performs very well with this Risk-Premium while for the other values the calculations are three times longer. The results are very sensitive with respect to the choice of Risk-Premium. This can be seen from the result for eTol: For Risk-Premium set to $\frac{1}{4}$ the runtime is four times longer and the number of meshpoints is doubled.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing54	10^{-6}	All	$\frac{1}{10}$	149	1.642
funsing54	10^{-6}	All	$\frac{1}{4}$	203	2.193
funsing54	10^{-6}	All	$\frac{1}{3}$	288	3.405
funsing54	10^{-6}	All	$\frac{1}{2}$	288	3.275
funsing54	10^{-6}	TolOpt	$\frac{1}{10}$	153	1.582
funsing54	10^{-6}	TolOpt	$\frac{1}{4}$	153	1.572
funsing54	10^{-6}	TolOpt	$\frac{1}{3}$	288	3.295
funsing54	10^{-6}	TolOpt	$\frac{1}{2}$	288	3.265
funsing54	10^{-6}	eTol	$\frac{1}{10}$	147	1.583
funsing54	10^{-6}	eTol	$\frac{1}{4}$	147	1.623
funsing54	10^{-6}	eTol	$\frac{1}{3}$	197	2.103
funsing54	10^{-6}	eTol	$\frac{1}{2}$	288	3.314

Table 3.38: Results for the combined test for Risk-Premium and Mesh-Distribution. Obviously the best results are achieved for Risk-Premium set to $\frac{1}{10}$ and the worst for the higher settings.

Figure 3.11 shows the advantage of the lower Risk-Premium. Earlier coherent refinements, especially in the case of Risk-Premium $\frac{1}{3}$ yield an increased number of meshpoints. The distribution of the meshpoints for Risk-Premium $\frac{1}{10}$ seems to take the characteristics of the solution curve into account. For the other Risk-Premiums the grids are quite balanced.

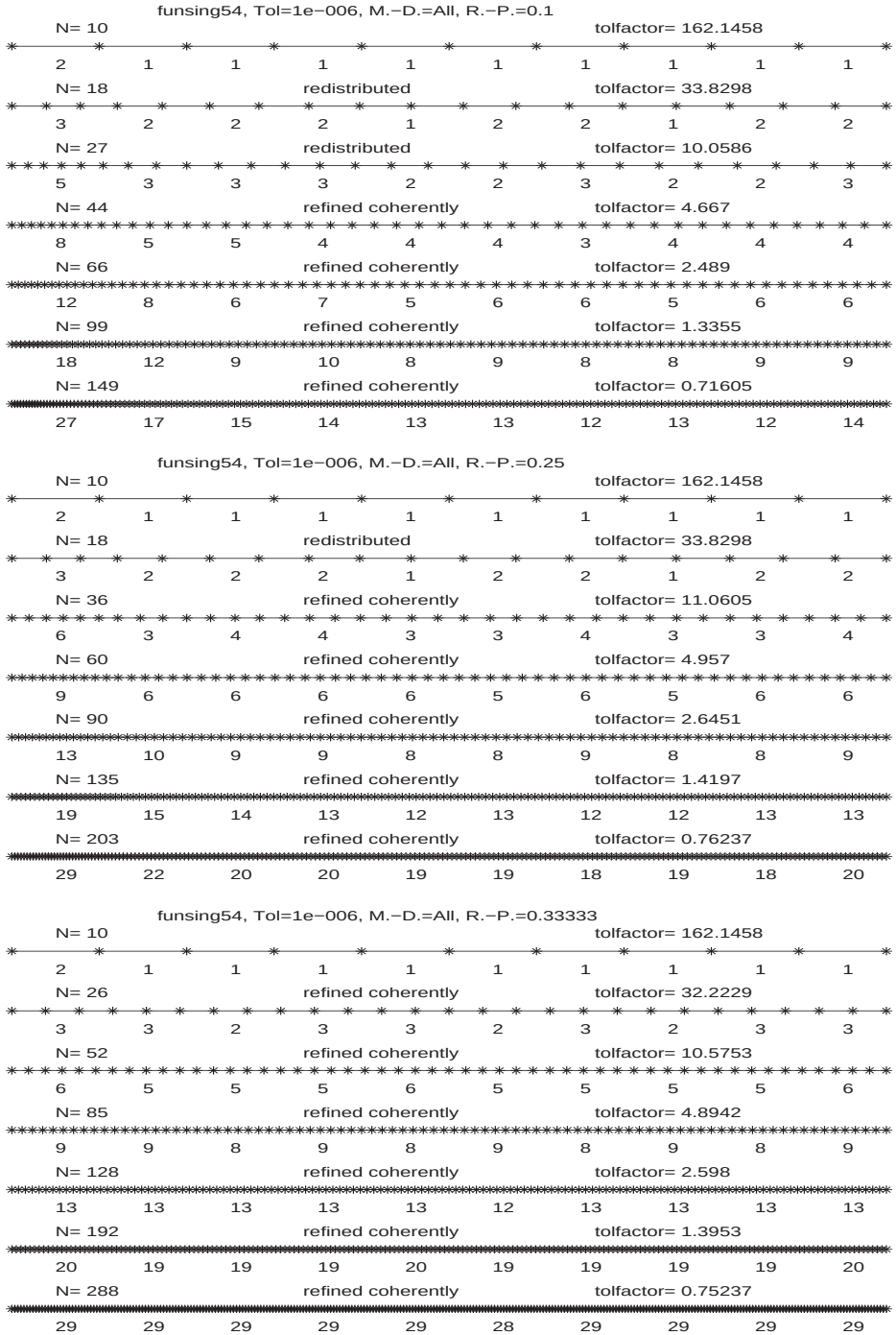


Figure 3.11: This figure shows the grid evolution of BVP 54 for the tolerance set to 10^{-6} and Risk-Premium at $\frac{1}{10}$, $\frac{1}{4}$ and $\frac{1}{3}$.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing54	10^{-3}	All	$\frac{1}{10}$	8	0.100
funsing54	10^{-3}	All	$\frac{1}{4}$	8	0.110
funsing54	10^{-3}	All	$\frac{1}{3}$	8	0.131
funsing54	10^{-3}	All	$\frac{1}{2}$	8	0.110
funsing54	10^{-3}	TolOpt	$\frac{1}{10}$	8	0.120
funsing54	10^{-3}	TolOpt	$\frac{1}{4}$	8	0.110
funsing54	10^{-3}	TolOpt	$\frac{1}{3}$	8	0.100
funsing54	10^{-3}	TolOpt	$\frac{1}{2}$	8	0.120
funsing54	10^{-3}	eTol	$\frac{1}{10}$	8	0.110
funsing54	10^{-3}	eTol	$\frac{1}{4}$	8	0.120
funsing54	10^{-3}	eTol	$\frac{1}{3}$	8	0.110
funsing54	10^{-3}	eTol	$\frac{1}{2}$	8	0.101

Table 3.39: Results for the combined test for Risk-Premium and Mesh-Distribution. As every configuration finishes after one refinement (8 meshpoints) the runtimes are highly comparable.

For the lower tolerances, especially 10^{-6}, one can see that the lower Risk-Premiums are better here and that $\frac{1}{10}$ is best. Though all three settings for Mesh-Distribution have about the same runtimes for this Risk-Premium, there is a minor disadvantage for All.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing55	10^{-8}	All	$\frac{1}{10}$	10	0.100
funsing55	10^{-8}	All	$\frac{1}{4}$	10	0.120
funsing55	10^{-8}	All	$\frac{1}{3}$	10	0.110
funsing55	10^{-8}	All	$\frac{1}{2}$	10	0.111
funsing55	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.101
funsing55	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.110
funsing55	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.100
funsing55	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.100
funsing55	10^{-8}	eTol	$\frac{1}{10}$	10	0.110
funsing55	10^{-8}	eTol	$\frac{1}{4}$	10	0.110
funsing55	10^{-8}	eTol	$\frac{1}{3}$	10	0.110
funsing55	10^{-8}	eTol	$\frac{1}{2}$	10	0.121

Table 3.40: Results for the combined test for Risk-Premium and Mesh-Distribution. Bvp 55 causes no problems.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing55	10^{-6}	All	$\frac{1}{10}$	10	0.100
funsing55	10^{-6}	All	$\frac{1}{4}$	10	0.090
funsing55	10^{-6}	All	$\frac{1}{3}$	10	0.080
funsing55	10^{-6}	All	$\frac{1}{2}$	10	0.090
funsing55	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.070
funsing55	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.090
funsing55	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.090
funsing55	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.090
funsing55	10^{-6}	eTol	$\frac{1}{10}$	10	0.080
funsing55	10^{-6}	eTol	$\frac{1}{4}$	10	0.080
funsing55	10^{-6}	eTol	$\frac{1}{3}$	10	0.100
funsing55	10^{-6}	eTol	$\frac{1}{2}$	10	0.080

Table 3.41: Results for the combined test for Risk-Premium and Mesh-Distribution. There is no difference in the results.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing55	10^{-3}	All	$\frac{1}{10}$	5	0.070
funsing55	10^{-3}	All	$\frac{1}{4}$	5	0.050
funsing55	10^{-3}	All	$\frac{1}{3}$	5	0.060
funsing55	10^{-3}	All	$\frac{1}{2}$	5	0.060
funsing55	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.051
funsing55	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.060
funsing55	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.060
funsing55	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.071
funsing55	10^{-3}	eTol	$\frac{1}{10}$	5	0.060
funsing55	10^{-3}	eTol	$\frac{1}{4}$	5	0.070
funsing55	10^{-3}	eTol	$\frac{1}{3}$	5	0.050
funsing55	10^{-3}	eTol	$\frac{1}{2}$	5	0.070

Table 3.42: Results for the combined test for Risk-Premium and Mesh-Distribution. Again there is no difference in the results.

BVP 55 is solved quickly and without any problem.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing56	10^{-8}	All	$\frac{1}{10}$	80	2.423
funsing56	10^{-8}	All	$\frac{1}{4}$	80	2.424
funsing56	10^{-8}	All	$\frac{1}{3}$	80	2.444
funsing56	10^{-8}	All	$\frac{1}{2}$	80	2.433
funsing56	10^{-8}	TolOpt	$\frac{1}{10}$	80	2.433
funsing56	10^{-8}	TolOpt	$\frac{1}{4}$	80	2.453
funsing56	10^{-8}	TolOpt	$\frac{1}{3}$	80	2.424
funsing56	10^{-8}	TolOpt	$\frac{1}{2}$	80	2.434
funsing56	10^{-8}	eTol	$\frac{1}{10}$	80	2.444
funsing56	10^{-8}	eTol	$\frac{1}{4}$	80	2.434
funsing56	10^{-8}	eTol	$\frac{1}{3}$	80	2.444
funsing56	10^{-8}	eTol	$\frac{1}{2}$	80	2.443

Table 3.43: Results for the combined test for Risk-Premium and Mesh-Distribution. There is hardly a difference in the results.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing56	10^{-6}	All	$\frac{1}{10}$	80	1.713
funsing56	10^{-6}	All	$\frac{1}{4}$	80	1.733
funsing56	10^{-6}	All	$\frac{1}{3}$	80	1.722
funsing56	10^{-6}	All	$\frac{1}{2}$	80	1.703
funsing56	10^{-6}	TolOpt	$\frac{1}{10}$	80	1.702
funsing56	10^{-6}	TolOpt	$\frac{1}{4}$	80	1.702
funsing56	10^{-6}	TolOpt	$\frac{1}{3}$	80	1.712
funsing56	10^{-6}	TolOpt	$\frac{1}{2}$	80	1.712
funsing56	10^{-6}	eTol	$\frac{1}{10}$	80	1.712
funsing56	10^{-6}	eTol	$\frac{1}{4}$	80	1.813
funsing56	10^{-6}	eTol	$\frac{1}{3}$	80	1.722
funsing56	10^{-6}	eTol	$\frac{1}{2}$	80	1.732

Table 3.44: Results for the combined test for Risk-Premium and Mesh-Distribution. Again the different runtimes are nearly equal.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing56	10^{-3}	All	$\frac{1}{10}$	120	1.653
funsing56	10^{-3}	All	$\frac{1}{4}$	120	1.673
funsing56	10^{-3}	All	$\frac{1}{3}$	120	1.633
funsing56	10^{-3}	All	$\frac{1}{2}$	120	1.643
funsing56	10^{-3}	TolOpt	$\frac{1}{10}$	120	1.612
funsing56	10^{-3}	TolOpt	$\frac{1}{4}$	120	1.662
funsing56	10^{-3}	TolOpt	$\frac{1}{3}$	120	1.612
funsing56	10^{-3}	TolOpt	$\frac{1}{2}$	120	1.623
funsing56	10^{-3}	eTol	$\frac{1}{10}$	120	1.622
funsing56	10^{-3}	eTol	$\frac{1}{4}$	120	1.563
funsing56	10^{-3}	eTol	$\frac{1}{3}$	120	1.633
funsing56	10^{-3}	eTol	$\frac{1}{2}$	120	1.653

Table 3.45: Results for the combined test for Risk-Premium and Mesh-Distribution. As seen before, BVP 56 does not help to decide between the different settings.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing6001	10^{-8}	All	$\frac{1}{10}$	15	0.230
funsing6001	10^{-8}	All	$\frac{1}{4}$	18	0.270
funsing6001	10^{-8}	All	$\frac{1}{3}$	18	0.241
funsing6001	10^{-8}	All	$\frac{1}{2}$	18	0.260
funsing6001	10^{-8}	TolOpt	$\frac{1}{10}$	15	0.230
funsing6001	10^{-8}	TolOpt	$\frac{1}{4}$	18	0.260
funsing6001	10^{-8}	TolOpt	$\frac{1}{3}$	18	0.261
funsing6001	10^{-8}	TolOpt	$\frac{1}{2}$	18	0.241
funsing6001	10^{-8}	eTol	$\frac{1}{10}$	15	0.250
funsing6001	10^{-8}	eTol	$\frac{1}{4}$	18	0.260
funsing6001	10^{-8}	eTol	$\frac{1}{3}$	18	0.281
funsing6001	10^{-8}	eTol	$\frac{1}{2}$	18	0.260

Table 3.46: Results for the combined test for Risk-Premium and Mesh-Distribution. $\frac{1}{10}$ is best here, and eTol seems least effective.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing6001	10^{-6}	All	$\frac{1}{10}$	15	0.190
funsing6001	10^{-6}	All	$\frac{1}{4}$	17	0.211
funsing6001	10^{-6}	All	$\frac{1}{3}$	17	0.210
funsing6001	10^{-6}	All	$\frac{1}{2}$	17	0.211
funsing6001	10^{-6}	TolOpt	$\frac{1}{10}$	15	0.200
funsing6001	10^{-6}	TolOpt	$\frac{1}{4}$	17	0.191
funsing6001	10^{-6}	TolOpt	$\frac{1}{3}$	17	0.210
funsing6001	10^{-6}	TolOpt	$\frac{1}{2}$	17	0.210
funsing6001	10^{-6}	eTol	$\frac{1}{10}$	15	0.211
funsing6001	10^{-6}	eTol	$\frac{1}{4}$	17	0.210
funsing6001	10^{-6}	eTol	$\frac{1}{3}$	17	0.220
funsing6001	10^{-6}	eTol	$\frac{1}{2}$	17	0.210

Table 3.47: Results for the combined test for Risk-Premium and Mesh-Distribution. Mesh-Distribution at eTol lags slightly behind the others here.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing6001	10^{-3}	All	$\frac{1}{10}$	10	0.180
funsing6001	10^{-3}	All	$\frac{1}{4}$	10	0.171
funsing6001	10^{-3}	All	$\frac{1}{3}$	10	0.160
funsing6001	10^{-3}	All	$\frac{1}{2}$	10	0.181
funsing6001	10^{-3}	TolOpt	$\frac{1}{10}$	10	0.171
funsing6001	10^{-3}	TolOpt	$\frac{1}{4}$	10	0.160
funsing6001	10^{-3}	TolOpt	$\frac{1}{3}$	10	0.170
funsing6001	10^{-3}	TolOpt	$\frac{1}{2}$	10	0.170
funsing6001	10^{-3}	eTol	$\frac{1}{10}$	10	0.160
funsing6001	10^{-3}	eTol	$\frac{1}{4}$	10	0.170
funsing6001	10^{-3}	eTol	$\frac{1}{3}$	10	0.170
funsing6001	10^{-3}	eTol	$\frac{1}{2}$	10	0.180

Table 3.48: Results for the combined test for Risk-Premium and Mesh-Distribution. Hardly a difference in the results.

Again Risk-Premium at $\frac{1}{10}$ is the best choice (Table 3.46) and All is a bit better than eTol and TolOpt. Another result that was often observed is that All is losing at tolerance 10^{-3}.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing6002	10^{-8}	All	$\frac{1}{10}$	10	0.090
funsing6002	10^{-8}	All	$\frac{1}{4}$	10	0.110
funsing6002	10^{-8}	All	$\frac{1}{3}$	10	0.090
funsing6002	10^{-8}	All	$\frac{1}{2}$	10	0.110
funsing6002	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.090
funsing6002	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.101
funsing6002	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.090
funsing6002	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.110
funsing6002	10^{-8}	eTol	$\frac{1}{10}$	10	0.110
funsing6002	10^{-8}	eTol	$\frac{1}{4}$	10	0.100
funsing6002	10^{-8}	eTol	$\frac{1}{3}$	10	0.090
funsing6002	10^{-8}	eTol	$\frac{1}{2}$	10	0.110

Table 3.49: Results for the combined test for Risk-Premium and Mesh-Distribution. Replaceable results are achieved for this BVP.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing6002	10^{-6}	All	$\frac{1}{10}$	10	0.070
funsing6002	10^{-6}	All	$\frac{1}{4}$	10	0.090
funsing6002	10^{-6}	All	$\frac{1}{3}$	10	0.070
funsing6002	10^{-6}	All	$\frac{1}{2}$	10	0.090
funsing6002	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.070
funsing6002	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.080
funsing6002	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.080
funsing6002	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.091
funsing6002	10^{-6}	eTol	$\frac{1}{10}$	10	0.080
funsing6002	10^{-6}	eTol	$\frac{1}{4}$	10	0.070
funsing6002	10^{-6}	eTol	$\frac{1}{3}$	10	0.070
funsing6002	10^{-6}	eTol	$\frac{1}{2}$	10	0.100

Table 3.50: Results for the combined test for Risk-Premium and Mesh-Distribution. Here, the results are quite similar.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing6002	10^{-3}	All	$\frac{1}{10}$	5	0.050
funsing6002	10^{-3}	All	$\frac{1}{4}$	5	0.050
funsing6002	10^{-3}	All	$\frac{1}{3}$	5	0.060
funsing6002	10^{-3}	All	$\frac{1}{2}$	5	0.070
funsing6002	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.050
funsing6002	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.070
funsing6002	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.050
funsing6002	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.070
funsing6002	10^{-3}	eTol	$\frac{1}{10}$	5	0.060
funsing6002	10^{-3}	eTol	$\frac{1}{4}$	5	0.080
funsing6002	10^{-3}	eTol	$\frac{1}{3}$	5	0.060
funsing6002	10^{-3}	eTol	$\frac{1}{2}$	5	0.050

Table 3.51: Results for the combined test for Risk-Premium and Mesh-Distribution. Once more very similar results are achieved for this Bvp.

Bvp 6002 is easy to solve and no configuration has remarkable advantages or disadvantages.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing7001b	10^{-8}	All	$\frac{1}{10}$	10	0.100
funsing7001b	10^{-8}	All	$\frac{1}{4}$	10	0.100
funsing7001b	10^{-8}	All	$\frac{1}{3}$	10	0.110
funsing7001b	10^{-8}	All	$\frac{1}{2}$	10	0.110
funsing7001b	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.111
funsing7001b	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.100
funsing7001b	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.110
funsing7001b	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.110
funsing7001b	10^{-8}	eTol	$\frac{1}{10}$	10	0.130
funsing7001b	10^{-8}	eTol	$\frac{1}{4}$	10	0.121
funsing7001b	10^{-8}	eTol	$\frac{1}{3}$	10	0.120
funsing7001b	10^{-8}	eTol	$\frac{1}{2}$	10	0.120

Table 3.52: Results for the combined test for Risk-Premium and Mesh-Distribution. A slight disadvantage for eTol can be observed here.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing7001b	10^{-6}	All	$\frac{1}{10}$	10	0.080
funsing7001b	10^{-6}	All	$\frac{1}{4}$	10	0.100
funsing7001b	10^{-6}	All	$\frac{1}{3}$	10	0.090
funsing7001b	10^{-6}	All	$\frac{1}{2}$	10	0.090
funsing7001b	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.100
funsing7001b	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.080
funsing7001b	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.080
funsing7001b	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.080
funsing7001b	10^{-6}	eTol	$\frac{1}{10}$	10	0.111
funsing7001b	10^{-6}	eTol	$\frac{1}{4}$	10	0.110
funsing7001b	10^{-6}	eTol	$\frac{1}{3}$	10	0.100
funsing7001b	10^{-6}	eTol	$\frac{1}{2}$	10	0.101

Table 3.53: Results for the combined test for Risk-Premium and Mesh-Distribution. Again eTol is a bit slower.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing7001b	10^{-3}	All	$\frac{1}{10}$	5	0.060
funsing7001b	10^{-3}	All	$\frac{1}{4}$	5	0.070
funsing7001b	10^{-3}	All	$\frac{1}{3}$	5	0.081
funsing7001b	10^{-3}	All	$\frac{1}{2}$	5	0.060
funsing7001b	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.080
funsing7001b	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.070
funsing7001b	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.070
funsing7001b	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.071
funsing7001b	10^{-3}	eTol	$\frac{1}{10}$	5	0.070
funsing7001b	10^{-3}	eTol	$\frac{1}{4}$	5	0.080
funsing7001b	10^{-3}	eTol	$\frac{1}{3}$	5	0.080
funsing7001b	10^{-3}	eTol	$\frac{1}{2}$	5	0.080

Table 3.54: Results for the combined test for Risk-Premium and Mesh-Distribution. Every configuration solves the problem very fast.

Again the runtimes are very low, but at the strict tolerance eTol has a little disadvantage.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing71	10^{-8}	All	$\frac{1}{10}$	45	0.451
funsing71	10^{-8}	All	$\frac{1}{4}$	53	0.581
funsing71	10^{-8}	All	$\frac{1}{3}$	53	0.621
funsing71	10^{-8}	All	$\frac{1}{2}$	53	0.511
funsing71	10^{-8}	TolOpt	$\frac{1}{10}$	42	0.440
funsing71	10^{-8}	TolOpt	$\frac{1}{4}$	53	0.500
funsing71	10^{-8}	TolOpt	$\frac{1}{3}$	53	0.530
funsing71	10^{-8}	TolOpt	$\frac{1}{2}$	53	0.501
funsing71	10^{-8}	eTol	$\frac{1}{10}$	39	0.411
funsing71	10^{-8}	eTol	$\frac{1}{4}$	39	0.431
funsing71	10^{-8}	eTol	$\frac{1}{3}$	53	0.601
funsing71	10^{-8}	eTol	$\frac{1}{2}$	53	0.671

Table 3.55: Results for the combined test for Risk-Premium and Mesh-Distribution. $\frac{1}{10}$ is the best choice here, whereas eTol is favorable when we look at the number of meshpoints for this setting of Risk-Premium.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing71	10^{-6}	All	$\frac{1}{10}$	28	0.200
funsing71	10^{-6}	All	$\frac{1}{4}$	51	0.381
funsing71	10^{-6}	All	$\frac{1}{3}$	51	0.420
funsing71	10^{-6}	All	$\frac{1}{2}$	51	0.361
funsing71	10^{-6}	TolOpt	$\frac{1}{10}$	39	0.301
funsing71	10^{-6}	TolOpt	$\frac{1}{4}$	51	0.361
funsing71	10^{-6}	TolOpt	$\frac{1}{3}$	51	0.370
funsing71	10^{-6}	TolOpt	$\frac{1}{2}$	51	0.370
funsing71	10^{-6}	eTol	$\frac{1}{10}$	32	0.290
funsing71	10^{-6}	eTol	$\frac{1}{4}$	32	0.271
funsing71	10^{-6}	eTol	$\frac{1}{3}$	32	0.310
funsing71	10^{-6}	eTol	$\frac{1}{2}$	51	0.460

Table 3.56: Results for the combined test for Risk-Premium and Mesh-Distribution. All with $\frac{1}{10}$ is the quickest and needs the least number of meshpoints.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing71	10^{-3}	All	$\frac{1}{10}$	14	0.120
funsing71	10^{-3}	All	$\frac{1}{4}$	24	0.181
funsing71	10^{-3}	All	$\frac{1}{3}$	24	0.200
funsing71	10^{-3}	All	$\frac{1}{2}$	24	0.191
funsing71	10^{-3}	TolOpt	$\frac{1}{10}$	18	0.190
funsing71	10^{-3}	TolOpt	$\frac{1}{4}$	24	0.180
funsing71	10^{-3}	TolOpt	$\frac{1}{3}$	24	0.200
funsing71	10^{-3}	TolOpt	$\frac{1}{2}$	24	0.200
funsing71	10^{-3}	eTol	$\frac{1}{10}$	17	0.190
funsing71	10^{-3}	eTol	$\frac{1}{4}$	17	0.180
funsing71	10^{-3}	eTol	$\frac{1}{3}$	24	0.210
funsing71	10^{-3}	eTol	$\frac{1}{2}$	24	0.201

Table 3.57: Results for the combined test for Risk-Premium and Mesh-Distribution. All with $\frac{1}{10}$ shows a significant advantage.

BVP 71 shows to be most favorable for eTol at the strict tolerance, while All is most efficient at the lower tolerances. While the difference for the tolerance at 10^{-8} is not very big, the difference for the other settings of the tolerance is significant. Figures 3.12 - 3.14 show the grid history for this BVP at Risk-Premium $\frac{1}{10}$, which yields excellent results here.

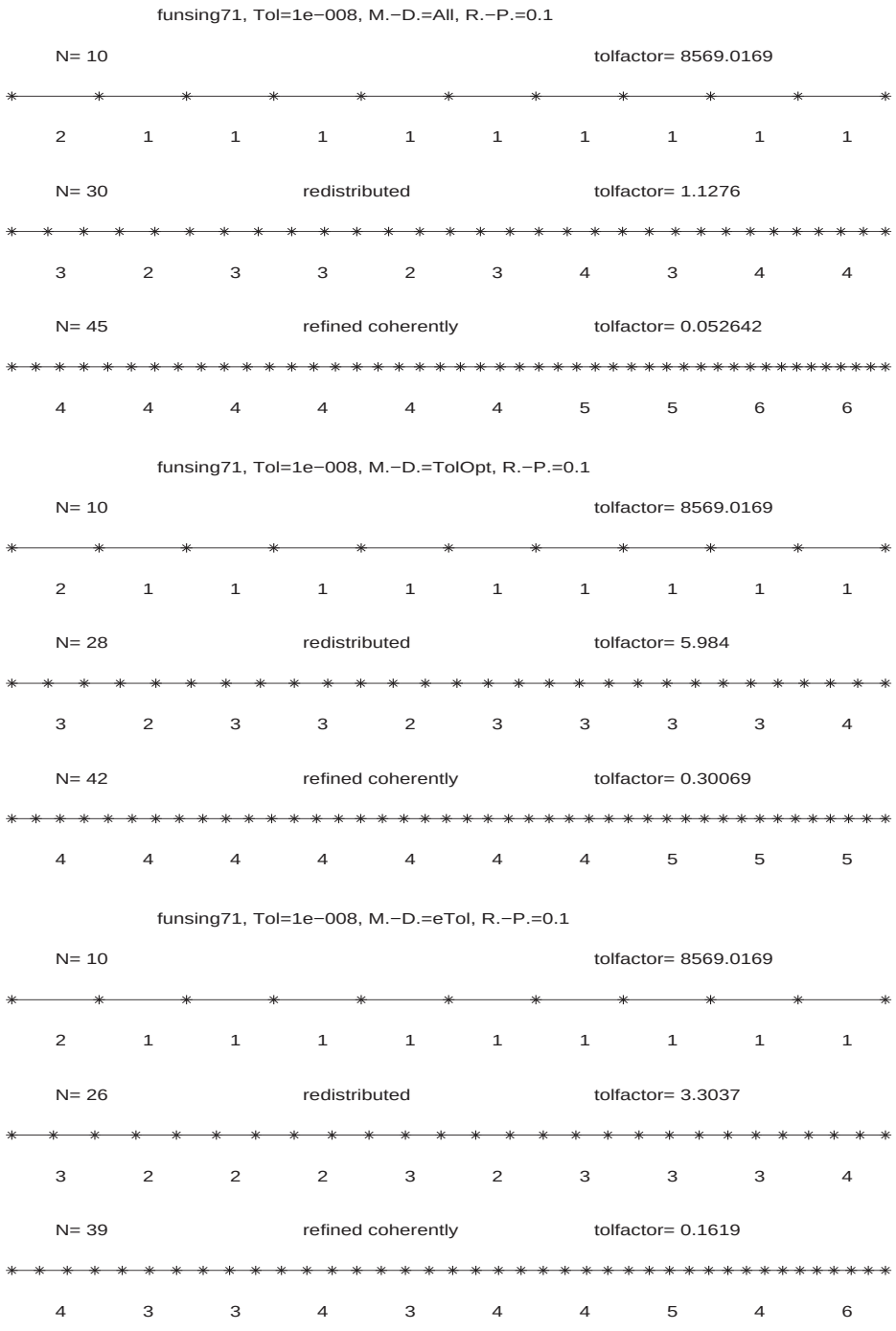


Figure 3.12: This figure shows the grid evolution of Bvp 71 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$.

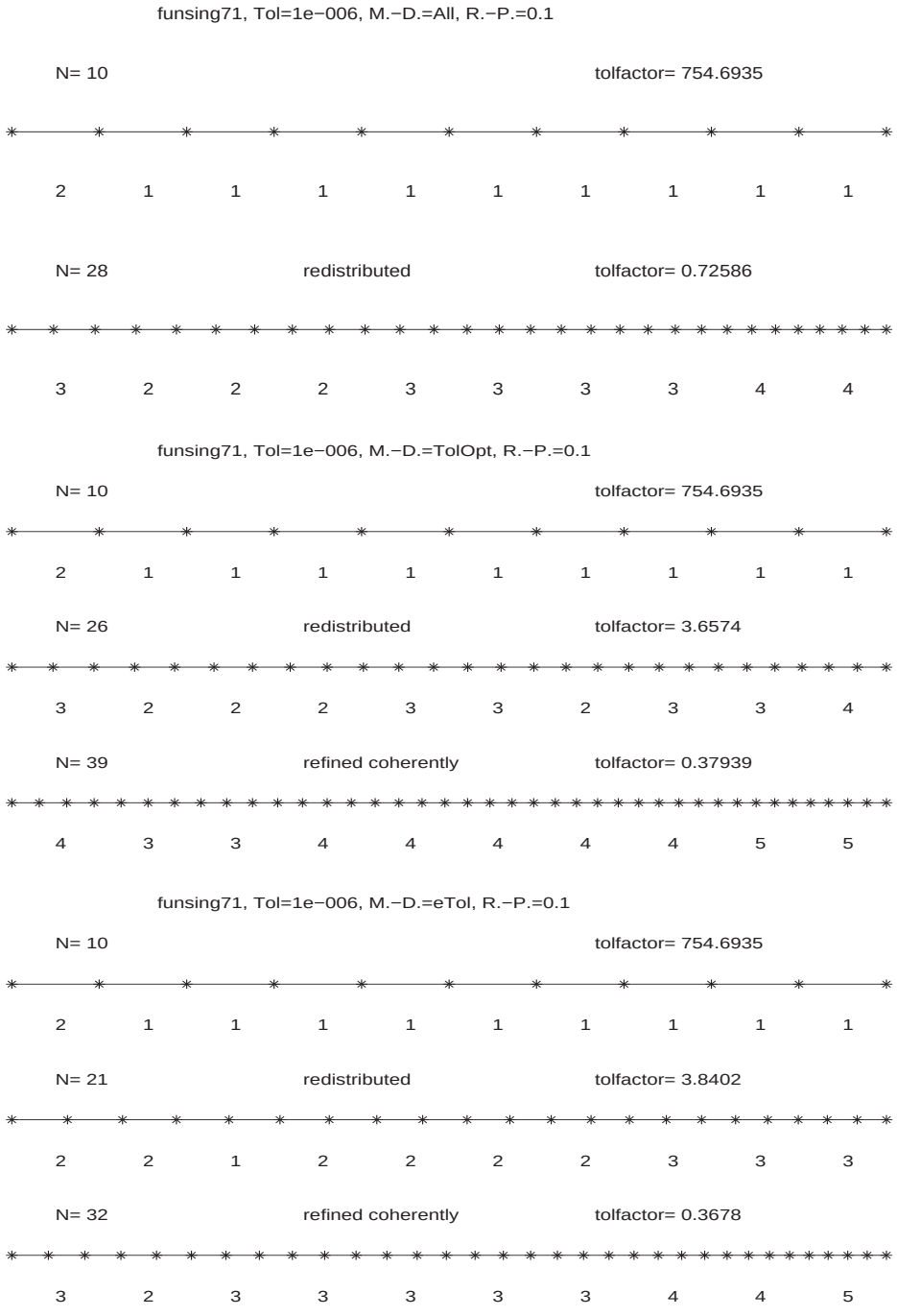


Figure 3.13: This figure shows the grid evolution of BVP 71 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$.

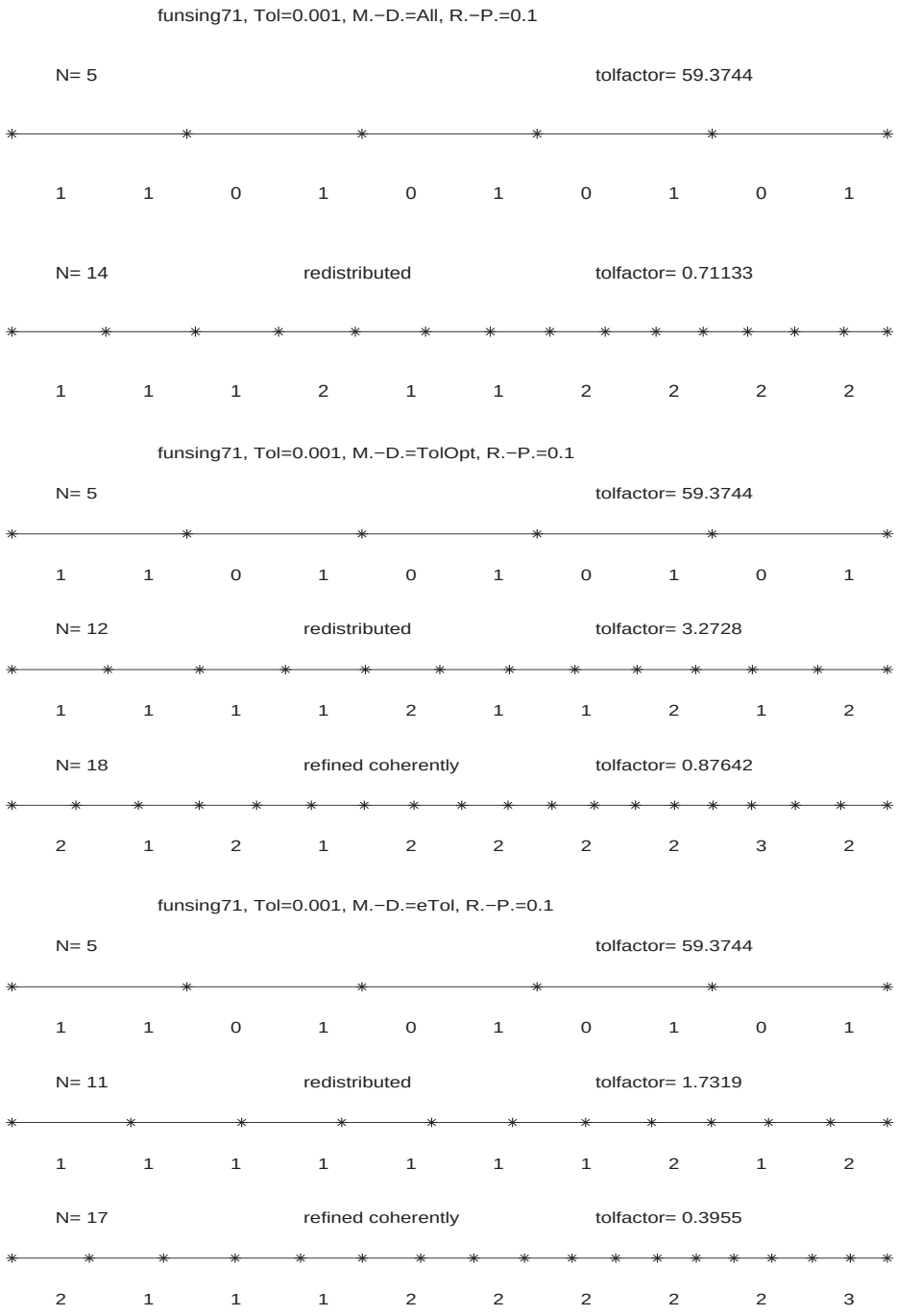


Figure 3.14: This figure shows the grid evolution of Bvp 71 for the tolerance set to 10^{-8} and Risk-Premium at $\frac{1}{10}$.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing73	10^{-8}	All	$\frac{1}{10}$	120	6.190
funsing73	10^{-8}	All	$\frac{1}{4}$	120	6.210
funsing73	10^{-8}	All	$\frac{1}{3}$	120	6.220
funsing73	10^{-8}	All	$\frac{1}{2}$	120	6.098
funsing73	10^{-8}	TolOpt	$\frac{1}{10}$	120	6.219
funsing73	10^{-8}	TolOpt	$\frac{1}{4}$	120	6.209
funsing73	10^{-8}	TolOpt	$\frac{1}{3}$	120	6.239
funsing73	10^{-8}	TolOpt	$\frac{1}{2}$	120	6.219
funsing73	10^{-8}	eTol	$\frac{1}{10}$	120	6.220
funsing73	10^{-8}	eTol	$\frac{1}{4}$	120	6.241
funsing73	10^{-8}	eTol	$\frac{1}{3}$	120	6.250
funsing73	10^{-8}	eTol	$\frac{1}{2}$	120	6.211

Table 3.58: Results for the combined test for Risk-Premium and Mesh-Distribution. The results are quite balanced.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing73	10^{-6}	All	$\frac{1}{10}$	240	8.802
funsing73	10^{-6}	All	$\frac{1}{4}$	240	8.812
funsing73	10^{-6}	All	$\frac{1}{3}$	495	24.365
funsing73	10^{-6}	All	$\frac{1}{2}$	495	24.345
funsing73	10^{-6}	TolOpt	$\frac{1}{10}$	240	8.863
funsing73	10^{-6}	TolOpt	$\frac{1}{4}$	240	8.813
funsing73	10^{-6}	TolOpt	$\frac{1}{3}$	495	24.365
funsing73	10^{-6}	TolOpt	$\frac{1}{2}$	495	24.355
funsing73	10^{-6}	eTol	$\frac{1}{10}$	240	8.803
funsing73	10^{-6}	eTol	$\frac{1}{4}$	240	8.790
funsing73	10^{-6}	eTol	$\frac{1}{3}$	495	24.345
funsing73	10^{-6}	eTol	$\frac{1}{2}$	495	24.365

Table 3.59: Results for the combined test for Risk-Premium and Mesh-Distribution. For this regular BVP the lower values for Risk-Premium are favorable.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing73	10^{-3}	All	$\frac{1}{10}$	240	4.736
funsing73	10^{-3}	All	$\frac{1}{4}$	240	4.737
funsing73	10^{-3}	All	$\frac{1}{3}$	240	4.757
funsing73	10^{-3}	All	$\frac{1}{2}$	240	4.726
funsing73	10^{-3}	TolOpt	$\frac{1}{10}$	240	4.799
funsing73	10^{-3}	TolOpt	$\frac{1}{4}$	240	4.767
funsing73	10^{-3}	TolOpt	$\frac{1}{3}$	240	4.767
funsing73	10^{-3}	TolOpt	$\frac{1}{2}$	240	4.727
funsing73	10^{-3}	eTol	$\frac{1}{10}$	240	4.636
funsing73	10^{-3}	eTol	$\frac{1}{4}$	240	4.727
funsing73	10^{-3}	eTol	$\frac{1}{3}$	240	4.757
funsing73	10^{-3}	eTol	$\frac{1}{2}$	240	4.747

Table 3.60: Results for the combined test for Risk-Premium and Mesh-Distribution. There is no difference in the results here.

Bvp 73 favors the low Risk-Premiums at tolerance 10^{-6}. The runtime for the high Risk-Premiums is three times the runtime of the lower ones.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing8001	10^{-8}	All	$\frac{1}{10}$	10	0.110
funsing8001	10^{-8}	All	$\frac{1}{4}$	10	0.110
funsing8001	10^{-8}	All	$\frac{1}{3}$	10	0.130
funsing8001	10^{-8}	All	$\frac{1}{2}$	10	0.120
funsing8001	10^{-8}	TolOpt	$\frac{1}{10}$	10	0.130
funsing8001	10^{-8}	TolOpt	$\frac{1}{4}$	10	0.130
funsing8001	10^{-8}	TolOpt	$\frac{1}{3}$	10	0.130
funsing8001	10^{-8}	TolOpt	$\frac{1}{2}$	10	0.120
funsing8001	10^{-8}	eTol	$\frac{1}{10}$	10	0.150
funsing8001	10^{-8}	eTol	$\frac{1}{4}$	10	0.151
funsing8001	10^{-8}	eTol	$\frac{1}{3}$	10	0.161
funsing8001	10^{-8}	eTol	$\frac{1}{2}$	10	0.141

Table 3.61: Results for the combined test for Risk-Premium and Mesh-Distribution. All is quickest for the low Risk-Premiums and eTol is slowest.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing8001	10^{-6}	All	$\frac{1}{10}$	10	0.090
funsing8001	10^{-6}	All	$\frac{1}{4}$	10	0.090
funsing8001	10^{-6}	All	$\frac{1}{3}$	10	0.100
funsing8001	10^{-6}	All	$\frac{1}{2}$	10	0.110
funsing8001	10^{-6}	TolOpt	$\frac{1}{10}$	10	0.100
funsing8001	10^{-6}	TolOpt	$\frac{1}{4}$	10	0.111
funsing8001	10^{-6}	TolOpt	$\frac{1}{3}$	10	0.101
funsing8001	10^{-6}	TolOpt	$\frac{1}{2}$	10	0.110
funsing8001	10^{-6}	eTol	$\frac{1}{10}$	10	0.120
funsing8001	10^{-6}	eTol	$\frac{1}{4}$	10	0.130
funsing8001	10^{-6}	eTol	$\frac{1}{3}$	10	0.130
funsing8001	10^{-6}	eTol	$\frac{1}{2}$	10	0.120

Table 3.62: Results for the combined test for Risk-Premium and Mesh-Distribution. Again All is a bit quicker at $\frac{1}{10}$ and $\frac{1}{4}$.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing8001	10^{-3}	All	$\frac{1}{10}$	5	0.090
funsing8001	10^{-3}	All	$\frac{1}{4}$	5	0.091
funsing8001	10^{-3}	All	$\frac{1}{3}$	5	0.080
funsing8001	10^{-3}	All	$\frac{1}{2}$	5	0.090
funsing8001	10^{-3}	TolOpt	$\frac{1}{10}$	5	0.080
funsing8001	10^{-3}	TolOpt	$\frac{1}{4}$	5	0.090
funsing8001	10^{-3}	TolOpt	$\frac{1}{3}$	5	0.060
funsing8001	10^{-3}	TolOpt	$\frac{1}{2}$	5	0.080
funsing8001	10^{-3}	eTol	$\frac{1}{10}$	5	0.090
funsing8001	10^{-3}	eTol	$\frac{1}{4}$	5	0.080
funsing8001	10^{-3}	eTol	$\frac{1}{3}$	5	0.080
funsing8001	10^{-3}	eTol	$\frac{1}{2}$	5	0.080

Table 3.63: Results for the combined test for Risk-Premium and Mesh-Distribution. No difference can be seen for this tolerance.

BVP 8001 shows balanced results with a little disadvantage for eTol at the stricter tolerances.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing8002	10^{-8}	All	$\frac{1}{10}$	15	0.230
funsing8002	10^{-8}	All	$\frac{1}{4}$	15	0.240
funsing8002	10^{-8}	All	$\frac{1}{3}$	15	0.230
funsing8002	10^{-8}	All	$\frac{1}{2}$	27	0.270
funsing8002	10^{-8}	TolOpt	$\frac{1}{10}$	15	0.210
funsing8002	10^{-8}	TolOpt	$\frac{1}{4}$	15	0.221
funsing8002	10^{-8}	TolOpt	$\frac{1}{3}$	15	0.240
funsing8002	10^{-8}	TolOpt	$\frac{1}{2}$	27	0.290
funsing8002	10^{-8}	eTol	$\frac{1}{10}$	15	0.221
funsing8002	10^{-8}	eTol	$\frac{1}{4}$	15	0.230
funsing8002	10^{-8}	eTol	$\frac{1}{3}$	15	0.251
funsing8002	10^{-8}	eTol	$\frac{1}{2}$	27	0.300

Table 3.64: Results for the combined test for Risk-Premium and Mesh-Distribution. Here the results of all choices of Mesh-Distribution are quite stable, especially for the lower Risk-Premiums.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing8002	10^{-6}	All	$\frac{1}{10}$	15	0.190
funsing8002	10^{-6}	All	$\frac{1}{4}$	15	0.190
funsing8002	10^{-6}	All	$\frac{1}{3}$	15	0.200
funsing8002	10^{-6}	All	$\frac{1}{2}$	25	0.210
funsing8002	10^{-6}	TolOpt	$\frac{1}{10}$	15	0.181
funsing8002	10^{-6}	TolOpt	$\frac{1}{4}$	15	0.180
funsing8002	10^{-6}	TolOpt	$\frac{1}{3}$	15	0.181
funsing8002	10^{-6}	TolOpt	$\frac{1}{2}$	25	0.210
funsing8002	10^{-6}	eTol	$\frac{1}{10}$	15	0.190
funsing8002	10^{-6}	eTol	$\frac{1}{4}$	15	0.181
funsing8002	10^{-6}	eTol	$\frac{1}{3}$	15	0.200
funsing8002	10^{-6}	eTol	$\frac{1}{2}$	25	0.210

Table 3.65: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt is most favorable for the lower Risk-Premiums, but the difference is very small.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing8002	10^{-3}	All	$\frac{1}{10}$	8	0.150
funsing8002	10^{-3}	All	$\frac{1}{4}$	8	0.140
funsing8002	10^{-3}	All	$\frac{1}{3}$	96	0.701
funsing8002	10^{-3}	All	$\frac{1}{2}$	96	0.781
funsing8002	10^{-3}	TolOpt	$\frac{1}{10}$	8	0.131
funsing8002	10^{-3}	TolOpt	$\frac{1}{4}$	8	0.140
funsing8002	10^{-3}	TolOpt	$\frac{1}{3}$	96	0.660
funsing8002	10^{-3}	TolOpt	$\frac{1}{2}$	96	0.671
funsing8002	10^{-3}	eTol	$\frac{1}{10}$	8	0.150
funsing8002	10^{-3}	eTol	$\frac{1}{4}$	8	0.140
funsing8002	10^{-3}	eTol	$\frac{1}{3}$	96	0.751
funsing8002	10^{-3}	eTol	$\frac{1}{2}$	96	0.701

Table 3.66: Results for the combined test for Risk-Premium and Mesh-Distribution. Catastrophic results are achieved for the higher settings of Risk-Premium.

The results for the different Mesh-Distributions are quite balanced. But as seen before the lower Risk-Premiums are favorable.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9001	10^{-8}	All	$\frac{1}{10}$	45	0.140
funsing9001	10^{-8}	All	$\frac{1}{4}$	45	0.141
funsing9001	10^{-8}	All	$\frac{1}{3}$	45	0.150
funsing9001	10^{-8}	All	$\frac{1}{2}$	45	0.141
funsing9001	10^{-8}	TolOpt	$\frac{1}{10}$	41	0.161
funsing9001	10^{-8}	TolOpt	$\frac{1}{4}$	41	0.170
funsing9001	10^{-8}	TolOpt	$\frac{1}{3}$	41	0.170
funsing9001	10^{-8}	TolOpt	$\frac{1}{2}$	45	0.141
funsing9001	10^{-8}	eTol	$\frac{1}{10}$	52	0.220
funsing9001	10^{-8}	eTol	$\frac{1}{4}$	52	0.200
funsing9001	10^{-8}	eTol	$\frac{1}{3}$	52	0.200
funsing9001	10^{-8}	eTol	$\frac{1}{2}$	45	0.140

Table 3.67: Results for the combined test for Risk-Premium and Mesh-Distribution. eTol is slower and All is quickest, especially for $\frac{1}{10}$ and $\frac{1}{4}$.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9001	10^{-6}	All	$\frac{1}{10}$	81	0.240
funsing9001	10^{-6}	All	$\frac{1}{4}$	81	0.241
funsing9001	10^{-6}	All	$\frac{1}{3}$	81	0.240
funsing9001	10^{-6}	All	$\frac{1}{2}$	81	0.221
funsing9001	10^{-6}	TolOpt	$\frac{1}{10}$	84	0.280
funsing9001	10^{-6}	TolOpt	$\frac{1}{4}$	84	0.261
funsing9001	10^{-6}	TolOpt	$\frac{1}{3}$	84	0.271
funsing9001	10^{-6}	TolOpt	$\frac{1}{2}$	81	0.241
funsing9001	10^{-6}	eTol	$\frac{1}{10}$	91	0.210
funsing9001	10^{-6}	eTol	$\frac{1}{4}$	91	0.210
funsing9001	10^{-6}	eTol	$\frac{1}{3}$	91	0.221
funsing9001	10^{-6}	eTol	$\frac{1}{2}$	81	0.230

Table 3.68: Results for the combined test for Risk-Premium and Mesh-Distribution. Surprisingly eTol has the best runtimes with the highest number of meshpoints. All is quicker than TolOpt.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9001	10^{-3}	All	$\frac{1}{10}$	107	0.260
funsing9001	10^{-3}	All	$\frac{1}{4}$	107	0.260
funsing9001	10^{-3}	All	$\frac{1}{3}$	107	0.281
funsing9001	10^{-3}	All	$\frac{1}{2}$	107	0.240
funsing9001	10^{-3}	TolOpt	$\frac{1}{10}$	54	0.160
funsing9001	10^{-3}	TolOpt	$\frac{1}{4}$	66	0.160
funsing9001	10^{-3}	TolOpt	$\frac{1}{3}$	66	0.170
funsing9001	10^{-3}	TolOpt	$\frac{1}{2}$	107	0.250
funsing9001	10^{-3}	eTol	$\frac{1}{10}$	72	0.150
funsing9001	10^{-3}	eTol	$\frac{1}{4}$	72	0.160
funsing9001	10^{-3}	eTol	$\frac{1}{3}$	72	0.150
funsing9001	10^{-3}	eTol	$\frac{1}{2}$	107	0.250

Table 3.69: Results for the combined test for Risk-Premium and Mesh-Distribution. $\frac{1}{2}$ is slowest. eTol performs best and All performs worst.

A very interesting behavior is observed for this singularly perturbed Bvp. All has the quickest runtimes for the stricter tolerance. It seems that the higher collocation order is an advantage for Bvp 9001.

While All is best in the difficult case, and runtime and number of meshpoints is better than for any other tolerance, eTol and TolOpt are performing better for the lower tolerances.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9002	10^{-8}	All	$\frac{1}{10}$	69	0.271
funsing9002	10^{-8}	All	$\frac{1}{4}$	69	0.280
funsing9002	10^{-8}	All	$\frac{1}{3}$	69	0.320
funsing9002	10^{-8}	All	$\frac{1}{2}$	69	0.251
funsing9002	10^{-8}	TolOpt	$\frac{1}{10}$	63	0.251
funsing9002	10^{-8}	TolOpt	$\frac{1}{4}$	69	0.270
funsing9002	10^{-8}	TolOpt	$\frac{1}{3}$	69	0.271
funsing9002	10^{-8}	TolOpt	$\frac{1}{2}$	69	0.270
funsing9002	10^{-8}	eTol	$\frac{1}{10}$	69	0.290
funsing9002	10^{-8}	eTol	$\frac{1}{4}$	69	0.280
funsing9002	10^{-8}	eTol	$\frac{1}{3}$	69	0.290
funsing9002	10^{-8}	eTol	$\frac{1}{2}$	69	0.280

Table 3.70: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt only needs 63 meshpoints for $\frac{1}{10}$.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9002	10^{-6}	All	$\frac{1}{10}$	92	0.261
funsing9002	10^{-6}	All	$\frac{1}{4}$	92	0.261
funsing9002	10^{-6}	All	$\frac{1}{3}$	92	0.270
funsing9002	10^{-6}	All	$\frac{1}{2}$	92	0.230
funsing9002	10^{-6}	TolOpt	$\frac{1}{10}$	93	0.210
funsing9002	10^{-6}	TolOpt	$\frac{1}{4}$	92	0.231
funsing9002	10^{-6}	TolOpt	$\frac{1}{3}$	92	0.240
funsing9002	10^{-6}	TolOpt	$\frac{1}{2}$	92	0.240
funsing9002	10^{-6}	eTol	$\frac{1}{10}$	139	0.310
funsing9002	10^{-6}	eTol	$\frac{1}{4}$	92	0.261
funsing9002	10^{-6}	eTol	$\frac{1}{3}$	92	0.270
funsing9002	10^{-6}	eTol	$\frac{1}{2}$	92	0.280

Table 3.71: Results for the combined test for Risk-Premium and Mesh-Distribution. TolOpt is most favorable for this BVP but All has the most stable behavior with respect to the number of meshpoints.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9002	10^{-3}	All	$\frac{1}{10}$	147	0.310
funsing9002	10^{-3}	All	$\frac{1}{4}$	147	0.311
funsing9002	10^{-3}	All	$\frac{1}{3}$	147	0.290
funsing9002	10^{-3}	All	$\frac{1}{2}$	147	0.291
funsing9002	10^{-3}	TolOpt	$\frac{1}{10}$	135	0.251
funsing9002	10^{-3}	TolOpt	$\frac{1}{4}$	135	0.250
funsing9002	10^{-3}	TolOpt	$\frac{1}{3}$	147	0.281
funsing9002	10^{-3}	TolOpt	$\frac{1}{2}$	147	0.281
funsing9002	10^{-3}	eTol	$\frac{1}{10}$	147	0.280
funsing9002	10^{-3}	eTol	$\frac{1}{4}$	147	0.301
funsing9002	10^{-3}	eTol	$\frac{1}{3}$	147	0.301
funsing9002	10^{-3}	eTol	$\frac{1}{2}$	147	0.320

Table 3.72: Results for the combined test for Risk-Premium and Mesh-Distribution. The best results are achieved by TolOpt and the lower Risk-Premiums.

At the preferred Risk-Premium All is better than eTol for the higher tolerances. The best results are achieved by TolOpt.

Bvp	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9003	10^{-8}	All	$\frac{1}{10}$	16	0.080
funsing9003	10^{-8}	All	$\frac{1}{4}$	16	0.090
funsing9003	10^{-8}	All	$\frac{1}{3}$	16	0.081
funsing9003	10^{-8}	All	$\frac{1}{2}$	16	0.080
funsing9003	10^{-8}	TolOpt	$\frac{1}{10}$	16	0.080
funsing9003	10^{-8}	TolOpt	$\frac{1}{4}$	16	0.090
funsing9003	10^{-8}	TolOpt	$\frac{1}{3}$	16	0.090
funsing9003	10^{-8}	TolOpt	$\frac{1}{2}$	16	0.090
funsing9003	10^{-8}	eTol	$\frac{1}{10}$	16	0.090
funsing9003	10^{-8}	eTol	$\frac{1}{4}$	16	0.101
funsing9003	10^{-8}	eTol	$\frac{1}{3}$	16	0.101
funsing9003	10^{-8}	eTol	$\frac{1}{2}$	16	0.081

Table 3.73: Results for the combined test for Risk-Premium and Mesh-Distribution. Similar results for each configuration.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9003	10^{-6}	All	$\frac{1}{10}$	27	0.110
funsing9003	10^{-6}	All	$\frac{1}{4}$	27	0.120
funsing9003	10^{-6}	All	$\frac{1}{3}$	27	0.131
funsing9003	10^{-6}	All	$\frac{1}{2}$	27	0.110
funsing9003	10^{-6}	TolOpt	$\frac{1}{10}$	27	0.110
funsing9003	10^{-6}	TolOpt	$\frac{1}{4}$	27	0.120
funsing9003	10^{-6}	TolOpt	$\frac{1}{3}$	27	0.120
funsing9003	10^{-6}	TolOpt	$\frac{1}{2}$	27	0.120
funsing9003	10^{-6}	eTol	$\frac{1}{10}$	27	0.120
funsing9003	10^{-6}	eTol	$\frac{1}{4}$	27	0.130
funsing9003	10^{-6}	eTol	$\frac{1}{3}$	27	0.130
funsing9003	10^{-6}	eTol	$\frac{1}{2}$	27	0.110

Table 3.74: Results for the combined test for Risk-Premium and Mesh-Distribution. Similar results for each configuration.

BVP	Tol	Mesh.-D.	Risk-Prem.	N	Runtime
funsing9003	10^{-3}	All	$\frac{1}{10}$	10	0.060
funsing9003	10^{-3}	All	$\frac{1}{4}$	10	0.070
funsing9003	10^{-3}	All	$\frac{1}{3}$	10	0.060
funsing9003	10^{-3}	All	$\frac{1}{2}$	10	0.050
funsing9003	10^{-3}	TolOpt	$\frac{1}{10}$	10	0.060
funsing9003	10^{-3}	TolOpt	$\frac{1}{4}$	10	0.070
funsing9003	10^{-3}	TolOpt	$\frac{1}{3}$	10	0.071
funsing9003	10^{-3}	TolOpt	$\frac{1}{2}$	10	0.061
funsing9003	10^{-3}	eTol	$\frac{1}{10}$	10	0.070
funsing9003	10^{-3}	eTol	$\frac{1}{4}$	10	0.070
funsing9003	10^{-3}	eTol	$\frac{1}{3}$	10	0.050
funsing9003	10^{-3}	eTol	$\frac{1}{2}$	10	0.060

Table 3.75: Results for the combined test for Risk-Premium and Mesh-Distribution. Similar results for each configuration.

This BVP is solved rather quickly and with no problems.

3.1.2 Conclusion

The results do not give a clear indication of the best setting for Mesh-Distribution and Risk-Premium.

First of all Risk-Premium is chosen rather easily. Due to the test of Bvp 54 (refer to Table 3.37) at the tolerance set to 10^{-8} it is obvious that $\frac{1}{10}$ is the most promising choice. As this setting does not show any major disadvantage throughout the tests and there were many excellent results (e. g. refer to Tables 3.46 or 3.55), we chose this value for Risk-Premium.

The situation for Mesh-Distribution is quite different. On the one hand there is TolOpt with very good results for many of the Bvps, but on the other hand TolOpt shows some weaknesses and it has mixed results on the different settings of Risk-Premium. When Mesh-Distribution is set to eTol the favorable result for Bvp 54 at the strict tolerance is striking and cannot be ignored. Otherwise eTol often is quite slow and for Bvp 71 it shows very poor results that put the good ones for Bvp 54 into perspective. Moreover, All shows stable results that were best for the chosen Risk-Premium at the high tolerance.

So the decision remains between eTol and All. Finally because of the balanced results of All, except for Bvp 54 at the tolerance at 10^{-8}, this setting for Mesh-Distribution is chosen.

The following Tables 3.76 - 3.77 affirm these decisions.

After Bvp 54 is neglected, Table 3.76 shows the sum of the absolute runtime over all Bvps in each configuration. Clearly summing up the runtime of all Bvps is controversial because many results showed very small differences which only were caused by random lags. Otherwise random differences should compensate for the number of Bvps used. If one configuration is remarkably slower, even many random lags in favor of another configuration cannot compensate.

Configuration	10^{-3}	10^{-6}	10^{-8}
All, $\frac{1}{10}$	9.233	14.097	12.697
All, $\frac{1}{4}$	9.518	14.363	12.960
All, $\frac{1}{3}$	10.214	30.092	13.200
All, $\frac{1}{2}$	10.527	30.245	12.808
TolOpt, $\frac{1}{10}$	9.648	14.312	13.480
TolOpt, $\frac{1}{4}$	9.210	14.275	13.067
TolOpt, $\frac{1}{3}$	9.793	29.764	13.130
TolOpt, $\frac{1}{2}$	10.488	29.752	13.009
eTol, $\frac{1}{10}$	9.162	14.189	12.791
eTol, $\frac{1}{4}$	9.413	14.199	12.949
eTol, $\frac{1}{3}$	10.104	29.914	13.170
eTol, $\frac{1}{2}$	10.658	30.473	13.190

Table 3.76: The sum of absolute runtimes for different settings of Mesh-Distribution and Risk-Premium. As a reason of the complicated results for Bvp 54 at tolerance 10^{-8} the Bvp was cleared for this table.

It is obvious that the higher Risk-Premiums have worse results. Table 3.77 shows the absolute runtimes from Table 3.76 summed up over the different tolerances.

	All	TolOpt	eTol
Risk-Premium $\frac{1}{10}$	36.027	37.440	36.142
Risk-Premium $\frac{1}{4}$	36.841	36.552	36.561

Table 3.77: The sum of absolute runtimes. The results for all three tolerances are summed up here.

If we assume that half a second cannot be countervailed by random lags in the solution process of 72 different BVPS, especially when we think that probability is in favor of compensation of random lags, only two different configurations remain: Mesh-Distribution All and eTol, both with Risk-Premium $\frac{1}{10}$. Since All is quicker for the harder tolerances this choice is more favorable.

Hence, our final values are Mesh-Distribution All and Risk-Premium $\frac{1}{10}$.

3.2 MonitorFunction

After Risk-Premium and Mesh-Distribution were fixed, the next test was concerned with the smoothing process for the MonitorFunction. Since the MonitorFunction's settings are 1 and 0, which means that the MonitorFunction is either smoothed or not, the number of test runs necessary for this investigation is smaller.

If $\psi = (\psi_1, \ldots, \psi_n)$ denotes the MonitorFunction and $\tilde{\psi}$ is the smoothed MonitorFunction, $\tilde{\psi}$ is calculated in the following way:

$$\tilde{\psi} := (\tilde{\psi}_1, \ldots, \tilde{\psi}_n) \quad with \quad \tilde{\psi}_j = \max(\bar{\psi}_j, \psi_j).$$

If $s = \max(2, \lfloor N * (p+1) * S_F \rfloor)$, where N is the number of meshpoints, p is the collocation order and S_F is the factor by which the MonitorFunction is going to be smoothed, $\bar{\psi}$ is calculated via

$$\bar{\psi}_j = \frac{\sum_{i=1}^{s+j} \psi_i}{s+j}, \qquad j = 1, \ldots, s+1,$$

$$\bar{\psi}_{n-j+1} = \frac{\sum_{i=n-(s+j)+1}^{n} \psi_i}{s+j}, \qquad j = 1, \ldots, s+1.$$

And finally

$$\bar{\psi}_j = \frac{\sum_{i=j-s}^{j+s} \psi_i}{2s+1}, \qquad j = s+2, \ldots, n-s-1.$$

3.2.1 Test Results

In this section the Tables 3.78 - 3.99 show the runtime and the final number of meshpoints. The Figures 3.15 - 3.35 show the final error estimates $yD - y$ for the corresponding BVPS.

Since this test is not very complex the conclusion is found rather quickly. First the effect of turning smoothing on or off is quite small, but if there is a noticeable difference the advantages are on the side of the smoothed version, cf. for example Table 3.78 at the tolerance set to 10^{-8}, Table 3.79 at the tolerance set to 10^{-3} or Table 3.86 for the tolerance 10^{-3}. A dramatically bad result can be seen in Table 3.90 for the mild tolerance 10^{-3}, though the un-smoothed version is quicker for the stricter tolerances there. Finally the regular BVP 71 shows advantages for the smoothed version too, cf. Table 3.94.

Clearly, it was easy to decide in favor of the smoothed MonitorFunction.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing0026	10^{-3}	0	5	0.080
funsing0026	10^{-3}	1	5	0.060
funsing0026	10^{-6}	0	10	0.100
funsing0026	10^{-6}	1	10	0.101
funsing0026	10^{-8}	0	10	0.140
funsing0026	10^{-8}	1	10	0.100

Table 3.78: Runtimes and number of meshpoints when testing the MonitorFunction. A slight advantage for the smoothed MonitorFunction can be observed.

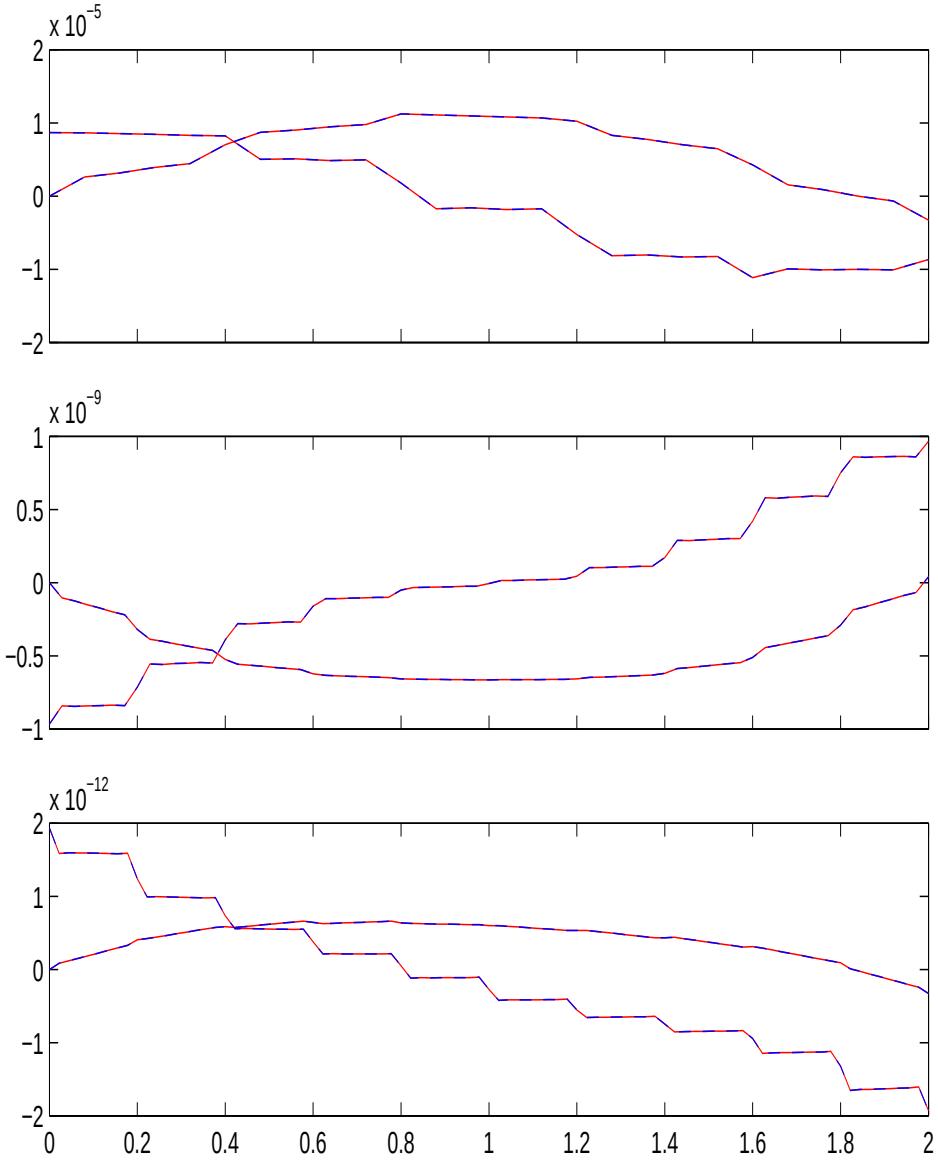


Figure 3.15: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 0026.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing1001	10^{-3}	0	40	0.160
funsing1001	10^{-3}	1	40	0.090
funsing1001	10^{-6}	0	60	0.200
funsing1001	10^{-6}	1	60	0.201
funsing1001	10^{-8}	0	58	0.200
funsing1001	10^{-8}	1	58	0.190

Table 3.79: Runtimes and number of meshpoints when testing the MonitorFunction. A slight advantage for the smoothed MonitorFunction can be observed.

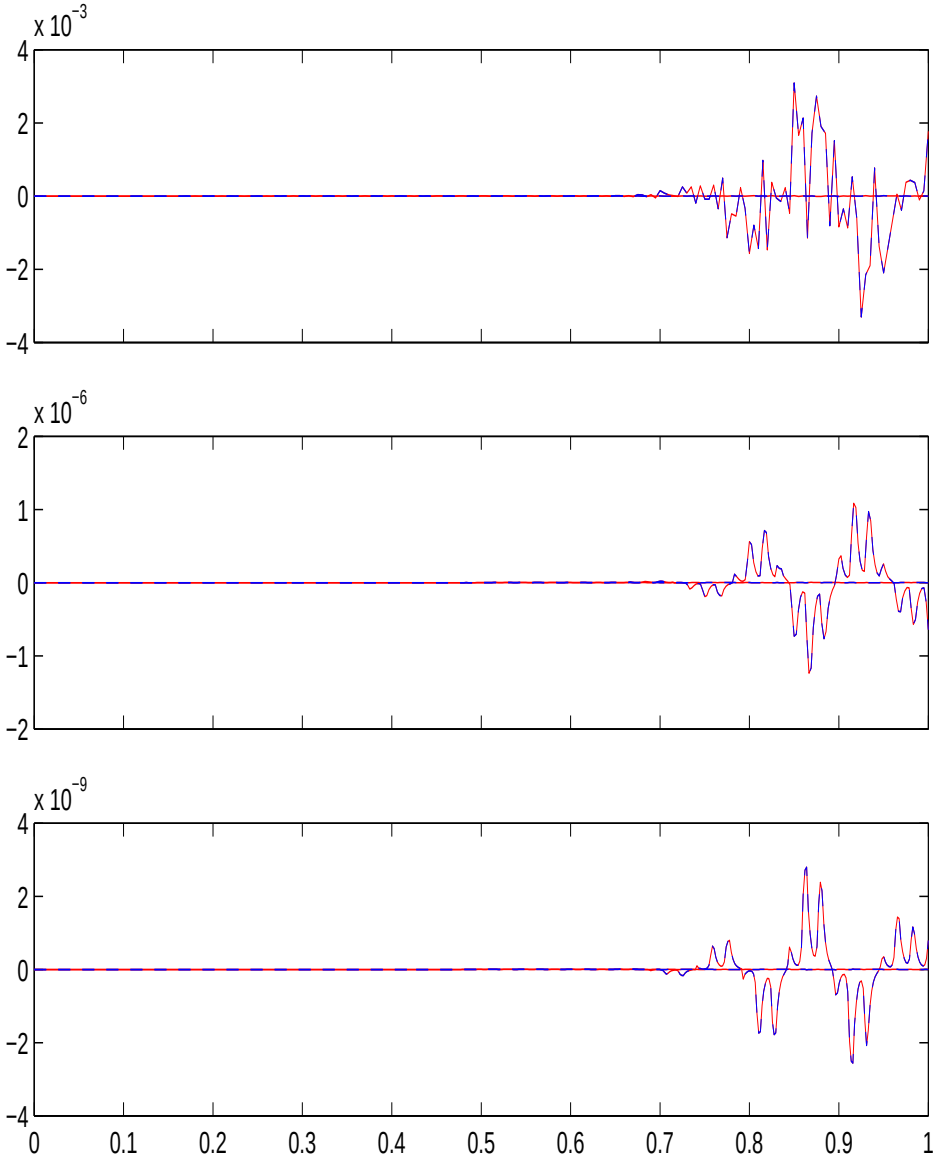


Figure 3.16: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 1001.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing1002	10^{-3}	0	23	0.090
funsing1002	10^{-3}	1	23	0.110
funsing1002	10^{-6}	0	33	0.131
funsing1002	10^{-6}	1	23	0.090
funsing1002	10^{-8}	0	32	0.150
funsing1002	10^{-8}	1	22	0.090

Table 3.80: Runtimes and number of meshpoints when testing the MonitorFunction. A slight advantage for the smoothed MonitorFunction can be observed.

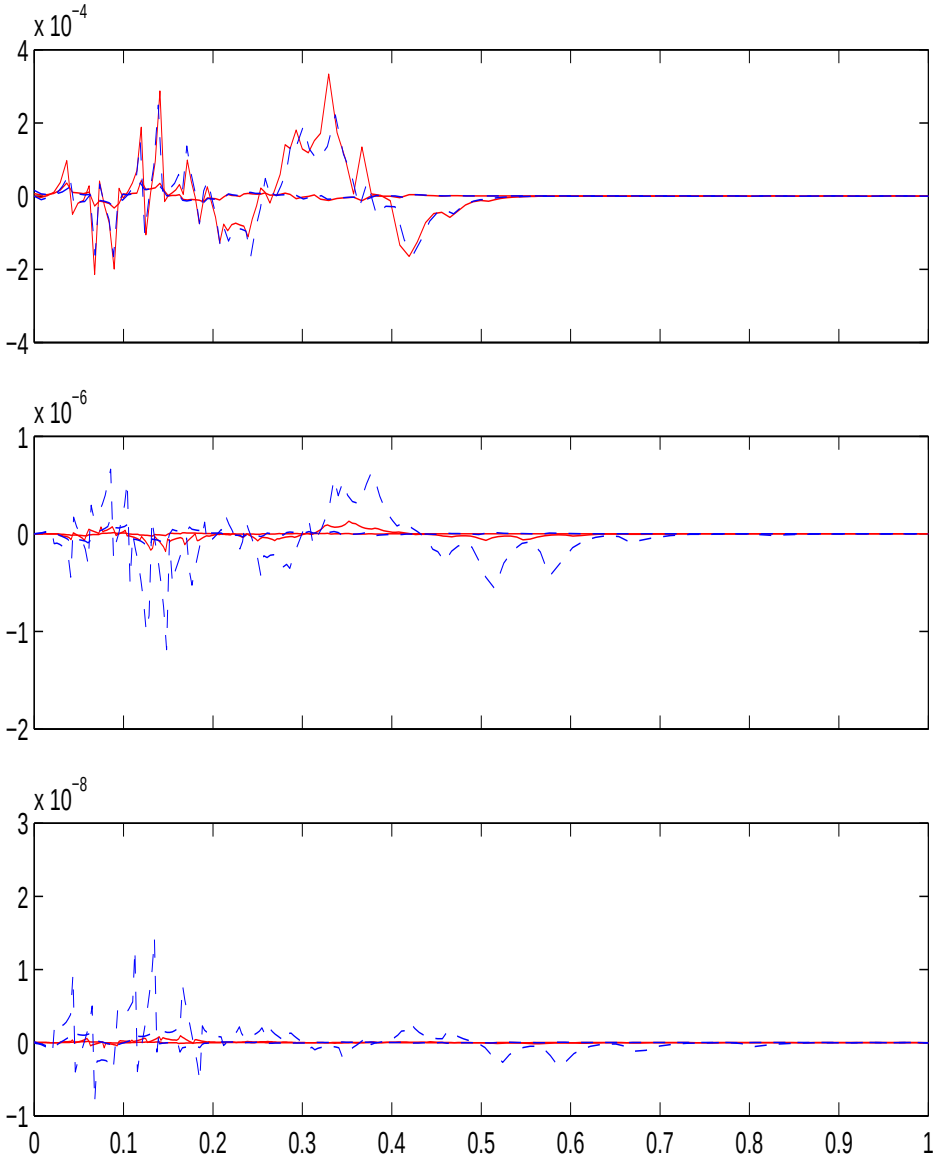


Figure 3.17: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 1002.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing1004	10^{-3}	0	9	0.050
funsing1004	10^{-3}	1	9	0.050
funsing1004	10^{-6}	0	15	0.070
funsing1004	10^{-6}	1	15	0.080
funsing1004	10^{-8}	0	15	0.080
funsing1004	10^{-8}	1	15	0.090

Table 3.81: Runtimes and number of meshpoints when testing the MonitorFunction. The runtimes are about the same.

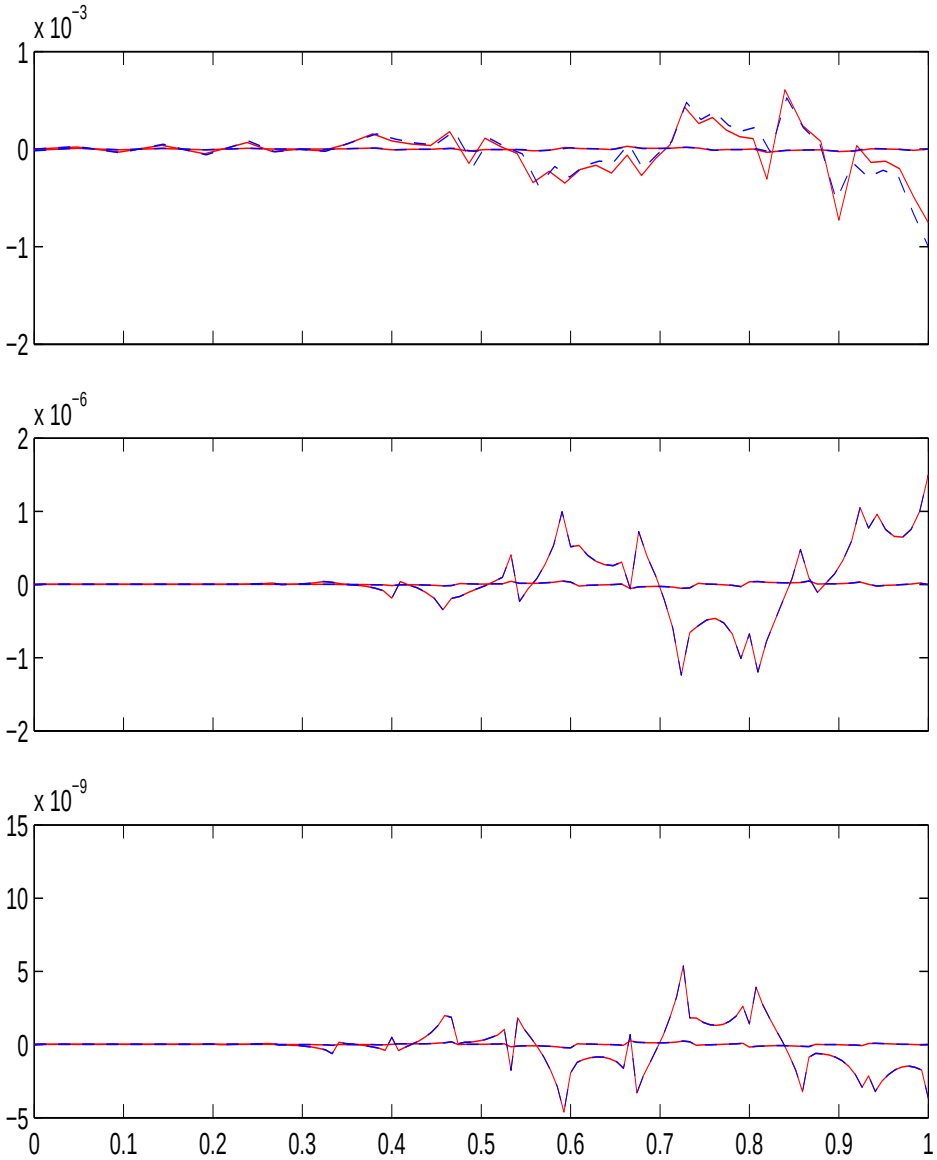


Figure 3.18: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 1004.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing1005	10^{-3}	0	240	0.541
funsing1005	10^{-3}	1	240	0.561
funsing1005	10^{-6}	0	120	0.401
funsing1005	10^{-6}	1	120	0.400
funsing1005	10^{-8}	0	138	0.651
funsing1005	10^{-8}	1	140	0.661

Table 3.82: Runtimes and number of meshpoints when testing the MonitorFunction. With the MonitorFunction set to 0, the strict tolerance is satisfied with fewer points.

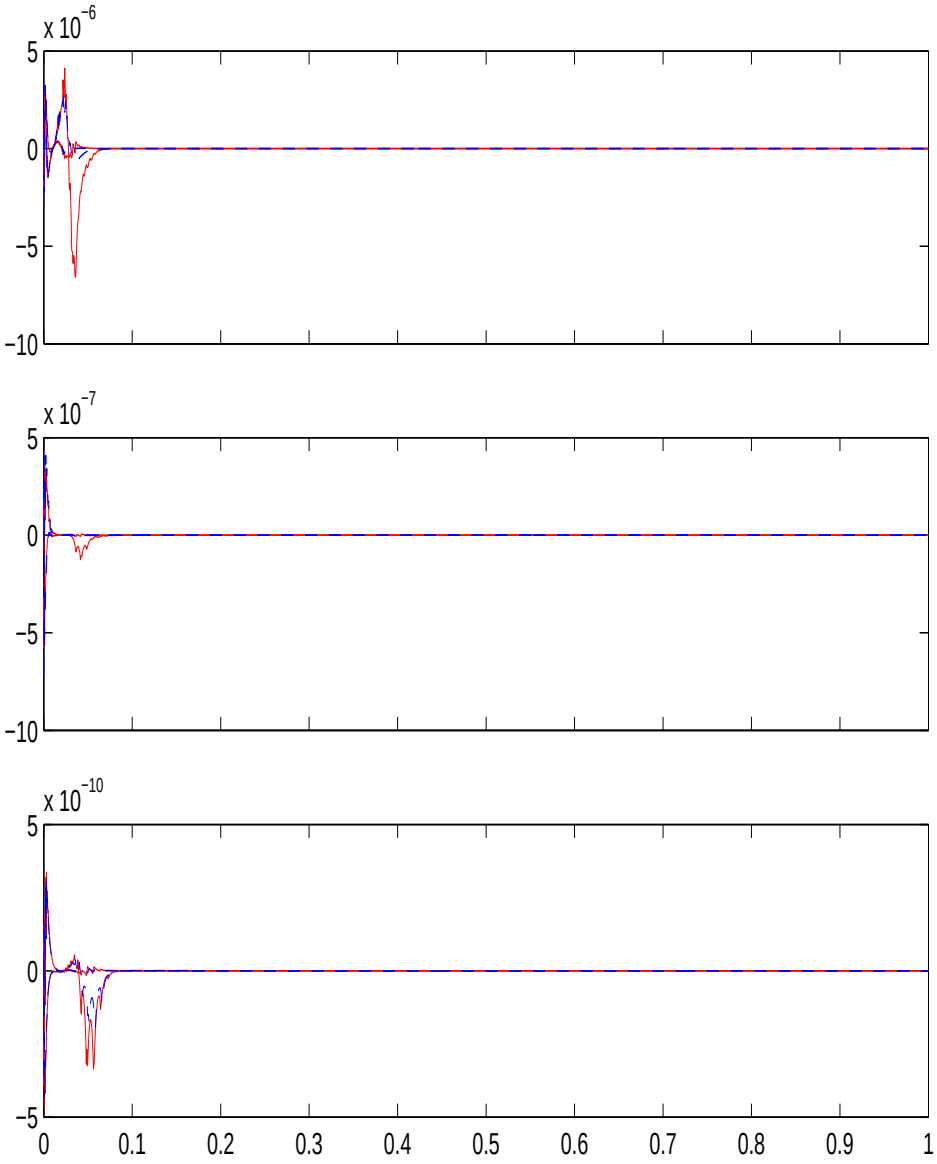


Figure 3.19: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 1005.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing015	10^{-3}	0	20	0.060
funsing015	10^{-3}	1	20	0.060
funsing015	10^{-6}	0	33	0.130
funsing015	10^{-6}	1	33	0.131
funsing015	10^{-8}	0	26	0.141
funsing015	10^{-8}	1	26	0.141

Table 3.83: Runtimes and number of meshpoints when testing the MonitorFunction. The runtimes are about the same.

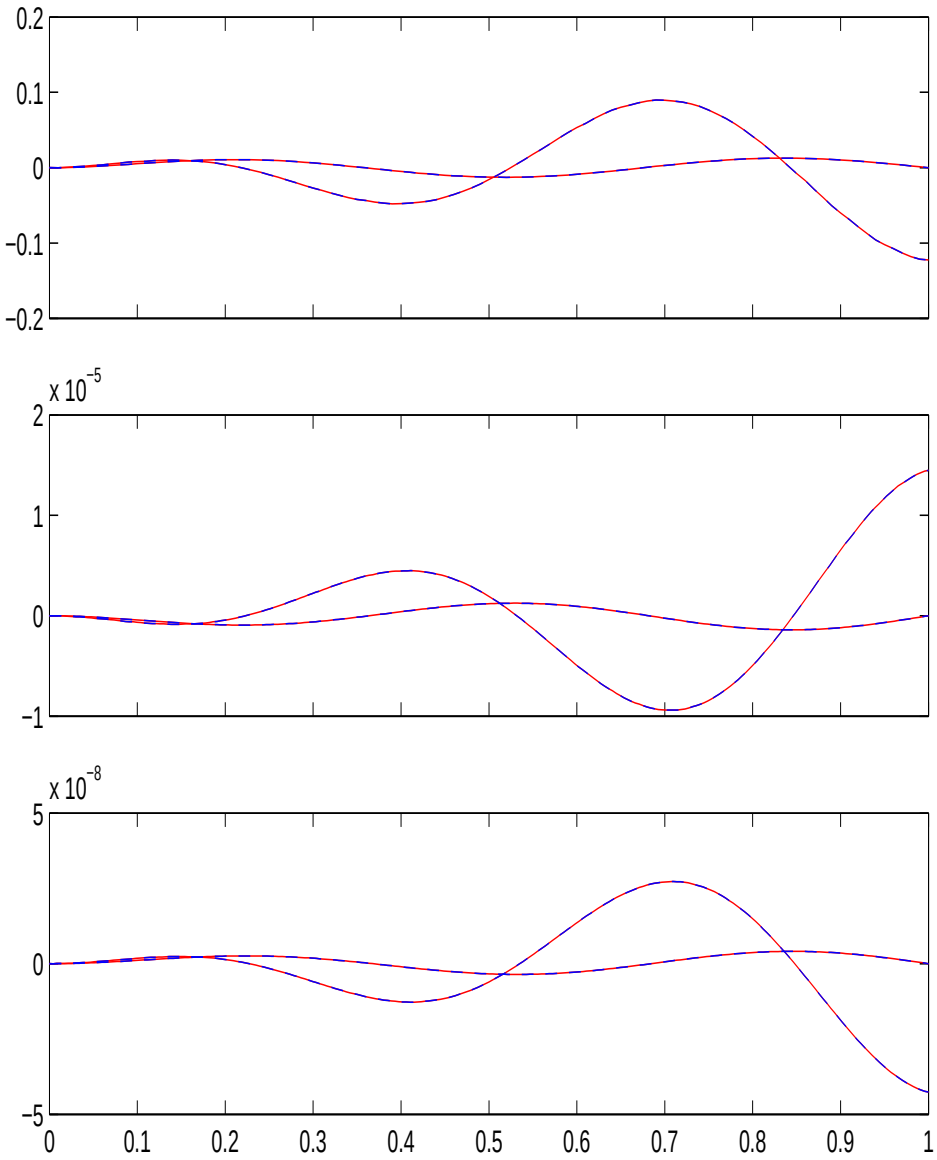


Figure 3.20: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 015.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing15	10^{-3}	0	33	0.110
funsing15	10^{-3}	1	33	0.120
funsing15	10^{-6}	0	33	0.130
funsing15	10^{-6}	1	33	0.130
funsing15	10^{-8}	0	26	0.130
funsing15	10^{-8}	1	26	0.140

Table 3.84: Runtimes and number of meshpoints when testing the MonitorFunction. The runtimes are about the same.

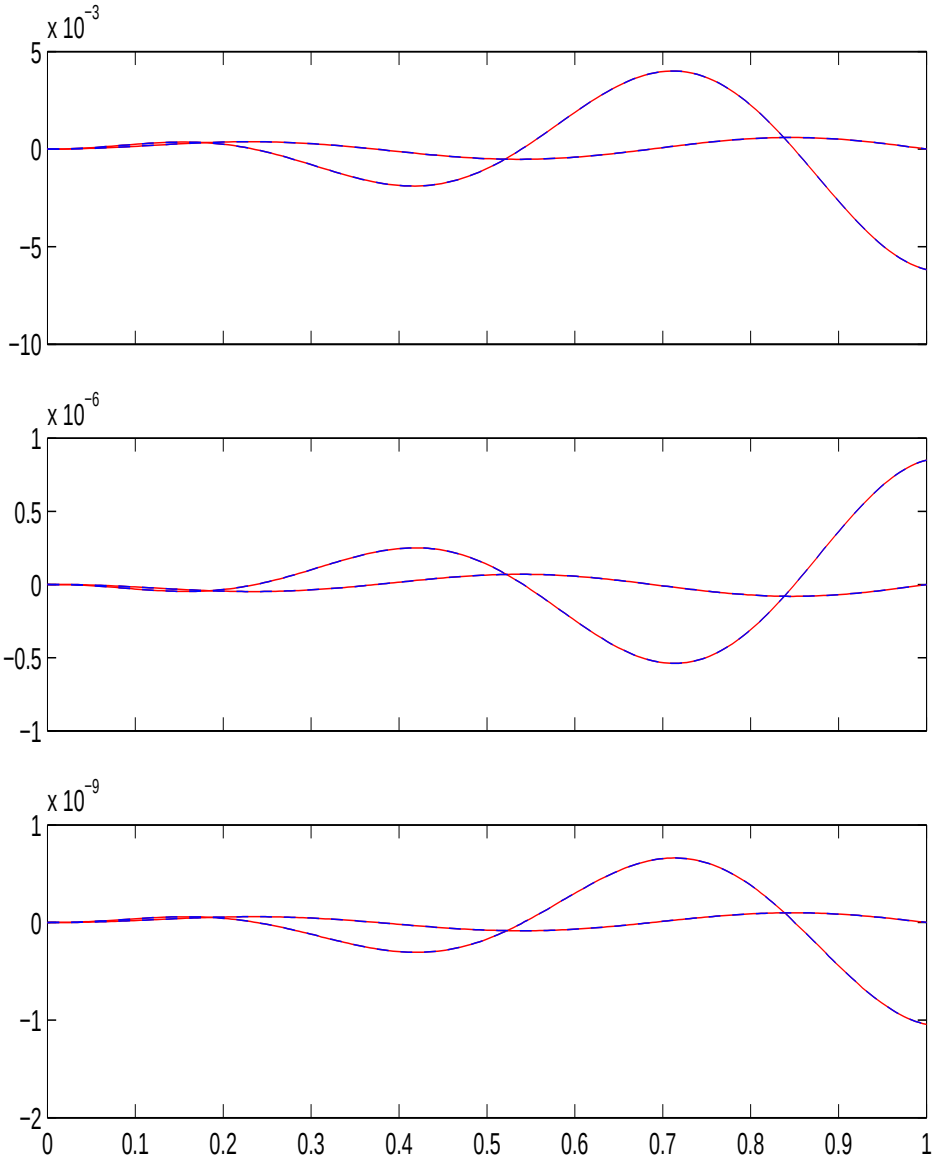


Figure 3.21: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 15.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing2002	10^{-3}	0	48	0.140
funsing2002	10^{-3}	1	34	0.090
funsing2002	10^{-6}	0	74	0.210
funsing2002	10^{-6}	1	75	0.211
funsing2002	10^{-8}	0	53	0.210
funsing2002	10^{-8}	1	54	0.220

Table 3.85: Runtimes and number of meshpoints when testing the MonitorFunction. While only 75 % of the number of meshpoints are used for the tolerance at 10^{-3}, the other results are a bit worse for the smoothed MonitorFunction.

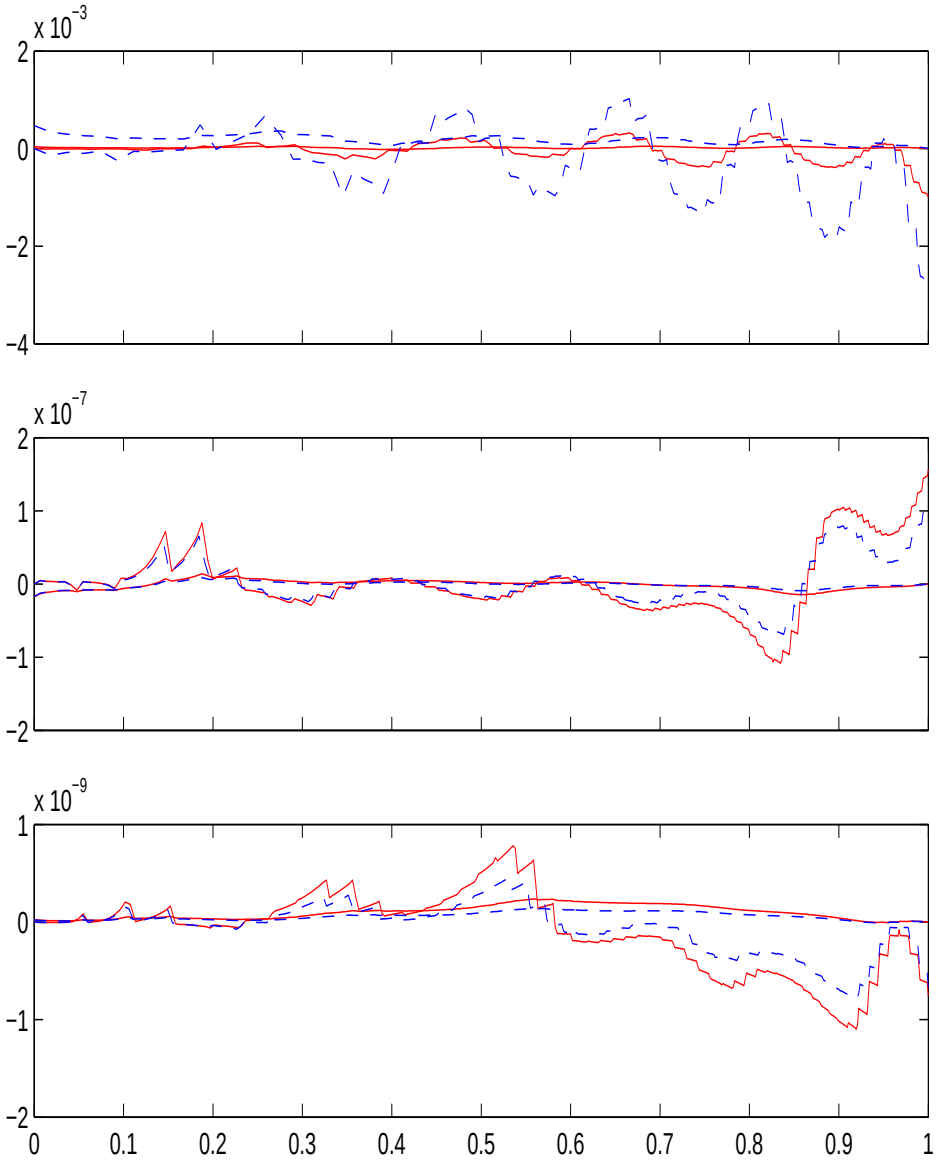


Figure 3.22: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 2002.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing2008	10^{-3}	0	147	0.300
funsing2008	10^{-3}	1	90	0.170
funsing2008	10^{-6}	0	161	0.500
funsing2008	10^{-6}	1	159	0.520
funsing2008	10^{-8}	0	132	0.391
funsing2008	10^{-8}	1	133	0.381

Table 3.86: Runtimes and number of meshpoints when testing the Monitor-Function. Better results for the smoothed MonitorFunction with the tolerance at 10^{-3}.

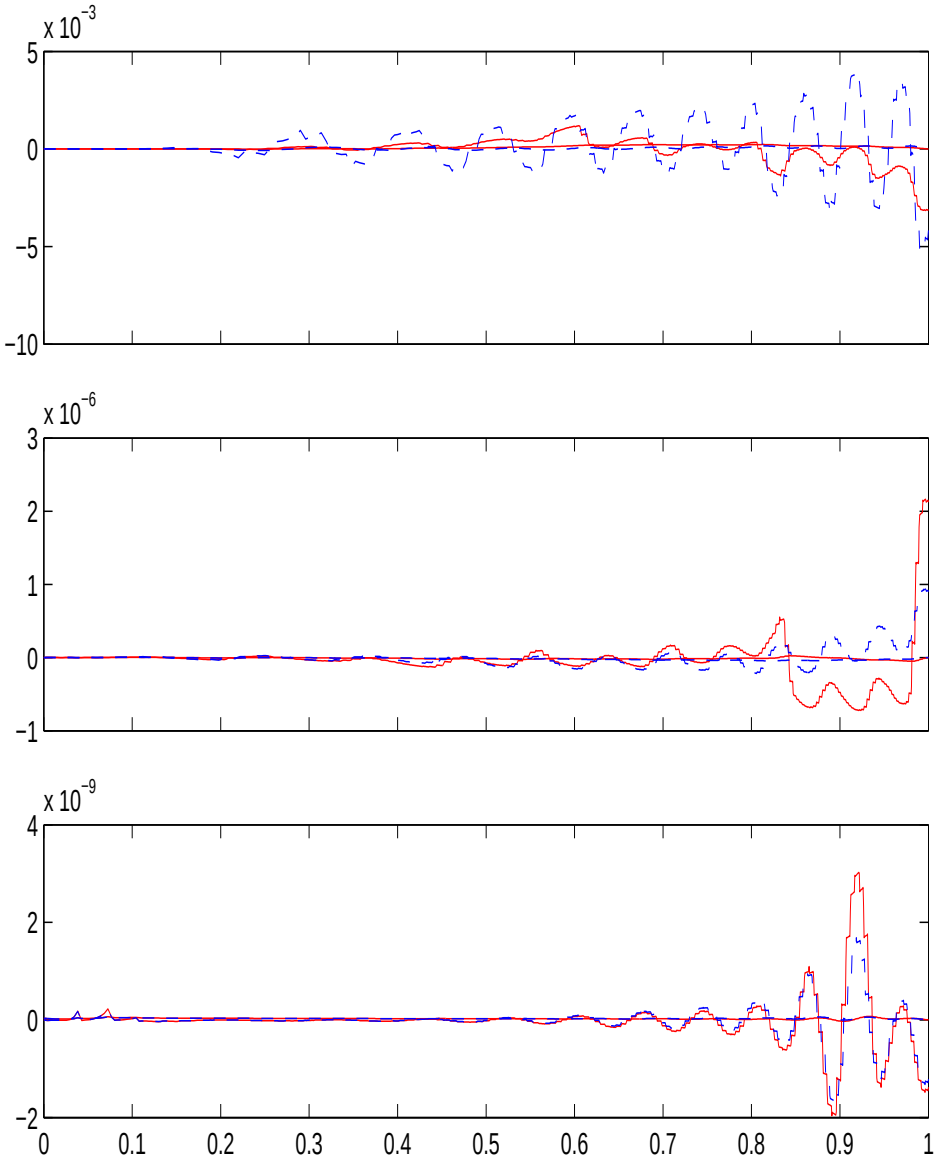


Figure 3.23: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 2008.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing21a	10^{-3}	0	5	0.070
funsing21a	10^{-3}	1	5	0.070
funsing21a	10^{-6}	0	10	0.090
funsing21a	10^{-6}	1	10	0.090
funsing21a	10^{-8}	0	10	0.100
funsing21a	10^{-8}	1	10	0.110

Table 3.87: Runtimes and number of meshpoints when testing the MonitorFunction. There is hardly a difference in the runtimes.

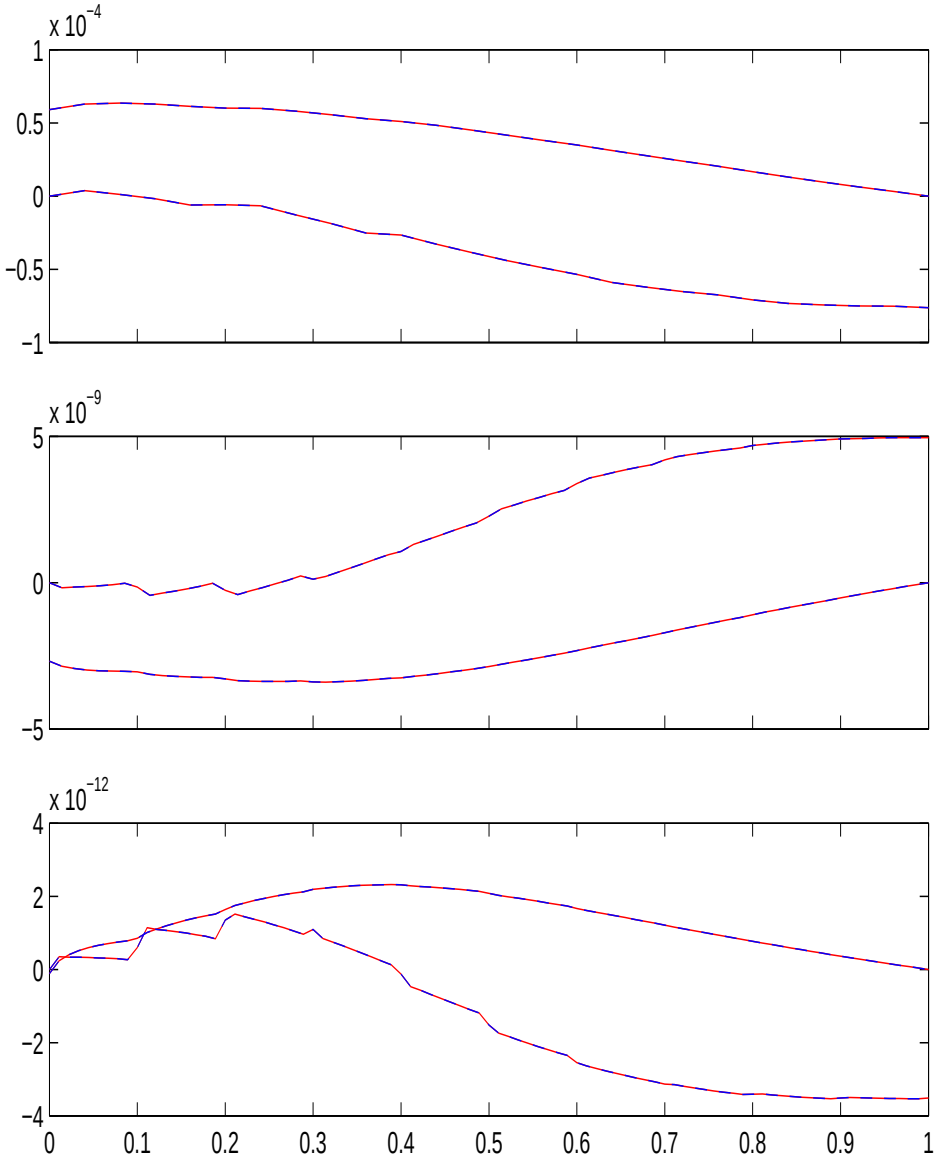


Figure 3.24: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 21a.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing400	10^{-3}	0	5	0.040
funsing400	10^{-3}	1	5	0.040
funsing400	10^{-6}	0	10	0.050
funsing400	10^{-6}	1	10	0.060
funsing400	10^{-8}	0	10	0.060
funsing400	10^{-8}	1	10	0.040

Table 3.88: Runtimes and number of meshpoints when testing the MonitorFunction. A slight advantage for the smoothed MonitorFunction can be observed.

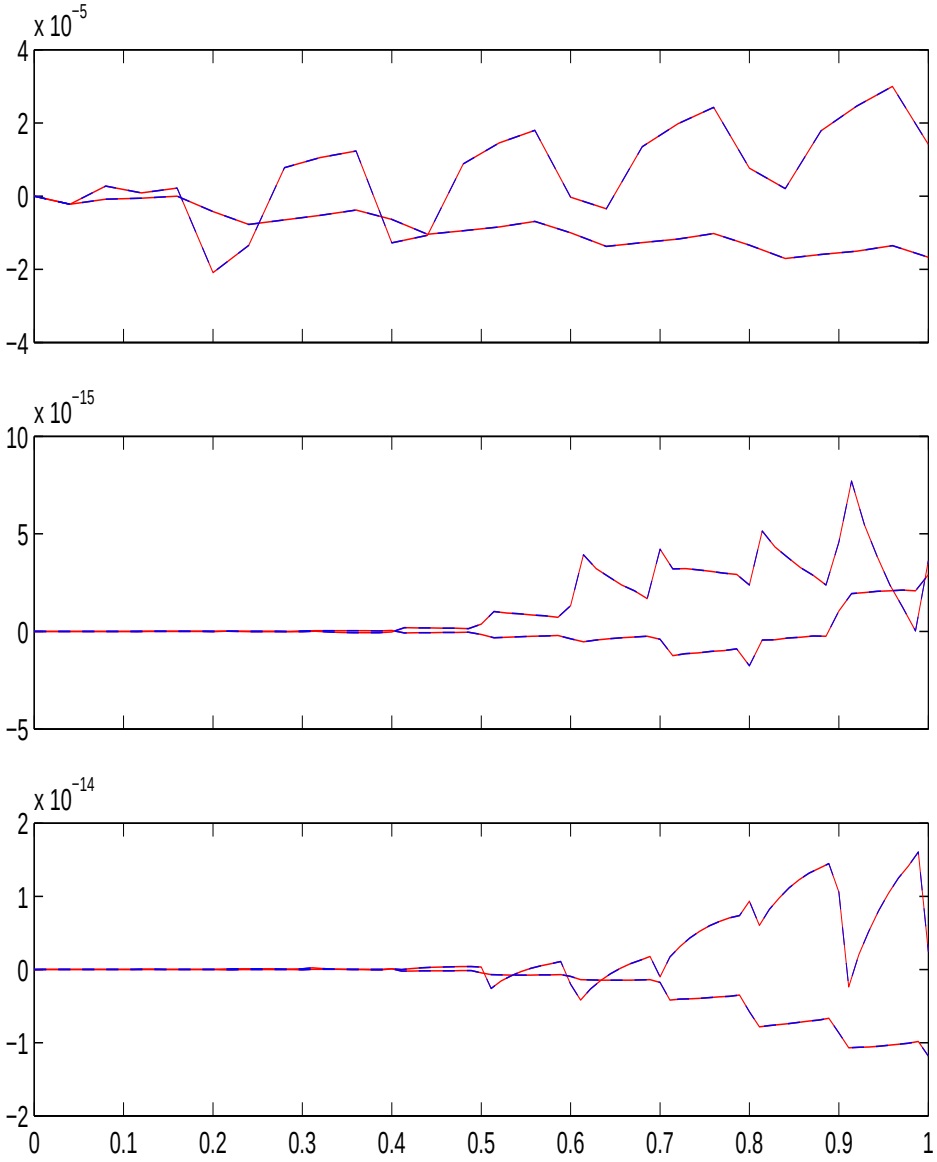


Figure 3.25: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 400.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing500	10^{-3}	0	5	0.040
funsing500	10^{-3}	1	5	0.040
funsing500	10^{-6}	0	10	0.051
funsing500	10^{-6}	1	10	0.040
funsing500	10^{-8}	0	10	0.040
funsing500	10^{-8}	1	10	0.040

Table 3.89: Runtimes and number of meshpoints when testing the MonitorFunction. This example is solved very quickly. No difference between both options.

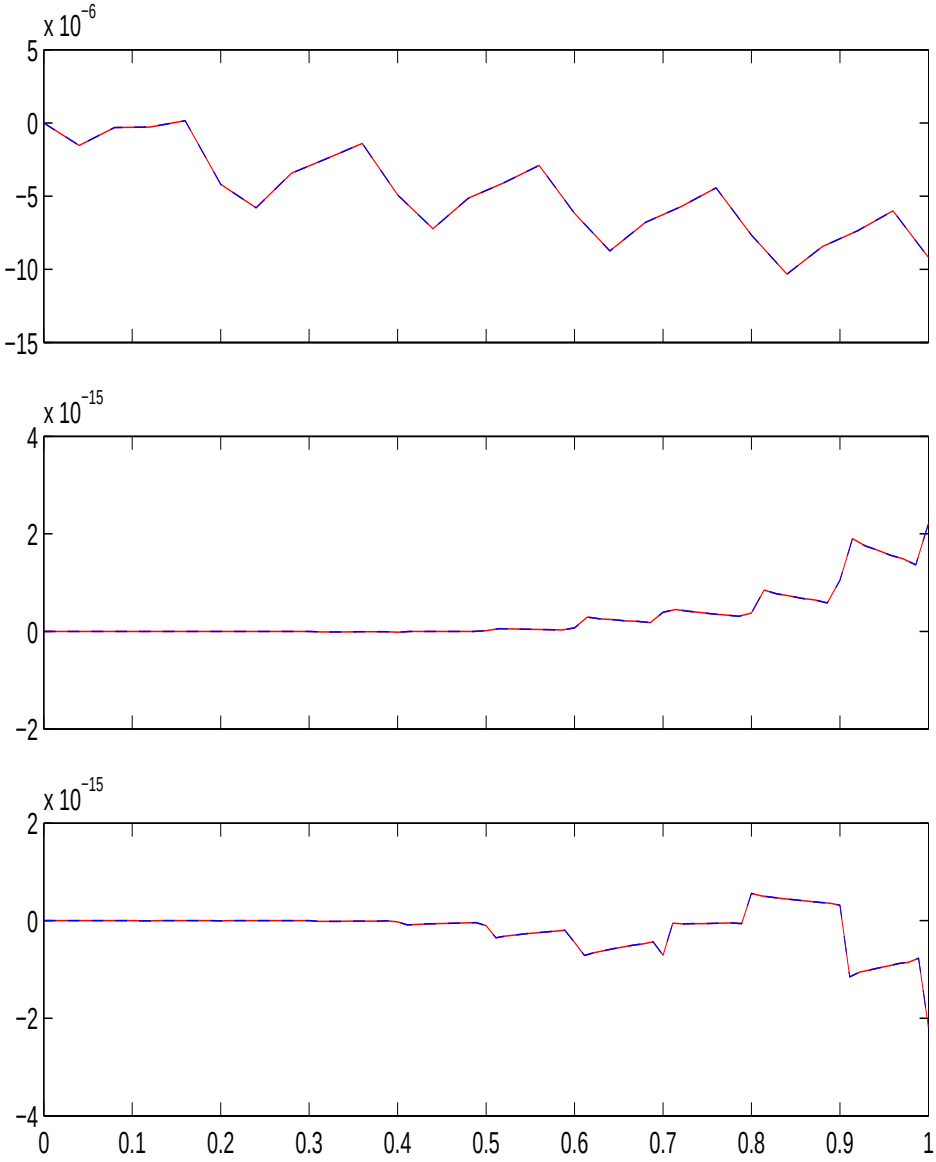


Figure 3.26: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 500.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing56	10^{-3}	0	192	74.758
funsing56	10^{-3}	1	128	0.260
funsing56	10^{-6}	0	80	45.896
funsing56	10^{-6}	1	80	47.028
funsing56	10^{-8}	0	80	74.126
funsing56	10^{-8}	1	80	76.060

Table 3.90: Runtimes and number of meshpoints when testing the MonitorFunction. The high runtime for the tolerance set to 10^{-3} is striking.

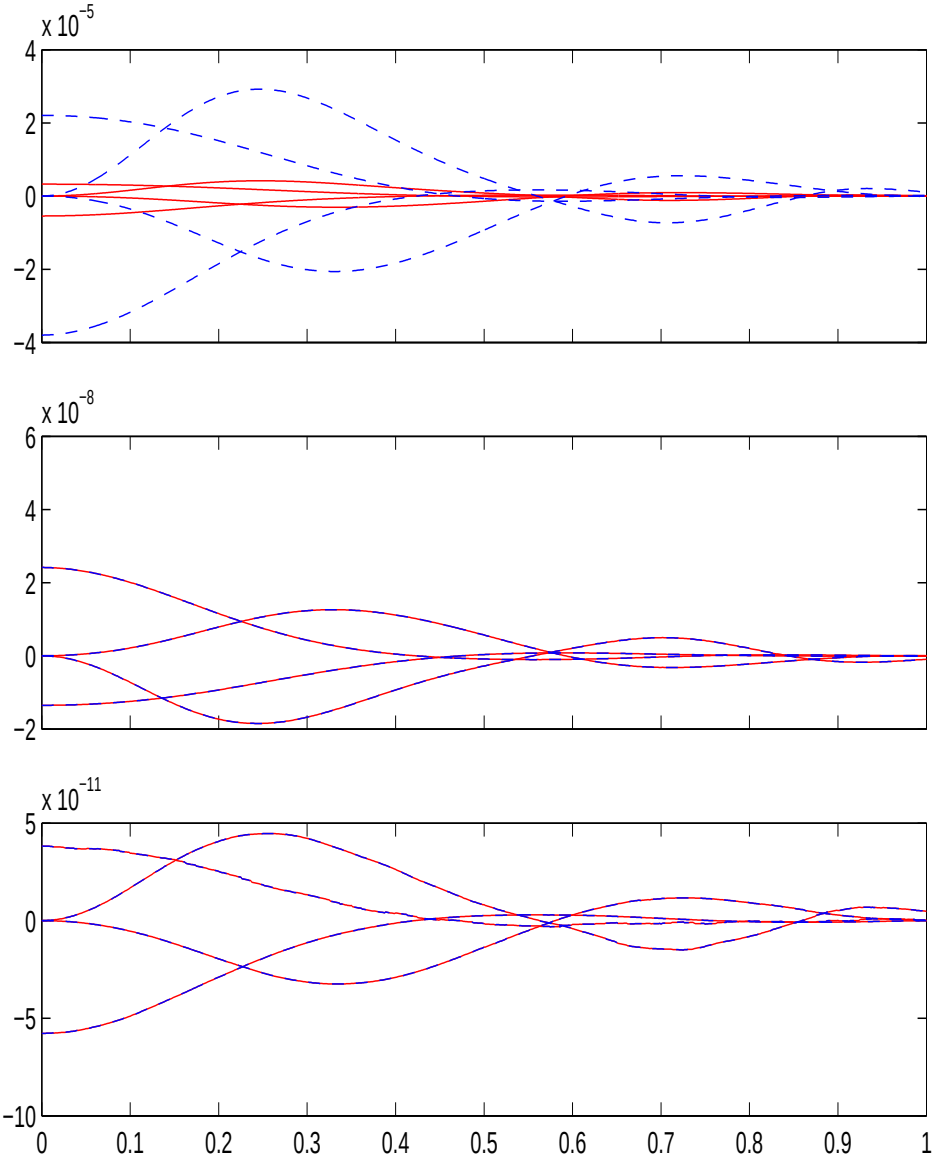


Figure 3.27: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 56.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing6001	10^{-3}	0	10	0.140
funsing6001	10^{-3}	1	10	0.130
funsing6001	10^{-6}	0	15	0.170
funsing6001	10^{-6}	1	15	0.180
funsing6001	10^{-8}	0	15	0.230
funsing6001	10^{-8}	1	15	0.230

Table 3.91: Runtimes and number of meshpoints when testing the MonitorFunction. About the same results for both option settings.

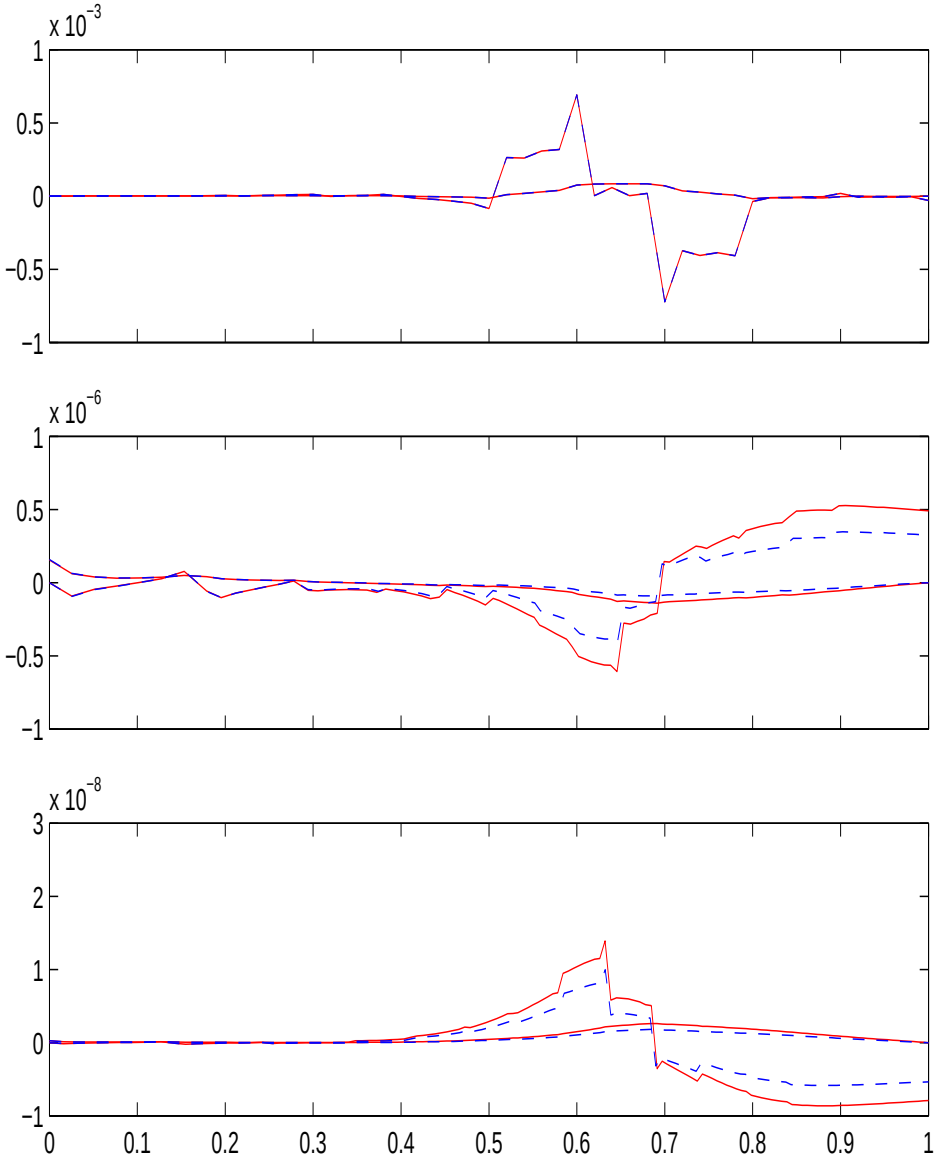


Figure 3.28: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 6001.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing6002	10^{-3}	0	5	0.060
funsing6002	10^{-3}	1	5	0.060
funsing6002	10^{-6}	0	10	0.071
funsing6002	10^{-6}	1	10	0.070
funsing6002	10^{-8}	0	10	0.090
funsing6002	10^{-8}	1	10	0.090

Table 3.92: Runtimes and number of meshpoints when testing the MonitorFunction. Again comparable results.

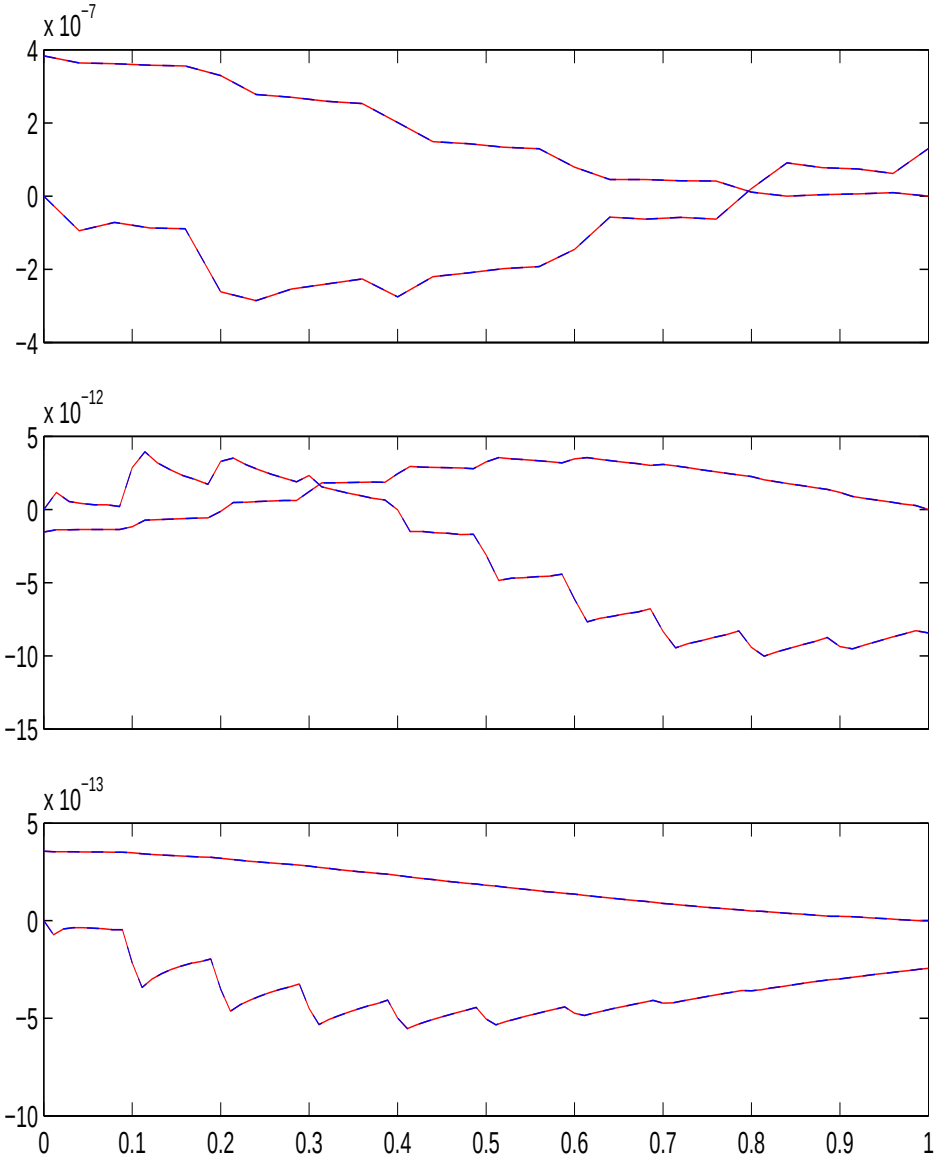


Figure 3.29: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 6002.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing7001b	10^{-3}	0	5	0.060
funsing7001b	10^{-3}	1	5	0.060
funsing7001b	10^{-6}	0	10	0.080
funsing7001b	10^{-6}	1	10	0.090
funsing7001b	10^{-8}	0	10	0.100
funsing7001b	10^{-8}	1	10	0.100

Table 3.93: Runtimes and number of meshpoints when testing the MonitorFunction. Both options perform well for this Bvp.

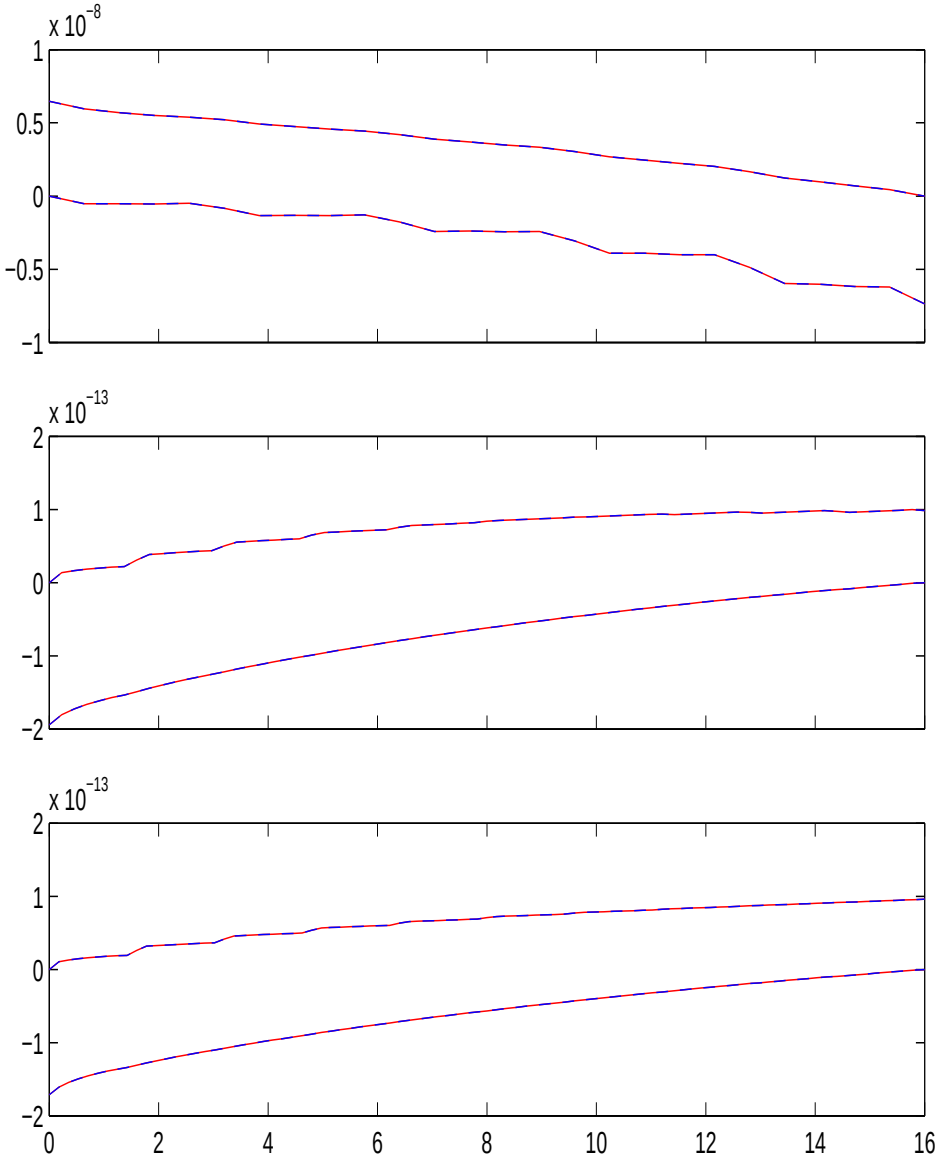


Figure 3.30: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 7001b.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing71	10^{-3}	0	21	0.180
funsing71	10^{-3}	1	14	0.120
funsing71	10^{-6}	0	42	0.311
funsing71	10^{-6}	1	28	0.190
funsing71	10^{-8}	0	45	0.441
funsing71	10^{-8}	1	45	0.431

Table 3.94: Runtimes and number of meshpoints when testing the MonitorFunction. Better results for the smoothed MonitorFunction in two cases, a comparable result in the third case.

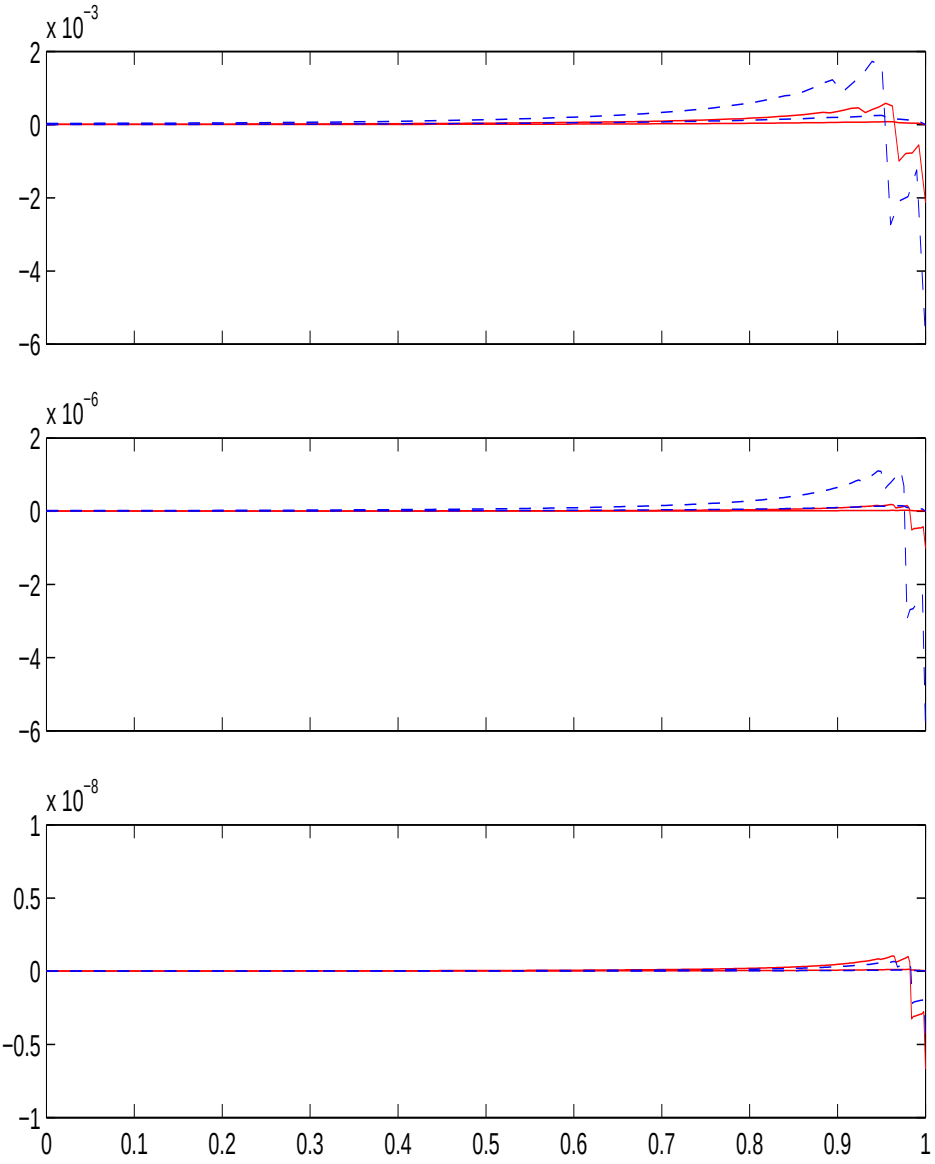


Figure 3.31: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 71.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing8001	10^{-3}	0	5	0.120
funsing8001	10^{-3}	1	5	0.070
funsing8001	10^{-6}	0	10	0.100
funsing8001	10^{-6}	1	10	0.090
funsing8001	10^{-8}	0	10	0.130
funsing8001	10^{-8}	1	10	0.120

Table 3.95: Runtimes and number of meshpoints when testing the MonitorFunction. A slight advantage for the smoothed MonitorFunction can be observed.

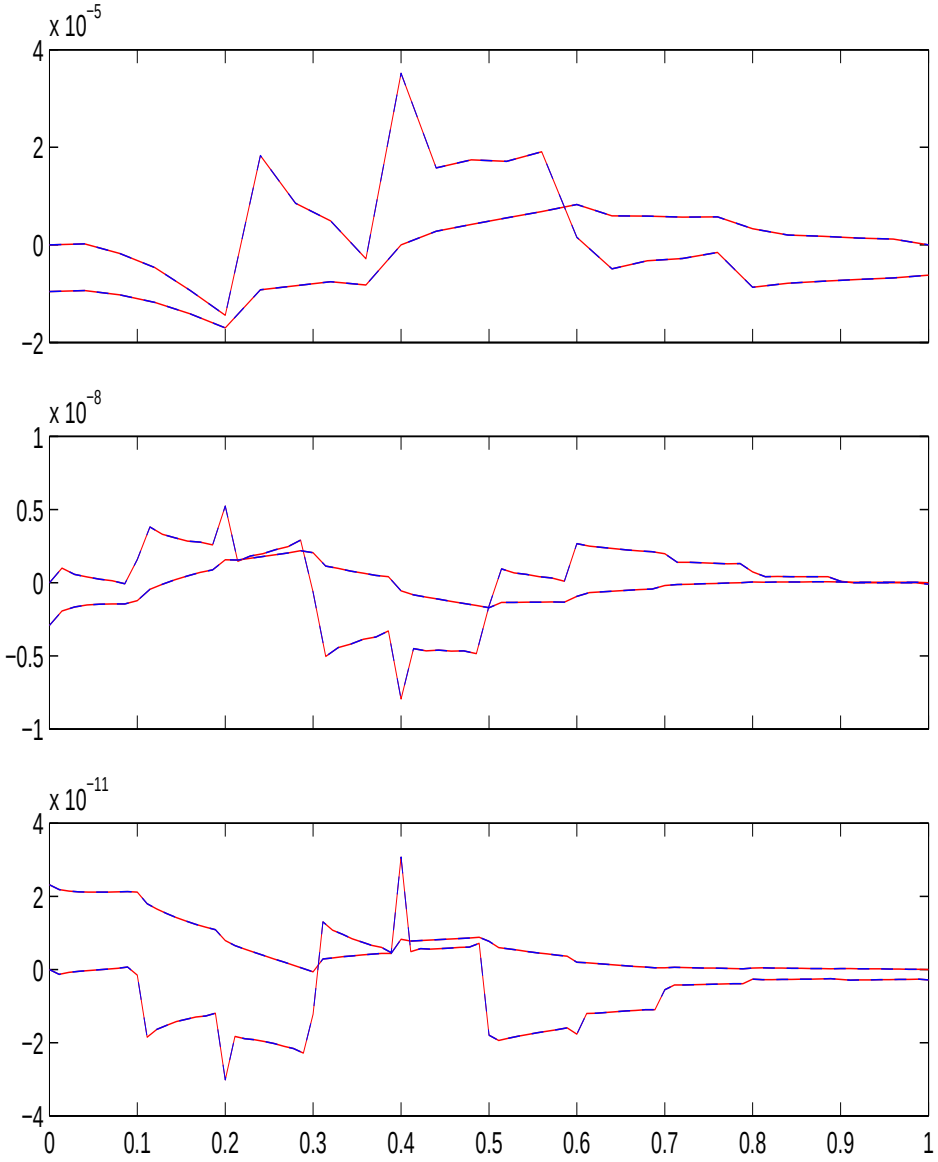


Figure 3.32: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 8001.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing8002	10^{-3}	0	8	0.140
funsing8002	10^{-3}	1	8	0.130
funsing8002	10^{-6}	0	15	0.180
funsing8002	10^{-6}	1	15	0.170
funsing8002	10^{-8}	0	15	0.220
funsing8002	10^{-8}	1	15	0.230

Table 3.96: Runtimes and number of meshpoints when testing the MonitorFunction. There is hardly a difference for BVP 8002.

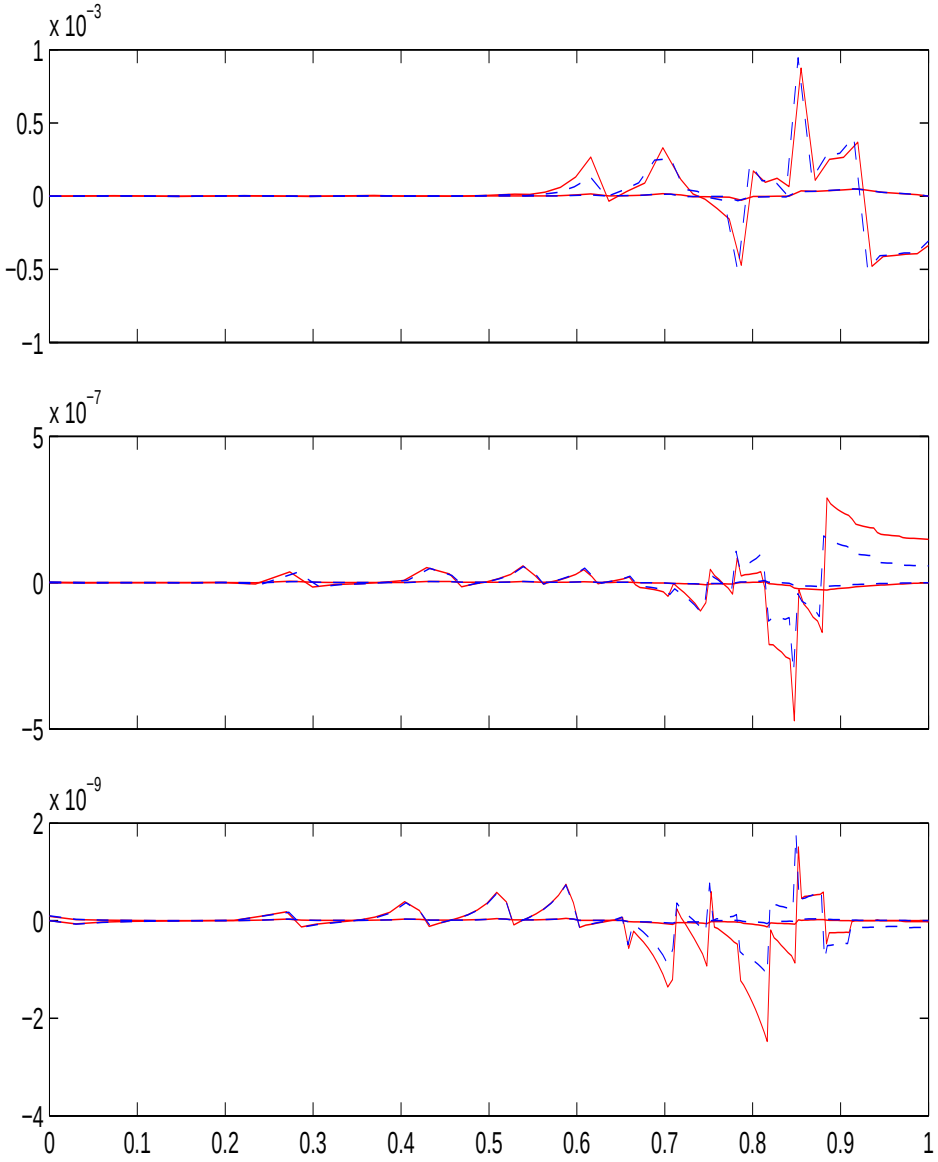


Figure 3.33: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 8002.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing9001	10^{-3}	0	107	0.240
funsing9001	10^{-3}	1	107	0.270
funsing9001	10^{-6}	0	81	0.220
funsing9001	10^{-6}	1	81	0.220
funsing9001	10^{-8}	0	45	0.120
funsing9001	10^{-8}	1	45	0.140

Table 3.97: Runtimes and number of meshpoints when testing the MonitorFunction. The runtimes are comparable.

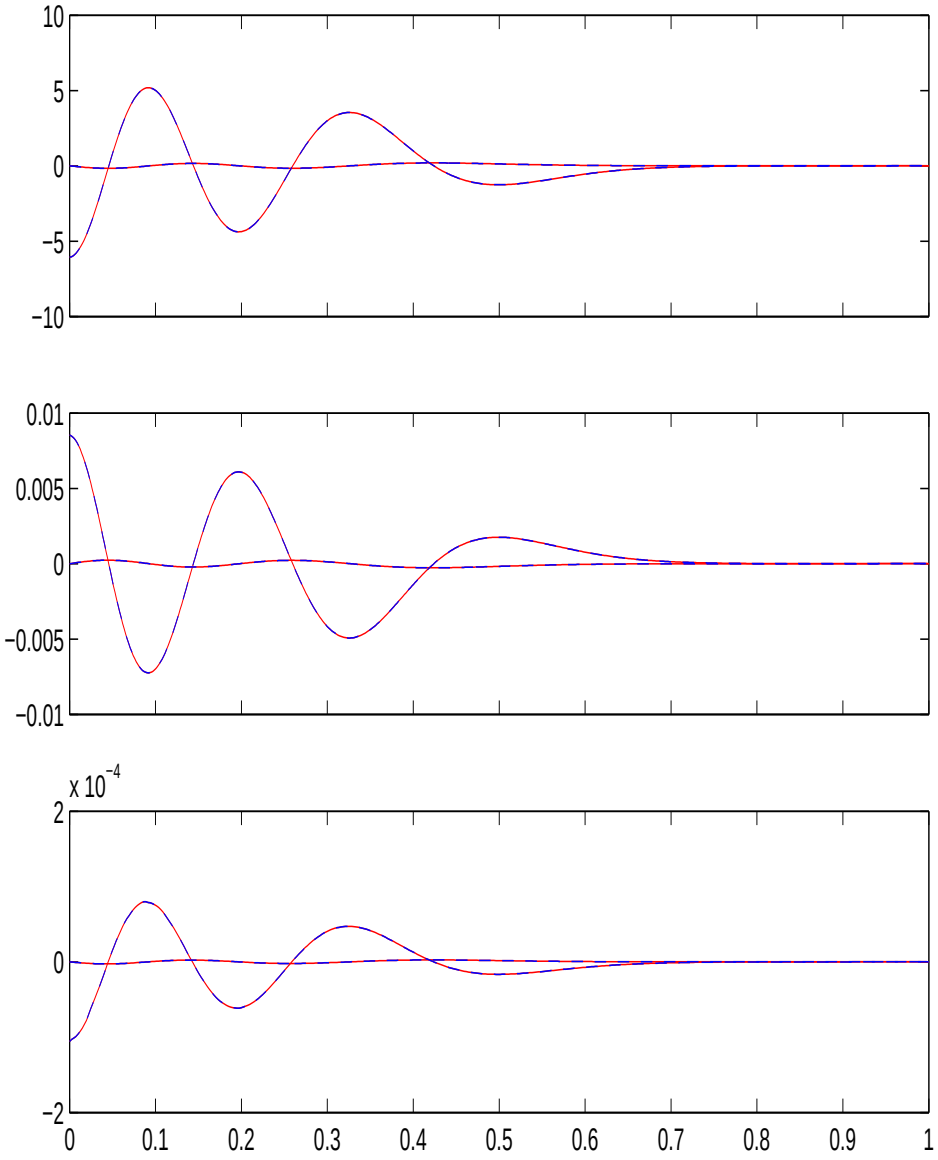


Figure 3.34: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 9001.

BVP	Tol	Mon.-F.	No. of MP	Runtime
funsing9002	10^{-3}	0	147	0.280
funsing9002	10^{-3}	1	147	0.280
funsing9002	10^{-6}	0	92	0.230
funsing9002	10^{-6}	1	92	0.241
funsing9002	10^{-8}	0	69	0.270
funsing9002	10^{-8}	1	69	0.270

Table 3.98: Runtimes and number of meshpoints when testing the MonitorFunction. The runtimes are comparable.

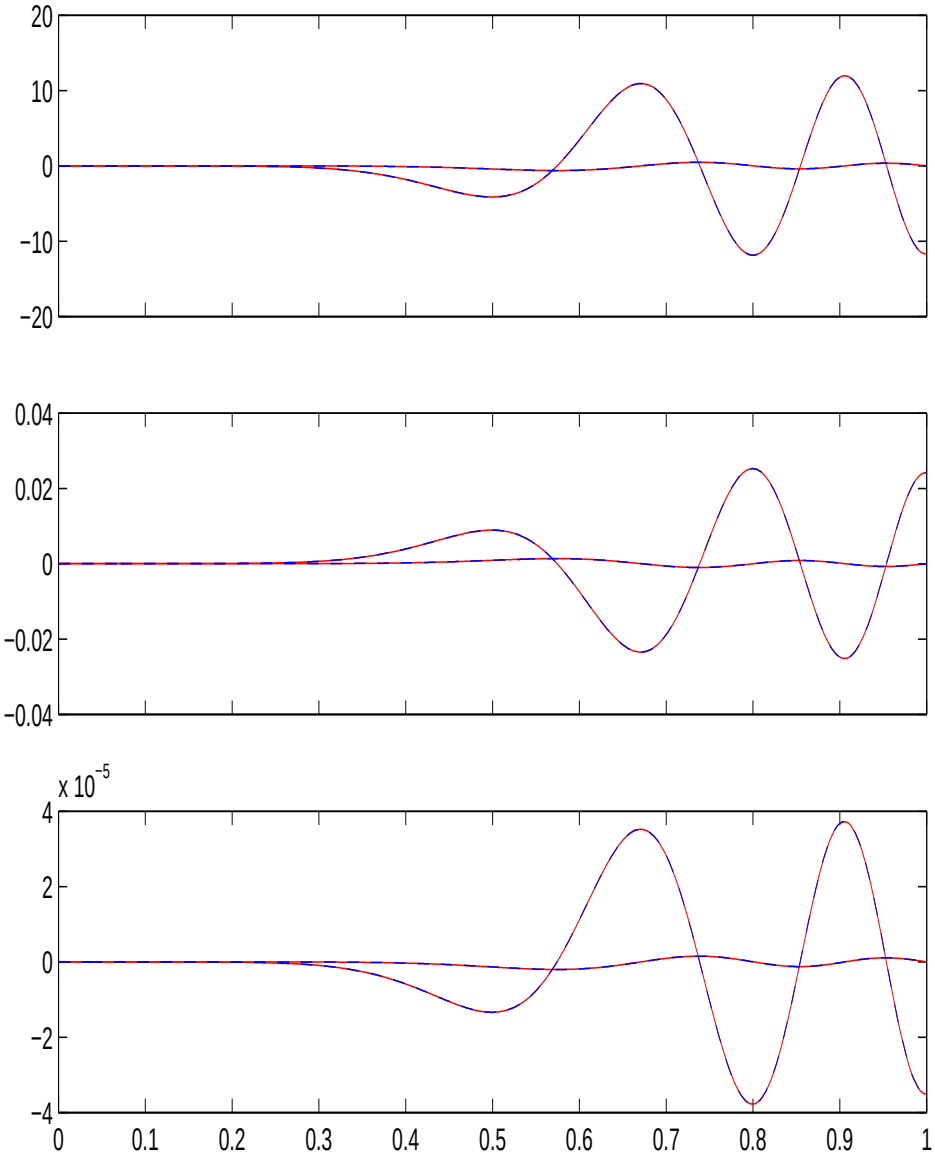


Figure 3.35: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving BVP 9002.

Bvp	Tol	Mon.-F.	No. of MP	Runtime
funsing9003	10^{-3}	0	10	0.061
funsing9003	10^{-3}	1	10	0.040
funsing9003	10^{-6}	0	27	0.110
funsing9003	10^{-6}	1	27	0.120
funsing9003	10^{-8}	0	16	0.090
funsing9003	10^{-8}	1	16	0.090

Table 3.99: Runtimes and number of meshpoints when testing the MonitorFunction. Again the results are about the same.

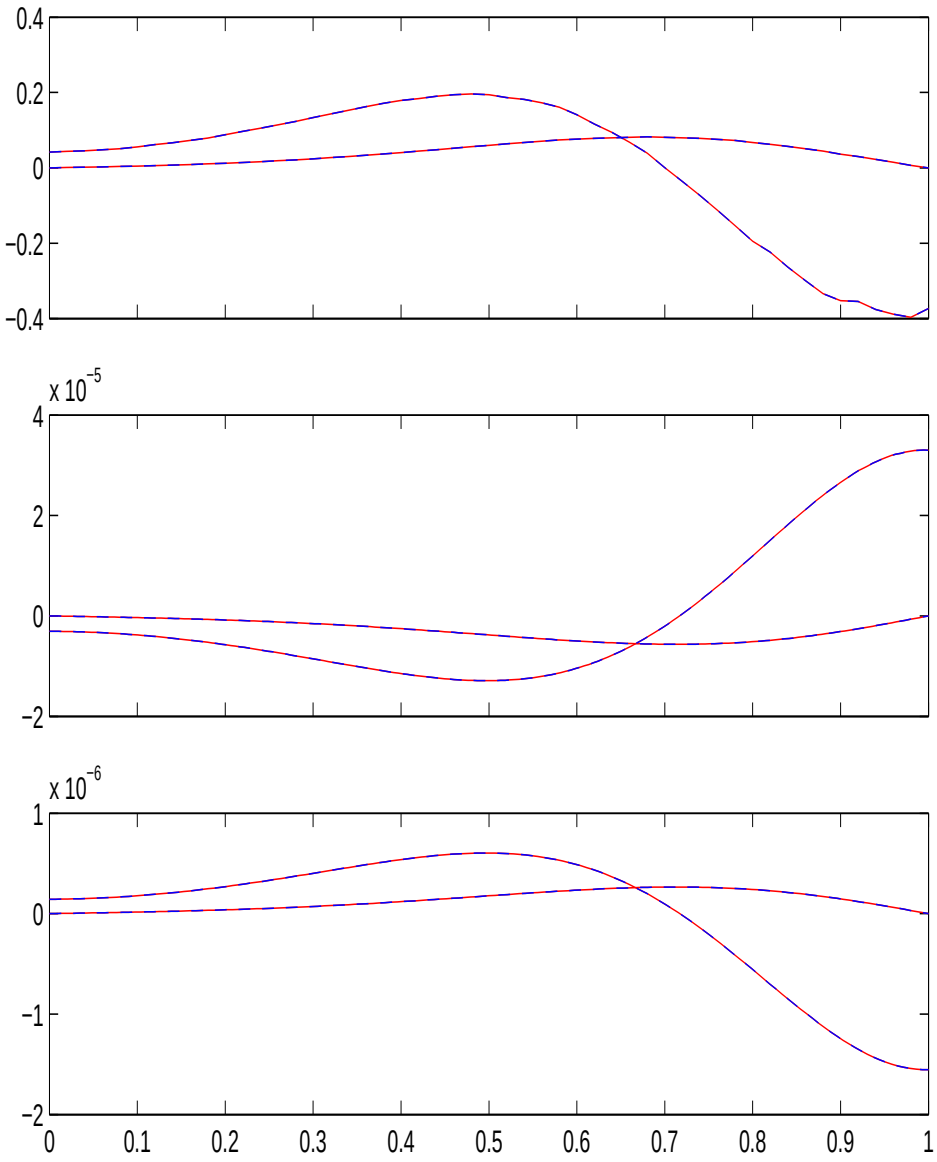


Figure 3.36: Final error function for tolerances 10^{-3}, 10^{-6}, 10^{-8} when using a smoothed or non-smoothed MonitorFunction for solving Bvp 9003.

3.3 Initial Mesh

One of the last questions in optimizing SBVP 2.0 is the optimization of the starting grid. SBVP 1.0 used to determine the number of meshpoints N_0 in the starting grid depending on the absolute tolerance and the collocation order,

$$N_0 = \min(100, \max(5, \left\lfloor \left(\frac{1}{T_{abs}}\right)^{\frac{1}{p}} \right\rfloor)).$$

With this formula at least 5 meshpoints are used and a maximal number of meshpoints in the initial mesh is 100. To find out if 5 meshpoints are worth the risk of an additional step of refinement, the standard method is compared with grids that always start with 5, 10 and 20 points.

3.3.1 Test Results

The investigations for the starting grids cover two series of tests. The first uses automatic degree-selections, therefore the automatic grid determination leads to 5 points for the tolerance 10^{-3} and 10 points for the tolerances 10^{-6} and 10^{-8}. The results in Tables 3.100 - 3.126 show the cases with the fixed starting grids with 5, 10 and 20 equidistant meshpoints.

The main reason for the test for 5 meshpoints is to see if the results are stable and if the cases where 5 meshpoints satisfy the tolerances are a big advantage. The test for 20 meshpoints is motivated by the question how often the mesh with 20 points is fine enough to satisfy the tolerances. Additionally the distribution of the error at 20 meshpoints resolves the critical zones better than the distribution at 5 meshpoints.

BVP	Tol	N_0	N	Time
funsing0026	10^{-3}	5	5	0.080
funsing0026	10^{-3}	10	10	0.080
funsing0026	10^{-3}	20	20	0.090
funsing0026	10^{-6}	5	5	0.090
funsing0026	10^{-6}	10	10	0.080
funsing0026	10^{-6}	20	20	0.121
funsing0026	10^{-8}	5	5	0.071
funsing0026	10^{-8}	10	10	0.100
funsing0026	10^{-8}	20	20	0.130

Table 3.100: Results for different starting grids. Here, the starting mesh coincides with the final one. Therefore, the finer starting meshes are less favorable.

BVP	Tol	N_0	N	Time
funsing1001	10^{-3}	5	40	0.120
funsing1001	10^{-3}	10	40	0.091
funsing1001	10^{-3}	20	40	0.080
funsing1001	10^{-6}	5	60	0.220
funsing1001	10^{-6}	10	60	0.211
funsing1001	10^{-6}	20	60	0.200
funsing1001	10^{-8}	5	58	0.241
funsing1001	10^{-8}	10	58	0.220
funsing1001	10^{-8}	20	58	0.201

Table 3.101: Results for different starting grids. Tolerance 10^{-3} is satisfied rather quickly when starting with 20 points. Also 10 points are advantageous.

BVP	Tol	N_0	N	Time
funsing1002	10^{-3}	5	23	0.100
funsing1002	10^{-3}	10	23	0.090
funsing1002	10^{-3}	20	33	0.080
funsing1002	10^{-6}	5	35	0.140
funsing1002	10^{-6}	10	35	0.130
funsing1002	10^{-6}	20	45	0.170
funsing1002	10^{-8}	5	33	0.161
funsing1002	10^{-8}	10	22	0.090
funsing1002	10^{-8}	20	30	0.130

Table 3.102: Results for different starting grids. This BVP shows the best results for 10 points.

As Figure 3.37 shows, the better knowledge of the error distribution when starting with 10 points is of advantage after the first step of refinement, because the redistributed 22 points satisfy the tolerances, whereas the redistribution after starting with 5 points fails. The distribution of only 20 points is not suitable to reach the tolerance.

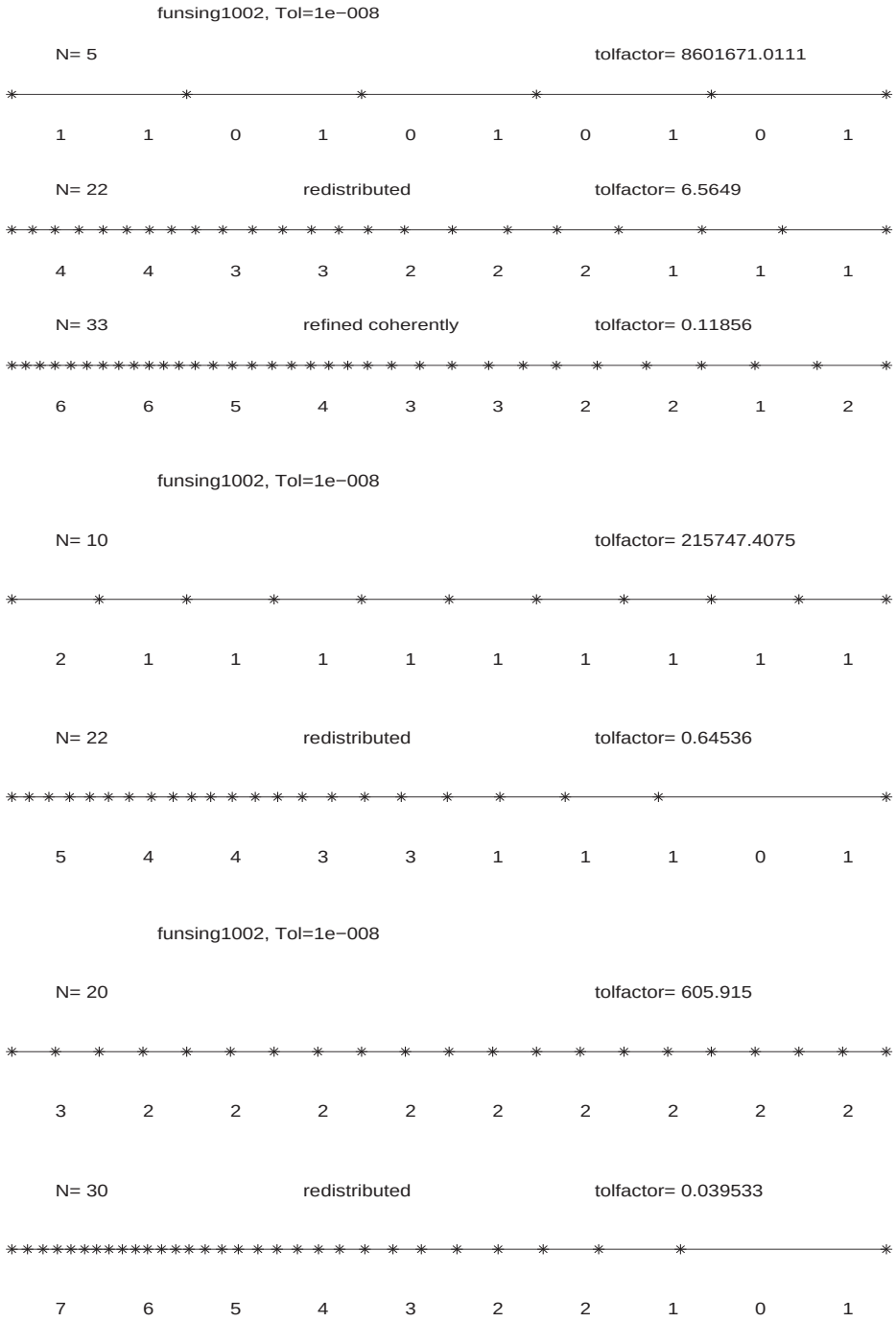


Figure 3.37: BVP 1002: Grid evolution for the strict tolerance 10^{-8}.

BVP	Tol	N_0	N	Time
funsing1004	10^{-3}	5	9	0.050
funsing1004	10^{-3}	10	10	0.030
funsing1004	10^{-3}	20	20	0.040
funsing1004	10^{-6}	5	12	0.060
funsing1004	10^{-6}	10	15	0.100
funsing1004	10^{-6}	20	20	0.050
funsing1004	10^{-8}	5	17	0.200
funsing1004	10^{-8}	10	15	0.080
funsing1004	10^{-8}	20	20	0.081

Table 3.103: Results for different starting grids. For the tolerance at 10^{-8} 5 points are far too few, so that the additional refinements take more time than the calculation for the higher number of meshpoints in the starting grid.

BVP	Tol	N_0	N	Time
funsing1005	10^{-3}	5	240	0.581
funsing1005	10^{-3}	10	240	0.591
funsing1005	10^{-3}	20	240	0.581
funsing1005	10^{-6}	5	120	0.411
funsing1005	10^{-6}	10	120	0.421
funsing1005	10^{-6}	20	120	0.411
funsing1005	10^{-8}	5	96	0.471
funsing1005	10^{-8}	10	96	0.460
funsing1005	10^{-8}	20	96	0.440

Table 3.104: Results for different starting grids. When more refinements are required to satisfy the tolerances, all three starting grids lead to comparable results.

BVP	Tol	N_0	N	Time
funsing015	10^{-3}	5	20	0.050
funsing015	10^{-3}	10	20	0.070
funsing015	10^{-3}	20	20	0.040
funsing015	10^{-6}	5	26	0.070
funsing015	10^{-6}	10	33	0.130
funsing015	10^{-6}	20	34	0.100
funsing015	10^{-8}	5	27	0.151
funsing015	10^{-8}	10	26	0.140
funsing015	10^{-8}	20	20	0.090

Table 3.105: Results for different starting grids. Since 20 points are necessary to satisfy the tolerance 10^{-3}, the starting grid, which already contains 20 points, performs best. Surprisingly, only at the tolerance set to 10^{-6} the starting grid with 20 points needs to be refined.

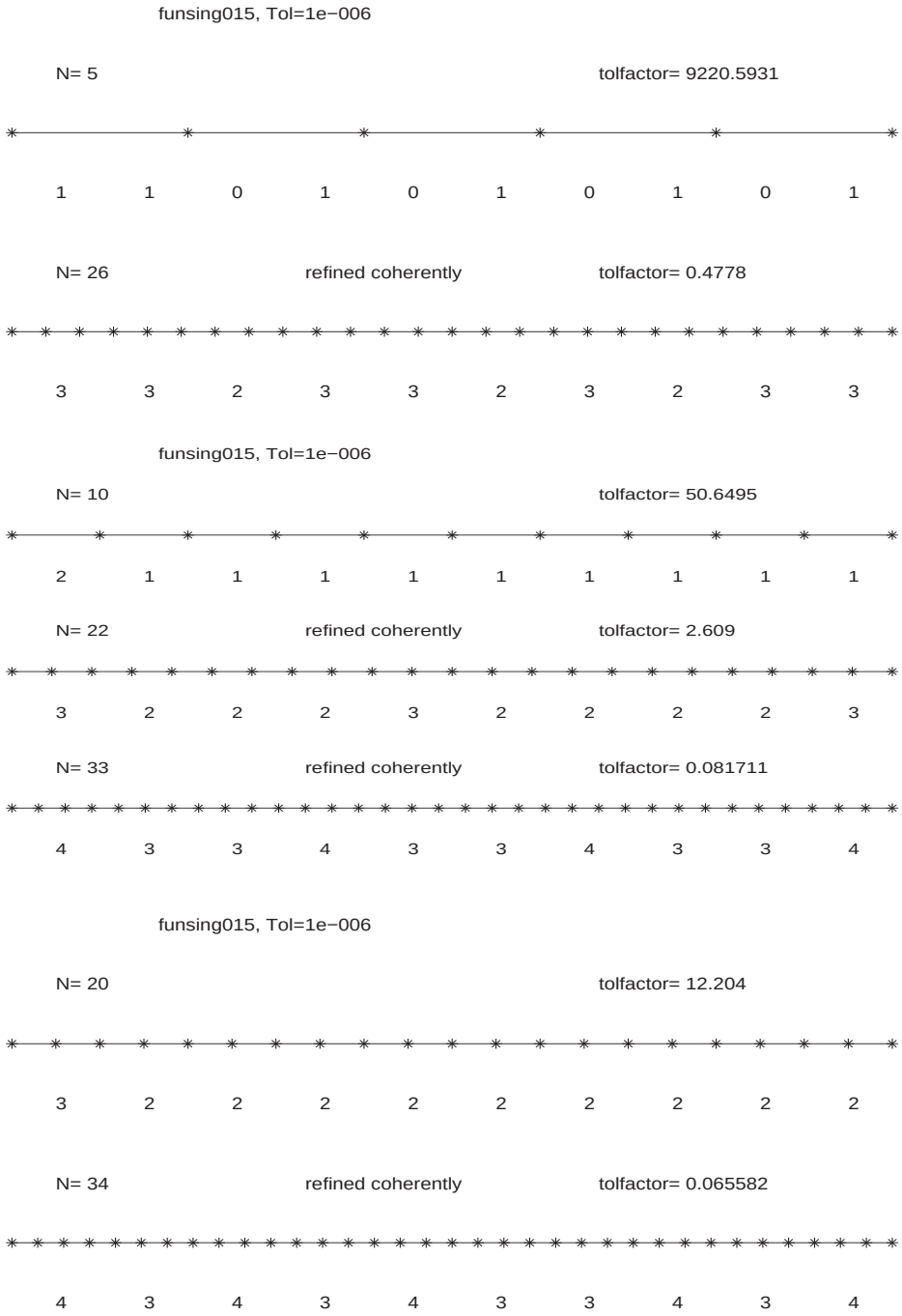


Figure 3.38: BVP 015: Grid evolution for the tolerance at 10^{-6}.

BVP	Tol	N_0	N	Time
funsing15	10^{-3}	5	33	0.130
funsing15	10^{-3}	10	44	0.150
funsing15	10^{-3}	20	45	0.140
funsing15	10^{-6}	5	33	0.150
funsing15	10^{-6}	10	33	0.131
funsing15	10^{-6}	20	35	0.111
funsing15	10^{-8}	5	21	0.120
funsing15	10^{-8}	10	26	0.140
funsing15	10^{-8}	20	20	0.081

Table 3.106: Results for different starting grids. Though the reduction in meshpoints is about 25 % for the mild tolerance for $N_0 = 5$, the differences in runtimes are not very big.

BVP	Tol	N_0	N	Time
funsing1a	10^{-3}	5	5	0.100
funsing1a	10^{-3}	10	10	0.100
funsing1a	10^{-3}	20	20	0.130
funsing1a	10^{-6}	5	5	0.080
funsing1a	10^{-6}	10	10	0.100
funsing1a	10^{-6}	20	20	0.130
funsing1a	10^{-8}	5	5	0.080
funsing1a	10^{-8}	10	10	0.100
funsing1a	10^{-8}	20	20	0.171

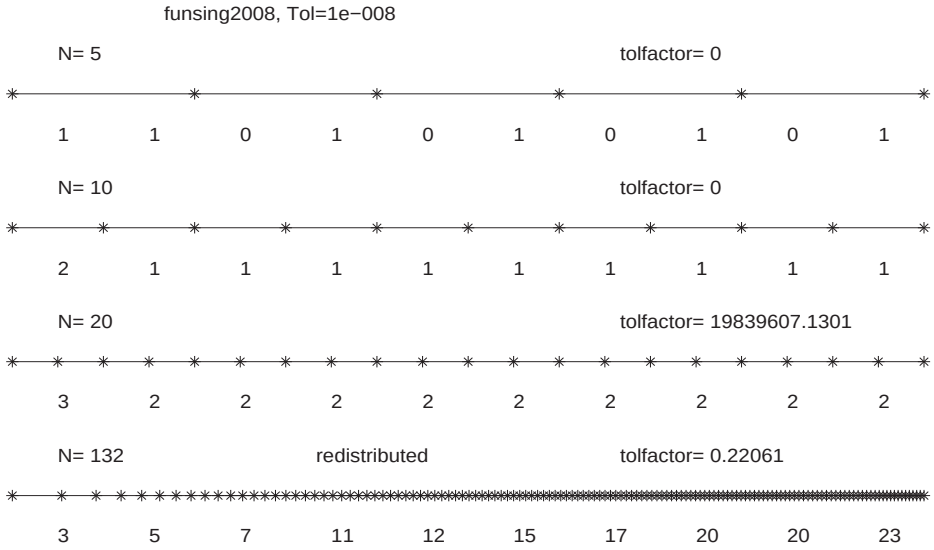
Table 3.107: Results for different starting grids. As the tolerances are satisfied on the starting grid even with 5 points, this is the best setting for the starting grid here.

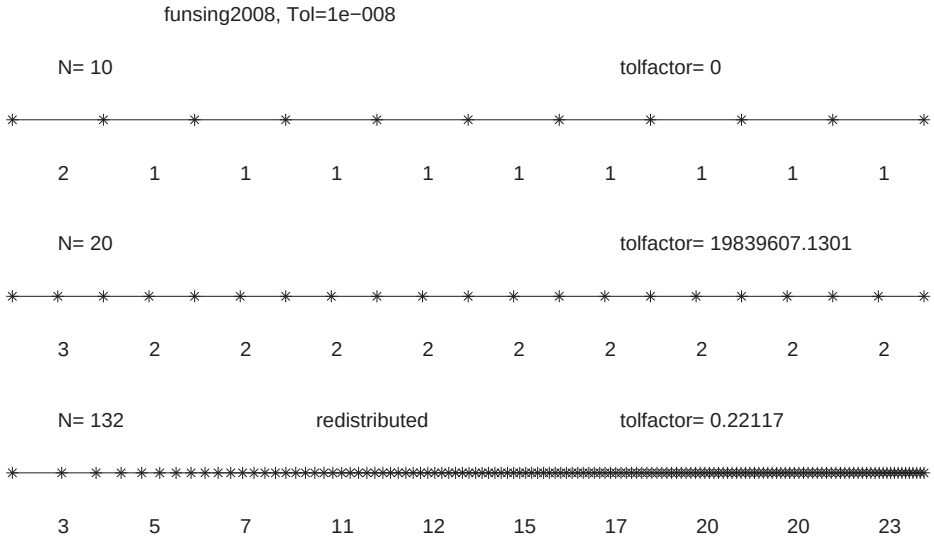
BVP	Tol	N_0	N	Time
funsing2002	10^{-3}	5	80	0.161
funsing2002	10^{-3}	10	80	0.180
funsing2002	10^{-3}	20	80	0.160
funsing2002	10^{-6}	5	75	0.240
funsing2002	10^{-6}	10	75	0.231
funsing2002	10^{-6}	20	62	0.210
funsing2002	10^{-8}	5	54	0.260
funsing2002	10^{-8}	10	54	0.261
funsing2002	10^{-8}	20	51	0.241

Table 3.108: Results for different starting grids. 20 meshpoints are favorable for the stricter tolerances.

Bvp	Tol	N_0	N	Time
funsing2008	10^{-3}	5	238	0.451
funsing2008	10^{-3}	10	238	0.411
funsing2008	10^{-3}	20	238	0.430
funsing2008	10^{-6}	5	158	0.560
funsing2008	10^{-6}	10	158	0.551
funsing2008	10^{-6}	20	158	0.510
funsing2008	10^{-8}	5	132	0.411
funsing2008	10^{-8}	10	132	0.410
funsing2008	10^{-8}	20	132	0.381

Table 3.109: Results for different starting grids. Again with 20 meshpoints the tolerances are satisfied faster on the same grid, which results from the smaller number of refinements, cf. Figures 3.39 - 3.41.





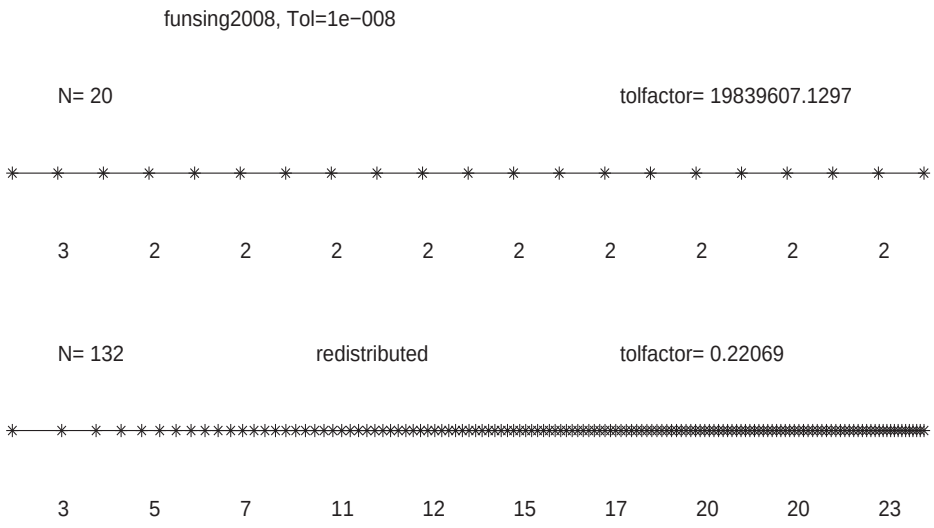


Figure 3.39: BVP 2008: Grid evolution for the tolerance at 10^{-8}.

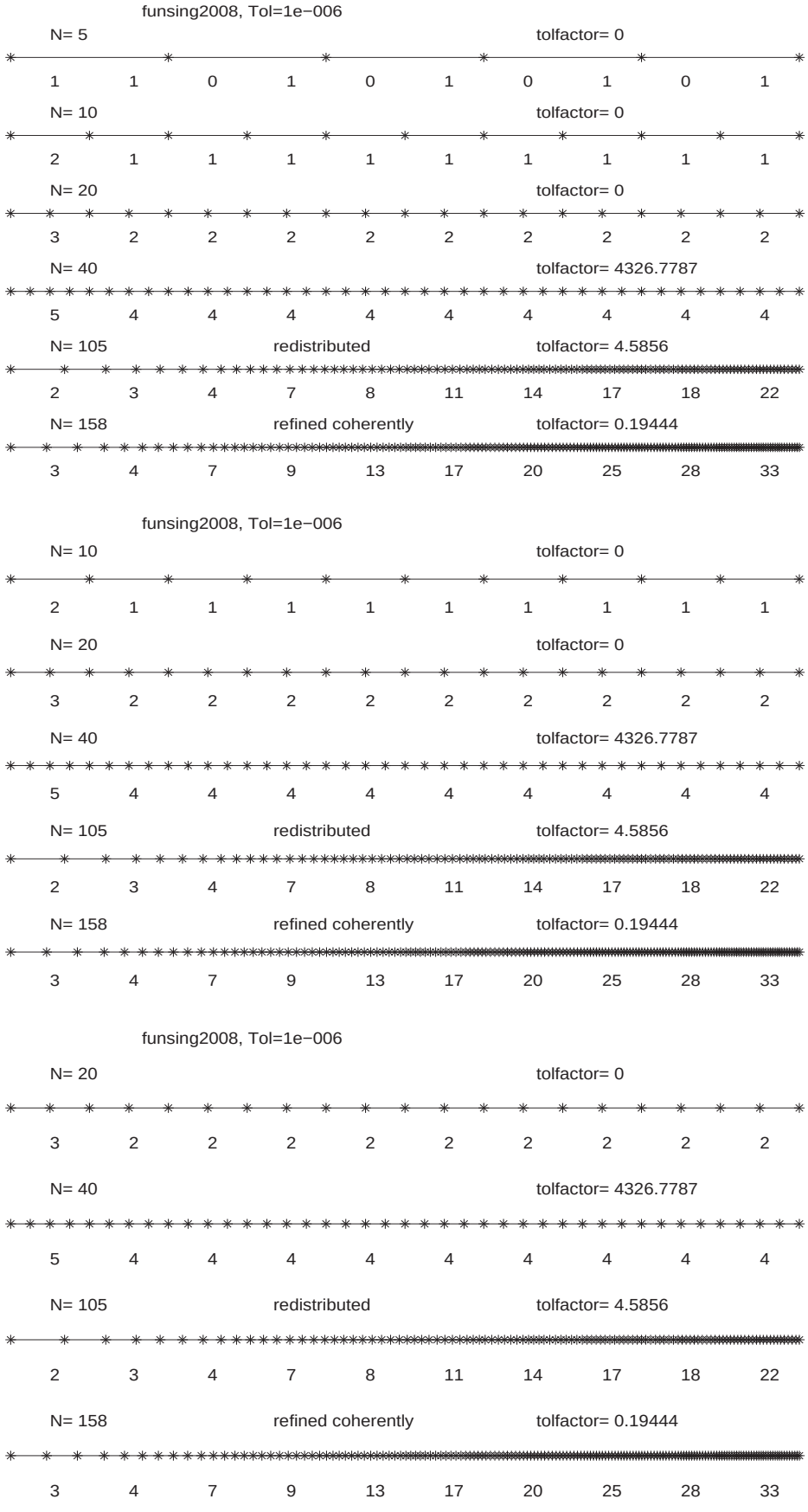


Figure 3.40: Bvp 2008: Grid evolution for the tolerance at 10^{-6}.

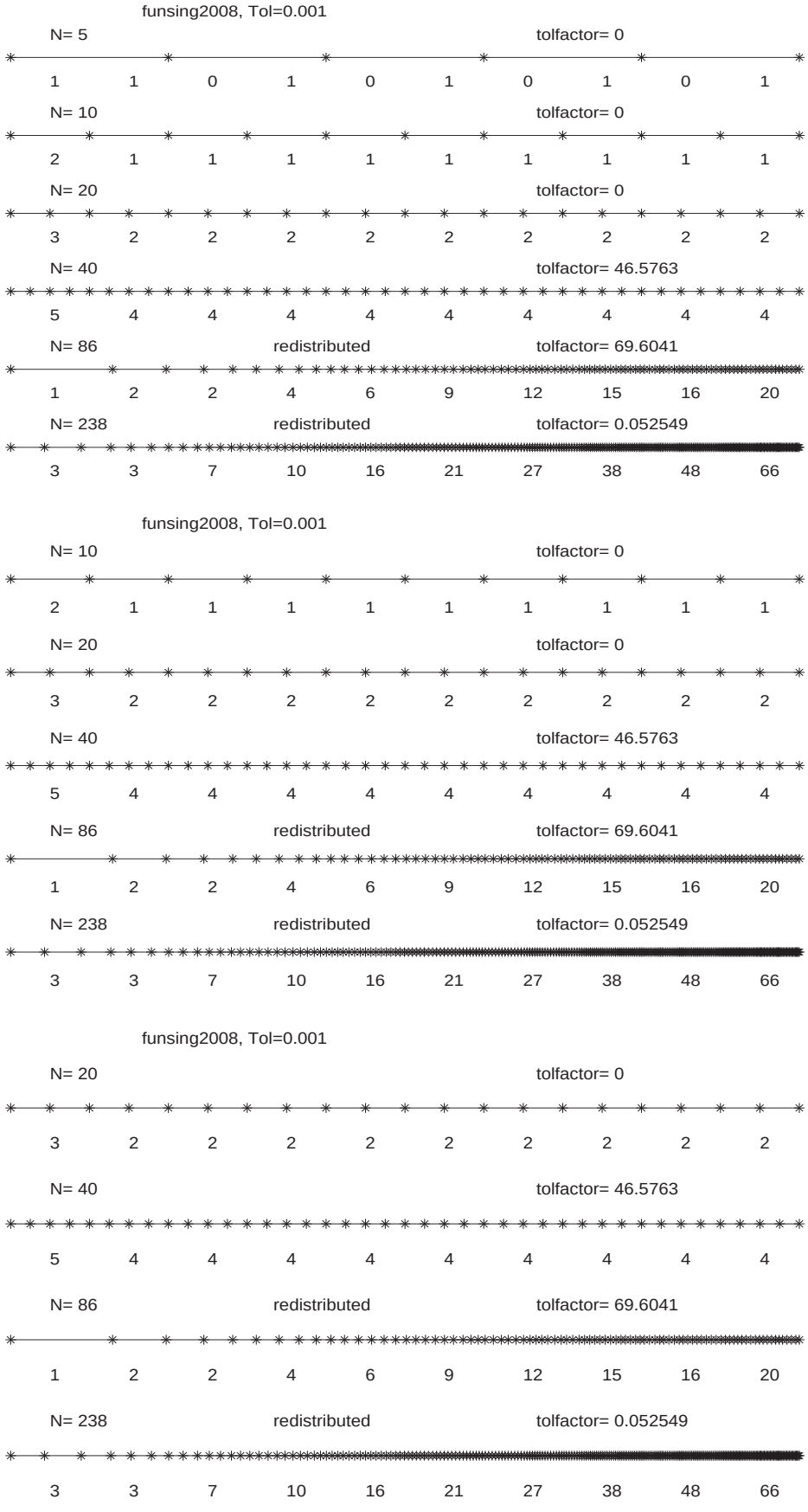


Figure 3.41: BVP 2008: Grid evolution for the tolerance at 10^{-3}.

BVP	Tol	N_0	N	Time
funsing21a	10^{-3}	5	5	0.070
funsing21a	10^{-3}	10	10	0.090
funsing21a	10^{-3}	20	20	0.120
funsing21a	10^{-6}	5	5	0.090
funsing21a	10^{-6}	10	10	0.110
funsing21a	10^{-6}	20	20	0.140
funsing21a	10^{-8}	5	5	0.080
funsing21a	10^{-8}	10	10	0.130
funsing21a	10^{-8}	20	20	0.150

Table 3.110: Results for different starting grids. Again the tolerances are satisfied already on the starting grids.

BVP	Tol	N_0	N	Time
funsing37c	10^{-3}	5	5	0.080
funsing37c	10^{-3}	10	10	0.091
funsing37c	10^{-3}	20	20	0.130
funsing37c	10^{-6}	5	5	0.080
funsing37c	10^{-6}	10	10	0.101
funsing37c	10^{-6}	20	20	0.140
funsing37c	10^{-8}	5	5	0.090
funsing37c	10^{-8}	10	10	0.140
funsing37c	10^{-8}	20	20	0.190

Table 3.111: Results for different starting grids. This is another BVP that is solved on the starting grid. 5 meshpoints are doing best, 10 meshpoints are not far behind but 20 meshpoints take twice as long for the tolerance at 10^{-8}.

BVP	Tol	N_0	N	Time
funsing400	10^{-3}	5	5	0.040
funsing400	10^{-3}	10	10	0.040
funsing400	10^{-3}	20	20	0.080
funsing400	10^{-6}	5	5	0.040
funsing400	10^{-6}	10	10	0.050
funsing400	10^{-6}	20	20	0.070
funsing400	10^{-8}	5	5	0.070
funsing400	10^{-8}	10	10	0.080
funsing400	10^{-8}	20	20	0.090

Table 3.112: Results for different starting grids. All the runtimes are very short.

BVP	Tol	N_0	N	Time
funsing500	10^{-3}	5	5	0.030
funsing500	10^{-3}	10	10	0.040
funsing500	10^{-3}	20	20	0.050
funsing500	10^{-6}	5	5	0.030
funsing500	10^{-6}	10	10	0.050
funsing500	10^{-6}	20	20	0.040
funsing500	10^{-8}	5	5	0.030
funsing500	10^{-8}	10	10	0.040
funsing500	10^{-8}	20	20	0.040

Table 3.113: Results for different starting grids. Again remarkably short runtimes for every setting.

BVP	Tol	N_0	N	Time
funsing54	10^{-3}	5	8	0.110
funsing54	10^{-3}	10	10	0.100
funsing54	10^{-3}	20	20	0.110
funsing54	10^{-6}	5	99	1.181
funsing54	10^{-6}	10	149	1.742
funsing54	10^{-6}	20	153	1.773
funsing54	10^{-8}	5	1115	42.671
funsing54	10^{-8}	10	1566	81.737
funsing54	10^{-8}	20	1337	59.215

Table 3.114: Results for different starting grids. This BVP is quite difficult to solve. 5 points are the best choice for the tolerance at 10^{-6} and even at 10^{-8}. Surprisingly 10 points do much worse than the other two configurations at the strictest tolerance.

BVP 54 is very interesting to look at. In the test for Mesh-Distribution and Risk-Premium turned out to be critical, see for example Table 3.37. Especially the chosen setting with Mesh-Distribution at All and Risk-Premium at $\frac{1}{10}$ showed bad results with a runtime of over 80 seconds. The starting grid was calculated automatically and it had 10 points. Now, we can see that exactly these 10 points give the worst result. With this insight the earlier bad result is partly explained, as this BVP is very sensitive with respect to the starting grid. For a better understanding what is happening, Figures 3.45 - 3.43 show how the grids evolved.

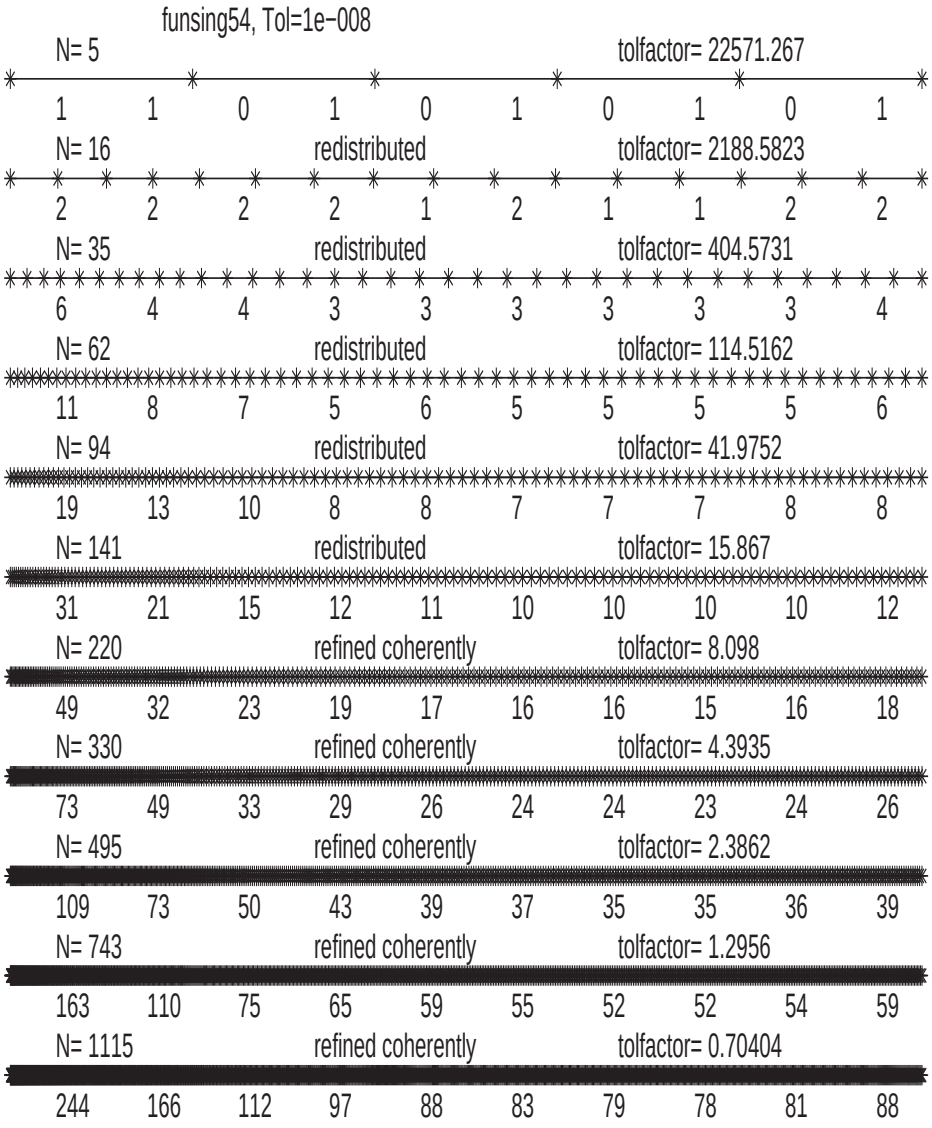


Figure 3.42: Bvp 54: Grid evolution for the tolerance at 10^{-8} starting with 5 points.

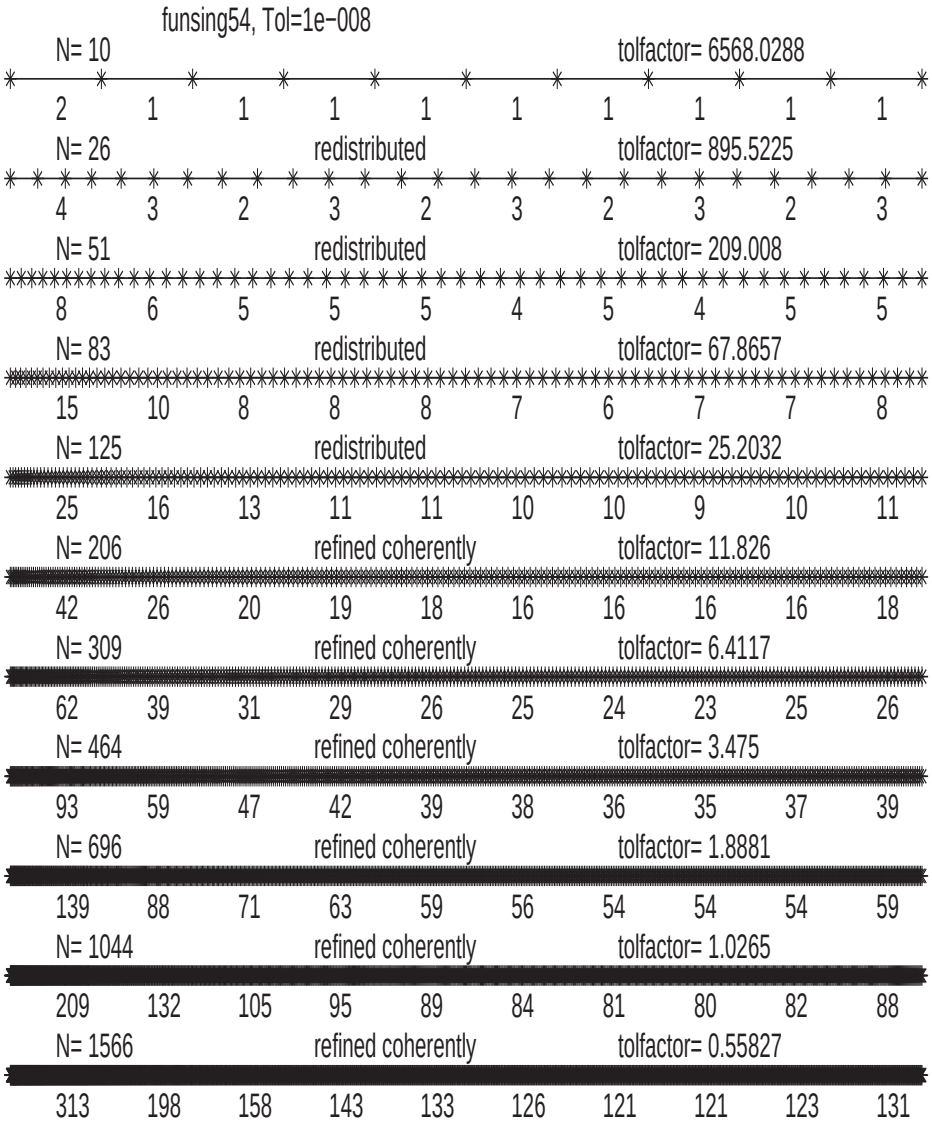


Figure 3.43: BVP 54: Grid evolution for the tolerance at 10^{-8} starting with 10 points.

Due to the slow convergence and the close failure of the tolerance with 1044 meshpoints (tolfactor=1.0265) the high runtime for 10 meshpoints on the starting grid is easily explained by the additional refinement which has a high price because of the huge number of meshpoints. This proves that the decision of not overrating the bad result of the combination of the Risk-Premium $\frac{1}{10}$ and the Mesh-Distribution All for BVP 54 at the tolerance 10^{-8} was right. It was simply 'bad luck' caused by the unusually slow convergence for this BVP.

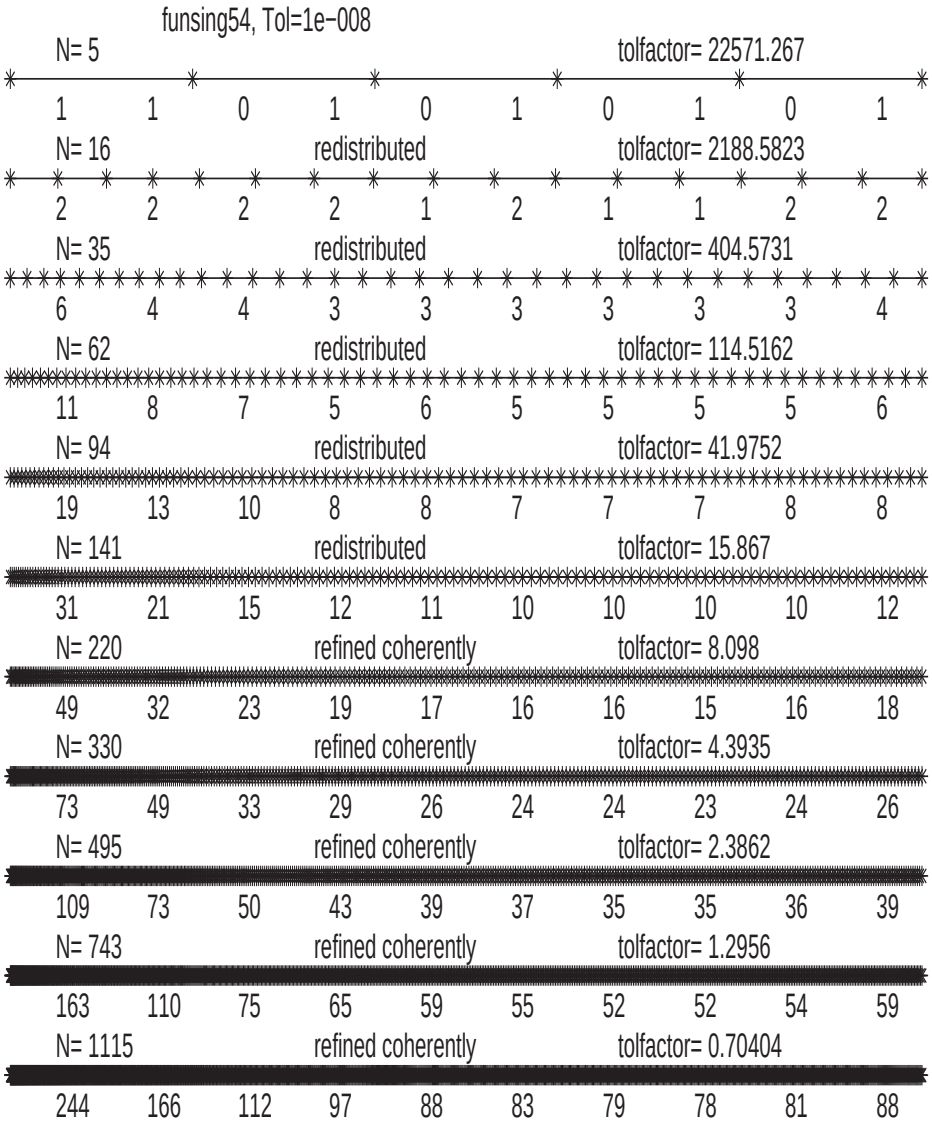


Figure 3.44: Bvp 54: Grid evolution for the tolerance at 10^{-8} starting with 20 points.

BVP	Tol	N_0	N	Time
funsing55	10^{-3}	5	5	0.070
funsing55	10^{-3}	10	10	0.070
funsing55	10^{-3}	20	20	0.090
funsing55	10^{-6}	5	5	0.060
funsing55	10^{-6}	10	10	0.070
funsing55	10^{-6}	20	20	0.140
funsing55	10^{-8}	5	5	0.070
funsing55	10^{-8}	10	10	0.120
funsing55	10^{-8}	20	20	0.180

Table 3.115: Results for different starting grids. On the starting grid the tolerances are satisfied. Therefore 5 meshpoints are favorable but the disadvantage of 10 meshpoints is nearly negligible.

BVP	Tol	N_0	N	Time
funsing56	10^{-3}	5	12	0.300
funsing56	10^{-3}	10	120	1.652
funsing56	10^{-3}	20	160	1.973
funsing56	10^{-6}	5	112	2.423
funsing56	10^{-6}	10	80	1.703
funsing56	10^{-6}	20	80	1.652
funsing56	10^{-8}	5	96	3.234
funsing56	10^{-8}	10	80	2.444
funsing56	10^{-8}	20	80	2.404

Table 3.116: Results for different starting grids. Excellent results for 5 meshpoints at the tolerance at 10^{-3} but the picture changes for stricter tolerances.

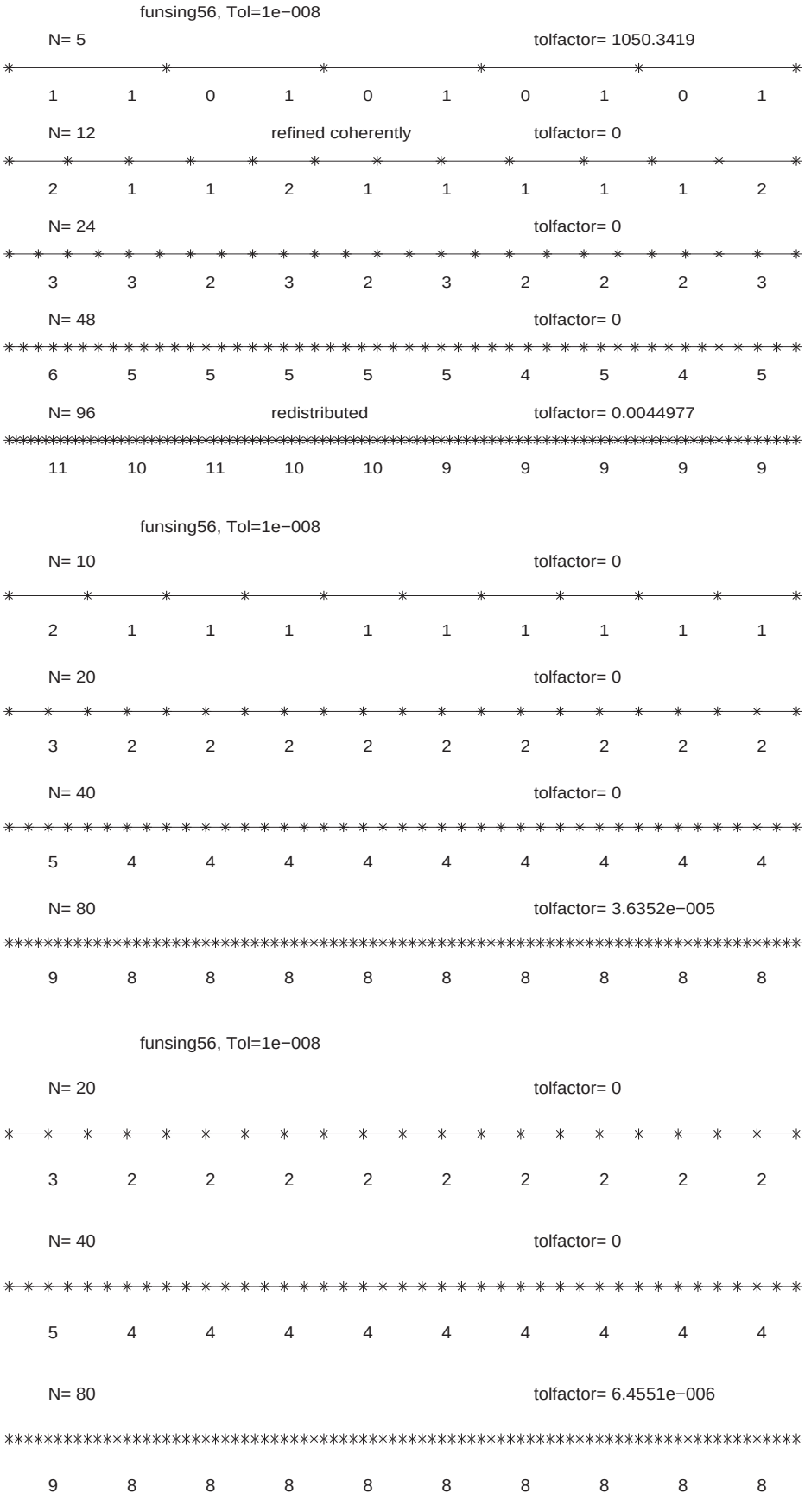


Figure 3.45: BVP 56: Grid evolution for the tolerance at 10^{-8}.

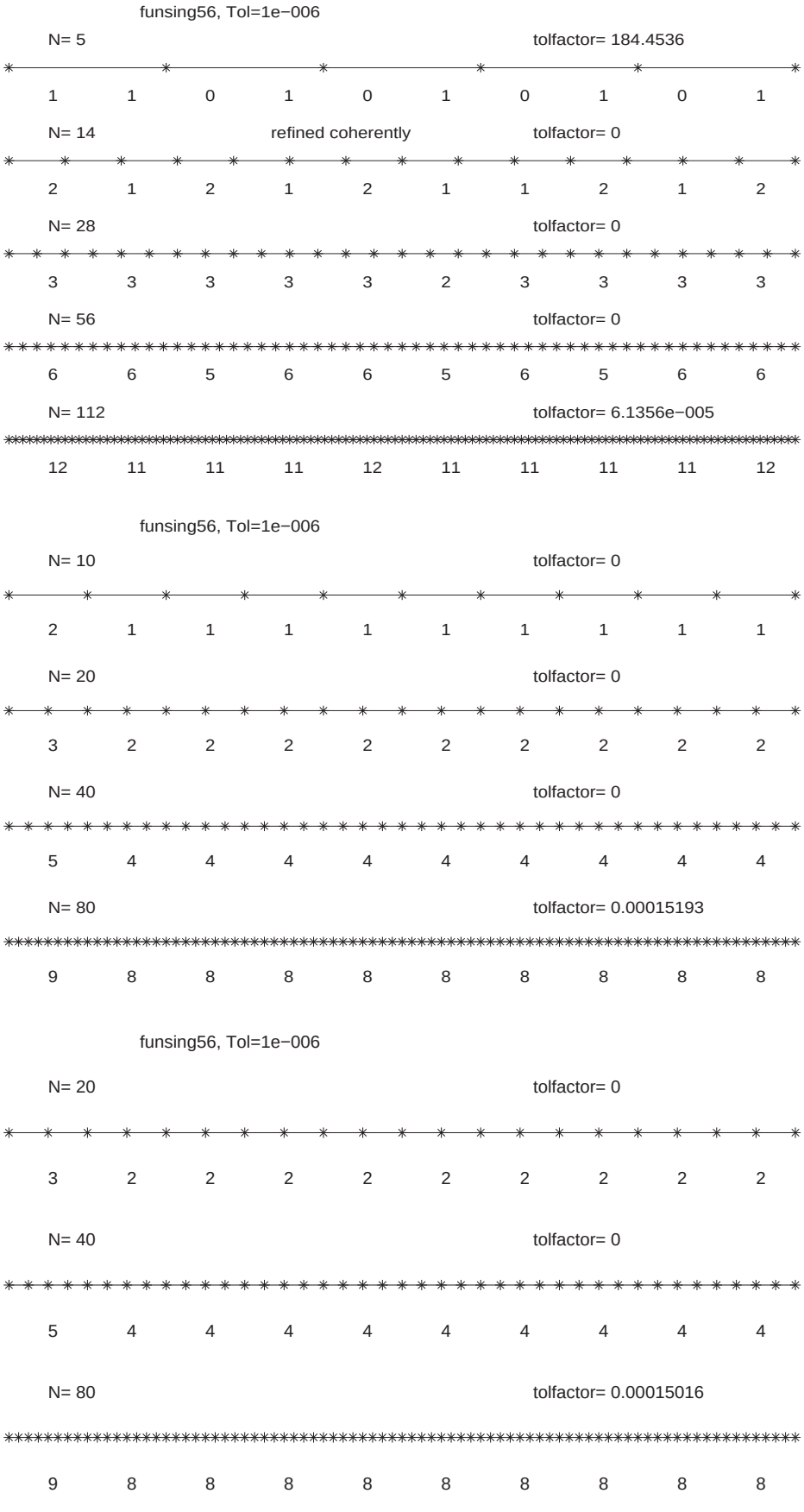


Figure 3.46: BVP 56: Grid evolution for the tolerance at 10^{-6}.

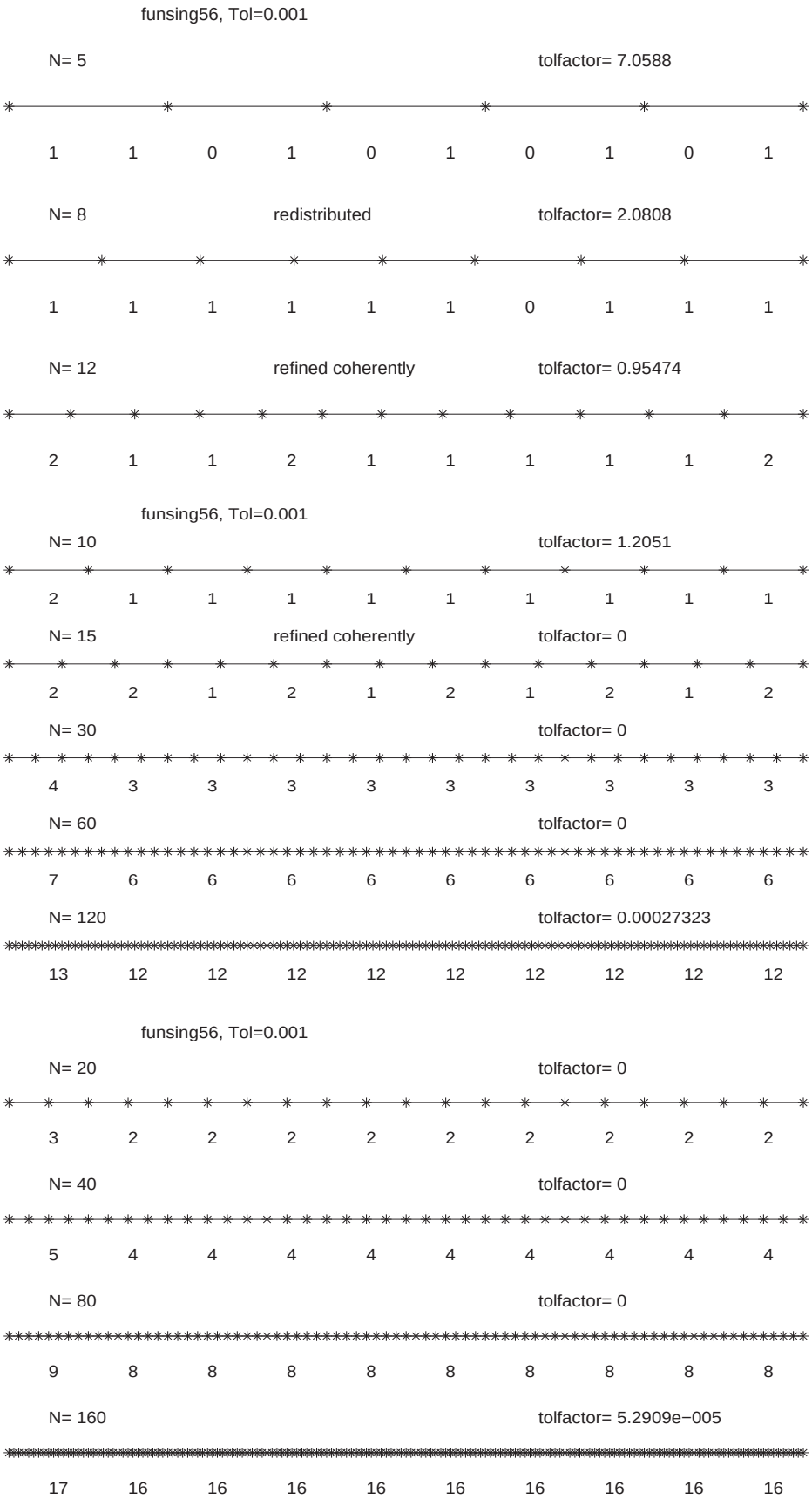


Figure 3.47: Bvp 56: Grid evolution for the tolerance at 10^{-3}.

The behavior of BVP 56 is not quite typical, because in nearly every configuration the error estimate is unreliable for some mesh. This is indicated by tolfactor 0 in the figures above. Nevertheless five points are quite risky here: Though they are favorable for tolerance 10^{-3} the results are worst for tolerances 10^{-6} and 10^{-8}. With tolerance 10^{-3} the starting grid with 5 meshpoints never runs into a failure of the error estimation and the problem is solved on an early mesh.

BVP	Tol	N_0	N	Time
funsing6001	10^{-3}	5	8	0.140
funsing6001	10^{-3}	10	10	0.100
funsing6001	10^{-3}	20	20	0.120
funsing6001	10^{-6}	5	11	0.181
funsing6001	10^{-6}	10	15	0.180
funsing6001	10^{-6}	20	20	0.170
funsing6001	10^{-8}	5	14	0.230
funsing6001	10^{-8}	10	15	0.261
funsing6001	10^{-8}	20	20	0.230

Table 3.117: Results for different starting grids. All three settings are quite competitive here.

BVP	Tol	N_0	N	Time
funsing6002	10^{-3}	5	5	0.070
funsing6002	10^{-3}	10	10	0.081
funsing6002	10^{-3}	20	20	0.100
funsing6002	10^{-6}	5	5	0.060
funsing6002	10^{-6}	10	10	0.091
funsing6002	10^{-6}	20	20	0.110
funsing6002	10^{-8}	5	5	0.070
funsing6002	10^{-8}	10	10	0.090
funsing6002	10^{-8}	20	20	0.151

Table 3.118: Results for different starting grids. Again we see that 10 points are a very competitive choice when no refinement is needed, whereas 20 meshpoints take about twice the time in the hardest case.

Bvp	Tol	N_0	N	Time
funsing7001b	10^{-3}	5	5	0.070
funsing7001b	10^{-3}	10	10	0.070
funsing7001b	10^{-3}	20	20	0.110
funsing7001b	10^{-6}	5	5	0.070
funsing7001b	10^{-6}	10	10	0.100
funsing7001b	10^{-6}	20	20	0.130
funsing7001b	10^{-8}	5	5	0.090
funsing7001b	10^{-8}	10	10	0.110
funsing7001b	10^{-8}	20	20	0.170

Table 3.119: Results for different starting grids. 10 meshpoints are very competitive as compared to 5 meshpoints, 20 points do not seem favorable here.

Bvp	Tol	N_0	N	Time
funsing71	10^{-3}	5	14	0.130
funsing71	10^{-3}	10	15	0.130
funsing71	10^{-3}	20	20	0.120
funsing71	10^{-6}	5	41	0.341
funsing71	10^{-6}	10	28	0.210
funsing71	10^{-6}	20	30	0.241
funsing71	10^{-8}	5	43	0.420
funsing71	10^{-8}	10	45	0.441
funsing71	10^{-8}	20	33	0.321

Table 3.120: Results for different starting grids. Unsurprisingly 20 meshpoints are a good choice when quite a low number of mesh refinements is needed.

Bvp	Tol	N_0	N	Time
funsing73	10^{-3}	5	240	4.887
funsing73	10^{-3}	10	240	4.747
funsing73	10^{-3}	20	240	4.656
funsing73	10^{-6}	5	240	9.103
funsing73	10^{-6}	10	240	8.983
funsing73	10^{-6}	20	240	8.852
funsing73	10^{-8}	5	120	6.329
funsing73	10^{-8}	10	120	6.209
funsing73	10^{-8}	20	120	5.928

Table 3.121: Results for different starting grids. This four-dimensional regular Bvp shows an advantage for 20 meshpoints, followed by 10 meshpoints rather closely. The starting grid with 5 meshpoints is the worst choice for all three tolerances. The reason are additional refinements required, cf. Figure 3.48.

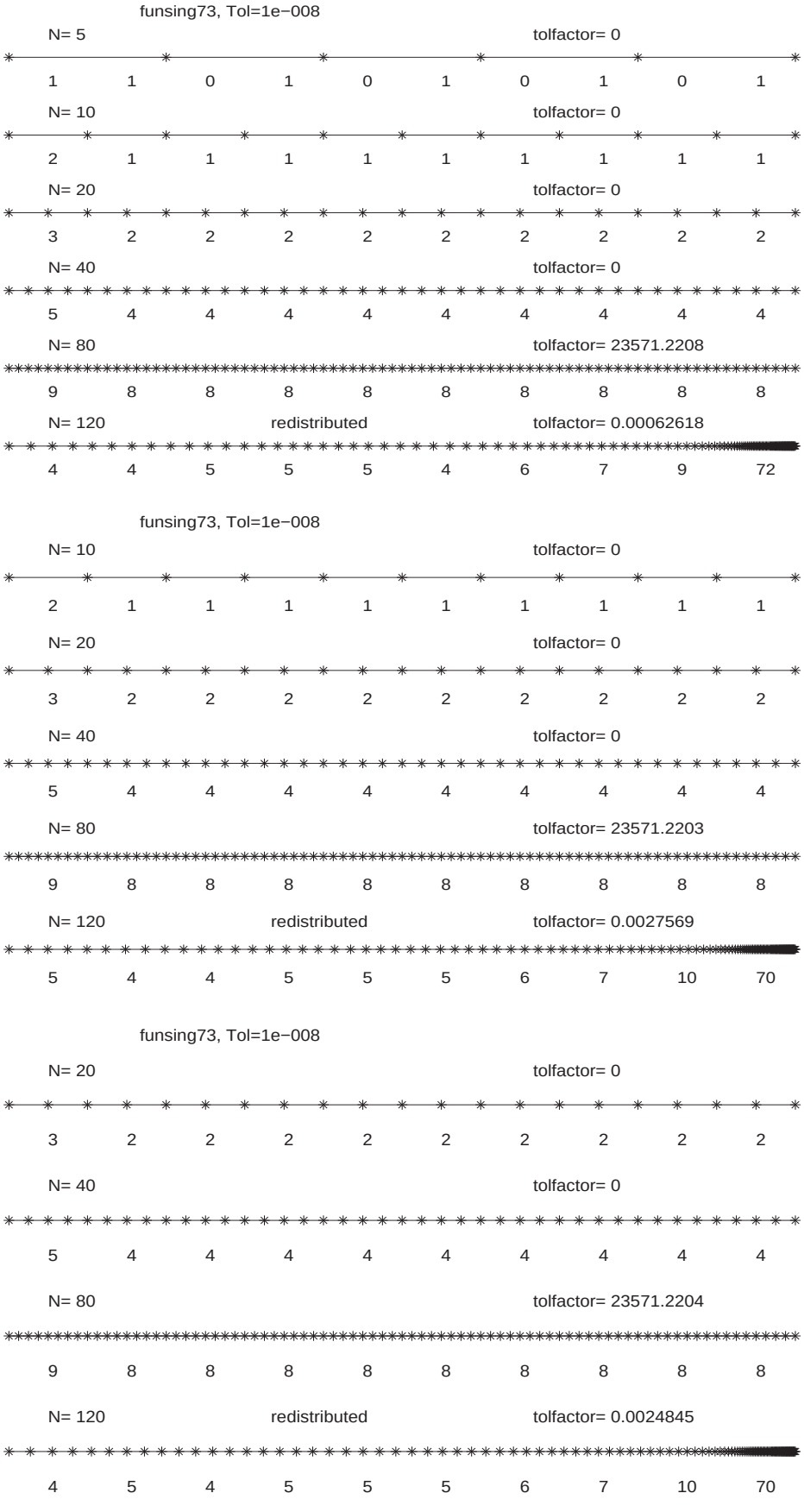


Figure 3.48: BVP 73: Grid evolution for the tolerance at 10^{-8}.

Bvp	Tol	N_0	N	Time
funsing8001	10^{-3}	5	5	0.080
funsing8001	10^{-3}	10	10	0.091
funsing8001	10^{-3}	20	20	0.120
funsing8001	10^{-6}	5	5	0.080
funsing8001	10^{-6}	10	10	0.100
funsing8001	10^{-6}	20	20	0.150
funsing8001	10^{-8}	5	8	0.141
funsing8001	10^{-8}	10	10	0.120
funsing8001	10^{-8}	20	20	0.200

Table 3.122: Results for different starting grids. For the tolerance 10^{-8} it is not surprising that 5 meshpoints are a bad choice because an additional refinement is needed, whereas 10 points satisfy the tolerance. The starting grid with 20 meshpoints performs worst for every single tolerance.

Bvp	Tol	N_0	N	Time
funsing8002	10^{-3}	5	8	0.140
funsing8002	10^{-3}	10	10	0.100
funsing8002	10^{-3}	20	20	0.130
funsing8002	10^{-6}	5	17	0.240
funsing8002	10^{-6}	10	15	0.190
funsing8002	10^{-6}	20	30	0.260
funsing8002	10^{-8}	5	12	0.221
funsing8002	10^{-8}	10	15	0.231
funsing8002	10^{-8}	20	30	0.331

Table 3.123: Results for different starting grids. Very good behavior with 10 points, because less than 20 meshpoints are needed to satisfy the tolerances. 5 meshpoints are not enough to have competitive results except for the case with the strict tolerance.

BVP	Tol	N_0	N	Time
funsing9001	10^{-3}	5	107	0.261
funsing9001	10^{-3}	10	107	0.260
funsing9001	10^{-3}	20	80	0.190
funsing9001	10^{-6}	5	81	0.250
funsing9001	10^{-6}	10	81	0.240
funsing9001	10^{-6}	20	87	0.260
funsing9001	10^{-8}	5	56	0.220
funsing9001	10^{-8}	10	45	0.140
funsing9001	10^{-8}	20	63	0.281

Table 3.124: Results for different starting grids. Since a considerable number of meshpoints is required to satisfy the tolerances, it is not surprising that a starting grid with 5 meshpoints is not the best choice here. The strikingly good result for 10 meshpoints in the strictest case has to be noticed.

This BVP is a good example to show why 5 meshpoints are not enough and 20 meshpoints are too many, see Figure 3.49.

It is easy to see, that with 10 meshpoints only one refinement is needed to satisfy the tolerance. The reason is that the tolfactor is smaller than in the case when starting with 5 meshpoints. The routine calculates a very good error estimate and the tolerance is satisfied at once. With 5 meshpoints, the number of meshpoints used in the refinement is underestimated and the tolerance is missed. The next refinement yields a highly accurate solution that requires a long runtime and large number of meshpoints. The starting grid with 20 meshpoints shows an overoptimistic result with the tolfactor at only 168.5926. Therefore, the first refinement is too coarse. Finally, the last refinement leads to a very accurate solution whose computation is time consuming, however.

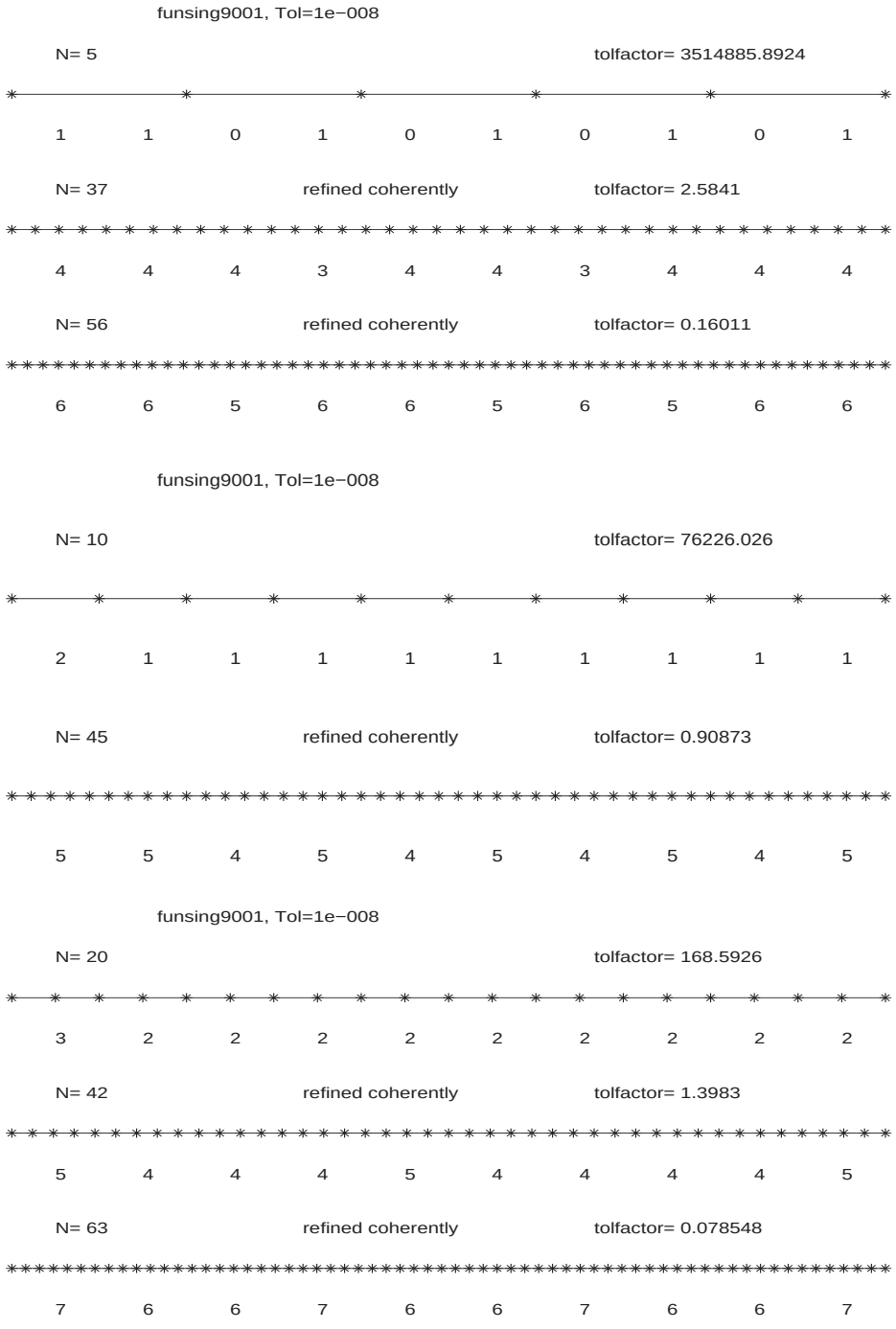


Figure 3.49: BVP 9001: Grid evolution for the tolerance at 10^{-8}.

BVP	Tol	N_0	N	Time
funsing9002	10^{-3}	5	147	0.300
funsing9002	10^{-3}	10	147	0.290
funsing9002	10^{-3}	20	147	0.280
funsing9002	10^{-6}	5	92	0.271
funsing9002	10^{-6}	10	92	0.250
funsing9002	10^{-6}	20	97	0.211
funsing9002	10^{-8}	5	53	0.150
funsing9002	10^{-8}	10	69	0.281
funsing9002	10^{-8}	20	68	0.210

Table 3.125: Results for different starting grids. Figure 3.50 suggests that the additional refinement for the starting grid at 10 points is quite costly.

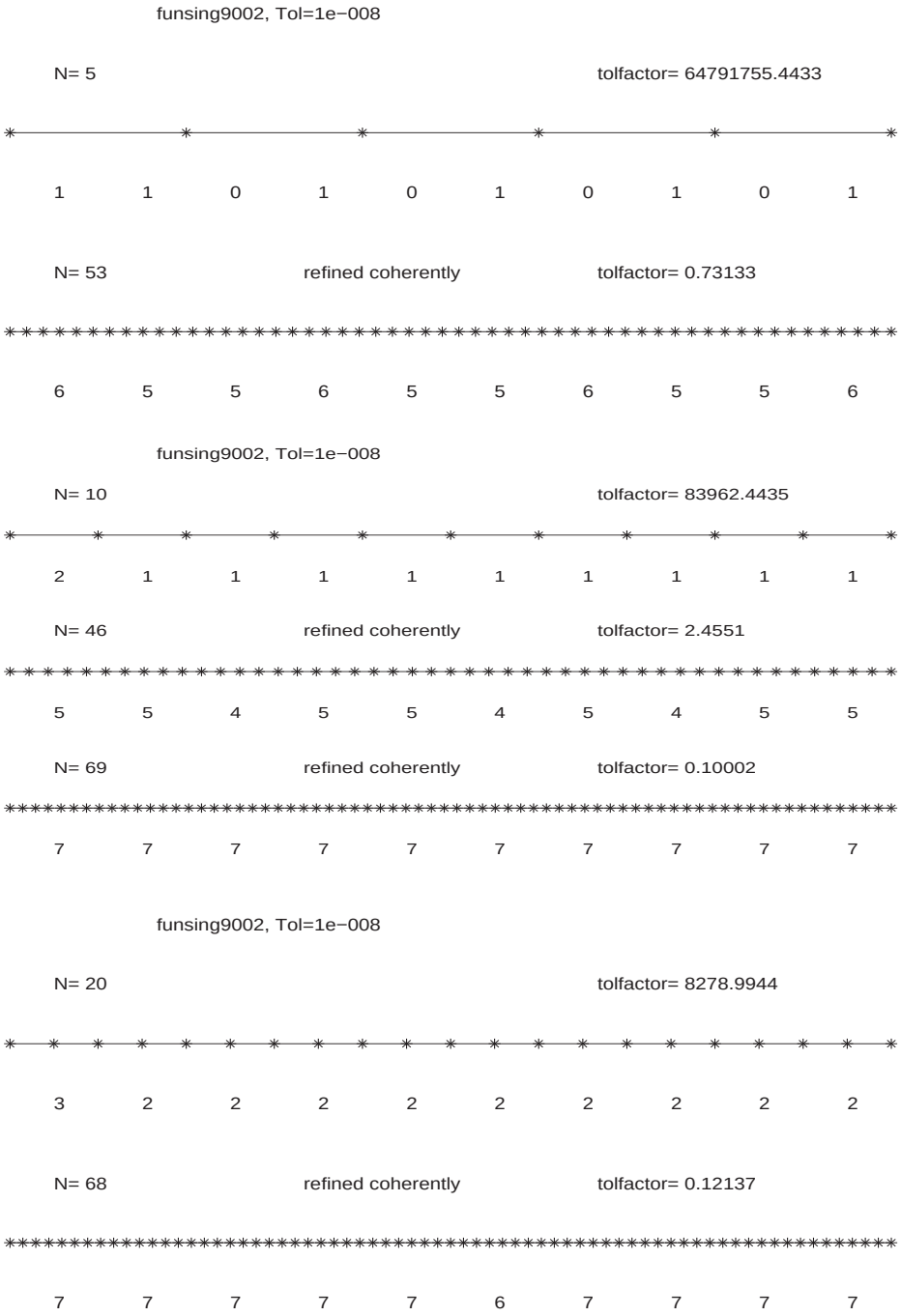


Figure 3.50: BVP 9002: Grid evolution for the tolerance at 10^{-8}.

BVP	Tol	N_0	N	Time
funsing9003	10^{-3}	5	10	0.050
funsing9003	10^{-3}	10	10	0.041
funsing9003	10^{-3}	20	20	0.040
funsing9003	10^{-6}	5	30	0.110
funsing9003	10^{-6}	10	27	0.120
funsing9003	10^{-6}	20	20	0.060
funsing9003	10^{-8}	5	20	0.130
funsing9003	10^{-8}	10	16	0.100
funsing9003	10^{-8}	20	20	0.060

Table 3.126: Results for different starting grids. This BVP is solved very quickly and because 20 points are enough to satisfy the tolerances, this case is the best choice for N_0 here.

The results of this section do not show a clear picture, which number to choose in the starting grid. In fact, and this is not very surprising, it strongly depends on the problem to solve. What can be seen is that in the cases where 5 points satisfy the requested tolerance, 10 points do not have a big loss in time, whereas the choice with 20 points is losing at the higher tolerances.

Due to this observation and the fact that 5 points are a greater risk for failing to recognize the characteristics of the problem early, it is a reasonable decision to set the lower limit to 10 points. The possible increase in runtime on the starting grid is negligible and the gain in stability and reliability of the routine certainly compensates for this. On the other hand 20 meshpoints even yield good results, but for the stricter tolerance they are definitely slower than in the case when starting with 10 meshpoints. The big advantage for 20 points is given in the case of only a small number of mesh refinements. The more refinements the smaller is the advantage of 20 points compared to starting mesh with only 10 points.

3.4 Altering Degrees

The tests above do not show too many details of how automatic mesh determination works. Therefore we conduct further tests where the degree of the collocation polynomial is chosen manually. The automatic selection of the initial mesh is compared with the cases with 10 and 20 meshpoints to reach a final decision about the starting grid.

Bvp	Tol	Order	N_0	N	Time
funsing0026	10^{-6}	4	10	10	0.060
funsing0026	10^{-6}	4	20	20	0.100
funsing0026	10^{-6}	4	31	31	0.100
funsing0026	10^{-8}	4	10	27	0.140
funsing0026	10^{-8}	4	20	30	0.140
funsing0026	10^{-8}	4	100	100	0.260
funsing0026	10^{-8}	6	10	10	0.080
funsing0026	10^{-8}	6	20	20	0.120
funsing0026	10^{-8}	6	21	21	0.130

Table 3.127: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for the automatic determination of the starting grid. As this Bvp is mostly successfully solved on the starting grid, automatic determination is not favorable, especially in the case with degree 4 and the tolerance 10^{-8}, where $N_0 = 100$.

Bvp	Tol	Order	N_0	N	Time
funsing1001	10^{-6}	4	10	972	3.165
funsing1001	10^{-6}	4	20	972	3.154
funsing1001	10^{-6}	4	31	744	1.802
funsing1001	10^{-8}	4	10	1020	2.674
funsing1001	10^{-8}	4	20	1020	2.674
funsing1001	10^{-8}	4	100	868	2.163
funsing1001	10^{-8}	6	10	480	1.783
funsing1001	10^{-8}	6	20	480	1.773
funsing1001	10^{-8}	6	21	504	1.892

Table 3.128: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. To solve this Bvp a very fine mesh is required, which generally favors automatic determination of the starting grid.

BVP	Tol	Order	N_0	N	Time
funsing1002	10^{-6}	4	10	111	0.200
funsing1002	10^{-6}	4	20	95	0.180
funsing1002	10^{-6}	4	31	87	0.180
funsing1002	10^{-8}	4	10	350	0.641
funsing1002	10^{-8}	4	20	197	0.241
funsing1002	10^{-8}	4	100	272	0.571
funsing1002	10^{-8}	6	10	48	0.120
funsing1002	10^{-8}	6	20	62	0.191
funsing1002	10^{-8}	6	21	62	0.200

Table 3.129: Results for different starting grids using manual degree selection. The test was performed for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Here the results show an advantage for 20 points in the case of the strict tolerance and collocation order 4, which is surprising because the starting grid at 100 points should be more advantageous.

With 10 meshpoints in the starting grid and with 100 meshpoints in the starting grid the tolerance is slightly failed and another refinement is required, cf. Figure 3.51. This behavior is similar to BVP 1002 in the tests from Section 3.3, refer to Figure 3.51. So it seems that the routine which determines how many meshpoints are needed works best if the tolfactor is in the range 10^5 - 10^6.

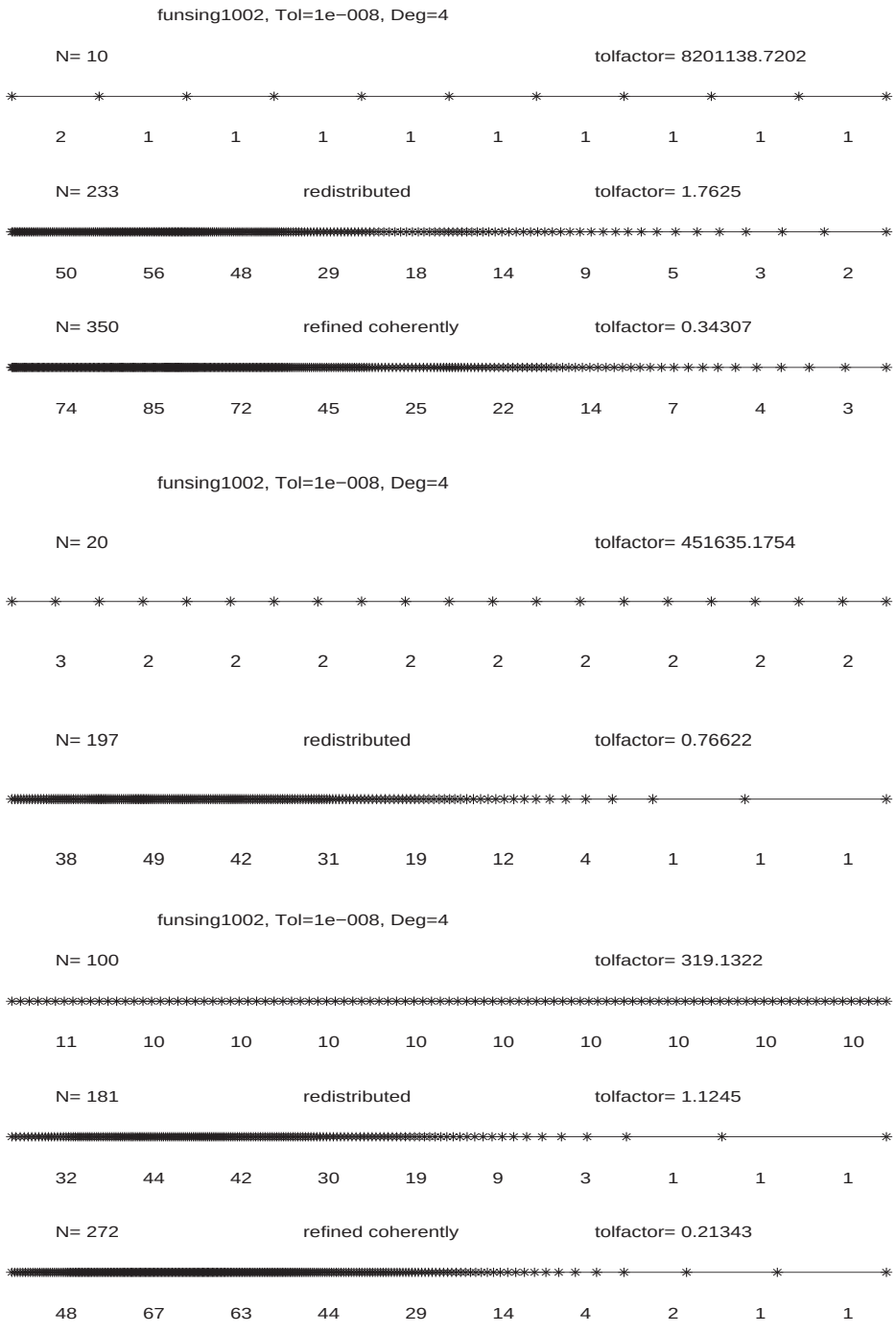


Figure 3.51: Bvp 1002: Grid evolution for the tolerance at 10^{-8} and collocation order 4.

BVP	Tol	Order	N_0	N	Time
funsing1004	10^{-6}	4	10	51	0.120
funsing1004	10^{-6}	4	20	53	0.130
funsing1004	10^{-6}	4	31	52	0.100
funsing1004	10^{-8}	4	10	162	0.290
funsing1004	10^{-8}	4	20	111	0.160
funsing1004	10^{-8}	4	100	159	0.271
funsing1004	10^{-8}	6	10	35	0.121
funsing1004	10^{-8}	6	20	33	0.100
funsing1004	10^{-8}	6	21	32	0.100

Table 3.130: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Here all three settings perform satisfactorily, especially when starting with 20 meshpoints and polynomial degree 4 at the tolerance at 10^{-8} but automatic determination is favorable for degree 4 at the tolerance 10^{-6}.

BVP	Tol	Order	N_0	N	Time
funsing1005	10^{-6}	4	10	240	0.551
funsing1005	10^{-6}	4	20	240	0.530
funsing1005	10^{-6}	4	31	186	0.411
funsing1005	10^{-8}	4	10	282	0.611
funsing1005	10^{-8}	4	20	282	0.581
funsing1005	10^{-8}	4	100	309	0.661
funsing1005	10^{-8}	6	10	120	0.390
funsing1005	10^{-8}	6	20	120	0.391
funsing1005	10^{-8}	6	21	126	0.401

Table 3.131: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. As seen before 20 meshpoints yield best results in the case of polynomial degree 4 and tolerance 10^{-8}.

Bvp	Tol	Order	N_0	N	Time
funsing015	10^{-6}	4	10	111	0.141
funsing015	10^{-6}	4	20	121	0.181
funsing015	10^{-6}	4	31	130	0.180
funsing015	10^{-8}	4	10	349	0.430
funsing015	10^{-8}	4	20	380	0.501
funsing015	10^{-8}	4	100	297	0.451
funsing015	10^{-8}	6	10	46	0.110
funsing015	10^{-8}	6	20	72	0.160
funsing015	10^{-8}	6	21	65	0.140

Table 3.132: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. This Bvp shows that 20 meshpoints in the starting grid are not always the best choice. Here this configuration is slowest for every configuration.

Bvp	Tol	Order	N_0	N	Time
funsing15	10^{-6}	4	10	158	0.280
funsing15	10^{-6}	4	20	154	0.190
funsing15	10^{-6}	4	31	191	0.240
funsing15	10^{-8}	4	10	330	0.400
funsing15	10^{-8}	4	20	485	0.671
funsing15	10^{-8}	4	100	387	0.560
funsing15	10^{-8}	6	10	47	0.110
funsing15	10^{-8}	6	20	76	0.160
funsing15	10^{-8}	6	21	67	0.150

Table 3.133: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Automatic determination shows an average performance for all three settings of tolerance and collocation order, while the behavior of the other choices varies for different configurations.

BVP	Tol	Order	N_0	N	Time
funsing1a	10^{-6}	4	10	10	0.090
funsing1a	10^{-6}	4	20	20	0.090
funsing1a	10^{-6}	4	31	31	0.140
funsing1a	10^{-8}	4	10	19	0.140
funsing1a	10^{-8}	4	20	20	0.111
funsing1a	10^{-8}	4	100	100	0.340
funsing1a	10^{-8}	6	10	10	0.080
funsing1a	10^{-8}	6	20	20	0.120
funsing1a	10^{-8}	6	21	21	0.121

Table 3.134: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Automatic determination is competitive in the last case but very slow in the other two cases. This is an example where few meshpoints are required to satisfy the tolerances.

BVP	Tol	Order	N_0	N	Time
funsing2002	10^{-6}	4	10	183	0.220
funsing2002	10^{-6}	4	20	183	0.210
funsing2002	10^{-6}	4	31	188	0.220
funsing2002	10^{-8}	4	10	578	0.841
funsing2002	10^{-8}	4	20	578	0.821
funsing2002	10^{-8}	4	100	603	0.951
funsing2002	10^{-8}	6	10	159	0.420
funsing2002	10^{-8}	6	20	131	0.351
funsing2002	10^{-8}	6	21	129	0.381

Table 3.135: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Once again when starting with 20 meshpoints the tolerance 10^{-8} is satisfied rather quickly with the collocation method of order 4.

Bvp	Tol	Order	N_0	N	Time
funsing2008	10^{-6}	4	10	723	1.782
funsing2008	10^{-6}	4	20	723	1.783
funsing2008	10^{-6}	4	31	783	1.963
funsing2008	10^{-8}	4	10	2286	12.117
funsing2008	10^{-8}	4	20	2286	12.108
funsing2008	10^{-8}	4	100	1788	5.077
funsing2008	10^{-8}	6	10	338	1.172
funsing2008	10^{-8}	6	20	338	1.152
funsing2008	10^{-8}	6	21	344	1.191

Table 3.136: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Here the runtimes, when trying to satisfy the tolerance of 10^{-8} at collocation order 4, are higher and automatic determination shows best performance.

From Figure 3.52 it can be seen that the tolfactor is within the range where the determination of the next mesh refinement is favorable. This corresponds to the results that have been encountered earlier, cf. Section 3.3.

Note that the starting grid with 100 meshpoints does not run into the problem of the unreliable error estimate. This can be observed in Figure 3.52 when looking at the first refinements. The tolfactor remains undefined, hence the number of meshpoints is doubled. Thus the results for starting with 10 and 20 meshpoints are quite the same. The same happens for the tolerance 10^{-8} with collocation order 6. But here even automatic determination leads to unreliable error estimation, caused by the fact that there is only one additional meshpoint in the starting grid.

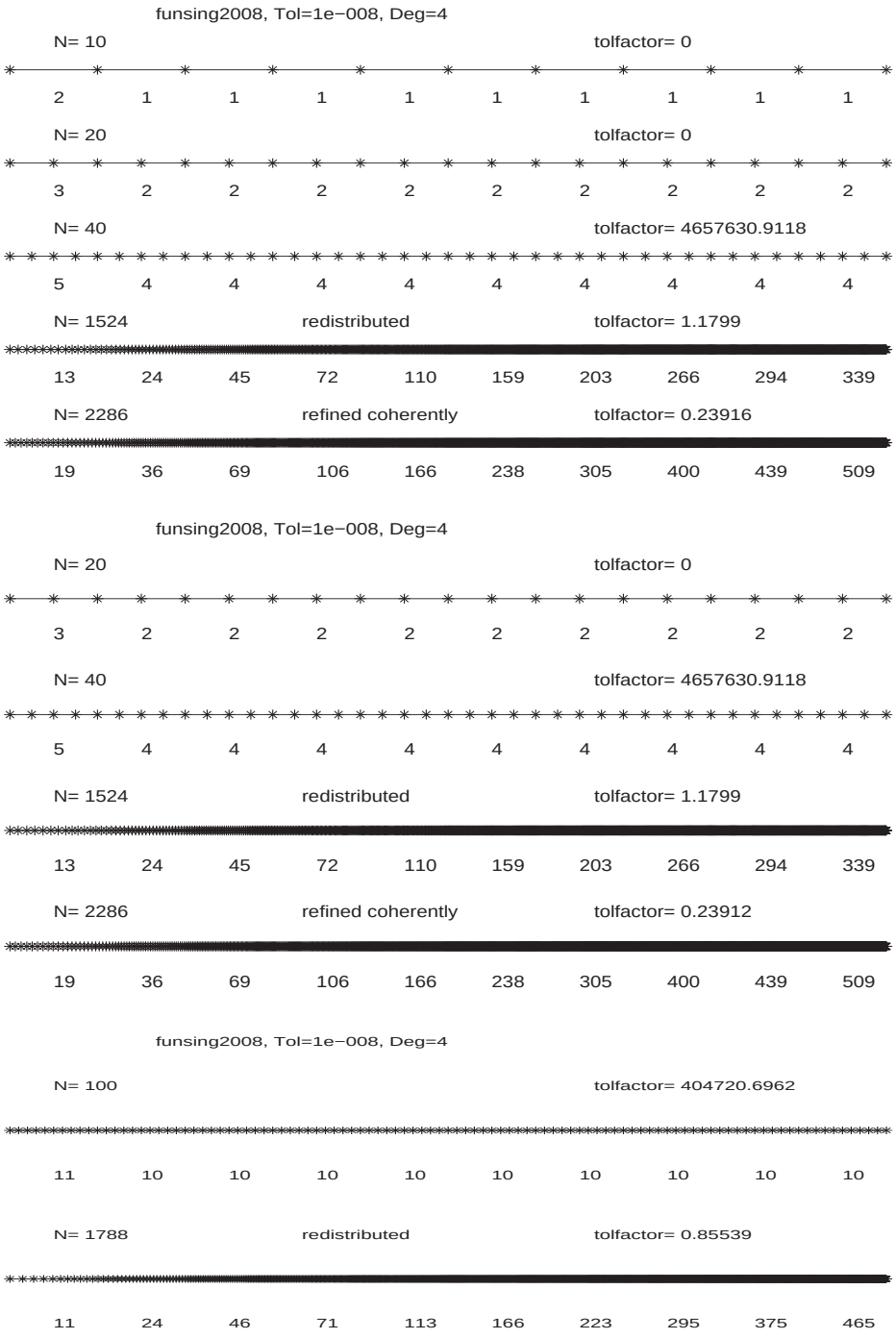


Figure 3.52: BVP 2008: Grid evolution for the tolerance 10^{-8} and collocation order 4.

BVP	Tol	Order	N_0	N	Time
funsing21a	10^{-6}	4	10	15	0.130
funsing21a	10^{-6}	4	20	20	0.090
funsing21a	10^{-6}	4	31	31	0.130
funsing21a	10^{-8}	4	10	44	0.161
funsing21a	10^{-8}	4	20	44	0.191
funsing21a	10^{-8}	4	100	100	0.300
funsing21a	10^{-8}	6	10	10	0.101
funsing21a	10^{-8}	6	20	20	0.130
funsing21a	10^{-8}	6	21	21	0.150

Table 3.137: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. For BVPS that are easy to solve, 100 meshpoints in the starting mesh are far too many. On the other hand, if the BVP is solved easily even 100 meshpoints do not take a long time.

BVP	Tol	Order	N_0	N	Time
funsing37c	10^{-6}	4	10	15	0.120
funsing37c	10^{-6}	4	20	20	0.130
funsing37c	10^{-6}	4	31	31	0.140
funsing37c	10^{-8}	4	10	34	0.170
funsing37c	10^{-8}	4	20	34	0.170
funsing37c	10^{-8}	4	100	100	0.341
funsing37c	10^{-8}	6	10	10	0.090
funsing37c	10^{-8}	6	20	20	0.140
funsing37c	10^{-8}	6	21	21	0.140

Table 3.138: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Again 100 meshpoints are far too many. BVP 37c is easy to solve.

BVP	Tol	Order	N_0	N	Time
funsing400	10^{-6}	4	10	10	0.040
funsing400	10^{-6}	4	20	20	0.070
funsing400	10^{-6}	4	31	31	0.060
funsing400	10^{-8}	4	10	34	0.070
funsing400	10^{-8}	4	20	33	0.101
funsing400	10^{-8}	4	100	100	0.140
funsing400	10^{-8}	6	10	10	0.050
funsing400	10^{-8}	6	20	20	0.060
funsing400	10^{-8}	6	21	21	0.070

Table 3.139: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Again, very fast results for a very easy to solve BVP.

BVP	Tol	Order	N_0	N	Time
funsing500	10^{-6}	4	10	10	0.030
funsing500	10^{-6}	4	20	20	0.030
funsing500	10^{-6}	4	31	31	0.050
funsing500	10^{-8}	4	10	38	0.100
funsing500	10^{-8}	4	20	30	0.060
funsing500	10^{-8}	4	100	100	0.100
funsing500	10^{-8}	6	10	10	0.040
funsing500	10^{-8}	6	20	20	0.040
funsing500	10^{-8}	6	21	21	0.040

Table 3.140: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Every solution is calculated extremely fast.

In Tables 3.137 - 3.140 it is obvious that automatic determination of the starting grid is providing too many meshpoints. But as these BVPS are solved so quickly even the large number of meshpoints does not cause a real problem. So if the optimization is done with care, these four BVPS are not showing representative results because one of the biggest advantages of automatical determination of the starting grid is that it provides stability and safety. The approximation on a mesh with 100 points is more accurate and causes less problems for the mesh adaptation routine than the one on a mesh with only 20 points. This fact must not be overlooked when looking at the 'easy' BVPS .

Bvp	Tol	Order	N_0	N	Time
funsing54	10^{-6}	4	10	107	0.701
funsing54	10^{-6}	4	20	234	1.622
funsing54	10^{-6}	4	31	188	1.252
funsing54	10^{-8}	4	10	1524	16.975
funsing54	10^{-8}	4	20	1133	10.866
funsing54	10^{-8}	4	100	2060	28.210
funsing54	10^{-8}	6	10	1227	25.937
funsing54	10^{-8}	6	20	1638	43.503
funsing54	10^{-8}	6	21	1661	44.944

Table 3.141: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid.

The convergence of Bvp 54 is rather slow, see the tolfactors in Figures 3.53 - 3.55. The bad result for 20 meshpoints at the tolerance at 10^{-6} and collocation order 4 is explained by the additional refinement needed, caused by the slightly failed tolerance. The same explanation can be given for the good runtime with 20 meshpoints compared to 10 meshpoints in the case of tolerance 10^{-8} and collocation order 4. The bad behavior for automatic determination is inexplicable. Though even more meshpoints were used compared to the last results when starting with 20 meshpoints, before the last refinement, the tolerance was not satisfied and refinement was necessary.

For the last configuration the number of refinements is about the same but with 10 points in the starting mesh, the number of redistributions of the mesh was greater by one. Thus it seems that this reduction in meshpoints led to the excellent result where the tolerances are satisfied with 400 points less.

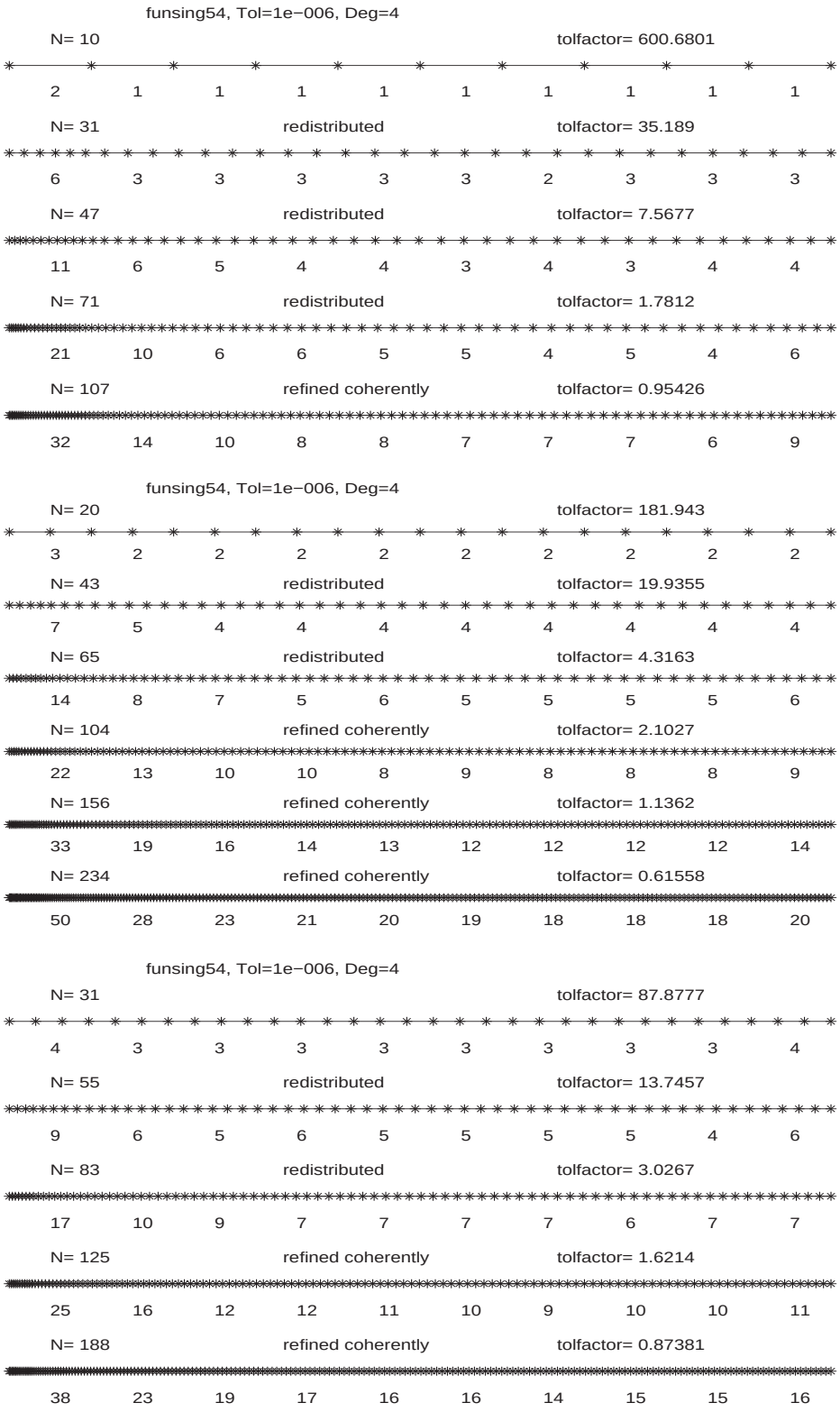


Figure 3.53: BVP 54: Grid evolution for the tolerance at 10^{-6} and collocation order 4.

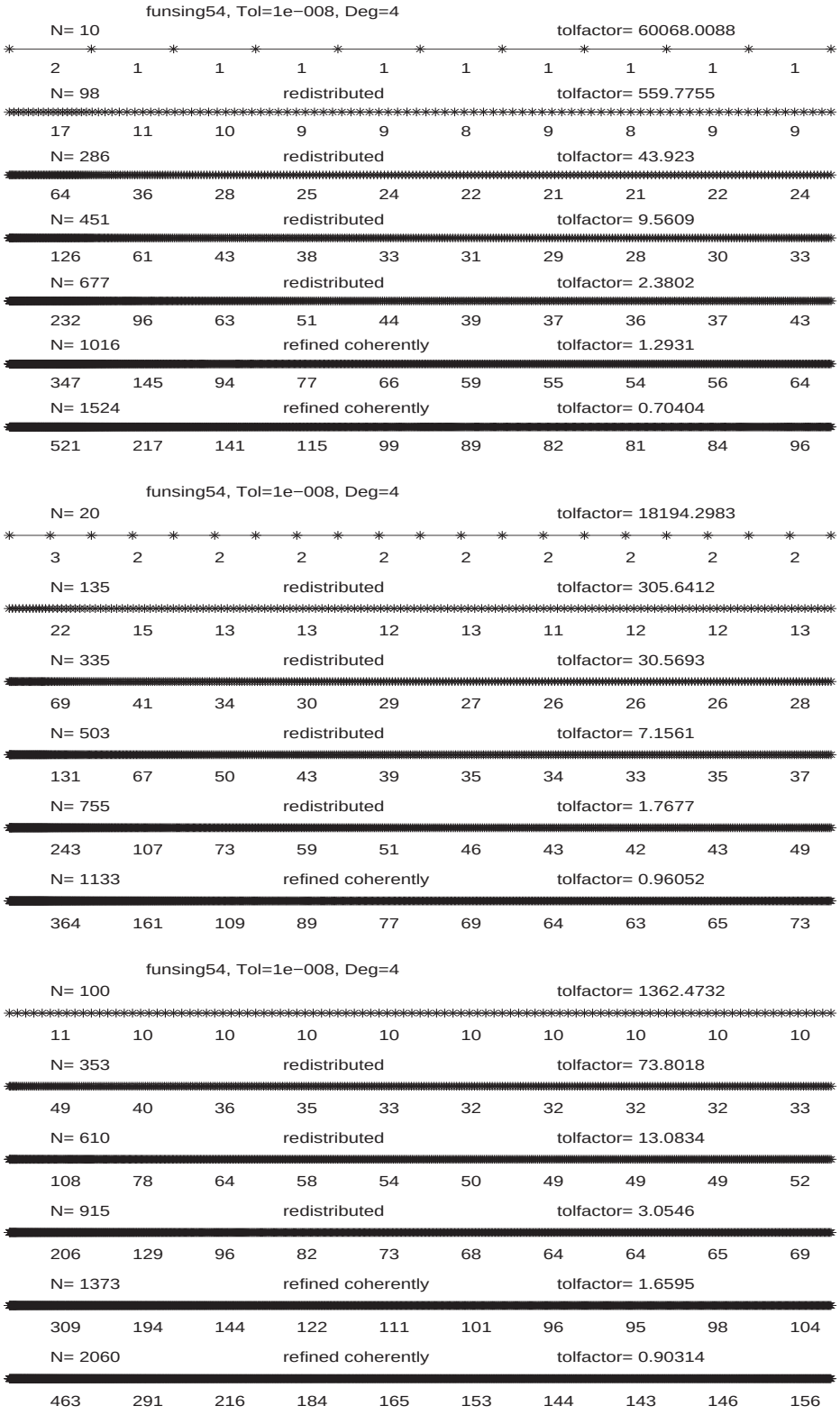


Figure 3.54: Bvp 54: Grid evolution for the tolerance at 10^{-8} and collocation order 4.

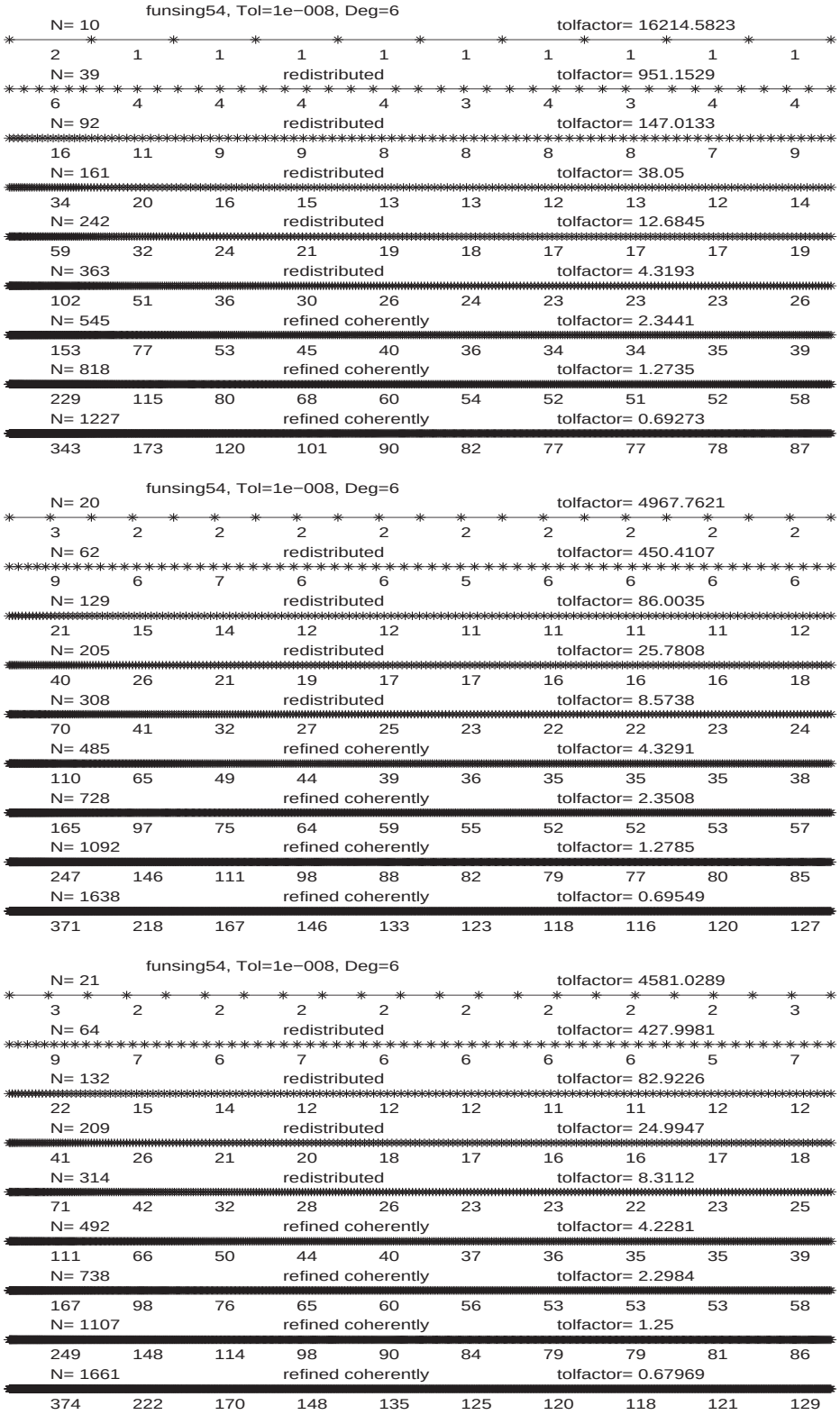


Figure 3.55: BVP 54: Grid evolution for the tolerance at 10^{-8} and collocation order 6.

Bvp	Tol	Order	N_0	N	Time
funsing55	10^{-6}	4	10	10	0.070
funsing55	10^{-6}	4	20	20	0.090
funsing55	10^{-6}	4	31	31	0.120
funsing55	10^{-8}	4	10	10	0.060
funsing55	10^{-8}	4	20	20	0.090
funsing55	10^{-8}	4	100	100	0.370
funsing55	10^{-8}	6	10	10	0.090
funsing55	10^{-8}	6	20	20	0.130
funsing55	10^{-8}	6	21	21	0.150

Table 3.142: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Bvp 55 is solved quickly and it does not cause problems at all, because for all configurations no refinements are needed. Therefore the 100 meshpoints are the worst choice here.

Bvp	Tol	Order	N_0	N	Time
funsing56	10^{-6}	4	10	200	2.393
funsing56	10^{-6}	4	20	160	1.942
funsing56	10^{-6}	4	31	124	1.572
funsing56	10^{-8}	4	10	708	18.266
funsing56	10^{-8}	4	20	360	6.840
funsing56	10^{-8}	4	100	300	4.827
funsing56	10^{-8}	6	10	80	1.673
funsing56	10^{-8}	6	20	80	1.593
funsing56	10^{-8}	6	21	84	1.702

Table 3.143: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. This Bvp demonstrates the advantages of automatic mesh determination. The problems are solved quickly and with less meshpoints in the final grid.

BVP	Tol	Order	N_0	N	Time
funsing6001	10^{-6}	4	10	41	0.280
funsing6001	10^{-6}	4	20	30	0.210
funsing6001	10^{-6}	4	31	47	0.271
funsing6001	10^{-8}	4	10	126	0.500
funsing6001	10^{-8}	4	20	132	0.541
funsing6001	10^{-8}	4	100	150	0.701
funsing6001	10^{-8}	6	10	36	0.340
funsing6001	10^{-8}	6	20	33	0.271
funsing6001	10^{-8}	6	21	34	0.290

Table 3.144: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Except for the tolerance 10^{-8} and the collocation order 4, automatic determination is quite competitive.

BVP	Tol	Order	N_0	N	Time
funsing6002	10^{-6}	4	10	10	0.060
funsing6002	10^{-6}	4	20	20	0.090
funsing6002	10^{-6}	4	31	31	0.120
funsing6002	10^{-8}	4	10	15	0.100
funsing6002	10^{-8}	4	20	20	0.100
funsing6002	10^{-8}	4	100	100	0.261
funsing6002	10^{-8}	6	10	10	0.070
funsing6002	10^{-8}	6	20	20	0.120
funsing6002	10^{-8}	6	21	21	0.100

Table 3.145: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. As the BVP is solved rather quickly, $N_0 = 100$ meshpoints are far too many.

Bvp	Tol	Order	N_0	N	Time
funsing7001b	10^{-6}	4	10	10	0.070
funsing7001b	10^{-6}	4	20	20	0.110
funsing7001b	10^{-6}	4	31	31	0.140
funsing7001b	10^{-8}	4	10	10	0.080
funsing7001b	10^{-8}	4	20	20	0.100
funsing7001b	10^{-8}	4	100	100	0.291
funsing7001b	10^{-8}	6	10	10	0.090
funsing7001b	10^{-8}	6	20	20	0.140
funsing7001b	10^{-8}	6	21	21	0.140

Table 3.146: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Again a quick solution is achieved and so automatic determination is much too pessimistic.

Bvp	Tol	Order	N_0	N	Time
funsing71	10^{-6}	4	10	81	0.260
funsing71	10^{-6}	4	20	81	0.261
funsing71	10^{-6}	4	31	79	0.270
funsing71	10^{-8}	4	10	254	0.621
funsing71	10^{-8}	4	20	254	0.621
funsing71	10^{-8}	4	100	242	0.752
funsing71	10^{-8}	6	10	61	0.300
funsing71	10^{-8}	6	20	64	0.330
funsing71	10^{-8}	6	21	64	0.341

Table 3.147: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. The results are quite close except for the case with tolerance 10^{-8} and collocation order 4, where automatic determination is slower.

BVP	Tol	Order	N_0	N	Time
funsing73	10^{-6}	4	10	365	8.302
funsing73	10^{-6}	4	20	365	8.202
funsing73	10^{-6}	4	31	372	7.380
funsing73	10^{-8}	4	10	1034	25.647
funsing73	10^{-8}	4	20	1034	25.567
funsing73	10^{-8}	4	100	1033	25.907
funsing73	10^{-8}	6	10	240	8.843
funsing73	10^{-8}	6	20	240	8.642
funsing73	10^{-8}	6	21	252	9.193

Table 3.148: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. One time automatic determination is best, one time competitive, one time worst.

BVP	Tol	Order	N_0	N	Time
funsing8001	10^{-6}	4	10	15	0.130
funsing8001	10^{-6}	4	20	20	0.120
funsing8001	10^{-6}	4	31	31	0.150
funsing8001	10^{-8}	4	10	50	0.240
funsing8001	10^{-8}	4	20	50	0.270
funsing8001	10^{-8}	4	100	100	0.380
funsing8001	10^{-8}	6	10	10	0.100
funsing8001	10^{-8}	6	20	20	0.141
funsing8001	10^{-8}	6	21	21	0.160

Table 3.149: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Since far less than 100 points are needed to satisfy the tolerances, automatic determination is lagging behind at the strict tolerance with the low degree of the collocation polynomial.

Bvp	Tol	Order	N_0	N	Time
funsing8002	10^{-6}	4	10	28	0.181
funsing8002	10^{-6}	4	20	30	0.181
funsing8002	10^{-6}	4	31	47	0.241
funsing8002	10^{-8}	4	10	87	0.331
funsing8002	10^{-8}	4	20	123	0.511
funsing8002	10^{-8}	4	100	150	0.651
funsing8002	10^{-8}	6	10	25	0.210
funsing8002	10^{-8}	6	20	30	0.230
funsing8002	10^{-8}	6	21	32	0.251

Table 3.150: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. 10 points lead to the best results here, automatic determination yields the worst results.

Bvp	Tol	Order	N_0	N	Time
funsing9003	10^{-6}	4	10	62	0.090
funsing9003	10^{-6}	4	20	103	0.130
funsing9003	10^{-6}	4	31	69	0.111
funsing9003	10^{-8}	4	10	194	0.211
funsing9003	10^{-8}	4	20	325	0.371
funsing9003	10^{-8}	4	100	166	0.270
funsing9003	10^{-8}	6	10	38	0.100
funsing9003	10^{-8}	6	20	44	0.100
funsing9003	10^{-8}	6	21	43	0.130

Table 3.151: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. 20 points are disadvantageous for the first two settings, 10 points seem a reasonable choice.

BVP	Tol	Order	N_0	N	Time
funsing9004	10^{-6}	4	10	252	0.320
funsing9004	10^{-6}	4	20	252	0.301
funsing9004	10^{-6}	4	31	244	0.301
funsing9004	10^{-8}	4	10	794	1.162
funsing9004	10^{-8}	4	20	794	1.152
funsing9004	10^{-8}	4	100	644	0.931
funsing9004	10^{-8}	6	10	138	0.240
funsing9004	10^{-8}	6	20	131	0.230
funsing9004	10^{-8}	6	21	131	0.240

Table 3.152: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. For this BVP automatic determination performs best in the most significant case.

BVP	Tol	Order	N_0	N	Time
funsing9005	10^{-6}	4	10	960	2.463
funsing9005	10^{-6}	4	20	960	2.474
funsing9005	10^{-6}	4	31	1488	4.736
funsing9005	10^{-8}	4	10	1501	5.618
funsing9005	10^{-8}	4	20	1501	5.618
funsing9005	10^{-8}	4	100	1500	5.598
funsing9005	10^{-8}	6	10	960	4.627
funsing9005	10^{-8}	6	20	960	4.637
funsing9005	10^{-8}	6	21	1008	4.977

Table 3.153: Results for different starting grids using manual degree selection. The test was done for 10 and 20 points in the starting grid and for automatic determination of the starting grid. Yet another surprisingly bad result for automatic grid determination in the first case.

3.4.1 Conclusion

In the tests from Section 3.4 the first important thing to observe is that for the tolerance at 10^{-8} and collocation order 4, automatic grid determination is a bad choice for the cases where the tolerances are satisfied on grids with far less meshpoints than 100. This is, of course, the case for easy to solve BVPS.

It is important to recall the following fact: When the tolerances are satisfied on about 20 points the same holds on 100 points. Because of the low degree the solution at 100 points is still calculated reasonably fast. More interestingly consider the cases, where 100 points make sense. This is the case for difficult problems, where 100 points give dependable information about the behavior of the solution which is not resolved using fewer points. On the other hand the meshes with up to 100 points are not computationally expensive and we may expect 100 points resulting from mesh refinement to be better distributed than 100 equidistributed

points on the starting grid. The question is whether the information gained during the refinements up to 100 points justifies the additional time needed for the refinements. Moreover, the information from coarse grids may be unreliable and even misleading.

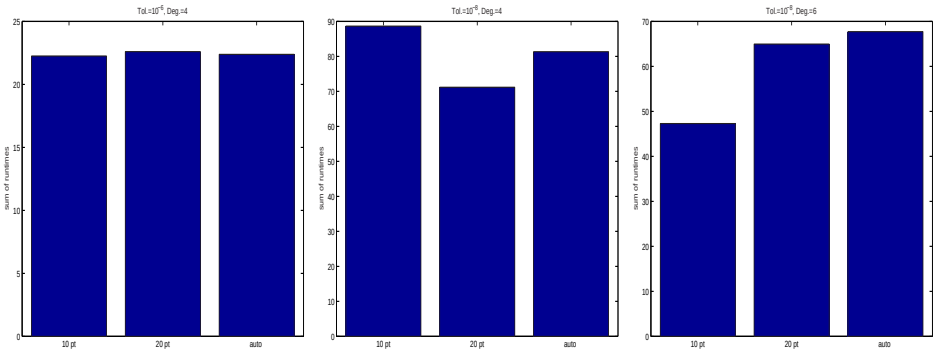


Figure 3.56: Comparison of the total sum of runtimes for the tests with manual degree selection.

The results given in Figure 3.56 are quite biased because for BVPS 54 and 56 we observe very large differences in runtime at very high runtimes, so the peaks are dominated by these examples. Figure 3.57 shows the results, where these two examples are excluded.

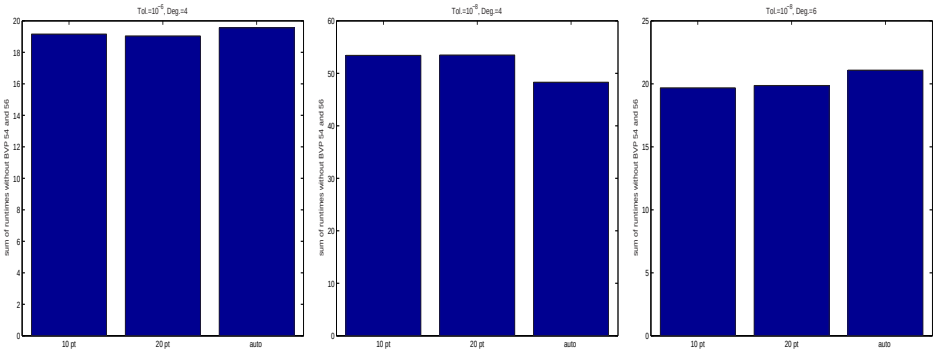


Figure 3.57: Comparison of the total sum of runtimes for the tests with manual degree selection. Results for BVPS 54 and 56 are removed.

Obviously automatic grid choice performs worst in the cases with tolerance 10^{-8} and collocation order 6 and tolerance 10^{-6} and collocation order 4 but is impressive for the remaining choices. 10 meshpoints and 20 meshpoints are balanced. The good result for automatic determination is related to another BVP that needs quite a long time to be solved, BVP 2008. If the results of BVP 2008 are eliminated too, Figure 3.58 shows the corresponding statistics.

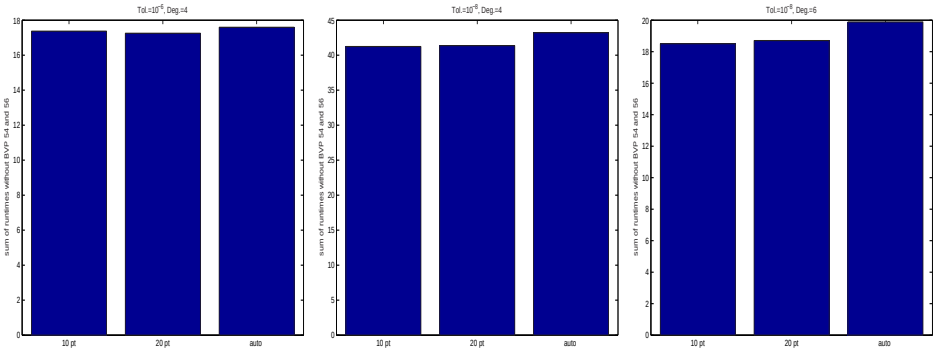


Figure 3.58: Comparison of the total sum of runtimes for the tests with manual degree selection. Results for BVPS 2008, 54 and 56 are removed.

At average, the automatic grid detection is not the best choice. First of all beacuase many BVPS are finished on grids with only few meshpoints. But these results are calculated so quickly that even 100 points do not take too much time. There is no big difference between 10 or 20 points, so the first decision is to choose from the automatic determination and 10 points on the starting grid.

When we take a look at Figures 3.56 - 3.58, there might be an advantage for strictly using 10 points in the starting grid throughout. But especially the results for BVPS 2008 and 56, the tolerance 10^{-8}, and the collocation order 4 show that automatic detection can be favorable in certain cases. On the other hand the best results for 10 meshpoints are achieved for BVP 54 which we found to be sensitive with respect to the starting grid.

Finally, we note that automatic determination of the collocation order results in a choice of $N_0 = 10$ in the initial mesh. Therefore the decision was made to retain automatic grid determination for reasons of stability for difficult problems when a low collocation order is used, even if this may imply a disadvantage for easy problems.

3.5 Intmaxminratio

The ratio of the maximal to minimal stepsize in a mesh IntMaxMinRatio K, is set to 10 in the routine. Our goal was to find out if 100 or maybe even 1000 is more suitable. Therefore the usual tests are executed (i. e. the tolerance is set to 10^{-3}, 10^{-6} and 10^{-8}) for the different settings of the ratio.

For a mesh $\tau := (\tau_1, \tau_2, \ldots, \tau_n)$ with the stepsizes $h := (h_1, \ldots, h_{n-1})$ such that $h_i = \tau_{i+1} - \tau_i$, the ratio hmax2hmin is calculated via

$$hmax2hmin = \frac{\max\limits_i h_i}{\min\limits_i h_i},$$

i. e. it is the ratio between the longest and the shortest subinterval in the mesh. The IntMaxMinRatio governs the maximal ratio allowed. It is used for the correction of the new mesh density function $\bar{\rho}$ if the ratio were too large. First the mesh density function ρ is calculated from the collocation grid. A new mesh density function $\tilde{\rho}$ is calculated using the MonitorFunction $\tilde{\theta}$ and then

$$\kappa = \frac{\max\limits_i \tilde{\rho}_i}{K}, \qquad \bar{\rho} = (\max(\tilde{\rho}_1, \kappa), \ldots, \max(\tilde{\rho}_n, \kappa)).$$

This is done to avoid too wide intervals, which could cause problems if an essential characteristic of the solution progression is missed. Moreover, hmax2hmin influences the stability constant of the scheme and should not be too large.

3.5.1 Test Results

In the following tables the last calculated hmax2hmin ratio is enlisted. Entries 'n. calc.' mean that this quantity has not been calculated during the solution process. This might happen when the solution is reached on the starting grid or when the error estimate was unreliable and the mesh was refined for this reason. In both cases this means that the ratio is 1.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing0026	10^{-3}	10	n. calc.	5	0.050
funsing0026	10^{-3}	100	n. calc.	5	0.060
funsing0026	10^{-3}	1000	n. calc.	5	0.050
funsing0026	10^{-6}	10	n. calc.	10	0.090
funsing0026	10^{-6}	100	n. calc.	10	0.080
funsing0026	10^{-6}	1000	n. calc.	10	0.080
funsing0026	10^{-8}	10	n. calc.	10	0.101
funsing0026	10^{-8}	100	n. calc.	10	0.100
funsing0026	10^{-8}	1000	n. calc.	10	0.100

Table 3.154: Test for the IntMaxMinRatio set to 10, 100 and 1000. Tolerances are reached on the starting grid. Without a single mesh refinement the IntMaxMinRatio was not used.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing1001	10^{-3}	10	n. calc.	40	0.110
funsing1001	10^{-3}	100	n. calc.	40	0.110
funsing1001	10^{-3}	1000	n. calc.	40	0.110
funsing1001	10^{-6}	10	1.000	60	0.211
funsing1001	10^{-6}	100	1.000	60	0.200
funsing1001	10^{-6}	1000	1.000	60	0.200
funsing1001	10^{-8}	10	1.000	58	0.200
funsing1001	10^{-8}	100	1.000	58	0.200
funsing1001	10^{-8}	1000	1.000	58	0.201

Table 3.155: Test for the IntMaxMinRatio set to 10, 100 and 1000. Only coherent refinements have been made.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing1002	10^{-3}	10	8.147	23	0.101
funsing1002	10^{-3}	100	8.847	23	0.090
funsing1002	10^{-3}	1000	8.847	23	0.090
funsing1002	10^{-6}	10	9.591	23	0.090
funsing1002	10^{-6}	100	16.815	35	0.140
funsing1002	10^{-6}	1000	16.815	35	0.130
funsing1002	10^{-8}	10	9.523	22	0.100
funsing1002	10^{-8}	100	13.041	22	0.090
funsing1002	10^{-8}	1000	13.041	22	0.111

Table 3.156: Test for the IntMaxMinRatio set to 10, 100 and 1000. For the tolerance set to 10^{-6} the IntMaxMinRatio is best at 10.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing1004	10^{-3}	10	3.100	9	0.040
funsing1004	10^{-3}	100	3.100	9	0.071
funsing1004	10^{-3}	1000	3.100	9	0.050
funsing1004	10^{-6}	10	1.000	15	0.080
funsing1004	10^{-6}	100	1.000	15	0.060
funsing1004	10^{-6}	1000	1.000	15	0.070
funsing1004	10^{-8}	10	1.000	15	0.070
funsing1004	10^{-8}	100	1.000	15	0.080
funsing1004	10^{-8}	1000	1.000	15	0.090

Table 3.157: Test for the IntMaxMinRatio set to 10, 100 and 1000. Very low runtimes, so there is no tendency to see.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing1005	10^{-3}	10	9.753	240	0.591
funsing1005	10^{-3}	100	98.667	240	0.591
funsing1005	10^{-3}	1000	987.575	960	2.744
funsing1005	10^{-6}	10	9.811	120	0.411
funsing1005	10^{-6}	100	99.448	120	0.421
funsing1005	10^{-6}	1000	995.673	960	5.118
funsing1005	10^{-8}	10	9.127	140	0.681
funsing1005	10^{-8}	100	94.668	96	0.431
funsing1005	10^{-8}	1000	959.218	576	4.596

Table 3.158: Test for the IntMaxMinRatio set to 10, 100 and 1000. Very bad results are achieved for the IntMaxMinRatio at 1000 but an excellent behavior is shown at 10^{-8} for 100.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing015	10^{-3}	10	1.000	20	0.060
funsing015	10^{-3}	100	1.000	20	0.070
funsing015	10^{-3}	1000	1.000	20	0.080
funsing015	10^{-6}	10	1.000	33	0.120
funsing015	10^{-6}	100	1.000	33	0.120
funsing015	10^{-6}	1000	1.000	33	0.120
funsing015	10^{-8}	10	1.000	26	0.141
funsing015	10^{-8}	100	1.000	26	0.131
funsing015	10^{-8}	1000	1.000	26	0.141

Table 3.159: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing15	10^{-3}	10	1.000	33	0.110
funsing15	10^{-3}	100	1.000	33	0.120
funsing15	10^{-3}	1000	1.000	33	0.130
funsing15	10^{-6}	10	1.000	33	0.130
funsing15	10^{-6}	100	1.000	33	0.120
funsing15	10^{-6}	1000	1.000	33	0.130
funsing15	10^{-8}	10	1.000	26	0.140
funsing15	10^{-8}	100	1.000	26	0.131
funsing15	10^{-8}	1000	1.000	26	0.150

Table 3.160: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing2002	10^{-3}	10	8.394	34	0.110
funsing2002	10^{-3}	100	19.925	80	0.170
funsing2002	10^{-3}	1000	19.925	80	0.170
funsing2002	10^{-6}	10	7.484	75	0.221
funsing2002	10^{-6}	100	7.484	75	0.220
funsing2002	10^{-6}	1000	7.484	75	0.230
funsing2002	10^{-8}	10	5.675	54	0.231
funsing2002	10^{-8}	100	5.675	54	0.220
funsing2002	10^{-8}	1000	5.675	54	0.230

Table 3.161: Test for the IntMaxMinRatio set to 10, 100 and 1000. The good results for 10 with the tolerance 10^{-3} are not repeated for stricter tolerances.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing2008	10^{-3}	10	8.471	90	0.170
funsing2008	10^{-3}	100	38.202	238	0.421
funsing2008	10^{-3}	1000	38.202	238	0.441
funsing2008	10^{-6}	10	8.862	159	0.541
funsing2008	10^{-6}	100	17.705	158	0.561
funsing2008	10^{-6}	1000	17.705	158	0.551
funsing2008	10^{-8}	10	9.839	133	0.381
funsing2008	10^{-8}	100	13.828	132	0.381
funsing2008	10^{-8}	1000	13.828	132	0.390

Table 3.162: Test for the IntMaxMinRatio set to 10, 100 and 1000. Since this BVP is related to BVP 2002 the results demonstrate a similar behavior.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing21a	10^{-3}	10	n. calc.	5	0.080
funsing21a	10^{-3}	100	n. calc.	5	0.060
funsing21a	10^{-3}	1000	n. calc.	5	0.060
funsing21a	10^{-6}	10	n. calc.	10	0.080
funsing21a	10^{-6}	100	n. calc.	10	0.081
funsing21a	10^{-6}	1000	n. calc.	10	0.090
funsing21a	10^{-8}	10	n. calc.	10	0.101
funsing21a	10^{-8}	100	n. calc.	10	0.100
funsing21a	10^{-8}	1000	n. calc.	10	0.101

Table 3.163: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing400	10^{-3}	10	n. calc.	5	0.030
funsing400	10^{-3}	100	n. calc.	5	0.040
funsing400	10^{-3}	1000	n. calc.	5	0.041
funsing400	10^{-6}	10	n. calc.	10	0.050
funsing400	10^{-6}	100	n. calc.	10	0.050
funsing400	10^{-6}	1000	n. calc.	10	0.050
funsing400	10^{-8}	10	n. calc.	10	0.060
funsing400	10^{-8}	100	n. calc.	10	0.061
funsing400	10^{-8}	1000	n. calc.	10	0.050

Table 3.164: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing500	10^{-3}	10	n. calc.	5	0.040
funsing500	10^{-3}	100	n. calc.	5	0.031
funsing500	10^{-3}	1000	n. calc.	5	0.040
funsing500	10^{-6}	10	n. calc.	10	0.030
funsing500	10^{-6}	100	n. calc.	10	0.050
funsing500	10^{-6}	1000	n. calc.	10	0.040
funsing500	10^{-8}	10	n. calc.	10	0.050
funsing500	10^{-8}	100	n. calc.	10	0.050
funsing500	10^{-8}	1000	n. calc.	10	0.060

Table 3.165: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing55	10^{-3}	10	n. calc.	5	0.050
funsing55	10^{-3}	100	n. calc.	5	0.070
funsing55	10^{-3}	1000	n. calc.	5	0.060
funsing55	10^{-6}	10	n. calc.	10	0.090
funsing55	10^{-6}	100	n. calc.	10	0.080
funsing55	10^{-6}	1000	n. calc.	10	0.080
funsing55	10^{-8}	10	n. calc.	10	0.120
funsing55	10^{-8}	100	n. calc.	10	0.100
funsing55	10^{-8}	1000	n. calc.	10	0.100

Table 3.166: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing56	10^{-3}	10	1.000	120	1.632
funsing56	10^{-3}	100	1.000	120	1.642
funsing56	10^{-3}	1000	1.000	120	1.622
funsing56	10^{-6}	10	n. calc.	80	1.702
funsing56	10^{-6}	100	n. calc.	80	1.722
funsing56	10^{-6}	1000	n. calc.	80	1.723
funsing56	10^{-8}	10	n. calc.	80	2.413
funsing56	10^{-8}	100	n. calc.	80	2.433
funsing56	10^{-8}	1000	n. calc.	80	2.413

Table 3.167: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing6001	10^{-3}	10	n. calc.	10	0.150
funsing6001	10^{-3}	100	n. calc.	10	0.140
funsing6001	10^{-3}	1000	n. calc.	10	0.140
funsing6001	10^{-6}	10	4.116	15	0.220
funsing6001	10^{-6}	100	4.116	15	0.200
funsing6001	10^{-6}	1000	4.116	15	0.201
funsing6001	10^{-8}	10	2.835	15	0.230
funsing6001	10^{-8}	100	2.835	15	0.230
funsing6001	10^{-8}	1000	2.835	15	0.240

Table 3.168: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing6002	10^{-3}	10	n. calc.	5	0.060
funsing6002	10^{-3}	100	n. calc.	5	0.060
funsing6002	10^{-3}	1000	n. calc.	5	0.071
funsing6002	10^{-6}	10	n. calc.	10	0.080
funsing6002	10^{-6}	100	n. calc.	10	0.080
funsing6002	10^{-6}	1000	n. calc.	10	0.070
funsing6002	10^{-8}	10	n. calc.	10	0.100
funsing6002	10^{-8}	100	n. calc.	10	0.080
funsing6002	10^{-8}	1000	n. calc.	10	0.101

Table 3.169: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing7001b	10^{-3}	10	n. calc.	5	0.070
funsing7001b	10^{-3}	100	n. calc.	5	0.070
funsing7001b	10^{-3}	1000	n. calc.	5	0.060
funsing7001b	10^{-6}	10	n. calc.	10	0.100
funsing7001b	10^{-6}	100	n. calc.	10	0.090
funsing7001b	10^{-6}	1000	n. calc.	10	0.100
funsing7001b	10^{-8}	10	n. calc.	10	0.120
funsing7001b	10^{-8}	100	n. calc.	10	0.100
funsing7001b	10^{-8}	1000	n. calc.	10	0.100

Table 3.170: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing71	10^{-3}	10	2.165	14	0.130
funsing71	10^{-3}	100	2.165	14	0.131
funsing71	10^{-3}	1000	2.165	14	0.120
funsing71	10^{-6}	10	1.942	28	0.201
funsing71	10^{-6}	100	1.942	28	0.201
funsing71	10^{-6}	1000	1.942	28	0.210
funsing71	10^{-8}	10	1.648	45	0.470
funsing71	10^{-8}	100	1.648	45	0.461
funsing71	10^{-8}	1000	1.648	45	0.461

Table 3.171: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing8001	10^{-3}	10	n. calc.	5	0.080
funsing8001	10^{-3}	100	n. calc.	5	0.081
funsing8001	10^{-3}	1000	n. calc.	5	0.070
funsing8001	10^{-6}	10	n. calc.	10	0.091
funsing8001	10^{-6}	100	n. calc.	10	0.100
funsing8001	10^{-6}	1000	n. calc.	10	0.100
funsing8001	10^{-8}	10	n. calc.	10	0.110
funsing8001	10^{-8}	100	n. calc.	10	0.110
funsing8001	10^{-8}	1000	n. calc.	10	0.120

Table 3.172: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing8002	10^{-3}	10	5.400	8	0.140
funsing8002	10^{-3}	100	5.487	8	0.130
funsing8002	10^{-3}	1000	5.487	8	0.121
funsing8002	10^{-6}	10	9.616	15	0.171
funsing8002	10^{-6}	100	13.438	15	0.170
funsing8002	10^{-6}	1000	13.438	15	0.170
funsing8002	10^{-8}	10	9.279	15	0.231
funsing8002	10^{-8}	100	10.484	15	0.220
funsing8002	10^{-8}	1000	10.484	15	0.220

Table 3.173: Test for the IntMaxMinRatio set to 10, 100 and 1000.

BVP	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing9001	10^{-3}	10	1.000	107	0.261
funsing9001	10^{-3}	100	1.000	107	0.251
funsing9001	10^{-3}	1000	1.000	107	0.250
funsing9001	10^{-6}	10	1.000	81	0.231
funsing9001	10^{-6}	100	1.000	81	0.230
funsing9001	10^{-6}	1000	1.000	81	0.220
funsing9001	10^{-8}	10	1.000	45	0.130
funsing9001	10^{-8}	100	1.000	45	0.141
funsing9001	10^{-8}	1000	1.000	45	0.130

Table 3.174: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing9002	10^{-3}	10	1.000	147	0.280
funsing9002	10^{-3}	100	1.000	147	0.290
funsing9002	10^{-3}	1000	1.000	147	0.290
funsing9002	10^{-6}	10	1.000	92	0.250
funsing9002	10^{-6}	100	1.000	92	0.250
funsing9002	10^{-6}	1000	1.000	92	0.260
funsing9002	10^{-8}	10	1.000	69	0.271
funsing9002	10^{-8}	100	1.000	69	0.260
funsing9002	10^{-8}	1000	1.000	69	0.280

Table 3.175: Test for the IntMaxMinRatio set to 10, 100 and 1000.

Bvp	Tol	IntMaxMin	hmax2hmin	N	Runtime
funsing9003	10^{-3}	10	1.000	10	0.050
funsing9003	10^{-3}	100	1.000	10	0.050
funsing9003	10^{-3}	1000	1.000	10	0.050
funsing9003	10^{-6}	10	1.000	27	0.120
funsing9003	10^{-6}	100	1.000	27	0.110
funsing9003	10^{-6}	1000	1.000	27	0.110
funsing9003	10^{-8}	10	1.000	16	0.080
funsing9003	10^{-8}	100	1.000	16	0.090
funsing9003	10^{-8}	1000	1.000	16	0.081

Table 3.176: Test for the IntMaxMinRatio set to 10, 100 and 1000.

3.5.2 Conclusion

When looking at Tables 3.154 - 3.175 one finds only 5 Bvps with differences for the different settings of the ratio. So the first thing to keep in mind is that the question of the influence of the ratio is limited to 5 out of 23 Bvps.

When these 5 Bvps are analyzed (Bvps 1002, 1005, 2002, 2008 and 8002) there are 3 wins for the ratio at 10 and only 1 win for the ratio at 100 and even one tie can be discovered (at Bvp 8002). But the most striking result is the big loss of 1000 at Bvp 1005. Therefore 1000 is excluded from further considerations.

Though 10 performs better for 3 Bvps, it must be noted that these occur for the tolerances set to 10^{-3} and 10^{-6}, whereas the one case where 100 wins the tolerance is set to 10^{-8}. Additionally, the results for Bvp 2008 for the higher tolerances show that IntMaxMinRatio 100 has one meshpoint less in the final grid with a comparable runtime. Since the results are so balanced, the decision is made to set the IntMaxMinRatio to 100 to allow a wider range of possible grids, because this choice shows good results for stricter tolerances.

Chapter 4

Comparisons

Now, after setting the parameters of SBVP 2.0, it is time to test the new version of our code against the old one and also against the routines BVP4C and COLNEW.

4.1 BVP4C

BVP4C is the standard routine for solving two-point boundary value problems for ordinary differential equations in MATLAB . It was implemented by Shampine, Kierzenka and Reichelt, cf. [21]. The version that was used for the tests has been Revision 1.21 which is a part of Mathworks Matlab 6.5.0.180913a Release 13.

For further information about BVP4C refer to [21] or to the Matlab help pages which can be found on the website www.mathworks.com.

4.2 SBVP 1.0 vs. SBVP 2.0 vs. BVP4C

In the following tests the runtimes of the different routines are referred to as $T_{2.0}$ for the runtime of SBVP 2.0 , $T_{1.0}$ for the runtime of SBVP 1.0 and T_{4c} for the runtime of BVP4C. In the same way the number of meshpoints are $N_{2.0}$, $N_{1.0}$ and N_{4c}.

The test covers 26 different BVPS and three values of tolerances already used before. The runtime and the number of meshpoints are the most important test criteria. Another important criterion is the number of dfcounts and linked with it the number of fcounts. As dfcounts are computationally more expensive than fcounts these two numbers cannot be summed up into one performance characteristic, so the comparison of fcounts and dfcounts is always a bit difficult.

Looking at the test results it is important to keep in mind that BVP4C aims for solutions at moderate tolerances and therefore it uses only 4 order method to solve the BVPS. Thus we expect the results provided by BVP4C to be worse than those of the other routines, especially for strict tolerances.

Bvp	solver	Tol	fct	dfct	N	Time
funsing015	Sbvp 1.0	10^{-8}	525	526	32	0.150
funsing015	Sbvp 2.0	10^{-8}	957	958	26	0.151
funsing015	Bvp4c	10^{-8}	21732	0	671	2.213
funsing015	Sbvp 1.0	10^{-6}	633	634	48	0.160
funsing015	Sbvp 2.0	10^{-6}	913	914	33	0.120
funsing015	Bvp4c	10^{-6}	6618	0	191	0.711
funsing015	Sbvp 1.0	10^{-3}	503	504	76	0.131
funsing015	Sbvp 2.0	10^{-3}	303	304	20	0.080
funsing015	Bvp4c	10^{-3}	1242	0	30	0.150

Table 4.1: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. Strikingly bad results for Bvp4c at high tolerances.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1001	Sbvp 1.0	10^{-8}	553	544	50	0.140
funsing1001	Sbvp 2.0	10^{-8}	1587	1588	58	0.191
funsing1001	Bvp4c	10^{-8}	6324	0	314	0.832
funsing1001	Sbvp 1.0	10^{-6}	433	424	50	0.110
funsing1001	Sbvp 2.0	10^{-6}	1823	1824	60	0.191
funsing1001	Bvp4c	10^{-6}	5469	0	187	0.781
funsing1001	Sbvp 1.0	10^{-3}	343	339	37	0.102
funsing1001	Sbvp 2.0	10^{-3}	703	704	40	0.090
funsing1001	Bvp4c	10^{-3}	1377	0	45	0.210

Table 4.2: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. Sbvp 1.0 is quicker for this linear Bvp.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1002	Sbvp 1.0	10^{-8}	588	589	33	0.150
funsing1002	Sbvp 2.0	10^{-8}	579	580	22	0.110
funsing1002	Bvp4c	10^{-8}	20065	0	814	2.573
funsing1002	Sbvp 1.0	10^{-6}	234	235	23	0.321
funsing1002	Sbvp 2.0	10^{-6}	955	956	35	0.120
funsing1002	Bvp4c	10^{-6}	5846	0	203	0.801
funsing1002	Sbvp 1.0	10^{-3}	158	154	25	0.080
funsing1002	Sbvp 2.0	10^{-3}	483	484	23	0.090
funsing1002	Bvp4c	10^{-3}	1049	0	37	0.151

Table 4.3: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. Best results for the new version and Bvp4c is not competitive for stricter tolerances.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1004	Sbvp 1.0	10^{-8}	228	229	15	0.091
funsing1004	Sbvp 2.0	10^{-8}	453	454	15	0.090
funsing1004	Bvp4c	10^{-8}	10278	0	452	1.351
funsing1004	Sbvp 1.0	10^{-6}	178	179	15	0.080
funsing1004	Sbvp 2.0	10^{-6}	353	354	15	0.080
funsing1004	Bvp4c	10^{-6}	3072	0	125	0.441
funsing1004	Sbvp 1.0	10^{-3}	68	69	8	0.070
funsing1004	Sbvp 2.0	10^{-3}	103	104	10	0.040
funsing1004	Bvp4c	10^{-3}	540	0	21	0.090

Table 4.4: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. While Bvp4c is competitive for the tolerance set to 10^{-3} it loses ground at tolerances 10^{-6} and 10^{-8}.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1005	Sbvp 2.0	10^{-8}	3711	3712	96	0.411
funsing1005	Bvp4c	10^{-8}	9438	0	387	1.162
funsing1005	Sbvp 1.0	10^{-6}	2015	2006	226	0.471
funsing1005	Sbvp 2.0	10^{-6}	3783	3784	120	0.410
funsing1005	Bvp4c	10^{-6}	8111	0	172	1.062
funsing1005	Sbvp 1.0	10^{-3}	1883	1879	241	0.370
funsing1005	Sbvp 2.0	10^{-3}	5503	5504	240	0.541
funsing1005	Bvp4c	10^{-3}	3345	0	34	0.481

Table 4.5: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. The higher the tolerance the better Sbvp 2.0. Sbvp 1.0 cannot handle this Bvp for the tolerance at 10^{-8}.

Bvp	solver	Tol	fct	dfct	N	Time
funsing15	Sbvp 1.0	10^{-8}	444	445	24	0.150
funsing15	Sbvp 2.0	10^{-8}	957	958	26	0.130
funsing15	Bvp4c	10^{-8}	13804	0	572	1.463
funsing15	Sbvp 1.0	10^{-6}	388	389	27	0.111
funsing15	Sbvp 2.0	10^{-6}	913	914	33	0.120
funsing15	Bvp4c	10^{-6}	4312	0	163	0.530
funsing15	Sbvp 1.0	10^{-3}	518	519	42	0.150
funsing15	Sbvp 2.0	10^{-3}	1023	1024	33	0.130
funsing15	Bvp4c	10^{-3}	1128	0	27	0.150

Table 4.6: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. The slightly increased number of meshpoints needed by Sbvp 2.0 compared to Sbvp 1.0 at tolerance 10^{-8} is not reflected in the runtime.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1a	Sbvp 1.0	10^{-8}	843	934	10	0.130
funsing1a	Sbvp 2.0	10^{-8}	2713	354	10	0.110
funsing1a	Bvp4c	10^{-8}	1162	0	68	0.160
funsing1a	Sbvp 1.0	10^{-6}	643	714	10	0.120
funsing1a	Sbvp 2.0	10^{-6}	2053	274	10	0.100
funsing1a	Bvp4c	10^{-6}	356	0	18	0.061
funsing1a	Sbvp 1.0	10^{-3}	203	229	5	0.160
funsing1a	Sbvp 2.0	10^{-3}	1443	194	10	0.080
funsing1a	Bvp4c	10^{-3}	101	0	4	0.020

Table 4.7: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This is an easy Bvp where Bvp4c can compete even for the strict tolerance, though it needs more meshpoints.

Bvp	solver	Tol	fct	dfct	N	Time
funsing2002	Sbvp 1.0	10^{-8}	849	850	51	0.241
funsing2002	Sbvp 2.0	10^{-8}	1803	1804	54	0.211
funsing2002	Bvp4c	10^{-8}	46196	0	1398	5.237
funsing2002	Sbvp 1.0	10^{-6}	843	844	66	0.190
funsing2002	Sbvp 2.0	10^{-6}	1893	1894	75	0.211
funsing2002	Bvp4c	10^{-6}	10574	0	400	1.231
funsing2002	Sbvp 1.0	10^{-3}	353	349	39	0.110
funsing2002	Sbvp 2.0	10^{-3}	1433	1434	80	0.160
funsing2002	Bvp4c	10^{-3}	1446	0	58	0.190

Table 4.8: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. For this Bvp Bvp4c needs nearly 30 times more meshpoints compared to Sbvp 1.0 or Sbvp 2.0.

Bvp	solver	Tol	fct	dfct	N	Time
funsing2008	Sbvp 1.0	10^{-8}	2353	2344	120	0.621
funsing2008	Sbvp 2.0	10^{-8}	2919	2920	132	0.381
funsing2008	Bvp4c	10^{-8}	108676	0	3182	12.909
funsing2008	Sbvp 1.0	10^{-6}	2211	2202	153	0.541
funsing2008	Sbvp 2.0	10^{-6}	4665	4666	158	0.490
funsing2008	Bvp4c	10^{-6}	31137	0	872	3.566
funsing2008	Sbvp 1.0	10^{-3}	1253	1254	196	0.270
funsing2008	Sbvp 2.0	10^{-3}	3943	3944	238	0.421
funsing2008	Bvp4c	10^{-3}	4374	0	127	0.491

Table 4.9: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. Bvp4c once again is competitive at the tolerance set to 10^{-3}, but it is very slow and ineffective for strict tolerances.

Bvp	solver	Tol	fct	dfct	N	Time
funsing21a	Sbvp 1.0	10^{-8}	1013	1104	10	0.191
funsing21a	Sbvp 2.0	10^{-8}	1953	434	10	0.110
funsing21a	Bvp4c	10^{-8}	2268	0	129	0.281
funsing21a	Sbvp 1.0	10^{-6}	773	844	10	0.110
funsing21a	Sbvp 2.0	10^{-6}	1633	334	10	0.090
funsing21a	Bvp4c	10^{-6}	752	0	37	0.110
funsing21a	Sbvp 1.0	10^{-3}	273	299	5	0.100
funsing21a	Sbvp 2.0	10^{-3}	1233	234	10	0.080
funsing21a	Bvp4c	10^{-3}	165	0	5	0.050

Table 4.10: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. Sbvp 2.0 is faster than Sbvp 1.0 in all cases. Bvp4c shows its usual behavior.

Bvp	solver	Tol	fct	dfct	N	Time
funsing37c	Sbvp 1.0	10^{-8}	843	934	10	0.130
funsing37c	Sbvp 2.0	10^{-8}	3023	354	10	0.110
funsing37c	Bvp4c	10^{-8}	2738	0	152	0.320
funsing37c	Sbvp 1.0	10^{-6}	643	714	10	0.110
funsing37c	Sbvp 2.0	10^{-6}	2353	274	10	0.100
funsing37c	Bvp4c	10^{-6}	880	0	44	0.120
funsing37c	Sbvp 1.0	10^{-3}	203	229	5	0.080
funsing37c	Sbvp 2.0	10^{-3}	1523	99	10	0.080
funsing37c	Bvp4c	10^{-3}	167	0	7	0.040

Table 4.11: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This Bvp shows results that are quite similar to those of 4.10.

Bvp	solver	Tol	fct	dfct	N	Time
funsing400	Sbvp 1.0	10^{-8}	93	94	10	0.060
funsing400	Sbvp 2.0	10^{-8}	183	184	10	0.040
funsing400	Bvp4c	10^{-8}	3295	0	182	0.411
funsing400	Sbvp 1.0	10^{-6}	73	74	10	0.041
funsing400	Sbvp 2.0	10^{-6}	143	144	10	0.060
funsing400	Bvp4c	10^{-6}	1056	0	55	0.160
funsing400	Sbvp 1.0	10^{-3}	28	29	5	0.030
funsing400	Sbvp 2.0	10^{-3}	103	104	10	0.050
funsing400	Bvp4c	10^{-3}	146	0	8	0.051

Table 4.12: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. The runtimes for this Bvp are too short to draw any conclusions, but Bvp4c does not seem favorable here.

Bvp	solver	Tol	fct	dfct	N	Time
funsing500	Sbvp 1.0	10^{-8}	93	94	10	0.040
funsing500	Sbvp 2.0	10^{-8}	183	184	10	0.040
funsing500	Bvp4c	10^{-8}	2056	0	144	0.240
funsing500	Sbvp 1.0	10^{-6}	73	74	10	0.040
funsing500	Sbvp 2.0	10^{-6}	143	144	10	0.030
funsing500	Bvp4c	10^{-6}	657	0	43	0.090
funsing500	Sbvp 1.0	10^{-3}	28	29	5	0.040
funsing500	Sbvp 2.0	10^{-3}	103	104	10	0.060
funsing500	Bvp4c	10^{-3}	91	0	6	0.020

Table 4.13: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. Again very slow runtimes of Bvp4c for strict tolerances.

Bvp	solver	Tol	fct	dfct	N	Time
funsing54	Sbvp 1.0	10^{-8}	859811	868588	2952	798.658
funsing54	Sbvp 2.0	10^{-8}	327428	160384	1566	80.095
funsing54	Bvp4c	10^{-8}	158627	0	376	14.271
funsing54	Sbvp 1.0	10^{-6}	34367	34852	177	4.357
funsing54	Sbvp 2.0	10^{-6}	27500	11215	149	1.643
funsing54	Bvp4c	10^{-6}	76381	0	108	3.495
funsing54	Sbvp 1.0	10^{-3}	640	654	8	0.150
funsing54	Sbvp 2.0	10^{-3}	1093	234	10	0.080
funsing54	Bvp4c	10^{-3}	55091	0	21	0.520

Table 4.14: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This Bvp is hard to solve for strict tolerances. The gain of performance for Sbvp 2.0 compared to Sbvp 1.0 is striking . But the results for Bvp4c are also convincing.

Bvp	solver	Tol	fct	dfct	N	Time
funsing55	Sbvp 1.0	10^{-8}	343	434	10	0.121
funsing55	Sbvp 2.0	10^{-8}	433	354	10	0.090
funsing55	Sbvp 1.0	10^{-6}	263	334	10	0.090
funsing55	Sbvp 2.0	10^{-6}	333	274	10	0.080
funsing55	Sbvp 1.0	10^{-3}	93	119	5	0.090
funsing55	Sbvp 2.0	10^{-3}	233	194	10	0.070

Table 4.15: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This is one of two Bvps where Bvp4c fails in every single case. The other two routines perform very well and finish on the first grid.

BVP	solver	Tol	fct	dfct	N	Time
funsing56	SBVP 1.0	10^{-8}	168333	48085	96	14.691
funsing56	SBVP 2.0	10^{-8}	20963	6934	80	2.403
funsing56	BVP4C	10^{-8}	96748	0	108	0.391
funsing56	SBVP 1.0	10^{-6}	7623	1884	10	0.450
funsing56	SBVP 2.0	10^{-6}	25223	5634	80	1.903
funsing56	BVP4C	10^{-6}	102399	0	191	1.001
funsing56	SBVP 1.0	10^{-3}	11292	3653	18	0.771
funsing56	SBVP 2.0	10^{-3}	33623	5414	120	1.622
funsing56	BVP4C	10^{-3}	96206	0	29	0.381

Table 4.16: Comparison between SBVP 1.0, SBVP 2.0 and BVP4C. This is a difficult BVP where the setting of 5 meshpoints for the starting grid for SBVP 1.0 in the case with tolerances at 10^{-3} leads to a bad result. Stunning results for BVP4C and 10^{-8} while SBVP 2.0 has worst results for 10^{-6}.

BVP	solver	Tol	fct	dfct	N	Time
funsing6001	SBVP 1.0	10^{-8}	2843	2989	15	0.331
funsing6001	SBVP 2.0	10^{-8}	5658	1119	15	0.230
funsing6001	BVP4C	10^{-8}	47399	0	420	0.871
funsing6001	SBVP 1.0	10^{-6}	2013	2129	15	0.250
funsing6001	SBVP 2.0	10^{-6}	4473	859	15	0.190
funsing6001	BVP4C	10^{-6}	42467	0	113	0.320
funsing6001	SBVP 1.0	10^{-3}	701	747	8	0.140
funsing6001	SBVP 2.0	10^{-3}	1743	314	10	0.100
funsing6001	BVP4C	10^{-3}	40760	0	17	0.100

Table 4.17: Comparison between SBVP 1.0, SBVP 2.0 and BVP4C. This is a typical BVP with SBVP 2.0 at its best.

BVP	solver	Tol	fct	dfct	N	Time
funsing6002	SBVP 1.0	10^{-8}	763	854	10	0.131
funsing6002	SBVP 2.0	10^{-8}	1583	354	10	0.090
funsing6002	BVP4C	10^{-8}	6737	0	60	0.140
funsing6002	SBVP 1.0	10^{-6}	583	654	10	0.100
funsing6002	SBVP 2.0	10^{-6}	1203	274	10	0.070
funsing6002	BVP4C	10^{-6}	6035	0	18	0.070
funsing6002	SBVP 1.0	10^{-3}	158	184	5	0.070
funsing6002	SBVP 2.0	10^{-3}	873	194	10	0.080
funsing6002	BVP4C	10^{-3}	5771	0	4	0.020

Table 4.18: Comparison between SBVP 1.0, SBVP 2.0 and BVP4C. While both versions of SBVP solve the problem on the starting grid, BVP4C needs 60 meshpoints for 10^{-8}, still the computation of the solution is surprisingly efficient.

Bvp	solver	Tol	fct	dfct	N	Time
funsing7001b	Sbvp 1.0	10^{-8}	843	934	10	0.120
funsing7001b	Sbvp 2.0	10^{-8}	1823	434	10	0.120
funsing7001b	Bvp4c	10^{-8}	2401	0	13	0.060
funsing7001b	Sbvp 1.0	10^{-6}	643	714	10	0.110
funsing7001b	Sbvp 2.0	10^{-6}	1383	334	10	0.080
funsing7001b	Bvp4c	10^{-6}	2237	0	6	0.050
funsing7001b	Sbvp 1.0	10^{-3}	178	204	5	0.070
funsing7001b	Sbvp 2.0	10^{-3}	993	234	10	0.081
funsing7001b	Bvp4c	10^{-3}	2180	0	4	0.030

Table 4.19: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. A Bvp where all routines finish quickly.

Bvp	solver	Tol	fct	dfct	N	Time
funsing71	Sbvp 1.0	10^{-8}	9079	8787	39	0.791
funsing71	Sbvp 2.0	10^{-8}	11393	3299	45	0.461
funsing71	Bvp4c	10^{-8}	182154	0	187	5.448
funsing71	Sbvp 1.0	10^{-6}	5443	5322	33	0.451
funsing71	Sbvp 2.0	10^{-6}	4655	1270	28	0.200
funsing71	Bvp4c	10^{-6}	144230	0	56	1.352
funsing71	Sbvp 1.0	10^{-3}	2031	1910	18	0.200
funsing71	Sbvp 2.0	10^{-3}	2658	599	15	0.130
funsing71	Bvp4c	10^{-3}	134355	0	11	0.210

Table 4.20: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This regular Bvp is a good example for the improvement of Sbvp 2.0.

Bvp	solver	Tol	fct	dfct	N	Time
funsing73	Sbvp 1.0	10^{-8}	58146	31024	72	12.488
funsing73	Sbvp 2.0	10^{-8}	69443	12164	120	6.179
funsing73	Sbvp 1.0	10^{-6}	77313	34933	173	14.821
funsing73	Sbvp 2.0	10^{-6}	121223	16684	240	8.863
funsing73	Sbvp 1.0	10^{-3}	56739	26147	203	8.402
funsing73	Sbvp 2.0	10^{-3}	91963	11184	240	4.676

Table 4.21: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This is the next Bvp with a fatal result for Bvp4c. Good to see is the improved performance of Sbvp 2.0.

BVP	solver	Tol	fct	dfct	N	Time
funsing8001	SBVP 1.0	10^{-8}	1173	1184	10	0.171
funsing8001	SBVP 2.0	10^{-8}	2013	514	10	0.110
funsing8001	BVP4C	10^{-8}	21745	0	157	0.351
funsing8001	SBVP 1.0	10^{-6}	833	844	10	0.120
funsing8001	SBVP 2.0	10^{-6}	1673	394	10	0.110
funsing8001	BVP4C	10^{-6}	19957	0	44	0.150
funsing8001	SBVP 1.0	10^{-3}	313	319	5	0.080
funsing8001	SBVP 2.0	10^{-3}	1203	274	10	0.100
funsing8001	BVP4C	10^{-3}	19293	0	7	0.040

Table 4.22: Comparison between SBVP 1.0, SBVP 2.0 and BVP4C. Once more, SBVP 2.0 performs well and BVP4C is not competitive for strict tolerances.

BVP	solver	Tol	fct	dfct	N	Time
funsing8002	SBVP 1.0	10^{-8}	2927	2954	16	0.321
funsing8002	SBVP 2.0	10^{-8}	5188	1039	15	0.200
funsing8002	SBVP 1.0	10^{-6}	2343	2370	16	0.280
funsing8002	SBVP 2.0	10^{-6}	4203	799	15	0.161
funsing8002	SBVP 1.0	10^{-3}	722	756	8	0.130
funsing8002	SBVP 2.0	10^{-3}	2233	234	10	0.100
funsing8002	BVP4C	10^{-3}	128219	0	99	13.930

Table 4.23: Comparison between SBVP 1.0, SBVP 2.0 and BVP4C. BVP4C only handles the easy case, albeit very inefficiently, while the versions of SBVP do quite well, SBVP 2.0 even a bit better.

BVP	solver	Tol	fct	dfct	N	Time
funsing9001	SBVP 1.0	10^{-8}	1029	1030	78	0.260
funsing9001	SBVP 2.0	10^{-8}	993	994	45	0.130
funsing9001	SBVP 1.0	10^{-6}	997	998	99	0.221
funsing9001	SBVP 2.0	10^{-6}	2033	2034	81	0.220
funsing9001	BVP4C	10^{-6}	21279	0	97	2.313
funsing9001	SBVP 1.0	10^{-3}	1868	1864	304	0.380
funsing9001	SBVP 2.0	10^{-3}	2343	2344	107	0.241
funsing9001	BVP4C	10^{-3}	3334	0	19	0.350

Table 4.24: Comparison between SBVP 1.0, SBVP 2.0 and BVP4C. No result for BVP4C for 10^{-8} and hardly any problems for both versions of SBVP. Note the low number of meshpoints required by SBVP 2.0 for 10^{-8}.

Bvp	solver	Tol	fct	dfct	N	Time
funsing9002	Sbvp 1.0	10^{-8}	1677	1678	132	0.391
funsing9002	Sbvp 2.0	10^{-8}	2253	2254	69	0.260
funsing9002	Sbvp 1.0	10^{-6}	2131	2132	252	0.511
funsing9002	Sbvp 2.0	10^{-6}	2131	2132	92	0.230
funsing9002	Bvp4c	10^{-6}	21023	0	94	2.263
funsing9002	Sbvp 1.0	10^{-3}	3763	3759	618	0.911
funsing9002	Sbvp 2.0	10^{-3}	2753	2754	147	0.260
funsing9002	Bvp4c	10^{-3}	3270	0	19	0.350

Table 4.25: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. This table shows comparable results to Table 4.24.

Bvp	solver	Tol	fct	dfct	N	Time
funsing9003	Sbvp 1.0	10^{-8}	237	238	16	0.080
funsing9003	Sbvp 2.0	10^{-8}	471	472	16	0.050
funsing9003	Sbvp 1.0	10^{-6}	416	417	30	0.120
funsing9003	Sbvp 2.0	10^{-6}	773	774	27	0.100
funsing9003	Bvp4c	10^{-6}	82784	0	345	8.162
funsing9003	Sbvp 1.0	10^{-3}	83	84	11	0.070
funsing9003	Sbvp 2.0	10^{-3}	103	104	10	0.100
funsing9003	Bvp4c	10^{-3}	7910	0	52	0.711

Table 4.26: Comparison between Sbvp 1.0, Sbvp 2.0 and Bvp4c. The last member of this family of Bvps shows the best behavior for both versions of Sbvp.

4.2.1 Conclusions

For better comparability the runtimes have been converted to relative runtimes using the simple formula

$$S_j = \frac{\min(T_{2.0}, T_{1.0}, T_{4c})}{T_j}, \qquad j \in \{2.0, 1.0, 4c\}.$$

The results in Table 4.27 are quite impressive. Sbvp 2.0 is not failing a single Bvp, Sbvp 1.0 is failing to solve one and Bvp 4c has problems with 11 Bvps, most often for stricter tolerances. Sbvp 2.0 is quickest in 43 different configuration and only the slowest in 3, whereas Bvp4c is worst for 45 test configurations.

	Sbvp2.0	**Sbvp1.0**	**Bvp4c**
No results for	0	1	11
Best time at	43	11	24
Worst time at	3	30	45
$S_j > 0.9$	56	17	14
$S_j < 0.6$	9	28	44
10^{-3}	0.8190	0.6021	0.6665
10^{-6}	0.9096	0.7474	0.3970
10^{-8}	0.9077	0.6587	0.3356
Overall	0.8788	0.6695	0.4752

Table 4.27: Some characteristics of the comparison between the different Matlab-solvers.

Discussing the numbers from the second part of the table is a bit more complicated. Though taking the mean over numbers that have not been normalized with respect to a fixed value, but by the corresponding best runtime is not quite intuitive, the motivation for doing this comes from the fact that these values give a good picture about how the routines compare to the very best runtime at each configuration. That means that the higher the number the faster the routine. The 0.9077 for the tolerance set to 10^{-8} for SBVP 2.0 means: SBVP 2.0 is about 10 % behind the best runtime on average.

We conclude that SBVP 2.0 is on average indeed an improvement of SBVP 1.0, while BVP4C is suitable only for low accuracies due to the low order method used.
Further comparisons were therefore limited to the different versions of SBVP.

4.2.2 The Evolution is Completed

For better comparability the times are now normalized in the following way:

$$Z_j = \frac{T_{2.0,j}}{T_{1.0,j}}, \qquad j \in \{-3, -6, -8\},$$

so Z_j denotes the relative runtime $T_{rel} := \frac{T_{2.0}}{T_{1.0}}$ for tolerance 10^j.

In Figure 4.1 the excellent results for SBVP 2.0 can be seen. If the bar is lower than 1, the new version is faster than the old one. If it is at $\frac{1}{2}$ SBVP 1.0 takes twice the time SBVP 2.0 does. Only 10 out of 78 bars are higher than 1 and only one bar shows a really critical result. Indeed, for BVP 56 and tolerance 10^{-6}, the new version takes nearly 4 times longer than SBVP 1.0.

Figure 4.2 shows where the problems arise when we distinguish between the different tolerances. As the approach of all the tests was to optimize toward the performance in difficult situations the results for the tolerance at 10^{-8} with only one bar over the 1-line are very satisfying. As 4 bars are higher for 10^{-3}, this is the highest number at all. Again a quite satisfactory result.

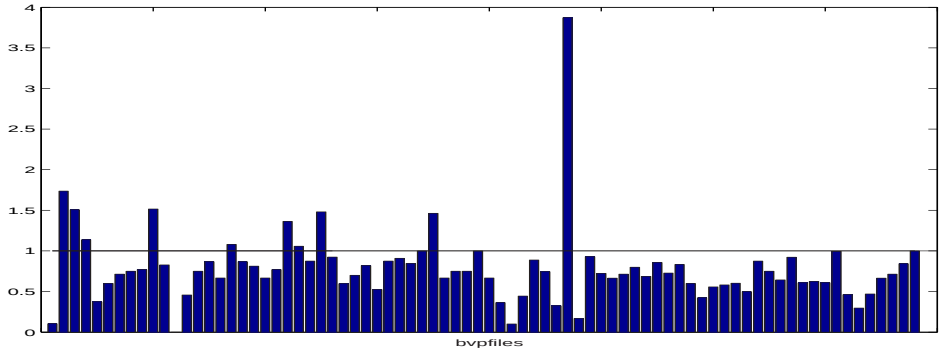


Figure 4.1: This bars above show Z_j for all 26 BVPS and the tolerances 10^{-3}, 10^{-6} and 10^{-8}.

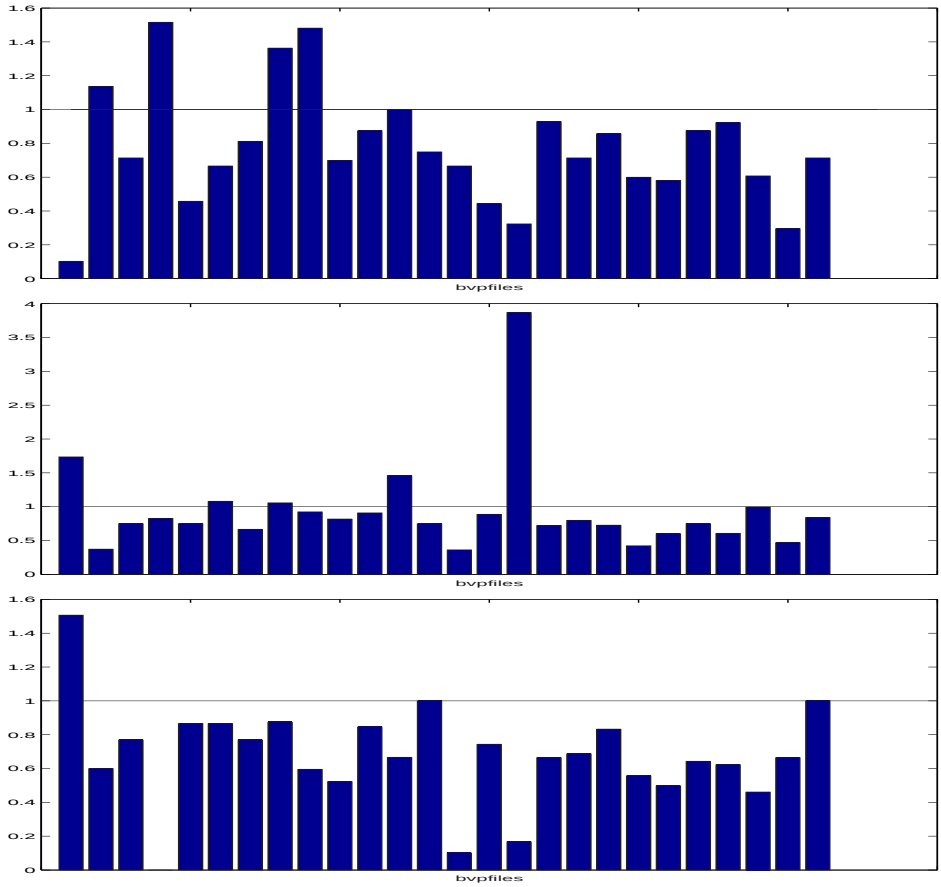


Figure 4.2: This graphs' bars show Z_j for all 26 BVPS and the tolerances 10^{-3}, 10^{-6} and 10^{-8} each in a different graph.

To conclude, the means of Z_j of all the bvpfiles are listed in Table 4.57.

overall	10^{-3}	10^{-6}	10^{-8}
0.8028	0.7735	0.9297	0.7013

Table 4.28: A final statistics of the improvement of SBVP 2.0 .

Consequently, Sbvp 2.0 is about 20 % faster overall, 23 % for the tolerance at 10^{-3}, 7 % for 10^{-6} and 30 % for the tolerance of 10^{-8}. The weak result for 10^{-6} is caused by Bvp 56. If it is eliminated from this statistic this leads to the final Table 4.29.

overall	10^{-3}	10^{-6}	10^{-8}
0.7625	0.7735	0.8121	0.7013

Table 4.29: A final statistic about the improvement of Sbvp 2.0 .

The excellent results are valid also for the tolerance 10^{-6}.

4.3 COLNEW

COLNEW is a modification of the package COLSYS by Ascher, Christiansen and Russell that incorporates a new basis representation replacing B-splines and improvements for the linear and nonlinear algebraic equation solvers. COLSYS is perhaps the best established code for solving boundary value problems in FORTRAN.

For further information about COLSYS refer to [1].

4.4 SBVP 2.0 vs. COLNEW

Analogous tests as in Section 4.2 are prepared in this section, but SBVP 1.0, SBVP 2.0 and COLNEW are assessed. Since COLNEW is not a MATLAB routine, but a FORTRAN solver, the test environment is changed. This is the only test series that was not performed on the Windows platform but on a Unix platform.

The problem in comparing COLNEW to the other routines is that the runtime, which was used as the major test characteristic throughout this study is not an option, since the FORTRAN code COLNEW cannot be compared with the MATLAB implementations in that respect. So the only measure that can be compared straight forwardly between the routines is the number of meshpoints. But this number is not as relevant as the runtime. In earlier tests we could see that runtime and number of meshpoints are not strictly related to each other. The second possibility for comparison are the fcounts and dfcounts but because these are not related via a constant formula, it is quite hard to quantify the effects of those measure for the overall performance of the codes.

Additionally, COLNEW does not specify the order of the method in relation to the requested tolerances, but uses the same polynomial degree throughout the test. For this study the collocation order of COLNEW is set to 6.

4.4.1 Test Results

The first goal is to compare the new version of SBVP to the old version. The results on the Windows platform show that SBVP 2.0 is a far better code. The second task is to compare SBVP 2.0 to COLNEW using the final number of meshpoints, the fcounts and the dfcounts. The results of the respective test series are given in Tables 4.30 - 4.55.

Bvp	solver	Tol	fct	dfct	N	Time
funsing015	Colnew	1e-008	450	450	40	—
funsing015	Sbvp 1.0	1e-008	525	526	32	1.580
funsing015	Sbvp 2.0	1e-008	957	958	26	1.430
funsing015	Colnew	1e-006	210	210	20	—
funsing015	Sbvp 1.0	1e-006	633	634	48	1.680
funsing015	Sbvp 2.0	1e-006	913	914	33	1.320
funsing015	Colnew	0.001	90	90	10	—
funsing015	Sbvp 1.0	0.001	503	504	76	1.400
funsing015	Sbvp 2.0	0.001	303	304	20	0.610

Table 4.30: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. The runtimes show that Sbvp 2.0 is much better than Sbvp 1.0. Since a meaningful comparison of the runtimes for Colnew and the Matlab codes is not possible, we focus on the mesh size, fcount and dfcount. For the strict tolerance the number of required meshpoints is smaller for Sbvp 2.0, yet the numbers of function and Jacobian evaluations is higher than for Colnew.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1001	Colnew	1e-008	90	90	10	—
funsing1001	Sbvp 1.0	1e-008	553	544	50	1.500
funsing1001	Sbvp 2.0	1e-008	1587	1588	58	2.010
funsing1001	Colnew	1e-006	90	90	10	—
funsing1001	Sbvp 1.0	1e-006	433	424	50	1.200
funsing1001	Sbvp 2.0	1e-006	1823	1824	60	2.110
funsing1001	Colnew	0.001	90	90	10	—
funsing1001	Sbvp 1.0	0.001	343	339	37	4.210
funsing1001	Sbvp 2.0	0.001	703	704	40	1.090

Table 4.31: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Colnew needs the least meshpoints, fcounts and dfcounts.

BVP	solver	Tol	fct	dfct	N	Time
funsing1002	COLNEW	1e-008	516	516	34	—
funsing1002	SBVP 1.0	1e-008	588	589	33	1.720
funsing1002	SBVP 2.0	1e-008	579	580	22	0.980
funsing1002	COLNEW	1e-006	390	390	20	—
funsing1002	SBVP 1.0	1e-006	234	235	23	0.930
funsing1002	SBVP 2.0	1e-006	955	956	35	1.330
funsing1002	COLNEW	0.001	90	90	10	—
funsing1002	SBVP 1.0	0.001	158	154	25	0.660
funsing1002	SBVP 2.0	0.001	483	484	23	0.980

Table 4.32: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. SBVP 2.0 is far better in the strict case concerning meshpoints, but COLNEW needs less fcounts and dfcounts.

BVP	solver	Tol	fct	dfct	N	Time
funsing1004	COLNEW	1e-008	210	210	20	—
funsing1004	SBVP 1.0	1e-008	228	229	15	0.920
funsing1004	SBVP 2.0	1e-008	453	454	15	0.830
funsing1004	COLNEW	1e-006	90	90	10	—
funsing1004	SBVP 1.0	1e-006	178	179	15	0.780
funsing1004	SBVP 2.0	1e-006	353	354	15	0.720
funsing1004	COLNEW	0.001	90	90	10	—
funsing1004	SBVP 1.0	0.001	68	69	8	0.550
funsing1004	SBVP 2.0	0.001	103	104	10	0.520

Table 4.33: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. First thing to see is that the new version is quicker than the old one. Second, COLNEW needs the highest number of meshpoints for the tolerance at 10^{-8}.

BVP	solver	Tol	fct	dfct	N	Time
funsing1005	COLNEW	1e-008	978	978	46	—
funsing1005	SBVP 2.0	1e-008	3711	3712	96	4.290
funsing1005	COLNEW	1e-006	684	684	22	—
funsing1005	SBVP 1.0	1e-006	2015	2006	226	4.750
funsing1005	SBVP 2.0	1e-006	3783	3784	120	4.080
funsing1005	COLNEW	0.001	270	270	10	—
funsing1005	SBVP 1.0	0.001	1883	1879	241	4.290
funsing1005	SBVP 2.0	0.001	5503	5504	240	5.770

Table 4.34: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. SBVP 1.0 fails for the strict tolerance. SBVP 2.0 has quite ambivalent results compared to SBVP 1.0 and is not competitive compared to COLNEW.

Bvp	solver	Tol	fct	dfct	N	Time
funsing15	Colnew	1e-008	450	450	40	—
funsing15	Sbvp 1.0	1e-008	444	445	24	1.400
funsing15	Sbvp 2.0	1e-008	957	958	26	1.440
funsing15	Colnew	1e-006	210	210	20	—
funsing15	Sbvp 1.0	1e-006	388	389	27	1.210
funsing15	Sbvp 2.0	1e-006	913	914	33	1.300
funsing15	Colnew	0.001	90	90	10	—
funsing15	Sbvp 1.0	0.001	518	519	42	1.490
funsing15	Sbvp 2.0	0.001	1023	1024	33	1.210

Table 4.35: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Again Colnew is not competitive concerning the number of meshpoints for the strict tolerance.

Bvp	solver	Tol	fct	dfct	N	Time
funsing1a	Colnew	1e-008	300	210	10	—
funsing1a	Sbvp 1.0	1e-008	843	934	10	1.440
funsing1a	Sbvp 2.0	1e-008	2713	354	10	1.060
funsing1a	Colnew	1e-006	270	180	10	—
funsing1a	Sbvp 1.0	1e-006	643	714	10	1.160
funsing1a	Sbvp 2.0	1e-006	2053	274	10	0.930
funsing1a	Colnew	0.001	300	150	10	—
funsing1a	Sbvp 1.0	0.001	203	229	5	0.660
funsing1a	Sbvp 2.0	0.001	1443	194	10	0.710

Table 4.36: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. This Bvp is solved on the starting grid and Sbvp 2.0 is better than Sbvp 1.0 for the tolerance 10^{-8}. Because of the low number of fcounts and dfcounts it is obvious that Colnew has the best behavior.

Bvp	solver	Tol	fct	dfct	N	Time
funsing2002	Colnew	1e-008	1260	1260	112	—
funsing2002	Sbvp 1.0	1e-008	849	850	51	2.100
funsing2002	Sbvp 2.0	1e-008	1803	1804	54	2.140
funsing2002	Colnew	1e-006	588	588	56	—
funsing2002	Sbvp 1.0	1e-006	843	844	66	1.960
funsing2002	Sbvp 2.0	1e-006	1893	1894	75	2.090
funsing2002	Colnew	0.001	252	252	28	—
funsing2002	Sbvp 1.0	0.001	353	349	39	1.070
funsing2002	Sbvp 2.0	0.001	1433	1434	80	1.780

Table 4.37: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Colnew has problems at the strict tolerance but performs best in other cases.

BVP	solver	Tol	fct	dfct	N	Time
funsing2008	COLNEW	1e-008	2688	2688	224	—
funsing2008	SBVP 1.0	1e-008	2353	2344	120	5.300
funsing2008	SBVP 2.0	1e-008	2919	2920	132	3.360
funsing2008	COLNEW	1e-006	1344	1344	112	—
funsing2008	SBVP 1.0	1e-006	2211	2202	153	4.690
funsing2008	SBVP 2.0	1e-006	4665	4666	158	4.710
funsing2008	COLNEW	0.001	462	462	36	—
funsing2008	SBVP 1.0	0.001	1253	1254	196	2.680
funsing2008	SBVP 2.0	0.001	3943	3944	238	4.090

Table 4.38: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. As SBVP 2.0 is optimized for high accuracy the good behavior for the tolerance at 10^{-8} is not surprising. COLNEW is not competitive in the hardest case, because the number of dfcounts is about 8 times the number of dfcounts for SBVP 2.0 and the number of meshpoints is twice the number of meshpoints there.

BVP	solver	Tol	fct	dfct	N	Time
funsing21a	COLNEW	1e-008	180	90	10	—
funsing21a	SBVP 1.0	1e-008	1013	1104	10	1.670
funsing21a	SBVP 2.0	1e-008	1953	434	10	1.020
funsing21a	COLNEW	1e-006	180	90	10	—
funsing21a	SBVP 1.0	1e-006	773	844	10	1.320
funsing21a	SBVP 2.0	1e-006	1633	334	10	0.890
funsing21a	COLNEW	0.001	180	90	10	—
funsing21a	SBVP 1.0	0.001	273	299	5	0.770
funsing21a	SBVP 2.0	0.001	1233	234	10	0.700

Table 4.39: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. SBVP 2.0 is obviously improved compared to the old version but not competitive with COLNEW.

Bvp	solver	Tol	fct	dfct	N	Time
funsing37c	Colnew	1e-008	300	210	10	—
funsing37c	Sbvp 1.0	1e-008	843	934	10	1.560
funsing37c	Sbvp 2.0	1e-008	3023	354	10	1.180
funsing37c	Colnew	1e-006	270	180	10	—
funsing37c	Sbvp 1.0	1e-006	643	714	10	1.240
funsing37c	Sbvp 2.0	1e-006	2353	274	10	1.040
funsing37c	Colnew	0.001	300	150	10	—
funsing37c	Sbvp 1.0	0.001	203	229	5	0.680
funsing37c	Sbvp 2.0	0.001	1593	194	10	0.790

Table 4.40: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Again the stricter the tolerance the better Sbvp 2.0 compared to Sbvp 1.0. But Colnew is even better.

Bvp	solver	Tol	fct	dfct	N	Time
funsing400	Colnew	1e-008	180	90	10	—
funsing400	Sbvp 1.0	1e-008	93	94	10	0.540
funsing400	Sbvp 2.0	1e-008	183	184	10	0.540
funsing400	Colnew	1e-006	180	90	10	—
funsing400	Sbvp 1.0	1e-006	73	74	10	0.490
funsing400	Sbvp 2.0	1e-006	143	144	10	0.480
funsing400	Colnew	0.001	180	90	10	—
funsing400	Sbvp 1.0	0.001	28	29	5	0.390
funsing400	Sbvp 2.0	0.001	103	104	10	0.360

Table 4.41: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. The results do not differ much.

Bvp	solver	Tol	fct	dfct	N	Time
funsing500	Colnew	1e-008	90	90	10	—
funsing500	Sbvp 1.0	1e-008	93	94	10	0.450
funsing500	Sbvp 2.0	1e-008	183	184	10	0.420
funsing500	Colnew	1e-006	90	90	10	—
funsing500	Sbvp 1.0	1e-006	73	74	10	0.420
funsing500	Sbvp 2.0	1e-006	143	144	10	0.390
funsing500	Colnew	0.001	90	90	10	—
funsing500	Sbvp 1.0	0.001	28	29	5	0.370
funsing500	Sbvp 2.0	0.001	103	104	10	0.330

Table 4.42: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Sbvp 2.0 is obviously better than Sbvp 1.0 , though only by a narrow margin.

BVP	solver	Tol	fct	dfct	N	Time
funsing54	COLNEW	1e-008	3396	1182	36	—
funsing54	SBVP 1.0	1e-008	588227	594052	2952	546.170
funsing54	SBVP 2.0	1e-008	327428	160384	1566	356.570
funsing54	COLNEW	1e-006	1020	390	20	—
funsing54	SBVP 1.0	1e-006	34367	34852	177	36.550
funsing54	SBVP 2.0	1e-006	27500	11215	149	14.470
funsing54	COLNEW	0.001	360	150	10	—
funsing54	SBVP 1.0	0.001	640	654	8	1.210
funsing54	SBVP 2.0	0.001	1093	234	10	1.140

Table 4.43: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. The improvement of SBVP 2.0 is striking, even for fcounts and dfcounts but performance compared to COLNEW is poor.

BVP	solver	Tol	fct	dfct	N	Time
funsing55	COLNEW	1e-008	180	90	10	—
funsing55	SBVP 1.0	1e-008	343	434	10	1.370
funsing55	SBVP 2.0	1e-008	433	354	10	0.920
funsing55	COLNEW	1e-006	180	90	10	—
funsing55	SBVP 1.0	1e-006	263	334	10	1.060
funsing55	SBVP 2.0	1e-006	333	274	10	0.740
funsing55	COLNEW	0.001	180	90	10	—
funsing55	SBVP 1.0	0.001	93	119	5	0.610
funsing55	SBVP 2.0	0.001	233	194	10	0.500

Table 4.44: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. Similarly as for various other problems, dfcounts are reduced in SBVP 2.0, but COLNEW still performs better.

BVP	solver	Tol	fct	dfct	N	Time
funsing56	COLNEW	1e-008	1530	660	40	—
funsing56	SBVP 1.0	1e-008	168333	48085	96	130.860
funsing56	SBVP 2.0	1e-008	20963	6934	80	16.190
funsing56	COLNEW	1e-006	810	420	20	—
funsing56	SBVP 1.0	1e-006	7623	1884	10	4.990
funsing56	SBVP 2.0	1e-006	25223	5634	80	12.270
funsing56	COLNEW	0.001	420	270	10	—
funsing56	SBVP 1.0	0.001	11292	3653	18	8.790
funsing56	SBVP 2.0	0.001	33623	5414	120	2.540

Table 4.45: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. Excellent results for COLNEW oppose inexplicably high values of meshpoints in the case of the tolerance at 10^{-6}.

Bvp	solver	Tol	fct	dfct	N	Time
funsing6001	Colnew	1e-008	840	420	20	—
funsing6001	Sbvp 1.0	1e-008	2843	2989	15	4.420
funsing6001	Sbvp 2.0	1e-008	5658	1119	15	2.240
funsing6001	Colnew	1e-006	480	300	10	—
funsing6001	Sbvp 1.0	1e-006	2013	2129	15	3.180
funsing6001	Sbvp 2.0	1e-006	4473	859	15	1.850
funsing6001	Colnew	0.001	450	270	10	—
funsing6001	Sbvp 1.0	0.001	701	747	8	1.530
funsing6001	Sbvp 2.0	0.001	1743	314	10	1.510

Table 4.46: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Colnew is far better than both versions of Sbvp when we look at the fcounts and dfcounts, though there are more points on the final grid for 10^{-8}.

Bvp	solver	Tol	fct	dfct	N	Time
funsing6002	Colnew	1e-008	240	150	10	—
funsing6002	Sbvp 1.0	1e-008	763	854	10	1.430
funsing6002	Sbvp 2.0	1e-008	1583	354	10	0.880
funsing6002	Colnew	1e-006	240	150	10	—
funsing6002	Sbvp 1.0	1e-006	583	654	10	1.140
funsing6002	Sbvp 2.0	1e-006	1203	274	10	0.750
funsing6002	Colnew	0.001	270	120	10	—
funsing6002	Sbvp 1.0	0.001	158	184	5	0.620
funsing6002	Sbvp 2.0	0.001	873	194	10	0.590

Table 4.47: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Finished on the starting grid, Sbvp 2.0 is much faster than Sbvp 1.0.

Bvp	solver	Tol	fct	dfct	N	Time
funsing7001b	Colnew	1e-008	300	180	10	—
funsing7001b	Sbvp 1.0	1e-008	843	934	10	1.420
funsing7001b	Sbvp 2.0	1e-008	1823	434	10	0.990
funsing7001b	Colnew	1e-006	270	150	10	—
funsing7001b	Sbvp 1.0	1e-006	643	714	10	1.140
funsing7001b	Sbvp 2.0	1e-006	1383	334	10	0.870
funsing7001b	Colnew	0.001	300	120	10	—
funsing7001b	Sbvp 1.0	0.001	178	204	5	0.610
funsing7001b	Sbvp 2.0	0.001	993	234	10	0.640

Table 4.48: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Sbvp 2.0 is fast, Colnew seems to be even faster.

BVP	solver	Tol	fct	dfct	N	Time
funsing71	COLNEW	1e-008	1728	666	32	—
funsing71	SBVP 1.0	1e-008	9079	8787	39	8.850
funsing71	SBVP 2.0	1e-008	11393	3299	45	4.170
funsing71	COLNEW	1e-006	948	408	16	—
funsing71	SBVP 1.0	1e-006	5443	5322	33	5.330
funsing71	SBVP 2.0	1e-006	4655	1270	28	1.960
funsing71	COLNEW	0.001	552	264	6	—
funsing71	SBVP 1.0	0.001	2031	1910	18	2.400
funsing71	SBVP 2.0	0.001	2658	599	14	1.310

Table 4.49: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. Very good results for SBVP 2.0, but not competitive with COLNEW.

BVP	solver	Tol	fct	dfct	N	Time
funsing73	COLNEW	1e-008	2670	960	40	—
funsing73	SBVP 1.0	1e-008	58146	31024	72	102.360
funsing73	SBVP 2.0	1e-008	69443	12164	120	40.020
funsing73	COLNEW	1e-006	1668	666	14	—
funsing73	SBVP 1.0	1e-006	77313	34933	173	112.660
funsing73	SBVP 2.0	1e-006	121223	16684	240	55.610
funsing73	COLNEW	0.001	1170	540	10	—
funsing73	SBVP 1.0	0.001	56739	26147	203	71.930
funsing73	SBVP 2.0	0.001	91963	11184	240	33.460

Table 4.50: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. Three times more meshpoints for 10^{-8} and much higher fcounts and dfcounts for SBVP 2.0 compared to COLNEW show a bad behavior of the MATLAB solver.

BVP	solver	Tol	fct	dfct	N	Time
funsing8001	COLNEW	1e-008	360	240	10	—
funsing8001	SBVP 1.0	1e-008	1173	1184	10	1.790
funsing8001	SBVP 2.0	1e-008	2013	514	10	1.110
funsing8001	COLNEW	1e-006	360	240	10	—
funsing8001	SBVP 1.0	1e-006	833	844	10	1.300
funsing8001	SBVP 2.0	1e-006	1673	394	10	0.950
funsing8001	COLNEW	0.001	390	210	10	—
funsing8001	SBVP 1.0	0.001	313	319	5	0.790
funsing8001	SBVP 2.0	0.001	1203	274	10	0.720

Table 4.51: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. COLNEW finishes on the same grid with a smaller number of fcounts and dfcounts.

Bvp	solver	Tol	fct	dfct	N	Time
funsing8002	Colnew	1e-008	1194	534	28	—
funsing8002	Sbvp 1.0	1e-008	2927	2954	16	3.840
funsing8002	Sbvp 2.0	1e-008	5188	1039	15	2.040
funsing8002	Colnew	1e-006	660	300	10	—
funsing8002	Sbvp 1.0	1e-006	2343	2370	16	2.980
funsing8002	Sbvp 2.0	1e-006	4203	799	15	1.730
funsing8002	Colnew	0.001	360	180	10	—
funsing8002	Sbvp 1.0	0.001	722	756	8	1.360
funsing8002	Sbvp 2.0	0.001	2233	234	10	1.360

Table 4.52: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Sbvp 2.0 is not competitive in fcounts and dfcounts, but the number of meshpoints is smaller.

Bvp	solver	Tol	fct	dfct	N	Time
funsing9001	Colnew	1e-008	990	990	80	—
funsing9001	Sbvp 1.0	1e-008	1029	1030	78	2.520
funsing9001	Sbvp 2.0	1e-008	993	994	45	1.320
funsing9001	Colnew	1e-006	510	510	40	—
funsing9001	Sbvp 1.0	1e-006	997	998	99	2.330
funsing9001	Sbvp 2.0	1e-006	2033	2034	81	2.230
funsing9001	Colnew	0.001	300	300	20	—
funsing9001	Sbvp 1.0	0.001	1868	1864	304	4.020
funsing9001	Sbvp 2.0	0.001	2343	2344	107	2.630

Table 4.53: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Though all three routines have similar numbers of fcounts and dfcounts for the strict tolerance, the results differ a lot in number of meshpoints.

Bvp	solver	Tol	fct	dfct	N	Time
funsing9002	Colnew	1e-008	1416	1416	92	—
funsing9002	Sbvp 1.0	1e-008	1677	1678	132	3.940
funsing9002	Sbvp 2.0	1e-008	2253	2254	69	2.600
funsing9002	Colnew	1e-006	450	450	40	—
funsing9002	Sbvp 1.0	1e-006	2131	2132	252	4.760
funsing9002	Sbvp 2.0	1e-006	2131	2132	92	2.370
funsing9002	Colnew	0.001	210	210	20	—
funsing9002	Sbvp 1.0	0.001	3763	3759	618	8.400
funsing9002	Sbvp 2.0	0.001	2753	2754	147	2.920

Table 4.54: Comparison between Sbvp 1.0, Sbvp 2.0 and Colnew. Another win for Colnew in function evaluations and Jacobian evaluations. Sbvp 2.0 satisfies the tolerances with the least number of meshpoints for tolerance 10^{-8}.

BVP	solver	Tol	fct	dfct	N	Time
funsing9003	COLNEW	1e-008	210	210	20	—
funsing9003	SBVP 1.0	1e-008	237	238	16	0.910
funsing9003	SBVP 2.0	1e-008	471	472	16	0.840
funsing9003	COLNEW	1e-006	90	90	10	—
funsing9003	SBVP 1.0	1e-006	416	417	30	1.230
funsing9003	SBVP 2.0	1e-006	773	774	27	1.130
funsing9003	COLNEW	0.001	90	90	10	—
funsing9003	SBVP 1.0	0.001	83	84	11	0.570
funsing9003	SBVP 2.0	0.001	103	104	10	0.510

Table 4.55: Comparison between SBVP 1.0, SBVP 2.0 and COLNEW. COLNEW is better for 10^{-3} and 10^{-6} but loses for 10^{-8} at least concerning the number of meshpoints.

4.4.2 Confirming the Improvements

Comparing the average runtimes using Z_j as described in Section 4.2.2 the following table is the result:

overall	10^{-3}	10^{-6}	10^{-8}
0.8301	0.8958	0.8933	0.6960

Table 4.56: A final statistic about the improvement of SBVP 2.0 .

The results from the Windows platform are somewhat confirmed. With the highest gain of about 30 % for the strictest tolerance, the overall gain in performance is about 17 %. In Figure 4.3 the relation between $T_{2.0}$ and $T_{1.0}$ can be seen.

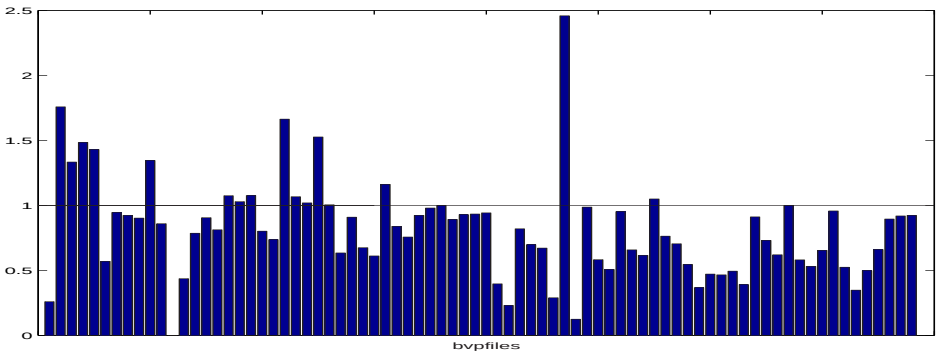


Figure 4.3: This graph's bars show Z_j for all 26 BVPS and the tolerances 10^{-3}, 10^{-6} and 10^{-8} on the Unix platform.

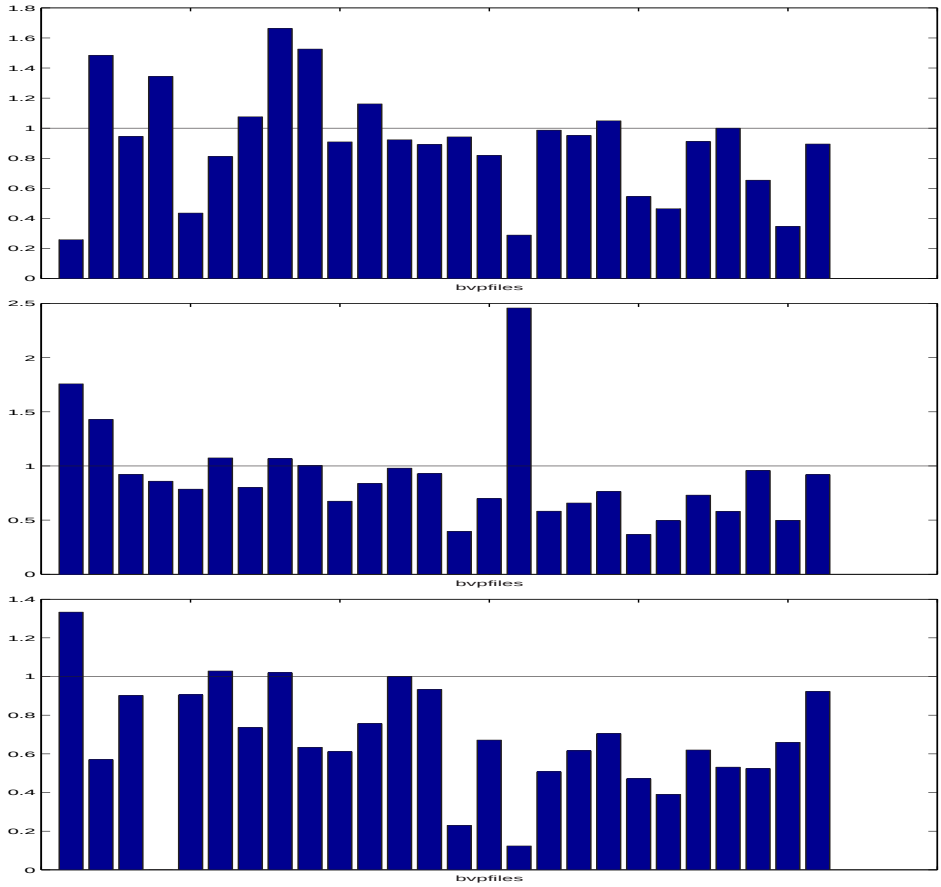


Figure 4.4: This graphs' bars show Z_j for all 26 BVPS and the tolerances 10^{-3}, 10^{-6} and 10^{-8} each in a different graph.

Figure 4.3 shows the same profile as the corresponding Figure 4.1 for the test series on the Windows platform, though the maverick at BVP 56 is not so high. If again BVP 56 is eliminated from the test this leads to the final table:

overall	10^{-3}	10^{-6}	10^{-8}
0.8250	0.9201	0.8307	0.7199

Table 4.57: A final statistics for the improvement of SBVP 2.0 .

This result is similar to the result on the Windows platform.

4.4.3 Positioning Against the FORTRAN Solver

COLNEW is compared to MATLAB solvers via the fcounts, dfcounts and the number of meshpoints on the final grid.

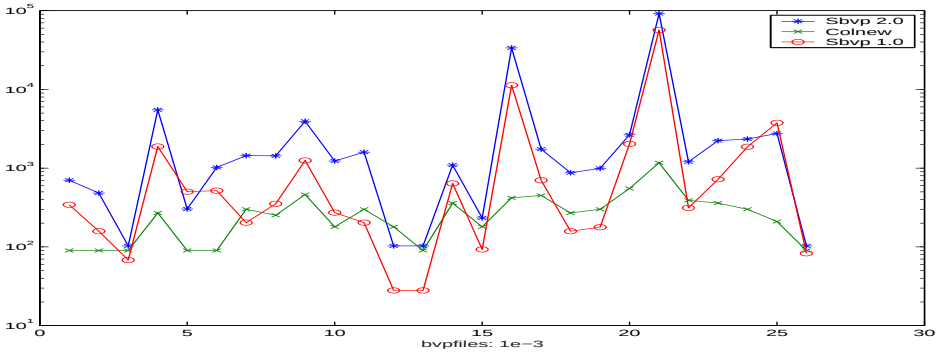


Figure 4.5: The graph shows the number of fcounts for all BVPS for tolerance 10^{-3}.

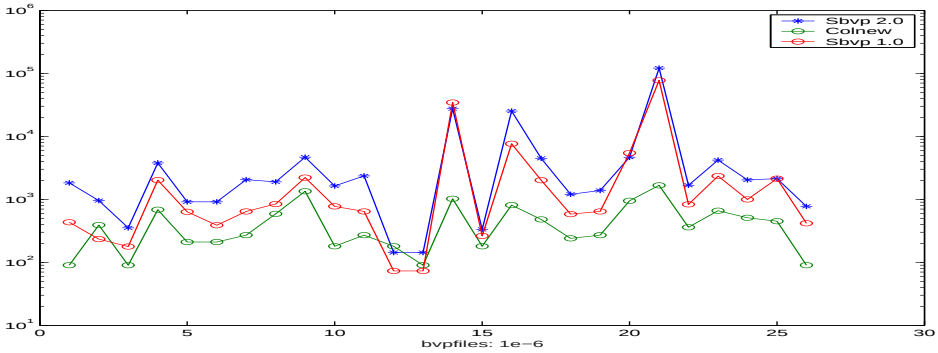


Figure 4.6: The graph shows the number of fcounts for all BVPS for tolerance 10^{-6}.

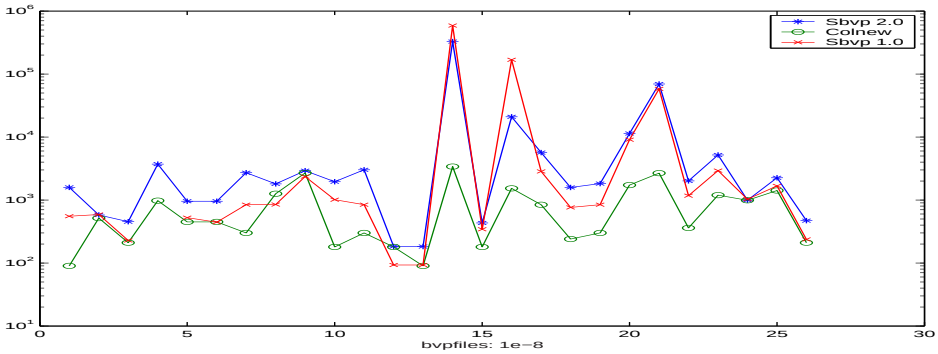


Figure 4.7: The graph shows the number of fcounts for all BVPS for tolerance 10^{-8}.

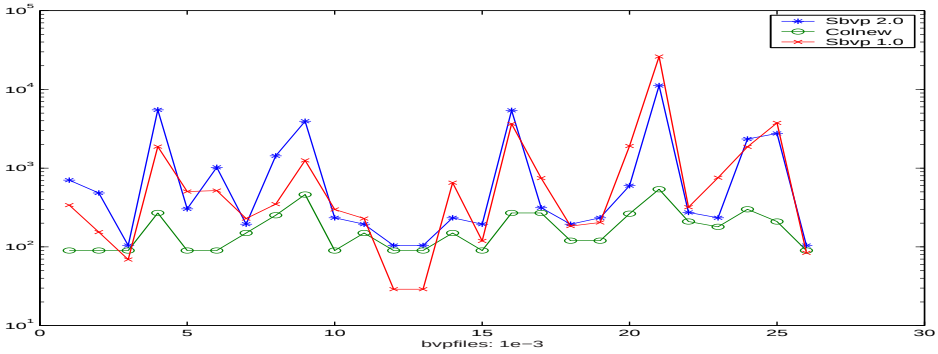


Figure 4.8: The graph shows the number of dfcounts for all BVPS for tolerance 10^{-3}.

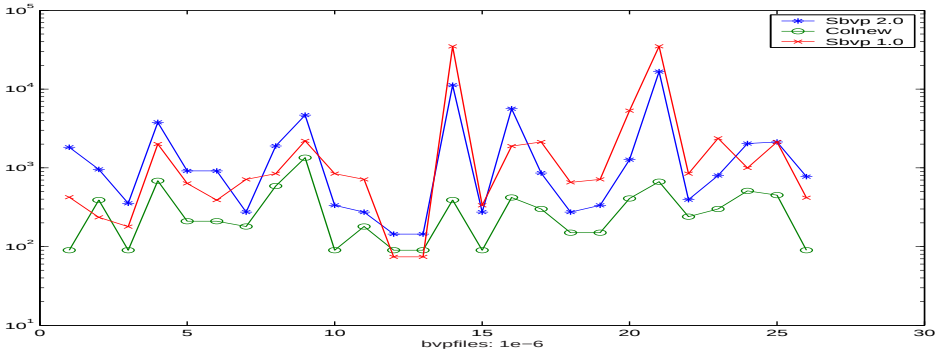


Figure 4.9: The graph shows the number of dfcounts for all BVPS for tolerance 10^{-6}.

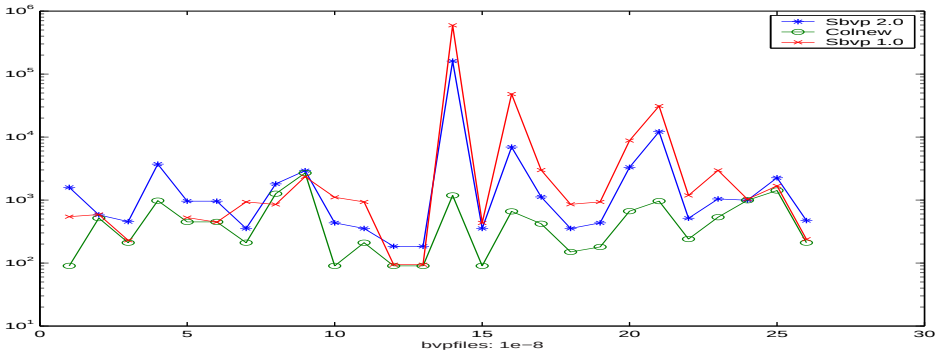


Figure 4.10: The graph shows the number of dfcounts for all BVPS for tolerance 10^{-8}.

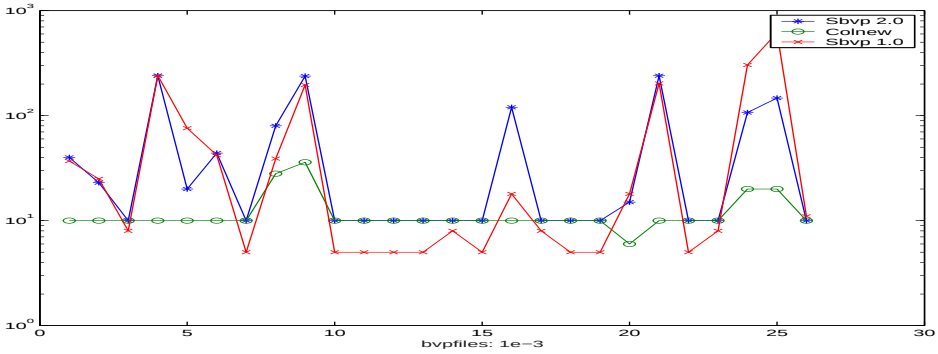


Figure 4.11: The graph shows the number of meshpoints for all BVPS for tolerance 10^{-3}.

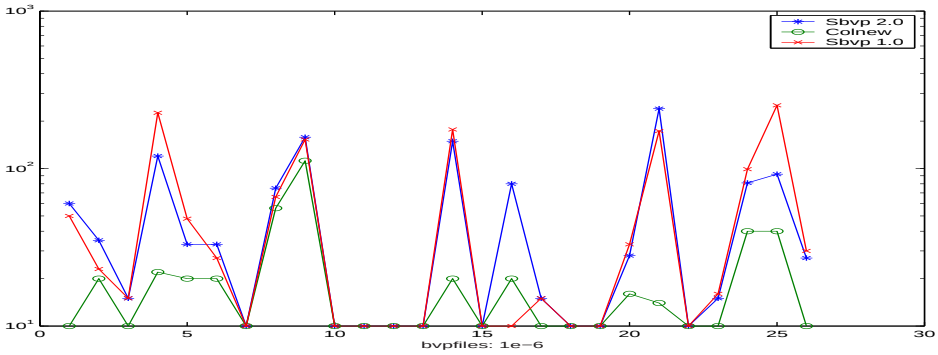


Figure 4.12: The graph shows the number of meshpoints for all BVPS for tolerance 10^{-6}.

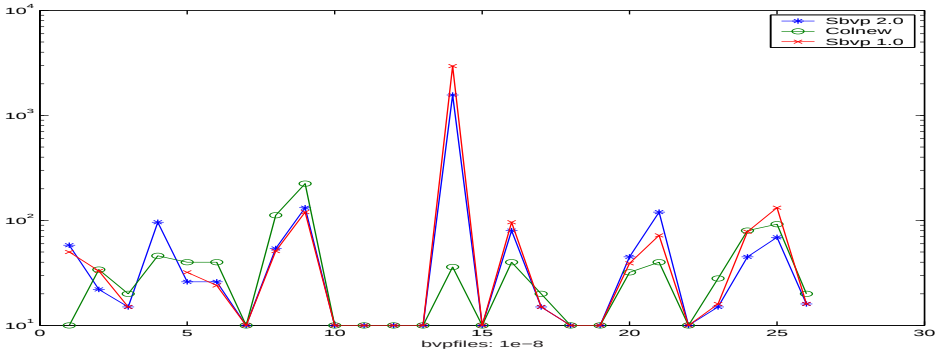


Figure 4.13: The graph shows the number of meshpoints for all BVPS for tolerance 10^{-8}.

Figures 4.5 - 4.13 show how the performance characteristics for SBVP 2.0 compare with COLNEW. The corresponding test results for SBVP 1.0 are listed in the figures too.

The line for COLNEW is quite low compared to those of the two versions of SBVP, independently of the different tolerances. It does not matter if fcounts, dfcounts or the number of meshpoints are compared. Figures 4.5 - 4.7 show that the number of fcounts of SBVP 2.0 has increased compared to SBVP 1.0. On the other hand the number of dfcounts has decreased for many BVPS, see Figures 4.8 - 4.10. Obviously COLNEW is the best choice, though especially in the case of tolerance 10^{-8} it requires the highest number of meshpoints for satisfying

the tolerances, cf. Figure 4.13.

For the highest tolerance the gap between COLNEW and SBVP 2.0 is smaller though the constantly good performance of the FORTRAN routine is impressive. For some problems however, SBVP 2.0 can compete with COLNEW, compare Figure 4.13.

The least improvement of SBVP 2.0 can be seen for the tolerance 10^{-3}. But as stated above, the tests in this study always aimed for stability and performance in the stricter configurations.

4.5 The Final Impression

Finally, we give a larger sample of comparisons of the runtime of the two versions of SBVP. Table 4.58 contains entries

$$T_{rel} := \frac{T_{2.0}}{T_{1.0}},$$

where T_i is the time the program took to solve the BVP from the bvpfile at the respective tolerance.

	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
Faster	11	18	19	16	19	23	23
Equal	3	3	2	5	2	0	0
Slower	12	5	5	5	5	3	3

Table 4.59: Absolute number of wins, Sbvp2.0 vs. Sbvp1.0

Tables 4.58 and 4.59 show how SBVP 2.0 compares with SBVP 1.0 for 6 different tolerances on 26 different BVPS. SBVP 2.0 is winning all comparisons and becomes better the stricter the tolerance.

So the evolution from SBVP 1.0 to SBVP 2.0 was a success leading to a faster routine that is working in a more stable manner, particularly for high accuracies.

	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
1001	0.6667	11.8500	0.4440	1.7273	6.0143	1.2000
1002	1.0000	0.8911	1.2625	1.2222	0.8235	0.4706
1004	0.5714	0.8571	0.6667	0.8571	0.8000	0.7778
1005	1.3937	Inf	Inf	Inf	Inf	Inf
015	0.4000	0.3867	0.5000	0.8067	0.5000	0.8609
15	0.8067	1.0000	0.9901	1.0000	0.5625	0.9286
1a	0.4678	0.8750	1.0000	0.7273	0.7692	0.7692
2002	1.2500	1.1313	0.8294	1.1047	0.7187	1.0048
2008	1.4797	1.3248	1.1667	1.0403	1.9000	0.6733
21a	0.7000	0.6931	1.0000	0.7500	0.6429	0.7092
37c	1.0000	0.8889	1.0000	0.9091	0.8462	0.7351
400	0.6667	0.8000	0.8000	0.8000	0.8333	0.5556
500	1.0000	1.0000	0.6000	0.6000	0.7500	1.3667
54	0.5333	0.6312	0.4918	0.3888	0.1168	0.1030
55	0.6250	0.7407	0.8889	0.8000	0.7500	0.6923
56	2.0349	2.9920	1.9234	3.7006	0.2988	0.1669
6001	0.6429	0.7778	0.9000	0.7359	0.7097	0.6677
6002	0.6667	0.6667	0.8750	0.7000	0.6612	0.6154
7001b	1.0143	0.7778	1.0000	0.7207	0.6923	0.7143
71	0.5909	0.5166	0.4722	0.4121	0.3881	0.5575
73	0.5493	0.5756	1.8135	0.5972	3.0796	0.4877
8001	0.8000	0.8000	0.9000	0.6154	0.6667	0.6111
8002	0.7143	0.8772	0.9444	0.6800	0.5919	0.6176
9001	10.5882	1.5700	1.4414	1.0000	0.7029	0.5000
9002	0.2793	0.5972	0.3333	0.4501	0.4546	0.6250
9003	0.5000	0.8108	1.0000	0.8333	0.9100	0.8000

Table 4.58: Relative Runtimes $\frac{T_{2.0}}{T_{1.0}}$

Chapter 5

Appendix

The appendix contains the problem formulations of the test examples considered in this study.

5.1 Models in Boundary Value Problems

5.1.1 BVPS 1001,1002,1004,1005

This BVP was used to show the performance of the mesh selection procedure in [8]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 1+\alpha^2 t^2 & 0 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ ct^{k-1}\mathbf{e}^{-\alpha t}(k^2-1-\alpha t(1+2k)) \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ c\mathbf{e}^{-\alpha} \end{pmatrix},$$

with $c = \left(\frac{\alpha}{k}\right)^k e^k$ and $\alpha = 360,\ k = 324$ for BVP 1001, $\alpha = 80,\ k = 16$ for BVP 1002, $\alpha = 40,\ k = 40$ for BVP 1004 and $\alpha = 400,\ k = 4$ for BVP 1005.

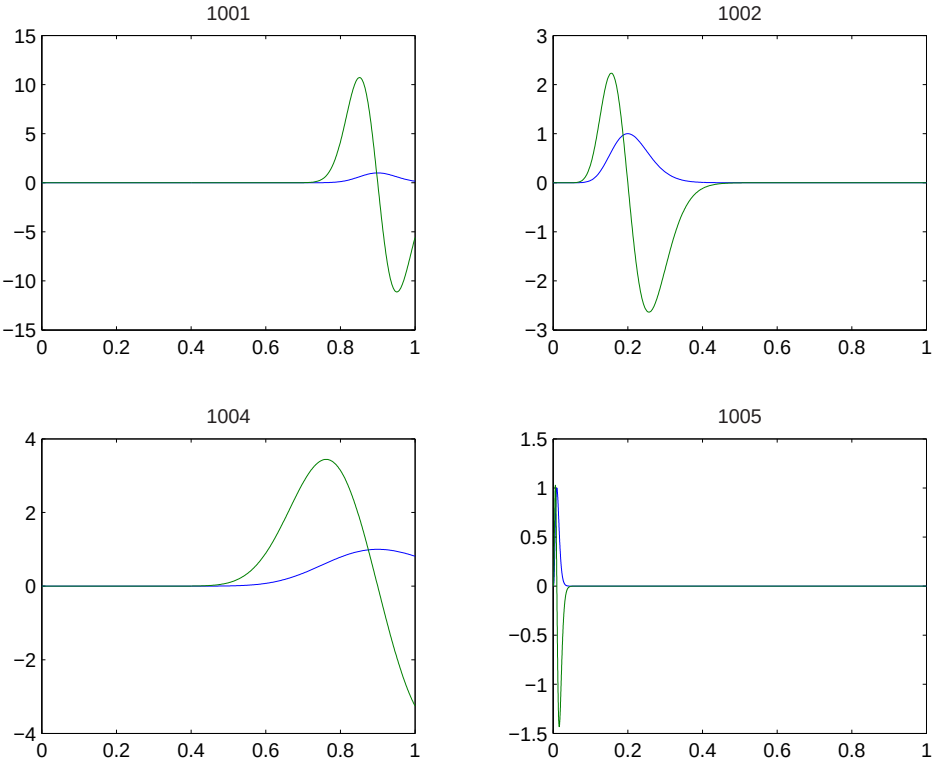


Figure 5.1: Solutions of BVPS 1001, 1002, 1004 and 1005

The exact solution of this BVP is given by

$$z(t) = (ct^k\mathbf{e}^{-\alpha t}, ct^k\mathbf{e}^{-\alpha t}(k-\alpha t))^T.$$

5.1.2 BVP 15

BVP 15 originated from [9]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ -100t^2 & 2 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ 1000t^2 + 10\cos(10t) - 10 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 10 - \sin(10) \\ 0 \end{pmatrix}.$$

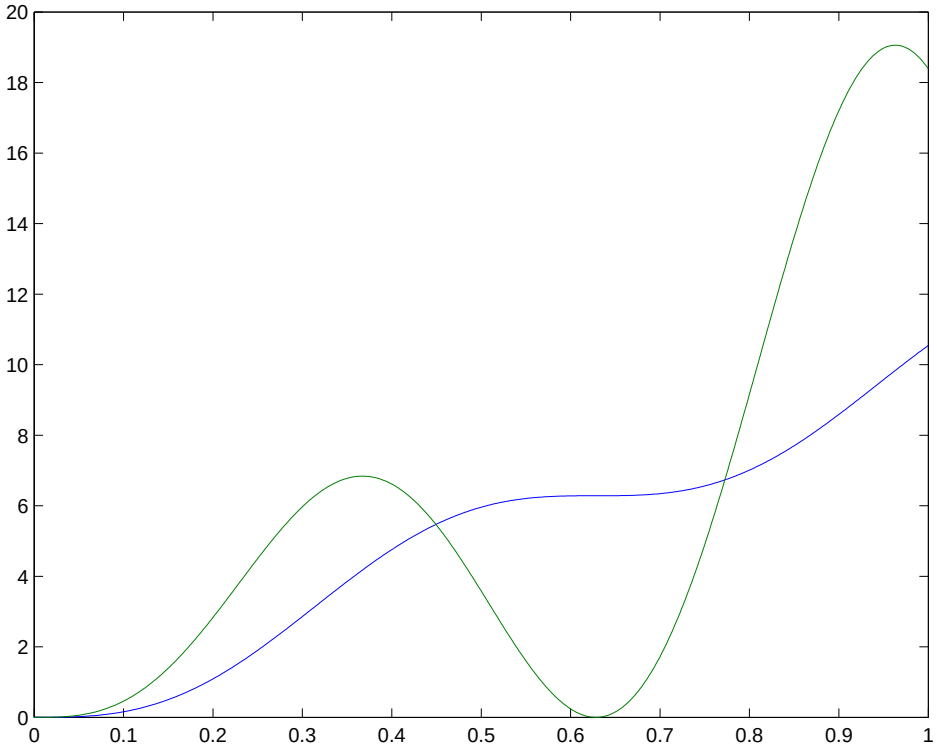


Figure 5.2: Solutions of BVPS 015 and 15

The exact solution of BVP 15 is given by

$$z(t) = (10t - \sin(10t), 10t - 10t\cos(10t))^T.$$

5.1.3 BVP 015

For BVP 015 the factor in the perturbation in the second line of BVP 15 was lowered from 1000 to 500:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ -100t^2 & 2 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ 500t^2 + 10\cos(10t) - 10 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 10 - \sin(10) \\ 0 \end{pmatrix}.$$

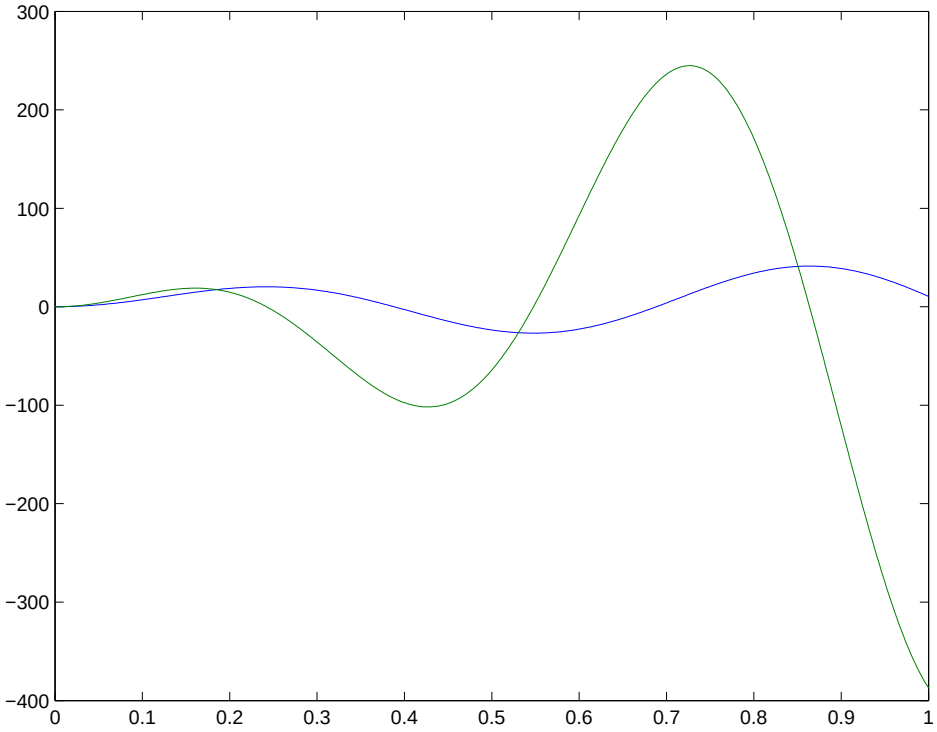


Figure 5.3: Solutions of BVPS 015 and 15

5.1.4 BVP 1a

BVP 1a originates from [10]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} z(t) - \begin{pmatrix} 0 \\ t z_1^5(t) \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ \frac{\sqrt{3}}{2} \end{pmatrix}.$$

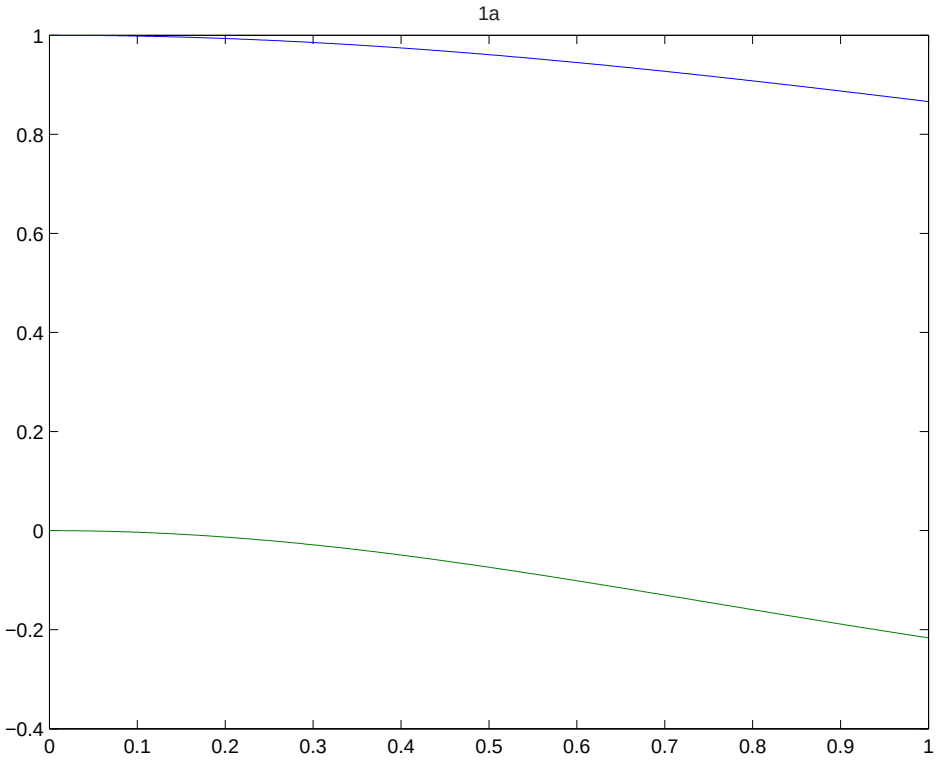


Figure 5.4: Solution of BVP 1a

The exact solution of the problem is given by

$$z(t) = \left(1/\sqrt{1+\frac{t^2}{3}}, -t^2/\left(3\sqrt{(1+\frac{t^2}{3})^3}\right)\right)^T.$$

5.1.5 BVPS 2002 and 2008

BVP 2002 was taken from [18]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 2 & 6 \end{pmatrix} z(t) - \begin{pmatrix} 0 \\ 4k^4t^5\sin(k^2t^2) + 10t\sin(k^2t^2) \end{pmatrix},$$
$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ \sin(k^2) \end{pmatrix},$$

with $k = 5$ for BVP 2002 and $k = 8$ for BVP 2008.

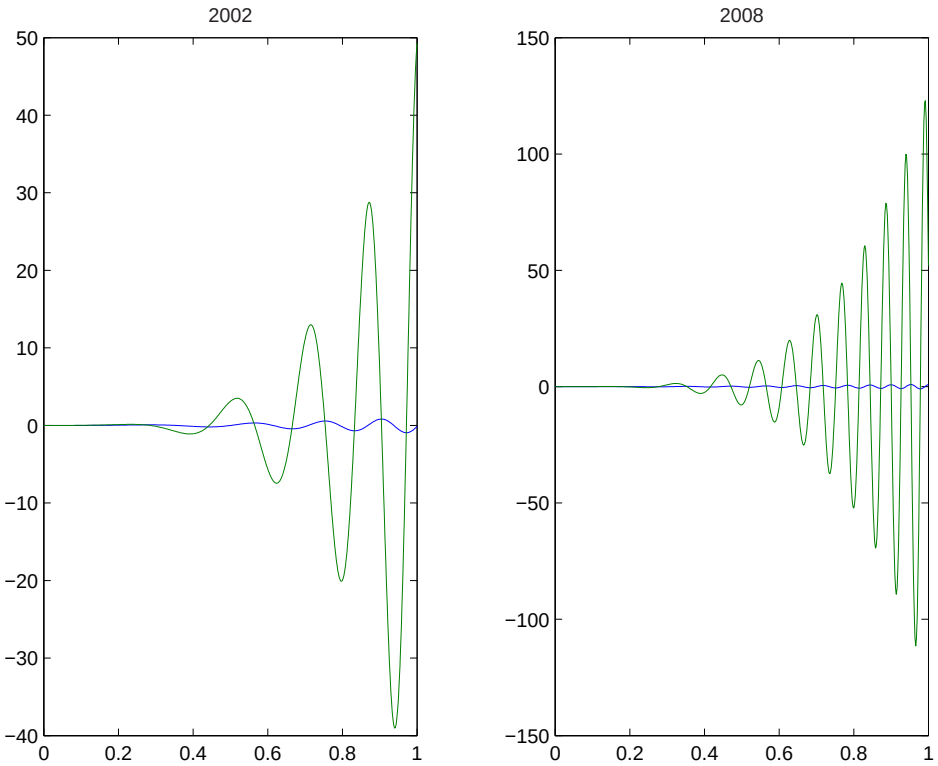


Figure 5.5: Solutions of BVPS 2002 and 2008

The exact solution to BVPS 2002 and 2008 is given by

$$z(t) = (t^2\sin(k^2t^2), 2k^2t^4\cos(k^2t^2) + 2t^2\sin(k^2t^2))^T.$$

5.1.6 Bvp 21a

This Bvp can be found in [15]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ -3tz_1^5(t) + tz_1^3(t) \end{pmatrix},$$
$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} z(0) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(1) = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix}.$$

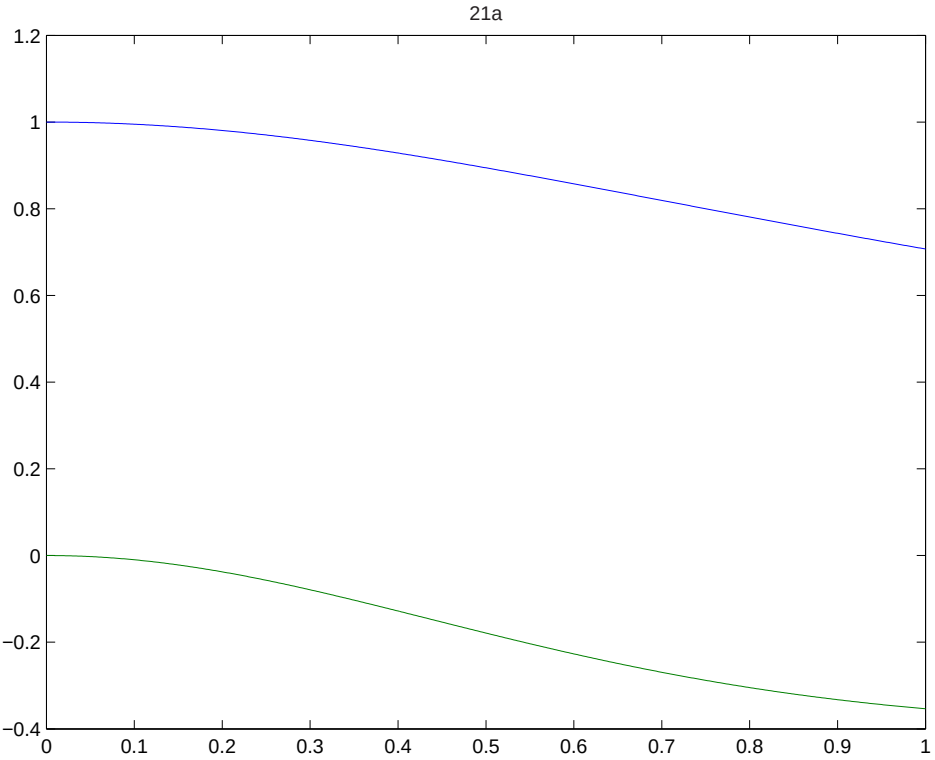


Figure 5.6: Solution of Bvp 21a

The exact solution to Bvp 21a is given by

$$z(t) = (1/\sqrt{1+t^2}, -t^2/\sqrt{(1+t^2)^3})^T.$$

5.1.7 BVP 37c

For BVP 37c see [7].

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ t\frac{-2(t^2+2)-8}{(t^2+2)^2} z_1^2(t) + \frac{8t^3}{(t^2+2)^2} z_1^3(t) \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} z(0) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(1) = \begin{pmatrix} \frac{1}{\ln(3)} \\ 0 \end{pmatrix}.$$

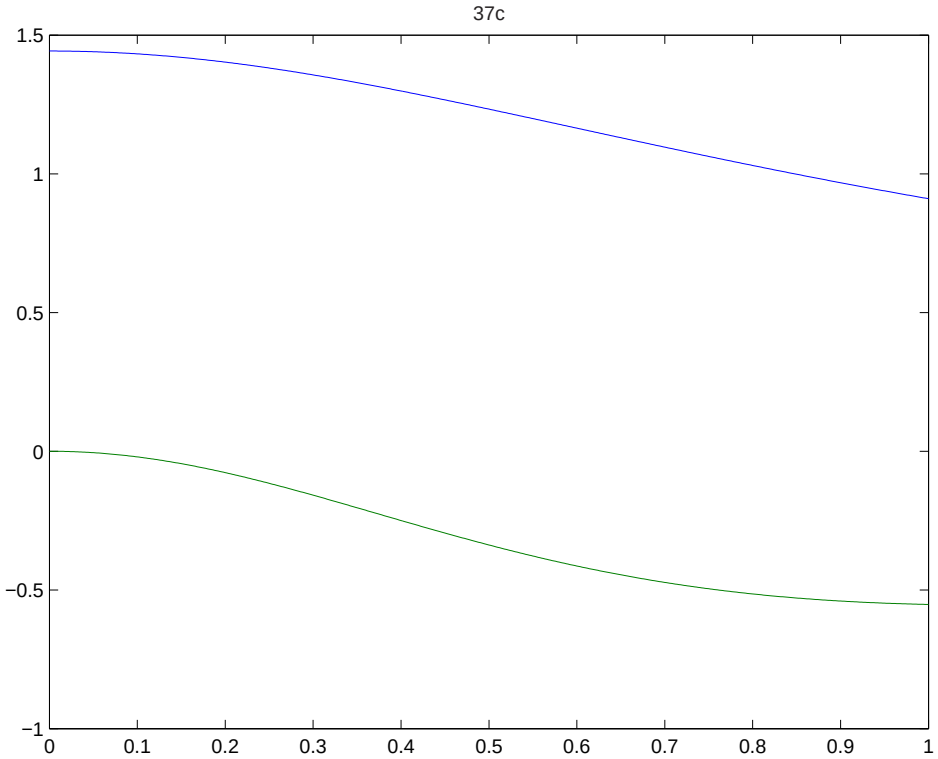


Figure 5.7: Solution of BVP 37c

The exact solution of BVP 37c is given by

$$z(t) = (1/\ln(t^2+2), -2t^2/(\ln^2(t^2+2)(t^2+2)))^T.$$

5.1.8 BVP 400

BVP 400 can be found in [22]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ -2p^2 & -3p \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ (6p^2+5p+1)t^p \end{pmatrix},$$

$$z(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

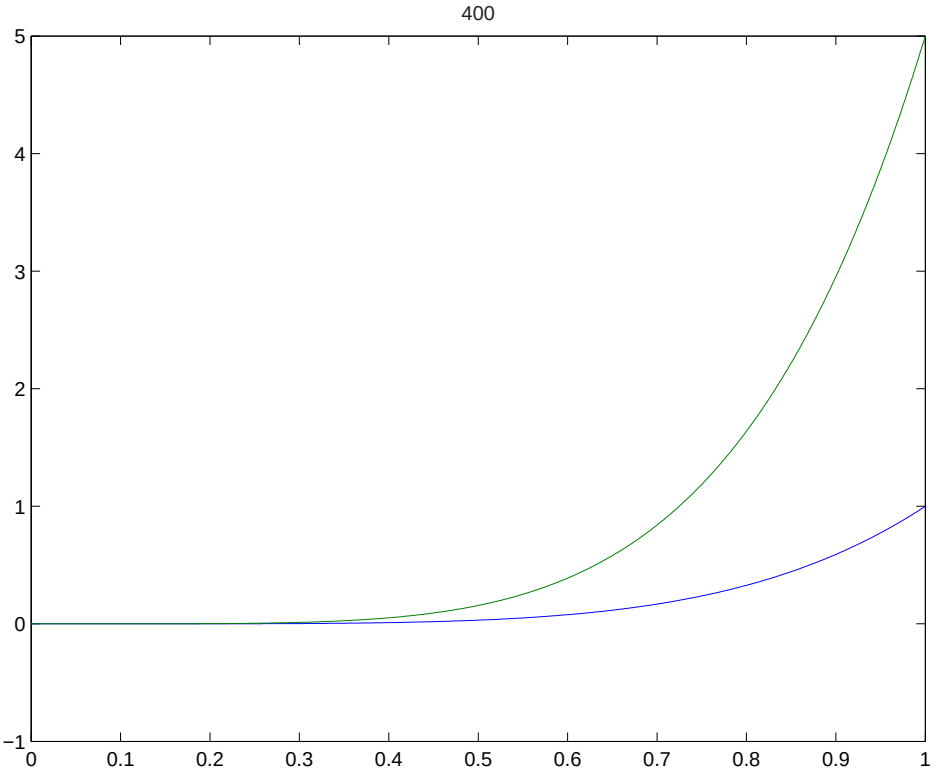


Figure 5.8: Solution of BVP 400

The exact solution of BVP 400 reads:

$$z(t) = (t^{p+1}, (p+1)t^{p+1})^T.$$

5.1.9 Bvp 500

Bvp 500 was taken from [11]:

$$z'(t) = -\frac{p}{t}z(t) + (2p+1)t^p,$$
$$z(0) = 0,$$

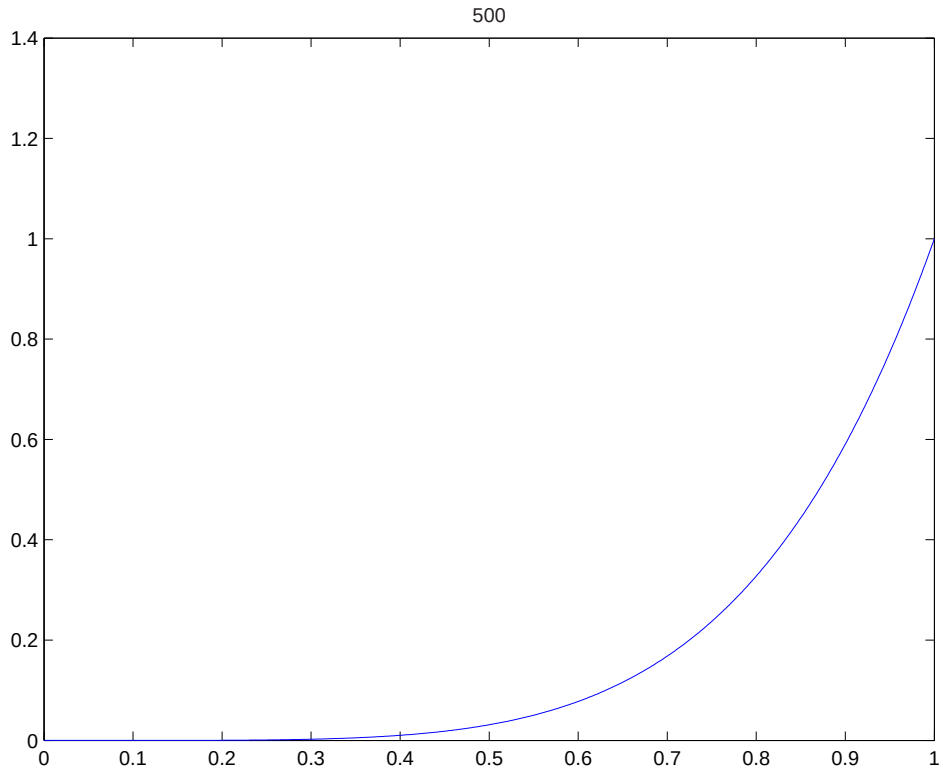


Figure 5.9: Solution of Bvp 500

and its solution is

$$z(t) = t^{p+1}.$$

5.1.10 BVP 54

BVP 54 was taken from [19]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ t^{1/2} z_1(t) \end{pmatrix},$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(b) = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

with $b = 5$.

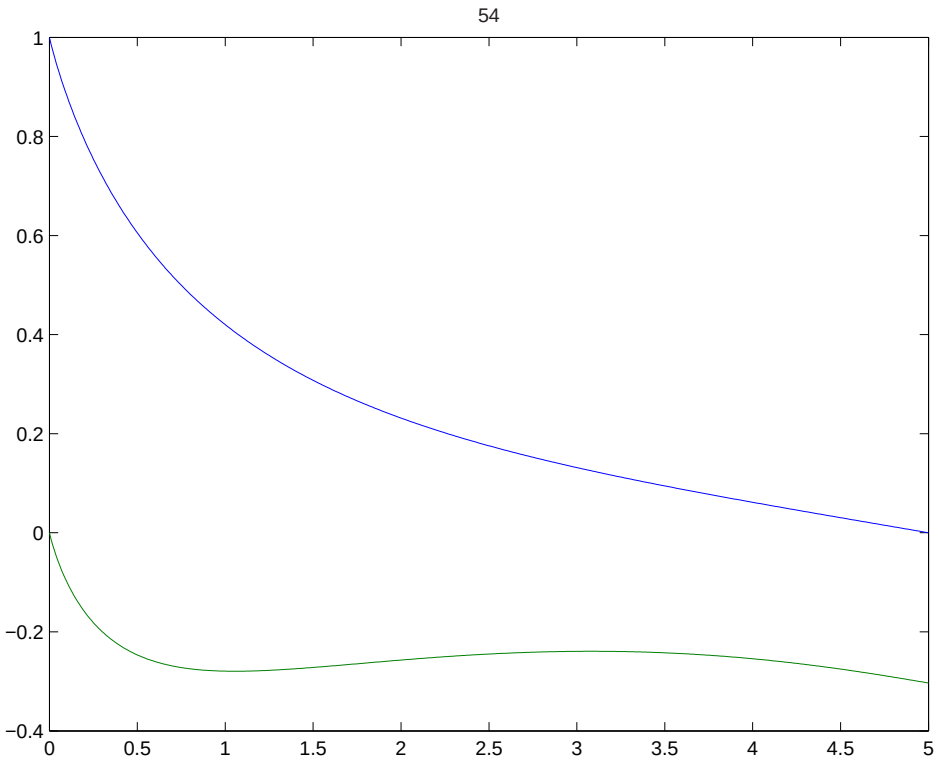


Figure 5.10: Solution of BVP 54

5.1.11 BVP 55

BVP 55 was formulated in [19]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} z(t) + t \begin{pmatrix} 0 \\ 0 \\ k^2 z_1(z_1^2 - 1 + z_2^2 - \frac{2}{kt} z_2) \\ z_1^2(z_2 - \frac{1}{kt}) \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} z(r) = \begin{pmatrix} 0 \\ 0 \\ R \\ 0 \end{pmatrix},$$

with $r = 10,\ k = 1,\ R = 1.$

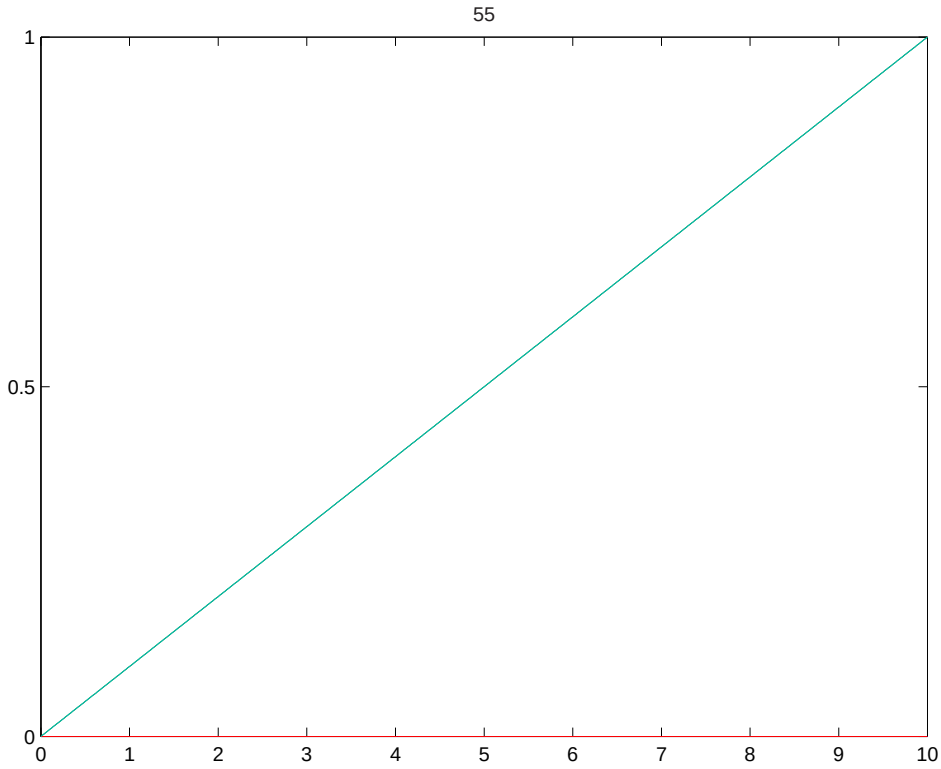


Figure 5.11: Solution of BVP 55

5.1.12 BVP 56

BVP 56 can be found in [19]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -t^2\mu^2 & -2 & 0 \\ t^2\mu^2 & 0 & 0 & -2 \end{pmatrix} z(t) + t\begin{pmatrix} 0 \\ 0 \\ z_1 z_2 - 2\gamma \\ -\frac{z_1^2}{2} \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & (1-\nu) & 0 & 1 \end{pmatrix} z(r) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

with $\nu = \frac{1}{3},\ \mu = 9,\ \gamma = 6010.$

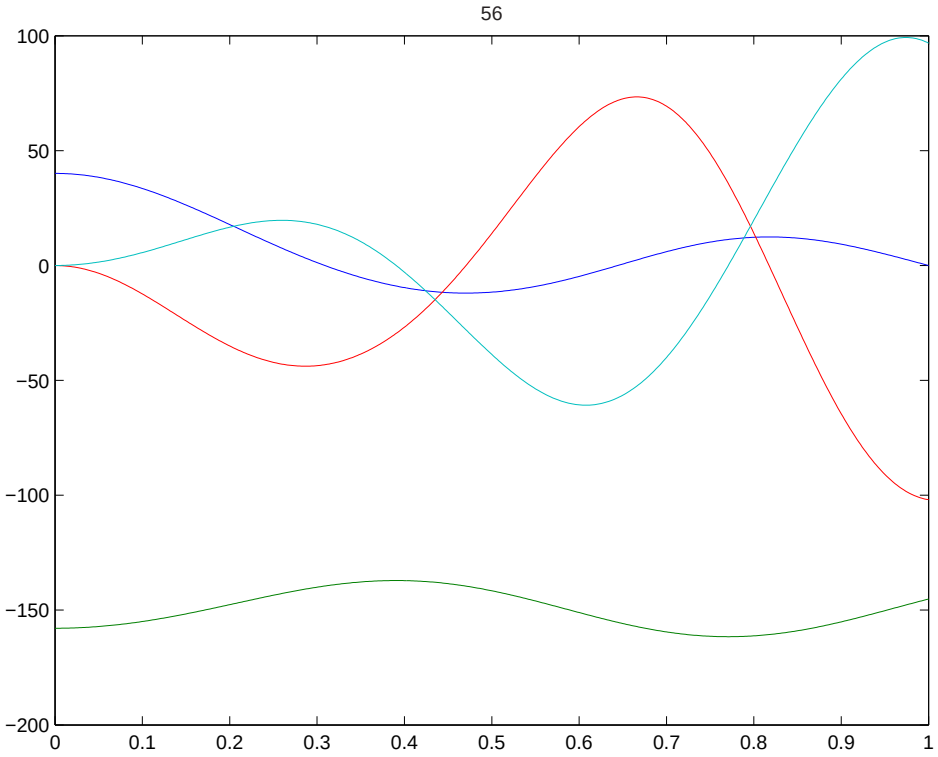


Figure 5.12: Solution of BVP 56

5.1.13 BVP 6001

BVP 6001 is a problem of second order and can be found in [5]. Instead of using the standard transformation $y(t) \to (y(t), y'(t))$ to obtain a system of first order, the Euler transformation was used $y(t) \to (y(t), ty'(t)) =: z(t)$, which yields:

$$z'(t) = \frac{1}{t} \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ t\phi^2 z_1(t) \exp\left(\frac{\gamma\beta(1-z_1(t))}{1+\beta(1-z_1(t))}\right) \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

For the parameters set to $\phi = 0.6,\ \gamma = 40,\ \beta = 0.2$, this problem has multiple solutions. Using the starting approximation $z(t) \equiv (0, 0)^T$, the numerical solution of the BVP obtained with SBVP 2.0 is shown in Figure 5.13.

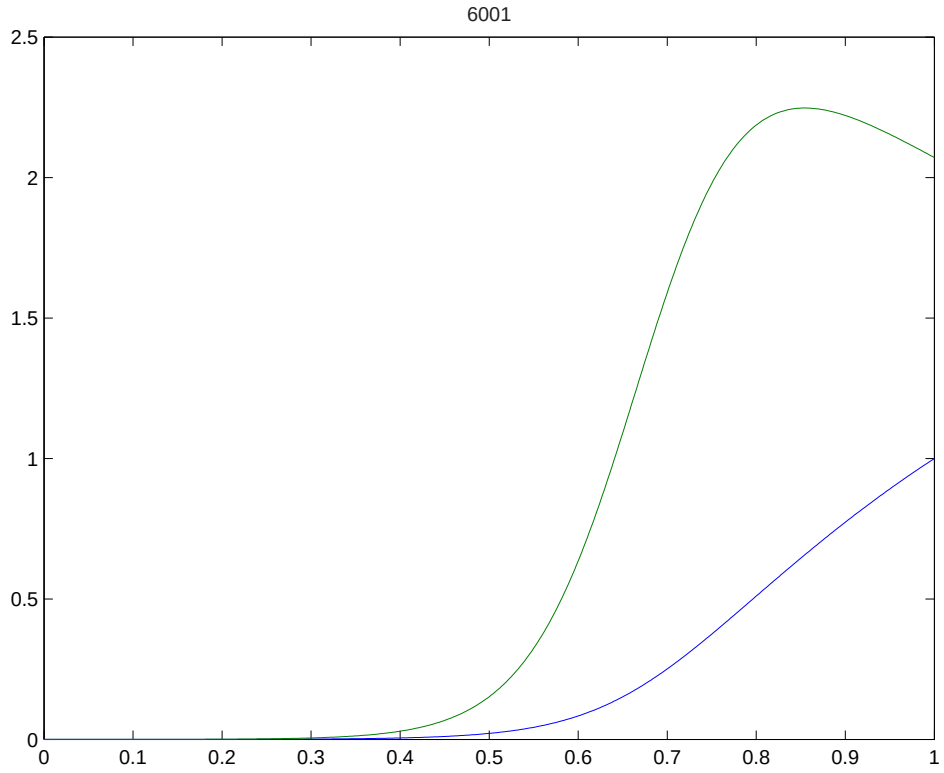


Figure 5.13: Solution of BVP 6001

5.1.14 BVP 6002

BVP 6002 is a modification of BVP 6001 with an altered right-hand side of the equation:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & t \\ 0 & -2 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ \phi^2 z_1(t) \exp\left(\frac{\gamma\beta(1-z_1(t))}{1+\beta(1-z_1(t))}\right) \end{pmatrix},$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

With the same parameters as for BVP 6001 (cf. 5.1.13), BVP 6002 has the solution shown in Figure 5.14, when using the starting approximation $z(t) \equiv (0, 0)^T$. Again this solution was obtained using SBVP 2.0.

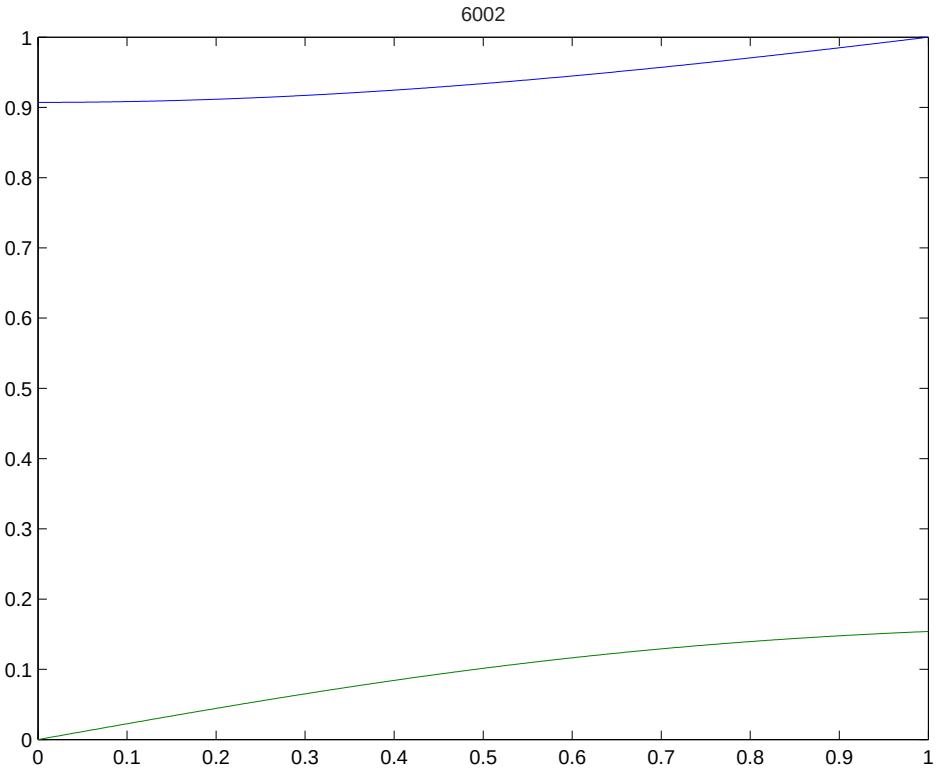


Figure 5.14: Solution of BVP 6002

5.1.15 Bvp 7001b

Bvp 7001b originates from [21]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 0 & \frac{1}{2} \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ \frac{z_1^3(t)}{2} \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} z(0) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(16) = \begin{pmatrix} \frac{1}{6} \\ 0 \end{pmatrix}.$$

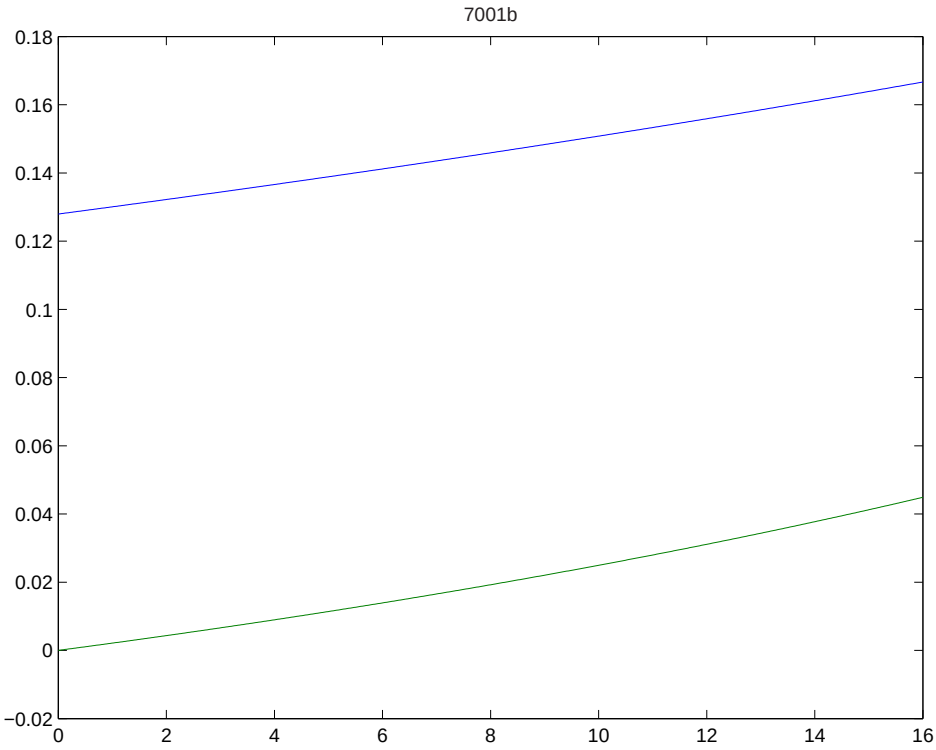


Figure 5.15: Solution of Bvp 7001b

5.1.16 BVP 71

BVP 71 was taken from [16]:

$$z''(t) = n \sinh(nz),$$
$$z(0) = 0, \quad z(1) = 1,$$

with n set to 4 in the tests. The BVP was transformed via the standard transformation into a system of first order.

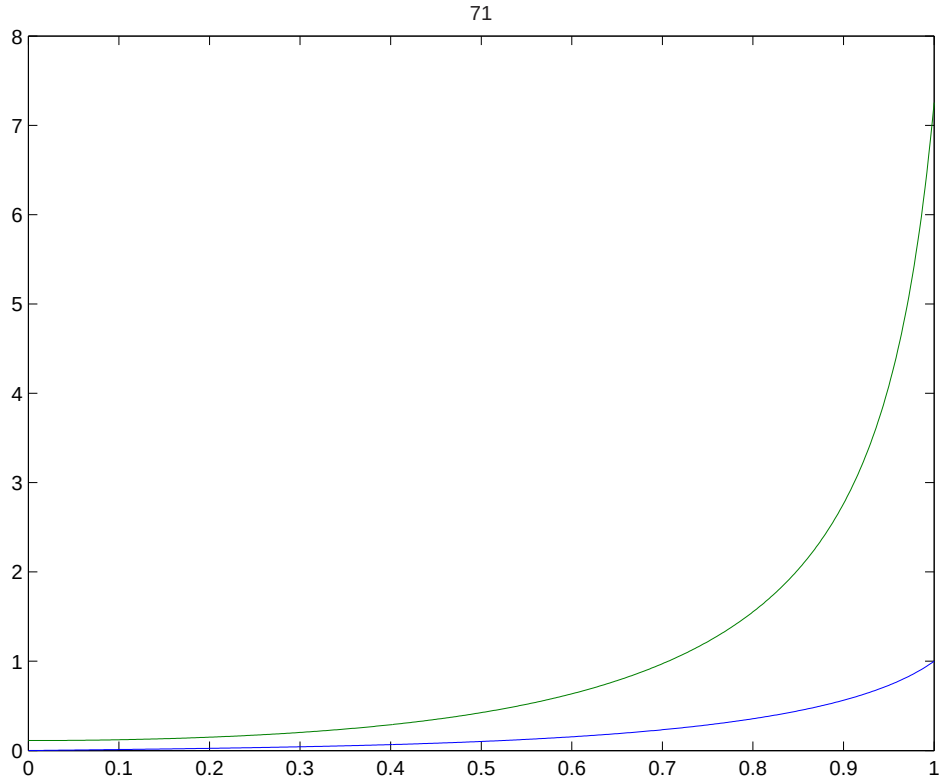


Figure 5.16: Solution of BVP 71

5.1.17 BVP 73

BVP 73 was also taken from [16]:

$$z'(t) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ Pe_H.\beta & 0 & Pe_H & 0 \\ 0 & 0 & 0 & Pe_M \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ 0 \\ \Phi \\ \Psi \end{pmatrix},$$

$$\begin{pmatrix} Pe_H & 0 & -1 & 0 \\ 0 & Pe_M & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} z(1) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

with

$$\Phi = Pe_H(\beta\theta_C - BDa(1 - z_2)\exp(\frac{z_1}{1 + \frac{z_1}{\gamma}}),$$

$$\Psi = -Pe_M Da(1 - z_2)\exp(\frac{z_1}{1 + \frac{z_1}{\gamma}}),$$

$\gamma = 20,\ B = 20,\ Pe_M = Pe_H = 291.3,\ Da = 0.05826,\ \beta = 0,$ and $\theta_C = 1.$

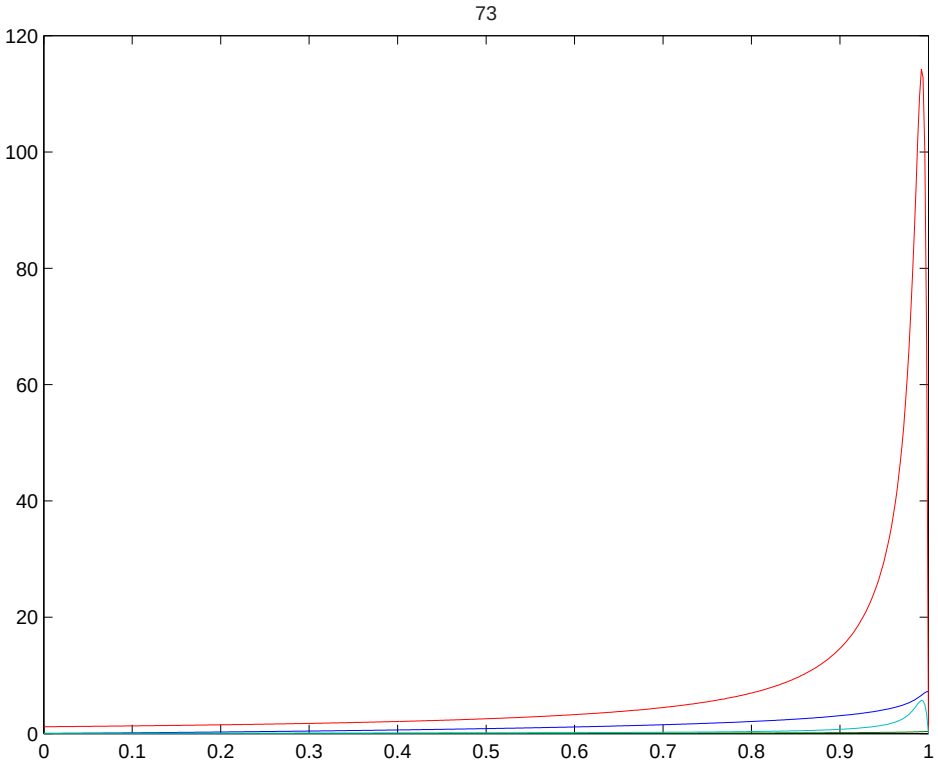


Figure 5.17: Solution of BVP 73

5.1.18 BVPS 8001 and 8002

These BVPS can be found in [21]:

$$z'(t) = \frac{1}{t}\begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} z(t) + \begin{pmatrix} 0 \\ \frac{tz_1(t)}{\varepsilon(z_1(t)+k)} \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} z(0) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} z(1) = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

with $k = 0.1$ and $\varepsilon = 10^{-1}$ for BVP 8001 and $\varepsilon = 10^{-2}$ for BVP 8002.

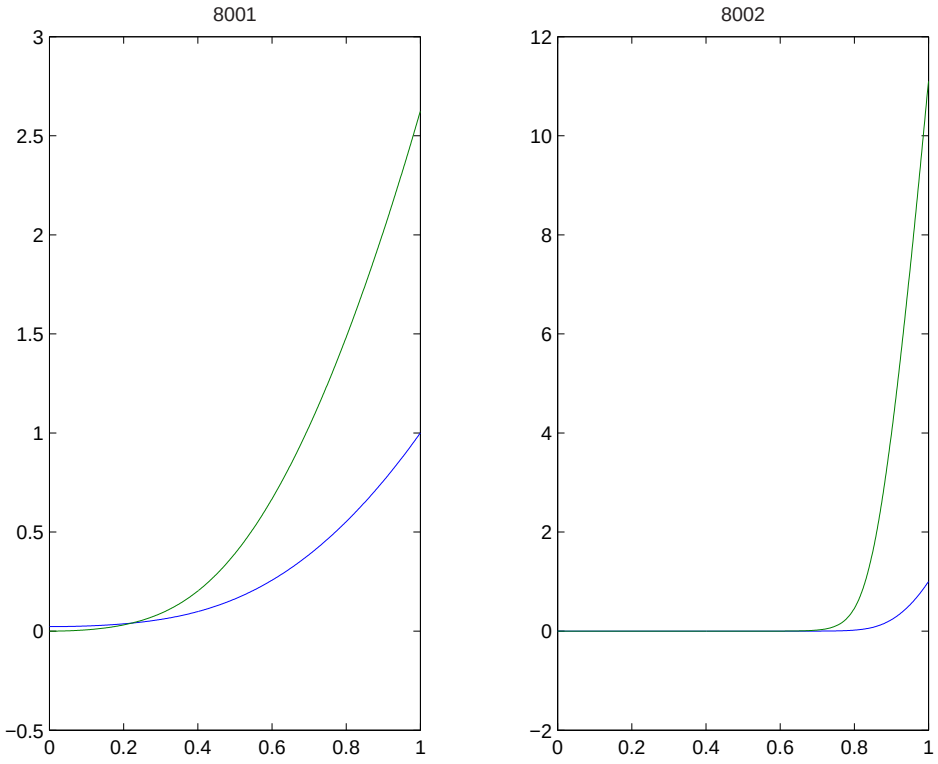


Figure 5.18: Solutions of BVPS 8001 and 8002

5.1.19 BVPS 9001, 9002 and 9003

These singular perturbed BVPS have very steep solutions, see Figure 5.19:

$$z'(t) = \begin{pmatrix} 0 & \mu\frac{t-\frac{1}{2}}{\epsilon} \\ 1 & 0 \end{pmatrix} z(t),$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} z(1) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

with $\mu = 1,\ \epsilon = 0.02,\ \alpha = 1,\ \beta = 3$ for BVP 9001, $\mu = -1,\ \epsilon = 0.02,\ \alpha = 1,\ \beta = 3$ for BVP 9002 and $\mu = -1,\ \epsilon = 0.1,\ \alpha = 1,\ \beta = 3$ for BVP 9003.

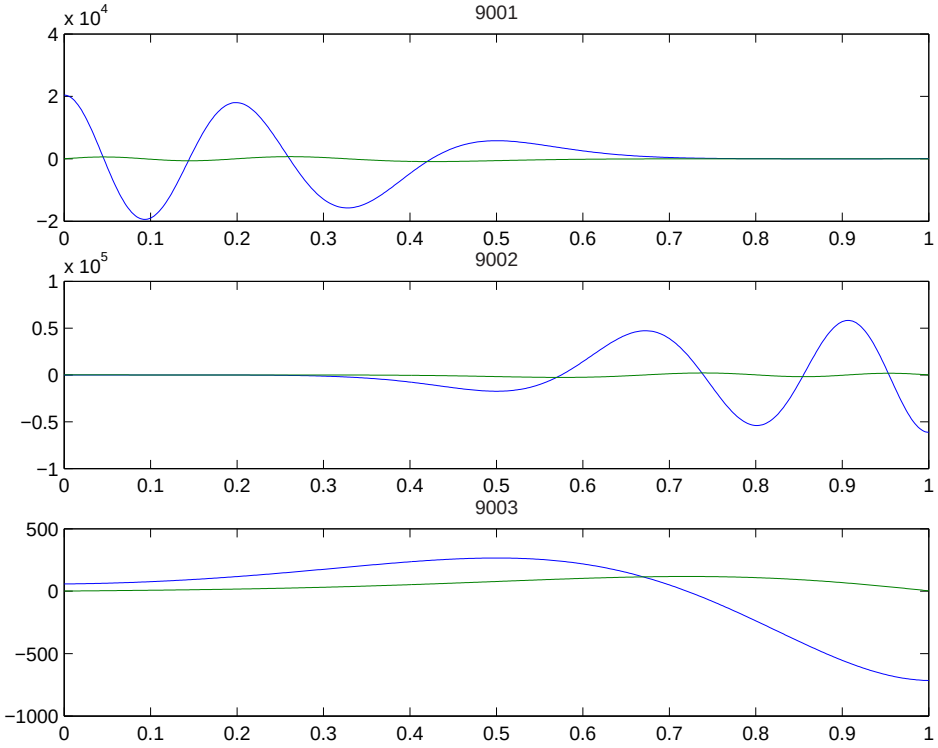


Figure 5.19: Solutions of BVPS 9001, 9002 and 9003

5.1.20 BVPS 9004 and 9005

These BVPS were taken from [3]. The second order equations have been transformed to first order systems using the standard transformation $y(t) \to (y(t), y'(t))$:

$$z'(t) = \begin{pmatrix} \mu \frac{t-\frac{1}{2}}{\epsilon} & 0 \\ 1 & 0 \end{pmatrix} z(t),$$

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} z(0) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} z(1) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

With $\mu = 1,\ \epsilon = 0.1,\ \alpha = 1,\ \beta = 3$ we have BVP 9004 which is a singularly perturbed problem with a boundary layer at both endpoints. With $\mu = -1,\ \epsilon = 0.02,\ \alpha = 1,\ \beta = 3$ we obtain BVP 9005 with a sharp interior layer.

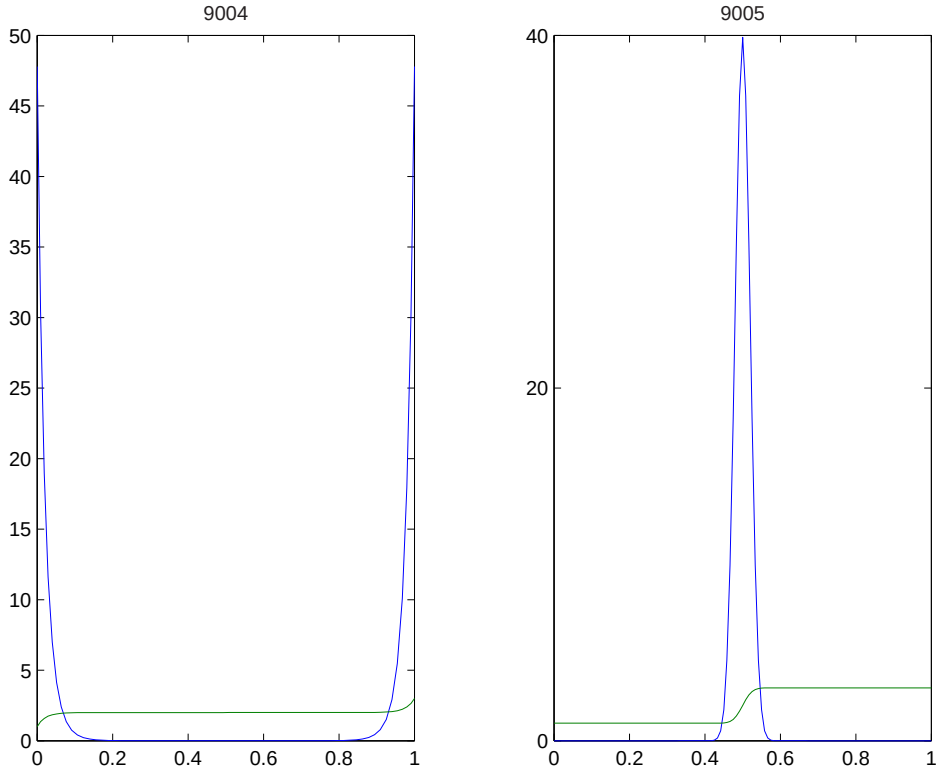


Figure 5.20: Solutions of BVPS 9004 and 9005

List of Tables

List of Figures

References

[1] U. Ascher, J. Christiansen, and R. Russell, *A collocation solver for mixed order systems of boundary values problems*, Math. Comp., 33 (1978), pp. 659–679.

[2] ——, *Collocation software for boundary value ODEs*, ACM Transactions on Mathematical Software, 7 (1981), pp. 209–222.

[3] W. Auzinger, *Numerical Solutions of Ordinary Differential Equations*, Inst. for Appl. Math. and Numer. Anal., Vienna Univ. of Technology, Austria, 1999.

[4] W. Auzinger, G. Kneisl, O. Koch, and E. Weinmüller, *SBVP 1.0 — A Matlab Solver for Singular Boundary Value Problems*, Inst. for Appl. Math. and Numer. Anal., Vienna Univ. of Technology, Austria, 2002. Also available as ANUM Preprint Nr. 2/02 at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.

[5] ——, *A solution routine for singular boundary value problems*, Techn. Rep. ANUM Preprint Nr. 1/02, Inst. for Appl. Math. and Numer. Anal., Vienna Univ. of Technology, Austria, 2002. Available at `http://www.math.tuwien.ac.at/~inst115/preprints.htm`.

[6] ——, *A collocation code for boundary value problems in ordinary differential equations*, Numer. Algorithms, 33 (2003), pp. 27–39.

[7] W. Auzinger, O. Koch, P. Kofler, and E. Weinmüller, *The application of shooting to singular boundary value problems*, Techn. Rep. Nr. 126/99, Inst. for Appl. Math. and Numer. Anal., Vienna Univ. of Technology, Austria, 1999. Available at `http://fsmat.at/~othmar/research.html`.

[8] W. Auzinger, O. Koch, and E. Weinmüller, *Efficient collocation schemes for singular boundary value problems*, Numer. Algorithms, 31 (2002), pp. 5–25.

[9] J. Cash and H. Silva, *On the numerical solution of a class of singular two-point boundary value problems*, J. Comput. Appl. Math., 45 (1993), pp. 91–102.

[10] F. d. Hoog and R. Weiss, *Difference methods for boundary value problems with a singularity of the first kind*, SIAM J. Numer. Anal., 13 (1976), pp. 775–813.

[11] ——, *Collocation methods for singular boundary value problems*, SIAM J. Numer. Anal., 15 (1978), pp. 198–217.

[12] ——, *The numerical solution of boundary value problems with an essential singularity*, SIAM J. Numer. Anal., 16 (1979), pp. 637–669.

[13] ——, *An approximation theory for boundary value problems on infinite intervals*, Computing, 24 (1980), pp. 227–239.

[14] ——, *On the boundary value problem for systems of ordinary differential equations with a singularity of the second kind*, SIAM J. Math. Anal., 11 (1980), pp. 41–60.

[15] ——, *The application of Runge-Kutta schemes to singular initial value problems*, Math. Comp., 44 (1985), pp. 93–103.

[16] M. Kubíček, V. Hlaváček, and M. Holodniok, *Test examples for comparison of codes for nonlinear boundary value problems in ordinary differential equations*, in Codes for Boundary-Value Problems in Ordinary Differential Equations, B. Childs, ed., vol. 76 of Lecture Notes in Computer Science, Berlin-Heidelberg, 1979, Springer Verlag, pp. 325–346.

[17] P. Markowich and C. Ringhofer, *Collocation methods for boundary value problems on "long" intervals*, Math. Comp., 40 (1983), pp. 123–150.

[18] W. Polster, *Ein Algorithmus zur Gittersteuerung bei Kollokationsverfahren für singuläre Randwertprobleme*, Master Thesis, Inst. for Appl. Math. and Numer. Anal., Vienna Univ. of Technology, Austria, 2001.

[19] P. Rentrop, *Eine Taylorreihenmethode zur numerischen Lösung von Zwei-Punkt Randwertproblemen mit Anwendung auf singuläre Probleme der nichtlinearen Schalentheorie.* TUM-MATH-7733. Technische Universität München 1977.

[20] L. Shampine, *Singular boundary value problems in odes*, Appl. Math. Comput., 138 (2003), pp. 99–112.

[21] L. Shampine, J. Kierzenka, and M. Reichelt, *Solving Boundary Value Problems for Ordinary Differential Equations in* Matlab *with* `bvp4c`, 2000. Available at `ftp://ftp.mathworks.com/pub/doc/papers/bvp/`.

[22] E. Weinmüller, *Collocation for singular boundary value problems of second order*, SIAM J. Numer. Anal., 23 (1986), pp. 1062–1095.

[23] W. Auzinger, O. Koch, E. Weinmüller, *Efficient Mesh Selection for Collocation Methods Applied to Singular BVPs*, submitted to J. Comput. Appl. Math.