Testing an Algorithm for an Adaptive Wavelet Based Method in Mathematica and Comparing the Method with NDSolve

M. Schuchmann and M. Rasguljajew from the Darmstadt University of Applied Sciences

Abstract

M. Schuchmann developed in 2013 and 2014 an adaptive wavelet based algorithm to numerically solve differential equations. This algorithm can be applied to ODEs with various orders and even to PDEs and differential-algebraic equations (DAEs). In this Paper we compare the solutions of boundary value problems of second order ODEs calculated by the adaptive wavelet algorithm with the solution of the Mathematica function NDSolve. In NDSolve we have different explicit or implicit integration methods (like implicit or explicit Runge-Kutta, Adams or BDF method) and it will automatically reduce the step size or change the method depending on the problem, so NDSolve can handle stiff and non-stiff problems.

We have implemented the algorithm in a Mathematica module and applied several tests. The used wavelet collocation method is costly, like collocation method are costly in general. But for stiff or unstable problems it has an advantage and for these cases explicit methods cannot be used. We used the Shannon wavelet in our module, but it can be modified for other wavelets and the ODE can be given in implicit form.

Introduction

Our module can be used for ODEs of order 1 and 2. It can easily be adjusted for higher order ODEs. It uses an adaptive algorithm to adjust the parameters of the wavelet based method.

The method uses an approximation function \( y_j \) constructed with a subset of the bases of \( V_j \). With that approximation function we minimize the sum of squares of the residuals at so called collocation points \( t_i \). For a certain number of collocation points this method is equivalent to the collocation method using that ‘wavelet bases’. The initial points or boundary values are integrated in the minimization process, but they could also be used as restraints.

Here we have to adjust several parameters like the solution index \( j \), the number of bases functions out of \( V_j \) and the number and the position of the collocation points \( t_i \). The adjustment of these parameters is the objective of the adaptive algorithm. For the construction of the algorithm theoretical investigations have been necessary and M. Schuchmann has used an error estimation (see [16]) which is based on the criteria \( Q_a \) (see (2)) and he also evaluated many simulation to use the information of the minimum of \( Q \) (see (1)) and \( Q_a \) in order to adjust the parameters.

The advantage of the wavelet collocation method is that like other collocation method it can also be applied to stiff differential equations. Moreover, it can even be used for non-stable problems and even for DAEs. As an approximation we not only get points but an approximation function. Compared to collocation methods, for example, based on polynomials (see [3]), we get one approximation function for the whole approximation Interval and it can also cover a larger interval. The method can also be used for implicit ODEs.
These are the advantages of a wavelet collocation method as well as using the approximation for extrapolation outside the original approximation interval. A disadvantage is that these methods, like many boundary value methods or implicit methods, are very costly. But the method is robust an it can applied to stiff or instable systems as well to implicit ODEs or DAEs.

Theory

In the wavelet theory a scaling function $\phi$ is used, which belongs to a MSA (multi scale analysis). From the MSA we know, that we can construct an orthonormal basis of a closed subspace $V_j$, where $V_j$ belongs to a the sequence of subspaces with the following property:

$$... cV_{-1} \subseteq V_0 \subseteq V_1 \subseteq ... \subseteq L^2(R),$$

$\{\phi_{j,k}(t)\}_{k \in \mathbb{Z}}$ is an orthonormal basis of $V_j$ with $\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - k)$. We use the following approximation function

$$y_j(t) := \sum_{k = k_{\text{min}}}^{k_{\text{max}}} c_k \cdot \phi_{j,k}(t) \quad \text{with } \phi \in C^s(R).$$

$k_{\text{max}}$ and $k_{\text{min}}$ depend on the approximation interval $[t_0, t_{\text{end}}]$ (see [14]). $s$ is the order of the ODE.

Now we can approximate the solution of an initial value problem $y' = f(y,t)$ and $y(t_0) = y_0$ by minimization of the following function

$$(1) \quad Q(c) = \sum_{i=1}^{m} \|y_j'(t_i) - f(y_j(t_i), t_i)\|_2^2 + \|y_j(t_0) - y_0\|_2^2 .$$

For $m = |k_{\text{max}} - k_{\text{min}}|$ we get an equivalent problem:

$$y_j'(t_i) = f(y_j(t_i), t_i) \quad \text{for } i = 1, 2, ..., m \quad \text{and} \quad y_j(t_0) = y_0 .$$

This is the classical collocation method, whereas the minimum residual method is more general.

We will use equidistant points or collocation points $t_i$ with $t_i = t_0 + i \cdot h$ and

$$h = \frac{t_{\text{end}} - t_0}{m} .$$

To detect large residuals in other places than the collocation points, we have a further value used for comparison with $Q_{\text{min}}$ (here in $y_j$ the vector $c$ will be set to the value in the minimum of $Q$, see (1)).

$$(2) \quad Q_a = \sum_{i=1}^{m} \|y_j'(\tau_i) - f(y_j(\tau_i), \tau_i)\|_2^2 + \|y_j(t_0) - y_0\|_2^2 .$$
with \( \tau_i = t_0 + i \frac{h}{a} \). \( m_a = a \cdot m \) with \( a > 1 \) as an integer. Since the wavelet collocation method provides a whole approximation function \( y_j \) and not only points, we can calculate \( Q_a \) without additional effort. If \( Q_a \gg Q_{\min} \) (and \( Q_{\min} \) was very small) then \( m \) (the number of collocation points) should be increased. When comparing \( Q_{\min} \) with \( Q_a \), \( Q_a \) should be weighted by \( 1/a \) if \( a \) is large. In the simulations \( a = 2 \) proved sufficient.

\( Q_a \) can additionally be justified by an error estimation of the residuals at theoretically any number of points. This was derived by M. Schuchmann (see [16]). In this error estimate a certain value occurs as a factor. \( Q_a \) represents the Riemann sum for this value i.e. this can be approximated by \( Q_a \).

If we have a second Order ODE

\[
F(y''', y', y, t) = 0
\]

with boundary conditions

\[
y(t_0) = y_0 \\
\text{and} \\
y(t_{\text{end}}) = y_{\text{end}}
\]

In the following example, we minimize

\[
Q(c) = \sum_{i=1}^{m} \left\| F(y_j'''(t_i), y_j''(t_i), y_j'(t_i), y_j(t_i), t_i) \right\|^2 + \left\| y_j'(t_0) - y_{0}' \right\|^2 + \left\| y_j(t_{\text{end}}) - y_{\text{end}} \right\|^2.
\]

Analogous we treat conditions of the form

\[
y(t_0) = y_0 \\
\text{and} \\
y'(t_0) = y_0'
\]

\( t_0 \) in the conditions (boundary or initial value) must not be the same as \( t_0 \) from the approximation interval, that means we can even address the conditions \( y(\tilde{t}_0) = y_0 \) and \( y'(\tilde{t}_0) = y_0' \) with \( \tilde{t}_0 \neq t_0 \).

**The Modules**

We have two modules: one for the wavelet collocation method (WCollocationS2) and one for the algorithm (WCollocationS2Alg). The first module is for the minimization of \( Q \) with certain \( j, k_{\max} \) and \( r \). The second module applies the algorithm to calculate the ‘optimal’ \( j, k_{\max} \) and \( r \). Here we can solve numerically first order ODEs with initial values, where the ODE can be given implicit (or explicit). The module needs only the left side of the equation

\[
f(y'(t), y(t), t) = 0 \text{ or } y'(t) - f(y(t), t) = 0.
\]

Analogous we can solve numerically ODEs of second order with initial values or boundary values.
In the module for the algorithm (WCollocationS2Alg) we have implemented the following algorithm:

We only need specify \( k_{\text{max}}^{(0)} \) (as a positive integer). \( k_{\text{min}} \) will then be adjusted for the approximation interval. If the approximation interval \( I = [t_0, t_{\text{end}}] \) is symmetric around 0, \( k_{\text{min}} \) is set to \(-k_{\text{max}}^{(0)}\) (and \( k_{\text{max}} = k_{\text{max}}^{(0)} \)). In the other case the module calculates

\[
k_0 = \text{round}(2^j(t_0 + t_{\text{end}})/2)
\]

and sets \( k_{\text{max}} = k_{\text{max}}^{(0)} + k_0 \) and \( k_{\text{min}} = -k_{\text{max}}^{(0)} + k_0 \), where \( k_{\text{max}}^{(0)} \) can be changed during the iteration (see algorithm at the following page).

\( k_{\text{max}}^{(0)} \) is set to be at least 10.

The step size is calculated as follows:

\[
h = \frac{t_{\text{end}} - t_0}{m} \quad \text{with} \quad m = r|k_{\text{max}}^{(0)}|.
\]

The initial \( r \) can be set in the module. It should be set to a value of at least 2 (for first order ODEs we have with \( r = 2 \) the classical collocation case), but the module accepts 1, too. \( k_{\text{max}}^{(0)}, j \) and \( r \) are positive integers.

Suitable positive real numbers \( \epsilon_1 \) and \( \epsilon_2 \) should be chosen (if it does not get specified, the module choose \( \epsilon_1 = 10^{-6} \) and \( \epsilon_2 = 10^{-2} \)). The default value of \( a \) is 2, of \( k_{\text{max}}^{(0)} \) is 15 and \( j \) is 1.

**The implemented algorithm:**

The algorithm calculates \( Q_{\text{min}} \) and \( Q_a \).

If \( Q_{\text{min}} \leq \epsilon_1 \), then it is checked whether \( Q_a \leq \epsilon_2 \) applies (with \( a > 1 \), for example \( a = 2 \)). If both conditions are met, then the iteration is finished.

Generally we suggested the following algorithm: If \( Q_{\text{min}} \leq \epsilon_1 \) is not met, \( j \) is incremented by 1 (if a sufficient number of basis functions \( \phi_{j,k} \) are chosen with respect to the approximation interval \( I \)). If \( Q_{\text{min}} < \epsilon_1 \) is met but \( Q_a < \epsilon_2 \) not, then \( m \) should be increased.

The module works with maximum 7 steps:

In the first step, if \( Q_{\text{min}} > \epsilon_1 \) then \( k_{\text{max}}^{(0)} \) is increased by 5 if \( k_{\text{max}}^{(0)} \leq 40 \) and \( j \) is increased by 1 if \( k_{\text{max}}^{(0)} > 40 \). If \( Q_a > \epsilon_2 \) and \( Q_{\text{min}} \leq \epsilon_1 \) then \( r \) is increased by 1.

After the first step, 5 steps follow.

If \( Q_{\text{min}} > \epsilon_1 \) then \( k_{\text{max}}^{(0)} \) is increased by 5 (if \( k_{\text{max}}^{(0)} < 45 \)), \( j \) by 1 and \( r \) by 1 (if \( r < 9 \)).

If \( Q_a > \epsilon_2 \) and \( Q_{\text{min}} \leq \epsilon_1 \) then \( r \) is increased by 1. When \( j \) is incremented by 1 we could double \( k_{\text{max}} \), but this can lead to very big minimization problems (because the length of \( c \) would double). Many simulations showed, that with \( k_{\text{max}} \) must not be so large as a \( L^2(\mathbb{R}) \) approximation needs (see example iv).

If after all these steps \( Q_{\text{min}} > \epsilon_1 \) or \( Q_a > \epsilon_2 \), then \( Q \) will be minimized for the last time.
The step counter in the mathematica module for the algorithm (\texttt{WCollocationS2}) is called \(z\) (\(z = 0\) to \(z = z_{\text{max}}\), \(z_{\text{max}} = 4\)) and the maximum number of \(k_{\text{max}}\) is set to 45. The maximum number of \(r\) is 9. In example iv we needed a bigger \(k_{\text{max}}\) and here was set the upper bound of \(k_{\text{max}}\) to 100.

The Module for the algorithm can be called with

\[
\texttt{WCollocationS2Alg[ODE, } t_0, y_0, t_0', t_{\text{end }}, k_{\text{max}}^{(0)}(15), j(1), r(2), a(2), e_1(10^{-6}), e_2(10^{-2})]\]

and (in brackets the default values) the collocation can be called with:

\[
\texttt{WCollocationS2[ODE, } t_0, y_0, t_0', t_{\text{end }}, k_{\text{max}}^{(0)}(15), j(1), r(2), a(2)\]

The first argument is the left side of the ODE in the form \(F(y', y, t) = 0\) or \(F(y'', y', y, t) = 0\) in Mathematica notation, that means \(y\) is \(y[t]\), \(y'\) is \(y'[t]\). \(t_0\) is the initial value or a List in the form \(\{t_0, t_{\text{end}}\}\) an analogous \(y(t_0)\) or \(\{y(t_0), y(t_{\text{end}})\}\). If we have a initial value problem for an second order ODE, then we can specify \(t_0\) and \(\{y(t_0), y'(t_0)\}\). The approximation interval boundaries are here called \(t_0\) and \(t_{\text{end}}\) (must not be identically with the initial or boundary \(t_0\)). The next parameters are optional: \(j, r, a\) (for \(Q_a\)). For the algorithm module we have additional \(e_1\) and \(e_2\). Because of \(Q_{\text{min}} \leq Q_a\) (for \(a = 2, 3, \ldots\)) or in most practical cases \(Q_{\text{min}} < Q_a\) the module gives a warning if \(e_1 > e_2\) and sets \(e_2 = 10e_1\). If you need very good approximations you may choose a smaller \(e_1\) and \(e_2\), but \(e_2\) should not be too small for problems with large slopes or large curvatures (for example for stiff problems). The modules can be downloaded from http://algorithm.jatam.de/.

For the minimization the Mathematica function \texttt{FindMinimum} will be used. Here we can get the following comment from Mathematica:

\texttt{FindMinimum::szer0: The step size in the search has become less than the tolerance prescribed by the PrecisionGoal option, but the gradient is larger than the tolerance specified by the AccuracyGoal option. There is a possibility that the method has stalled at a point that is not a local minimum.}

We got this comment in almost all iterations during the first iteration steps, but without any bad consequences for the approximation.

The algorithm can be also applied on higher order ODEs or (according to a generalization) on PDEs.

The second order ODEs have been found on the website of Jeff Cash (Imperial College, London). In the tests we used Mathematica 9.0 and 10.3.

The modules can be downloaded over the address http://jatam.de/algorithm/Wavelet-Approximation-Algorithm.nb.
Example I

We apply the algorithm on a second order ODE with boundary conditions:

\[ y'' - \frac{y'}{\mu} = 0 \] with \( \mu = 1/100 \) and with \( y(0) = 1, y(1) = 0 \), the approximation interval is \( I = [0, 1] \).

We called the module for the algorithm with:

\[ \text{WCollocationS2Alg}[y'[t] - 100 \ y'[t],\{0,1\},\{1,0\},0..1..15,1,2,2] \]

Mathematica NDsolve has problems:

\[ \text{NDSolve::bvlu}: \text{The equations derived from the boundary conditions are numerically ill-conditioned. The boundary conditions may not be sufficient to uniquely define a solution. The computed solution may match the boundary conditions poorly.} \]

Here is the graph of \( h - y \) (\( y \) is the NDsolve solution), were we can see, that the numerical solution of NDsolve has big deviations:

Now we see the iteration-protocol:
For critical examples we could start with a higher $k_{max}$, $j$ and $r$.

Here are the graphs of $y_5$ and $y$ (we see no differences graphically):

Here is the graph of $y_5 - y$: 
Example II

We apply the algorithm on a second order ODE with boundary conditions:

\[ y'' = (-y' + (1 + \mu) y)/\mu \] with \( \mu = 1/50 \) and with \( y(-1) = 1 + e^2, \ y(1) = 1 + e^{2((1+\mu)/\mu)} \), the approximation interval is \( I = [-1, 1] \).

We called the module for the algorithm with:

\[
\text{WCollocationS2Alg}[-51y[t]+50y'[t]+y''[t],\{-1,1\},\{1+e^{-2}, 1+e^{-10}\},-1.,1.,30,3,2,2]
\]

Here is the graph of \( y - y(\eta) \) (\( \eta \) is the NDSolve solution), were we can see, that the numerical solution of NDSolve has no big deviations:

Now we see the iteration-protocol:
<table>
<thead>
<tr>
<th>$k_{\max}^{(0)}$</th>
<th>$j$</th>
<th>$r$</th>
<th>$Q_{\min}$</th>
<th>$Q_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>3</td>
<td>2</td>
<td>$6.93893 \times 10^{-9}$</td>
<td>18.6674</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>3</td>
<td>$8.46827 \times 10^{-6}$</td>
<td>1.18102</td>
</tr>
<tr>
<td>35</td>
<td>4</td>
<td>4</td>
<td>$3.44103 \times 10^{-10}$</td>
<td>0.000228873</td>
</tr>
</tbody>
</table>

For critical examples we could start with a higher $k_{\max}, j$ and $r$. The solution of the first step (with $j = 3$ and $r = 2$) is not bad, the big $Q_a$ of 18.6674 results form a big residual value at one point ($t = -59/60$).

Here are the graphs of $y_4$ and $y$, where we see a good approximation, too.

Here is the graph of $y_4 - y$:
Example III

We apply the algorithm on a second order ODE with boundary conditions:

\[ y'' = \left(-t y' - \mu \pi^2 \cos(\pi t) - \pi t \sin(\pi t)\right) / \mu \]

with \( \mu = 1/10 \) and with \( y(-1) = -2, y(1) = 0 \), the approximation interval is \( I = [-1, 1] \).

We called the module for the algorithm with:

\[
\text{WCollocationS2Alg}\left[ \pi^2 \cos(\pi t) + 10 \pi t \sin(\pi t) + 10 t \ y'[t] + y''[t], \{-1, 1\}, \{-2, 0\}, -1., 1., 15, 1, 2, 2 \right]
\]

Here is the graph of \( \eta - y \) (\( \eta \) is the NDSolve solution):

![Graph of \( \eta - y \)]

Here is the graph of \( \eta \):

![Graph of \( \eta \)]

Now we see the iteration-protocol:

<table>
<thead>
<tr>
<th>( k_{\text{max}}^{(0)} )</th>
<th>( j )</th>
<th>( r )</th>
<th>( Q_{\text{min}} )</th>
<th>( Q_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>2</td>
<td>5.8001 \times 10^{-13}</td>
<td>1.85559 \times 10^{-7}</td>
</tr>
</tbody>
</table>

For critical examples we could start with a higher \( k_{\text{max}}, j \) and \( r \).
Here are the graphs of $y_1$ and $y$ (we see no differences):

![Graph of $y_1$ and $y$]

Here is the graph of $y_1 - y$:

![Graph of $y_1 - y$]

At last we see graphically the relation between $a$ and $Q_\alpha/a$ in this example:

![Graph of $\ln(Q_\alpha/a)$]

When we set $\mu = 1/100$, then NDSolve get problems:

`NDSolve::bvluc`: The equations derived from the boundary conditions are numerically ill-conditioned. The boundary conditions may not be sufficient to uniquely define a solution. The computed solution may match the boundary conditions poorly.
NDSolve::berr: There are significant errors \(\{0., -454021.\}\) in the boundary value residuals. Returning the best solution found. \(\Rightarrow\)

Here we get a very bad approximation: 

\(\gamma - y\) (\(\gamma\) is the NDSolve solution):

The algorithm has no problems:

\[ WCollocationS2Alg[\pi^2 \cos(\pi t) + 100 \pi t \sin(\pi t) + 100 t \ y'[t] + y''[t], \{-1, 1\}, \{-2, 0\}, -1., 1., 15, 1, 2, 2] \]

The graphs of \(y_4\) and \(y\):

The graphs of \(y_4 - y\):
At last the iteration protocol:

<table>
<thead>
<tr>
<th>$k_{\text{max}}^{(0)}$</th>
<th>$j$</th>
<th>$r$</th>
<th>$Q_{\text{min}}$</th>
<th>$Q_{a}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>2</td>
<td>1.44579</td>
<td>319690.</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>2</td>
<td>1.97176</td>
<td>258.584</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>3</td>
<td>1.66245</td>
<td>195.992</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>4</td>
<td>0.00284779</td>
<td>5.50948</td>
</tr>
<tr>
<td>35</td>
<td>4</td>
<td>5</td>
<td>$4.75163 \times 10^{-14}$</td>
<td>$1.06944 \times 10^{-10}$</td>
</tr>
</tbody>
</table>
Example IV

In this example we will see, that the maximum number of $k_{\text{max}}$ in the module (the maximum value of $k_{\text{max}}$ was set to 45) should be larger in problems, which needs a large $j$. We apply the algorithm on a second order ODE with boundary conditions:

$$y'' = (-4t \cdot y' - 2y)/(\mu + t^2)$$

with $\mu = 1/50$ and with $y(-1) = 1/(1 + \mu)$, $y(1) = 1/(1 + \mu)$, the approximation interval is $I = [-1, 1]$.

We called the module for the algorithm with:

$$\text{WCollocationS2Alg}[-(-2y[t] - 4t \cdot y'[t])/(1/50+t^2)+y''[t],\{-1,1\},\{10/11,10/11\},-1.,1.,30,5,3,2]$$

Mathematica NDSolve has no problems:

Here is the graph of $\eta - y$ ($\eta$ is the NDSolve solution):

Here is the graph of $\eta$:

With the used starting values the algorithm stops after the maximum number of steps has been made and the warning came that the solution does not satisfy the convergence criteria $Q_{\min} \leq \epsilon_1$ and $Q_a \leq \epsilon_2$. So we know that the solution is not usable.

Now we see the iteration-protocol:
The module prints: Warning: $Q_{\text{min}}$ or $Q_a$ is bigger than the tolerance!

We see in the iteration protocol, that the module has a maximum number of $k_{\text{max}}$. For critical problems, where we need a bigger $j$, the maximum number of $k_{\text{max}}$ should be set to a higher value than 45 in $W_{\text{CollocationS2Alg}}$. With $k_{\text{max}}^{(0)}$ less than $2^j$ the method cannot get a solution (with a small $Q_{\text{min}}$, because $Q_{\text{min}}$ is in that case $y(0)^2 + y(1)^2$) with the Shannon $\phi$ with that boundary conditions, because at $t_{\text{end}} = 1$ we get the boundary condition

$$y_j(1) := \sum_{k=k_{\text{max}}}^{k_{\text{max}}} c_k \cdot \phi_{j,k}(1) = \sum_{k=k_{\text{max}}}^{k_{\text{max}}} c_k \cdot 2^{j/2} \phi(2^j \cdot 1 - k) = y(1)$$

and $\phi(m) = 0$ for integer $m \neq 0$ and $\phi(1) = 1$. So if $k_{\text{max}}$ is less than $2^j$ the boundary condition cannot be fulfilled if $y(1) \neq 0$. Because if $y(-1) \neq 0$ we get the same for $k_{\text{min}}$. So with that boundary conditions we get $k_{\text{min}} \leq -2^j$ and $k_{\text{max}} \geq 2^j$. Otherwise $y_j(\pm 1) = 0$. With integer values of the boundaries $t_0$ and $t_{\text{end}}$ general $k_{\text{max}}$ should be greater or equal $2^j t_{\text{end}}$. Because of $k_{\text{min}}$ should be less or equal $2^j t_0$, in the module $k_{\text{max}}^{(0)}$ should be greater or equal (only if the expression is integer) $(2^j t_{\text{end}} - 2^j t_0)/2$, because in the module $k_{\text{max}}^{(0)}$ is positive (the module shifts automatically the summation area, $k_{\text{max}} = k_{\text{max}}^{(0)} + k_0$ and $k_{\text{min}} = -k_{\text{max}}^{(0)} + k_0$).

When set $k_{\text{max}}$ = 100 an apply the method with

$W_{\text{CollocationS2Alg}}[-(-2y[t] - 4t*y'[t])/(1/50+t^2)+y''[t],\{-1,1\},\{10/11,10/11\},\{-1,1\},\{80,6,8,2\}]$

then the algorithm stops directly with $k_{\text{max}} = 80$, $j = 6$, $r = 8$, $Q_{\text{min}} = 1.88584 \times 10^{-12}$ and $Q_a = 9.84522 \times 10^{-10}$. Here are the graphs of $y_6 - y$: 

```

15
```
A direct approximation makes no problems, too. For example, with

\[
\text{WCollocationS2Alg}[y[t] - \text{fe}[t], -1, 50/51, -1., 1., 25, 2, 4, 2]
\]
\[
\text{fe}[t_] := \frac{50}{1 + 50t^2}
\]

the algorithm stops with \( k_{\text{max}} = 45, j = 5, r = 7, Q_{\text{min}} = 5.83423 \times 10^{-10} \) and \( Q_a = 4.52406 \times 10^{-9} \).

Here are the graphs of \( y_5 - y \):

![Graph of y5 - y](image)

The difference between the second derivations is relative large (graph of \( y_5'' - y'' \)):

![Graph of y5'' - y''](image)

Here we get:

\[
\sum_{i=0}^{m} (y_j(t_i) - y(t_i))^2 = 5.83423 \times 10^{-10}
\]

with \( h = \frac{\text{t}_{\text{end}} - \text{t}_0}{m} \) with \( m = r \cdot |k_{\text{max}}^{(0)}| = 45 \cdot 7, \text{t}_0 = -1 \) and \( \text{t}_{\text{end}} = 1 \).

The derivation of the second order derivatives is much larger:

\[
\sum_{i=1}^{m} (y_j''(t_i) - y''(t_i))^2 = 8308.59
\]

The biggest difference we get at the beginning and at the end of the approximation interval with 3970.8 and 4307.34.
Comparing the $L^2$ approximation with the direct approximation on the interval $[-1,1]$:

We cannot apply the information about the $L^2(R)$ approximation $\tilde{y}_j$ from $y$ on

$$\text{span} \{ \phi_{j,k} \}_{k=k_{\text{min}}, \ldots, k_{\text{max}}+1} \subset V_j$$

to get the right $k_{\text{min}}$ and $k_{\text{max}}$ for the algorithm, because the $L^2(R)$ approximation may need a lot bigger $k_{\text{max}}$ than the direct approximation through (4) on the interval $[-1, 1]$ (to get nearly the same quality of approximation), like in our example, where the decay of the coefficients $\tilde{c}_k$ is very poor.

For the $L^2(R)$ approximation the coefficients will be calculated as usual with orthogonal bases (we assume for easier notation, that the scaling function and $y$ is real valued):

$$\tilde{c}_k = \langle y, \phi_{j,k} \rangle_{L^2(R)} = \int_R y(t) \cdot \phi_{j,k}(t) dt$$

And so we get the orthogonal projection from $y$ on $S_j$:

$$\tilde{y}_j(t) := \sum_{k=k_{\text{min}}}^{k_{\text{max}}} \tilde{c}_k \cdot \phi_{j,k}(t)$$

Generally $\| \tilde{y}_j - y \|_{L^2(I)} \geq \| \tilde{y}_j - y \|_{L^2(R)}$ with $I \subset R$. Here $\tilde{y}_j$ is the best approximation on $R$, calculated through

$$\min_{y_j} \| y_j - y \|_{L^2(R)} = \| \tilde{y}_j - y \|_{L^2(R)}$$

and $\hat{y}_j$ is the best approximation on $I$, calculated through

$$\min_{y_j} \| y_j - y \|_{L^2(I)} = \| \hat{y}_j - y \|_{L^2(I)}.$$

The reason for that is because $\tilde{y}_j$ is the best approximation according to the $L^2(R)$ norm on $R$ but $\hat{y}_j$ is the best approximation from $y$ only on the interval $I$ as a part of $R$. The direct approximation is the numerical solution of the minimum problem (3) above, so $y_j$ is the numerical approximation of $\hat{y}_j$ or the solution of (4). Here - for easier notation - we named the solution of the minimum problems the same as the unknown functions.

Theoretically we would get the solution of the continuous minimum problem (3) through the following considerations:
We calculate instead of (3) the orthogonal projection of a function $\tilde{y}$ on $S_j$. The function $\tilde{y}$ is on $I$ identical to $y$ and on $R \setminus I$ identically to our function $y_j$ of $S_j$ as a part of $V_j$. So $\tilde{y} = y_I + y_{R \setminus I}$ (where $y_I$ vanishes on $R \setminus I$ and $y_{R \setminus I}$ vanishes on $I$):

$$\tilde{y}(t) = 1_I(t) \cdot y(t) + 1_{R \setminus I}(t) \cdot \sum_{k=k_{\min}}^{k_{\max}} c_k \cdot \phi_{j,k}(t)$$ with indicator function 1.

So we approximate $y$ only on $I$. Outside $I$ the approximation function has no restricts. An other and a worse approximation we would get through the orthogonal projection from $1_I \cdot y$ on $V_j$. The reason is that we would cut the function $y$ and this would lead generally to a bad decay behavior in the Fourier space, see [19].

Here we get the coefficients $c_k$ through:

$$c_k = \langle \tilde{y}, \phi_{j,k} \rangle_{L^2(R)} = \langle y, \phi_{j,k} \rangle_{L^2(I)} + \langle y_j, \phi_{j,k} \rangle_{L^2(R \setminus I)} = \langle y, \phi_{j,k} \rangle_{L^2(I)} + \sum_{k=k_{\max}}^{k_{\min}} c_l \cdot \langle \phi_{l,k}, \phi_{j,k} \rangle_{L^2(R \setminus I)}$$

For $I = R$ we would get the best approximation on $R$ through $\langle y, \phi_{j,k} \rangle_{L^2(R)} \cdot \langle \phi_{j,k} \rangle_k$ is general no orthogonal system on $L^2(R \setminus I)$ (only if the support of $\phi_{j,k}$ is in $R \setminus I$). So we get:

$$c_k - \sum_{k=k_{\max}}^{k_{\min}} c_l \cdot \langle \phi_{l,k}, \phi_{j,k} \rangle_{L^2(R \setminus I)} = \langle y, \phi_{j,k} \rangle_{L^2(I)}$$

$$\sum_{k=k_{\max}}^{k_{\min}} c_l \cdot \delta_{l,k} - c_l \cdot \langle \phi_{l,k}, \phi_{j,k} \rangle_{L^2(R \setminus I)} = \langle y, \phi_{j,k} \rangle_{L^2(I)}$$

$$\sum_{k=k_{\max}}^{k_{\min}} c_l \cdot \langle \phi_{l,k}, \phi_{j,k} \rangle_{L^2(R \setminus I)} = \langle y, \phi_{j,k} \rangle_{L^2(I)}$$

$$\sum_{k=k_{\max}}^{k_{\min}} c_l \cdot \langle \phi_{l,k}, \phi_{j,k} \rangle_{L^2(R \setminus I)} = \langle y, \phi_{j,k} \rangle_{L^2(I)}$$

$$\sum_{k=k_{\max}}^{k_{\min}} c_l \cdot \delta_{l,k} = \langle y, \phi_{j,k} \rangle_{L^2(I)}$$

That is the normal equation for the vector $c$: $Ac = u$ and if we calculate $c$ through this equation we get the approximation error of (3):

$$\min \| y_j - y \|_{L^2(I)} = \| \tilde{y}_j - y \|_{L^2(I)} = \sqrt{\| y \|_{L^2(I)}^2 - c^T Ac}$$

The approximation error of the global $\tilde{y}_j$ approximation on $I$ is:

$$\| \tilde{y}_j - y \|_{L^2(I)} = \sqrt{\| y \|_{L^2(I)}^2 + \tilde{c}^T A\tilde{c} - 2\tilde{c}^T u}$$ with $\tilde{c}_k = \langle y, \phi_{j,k} \rangle_{L^2(R)}$.
(6) we get through (which is general for every vector $c$ right):

$$\left\| y_j - y \right\|^2_{L^2(\Omega)} = \left\| y \right\|^2_{L^2(\Omega)} - 2 \left( y, y_j \right)_{L^2(\Omega)} + \left\| y_j \right\|^2_{L^2(\Omega)}$$

with $\left( y, y_j \right)_{L^2(\Omega)} = \left( y, \sum_{l=k_{\text{min}}}^{k_{\text{max}}} c_l \cdot \phi_{j,l} \right)_{L^2(\Omega)} = \sum_{l=k_{\text{max}}}^{k_{\text{max}}} c_l \cdot \left( y, \phi_{j,l} \right)_{L^2(\Omega)} = c^T u$ and

$$\left\| y_j \right\|^2_{L^2(\Omega)} = \sum_{l=k_{\text{min}}}^{k_{\text{max}}} \sum_{k=k_{\text{min}}}^{k_{\text{max}}} c_l \cdot c_k \cdot \left( \phi_{j,l}, \phi_{j,k} \right)_{L^2(\Omega)} = c^T A c$$

So:

$$\left\| y_j - y \right\|^2_{L^2(\Omega)} = \left\| y \right\|^2_{L^2(\Omega)} - 2 c^T u + c^T A c$$

For the approximation $\hat{y}_j$ (where we get $c$ through $A c = u$) we get equation (6).

With a special scalar product $\langle x, y \rangle_A := x^T A x$ and the induced norm (with positive definite $A$) $\| x \|_A = \sqrt{\langle x, x \rangle_A}$ we get:

$$\left\| \hat{y}_j - y \right\|^2_{L^2(\Omega)} - \left\| y_j - y \right\|^2_{L^2(\Omega)} = c^T A c + \tilde{c}^T A \tilde{c} - 2 \tilde{c}^T u = c^T A c + \tilde{c}^T A \tilde{c} - 2 \tilde{c}^T A c$$

$$= \langle c, c \rangle_A + \langle \tilde{c}, \tilde{c} \rangle_A - 2 \langle c, \tilde{c} \rangle_A = \| \tilde{c} - c \|_A^2$$

For example:
The orthogonal projection of $y$ on $S_5$ or $V_5$ is very near to $y$, because the differences of the $L^2(\mathbb{R})$ norm the $||Y|| - ||Y_j||$ is very small.

With the direct approximation through

$$\min \ Q(c) = \sum_{i=0}^{m} (y_j'(t_i) - y(t_i))^2$$

We set $k_{\text{max}} = 80$ and $j = 6$ ($r = 8$) and start the minimization:

$$fe[t_] := 50/(1+50t^2)$$

```
WCollocationS2Alg[y[t] - fe[t], -1, fe[-1] // N, -1., 1., 80, 6, 8, 2]
```

Here the algorithm stops directly with $k_{\text{max}} = 80$, $j = 6$, $r = 8$ and a very small $Q_{\text{min}} = 6.98752 \times 10^{-22}$ and $Q_2 = 4.99871 \times 10^{-21}$. We started with bigger parameters to get less steps. The difference of the orthogonal projection $\hat{y}_j$ from $y$ on $V_j$ and $y$ (with $k_{\text{max}} = 80$) has the following graph:
Here we get:
\[
\sum_{i=0}^{m} (\tilde{y}_6(t_i) - y(t_i))^2 = 0.00131592 \quad \text{with} \quad h = \frac{t_{\text{end}} - t_0}{m} \quad \text{with} \quad m = r |k_{\text{max}}^{(0)}| = 80.8.
\]

If we set \( k_{\text{max}} = 1000 \) we get with the same \( t_i \) the following sum of squares
\[
\sum_{i=0}^{m} (\tilde{y}_6(t_i) - y(t_i))^2 = 7.42953 \times 10^{-13}
\]
which is small but even much bigger than the sum of squares with \( k_{\text{max}} = 80 \) and the direct approximation \( y_6 \):
\[
\sum_{i=0}^{m} (y_6(t_i) - y(t_i))^2 = 6.98752 \times 10^{-22}
\]

Here are the graphs of \( y_6 - y \) (\( k_{\text{max}} = 80 \)):

Finally here are some graphs of \( y_j - y \) for selected combinations of \( j, k_{\text{max}} \) and \( r \). Here we can see, that not only \( Q_{\text{min}} \) but \( Q_2 \) must be small, too. That is what the algorithm does. With a too small \( r \) we get a big \( Q_2 \). In some cases \( Q_2 \) can be large because of big deviations at the edge of the approximation interval. \( Q_2 \) was theoretically studied in [16]. In that example we got in many simulations the first good approximations for a minimal value for \( j \) of 6 and for \( k_{\text{max}} \) the minimal value have been 70.

For \( j = 6, k_{\text{max}} = 72, \) and \( r = 2 \) we got a \( Q_{\text{min}} = 1.82438 \times 10^{-13} \) and \( Q_2 = 9.04114 \times 10^7 \):
Here \( Q_2 \) was too big and so we got a bad approximation. The following examples have decreasing values of \( Q_2 \) and the approximation will get successively better.

For \( j = 6, k_{\text{max}} = 72, \) and \( r = 3 \) we got a \( Q_{\text{min}} = 3.31586 \times 10^{-12} \) and \( Q_2 = 5.98751 \times 10^{-6} \):

For \( j = 6, k_{\text{max}} = 74, \) and \( r = 3 \) we got a \( Q_{\text{min}} = 3.35466 \times 10^{-12} \) and \( Q_2 = 3.62539 \times 10^{-7} \):

For \( j = 6, k_{\text{max}} = 74, \) and \( r = 5 \) we got a \( Q_{\text{min}} = 4.74696 \times 10^{-12} \) and \( Q_2 = 7.57142 \times 10^{-9} \):
For $j = 6$, $k_{\text{max}} = 74$, and $r = 8$ we got a $Q_{\text{min}} = 7.71771 \times 10^{-12}$ and $Q_2 = 5.44366 \times 10^{-11}$.

Here we can see how with decreasing values of $Q_2$ and with $Q_{\text{min}}$ in the same magnitude the approximation error decreases.
Example V

We apply the algorithm on a second order ODE with boundary conditions:

\[ y'' = \left( y - \left( \mu \pi^2 + 1 \right) \cos(\pi t) \right) / \mu \]

with \( \mu = 1/100 \) and with \( y(-1) = -1, \ y(1) = -1 \), the approximation interval is \( I = [-1, 1] \).

We called the module for the algorithm with:

\[
\text{WCollocationS2Alg[\{-100\{-1+\pi^2/100\} \cos[\pi t]+y[t]\}+y''[t],\{-1,1\},\{-1,-1\},
{-1.1.1.15,1,2,2\}]
\]

Mathematica NDSolve has problems:

NDSolve::bvluc : The equations derived from the boundary conditions are numerically ill-conditioned. The boundary conditions may not be sufficient to uniquely define a solution. The computed solution may match the boundary conditions poorly.

NDSolve::berr: There are significant errors \( \{0.,-1.76873\times10^{-7}\} \) in the boundary value residuals. Returning the best solution found. 

Here is the graph of \( \eta - y \) (\( \eta \) is the NDSolve solution), were we can see, that the numerical solution of NDSolve has big deviations:

Here is the graph of \( \eta \):
Now we see the iteration-protocol:

<table>
<thead>
<tr>
<th>$k_{max}^{(0)}$</th>
<th>$j$</th>
<th>$r$</th>
<th>$Q_{min}$</th>
<th>$Q_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>2</td>
<td>$6.092 \times 10^{-27}$</td>
<td>$1.08892 \times 10^{-22}$</td>
</tr>
</tbody>
</table>

Here are the graphs of $y_1$ and $y$ (we see no differences):

Now we see graphically the relation between $a$ und $Q_a/a$ in this example:
Even with $\mu = 1/1000$ we get after one step a very good approximation:

The algorithm stops directly with a small $k_{\text{max}} = 15$, $j = 1$, $r = 2$, $Q_{\text{min}} = 1.0731 \times 10^{-24}$ and $Q_a = 1.5481 \times 10^{-21}$.

Here is the graph of $y_1 - y$:

Here is the graph of $\eta - y$ ($\eta$ is the NDSolve solution), were we can see, that the numerical solution of NDSolve has very big deviations:
Example VI

We apply the algorithm on a second order ODE with boundary conditions:

\[ y'' = (y + y^2 - e^{-2t / \sqrt{\mu}}) / \mu \]

with \( \mu = 1/100 \) and with \( y(0) = 1 \), \( y(1) = e^{-1/\sqrt{\mu}} \), the approximation interval is \( I = [0, 1] \).

We called the module for the algorithm with:

\[
\text{WCollocationS2Alg}[-100 (-e^{2t} + y[t] + y[t]^2) + y''[t], \{0,1\}, \{1,1/e^t\},0.,1., 15,1,2,2]
\]

Mathematica NDSolve has problems:

NDSolve::ndsz: At \( _\_t_\_ == 0.9415179282009^\_\_ \), step size is effectively zero; singularity or stiff system suspected.

General::stop: Further output of NDSolve::ndsz will be suppressed during this calculation. \( \Rightarrow \) Divide::infy: Infinite expression \(-\{2.07326*10^{-289}/0.\}_\_ \) encountered.

Mathematica automatically quits the kernel. The algorithm has no problems.

Now we see the iteration-protocol:

<table>
<thead>
<tr>
<th>( k_{max}^{(0)} )</th>
<th>( J )</th>
<th>( r )</th>
<th>( Q_{min} )</th>
<th>( Q_a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
<td>2</td>
<td>8.24021*10^{-14}</td>
<td>7.66528*10^{-11}</td>
</tr>
</tbody>
</table>

For critical examples we could start with a higher \( k_{max}, J \) and \( r \).

Here are the graphs of \( y_4 \) and \( y \) (we see no differences):

Here is the graph of \( y_4 - y \):
At last we see graphically the relation between $a$ and $Q_a/a$ in this example:
References


