Event location at integration of ODEs with jumping nonlinearity

Jirí Benedikt

Department of Mathematics
University of West Bohemia
30614 Plzeň
Czech Republic

benedikt@kma.zcu.cz

The author has been supported by the Grant 1N04078 and by the Research Plan MSM 4977751301 of the Ministry of Education, Youth and Sports of the Czech Republic.

Abstract

We are interested in the so-called Fučík spectrum of the fourth–order Dirichlet boundary value problem

\[ u^{(4)}(t) = \mu u^+(t) - \nu u^-(t), \quad t \in [0, 1], \]
\[ u(0) = u'(0) = u(1) = u'(1) = 0, \]

where \( u^+ = \max\{u, 0\} \) and \( u^- = \max\{-u, 0\} \), i.e. \( u = u^+ - u^- \). The Fučík spectrum is defined as the set of all couples \( (\mu, \nu) \in \mathbb{R}^2 \) such that the problem has a nontrivial solution, and it can be figured by Mathematica as the zero–contour of a function \( D = D(\mu, \nu) \). Existence of a stable solution of an initial–boundary value problem that describes the oscillation of a suspension bridge is related to the shape of the Fučík spectrum of our problem.

Introduction

Mathematical model of a suspension bridge

There are many mathematical models of suspension bridge, at different levels of simplification. One of the most simplified models is the fourth–order Dirichlet boundary value problem
\[ U^4(t) = f(u(t)), \quad t \in [0, 1], \]
\[ u(0) = u'(0) = u(1) = u'(1) = 0, \]

where \( u : [0, 1] \to \mathbb{R} \) (the solution) stands for the displacement of the bridge deck and \( f : \mathbb{R} \to \mathbb{R} \) describes the external force. The boundary conditions are called Dirichlet and mean that the ends of the bridge deck are clamped.

The external force \( f \) depends on the displacement \( u \), as it is typical for the suspension bridge – if the bridge deck moves downwards (\( u > 0 \) – we consider the downwards direction to be positive), the cable stays pull it back, if it moves upwards (\( u < 0 \)), the cable stays do nothing.

Let us assume that \( f \) depends linearly on \( u \), but with different constants for \( u \) positive and \( u \) negative, i.e. \( f = \mu u^+ - \nu u^- \), where \( u^+ = \max\{u, 0\} \) is the positive and \( u^- = \max\{-u, 0\} \) the negative part of \( u \) (hence, \( u = u^+ - u^- \)). This is satisfied, if the cable stays behave like a spring with the modulus \( \mu \) for \( u \) positive and the modulus \( \nu \) for \( u \) negative (\( \nu = 0 \) in our case). Note that such \( f \) is called a jumping nonlinearity.

```mathematica
In[1] := PosPart[a_] := (1/2) * (a + Abs[a]);
NegPart[a_] := (1/2) * (Abs[a] - a);
Show[GraphicsArray[
   {Plot[1[[1]] * PosPart[t] - 1[[2]] * NegPart[t], {t, -1, 1},
    AspectRatio -> Automatic, DisplayFunction -> Identity] & @
    {{[0.2, 0.7], {1, 0}, {-0.5, -0.5}}}
   ]];
```

\( f \) is called a jumping nonlinearity.

\[ \lim_{s \to +\infty} \frac{f(s)}{s} = \mu \quad \text{and} \quad \lim_{s \to -\infty} \frac{f(s)}{s} = \nu, \]

The Fučík (generalized) spectrum is defined as the set of all couples \((\mu, \nu) \in \mathbb{R}^2\) such that the problem has a nontrivial solution (eigenfunction).

If \( f \) is at least asymptotically equal to \( \mu u^+ - \nu u^- \), i.e.

\[ \lim_{s \to +\infty} \frac{f(s)}{s} = \mu \quad \text{and} \quad \lim_{s \to -\infty} \frac{f(s)}{s} = \nu, \]

and \((\mu, \nu)\) does not belong to the Fučík spectrum, then the topological degree theory can be used to prove that the solution is a priori bounded (otherwise there may exist an unbounded sequence of solutions). Hence it is important to know the Fučík spectrum well. Our aim is to figure the Fučík spectrum by Mathematica (we use the version 5.1.1.0).

\section*{Shooting method}

\section*{Initial value problem}

Let us consider the fourth–order initial value problem (IVP)
where $\mu, \gamma > 0$ and $\delta \in R$. The solution can be proved to be unique, and so we may denote by $V = V(\mu, \gamma, \delta)$ the value of the solution at $t = 1$. Obviously, $V$ is an increasing continuous function of $\delta$, and $\lim_{\delta \to \pm \infty} V(\mu, \gamma, \delta) = \pm \infty$. Hence there exists exactly one constant $\delta_0 = \delta_0(\mu, \gamma)$ such that $V(\mu, \gamma, \delta_0(\mu, \gamma)) = 0$. Since clearly $V(\mu, \gamma, \delta) > 0$ for any $\delta \geq 0$, it must be $\delta_0(\mu, \gamma) < 0$ for all $\mu, \gamma > 0$. Let $D = D(\mu, \gamma)$ denote the first derivative of the solution of the IVP with $\delta = \delta_0(\mu, \gamma)$. If $D(\mu, \gamma) = 0$, then $(\mu, \gamma)$ belongs to the Fučík spectrum. Moreover, $(\gamma, \mu)$ belongs to the Fučík spectrum, too, and the corresponding eigenfunction is $-u$.

On the other hand, let us assume that $(\mu, \gamma), \mu, \gamma > 0$, belongs to the Fučík spectrum and let $u$ be a corresponding eigenfunction. It can be proved that $u^{(\gamma)}(0) \neq 0$. Let us assume first $u^{(\gamma)}(0) > 0$. Since $au, a > 0$, is an eigenfunction, too, we may suppose that $u^{(\gamma)}(0) = 1$. Then, clearly, $u$ is the solution of the IVP with $\delta = \delta_0(\mu, \gamma)$, and so $D(\mu, \gamma) = 0$. If $u^{(\gamma)}(0) < 0$, then $-u$ is an eigenfunction, corresponding to $(\gamma, \mu)$ with $(-u)^{(\gamma)}(0) > 0$. Consequently, $D(\gamma, \mu) = 0$ in this case.

We showed that $(\mu, \gamma), \mu, \gamma > 0$, belongs to the Fučík spectrum, if and only if $D(\mu, \gamma) = 0$ or $D(\gamma, \mu) = 0$. Hence it may be figured as a union of the zero contours of $D(\mu, \gamma)$ and $D(\gamma, \mu)$. Our aim now reduces to implementation of $D$.

### Implementation

#### The first problem: precision

From now on we write $u^4$ and $\gamma^4$ instead of $u$ and $\gamma$ in the differential equation. First we implement the function $V$. The simplest one might be

```math
\[ V(\mu, \gamma, \delta) := u[1] / . \text{NDSolve}[[\text{Derivative}[4][u][t] := \mu^4 * \text{PosPart}[u[t]] - \gamma^4 * \text{NegPart}[u[t]], u[0] = 0, \\
\text{Derivative}[1][u][0] = 0, \text{Derivative}[2][u][0] = 1, \\
\text{Derivative}[3][u][0] = \delta}, u, \{t, 0, 1\}][[1]]; \]
```

Now we look for $\delta_0(\mu, \gamma)$ as a root of $V(\mu, \gamma, \delta)$.

```math
\[ V[40, 30, \text{NDSolve}[[\text{Derivative}[4][u][t] := \mu^4 * \text{PosPart}[u[t]] - \gamma^4 * \text{NegPart}[u[t]], u[0] = 0, \\
\text{Derivative}[1][u][0] = 0, \text{Derivative}[2][u][0] = 1, \\
\text{Derivative}[3][u][0] = \delta}, u, \{t, 0, 1\}][[1]]; \]
```

We see that if we want to guarantee, e.g., $|V(\mu, \gamma, \delta_0(\mu, \gamma))| < 10^{-6}$, then the "machine precision" is not enough. So let us improve the implementation of $V$. 

![Graph showing the solution](https://example.com/image.png)
In[8]:= ValueAtOne[\[mu]_, \[upsilon]_, \[delta]_, goal_, working_] := 
    u[1] /. NDSolve[{Derivative[4][u][t] = 
        SetPrecision[\[mu]^4, working] * PosPart[u[t]] - 
        SetPrecision[\[upsilon]^4, working] * NegPart[u[t]], u[0] = 0, 
        Derivative[1][u][0] = 0, Derivative[2][u][0] = 1, 
        Derivative[3][u][0] = SetPrecision[\[delta], working], 
        u, {t, 0, 1}, PrecisionGoal -> goal, 
        WorkingPrecision -> working}[[1]]);

In[9]:= p1 = Plot[ValueAtOne[40, 30, t, 20, 40], 
    {t, -45, -35}, DisplayFunction -> Identity];

p2 = Plot[ValueAtOne[40, 30, t, 20, 40], {t, -40.37441848174, 
    -40.37441848177}, DisplayFunction -> Identity];
Show[GraphicsArray[{p1, p2}]];

Increasing of the working precision caused a significant change of the obtained values.
To find an optimal precision we can compare the values for different precisions.

In[12]:= TableForm[Table[ValueAtOne[40, 30, -40.37441848177, n, 2*n], 
    {n, 15, 25}], TableHeadings -> 
    {Table[2*n, {n, 15, 25}], {"working", "result"}}]

<table>
<thead>
<tr>
<th>Working Precision</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>4147.844009112627879036194219</td>
</tr>
<tr>
<td>16</td>
<td>4147.85020676900216790826638637</td>
</tr>
<tr>
<td>17</td>
<td>4147.8507159791465763152859015341</td>
</tr>
<tr>
<td>18</td>
<td>4147.850720342217179176002726697527</td>
</tr>
<tr>
<td>19</td>
<td>4147.85072103731465664076884363472049</td>
</tr>
<tr>
<td>20</td>
<td>4147.8507211735240502352115365517091080</td>
</tr>
<tr>
<td>21</td>
<td>4147.850721189333104132293720867813757838</td>
</tr>
<tr>
<td>22</td>
<td>4147.85072119207624366125682054932031268129</td>
</tr>
<tr>
<td>23</td>
<td>4147.85072119263215932478742155881657174450701</td>
</tr>
<tr>
<td>24</td>
<td>2.01380531575\times 10^{-6}</td>
</tr>
<tr>
<td>25</td>
<td>4147.85072119265397022706925474453125951484798668</td>
</tr>
</tbody>
</table>

Obviously, PrecisionGoal 20 and WorkingPrecision 40 is enough if we want
the (absolute) precision of the result to be less than $10^{-6}$. But what happened when the
WorkingPrecision was 48!? Let us plot the solution for this case.

Out[12]//TableForm=
In[13]:= 

\[ \text{sol = u /. NDSolve}\{\text{Derivative}[4][u][t] = \text{SetPrecision}[40^4, 48] \ast \text{PosPart}[u[t]] - \text{SetPrecision}[30^4, 48] \ast \text{NegPart}[u[t]], u[0] == 0, \text{Derivative}[1][u][0] == 0, \text{Derivative}[2][u][0] == 1, \text{Derivative}[3][u][0] = \text{SetPrecision}[40.37441848177, 48], u, \{t, 0, 1\}, \text{PrecisionGoal} \rightarrow 24, \text{WorkingPrecision} \rightarrow 48\}\[[1]] \]

\[ \text{Plot}\{\text{sol}[t], \{t, 0, 1\}, \text{PlotRange} \rightarrow \{\{0, 1\}, \{-0.01, 0.01\}\}\}; \]

NDSolve::mxst : Maximum number of 10000 steps reached at the point \( t = 0.158915875742668323463501024550776480465923783645 \).

Out[13]= \[
\text{InterpolatingFunction}\[\{\{0, 0.158915875742668323463501024550776480465923783645\}\}, <\] \]

\[ \text{InterpolatingFunction}\[\text{:dmval} : \text{Input value} (0.166892) \text{lies outside the range of data in the interpolating function. Extrapolation will be used. More...} \]

\[ \text{InterpolatingFunction}\[\text{:dmval} : \text{Input value} (0.162122) \text{lies outside the range of data in the interpolating function. Extrapolation will be used. More...} \]

\[ \text{InterpolatingFunction}\[\text{:dmval} : \text{Input value} (0.159579) \text{lies outside the range of data in the interpolating function. Extrapolation will be used. More...} \]

General::stop : Further output of InterpolatingFunction::dmval will be suppressed during this calculation. More...

The problem is that the solution was not computed on the whole interval \([0, 1]\) and extrapolation was used. We can fix the problem using the StiffnessSwitching method.

In[15]:= 

\[ \text{ValueAtOne}[\mu_, \gamma_, \delta_, \text{goal}_-, \text{working}_-] :=\]

\[ u[1] /. \text{NDSolve}\{\text{Derivative}[4][u][t] = \text{SetPrecision}[\mu^4, \text{working}] \ast \text{PosPart}[u[t]] - \text{SetPrecision}[\gamma^4, \text{working}] \ast \text{NegPart}[u[t]], u[0] == 0, \text{Derivative}[1][u][0] == 0, \text{Derivative}[2][u][0] == 1, \text{Derivative}[3][u][0] = \text{SetPrecision}[\delta, \text{working}], u, \{t, 0, 1\}, \text{PrecisionGoal} \rightarrow \text{goal}, \text{WorkingPrecision} \rightarrow \text{working}, \text{Method} \rightarrow \{\text{StiffnessSwitching}\}\[[1]]; \]
In[16]:= TableForm[Table[ValueAtOne[40, 30, -40.37441848177, n, 2*n], {n, 15, 25}], TableHeadings -> {Table[2*n, {n, 15, 25}], {"working", "result"}}]

Out[16]//TableForm=

<table>
<thead>
<tr>
<th></th>
<th>Working</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>4147.850539168947270588</td>
<td>01973</td>
</tr>
<tr>
<td>32</td>
<td>2354.624380592540479186154070999</td>
<td>8</td>
</tr>
<tr>
<td>34</td>
<td>4073.3797137632379322010284233788</td>
<td>8</td>
</tr>
<tr>
<td>36</td>
<td>4147.850720494023771567460056185661</td>
<td>8</td>
</tr>
<tr>
<td>38</td>
<td>4147.8335663037588578460793401693914</td>
<td>8</td>
</tr>
<tr>
<td>40</td>
<td>4147.850702963894582979450700973703438</td>
<td>8</td>
</tr>
<tr>
<td>42</td>
<td>4144.69076993244960926007724027825195768</td>
<td>8</td>
</tr>
<tr>
<td>44</td>
<td>4146.6185937252694455615946699370694494536</td>
<td>8</td>
</tr>
<tr>
<td>46</td>
<td>4146.3270592512689254466365793133653519215778</td>
<td>8</td>
</tr>
<tr>
<td>48</td>
<td>4147.683873484625938621052746662330203035864920</td>
<td>8</td>
</tr>
<tr>
<td>50</td>
<td>4147.85072119265405482969749315775887465114041394</td>
<td>8</td>
</tr>
</tbody>
</table>

The equation was always solved on the whole \([0, 1]\). However, the results are obviously not consistent.

☐ The second problem: jumping nonlinearity

The reason why we got such nonconsistent results is that our \(f\) (the right-hand side) is the jumping nonlinearity which is not differentiable at \(0\). It brings unpredictable errors into the solution as it changes its sign. Consequently, we must stop the integration whenever the solution attains the zero value, and restart the integration. This is easily accomplished with the EventLocator method, new in Mathematica 5.1.

In[17]:= state = First[NDSolve'ProcessEquations[
   {Derivative[4][u][t] = SetPrecision[40^4, 130] * PosPart[u[t]] - SetPrecision[30^4, 130] * NegPart[u[t]],
   u[0] = 0, Derivative[1][u][0] = 0, Derivative[2][u][0] = 1,
   Derivative[3][u][0] = SetPrecision[-40.37441848177, 130]},
   u, t, Method -> {EventLocator, "Event" -> u[t],
   Method -> StiffnessSwitching},
   WorkingPrecision -> 130, AccuracyGoal -> 30,
   PrecisionGoal -> 30, MaxSteps -> Infinity]);
result = Reap[While[Sow[state["CurrentTime"["Forward"]]] < 1, state = NDSolve'ReinitializeVector[
   state, state["CurrentTime"["Forward"]],
   state["SolutionVector"["Forward"]];
   NDSolve'Ite rate[state, 1];],
   state["SolutionVector"["Forward"]];
   SetPrecision[result[[1]][[1]], 30]]

Out[19]= 4147.85072119265408024709603968

The used precision appears to be sufficient for any \(\mu, \nu \in [0, 50]\). Using the EventLocator is not appropriate when \(\mu = \nu\), and so we finally define
Having the function \( V \) implemented, we can easily compute \( \delta_0(\mu, \nu) \) and \( D(\mu, \nu) \).

Finally, the Fučík spectrum may be figured by the \texttt{ContourPlot} function as follows:
\[
\text{ContourPlot}[\text{DerZero}[\mu, \nu] \cdot \text{DerZero}[\mu, \nu], \\
\{\mu, 0, 50\}, \{\nu, 0, 50\}, \text{Contours} \to \{0\}, \text{PlotPoints} \to 200]
\]
But it would take months...

\section*{Optimization}

The \texttt{ContourPlot} method evaluates the function uniformly at all \( n \times n \) points. We develop an adaptive iterative algorithm for figuring the zero contour of a function. It evaluates the function more densely near already found lines to make them smoother, but still it searches for another components of the contour.
Let us consider the following test function.

\[
\begin{align*}
 f(x, y) &= \sin(x + \cos(x + y)) \cdot \sin(y + 0.2) + x / 25 - y / 8 - 0.08; \\
 p &= \text{ContourPlot}[f(x, y), (x, y) \in (0, 10), \\
 &\quad \text{PlotRange} \to (0, 10), \text{Contours} \to \{0, 1\}, \\
 &\quad \text{ContourStyle} \to \text{Hue}[0], \text{PlotPoints} \to 65];
\end{align*}
\]

To draw the picture, \texttt{ContourPlot} needed to evaluate the function at \(65 \times 65 = 4225\) points. Our algorithm follows.

\[
\begin{align*}
\text{size} &= 10; \\
\text{proporref} &= .9; \\
\text{InsertNew}[\text{main}_, \text{pos}_] &= \text{Module}[\{\text{am}, \text{new}, \text{numnew}\}, \text{am} = \text{main}; \\
\text{new} &= \text{Complement}[\text{pos}, \text{am}[2]] /. \{x_?\text{NumericQ}, y_?\text{_}, f_\} \to \{x, y\}; \\
\text{numnew} &= \text{Length}[\text{new}]; \\
\text{am} &= \{\text{am}[1], \text{Join}[\text{am}[2], \text{new}]\}; \\
\text{am} &= \text{am} /. \{x_?\text{NumericQ}, y_?\text{_}, f_\} \to \{x, y, f[x, y]\}; \\
\text{am} &= \{\text{am}[1], \text{Sort}[\text{am}[2]], \text{#1}[2] < \#2[[2]] || \\
&\quad \#1[[2]] := \#2[[2]] && \#1[[1]] < \#2[[1]] &&\}; \\
\{\text{numnew}, \text{am}\}] \\
\text{Uni}[\text{main}_] &= \text{Module}[\{\text{am}, \text{posnew}, \text{numnew}\}, \text{am} = \text{MapAt}[\text{#} + 1 &, \text{main}, \{1, 1\}] \\
\text{posnew} &= \text{Flatten}[\text{Table}[\{i \times \text{size}, i \times \text{size}\}, \{i, 0, 1, \text{2} - \text{am}[1][1][1]\}, \{j, 0, 1, \text{2} - \text{am}[1][1][1]\}], 1]; \\
\{\text{numnew}, \text{am}\} &= \text{InsertNew}[\text{am}, \text{posnew}] \\
\text{am} &= \text{MapAt}[\text{#} - (\text{2} \times \text{am}[1][1][1] + 1) \times \text{2} - \text{am}[1][1][1] + \text{numnew}] &, \\
&\quad \text{am}, \{1, 4\}]; \\
\text{ReplacePart}[\text{am}, (\text{2} \times \text{am}[1][1][1] + 1) \times \text{2}, \{1, 3\}] \\
\text{RefDiv}[\text{main}_, \text{lev}_] &= \text{Module}[\{\text{am}, \text{posnew}, \text{numnew}\}, \text{am} = \text{main}; \\
\text{amchoice} &= \text{Select}[\text{am}[2]], \\
\text{posnew} &= \text{Select}[\text{amchoice}, \text{Module}[\{\text{thenext}\}, \text{thenext} = \text{Select}[\text{amchoice}, \text{Function}[\text{a}, \\
\text{Take}[\text{a}, 2] + \{\text{size}, 0 \times \text{2} - (\text{lev} - 1) = \text{Take}[\text{a}, 2], 1\]; \\
\text{thenext} = \text{\&\& \#[[3]] \times \text{thenext}[1][3] \leq 0] \&/. \{x_?\text{NumberQ}, y_?\text{_}, f_\} \to \{x + \text{size} \times \text{2} - \text{lev}, y\}; \\
\text{posnew} &= \text{Join}[\text{posnew}, \text{Select}[\text{amchoice}, \text{Module}[\{\text{thenext}\}, \text{thenext} = \text{Select}[\text{amchoice}, \text{Function}[\text{a}, \\
\text{Take}[\text{a}, 2] + \{\text{size}, 0 \times \text{2} - (\text{lev} - 1) = \text{Take}[\text{a}, 2], 1\]; \\
\text{thenext} = \text{\&\& \#[[3]] \times \text{thenext}[1][3] \leq 0] \&/. \{x_?\text{NumberQ}, y_?\text{_}, f_\} \to \{x + \text{size} \times \text{2} - \text{lev}, y\}; \\
\{\text{numnew}, \text{am}\} &= \text{InsertNew}[\text{am}, \text{posnew}]; \\
\text{MapAt}[\text{#} + \text{numnew} &, \text{am}, \{1, 4\}]]; \\
\text{RefAdd}[\text{main}_, \text{lev}_] &= \text{Module}[\{\text{am}, \text{posnew}, \text{numnew}\}, \text{am} = \text{main}; \\
\text{amchoice} &= \text{Select}[\text{am}[2]],
\end{align*}
\]
Mod[Take[#, 2], size*2^lev] = {0, 0} &;
posnew = Flatten[Select[am[[2]],
  Module[{thenext}, thenext = Select[amchoice, Function[a,
  Take[#, 2] + {0, size}*2^lev = Take[a, 2], 1];
  thenext = {} && #[[3]] * thenext[[1]] [[3]] <= 0 &] /. 
  {x_?NumberQ, y_, f_} -> {{x, y - size*2^lev}, {x, 
  y + size*2^lev}, {x + size*2^lev, y - size*2^lev}, 
  {x + size*2^lev, y + size*2^lev}}], 1];
posnew = Join[posnew, Flatten[Select[am[[2]],
  Module[{thenext}, thenext = Select[amchoice, Function[a,
  Take[#, 2] + {0, size}*2^lev = Take[a, 2], 1];
  thenext = {} && #[[3]] * thenext[[1]] [[3]] <= 0 &] /. 
  {x_?NumberQ, y_, f_} -> {{x - size*2^lev, y}, 
  {x + size*2^lev, y}, 
  {x + size*2^lev, y + size*2^lev}, 
  {x + size*2^lev, y + size*2^lev}}], 1];
posnew = Union[Select[posnew, #[[1]] >= 0 && 
  #[[1]] <= size && #[[2]] >= 0 && #[[2]] <= size &];
numnew, am] = InsertNew[am, posnew];
MapAt[#+ numnew &, am, {1, 4}];
MyRefine[main_] :=
  Module[{}, am = main; For[i = 1, i <= am[[1]][[2]], 
  i++, am = RefDiv[am, i]; am = RefAdd[am, i]]; am];
Step[main_] := Module[{}, ifrefine, orig, am = main; ifrefine = 
  am[[1]][[4]]/Plus @@ am[[1]][[1, 4, 3, 4]] < prporref;
  If[ifrefine, am = MapAt[# + 1 &, am, {1, 2}], 
  orig = am[[1]][[4]]; am = FixedPoint[MyRefine[#, &], am];
  If[! ifrefine, orig = am[[1]][[4]]; am = FixedPoint[MyRefine[#, &], am];
  am = MapAt[Max[#, am[[1]][[1]]] &, am, {1, 2}]]; am];
interp[main_, x_, y_] :=
  main[2][[Position[#, Min[#]][[1, 1]] &
    Norm[{x, y} - Take[#, 2] & @ main[2]][[1]][[3]]]]
  depict[main_] := Module[{}, p1 = 
    ContourPlot[interp[main, x, y], {x, 0, size}, {y, 0, size}, 
    PlotPoints -> 2*main[[1]][[2]] + 1, Contours -> 0, 
    ContourShading -> None, DisplayFunction -> Identity];
  p2 = Graphics[{PointSize[.012], 
    main[[2]] /. {x_?NumberQ, y_, f_} -> Point[{x, y}]]; 
  Show[p, p1, p2, DisplayFunction -> Identity, 
    PlotRange -> {(-.1, size + .1), (-.1, size + .1)}];

(*41*)
Val = {{{0, 0, 4, 0}, 
  Flatten[Table[{j* size, i* size, f[j* size, i* size]},
  {i, 0, 1}, {j, 0, 1}]], 1]};
Block[{eval = Plus @@ Last[Val][[1]][[3, 4]], 
  allc = 2*Last[Val][[1]][[2]] + 1 "2", 
  ToString[Last[Val][[1]]] <> ", " <> ToString[eval] <> "/ " <> 
  ToString[allc] <> ", " <> ToString[eval/allc*100.] <> "/ " <> 
  ToString[allc*100.] <> ", " <> ToString[eval/allc*100.] <> "/ " <> 
  ToString[allc*100.]}]
  pic1 = depict[Last[Val]]; 
  Val = Append[Val, Step[Last[Val]]];
  Block[{eval = Plus @@ Last[Val][[1]][[3, 4]], 
    allc = 2*Last[Val][[1]][[2]] + 1 "2", 
    ToString[Last[Val][[1]]] <> ", " <> ToString[eval] <> "/ " <> 
    ToString[allc] <> ", " <> ToString[eval/allc*100.] <> "/ " <> 
    ToString[allc*100.]}]
  pic2 = depict[Last[Val]]; 
  Show[GraphicsArray[{pic1, pic2}];

Out[40]=
  {0, 0, 4, 0}, 4/4 (100.)

Out[45]=
  {0, 1, 4, 5}, 9/9 (100.)
In[48]:= Val = Append[Val, Step[Last[Val]]];
Block[{
    eval = Plus @@ Last[Val][[1]][[3, 4]],
    allc = (2^Last[Val][[1]][[2]] + 1)^2,
    ToString[Last[Val][[1]]] <> ", " <> ToString[eval] <> "/ " <>
    ToString[allc] <> ", " <> ToString[eval/100. <> "] <> ">

pic1 = depict[Last[Val]];
Val = Append[Val, Step[Last[Val]]];
Block[{
    eval = Plus @@ Last[Val][[1]][[3, 4]],
    allc = (2^Last[Val][[1]][[2]] + 1)^2,
    ToString[Last[Val][[1]]] <> ", " <> ToString[eval] <> "/ " <>
    ToString[allc] <> ", " <> ToString[eval/100. <> "] <> ">

pic2 = depict[Last[Val]];
Show[GraphicsArray[{
    pic1, pic2}]];
Event location at integration of ODEs with jumping nonlinearity

Avignon, June 2006
8th International Mathematica Symposium
Val = Append[Val, Step[Last[Val]]];
Block[
{eval = Plus @@ Last[Val][[1]][[3, 4]]],
allc = (2^Last[Val][[1]][[2]] + 1)^2,
ToString[Last[Val][[1]]] <> "", " <> ToString[eval] <> " / " <>
ToString[allc] <> " (" <> ToString[eval / allc*100.] <> "%)"
pic1 = depict[Last[Val]];
Val = Append[Val, Step[Last[Val]]];
Block[
{eval = Plus @@ Last[Val][[1]][[3, 4]]],
allc = (2^Last[Val][[1]][[2]] + 1)^2,
ToString[Last[Val][[1]]] <> "", " <> ToString[eval] <> " / " <>
ToString[allc] <> " (" <> ToString[eval / allc*100.] <> "%)"
pic2 = depict[Last[Val]];
Show[GraphicsArray[{pic1, pic2}]];
Out[63]= {2, 4, 25, 121), 146 / 289 (50.519%)
Out[66]= {2, 5, 25, 296), 321 / 1089 (29.4766%)
We get the same result, needing only 19% of the points, that standard \texttt{ContourPlot} needed. Of course, we can continue in the refinement.
The author is a mathematician working in the theory of existence, uniqueness and bifurcations of nontrivial solutions of higher-order nonlinear differential equations.