The mfpic4ode package*

Robert Marik
marik@mendelu.cz

April 15, 2009

1 Introduction

The package `mfpic4ode` is a set of macros for drawing phase portraits and integral curves of differential equations and autonomous systems using `mfpic` macros. These macros have been used by the author to prepare some pictures for classrooms and the results seem to be accptable for this purpouse, but always remember that due to the fixed points arithmetics in Metapost, the error in computations could be significant.

2 Usage

You can load the package in LaTeX using standard `\usepackage{mfpic4ode}` command, or you can use the macros in plainTeX and load by `\input mfpic4ode.tex` command.

2.1 First order differential equation

To draw phase portrait of first order ordinary differential equation

$$y' = f(x, y)$$

we define commands `\ODEarrow` for drawing element of direction field and `\trajectory`, `\trajectoryRK` and `\trajectoryRKF` for drawing integral curves using Euler, second order Runge-Kutta and fourth order Runge-Kutta methods, respectively. Some important parameters, such as the number of steps, the length of step or the function from the right-hand side of the equations are stored in MetaPost variables and to keep the package simple and short, these variables are accessible using `\mfsrc` command.

`\ifcolorODEarrow`
`\colorODEarrowtrue`
`\colorODEarrowfalse`

If the TeX boolean variable `\ifcolorODEarrow` is true, then the arrows from direction field are blue if the solution is increasing and red if decreasing. If

*This document corresponds to mfpic4ode 0.3, dated 2009/04/15.

`\ifcolorODEarrow` is false, the mfpic color from `\drawcolor` and `\headcolor` macros is used. More precisely,

- if `\ifcolorODEarrow` is true and $f(x_0, y_0) > 0$, then the arrow at the point (x_0, y_0) is blue
- if `\ifcolorODEarrow` is true and $f(x_0, y_0) \leq 0$, then the arrow at the point (x_0, y_0) is red
- if `\ifcolorODEarrow` is false, then color from `\drawcolor` is used to draw the body of an arrow and color `\headcolor` is used to draw the head.

Arrows are drawn using mfpic `\draw\arrow\lines{...}` command and hence the parameters for customizing shape and size of the head from mfpic are also available. The MetaPost variable `ODEarrowlength` is used to customize the length of each arrow. If the arrow is horizontal, then the length of the arrow in mfpic coordinates is equal to `ODEarrowlength/xscale`. (This fixes the case when different `xscale` and `yscale` are used. All arrows have the same length.) You can set this variable using `\mfsrc` command, you can write e.g.

```
\mfsrc{ODEarrowlength:=0.07;}
```

in your document.

To draw arrows in regular rectangular grid you should use the `\ODEarrow` macro in a double cycle such as

```
\mfsrc{for j=0 step 0.07 until 1.2: for i:=0 step 0.5 until 10:}
\ODEarrow{i}{j}
\mfsrc{endfor;endfor;}
```

or using the `multido` package

```
\multido{\r=0.0+0.1}{15}{\multido{\R=0.0+0.5}{19}{\ODEarrow{\R}{\r}}}
```

`\ODEdefineequationf(x,y)`

The macro `\ODEdefineequation` is used to save the right hand side of the ODE, i. e. the function $f(x, y)$. You should write the expression in the MetaPost format, the independent variable is supposed to be x, the dependent variable is y.

`\trajectory{x0}{y0}`
`\trajectoryRK{x0}{y0}`
`\trajectoryRKF{x0}{y0}`

The macros `\trajectory`, `\trajectoryRK` and `\trajectoryRKF` are used for drawing integral curves with initial condition $y(x_0) = y_0$ using Euler, second order Runge-Kutta and fourth order Runge-Kutta methods, respectivelly. The length of each step is stored in MetaPost variable `ODEstepcount`, the length of each step is in the MetaPost variable `ODEstep`. You can set these variables using `\mfsrc` macro as follows

```
\mfsrc{ODEstep:=0.02; ODEstepcount:=500;}
```

The integral curve is drawn from short linear parts using `\ODEline` command which expands to `\lines` command from `mfpic` package by default. These linear parts are connected in connect environment and this allows to use prefixes like `\dotted` or `\dashed` to the trajectories. A simple test is used to keep the arithmetics in reasonable bounds: if after the step the curve leaves the horizontal strip between `yneg` and `ypos` variables, then the evaluation is stopped (in fact, in this case we do not change the independent variable and we do the remaining steps with the same last point). Recall that `yneg` and `ypos` variables are set when you call `mfpicture` environment. If you call the environment as follows

```
\begin{mfpic}[5][3]{-0.1}{1.5}{-0.1}{0.5}
   ...........
\end{mfpic}
```

then no more than one short linear part of the integral curve is outside the horizontal strip between $y = -0.1$ and $y = 0.5$.

`\trajectories` `\ODEarrows`

To draw more trajectories you can use `\trajectories` command. The command `\trajectories{x1,y1;x2,y2;x3,y3;....;xn,yn}` expands to n `\trajectoryRKF` commands with initial conditions $y(x_i) = y_i$ for $i = 1..n$. In a similar way, `\ODEarrows{x1,y1;x2,y2;x3,y3;....;xn,yn}` expands into n `\ODEarrow` commands.

2.2 Two-dimensional autonomous systems

Trajectories for two-dimensional autonomous system

$$\begin{aligned} x' &= f(x,y) \\ y' &= g(x,y) \end{aligned}$$

are drawn using a very simple method based on the direction field. This could be improved in the next release of the package, but till now the results obtained in this way are qualitatively correct and sufficiently accurate (remember that you cannot expect accurate approximation due to the limitation of arithmetics in MetaPost). Anyway, some users may prefer the fourth order Runge–Kutta method.

The macros `\ASdefineequation` `\ASarrow`, `\AStrajectory`, `\AStrajectoryRKF`, `\ASarrows` and `\AStrajectories` are counterparts to their `\ODE....` versions. The last point of each trajectory is stored in the `x1` and `x2` MetaPost variables. Hence, you can say `\AStrajectory{2}{2}` to draw trajectory with initial conditions $x(0) = 2$, $y(0) = 2$ and then you can continue this trajectory using `\AStrajectory{x1}{y1}` command. The macro `\AStrajectory` uses `ODEstep` and `ODEstepcount` variables, the macro `\AStrajectoryRKF` uses `TIMEstep` and `TIMEend` variables do perform the steps in the numerical solution. The number of steps is in the latter case evaluated as absolute value of the quotient `TIMEend/TIMEstep`. You can use negative value for `TIMEstep` to continue the trajectory backwards.

3 Troubleshooting

3.1 The catcode of @ is messed

We set the category of @ to 11 (letter) when we load the package and at the end of definitions for mfpic4ode we set the category to 12. This could be a source of rare problems, if you use different value in your document.

3.2 Metapost: Not implemented: (unknown numeric) ...

You have to set `ODEstep`, `ODEstepcount`, `TIMEstep` and `TIMEend` other variables using `\mfsrc` command (depending on the type of the problem).

4 Implementation

```
⟨*tex⟩
\catcode`\@=11

\newif\ifcolorODEarrow
%%%\colorODEarrowfalse
\colorODEarrowtrue

%%% The line from one point to another
\def\ODEline#1#2{\lines{#1,#2}}

%%% The variable ODErhs is used to store the function from the right
%%% hand side of ODE in the form y'=f(x,y). We use command
%%% ODEdefineequation to set up this variable.
\def\ODEdefineequation#1{\fdef{ODErhs}{x,y}{#1}}

%%% Integral curve using Euler method. The step of this method is
%%% ODEstep and the number of steps is ODEstepcount. The points are
%%% stored in metapost variables x1,y1.
\def\trajectory#1#2{
  \begin{connect}
    \mfsrc{x1:=#1;y1:=#2;
      for i=1 upto ODEstepcount:
      x2:=x1+ODEstep;
      y2:=y1+ODEstep*ODErhs(x1,y1);}
    \ODEline{z1}{z2}
    \mfsrc{
      if ((y2>yneg) and (y2<ypos)): x1:=x2; y1:=y2 fi;
      endfor
    }
  \end{connect}
}

%%% Integral curve using Runge--Kutta method.
```

```
\def\trajectoryRK#1#2{
  \begin{connect}
    \mfsrc{x1:=#1;y1:=#2;
      for i=1 upto ODEstepcount:
      k1:=ODErhs(x1,y1);
      x3:=x1+(ODEstep/2);
      y3:=y1+k1*(ODEstep/2);
      k2:=ODErhs(x3,y3);
      x2:=x1+ODEstep;
      y2:=y1+ODEstep*k2;}
    \ODEline{z1}{z2}
    \mfsrc{
      if ((y2>yneg) and (y2<ypos)): x1:=x2; y1:=y2 fi;
      endfor
    }
  \end{connect}
}
%%% Integral curve using fourth order Runge--Kutta method.
\def\trajectoryRKF#1#2{
  \begin{connect}
    \mfsrc{x1:=#1;y1:=#2;
      for i=1 upto ODEstepcount:
      k1:=ODErhs(x1,y1);
      x3:=x1+(ODEstep/2);
      y3:=y1+k1*(ODEstep/2);
      k2:=ODErhs(x3,y3);
      y4:=y1+k2*(ODEstep/2);
      k3:=ODErhs(x3,y4);
      y5:=y1+k3*(ODEstep/2);
      k4:=ODErhs(x3,y5);
      kk:=(k1+2*k2+2*k3+k4)/6;
      x2:=x1+ODEstep;
      y2:=y1+ODEstep*kk;}
    \ODEline{z1}{z2}
    \mfsrc{
      if ((y2>yneg) and (y2<ypos)): x1:=x2; y1:=y2 fi;
      endfor
    }
  \end{connect}
}
\def\ODEarrow#1#2{
  \mfsrc{x1:=#1; y1:=#2;
    x3:=x1+(ODEarrowlength)/((xscale)++(ODErhs(#1,#2)*yscale));
    y3:=y1+(ODEarrowlength*ODErhs(#1,#2))/((xscale)++(ODErhs(#1,#2)*yscale));
    if y3>y1:ODEcolorarrow:=blue else: ODEcolorarrow:=red fi;
  }
  \ifcolorODEarrow
    \drawcolor{ODEcolorarrow} \headcolor{ODEcolorarrow}
  \fi
  \draw\arrow\lines{z1,z3}
```

```
}

\def\ODEarrows#1{\ODE@cycle@points#1;,;}
\def\trajectories#1{\ODE@cycle@IC#1;,;}
\def\ODE@last@point{}
\def\ODE@cycle@points#1,#2;{\def\temp{#1}\ifx\temp\ODE@last@point\let\next\relax
  \else\ODEarrow{#1}{#2}\relax\let\next\ODE@cycle@points\fi\next}
\def\ODE@cycle@IC#1,#2;{\def\temp{#1}\ifx\temp\ODE@last@point\let\next\relax
  \else
  \trajectoryRKF{#1}{#2}\relax\let\next\ODE@cycle@IC\fi\next}
\mfsrc{path p,q;color ODEcolorarrow;}

%%% Onedimensional autonomous systems y'=f(y) where  '=d/dx
\def\ODEharrow#1{
  \mfsrc{x1:=#1;
    if ODErhs(0,x1)>0: x3:=x1+ODEarrowlength else: x3:=x1-ODEarrowlength fi;
    if ODErhs(0,x1)*ODErhs(0,x3)<0: x1:=-100;x3:=-100 fi;
    if x3>x1:ODEcolorarrow:=blue else: ODEcolorarrow:=red fi;
  }
  \ifcolorODEarrow \drawcolor{ODEcolorarrow}
  \headcolor{ODEcolorarrow} \fi
  \pen{1.5pt}
  \draw\arrow\lines{(x1,0),(x3,0)}
}

\def\ODEvarrow#1{
  \mfsrc{x1:=#1;
    if ODErhs(0,#1)>0:
    x3:=x1+(ODEarrowlength/yscale) else: x3:=x1-(ODEarrowlength/yscale) fi;
    if ODErhs(0,x1)*ODErhs(0,x3)<0: x1:=-100;x3:=-100 fi;
    if x3>x1:ODEcolorarrow:=blue else: ODEcolorarrow:=red fi;
  }
  \ifcolorODEarrow \drawcolor{ODEcolorarrow}
  \headcolor{ODEcolorarrow} \fi
  \pen{1.5pt}
  \draw\arrow\lines{(0,x1),(0,x3)}
}

%%% Twodimensional autonomous systems  x'=f(x,y), y'=g(x,y) where '=d/dt
\def\ASdefineequations#1#2{\fdef{ASf}{x,y}{#1}\fdef{ASg}{x,y}{#2}}

\def\AStrajectory#1#2{
  \begin{connect}
    \mfsrc{x1:=#1;y1:=#2;
      for i=1 upto ODEstepcount:
      x2:=x1+ODEstep*ASf(x1,y1);
      y2:=y1+ODEstep*ASg(x1,y1);}
    \ODEline{z1}{z2}
    \mfsrc{
      if ((y2>yneg) and (y2<ypos)): x1:=x2; y1:=y2 fi;
```

```
      endfor
    }
  \end{connect}
}
\def\ASarrow#1#2{
  \mfsrc{x1:=#1; y1:=#2;
    x3:=x1+(ODEarrowlength*ASf(#1,#2))/((ASf(#1,#2)*xscale)++(ASg(#1,#2)*yscale    ));
    y3:=y1+(ODEarrowlength*ASg(#1,#2))/((ASf(#1,#2)*xscale)++(ASg(#1,#2)*yscale    ));
    if y3>y1:ODEcolorarrow:=blue else: ODEcolorarrow:=red fi;
  }
  \ifcolorODEarrow
  \drawcolor{ODEcolorarrow} \headcolor{ODEcolorarrow}
  \fi
  \draw\arrow\lines{z1,z3}
}

\def\ASarrows#1{\AS@cycle@points#1;,;}
\def\AS@cycle@points#1,#2;{\def\temp{#1}\ifx\temp\ODE@last@point\let\next\relax
  \else\ASarrow{#1}{#2}\relax\let\next\AS@cycle@points\fi\next}
\def\AStrajectories#1{\AS@cycle@IC#1;,;}
\def\AS@cycle@IC#1,#2;{\def\temp{#1}\ifx\temp\ODE@last@point\let\next\relax
  \else
  \AStrajectoryRKF{#1}{#2}\relax\let\next\AS@cycle@IC\fi\next}
\def\AStrajectoryRKF#1#2{
  \begin{connect}
    \mfsrc{x1:=#1;y1:=#2;
      TIMEsteps:=abs(TIMEend/TIMEstep);
      TIME:=0;
      for i=1 upto TIMEsteps:
      k1:=ASf(x1,y1);
      l1:=ASg(x1,y1);
      k2:=ASf(x1+(TIMEstep*k1/2),y1+(TIMEstep*l1/2));
      l2:=ASg(x1+(TIMEstep*k1/2),y1+(TIMEstep*l1/2));
      k3:=ASf(x1+(TIMEstep*k2/2),y1+(TIMEstep*l2/2));
      l3:=ASg(x1+(TIMEstep*k2/2),y1+(TIMEstep*l2/2));
      k4:=ASf(x1+(TIMEstep*k3),y1+(TIMEstep*l3));
      l4:=ASg(x1+(TIMEstep*k3),y1+(TIMEstep*l3));
      k5:=((k1)/6)+((k2)/3)+((k3)/3)+((k4)/6);
      l5:=(l1/6)+(l2/3)+(l3/3)+(l4/6);
      x2:=x1+(TIMEstep*k5);
      y2:=y1+(TIMEstep*l5);}
    \ODEline{z1}{z2}
    \mfsrc{
      if ((y2>yneg) and (y2<ypos) and (x2<xpos) and (x2>xneg)): x1:=x2; y1:=y2 fi;
      endfor
    }
  \end{connect}
}

\catcode`\@12\relax
```

184 ⟨/tex⟩
185 ⟨sty⟩`\input mfpic4ode.tex\relax`