PPODE SUITE
Tutorial

Pascal A. Pieters
p.a.pieters@student.tue.nl

June 1, 2014

This document provides a small tutorial to get started with the PPODE SUITE. For more details on specific options or the internals of the suite, please read the PPODE SUITE Manual provided with the package.

Installation

Dependencies

The PPODE SUITE currently only runs on Linux/Unix and requires MATLAB and GCC/GFortran to be installed. For details about the configuration of MATLAB's MEX please consult the MATLAB help and forums. Note that using newer versions of GCC than officially supported by Mathworks generally does not cause any issues, so warnings similar to the following can be ignored.

`"Warning: You are using gcc version .... The version currently supported with MEX is ..."`

When MEX commands raise the error "⟨MATLAB lib dir⟩`/libgfortran.so.3: version 'GFORTRAN_1.4' not found`", this can be solved by issuing the shell command:

```
$ sudo ln -sf ⟨GCC lib dir⟩/libgfortran.so ⟨MATLAB lib dir⟩/libgfortran.so.3
```

Where ⟨GCC lib dir⟩ is for example `"/usr/lib/gcc/x86_64-linux-gnu/4.8"`.

HPC ICMS Cluster

On the *HPC ICMS* cluster, all dependencies are already installed. MATLAB is however not correctly configured to use GFortran. This can be solved by issuing:

```
$ echo -e "\nalias g95=\"gfortran\"\n" >> ~/.profile && source ~/.profile
```

General

Unpack the PPODE SUITE package, the created directory will be referred to as ⟨Install Path⟩ from hereon.

```
$ tar -xzf PPODESUITE-0.*.tar.gz
```

Open MATLAB and change directory to the ⟨Install Path⟩.

```
» cd ⟨Install Path⟩
```

Build the libraries of the PPODE SUITE package.

```
» PPODE_init
```

This concludes the installation, for next sessions it will suffice to change directory to the ⟨Install Path⟩ and run the addPaths script to add the PPODE SUITE paths to the MATLAB path variable.

```
» PPODE_addPaths
```

Startup

To automatically load the PPODE SUITE paths into MATLAB upon statup, start MATLAB as root:

```
$ sudo matlab
```

And edit the startup script using the command:

```
» edit startup
```

Add the following lines to the file:

```
addpath '⟨Install Path⟩'
PPODE_addPaths
```

Prepartion

In MATLAB, change directory to the path of the MATLAB ODE file that should be evaluated using the PPODE SUITE. Open the file that contains the ODE definitions, i.e. the function file that is usually supplied as first argument to the MATLAB ODE solver (`ode15s(@`⟨odefunc⟩`, ...)`, which is defined in the file ⟨odefunc⟩.m). This function definition should have a header of the following form:

» ⟨dx⟩ = func(⟨t⟩, ⟨x⟩, ⟨par⟩, ⟨neq⟩, ⟨np⟩)

Where ⟨t⟩ is the independent variable, ⟨x⟩ the dependent variable(s) and ⟨par⟩ the parameter values. The last two arguments are optional and represent the number of equations (⟨neq⟩) and number of parameters (⟨np⟩). For further restrictions on the ODE function, see the PPODE SUITE Manual.

If the function fullfills the requirements of the PPODE SUITE, it can be parsed to the programming language of the PPODE SUITE (Fortran 95).

» PPODE_translate('⟨odefunc⟩')

This function will generate the file '⟨odefunc⟩.F', which can be build against the solver libraries provided.

» PPODE_build('⟨odefunc⟩.F', '⟨execID⟩')

Where ⟨execID⟩ can be any identifier for the executable, e.g. 'odefunc_StiffSolver'. This command does not specify any options, so all default values will be used. Therefore also the default solver is used, which is a fully automatic solver, for the use of other solvers consult the PPODE SUITE Manual.

Execution

The ODEs can now be solved by calling the executable generated by the build command.

» [⟨t⟩, ⟨y⟩] = ⟨execID⟩(⟨neq⟩, ⟨abstol⟩, ... ⟨reltol⟩, ⟨times⟩, ⟨par⟩, ⟨y0⟩)

Where ⟨neq⟩ is the number of equations, ⟨abstol⟩ and phreltol are the absolute and relative tolerances respectively, ⟨times⟩ is a vector of time points at which output is desired, ⟨par⟩ is a vector of parameter values and ⟨y0⟩ is a vector of the initial values of the states. The function returns the vector ⟨t⟩ with the time points at which the values of the states are calculated. The matrix ⟨y⟩ contains the values of all states at the time points specified by ⟨t⟩.

Note that the number of equations (⟨neq⟩) has a special role in the PPODE SUITE, since it was developed to work with large systems of ODEs that can have a variable number of equations. Therefore, it is important to always provide a correct number of equations. So if a system has a fixed number of equations, this number still has to be supplied. An easy way to do this is to use the size of the intial values, i.e. provide `length(⟨y0⟩)` as first argument.

Example

An example of this workflow can be found in the examples/rigid folder of the package. The function `compareSolvers` evaluates a simple stiff problem using various solvers. The ODEs are defined in the file rigid.m. The `compareSolvers` function nicely shows the differences between the use of a MATLAB ODE solver and the PPODE SUITE.

Listing 1: Code Snippet

```
PPODE_translate('rigid');
y0 = [0 1 1];
par = [-0.51];
t = 0:0.01:20;
options = odeset('RelTol', 1e-4, 'AbsTol', 1e-4);

% The ode15s solver
[t1, y1] = ode15s(@(t,y)(rigid(t, y, par)), ...
                  t, y0, options);
% The Adams-Moulton solver
PPODE_build('rigid.F', 'rigid_AM', ...
            'Solver','Adams-Moulton');
[t2, y2] = rigid_AM(length(y0), ...
                    options.AbsTol, ...
                    options.RelTol, ...
                    t, par, y0);

figure(1);
subplot(2, 1, 1);
plot(t1, y1(:, 1), 'r-', t1, y1(:, 2), 'g-', ...
     t1, y1(:, 3), 'b-');
xlabel('t');
ylabel('y(t)');
title('Rigid Example - ode15s');
subplot(2, 1, 2);
plot(t2, y2(:, 1), 'r-', t2, y2(:, 2), 'g-', ...
     t2, y2(:, 3), 'b-');
xlabel('t');
ylabel('y(t)');
title('Rigid Example - AM');
```

The output of this code is: