"StiffnessSwitching" Method for NDSolve

Introduction

The basic idea behind the "StiffnessSwitching" method is to provide an automatic means of switching between a nonstiff and a stiff solver.

The "StiffnessTest" and "NonstiffTest" options (described within "Stiffness Detection in NDSolve") provides a useful means of detecting when a problem appears to be stiff.

The "StiffnessSwitching" method traps any failure code generated by "StiffnessTest" and switches to an alternative solver. The "StiffnessSwitching" method also uses the method specified in the "NonstiffTest" option to switch back from a stiff to a nonstiff method.

"Extrapolation" provides a powerful technique for computing highly accurate solutions using dynamic order and step size selection (see "Extrapolation Method for NDSolve" for more details) and is therefore used as the default choice in "StiffnessSwitching".

Examples

This loads some useful packages:

```
1 »  Needs["DifferentialEquations`NDSolveProblems`"];
     Needs["DifferentialEquations`NDSolveUtilities`"];
```

This selects a stiff problem and specifies a longer integration time interval than the default specified by NDSolveProblem:

```
3 »  system = GetNDSolveProblem["VanderPol"];
     time = {T, 0, 10};
```

The default "Extrapolation" base method is not appropriate for stiff problems and gives up quite quickly:

```
5 »  NDSolve[system, time, Method → "Extrapolation"]
```

```
NDSolve: At T == 0.02232689421118968`, system appears to be stiff. Methods Automatic, BDF, or
   StiffnessSwitching may be more appropriate.
```

```
5 ›  {{Y1[T] → InterpolatingFunction[ Domain: {{0., 0.0223}} ][T]
```

```
{{Y1[T] → InterpolatingFunction[ ⊞ Output: scalar ][T],

    Y2[T] → InterpolatingFunction[ ⊞ Domain: {{0., 0.0223}}
                                      Output: scalar ][T]}}
```

Instead of giving up, the "StiffnessSwitching" method continues the integration with a stiff solver:

```
6 » NDSolve[system, time, Method → "StiffnessSwitching"]

6 › {{Y1[T] → InterpolatingFunction[ ⊞ Domain: {{0., 10.}}
                                      Output: scalar ][T],

    Y2[T] → InterpolatingFunction[ ⊞ Domain: {{0., 10.}}
                                      Output: scalar ][T]}}
```

The "StiffnessSwitching" method uses a pair of extrapolation methods as the default. The nonstiff solver uses the "ExplicitModifiedMidpoint" base method, and the stiff solver uses the "LinearlyImplicitEuler" base method.

For small values of the AccuracyGoal and PrecisionGoal tolerances, it is sometimes preferable to use an explicit Runge–Kutta method for the nonstiff solver.

The "ExplicitRungeKutta" method eventually gives up when the problem is considered to be stiff:

```
7 » NDSolve[system, time, Method → "ExplicitRungeKutta", AccuracyGoal → 5, PrecisionGoal → 4]

NDSolve: At T == 0.030750840966067998`, system appears to be stiff. Methods Automatic, BDF, or
  StiffnessSwitching may be more appropriate.

7 › {{Y1[T] → InterpolatingFunction[ ⊞ Domain: {{0., 0.0308}}
                                      Output: scalar ][T],

    Y2[T] → InterpolatingFunction[ ⊞ Domain: {{0., 0.0308}}
                                      Output: scalar ][T]}}
```

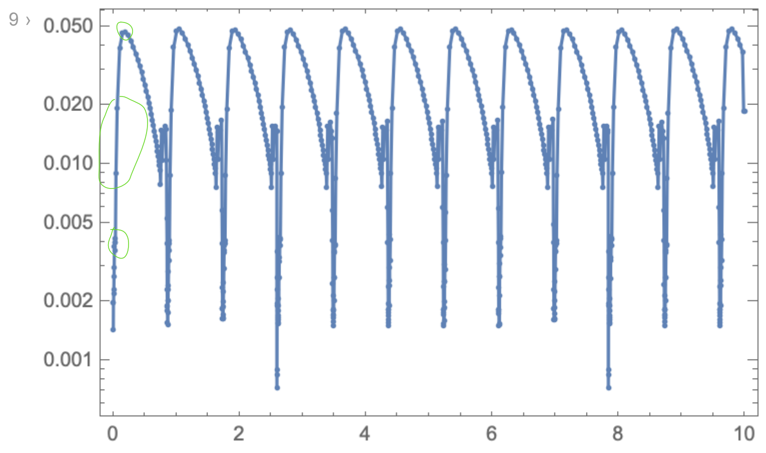
This sets the "ExplicitRungeKutta" method as a submethod of "StiffnessSwitching":

```
8 » sol = NDSolve[system, time,
       Method → {"StiffnessSwitching", Method → {"ExplicitRungeKutta", Automatic}},
       AccuracyGoal → 5, PrecisionGoal → 4]

8 › {{Y1[T] → InterpolatingFunction[ ⊞ Domain: {{0., 10.}}
                                      Output: scalar ][T],

    Y2[T] → InterpolatingFunction[ ⊞ Domain: {{0., 10.}}
                                      Output: scalar ][T]}}
```

A switch to the stiff solver occurs at $T \approx 0.0282294$, and a plot of the step sizes used shows that the stiff solver takes much larger steps:

```
9 » StepDataPlot[sol]
```

Top



Option Summary

option name	default value
Method	{Automatic, Automatic}
specify the methods to use for the nonstiff and stiff solvers respectively	
"NonstiffTest"	Automatic
specify the method to use for deciding whether to switch to a nonstiff solver	

Options of the method "StiffnessSwitching".

Feedback Top

Introduction for Programmers Introductory Book

Wolfram Function Repository | Wolfram Data Repository | Wolfram Data Drop | Wolfram Language Products

Top