Solver comparison (ode45 vs ode15s) with stiff/non-stiff ODEs

This code was successfully run in Matlab **R2021b**

Presentación en vídeo: http://personales.upv.es/asala/YT/V/ode45vs15sEN.html

Objetive: ODEs which are "*stiff*" have some time constants (poles of the linearisation) which are much faster than others. In some cases, a different solver from **`ode45`** might be recommended; we'll benchmark it against **`ode15s`**.

Table of Contents

Preliminaries

Let us have a linear model (the ideas do apply to non-linear ODEs, of course):

```
A=[-1 0.05 0;0 -1.25*30 0.05;0.0 0 -1.5*9000]
```

```
A = 3×3
10^4 ×
   -0.0001    0.0000         0
         0   -0.0037    0.0000
         0         0   -1.3500
```

```
eig(A)
```

```
ans = 3×1
10^4 ×
   -0.0001
   -0.0037
   -1.3500
```

```
odefun=@(t,x) A*x+[1; 2+2*cos(t/10); 3]*sin(t)^3;
```

So we wish to simulate it from a given initial condition, up to a final time, with some tolerances:

```
x0=[1;-1;1];
tf=500;
opts=odeset(AbsTol=1e-4,RelTol=1e-5);
```

Numerical integration, solver comparison

Using ode45

```
tic
```

```
[T,X]=ode45(odefun,[0 tf],x0,opts);
toc
```

```
Elapsed time is 16.534063 seconds.
```

```
size(T)
```

```
ans = 1×2
     8135289           1
```

Using ode15s

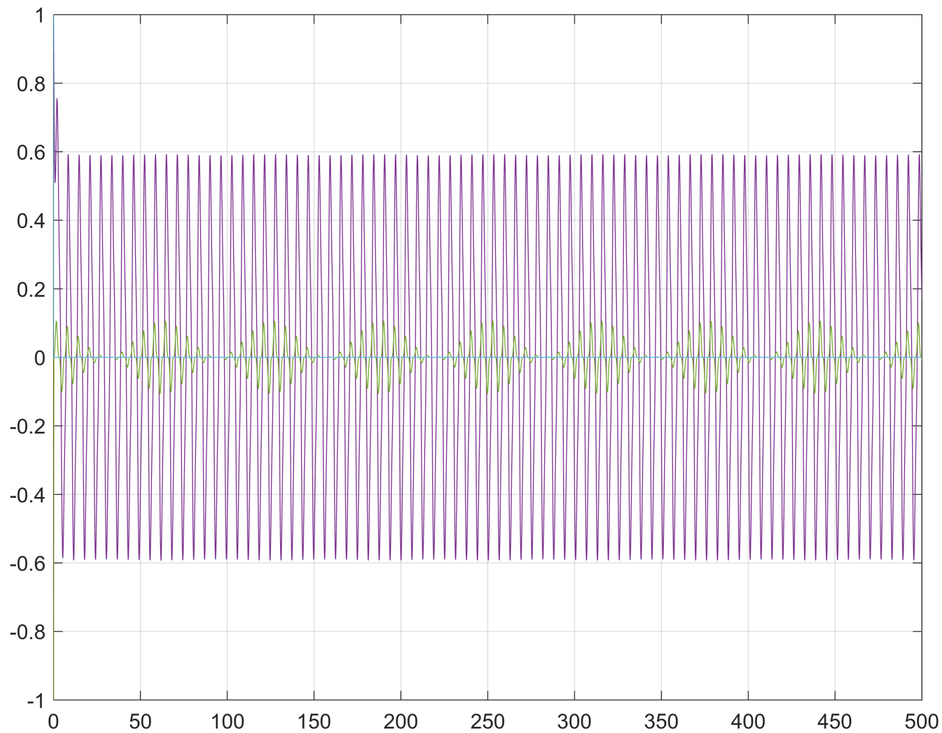
```
tic
[T2,X2]=ode15s(odefun,[0 tf],x0,opts);
toc
```

```
Elapsed time is 0.071414 seconds.
```

```
size(T2)
```

```
ans = 1×2
        3981           1
```

```
plot(T,X), hold on
plot(T2,X2), hold off, grid on
```



Conclusions

ode45 is the basic "try this first" recommendation in Matlab documentation for smooth non-stiff problems (all time constants in the same order of magnitude). With very different time constants, it might take longer than other solvers. For instance, ODE solver **ode15s** takes longer than ode45 in non-stiff problems, but it is faster in stiff ones.

In complex models (high order with tens of thousands of state variables, say finite element ones), wildly varying time constants, discontinuities due to impacts, singularities, impulsive initial conditions... well, choosing the most appropriate solver may require a careful assessment by an expert in the topic before running a simulation for "two weeks" yielding a "bad/wrong result".

Actually, Matlab lists a lot of choices for ODE solver:

ode45 Nonstiff Medium

Most of the time. ode45 should be the first solver you try.

ode23 Low

ode23 can be more efficient than ode45 at problems with crude tolerances, or in the presence of moderate stiffness.

ode113 Low to High

ode113 can be more efficient than ode45 at problems with stringent error tolerances, or when the ODE function is expensive to evaluate.

ode78 High

ode78 can be more efficient than ode45 at problems with smooth solutions that have high accuracy requirements.

ode89 High

ode89 can be more efficient than ode78 on very smooth problems, when integrating over long time intervals, or when tolerances are especially tight.

ode15s Stiff Low to Medium

Try ode15s when ode45 fails or is inefficient and you suspect that the problem is stiff. Also use ode15s when solving differential algebraic equations (DAEs).

`ode23s` Low

`ode23s` can be more efficient than `ode15s` at problems with crude error tolerances. It can solve some stiff problems for which `ode15s` is not effective.

`ode23s` computes the Jacobian in each step, so it is beneficial to provide the Jacobian via `odeset` to maximize efficiency and accuracy.

If there is a mass matrix, it must be constant.

`ode23t` Low

Use `ode23t` if the problem is only moderately stiff and you need a solution without numerical damping.

`ode23t` can solve differential algebraic equations (DAEs).

`ode23tb` Low

Like `ode23s`, the `ode23tb` solver might be more efficient than `ode15s` at problems with crude error tolerances.

`ode15i` Fully implici tLow

Use `ode15i` for fully implicit problems *f(t,y,y') = 0* and for differential algebraic equations (DAEs) of index 1.