ode45 in MATLAB

Dr Willie Brink

ODE45 is 'n funksie in Matlab wat stelsels van 1ste-orde aanvangswaardeprobleme numeries oplos. Dit is 'n aanpasbare tydstap metode gebaseer op 'n 4de-orde Runge-Kutta formule gekombineer met 'n 5de-orde formule vir foutskatting. Hierdie dokument verduidelik kortliks hoe die funksie gebruik kan word. Matlab se dokumentasie (`doc ode45`) kan geraadpleeg word vir verdere inligting.

ODE45 is a function in Matlab that solves systems of 1st-order initial value problems numerically. It is an adaptive time step method based on a 4th-order Runge-Kutta formula combined with a 5th-order formula for error estimation. This document explains briefly how the function can be used. Matlab's documentation (`doc ode45`) can be consulted for further information.

Beskou die volgende stelsel van 1ste-orde differensiaalvergelykings met aanvangsvoorwaardes (hier het ons n afhanklike veranderlikes):

Consider the following system of 1st-order differential equations with initial conditions (here we have n dependent variables):

$$\begin{aligned}
\frac{dv_1}{dt} &= f_1(t, v_1, v_2, \ldots, v_n), \quad v_1(t_0) = a_1,\\
\frac{dv_2}{dt} &= f_2(t, v_1, v_2, \ldots, v_n), \quad v_2(t_0) = a_2,\\
&\vdots\\
\frac{dv_n}{dt} &= f_n(t, v_1, v_2, \ldots, v_n), \quad v_n(t_0) = a_n.
\end{aligned}$$

'n Oplossing vir hierdie stelsel is funksies $v_1(t)$, $v_2(t)$, …, $v_n(t)$ wat bostaande DVs en aanvangswaardes bevredig.

A solution for this system is a set of functions $v_1(t)$, $v_2(t)$, …, $v_n(t)$ that satisfies the DEs and initial conditions above.

Hierdie stelsel kan in matriksvorm geskryf word:

This system can be written in matrix form:

$$\frac{d\mathbf{v}}{dt} = \mathbf{f}(t, \mathbf{v}), \quad \mathbf{v}(t_0) = \mathbf{a},$$

waar

where

$$\mathbf{v}(t) = [v_1(t),\ v_2(t),\ \ldots,\ v_n(t)]^T, \quad \mathbf{f} = [f_1(t,\mathbf{v}),\ f_2(t,\mathbf{v}),\ \ldots,\ f_n(t,\mathbf{v})]^T, \quad \mathbf{a} = [a_1,\ a_2,\ \ldots,\ a_n]^T.$$

Die funksie `ode45` aanvaar die volgende sintaks:

The function `ode45` allows the following syntax:

```
[t,v] = ode45(f, [tmin tmax], a);
```

Die toevoer-parameters is:

(1) `f` : 'n vektor van funksies wat die regterkant van bostaande stelsel van DVs beskryf;
(2) `[tmin tmax]` : 'n interval vir t waaroor die oplossing gevind moet word (let op: `tmin` moet ooreenstem met t_0 hierbo);
(3) `a` : die aanvangswaardes, soos $\mathbf{a}$ hierbo.

The input parameters are:

(1) `f` : a vector of functions that describes the righthand side of the above system of DEs;
(2) `[tmin tmax]` : an interval for t over which the solution is to be found (note: `tmin` should correspond to t_0 above);
(3) `a` : the initial values, like $\mathbf{a}$ above.

Die funksie lewer 'n kolomvektor `t`, sê van lengte m, en 'n $m \times n$ matriks `v`. Die elemente van `t` is diskrete tydstappe $t_1, t_2, \ldots, t_m$ waarby oplossings benaderd gevind is. Die element $v(i, j)$, d.w.s. in ry i en kolom j, is 'n numeries-berekende benaderde waarde van $v_j(t_i)$.

The function returns a column vector `t`, say of length m, and an $m \times n$ matrix `v`. The elements of `t` are discrete time steps $t_1, t_2, \ldots, t_m$ on which solutions were approximated. The element $v(i, j)$, i.e. in row i and column j, is a numerically calculated approximate value of $v_j(t_i)$.

Voorbeeld 1: Beskou die probleem

$$y' + 2y^2 = 6t + 5, \quad y(0) = 3,$$

waar ons belangstel in 'n oplossing $y = y(t)$ met $t \in [0, 2]$. Ons skryf die DV eers in die vorm

$$\frac{dy}{dt} = 6t + 5 - 2y^2,$$

en kan nou `ode45` inspan om die probleem numeries op te los:

```
>> f = @(t,y) 6*t + 5 - 2*y^2;
>> [t,y] = ode45(f, [0 2], 3);
```

Let op: ons het slegs een DV, dus is `f` 'n funksie wat twee getalle (`t` en `y`) afbeeld na 'n enkele getal (leer meer oor die gebruik van die `@`-simbool deur "anonymous functions" in Matlab se dokumentasie na te slaan). Ons het ook slegs een aanvangswaarde (`3`) in hierdie geval.

Bostaande kode lewer twee kolomvektore: `t` en `y`. In hierdie geval het elkeen 53 elemente. Ons kan nou 'n grafiek teken van y teenoor t:

__Example 1:__ Consider the problem

where we are interested in a solution $y = y(t)$ with $t \in [0, 2]$. We first write the DE in the form

and can now use `ode45` to solve the problem numerically:

Note: we have only one DE, therefore `f` is a function that maps two numbers (`t` and `y`) to a single number (learn more about the usage of the `@`-symbol by searching for "anonymous functions" in Matlab's documentation). In this case we also have one initial condition (`3`).

The code above produces two column vectors: `t` and `y`. In this case each one has 53 elements. We can now draw a graph of y vs t:

```
>> figure
>> plot(t,y)
```

Voorbeeld 2: Beskou die stelsel

$$\frac{dx}{dt} = \left(-\tfrac{2}{25}\right)x + \left(\tfrac{1}{50}\right)y, \quad x(0) = 25,$$
$$\frac{dy}{dt} = \left(\tfrac{2}{25}\right)x - \left(\tfrac{2}{25}\right)y, \quad y(0) = 0.$$

Ons stel belang in oplossings $x(t)$ en $y(t)$ met $t \in [0, 100]$. Ons kan die volgende in Matlab uitvoer:

__Example 2:__ Consider the system

We are interested in solutions $x(t)$ and $y(t)$ for $t \in [0, 100]$. We can execute the following in Matlab:

```
>> f = @(t,v) [(-2/25)*v(1) + (1/50)*v(2); (2/25)*v(1) - (2/25)*v(2)];
>> v0 = [25; 0];
>> [t,v] = ode45(f, [0 100], v0);
```

Let op dat die funksie `f` nou 'n 2×1 vektor lewer, met eerste element gelyk aan dx/dt en die tweede gelyk aan dy/dt. Dis ook belangrik om te sien dat `f` steeds twee parameters as toevoer neem (`t`, en `v` wat nou 'n vektor is) hoewel slegs een daarvan in die definisie voorkom. Dit is 'n konvensie wat deur `ode45` gevolg word. Let ook op dat daar nou twee aanvangswaardes gegee word in die vorm van die 2×1 vektor `v0`.

Om die afvoer te stip is steeds so maklik soos `plot(t,v)`. Matlab aanvaar dat twee kolomme in `v` dui op twee aparte funksies in `t`.

Note that the function `f` now produces a 2×1 vector, with first element equal to dx/dt and the second element equal to dy/dt. It is also important to see that `f` still requires two input parameters (`t`, and `v` which is now a vector) even though only one of them is used in the definition. This is a convention followed by `ode45`. Also note that there are now two initial conditions given in the form of the 2×1 vector `v0`.

Plotting the results is still as easy as `plot(t,v)`. Matlab assumes that two columns in `v` indicate two separate functions in `t`.
