PENYELESAIAN MASALAH SYARAT BATAS PERSAMAAN DIFERENSIAL BIASA DALAM *SOFTWARE* R DENGAN MENGGUNAKAN BVPSOLVE

W. ERLIANA[1], A. D. GARNADI[1], S. NURDIATI[1], M.T. JULIANTO[1]

Abstrak

Diuraikan penggunaan paket bvpSolve di lingkungan R untuk menyelesaikan masalah syarat batas sistem persamaan diferensial biasa. Tujuannya ialah agar pengguna dapat mempergunakan bvpSolve setelah membaca uraian penggunaannya. Penggunaan paket bvpSolve diilustrasikan dengan dua contoh yang memperlihatkan kegunaannya.

PENDAHULUAN

Persamaan Diferensial Biasa (PDB) sering muncul sebagai model permasalahan dalam berbagai bidang ilmu pengetahuan.Pencarian solusi PDB diperlukan untuk memperoleh interpretasi dari model permasalahan semula. Solusi analitik dari suatu PDB tidak selalu mudah diperoleh. Pada umumnya, untuk mencari salah satu solusi yang dikehendaki dari suatu sistem PDB, diperlukan suatu nilai awal yang memenuhi sistem PDB tersebut pada suatu titik $x = a$. Permasalahan seperti ini dikenal dengan nama Masalah Nilai Awal (MNA), yang secara ringkas dapat dinyatakan oleh:

$$\begin{aligned} y^n &= f(x, y, y', y'', \dots, y^{n-1}) \\ y(a) &= b \end{aligned} \tag{1}$$

Ada kalanya solusi yang sangat spesifik ingin didapatkan dari suatu sistem PDB dengan cara menentukan nilai-nilai awal pada lebih dari satu titik *x*. Permasalahan seperti ini dikenal dengan nama Masalah Syarat Batas (MSB), yang dapat dinyatakan oleh :

$$\begin{aligned} y^n &= f(x, y, y', y'', \dots, y^{n-1}) \\ y(a_1) &= b_1 \\ y(a_2) &= b_2 \end{aligned} \tag{2}$$

Paket (*package*) numerik yang mampu menyelesaikan MNA telah banyak dikembangkan. Namun, sedikit yang mampu menyelesaikan MSB. Sebuah paket yang digunakan untuk menyelesaikan MSB berdasarkan metode kolokasi disediakan dalam lingkungan R, paket ini bernama bvpSsolve. Selain dalam R, metode kolokasi juga digunakan dalam lingkungan Scilab untuk rutin bvode yang disediakannya [1].

---

[1]Departemen Matematika, Fakultas Ilmu Pengetahuan Alam, Jalan Meranti Kampus IPB Dramaga Bogor,16680.

Tulisan ini merupakan sebuah tutorial yang memberikan penuntun bagaimana memformulasikan masalah (menyusun perintah), mendapatkan solusi numerik, dan menggambarkan solusi secara grafis dari MSB dengan menggunakan paket bvpSolve. Tulisan ini juga merupakan studi pendahuluan numerik atas ketersediaan lingkungan pemecah masalah numeric, *problem solving environment* atau PSE, yang bersifat *open-source* sebagai alternatif dari PSE komersial populer MATLAB yang menawarkan rutin bvp4c untuk memecahkan MSB.

Salah satu tujuan tulisan ini bersifat pedagogis yang menguraikan secara sederhana bagaimana memperoleh solusi numerik MSB sebuah PDB. Dengan demikian, seorang pemula dapat mempergunakan bvpSolve untuk menyelesaikan MSB yang dihadapinya tanpa kesulitan. Selain itu, tulisan ini bertujuan untuk menyediakan dokumentasi tertulis berbahasa Indonesia. Tujuan lainnya ialah memberikan teladan penggunaan *Open Source* di lingkungan komputasi matematika Indonesia, yang secara tidak langsung memberi dukungan pada proyek nasional IGOS (*Indonesia Goes Open Source*).

Tulisan ini disusun dengan urutan sebagai berikut. Pertama diuraikan deskripsi dari bvpSolve. Kemudian diperlihatkan penggunaannya untuk tiga buah contoh di luar contoh yang diberikan dalam lingkungan R tentang bagaimana mempergunakan bvpSolve untuk menyelesaikan sebuah MSB. Kemudian diakhiri oleh kesimpulan.

DESKRIPSI BVPSOLVE

Paket bvpSolve yang tersedia dalam PSE R merupakan perangkat simulasi untuk menyelesaikan MSB dari suatu PDB. Paket bvpSolve pada dasarnya mempergunakan pustaka colnew [2] dan colsys [3], juga mengimplementasikan *shooting method*. Bentuk MSB yang diasumsikan oleh bvpSolve ialah:

$$\frac{d^{m_i}y_i}{dx^{m_i}} = f_i\left(x, y(x), \frac{dy}{dx}, \dots, \frac{d^{m_i-1}y_i}{dx^{m_i-1}}\right), \qquad 1 \le i \le n_c, \tag{3}$$

$$g_j\left(\zeta_j, y(\zeta_j), \dots, \frac{d^{m_*}y}{dx^{m_*}}(\zeta_j)\right) = 0, \qquad j = 1, \dots, m_* \tag{4}$$

dengan ζ_j merupakan posisi di mana syarat batas berlaju dan $a_L \le \zeta_j \le a_R$. Agar notasi tidak menyulitkan, tuliskan

$$m_* = m_1 + m_2 + \cdots + m_{n_c},$$

$$z(y) = \left[y, \frac{dy}{dx}, \dots, \frac{d^{m_*}y}{dx^{m_*}}\right].$$

Dengan demikian, bentuk umum MSB yang diasumsikan oleh bvpSolve ialah

$$\frac{d^{m_i} y_i}{dx^{m_i}} = f_i\left(x, z(y(x))\right), \qquad 1 \le i \le n_c, \qquad a_L \le \zeta_j \le a_R, \tag{5}$$

$$g_j\left(\zeta_j, z\left(y(\zeta_j)\right)\right) = 0, \qquad j = 1, \ldots, m_*. \tag{6}$$

Paket bvpSolve memiliki kemampuan untuk menyelesaikan MSB yang linear maupun non-linear. Karena itu, paket ini mengharuskan pengguna menyusun sendiri matriks Jacobian dari PDB yang hendak diselesaikan, sehingga untuk beberapa masalah, tingkat kerumitan yang paling besar akan terasa pada penyusunan matriks Jacobian ini.

Dalam paket bvpSolve terdapat beberapa fungsi, yaitu bvptwp, bvpcol, dan bvpshoot. Fungsi bvptwp menggunakan metode penyelesaian MSB berdasarkan formula *Mono-Implicit Runge-Kutta* (MIRK), sedangkan fungsi bvpcol menggunakan metode kolokasi untuk menyelesaikan MSB *multi-point*, dan bvpshoot menggunakan *shooting method* (menggunakan *solvers* dari paket deSolve dan rootSolve). Dalam tulisan ini, hanya difokuskan untuk menggunakan fungsi bvpcol. Tata cara pemanggilan bvpcol ialah sebagai berikut:

z = bvpcol (yini = NULL, x, func, yend = NULL, parms=NULL, order = NULL, ynames = NULL, xguess = NULL, yguess = NULL, jacfunc = NULL, bound = NULL, jacbound = NULL, leftbc = NULL, posbound = NULL, islin = FALSE, nmax = 1000, ncomp = NULL, atol = 1e-8, colp = NULL, bspline = FALSE, ...),

dengan z merupakan vektor baris yang berisi solusi numerik dari MSB yang ingin diselesaikan. Komponen `z(i,:)` merepresentasikan turunan ke-(*i*-1) dari solusi pada selang domain. Sebagai contoh, `z(1,:)` menyatakan *y*, `z(2,:)` menyatakan *y'*, dan seterusnya hingga `z(m*,:)` menyatakan $y^{(m^*-1)}$.

Berikut adalah penjelasan kegunaan masing-masing parameter yang digunakan oleh bvpcol.

yini : Vektor nilai awal untuk sistem PDB.

Jika yini berupa vektor, gunakan NA untuk nilai awal yang tidak ditentukan.

Jika yini = NULL, maka kondisi batas harus ditentukan melalui fungsi bound; jika tidak NULL maka yend juga harus tidak NULL.

x : Barisan dari peubah bebas untuk output yang diinginkan; nilai pertama dari x harus merupakan nilai awal (di mana yini didefinisikan), nilai akhir merupakan nilai yend.

func : Fungsi yang menghitung nilai turunan dari system PDB pada titik x, dapat didefinisikan sebagai berikut:

func = function(x, y, parms, …)

yend	:	Vektor nilai akhir untuk sistem PDB. Jika yend berupa vektor, gunakan NA untuk nilai akhir yang tidak ditentukan. Jika yend = NULL, maka kondisi batas harus ditentukan melalui fungsi bound; jika tidak NULL maka yini juga harus tidak NULL.
parms	:	Vektor atau daftar parameter yang berkaitan dengan func, jacfunc, bound, dan jacbound (jika ada). Jika eps diberikan nilainya, maka eps harus menjadi elemen pertama pada parms.
epsini	:	Nilai awal dari *continuation parameter*. Jika NULL dan eps diberikan nilainya, maka epsini memiliki default nilai sebesar 0.5. Untuk beberapa tipe masalah perturbasi, nilai $0.1 < eps < 1$ merepresentasikan masalah yang mudah. Pengguna harus menentukan masalah awal yang tidak terlalu rumit. Nilai epsini berada di antara 0 dan 1.
eps	:	Nilai presisi yang diinginkan oleh pengguna untuk menyelesaikan masalah. eps $\leq$ epsini.
ynames	:	Nama suatu peubah, digunakan untuk memberi nama output.
xguess	:	Vektor, *initial grid*. Jika xguess didefinisikan, maka yguess juga harus didefinisikan.
yguess	:	Tebakan awal nilai y, berhubungan dengan xguess; suatu matriks dengan banyaknya baris sama dengan banyaknya peubah, dan banyaknya kolom sama dengan panjang vektor xguess.
jacfunc	:	Jacobian dari func pada titik x, dapat didefinisikan sebagai berikut: jacfunc = function(x, y, parms, …).
bound	:	Fungsi batas (opsional) – didefinisikan jika yini dan yend tidak diberikan. Bound = function(i, y, parms, …)
jacbound	:	Jacobian dari fungsi batas (opsional) – didefinisikan jika bound diberikan.
posbound	:	Hanya digunakan jika bound diberikan.
islin	:	TRUE jika masalah linear.
nmax	:	Nilai maksimal dari subinterval.
order	:	Orde dari setiap turunan pada func.
ncomp	:	Banyaknya komponen atau persamaan diferensial.

atol	:	Toleransi error (skalar).
colp	:	Banyaknya titik kolokasi per subinterval.
bspline	:	Jika FALSE, maka *code colnew* digunakan sebagai default. Jika TRUE, maka digunakan *fortran code colsys*.

CONTOH

Pada bagian ini akan dibahas dua contoh bagaimana menyelesaikan MSB dengan mempergunakan bvpcol. Pada bagian ini juga akan ditunjukkan bagaimana cara menggambar masing-masing komponen solusi numerik yang didapat dari bvpcol.

Contoh 1. Dalam contoh ini diberikan ilustrasi bagaimana mencari solusi numerik dari MSB yang melibatkan parameter yang tidak diketahui. Permasalahannya ialah menghitung nilai eigen dari persamaan Mathieu berikut ini,

$$y'' + \big(\lambda - 2qcos(2x)\big)y = 0 \tag{7}$$

pada selang $[0, \pi]$, dengan syarat batas $y'(0) = 0$, $y'(\pi) = 0$ untuk $q = 5$. Solusi dinormalisasi dengan cara menetapkan solusi memenuhi $y(0) = 1$. Permasalahan sesungguhnya ialah mencari nilai λ yang memenuhi syarat batas $y'(\pi) = 0$. Nilai tebakan awal bagi λ menjadi keharusan dalam menyelesaikan masalah ini. MSB di atas diimplementasikan dalam R sebagai berikut ini. Persamaan diferensialnya didefinisikan dengan skrip berikut.

```
fun<-function(t, y, pars) {
dy1<-y[2]
dy2<-(-lamda+2*q*cos(2*t)*y[1])
return(list(c(dy1,dy2)))}.
```

Sementara syarat batasnya diberikan oleh:

```
init<-c(NA, 0)
end<-c(NA, 0).
```

Berikut sejumlah parameter yang diperlukan, yaitu

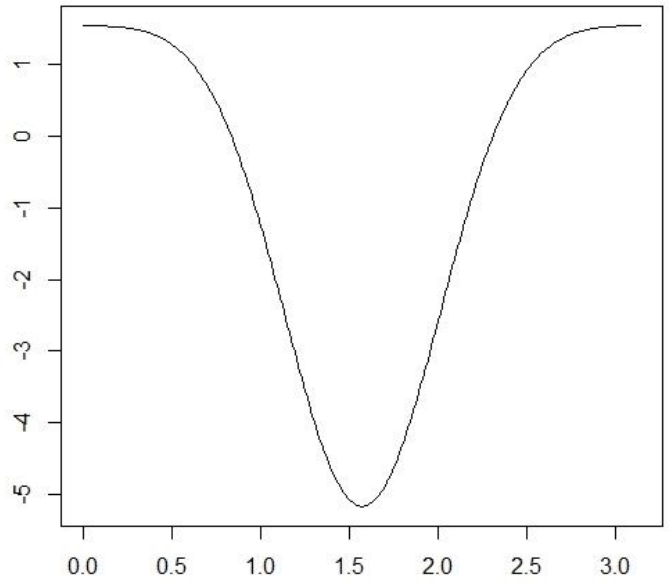
```
q<-5
lamda0<-15.
```

Jika tebakan awal bagi nilai eigen dipilih λ= 15, maka untuk menyelesaikan MSB pada persamaan (7) sekaligus mendapatkan nilai pendekatan untuk λ dapat dilakukan dengan menambahkan *loop* seperti pada *listing* berikut,

```
for(i in 1:0.001:20000){
lamda<-lamda0+i
sol<-bvpcol(yini = init, yend = end,
```

```
x = seq(0, pi, by = 0.01), func = fun)
if(abs(sol[i])<i)
break}
plot(sol, which = 1)
```

Dari hasil penghitungan yang dilakukan oleh bvpcol beserta *loop* di atas, didapat nilai pendekatan $\lambda = 16$ dan gambar grafik solusi $y(x)$ seperti yang terlihat pada Gambar 1. Penghitungan di atas mempergunakan mesh dengan lebar kisi $\Delta x = 0.01$ yang seragam untuk selang $[0,\pi]$.



Gambar 1 Fungsi eigen dari persamaan Mathieu terkait dengan nilai eigen yang digunakan dalam artikel

Contoh 2. Pada contoh berikut diberikan sebuah MSB yang berasal dari Kontrol Optimal. Biasanya, syarat cukup untuk kontrol optimal berupa MSB, yang biasanya lebih mudah menyelesaikannya untuk banyak masalah sederhana.

Perhatikan masalah sistem terkontrol

$$y' = y^2 + v$$

dan kita inginkan kontrol v mengendalikan lintasan dari 2 pada saat $t = 0$ ke -1 pada saat $t = 10$. Dengan demikian, kita inginkan v meminimumkan ongkos kuadratik berikut

$$J(y, v) = \int_0^{10} (y^2 + 10v^2)\, dt.$$

Fungsi ongkos ini menyebabkan variabel keadaan y dan variabel kontrol v terpenalisasi, sehingga menyebabkan variabel keadaan dan kontrol menjadi kecil. Syarat cukup dapat diperoleh dengan mendefinisikan fungsi Hamiltonian

$$H = (y^2 + 10v^2) + \lambda(y^2 + v)$$

dan kita tuliskan masalah syarat batas persamaan diferensial aljabar

$$\begin{aligned} y' &= \frac{\partial H}{\partial \lambda}, \\ \lambda' &= \frac{\partial H}{\partial y}, \\ 0 &= \frac{\partial H}{\partial v}, \\ y(0) = 2, y(10) &= 1, \end{aligned} \tag{8}$$

bila*v* dieliminasi, akan diperoleh MSB

$$\begin{aligned} y' &= y^2 + v, \\ \lambda' &= -2y - 2\lambda y, \\ 0 &= 20v + \lambda, \\ y(0) = 2, y(10) &= 1, \end{aligned} \tag{9}$$

MSB di atas diimplementasikan dalam R sebagai berikut ini. Persamaan diferensialnya didefinisikan dengan skrip berikut.

```
feval<-function(x,y, pars)
   return(list(c(y[1]^2 - y[2]/20, -2*y[1] - 2*y[2]*y[1]))).
```

Sementara syarat batasnya diberikan oleh:

```
yini<-c(2, NA)
yend<-c(1,NA).
```

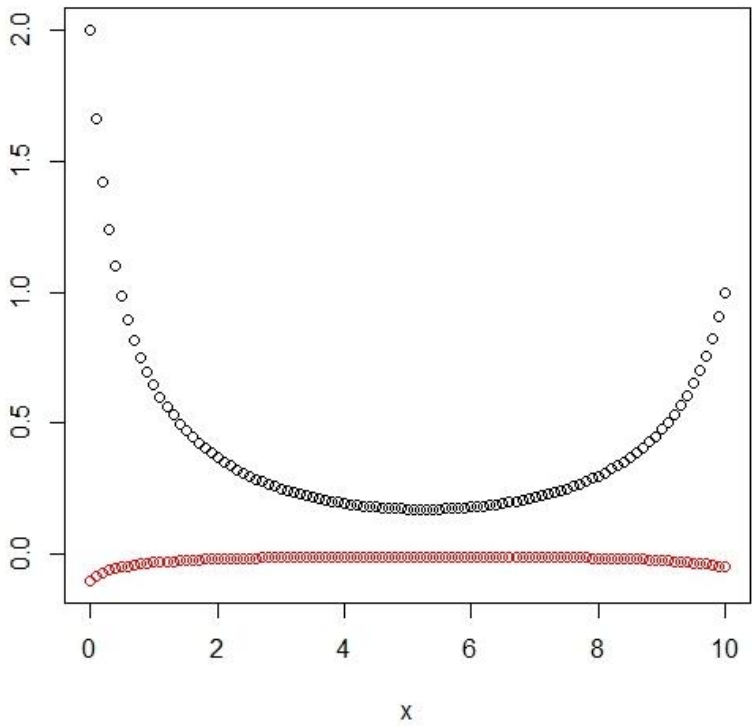
MSB pada persamaan (9) diselesaikan pada selang [0,10] dengan lebar kisi Δt = 0.1 yang seragam. Kemudian untuk memperoleh solusi numerik MSB, diberikan dalam skrip berikut:

```
x<-seq(from = 0, to = 10, by = 0.1)
sol<-bvpcol(yini = yini, yend = yend, x = x, func = feval)
sol1<-sol[,c("1")]
control<-(-1/20)*sol1.
```

Solusi numerik yang diperoleh, ditampilkan oleh Scilab dengan memberikan skrip berikut ini

```
plot(x,sol1,type='p',ylim=range(sol1,control),xlab='x',ylab=
'y')
points(x, control, type='p', col="red", xlab='x', ylab='y').
```

Tampilan grafisnya diperlihatkan pada Gambar 2.



Gambar 2 Optimal *y* (warna hitam) dan kontrol *v* (warna merah) untuk Contoh 2.

SIMPULAN

Masalah syarat batas sering muncul sebagai model dari berbagai bidang ilmu. Karena solusi analitik MSB tidak selalu tersedia, metode numerik menjadi salah satu cara untuk memperoleh solusinya. Selain itu, dengan tersedianya paket yang memudahkan pengguna untuk menyelesaikan MSB secara numerik, dimungkinkan melakukan simulasi secara berulang untuk menguji berbagai skenario pemodelan. Tulisan ini mendemonstrasikan paket simulasi MSB dalam lingkungan problem solving environment R yang bersifat *open source*. Perlu kiranya pengujian lebih lanjut atas paket ini, seperti halnya dilakukan oleh [4] yang menguji rutin simulasi MSB di lingkungan komersial MATLAB. Pengujian beragam MSB dari berbagai bidang ilmu sebagaimana diberikan oleh [5], dapat diujikan di lingkungan R untuk bahasan lanjutan.

UCAPAN TERIMA KASIH

Dibiayai secara parsial oleh Hibah Penelitian PUPT-IPB melalui nomor kontrak : 083/SP2H/PL/Dit.Litabmas/II/2015. Terima kasih kepada DAAD atas bantuan literatur ilmiah atas dasar kontrak no A/97/42703.

DAFTAR PUSTAKA

[1] Garnadi AD, Ayatullah F, Ekastrya D, Nurdiati S, Julianto MT. 2015. Menyelesaikan Masalah Syarat Batas Persamaan Diferensial Biasa dalam Scilab dengan menggunakan bvode. *Jurnal Matematika dan Aplikasinya*. 14(1): 55-68.

[2] Bader G, Ascher U. 1987. A new basis implementation for a mixed order boundary value ode solver, *siam j. scient.* stat. comput. 8: 487-483.

[3] Shampine L, Kierzenka J, Reichelt M. Solving Boundary Value Problems for Ordinary Differential Equations in Matlab with bvp4c, [http://www.mathworks.com/support/ solutions/files/s8314/bvp_paper.pdf]

[4] Ascher U, Christiansen J, Russell RD. 1981. Collocation software for boundary-value odes. *acm trans. math software*. 7: 209-222.

[5] Ascher U, Mattheij RM, Russel RD. *Numerical Solution of Boundary Value Problem*. Prentice-Hall, Englewood Cliffs, New Jersey.