Wykład 9: Metody Rungego-Kutty – sterowanie krokiem

ilustracja programu – ciekawostka(?) ostrzeżenie numeryczne

1 Metoda Butchera – podana w internecie

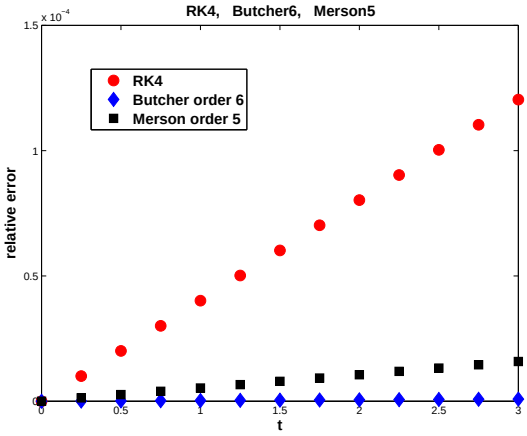
Pod adresem **http://elm.eeng.dcu.ie/~ee317/Matlab_Examples/ode/tutinfo[1].htm#ode3** była w roku ubiegłym praca dydaktyczna, ilustrująca wykorzystanie trzech metod RK.
Dziś adres ten nie istnieje, ale szukając skryptu ***mat_rkgen***, znajdziemy też funkcję ***rkgen*** z trzema metodami: ***Classical*** (RK4), ***Butchera*** oraz ***Mersona***.

```
function[tvals,yvals]=rkgen(f,tspan,startval,step,method)

% Runge Kutta methods for solving
% first order differential equation dy/dt = f(t,y).
% Example call:[tvals,yvals]=rkgen(f,tspan,startval,step,method)
% The initial and final values of t are given by tspan=[start finish].
% Initial y is given by startval and step size is given by step.
% The function f(t,y) must be defined by the user.
% For an example of this function definition, see page 160.
% The parameter method (1, 2 or 3) selects
% Classical, Butcher or Merson RK respectively.
b=[ ];c=[ ];d=[ ];
if method <1 | method >3
  disp('Method number unknown so using Classical');
  method=1;
end;
if method==1
  order=4;
  b=[ 1/6 1/3 1/3 1/6]; d=[0 .5 .5 1];
  c=[0 0 0 0;0.5 0 0 0;0 .5 0 0;0 0 1 0];
  disp('Classical method selected');
elseif method ==2
  order=6;
  b=[0.07777777778 0 0.355555556 0.13333333 0.355555556 0.0777777778];
  d=[0 .25 .25 .5 .75 1];
  c(1:4,:)=[0 0 0 0 0 0;0.25 0 0 0 0 0;0.125 0.125 0 0 0 0; 0 -0.5 1 0 0 0];
  c(5,:)=[.1875 0 0 0.5625 0 0];
  c(6,:)=[-.4285714 0.2857143 1.714286 -1.714286 1.1428571 0];
  disp('Butcher method selected');
else
  order=5;
  b=[1/6 0 0 2/3 1/6];
  d=[0 1/3 1/3 1/2 1];
  c=[0 0 0 0 0;1/3 0 0 0 0;1/6 1/6 0 0 0;1/8 0 3/8 0 0; 1/2 0 -3/2 2 0];
  disp('Merson method selected');
end;
steps=(tspan(2)-tspan(1))/step+1;
y=startval; t=tspan(1);
yvals=startval; tvals=tspan(1);
for j=2:steps
  k(1)=step*feval(f,t,y);
  for i=2:order
   k(i)=step*feval(f,t+step*d(i),y+c(i,1:i-1)*k(1:i-1)');
  end;
  y1=y+b*k'; t1=t+step;
  %collect values together for output
  tvals=[tvals, t1]; yvals=[yvals, y1];
  t=t1; y=y1;
end;
```

Funkcja ta została wykorzystana przez skrypt ilustracyjny ***mat_rkgen.m*** , dający prawidłowy obrazek

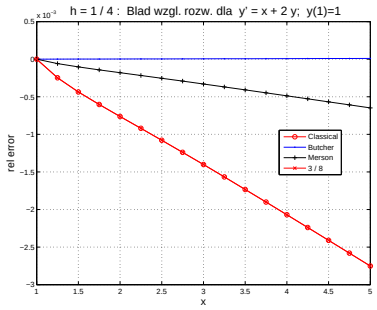
```
% rkgen function is Matlab's predefined function
% which implements Runge Kutta methods for solving
% first order differential equation
char=['o' '*' '+'];
for meth=1:3
  [t,x]=rkgen('f3',[0 3],1,.25,meth);
  re=(x-exp(-t))./exp(-t);
  plot(t,re,char(meth));
  axis([0 3 0 1.5e-4])
  xlabel('t');ylabel('relative error');
  hold on;
end;
hold off;
```



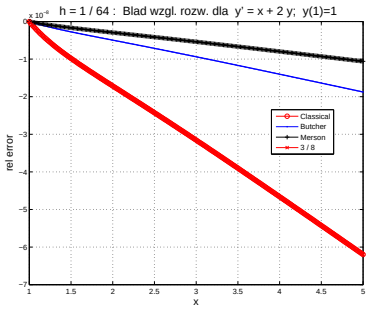
2 Ilustracja wyników zadania 6.3.5 – ostrzeżenie numeryczne

Współczynniki trzech metod znajdują się na stronie poprzedniej. Współczynniki metody czwartej **RK(3/8)** – na stronie następnej. Ilustracja przedstawia błędy rozwiązań numerycznych dla obu zadań:

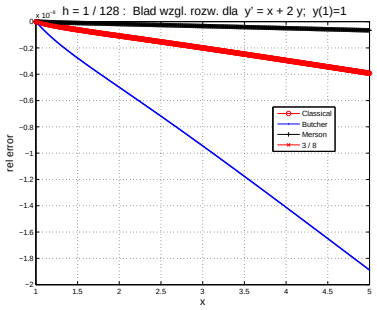
- zagadnienie początkowe $y' = x + 2y\,,\qquad y(1) = 1\,,\qquad$ rozwiązywane na odcinku $[1,\,5]\,,$



$\delta = 1e-3$

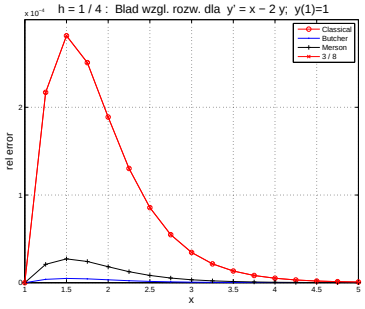


$\delta = 1e-8$

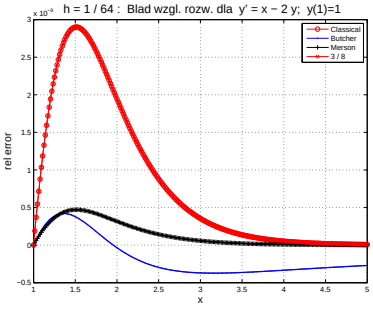


$\delta = 1e-8$

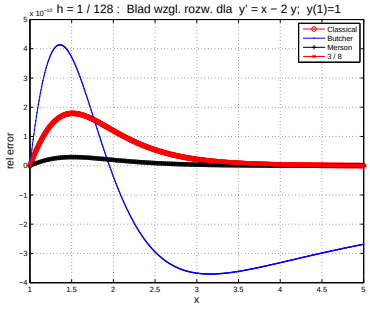
- zagadnienie początkowe $y' = x - 2y\,,\qquad y(1) = 1\,,\qquad$ rozwiązywane na odcinku $[1,\,5]\,,$



$\delta = 1e-4$

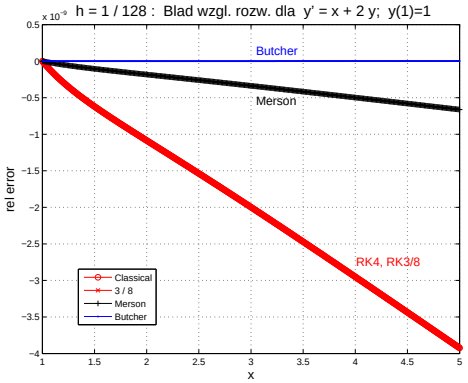


$\delta = 1e-9$

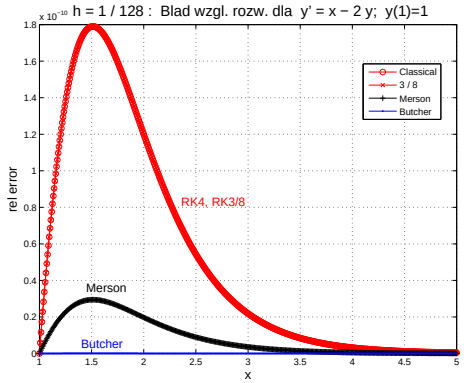


$\delta = 1e-10$

- **A teraz ostatnie ilustracje, otrzymane po poprawieniu współczynników metody Butchera**



$\delta = 1e-9$



$\delta = 1e-10$

- *oto dokładne współczynniki* ***metody Butchera*** *6-go(???) rzędu*, oraz **metody RK(3/8)** rzędu 4-go

0						
1/4	1/4					
1/4	1/8	1/8				
1/2	0	−1/2	1			
3/4	3/16	0	0	9/16		
1	−3/7	2/7	12/7	−12/7	8/7	
(5)	7/90	0	32/90	12/90	32/90	7/90
(4)	1/6	0	0	4/6	0	1/6

0				
1/3	1/3			
2/3	−1/3	1		
1	1	−1	1	
	1/8	3/8	3/8	1/8

- *Współczynniki* ***metody Butchera*** *5-go rzędu*, podane w książce
  John C. Butcher, *Numerical methods for ordinary differential equations*, 2nd Edition, Wiley, 2008.

0						
1/4	1/4					
1/4	1/8	1/8				
1/2	0	0	1/2			
3/4	3/16	−3/8	3/8	9/16		
1	−3/7	8/7	6/7	−12/7	8/7	
	7/90	0	32/90	12/90	32/90	7/90

3 Sterowanie wielkością kroku

3.1 Szacowanie błedu lokalnego metody jednokrokowej

Załóżmy, że jesteśmy w punkcie (x_n, y_n) dysponując ostatnio wyznaczoną wartością rozwiązania y_n (dla wygody niech to będzie liczba rzeczywista; ogólnie może to być wektor). Wykonajmy jeden krok długości h do punktu $x_{n+1} = x_n + h$ metodą rzędu p i oznaczmy otrzymaną wartość przez $\tilde{y}$:

$$\tilde{y} = y_n + h\Phi_f(h;, x_n, y_n)\,.$$

Dokładną wartość rozwiązania (odpowiadającą podanemu wyżej warunkowi początkowemu (x_n, y_n)), oznaczmy w nowym punkcie x_{n+1} przez y i już możemy napisać równość

$$y - \tilde{y} = \phi(x_n, y_n)h^{p+1} + O(h^{p+2})\,. \tag{1}$$

Wykonajmy teraz od punktu x_n dwa kroki długości $\frac{h}{2}$. Otrzymujemy kolejno

$$\hat{y}_{n+\frac{1}{2}} = y_n + \frac{h}{2}\Phi_f\left(\frac{h}{2};, x_n, y_n\right)\,.$$

$$\hat{y} = \hat{y}_{n+\frac{1}{2}} + \frac{h}{2}\Phi_f\left(\frac{h}{2};, x_{n+\frac{1}{2}}, \hat{y}_{n+\frac{1}{2}}\right)$$

i możemy napisać (pozostawiając największe wyrazy błędu), że

$$y - \hat{y} \simeq 2\phi(x_n, y_n)\left(\frac{h}{2}\right)^{p+1} + O\left(\left(\frac{h}{2}\right)^{p+2}\right)\,. \tag{2}$$

Odejmując teraz równość (2) od (1), otrzymujemy

$$\hat{y} - \tilde{y} \simeq \phi(x_n, y_n)h^{p+1}(1 - 2^{-p}) + O(h^{p+2})\,,$$

a dalej:

$$\phi(x_n, y_n)h^{p+1} \simeq \frac{\hat{y} - \tilde{y}}{1 - 2^{-p}} + O(h^{p+2})\,.$$

Dokładną wartość rozwiązania y możemy na podstawie (2) ekstrapolować wzorem

$$y \simeq \hat{y} + \frac{\hat{y} - \tilde{y}}{2^p - 1} + O(h^{p+2})\,, \tag{3}$$

lub na podstawie (1) - wzorem

$$y \simeq \tilde{y} + \frac{2^p}{2^p - 1}(\hat{y} - \tilde{y}) + O(h^{p+2})\,.$$

3.2 Akceptacja kroku

Poprawka we wzorze (3) stanowi największą część błędu metody (w sensie rozwinięcia w szereg potęgowy):

$$err(h) := \frac{\hat{y} - \tilde{y}}{2^p - 1}\,.$$

Jeżeli błąd ten jest dostatecznie mały, to możemy wykorzystać ostatnie wzory przyjmując

$$y_{n+1} = y$$

i pójść dalej. Jeżeli nie - obliczenia trzeba powtórzyć z mniejszym krokiem. Błąd może być mały w sensie jego wartości *bezwzględnej* lub *względnej* i warunek zaakceptowania kroku może mieć na przykład postać warunku

$$|err(h)| \leq \frac{|y_n| + |y_{n+1}|}{2}\epsilon_{rel} + \epsilon_{abs}\,. \tag{4}$$

Przyglądając się nierówności (4) widzimy, że

- w przypadku *małych* wartości y o zaakceptowaniu kroku decyduje występujące po prawej stronie ϵ_{abs},
- w przypadku *dużych* wartości y o zaakceptowaniu kroku decyduje ϵ_{rel}.

Uwaga: *Jeżeli rozwiązujemy układ równań różniczkowych* ($y \in R^m$), *to wartości bezwzględne, występujące w nierówności* (4) *należy zastąpić używanymi normami.*

3.3 Wybór nowej długości kroku h

Oglądając wzory na $err(h)$ widzimy, że po zmianie długości kroku h na αh otrzymamy

$$err((\alpha h) = \alpha^{p+1} err(h)\,.$$

Możemy więc, mając aktualnie policzoną wartość $err(h)$, tak dobrać nową długość kroku $h_{new} = \alpha h$, aby z dużym prawdopodobieństwem, od razu spełnić warunek akceptacji kroku

$$|err(h_{new})| \leq \frac{|y_n| + |y_{n+1}|}{2}\epsilon_{rel} + \epsilon_{abs}\,.$$

Wystarczy spełnić nierówność

$$|\alpha^{p+1} err(h)| \leq \frac{|y_n| + |y_{n+1}|}{2}\epsilon_{rel} + \epsilon_{abs}$$

wybierając na przykład

$$\alpha := 0.9 \sqrt[p+1]{\frac{\frac{|y_n|+|y_{n+1}|}{2}\epsilon_{rel} + \epsilon_{abs}}{|err(h)|}}\,. \tag{5}$$

Współczynnik 0.9 - czasem zwany współczynnikiem bezpieczeństwa - ma na celu uwzględnić ewentualny wzrost błędu w przypadku przemieszczenia się na sąsiedni odcinek zmiennej niezależnej po zaakceptowaniu kroku.

- **UWAGA: przy KSERO znajdują się jeszcze 4 strony z Hairera z metodami włożonymi...**

4 Zadania na ćwiczenia i na pracownię

1. Zbadać rząd metody Butchera 6-go***(???)*** rzędu; porównać ją numerycznie z metodą Butchera rzędu 5-go.

2. Propozycja:
   rozwiązywać zadaną metodą RR, którego rozwiązaniem jest wielomian, kolejno stopnia 1-go, 2-go,...
   Jak długo otrzymujemy "wynik dokładny"?

* * *