บทที่ 8
วิธีวิเคราะห์เชิงตัวเลขสำหรับการแก้สมการเชิงอนุพันธ์สามัญ (Ordinary Differential Equations)

บทที่ 8
วิธีวิเคราะห์เชิงตัวเลขสำหรับการแก้สมการเชิงอนุพันธ์สามัญ (Ordinary Differential Equations)

เราเรียกรูปความสัมพันธ์ที่อธิบายผลกระทบระหว่าง ตัวแปรและอัตราการเปลี่ยน-แปลง (นั่นคือ derivatives) ของตัวแปรว่า **สมการเชิงอนุพันธ์** (differential equations)

ในบทนี้จะศึกษาเฉพาะสมการเชิงอนุพันธ์สามัญ (Ordinary differential equations) ซึ่งประกอบด้วย ตัวแปรตามคือ y's ซึ่งขึ้นกับตัวแปรอิสระเพียงหนึ่งตัวคือ t

ปัญหาที่เรียกว่า **ปัญหาค่าเริ่มต้น** (Initial Value Problem (IVP)) คือ

$$dy/dx = f(t,y),\ y = y_o \text{ เมื่อ } t = t_o$$

วิธีที่จะศึกษา 2 กลุ่มใหญ่ ๆ ในบทนี้คือ

1) วิธีแบบขั้นเดียว (Single-Step method) เช่น
   1.1 วิธีของออยเลอร์ (Euler's method)
   1.2 วิธีของเทย์เลอร์ (Taylor's method)
   1.3 วิธีของรุงเง-คุตตา (Runge-Kutta methods) เช่น
       วิธีของออยเลอร์ที่ปรับปรุงแล้ว (Modified Euler method)
       และ วิธีของฮวน (Huen's method)
2) วิธีแบบหลายขั้น (Multistep method) เช่น
   วิธีตัวทำนาย-ตัวแก้อันดับที่สี่ของอาดามส์
   (Adams Fourth-Order Predictor-Corrector method (APC4))

8.1 การแก้ปัญหาซึ่งเรียกว่าปัญหาค่าเริ่มต้น (Initial Value Problem:IVP)

1st-order IVP:

$$dy/dx = f(t,y), \quad y = y_o \text{ เมื่อ } t = t_o \qquad \ldots(1)$$

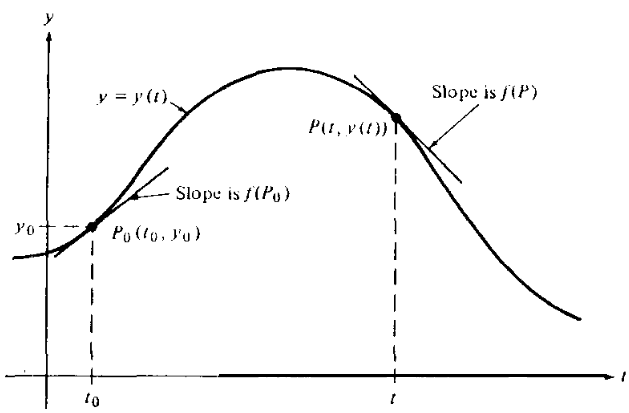
8.1A ผลเฉลยที่หาได้และการมีรูปเดียวของผลเฉลย

ผลเฉลยหนึ่งของ IVP คือฟังก์ชัน $y(t)$ ซึ่งสอดคล้องกับ (2)

$$d[y(t)]/dt = f[t,y(t)] \text{ และ } y(t_o) = y_o \qquad \ldots(2)$$

หรือกล่าวเป็นข้อความว่า

$y(t)$ คือ ผลเฉลยของ IVP ถ้าฟังก์ชันของ t [ซึ่งได้จากการแทน y ด้วย $y(t)$ ใน $f(t,y)$] คือ derivative ของ $y(t)$ และ $y = y(t)$ นั้นสอดคล้องกับ initial condition ที่ว่า $y = y_o$ เมื่อ $t = t_o$



รูป 8.1-1 กราฟของผลเฉลยของ (IVP): $y' = f(t,y), y(t_o) = y_o$

จากรูป กราฟของผลเฉลย สอดคล้องกับ

1) slope ที่จุด (t,y) คือ $f(t,y)$

2) เส้นกราฟผ่านจุด $P_o(t_o,y_o)$ ใน ty-plane

ตัวอย่าง สำหรับ IVP ต่อไปนี้ จงแสดงว่ามี $y(t)$ เป็นผลเฉลย และมีเพียงรูปเดียว

$$\text{IVP: } dy/dt = -ty^2 , \; y = 1 \text{ เมื่อ } t = 2 \qquad \text{...(3)}$$

Solution จาก IVP: $dy/dt = -ty^2$

จากการแยกตัวแปรและการอินทิเกรท จะเห็นว่าผลเฉลยใด ๆ ของ $y = y(t)$ ต้องสอดคล้องกับ (4)

$$-y^{-2}\, dy = t\, dt$$

$$-\int y^{-2}\, dy = \int t\, dt$$

$$1/y = (t^2/2) \text{ t } c$$

$$y = 2/(t^2 \text{ t } 2C) \qquad \text{...(4)}$$

เมื่อ C เป็นค่าคงที่ใด ๆ อย่างไรก็ดีเพื่อให้สอดคล้องกับ initial condition $y(2) = 1$

จาก (4) $y(t) = 2/(t^2 \text{ t } 2C)$

$$y(2) = 2/(4 \text{ t } 2C) = 1$$

$$c = -1$$

c ต้องมีค่าเท่ากับ -1

ดังนั้น $y(t) = 2/(t^2 - 2)$ ...(5)

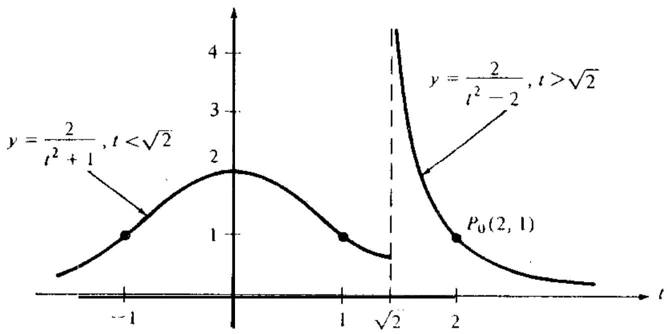
คือ ผลเฉลยที่เป็นได้เพียงผลเฉลยเดียว สำหรับ $t > 2$

$y(t)$ ใน (5) เป็นผลเฉลยของ IVP ใน (3) เพราะว่า

$$d[y(t)]/dt = -4t/(t^2 - 2)^2$$
$$= -t[2/(t^2 - 2)]^2$$
$$= -t[y(t)]^2$$
$$= f[t,y(t)]$$

และ $y(2) = 2/(4 - 2) = 1$

อย่างไรก็ดี initial condition ที่ $t = 2$ ไม่ใช่ทำให้มีผลเฉลยเดียว เมื่อหาผลเฉลยทางซ้ายของ $t = \sqrt{2}$ จริง ๆ แล้ว $y(t)$ อาจขยายไปทางซ้ายของ $t = \sqrt{2}$ โดยใช้ any value ของ C ใน (4) สำหรับ $t < \sqrt{2}$



รูป 8.1-2 การไม่เป็นเอกลักษณ์ของผลเฉลยเนื่องมาจาก singularity ของ y(t)

8.1B วิธีของออยเลอร์ (Euler's Method)

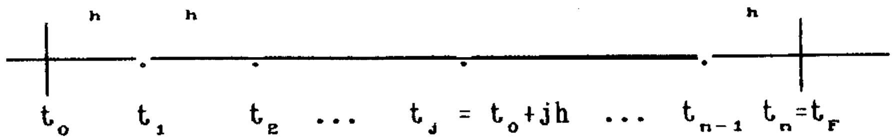
ให้ $[t_o, t_F]$ เป็น interval ซึ่งเราจะหาผลเฉลย $y(t)$ ของ

$$\text{(IVP)} \quad dy/dt = f(t,y), \quad y(t_o) = y_o$$

แบ่งช่วง $[t_o, t_F]$ เป็น n ส่วนย่อย โดยการใช้ stepsize

$$h = (t_F - t_o)/n \qquad \ldots(6)$$

เราได้จุด $t_1, \ldots t_{n-1}$ ดังแสดงในรูปต่อไปนี้



...(7)

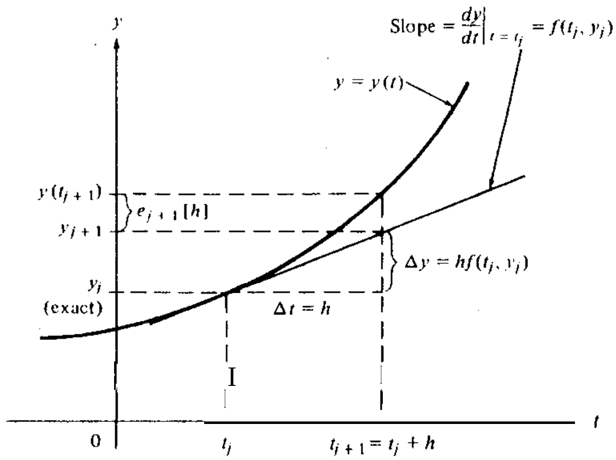
วิธีวิเคราะห์เชิงตัวเลขสำหรับ solve (IVP) จะเริ่มที่ $y_o = y(t_o)$ และสร้างค่า $y_1, \ldots, y_n$ ซึ่ง

y_j ประมาณค่าที่แท้จริง $y(t_j)$, $j = 1, \ldots, n$ ...(8)

สำหรับ h เล็ก ๆ กราฟของผลเฉลย $y = y(t)$ อาจถูกประมาณค่าในช่วง $[t_j, t_{j+1}]$ โดย tangent line ณ จุด $[t_j, y(t_j)]$ ดูรูป 8.1-3 recursive formula จากมาตรการนี้คือ

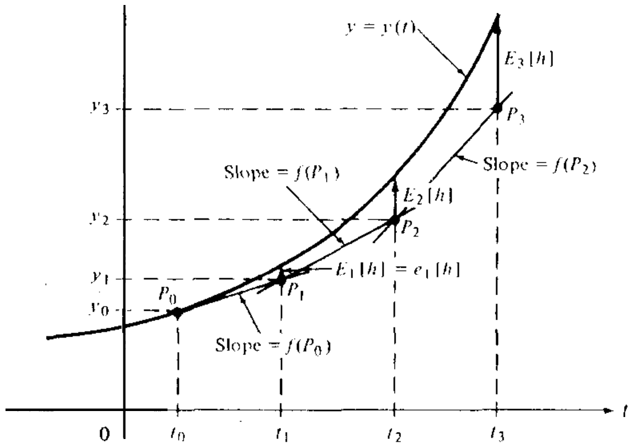
วิธีของออยเลอร์ : $y_{j+1} = \quad y_j \; h\, f(t_j, y_j)$, $j = 0, \ldots, n-1$

...(9)



รูป 8.1-3 การประมาณค่า $y(t_{j+1})$ โดย $y_j + h\, f(t_j, y_j)$ เมื่อ $y_j = y(t_j)$

วิธีวิเคราะห์เชิงตัวเลขที่ง่ายที่สุดสำหรับหาผลเฉลยของ (IVP) ดังแสดงในรูป 8.1-4



รูป 8.1-4 วิธีของออยเลอร์ แสดง $E_1[h]$, $E_2[h]$, $E_3[h]$

$E_{j+1}(h) = y(t_{j+1}) - y_{j+1}$ = ความคลาดเคลื่อนสะสมเนื่องจากการตัดปลาย ที่ t_{j+1}	...(10)

เราอาจเรียก (10) ว่า Global truncation error ที่ t_{j+1} ซึ่งเป็นผลมาจากการรวมอิทธิพลของความคลาดเคลื่อนของ tangent line ($e_{j+1}(h)$ ในรูป 8.1-3) และความจริงที่ว่า $j >= 1$ slope $f(t_j,y_j)$ ซึ่งถูกใช้ใน (9) นั้นโดยทั่ว ๆ ไป ไม่ใช่ exact slope $f(t_j,y(t_j))$

ตัวอย่าง จงใช้วิธีของออยเลอร์ บน [2,3] สำหรับ IVP

$$dy/dt = -ty^2 \ , \ y(2) = 1$$

เริ่มด้วย h = 0.1 แล้วใช้ h = 0.05 จงอภิปรายเกี่ยวกับความแม่นยำที่ได้ด้วย

Solution เนื่องจากว่า $f_j = -t_j y_j^2$ สูตรของออยเลอร์ (9) สำหรับ IVP นี้คือ

$$y_{j+1} = y_j + h[-t_j y_j^2] \approx y(t_{j+1}), \ t_{j+1} = 2+(j+1)h \qquad ...(11)$$

เริ่มจาก $t_o = 2$ และ $y_o = 1$ และใช้ h = 0.1 เราได้ 4s ดังนี้

$j=0$: $y_1=y_0 - h[t_0 y_0^2] = 1-0.1[(2)(1^2)] = 0.6000 \approx y(2.1)$

$j=1$: $y_2=y_1 - h[t_1 y_1^2]=0.8000-0.1[(2.1)(0.8000)^2] = 0.6656 \approx y(2.2)$

$j=2$: $y_3=y_2 - h[t_2 y_2^2]=0.6656-0.1[(2.2)(0.6656)^2] \doteq 0.5681 \approx y(2.3)$

$j=3$: $y_4=y_3 - h[t_3 y_3^2]=0.5681-0.1[(2.3)(0.5681)^2] \doteq 0.4939 \approx y(2.4)$

ทำนองเดียวกันใช้ h = 0.05 ใน (11) ให้ 4s

$j = 0$: $y_1 = 1 - 0.05[(2)(1^2)] = 0.9000 \approx y(2.05)$

$j = 1$: $y_2 = 0.9000 - 0.05[(2.05)(0.9000)^2] = 0.8170 \approx y(2.1)$

$j = 2$: $y_3 = 0.8170 - 0.05[(2.10)(0.8170)^2] \doteq 0.7469 \approx y(2.15)$

$j = 3$: $y_4 = 0.7469 - 0.05[(2.15)(0.7469)^2] \doteq 0.6869 \approx y(2.2)$

ผลเฉลยสำหรับ $t_j = 2, 2.1, 2.2, \ldots, 3$ เมื่อ h = 0.1, 0.05 แสดงในตาราง 8.1-1

Exact $y(t_j)$ values ได้มาจากผลเฉลย $y(t)$ ใน (5) นั่นคือ

$$y(t_j) = 2/(t_j^2 - 2)$$

ทำการคำนวณ โดยใช้ 13s arithmetic ดังนั้นความคลาดเคลื่อนที่เกิดขึ้นนั้นทั้งหมดเกิดจากความคลาดเคลื่อนสะสมเนื่องจากการตัดปลาย

ตาราง 8.1-1

วิธีของออยเลอร์สำหรับ $y' = -ty^2$, $y(2) = 1$

	Exact	Using $h = 0.1$		Using $h = 0.05$		Richardson ($n = 1$)
t_j	$y(t_j)$	$y_j[0.1]$	$E_j[0.1]$	$y_j[0.05]$	$E_j[0.05]$	$y_j[0.05]_{\text{improved}}$
$t_0 = 2.0$	I	1	0	1	0	1
2.1	**0.8299**	0.8000	0.0299	0.8 170	0.0129	0.8340
2.2	0.7042	0.6656	0.0386	0.6869	0.0173	0.7082
2.3	0.6019	0.5681	0.0398	0.5879	0.0182	0.6133
2.4	0.5319	0.4939	0.0380	0.5142	0.0177	0.5345
2.5	0.4706	0.4354	0.0352	0.4539	0.0167	0.4742
2.6	0.4202	0.3880	0.0322	0.4048	0.0154	0.4216
2.7	0.3781	0.3488	0.0292	0.3640	0.0141	0.3792
2.8	0.3425	0.3160	0.0265	0.3291	0.0128	0.3434
2.9	0.3126	0.2880	0.0240	0.3003	0.0117	0.3126
$t_F = 3.0$	0.2857	0.2640	0.0211	0.2151	0.0106	0.2862

จากคอลัมน์ E(0.1) และ E(0.05) ในตารางข้างบนแสดงว่า การแบ่งครึ่ง h นั้นจะแบ่งครึ่ง E(h) โดยประมาณด้วย

เราอาจใช้สูตรของริชาร์ดสัน ได้ดังนี้

จาก $F(h) = y_j(h)$, $h = 0.05$, $n = 1$ และ $r = 0.1/0.05 = 2$

ได้ค่าประมาณที่ปรับปรุงแล้วคือ

$$y_j(0.05)_{improved} = [2y_j(0.05)-y_j(0.1)]/(2-1)$$
$$= 2y_j(0.05) - y_j(0.1)$$

ค่าที่คำนวณได้ถูกแสดงไว้ในคอลัมน์ขวาสุดของตารางข้างบน ซึ่งจะเห็นว่าจะแม่นยำกว่า $y_j(0.05)$ อย่างน้อยทศนิยม 1 ตำแหน่ง

8.2 Self-Starting Methods: เทย์เลอร์ (Taylor) และ รุงเง-คุตตา (Runge-Kutta)

เราอาจได้วิธีของออยเลอร์จาก $O(h^2)$ Taylor approximation

$$y(t_{j+1}) = y(t_j+h) \approx y(t_j) + h\, y'(t_j) \qquad ...(1)$$

โดยการแทนค่าที่แท้จริงของ $y(t_j)$ และ $y'(t_j)$ ซึ่งไม่ทราบค่า โดยค่าประมาณที่มีอยู่

$$y(t_j) \approx y_j \text{ และ } y'(t_j) \approx f_j = f(t_j,y_j) \qquad ...(2)$$

สูตรซึ่งได้สำหรับวิธีอันดับที่หนึ่ง (1st-order method) อาจเขียนได้ดังนี้

$$y_{j+1} = y_j + h\, \Phi_{T,1} \text{ โดยที่ } \Phi_{T,1} = f_j \qquad ...(3)$$

8.2A วิธีของเทย์เลอร์ (Taylor's Method)

เพื่อขยาย (3) ไปเป็นวิธีอันดับที่ n (nth-order method) จะเริ่มด้วย $O(h^{n+1})$ Taylor Approximation

$$y(t_{j+1}) \approx y(t_j) + h\,y'(t_j) + (h^2/2!)\,y''(t_j) + \dots + (h^n/n!)\,y^{(n)}(t_j) \qquad \dots(4)$$

และแทน $y''(t_j),\dots,y^{(n)}(t_j)$ โดยค่าประมาณที่คำนวณได้คือ $y_j'',\dots,y_j^{(n)}$ ผลที่ได้คือ **สูตรของเทย์เลอร์อันดับที่ n** (nth-order Taylor's method)

$$\boxed{\begin{array}{l} y_{j+1} = y_j + h\,\Phi_{T,n} \\ \text{โดยที่ } \Phi_{T,n} = f_j + (h/2!)y_j'' + \dots + (h^{n-1}/n!)y_j^{(n)} \end{array}} \qquad ..(5)$$

เราใช้ Chain Rule สำหรับฟังก์ชันของตัวแปร 2 ตัว เพื่อการหาอนุพันธ์

$$y' = f(t,y) \text{ โดยที่ } y = y(t) \qquad \dots(6)$$

เทียบกับ t

$$y'' = df[t,y(t)]/dt = f_t + f_y\,y' = f_t + f_y\,f \qquad \dots(7)$$

โดยที่ $f_t = \partial f(t,y)/\partial t$

$f_y = \partial f(t,y)/\partial y$

และ $f = f(t,y)$

ในทำนองเดียวกัน

$$y''' = d(f_t)/dt + d(f_y\ y')/dt = (f_{tt} + f_{ty}y') + [f_y\ y'' + (f_{yt} + f_{yy}y')y']$$

เพราะว่า $f_{ty} = f_{yt}$ ดังนั้น

$$y''' = f_{tt} + 2f_{ty}y' + f_{yy}(y')^2 + f_y y'' \qquad \ldots(8a)$$

เขียน (8a) ใหม่ได้

$$y''' = f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_y[f_t + f_y f] \qquad \ldots(8b)$$

ดังนั้นเราอาจเขียน $y^{(k)}$ ในเทอมของ f และ partials ของ f นิพจน์เหล่านี้จะถูกนำไปใช้ในการประมาณ $y^{(k)}(t_j)$ ใน (5) ตัวอย่างเช่นจาก (7) $y''(t_j)$ ถูกประมาณโดย

$$y_j'' = [f_t + f_y f]_j = f_t(t_j, y_j) + f_y(t_j, y_j)f(t_j, y_j) \qquad \ldots(9)$$

แทนลงใน (5) โดยใช้ n = 2 จะทำให้ได้ **สูตรของเทย์เลอร์อันดับที่สอง (second-order Taylor's method)** ดังนี้

$$\boxed{y_{j+1} = y_j + h\,\Phi_{T,2} \text{ โดยที่ } \Phi_{T,2} = f_j + (h/2)[f_t + f_y f]_j} \qquad \ldots(10)$$

ในทำนองเดียวกันเราจะได้ สูตรของเทย์เลอร์อันดับที่สาม
(third-order Taylor's method)

$$\boxed{y_{j+1} = y_j + h\,\phi_{T,3} \text{ โดยที่ } \phi_{T,3} = \phi_{T,2} + (h^2/6)\,y_j'''} \qquad \ldots(11)$$

ทั้งนี้ y_j''' จะหาได้โดยการคำนวณ (8a) หรือ (8b) ที่ (t_j, y_j)

__ตัวอย่าง__ จงใช้ สูตรของเทย์เลอร์อันดับที่สอง บน [2,3] สำหรับ IVP

$$dy/dt = -ty^2\ ,\ y(2) = 1 \qquad \ldots(12)$$

ใช้ $h = 0.1$ แล้ว $h = 0.05$ เหมือนกับที่ใช้ในตัวอย่างในหัวข้อ 8.1B

__Solution__ สำหรับ $f(t,y) = -ty^2$

$$f_t = -y^2 \text{ และ } f_y = -2ty$$

ดังนั้น สูตรของเทย์เลอร์อันดับที่สองใน (10) กลายเป็น

$$\begin{aligned} y_{j+1} &= y_j + h\{-t_j y_j^2 + (h/2)[-y_j^2 + (-2t_j y_j)(t_j y_j^2)]\} \\ &= y_j + hy_j^2\{-t_j + (h/2)[-1+2t_j^2 y_j]\} \end{aligned}$$

ใช้ $h = 0.1$ และเริ่มต้นด้วย $t_0 = 2,\ y_0 = 1$ จะให้ (to 5s)

$j = 0$: $y(t_1) = y(2.1) \approx y_1 = y_0 + hy_0^2\{-t_0 + (h/2)[-1+2t_0^2 y_0]\}$

$y(t_1) \approx 1 + 0.1(1)^2\{-2+0.05[-1+2(2)^2(1)]\} \doteq 0.635$

$j = 1$: $y(t_2) = y(2.2) \approx y_2 = y_1 + hy_1^2\{-t_1 + (h/2)[-1+2t_1^2 y_1]\}$

$y(t_2) \approx 0.635 + 0.1(0.835)^2\{-2.1+0.05[-1+2(2.1)^2(0.835)]\}$

$\doteq 0.71077$

ทำนองเดียวกันใช้ $h = 0.05$ และเริ่มต้นด้วย $t_o = 2$, $y_o = 1$ จะให้ (to 5s)

$$j = 0:\ y(t_1) = y(2.05) \approx y_1 = 1+0.05(1)^2\{-2+0.025[-1+2(2)^2(1)]\}$$
$$= 0.90875$$
$$j = 1:\ y(t_2)=y(2.1) \approx y_2 = y_1+0.05y_1^2\{-2.05+0.025[-1+2(2.05)^2 y_1]\}$$
$$\doteq 0.83096$$ และเรื่อย ๆ ไป

ผลของการประมาณ $y(2.0)$, $y(2.1)$,..., $y(3.0)$ ถูกแสดงไว้ในตาราง 8.2-1 เนื่องจากว่าเราใช้วิธีอันดับที่สอง (second-order method) ค่าที่ถูกปรับปรุงหาได้โดยใช้ $n = 2$ ในสูตรของริชาร์ดสัน นั่นคือ

$$y_j[0.05]_{improved} = \{2^2\ y_j[0.05] - y_j[0.1]\}/(2^2-1)$$
$$= [4\ y_j[0.05] - y_j[0.1]\}/3 \qquad \ldots(13)$$

ค่าที่ถูกปรับปรุงแล้วแม่นยำถึงเกือบ 4d

การเปรียบเทียบค่าจาก สูตรของเทย์เลอร์อันดับที่สอง ในตาราง 8.2-1 กับค่าจาก สูตรของออยเลอร์อันดับที่หนึ่ง ในตาราง 8.1-1 จะเห็นได้ว่าค่าจากอันดับที่สองนั้นมีความแม่นยำมากกว่า สำหรับ h ที่กำหนดให้ และแสดงการปรับปรุงที่ดีขึ้นเมื่อ h มีค่าครึ่งหนึ่งของค่าเดิม ($E_j[0.05] \approx (1/4)E_j[0.1]$)

ตาราง 8.2-1

ค่าประมาณเมื่อใช้สูตรของเทย์เลอร์อันดับที่สองสำหรับ

$y' = -ty^2, \; y(2) = 1$

t_j	Exact $y(t_j)$	Using h = 0.1 $y_j[0.1]$	Using h = 0.1 $E_j[0.1]$	Using h = 0.05 $y_j[0.05]$	Using h = 0.05 $E_j[0.05]$	Richardson $y_j[0.05]_{\text{Improved}}$
t_0 = 2.0	1	1	0	1	0	1
2.1	0.8299	0.835	-0.0051	0.8310	-0.0011	0.8297
2.2	0.7042	0.7108	-0.0065	0.7056	-0.0014	0.7039
2.3	0.6079	0.6145	-0.0066	0.6093	-0.0014	0.6076
2.4	0.5319	0.5380	-0.0061	0.5332	-0.0013	0.5316
2.5	0.4706	0.4761	-0.0055	0.4718	-0.0012	0.4704
2.6	0.4202	0.4250	-0.0049	0.4212	-0.0010	0.4199
2.7	0.3781	0.3823	-0.0043	0.3790	-0.0009	0.3779
2.8	0.3425	0.3462	-0.0037	0.3433	-0.0008	0.3423
2.9	0.3120	0.3153	-0.0033	0.3127	-0.0007	0.3118
t_F = 3.0	0.2857	0.2886	-0.0029	0.2862	-0.0006	0.2855

ปัญหาในการเขียนโปรแกรมเพื่อใช้วิธีของเทย์เลอร์ คือผู้เขียนจะต้องหา (อย่างถูกต้อง) และเขียนคำสั่งคำนวณ partial derivative ของ f(t,y) สำหรับความเห็นของผู้ใช้แล้ว วิธีที่จะถูกใจมากกว่าคือวิธีซึ่งต้องเขียนคำสั่งเพื่อหาเฉพาะ f(t,y)เอง เท่านั้น วิธีของรุงเง-คุตตา ซึ่งจะกล่าวในหัวข้อต่อไปจะทำตามที่ต้องการ

8.2B วิธีรุงเง-คุตตาอันดับสอง (Second-Order Runge-Kutta Methods)

สูตรสำหรับวิธีของเทย์เลอร์อันดับสอง (Second-order Taylor method) คือ

$$y_{j+1} = y_j + h\,\Phi_{T,2} \quad \text{โดยที่} \quad \Phi_{T,2} = f_j + (h/2)[f_t(t_j,y_j)+f_j f_y(t_j,y_j)] \qquad \ldots(14)$$

เมื่อปลายศตวรรษที่ 19 นักคณิตศาสตร์ชาวเยอรมันชื่อรุงเง (Runge) ได้สังเกตว่านิพจน์ใน(14) สำหรับ $\Phi_{T,2}$ นั้นคล้ายกับ $O(h^2)$ Taylor approximation ซึ่งมีรูป

$$f(t_j+ph, y_j+qhf_j) \approx f_j + phf_t(t_j,y_j) + qhf_j f_y(t_j,y_j) \qquad \ldots(15)$$

เปรียบเทียบ (14) กับ (15) โดยใช้ $p = q = 1/2$ พบว่า

$$\Phi_{t,2} \approx f[t_j + (h/2), y + (h f_j/2)] \text{ ซึ่งมี } O(h^2) \text{ error} \qquad \ldots(16)$$

แทน (16) ใน (14) จะได้สูตรสำหรับ

วิธีของออยเลอร์ที่ปรับปรุงแล้ว : $y_{j+1} = y_j + h\, f[t_j+(h/2), y_j+(h\, f_j/2)]$

...(17)

เนื่องจาก (16) คือ $O(h^2)$ y_{j+1} ใน (17) ต่างไปจากใน (14) เท่ากับ $hO(h^2) = O(h^3)$ แต่ $e_{j+1}[h]$ สำหรับ (14) คือ $O(h^3)$ สิ่งที่ตามมาคือ ความคลาดเคลื่อนจากการตัดปลายต่อขั้นของวิธีของออยเลอร์ที่ปรับปรุงแล้วเป็น $O(h^3)$ ด้วย นั่นคือวิธีของออยเลอร์ที่ปรับปรุงแล้วเป็นวิธีอันดับที่สอง สูตรนี้ต้องใช้ $f(t,y)$ ที่ (t_j,y_j) แล้วที่ $[t_j+(h/2),y_j+(hf_j/2)]$ ในแต่ละขั้น

ใน (17) ซึ่งใช้ sample slope $f[t_j+(h/2),y_j+(hf_j/2)]$ แทนที่ $\Phi_{T,2}$ ยังมีวิธีอันดับที่สองอื่นซึ่งประมาณ $\Phi_{T,2}$ ด้วย weighted sum ของ sample slopes นั่นคือ

$$\Phi_{T,2} \approx a_1 f(t_j,y_j)+a_2 f(t_j+ph,y_j+qhf_j) \qquad \ldots(18)$$

โดยที่ weights a_1 และ a_2 และ scale factors p และ q จะถูกกำหนดเพื่อทำให้การประมาณนี้เป็น $O(h^2)$ เราเรียกขั้นตอนวิธีซึ่งใช้วิธีการนี้ว่า
วิธีของรุงเง-คุตตา (Runge-Kutta Methods)

แทนค่า $f(t_j+ph,y_j+qhf_j)$ จาก (15) ลงใน (18) แล้วเทียบกับส.ป.ส.ของ $f_t(t_j,y_j)$ และ $f_y(t_j,y_j)$ ใน (10) จะได้ว่า

$$a_1 + a_2 = 1 \text{ และ } a_2 p = a_2 q = 1/2$$

เราจะเลือก a_2 ให้มีค่าใด ๆ ก็ได้ สำหรับ

$$a_1 = 1 - a_2 \text{ และ } p = q = 1/(2a_2)\ (a_2 \neq 0) \qquad \ldots(19)$$

ใช้ $a_2 = 1$ จะทำให้ได้วิธีของออยเลอร์ที่ปรับปรุงแล้ว (17)

ใช้ $a_2 = 1/2$ (ดังนั้น $a_1 = 1/2,\ p = q = 1$) จะทำให้ได้สูตรสำหรับ

$$\boxed{\text{วิธีของฮวน :} \quad y_{j+1} = y_j+(h/2)[f_j+f(t_j+h,y_j+hf_j)]} \qquad \ldots(20)$$

<u>**ตัวอย่าง**</u> จงหา y_1 และ y_2 โดยใช้ $h = 0.1$ สำหรับ

(IVP) $y' = -ty^2$, $y(2) = 1$

โดยใช้ (a) วิธีของออยเลอร์ที่ปรับปรุงแล้ว และ

(b) วิธีของฮวน

<u>Solution</u>

(a) สำหรับ $f(t,y) = -ty^2$

จาก (17) $y_{j+1} \doteq y_j-0.1(t_j+0.05)[y_j+0.05f_j]^2$ ...(21)

โดยที่ $f_j = -t_jy_j^2$

$j = 0$: $t_0 = 2$ และ $y_0 = 1$ ดังนั้น $f_0 = -2(1)^2 = -2$

จาก (21) ดังนั้น $y_1 = 1-0.1(2+0.05)[1+0.05(-2)]^2$

$= 0.63395\ (E_1[0.1] \doteq -0.0041)$

$j = 1$: $t_1 = 2.1$ และ $y_1 \doteq 0.83395$ ดังนั้น $f_1 = -t_1y_1^2 \doteq -1.46049$

ดังนั้น $y_2 = 0.83395-0.1(2.1+0.05)[0.83395+0.05(-1.46049)]^2$

$\doteq 0.70946\ (E_2[0.1] \doteq -0.0053)$

(b) จาก (20) $y_{j+1} = y_j+0.05[f_j-(t_j+0.1)(y_j+0.1f_j)^2]$ ...(22)

โดยที่ $f_j = -t_jy_j^2$

$j = 0$: $t_0 = 2$, y_0 1 และ $f_0 = -2$ จาก (22)

$$y_1 = 1+0.05[(-2)-(2+0.1)(1+0.1(-2))^2]$$

$$\doteq 0.83280 \quad (E_1[0.1] \doteq -0.0029)$$

$j = 1$: $t_1 = 2.1$ และ $y_1 = 0.83280$ ดังนั้น $f_1 = -t_1 y_1^2 \doteq -1.45647$ และ

$$y_2 = 0.83280+0.05\{(-1.45647)-(2.1+0.1)[0.83280+0.1(-1.45647)]^2\}$$

$$\doteq 0.70804 \quad (E_2[0.1] \doteq -0.0042)$$

จากการเปรียบเทียบค่าเหล่านี้กับค่าที่ได้จากตาราง 8.2-1 พบว่าสูตรของรุงเง-คุตตา (17) และ (20) นั้นให้ความแม่นยำเท่าเทียมกับวิธีของเทย์เลอร์อันดับสอง และยังไม่ต้องใช้ partial derivatives อีกด้วย

8.2C วิธีรุงเง-คุตตาอันดับสูง (Higher-Order Runge-Kutta Methods)

ในการหาสูตรของวิธีรุงเง-คุตตาอันดับสี่ จะเริ่มที่สูตรของวิธีของเทย์เลอร์อันดับสี่

$$Y_{j+1} = y_j + h\Phi_{T,4} \text{ โดยที่ } \Phi_{T,4} = f_j + (h/2)y_j'' + (h^2/6)y_j''' + (h^3/24)y_j^{(iv)} \qquad \ldots(23)$$

แล้วประมาณค่า $\Phi_{T,4}$ โดย weighted sum

$$w_1 m_1 + w_2 m_2 + w_3 m_3 + w_4 m_4 \qquad \ldots(24a)$$

โดยที่ sample slopes m_1, m_2, m_3 และ m_4 คือ

$$m_1 = f(t_j, y_j) = f_j$$

$$m_2 = f(t_j+p_2h, y_j+h[q_{21}m_1])$$

$$m_3 = f(t_j+p_3h, y_j+h[q_{31}m_1+q_{32}m_2])$$

$$m_4 = f(t_j+h, y_j+h[q_{41}m_1+q_{42}m_2+q_{43}m_3]) \qquad \ldots(24b)$$

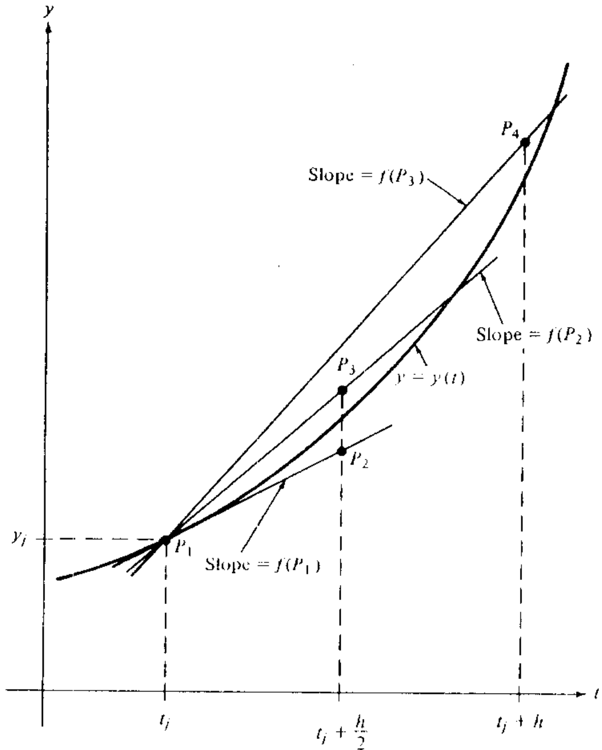
weights w ใน (24a) และ scale factors p และ q ใน (24b) จะหาได้จากการแทนนิพจน์สำหรับ m_2, m_3 และ m_4 ใน (24b) ด้วย $O(h^4)$ Taylor approximation ของ m ทั้ง 3 ตัว แล้วเทียบส.ป.ส.ของ partials ของ f ที่ (t_j,y_j) ใน (24a) กับส.ป.ส.ใน (23)

สูตรของรุงเง-คุตตาอันดับ 4 (Fourth-Order **Runge-Kutta** Formula **(RK4)**) คือ

$$y_{j+1} = y_j + (h/6)\{m_1+2(m_2+m_3)+m_4\} \qquad \text{...(25a)}$$

โดยที่
$$
\begin{aligned}
m_1 &= f(t_j,y_j)\\
m_2 &= f(t_j+h/2,\, y_j+hm_1/2)\\
m_3 &= f(t_j+h/2,\, y_j+hm_2/2)\\
m_4 &= f(t_j+h,\, y_j+hm_3)
\end{aligned}
\qquad \text{...(25b)}
$$

Sample slopes $m_1,\ldots,m_4$ คือค่าของ f ที่ sample points $P_1,\ldots,P_4$ ดังแสดงในรูป 6.2-1 ถ้า $f(t,y)$ ขึ้นอยู่กับ t เท่านั้น นั่นคือ $f(t,y) = g(t)$ แล้ว (25) จะเป็นการใช้กฎของซิมป์สันเพื่ออินทิเกรท $y'(t) = g(t)$ จาก t_j ถึง t_{j+1}



รูป 8.2-1 Sample points ที่ถูกใช้หา t_{j+1} โดย RK4 เมื่อ y_j เป็นค่าแม่นตรง

ตัวอย่าง* จงใช้ RK4 กำหนด h = 0.1 บน [2,3] สำหรับ IVP

$$y' = -ty^2\ ,\ y(2) = 1\quad [\text{ผลเฉลยแม่นตรง: } y(t) = 2/(t^2-2)]$$

Solution เริ่มต้นด้วย $t_o = 2$, $y_o = 1$ จาก (25) หา

$m_1 = f(2.0,1) = -2(1)^2 = -2$

$m_2 = f(2.05,1+0.05(-2)) = -(2.05)(0.9)^2 = -1.6605$

$m_3 = f(2.05,1+0.05(-1.6605)) = -(2.05)(0.916975)^2 \doteq -1.72373$

$m_4 = f(2.1,1+0.1(-1.72373)) = -(2.1)(0.82763)^2 \doteq -1.43843$

$y_1 = y_o -(0.1/6)[2+2(1.6605+1.72373)+1.43843] \doteq 0.829885$

ตาราง 8.2-2 แสดงค่าที่คำนวณจาก RK4 และค่าที่แท้จริงของ y(2.0), y(2.1),..., y(3.0) ซึ่งถูกปัดให้เหลือทศนิยม 6 ตำแหน่ง ค่าจาก RK4(y_j) ได้จาก โปรแกรมในภาษาฟอร์แทรนดังแสดงในรูป 8.2-2

ตาราง 8.2-2

ค่าคำนวณจาก RK4 สำหรับ $y' = -ty^2$, $y(2) = 1$ ($h = 0.1$)

t_j	y_j(RK4-value)	$y(t_j)$ = exact value
2.0	1.000000	1.000000
2.1	0.829885	0.829876
2.2	0.704237	0.704225
2.3	0.607914	0.607903
2.4	0.531924	0.531915
2.5	0.470596	0.470588
2.6	0.420175	0.420168
2.7	0.378078	0.378072
2.8	0.342471	0.342466
2.9	0.312017	0.312012
3.0	0.285718	0.285714

```
C     * * * * * * * * CALRK4 * * * * * * * * * * * * *
C     CALLING PROGRAM FOR 4TH ORDER RUNGE-KUTTA METHOD
      DOUBLE PRECISION YO
      EXTERNAL FTEST
      DATA IW, IR /5, 5/
C
      WRITE(IW,1)
    1 FORMATC'OINPUT (SEPARATED BY COMMAS) TO, TF, YO')
      READ(IR,*) TO, TF, YO
      WRITE(IW,2)
    2 FORMATC'OINPUT # STEPS, NPRINT')
      READ(IR,*) NSTEPS, NPRINT
      WRITE(IW, 3)
    3 FORMAT(6X,'T',12X,'Y')
C
      CALL RK4(FTEST, TO, TF, NSTEPS, NPRINT, YO, IW)
C
      STOP
      END

      FUNCTION FTEST(T,Y)
      DOUBLE PRECISION Y
         FTEST = -T*Y*Y
      RETURN
      END
```

```
      SUBROUTINE RK4(F, T, TF, NSTEPS, NPRINT, Y, IW)
      REAL Ml, M2, M3, M4
      DOUBLE PRECISION Y
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - C
C   THIS SUBROUTINE INTEGRATES FROM TO TO TF THE 1ST ORDER IVP  C
C       Y' = F(T,Y)    Y(TO) = YO (INITIAL T,Y)                  C
C   USING NSTEPS STEPS OF THE 4TH ORDER RUNGE-KUTTA METHOD.      C
C   IF NPRINT>O, IT PRINTS T AND Y EVERY NPRINT STEPS.           C
C   NOTE: F MUST BE DECLARED EXTERNAL IN THE CALLING PROGRAM     C
C - - - - - - - - - - - - - - - - - - - - - - VERSION 1: 5/1/81 C
      IF CNPRINT .GT. 0) WRITE (IW,1) T, Y
    1 FORMAT (F10.3, 3X, E14.7)
C
      H = (TF - T)/NSTEPS
      DO 10 J=1,NSTEPS
         Ml = F(T,Y)
         M2 = F(T + 0.5*H, Y + 0.5*H*M1)
         M3 = F(T + 0.5*H, Y + 0.5*H*M2)
         M4 = F(T + H, Y + H*M3)
         T = T + H
         Y = Y + H*(M1 + 2*(M2 + M3) + M4)/6
         IF CNPRINT .GT. 0 .AND. MOD(J,NPRINT) .EQ. 0)
     &      WRITE(IW,1) T, Y
   10 CONTINUE
      RETURN
      END
```

รูป 8.2-2 โปรแกรมภาษาฟอร์แทรนสำหรับการหาผลเฉลยของ

$y' = -ty^2$, $y(2) = 1$ โดย RK4

ในบรรดาวิธีอันดับสี่ด้วยกัน วิธีหนึ่งซึ่งมีประสิทธิภาพมากในหลาย ๆ วิธีคือ**วิธีรุงเง-คุตตา-เฟลเบิร์กอันดับสี่** (fourth-order **Runge-Kutta-Fehlberg method (RKF4)**) **RKF4 algorithm** แสดงในรูป 8.2-3

Algorithm: RKF4 (Fourth-Order Runge-Kutta-Fehlberg Method) *

Purpose: To solve to a prescribed accuracy on $[t_0, t_F]$ the IVP

$$y' = f(t, y), \qquad y = y_0 \text{ when } t = t_0$$

{*initialize*}
GET t_0, t_F, y_0, (parameters of IVP}
 $Rmax$, (accuracy control parameter]
 $ScaleMin$, $ScaleMax$ (stepsize control parameters)
$h \leftarrow (Rmax)^{1/4}$ [initial stepsize}
$Hmin \leftarrow h * 10^{-4}$ (minimum allowable stepsize)
$t \leftarrow t_0$; $y \leftarrow y_0$ {(t, y) is current (t_j, y_j)}

{*iterate*}
DO UNTIL **termination test** is satisfied
 BEGIN
 If $t + h > t_F$ THEN $h \leftarrow t_F - t$ {stepsize for final step}
 {estimate next $e[h]$}
 $k_1 \leftarrow hf(t_j, y_j)$
 $k_2 \leftarrow hf(t_j + \frac{1}{4}h, y_j + \frac{1}{4}k_1)$
 $k_3 \leftarrow hf(t_j + \frac{3}{8}h, y_j + \frac{3}{32}k_1 + \frac{9}{32}k_2)$
 $k_4 \leftarrow hf(t_j + \frac{12}{13}h, y_j + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3)$
 $k_5 \leftarrow hf(t_j + h, y_j + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4)$
 $k_6 \leftarrow hf(t_j + \frac{1}{2}h, y_j - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5)$
 $ErrorEstimate \leftarrow \frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{75,240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6$ {$\approx e[h]$}
 {accuracy test}
 $Ratio \leftarrow |ErrorEstimate|/h$
 IF $Ratio \leq Rmax$ THEN {accuracy of next y is acceptable}
  BEGIN
  $t \leftarrow t + h$ {$t = t_F$ for final step}
  $y \leftarrow y + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5$ {Now $y \approx y(t)$.}
  OUTPUT (t, h, y)
  END

 (set next h: $h * ScaleMin \leq$ next $h \leq h * ScaleMax$}
 $ScaleFactor \leftarrow 0.84 * (Rmax/Ratio)^{1/4}$
 If $ScaleFactor < ScaleMin$ THEN $ScaleFactor \leftarrow ScaleMin$
 If $ScaleFactor > ScaleMax$ THEN $ScaleFactor \leftarrow ScaleMax$
 $h \leftarrow ScaleFactor * h$
 (termination test: $t = t_F$ or $h < Hmin$}
 END

IF $t = t_F$ THEN OUTPUT (current y approximates $y(t_F)$ to the desired accuracy)
 ELSE OUTPUT ($h < Hmin$ occurred; apparent singularity near current t)

รูป 8.2-3 รหัสเทียมสำหรับขั้นตอนวิธี Runge-Kutta-Fehlberg (RKF4)

ค่าต่าง ๆ ที่ผู้ใช้โปรแกรมจะต้องกำหนดเองมักจะใช้ค่าต่อไปนี้

$$Rmax = 10^{-4} \text{ , ScaleMin} = 0.1 \text{ และ ScaleMax} = 4.0 \qquad \ldots(26)$$

รูป 8.2-4 แสดงผลของการแทน SUBROUTINE RK4 ของรูป 8.2-2 โดยซับรูทีน RKF4 ในรูป 8.2-5 ซึ่งจะหาค่าตาม วิธีรุงเง-คุตตา-เฟลเบิร์กอันดับสี่

โดยทั่ว ๆ ไปแล้ว RKF4 เป็นที่นิยมมากกว่า RK4 ทั้งนี้นอกเสียจากว่ามีความต้องการจะใช้ค่า h ที่คงที่ตลอดเวลา

J	T	H	Y
0	2.0000	0.1000000E+00	0.1000000E+01
1	2.1000	0.1000000E+00	0.8298735E+00
2	2.2115	0.1114898E+00	0.6918740E+00
3	2.3496	0.1381092E+00	0.5680786E+00
4	2.5204	0.1708108E+00	0.4595052E+00
5	2.7342	0.2137747E+00	0.3652411E+00
6	3.0050	0.2707763E+00	0.2844991E+00
7	3.3529	0.3478943E+00	0.2164084E+00
8	3.8076	0.4547751E+00	0.1600242E+00
9	4.0000	0.1923698E+00	0.1428565E+00

รูป 8.2-4 ค่าจากสูตร RKF4 สำหรับ $y' = -ty^2$, $y(2) = 1$ บน [2,4]

```
      SUBROUTINE RKF4(RMAX, SCAMIN, SCAMAX, NPRINT, IW)
      REAL K(10,6)
      DOUBLE PRECISION Y(10)
      COMMON N, T, Tf, Y
      COMMON /KCALC/ H, K
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - C
C  THIS SUBROUTINE USES THE 4TH ORDER RUNGE-KUTTA-FEHLBERG ALGORITHM  C
C  TO INTEGRATE N (N < 11) COUPLED FIRST ORDER IVPS, I.E.              C
C        Y' = F(T,Y) Y(TO) = YO    (TO,YO ARE INITIAL T,Y)             C
C  FROM TO TO TF, WHERE TF CAN BE TO THE LEFT OF TO.                   C
C  VALUES OF T, H, AND VECTOR Y ARE PRINTED EVERY NPRINT ITERATIONS.  C
C  IF NPRINT = 0, NOTHING IS PRINTED (UNLESS H BECOMES TOO SMALL).    C
C - - - - - - - - - - - - - - - - - - - - - - - VERSION 1: 5/1/81     C
C  INITIALIZE STEPSIZE, MINIMUM ALLOWABLE STEPSIZE, AND COUNTER:
      H = SIGN(RMAX**.25, TF-T)
      HMIN = 0.5E-4*H
      ITER = 0
C
      IF (NPRINT.GT.0) WRITE(IW,1) ITER, T, H, (Y(I),I=1,N)
    1 FORMAT(I6, F9.4, E15.7, E15.7, 9E15.7)
C
C  ITERATE:
C     **BEGIN LOOP BY SETTING H TO TF-T IF T+H PASSES TF
   10   IF ( H*(T+H-TF) .GE. 0.) H = TF - T
C
C     **PUT VECTOR KJ IN JTH COLUMN OF MATRIX K, J=1,..,6
        CALL SUMK(1, 0., 0., 0., 0., 0., 0.)
        CALL SUMK(2, .25, .25, il., 0., 0., 0.)
        CALL SUMK(3, .375, 3./32, 9./32. 0.. 0.. 0.)
        CALL SUMK(4, 12./13, 1932./2197. -7200./2197, 7296.12197.. 0.,0.)
        CALL SUMK(5, 1., 439.1216, -8., 3680./513, -845./4104, 0.)
        CALL SUMK(6, .5, -8.127, 2., -3544./2565, 1859./4104, -11./40)
C
C     **FORM ERREST = ESTIMATE OF ERROR OF NEXT Y(I) FOR I = 1,..,N
C     **AND FIND RATIO = THE LARGEST OF THE ERREST/H RATIOS
        RATIO = 0.
        DO 20 I=1,N
          ERREST = K(I,1)/360 - 128*K(I,3)/4275 - 2197*K(I,4)/75240
     &                 + K(I,5)/50 + 2*K(I,6)/55
          RATIO = AMAX1(RATIO,ABS(ERREST/H))
   20   CONTINUE
C
C     **TEST ACCURACY OF NEXT Y.  IF OK, UPDATE T, Y AND ITER
        IF (RATIO .GT. RMAX) GOTO 30
          T = T + H
          CALL SUMK(0, 0., 25./216, 0., 1408./2565, 2197./4104, -0.2)
          ITER = ITER + 1
C
          IF (NPRINT.GT.0 .AND. (MOD(ITER,NPRINT).EQ.0 .OR. T.EQ.TF))
     &      WRITE(IW,1) ITER, T, H, (Y(I), I=1,N)
C
C     **SET SCALE (BETWEEN SCAMIN AND SCAMAX) AND UPDATE H
   30   SCALE = 0.84*(RMAX/RATIO)**.25
        IF (SCALE .LT. SCAMIN) SCALE = SCAMIN
        IF (SCALE .GT. SCAMAX) SCALE = SCAMAX
        H = SCALE*H
C
C     **TERMINATION TESTS
        IF CT .EQ. TF) RETURN
        IF (ABS(H) .GT. HMIN) GOTO 10
C     **END OF LOOP
C
      WRITE(IW,2) T
    2 FORMAT('0APPARENT SINGULARITY NEAR T =',E10.3)
      RETURN
      END
```

รูป 8.2-5 ซับรูทีน RKF4

```
      SUBROUTINE SUMK(J, P, Q1, Q2, Q3, Q4, Q5)
      REAL K(10,6), F(10)
      DOUBLE PRECISION Y(10), SUM(10)
      COMMON N, T, TF, Y
      COMMON /KCALC/ H, K
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - C
C       THIS SUBROUTINE EVALUATES THE N-VECTOR                      C
C                                                                   C
C         SUM = Y + Q1*ROW1(K) + . . . + Q5*ROW5(K)                 C
C                                                                   C
C       IF J = 0, IT SETS Y = SUM.                                  C
C       IF 1 <= J <= 5, IT PUTS F(T+P*H,SUM) IN JTH COLUMN OF K     C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - C
      DO 10 I=1,N
         SUM(I) = Y(I) + DBLE(Q1)*K(I,1) + DBLE(Q2)*K(I,2) +
     &            DBLE(Q3)*K(I,3) + DBLE(Q4)*K(I,4) + DBLE(Q5)*K(I,5)
         IF (J .EQ. 0) Y(I) = SUM(I)
   10 CONTINUE
      IF (J .EQ. 0) RETURN

         CALL EVALF(T+P*H, SUM, F)
         DO 20 I=1,N
            K(I,J) = H*F(I)
   20    CONTINUE
      RETURN
      END
```

```
C * * * CALLING PROGRAM FOR SUBROUTINE RKF4 * * *
      DOUBLE PRECISION YO(10)
      COMMON N, TO, TF, YO
      DATA IW, IR /5, 5/
      DATA SCAMIN, SCAMAX, RMAX /0.1, 4.0, 1.E-4/
C
      WRITE(IW,1)
    1 FORMAT('OINPUT N, TO, TF, NPRINT')
      READ(IR,*) N, TO, TF, NPRINT
C
      WRITE(IW,2) N
    2 FORMAT(' INPUT',I2,' COMPONENTS OF YO')
      READ(IR,*) (YO(I), I=1,N)
C
      WRITE(IW,3) TO, TF, NPRINT
    3 FORMAT('OINTEGRATING FROM',F8.4,' TO',F8.4,
     &       ', PRINTING EVERY',I5,' STEPS'//5X,
     &       'J',6X,'T',10X,'H',8X,'-----Y------>')
C
      CALL RKF4(RMAX, SCAMIN, SCAMAX, NPRINT, IW)
C
      STOP
      END
```

```
         SUBROUTINE EVALF(T,Y,F)
         DIMENSION F(1)
         DOUBLE PRECISION Y(1)
C
         F(1) = -T*Y(1)*Y(1)
C
         RETURN
         END
```

รูป 8.2-6 ซับรูทีน SUMK

8.3 วิธีแบบหลายขั้น (Multistep Methods)

มาตรการตัวทำนาย-ตัวแก้ (Predictor-Corrector Strategies)

สำหรับวิธีที่เราได้พิจารณามาแล้ว (เทย์เลอร์ และ รุงเง-คุตตา) นั้น การคำนวณค่า f(t,y) ทุกค่า (ซึ่งต้องใช้เพื่อคำนวณ y_{j+1}) นั้นจะต้องหาหลังจากที่ได้หา y_j แล้ว วิธีที่มีคุณสมบัติดังกล่าวถูกเรียกว่า self-starting เพราะว่าวิธีถูกประยุกต์ใช้โดยการเริ่มต้นด้วยค่าเริ่มต้น y_o แต่โชคไม่ดีที่วิธีเหล่านี้ไม่ได้ใช้ประโยชน์จากค่าที่คำนวณได้มาก่อนแล้ว เช่น

$$\ldots, y_{j-4}, y_{j-3}, y_{j-2}, y_{j-1} \text{ โดยที่ } y_i \approx y(t_i) \qquad \ldots(1a)$$

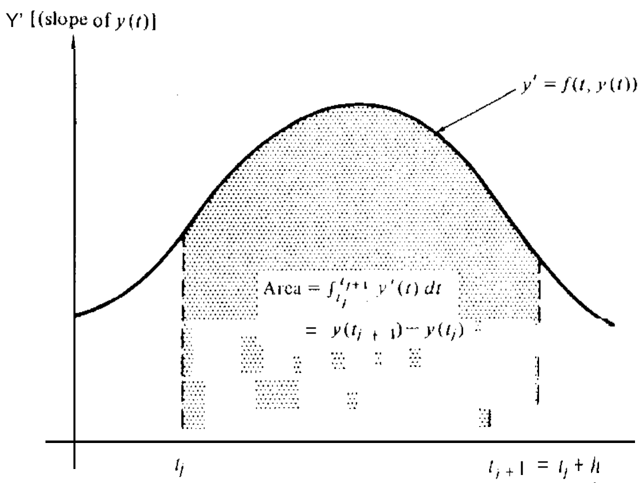
$$\ldots, f_{j-4}, f_{j-3}, f_{j-2}, f_{j-1} \text{ โดยที่ } f_i = f(t_i,y_i) \approx y'(t_i) \qquad \ldots(1b)$$

วิธีการที่จะใช้ข้อมูลเกี่ยวกับ y(t) และ y'(t) สำหรับ $t < t_j$ ในการคำนวณ y_{j+1} ถูกเรียกว่า **วิธีแบบหลายขั้น (multistep methods)**

เนื่องจากว่าค่าใน (1) นั้นไม่มีเมื่อ j = 0 วิธีแบบหลายขั้นจึงไม่เป็น self-starting method อย่างไรก็ดีเมื่อได้เริ่มต้นแล้ว ค่าใน (1) จะถูกใช้ในการหา y_1, y_2,...ให้ได้ความแม่นยำตามต้องการด้วยการคำนวณ f(t,y) น้อยครั้งลงในแต่ละขั้นกว่า self-starting method

จากวิชาแคลคูลัสเราจะได้ (ดูรูป 8.3-1)

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} f(t,y(t))dt \qquad \ldots(2)$$



รูป 8 . 3 - 1 การอินทิเกรท f'(t) = f(t,y(t)) เพื่อหา $y(t_{j+1}) - y((t_j)$

เนื่องจากว่า โดยทั่ว ๆ ไปเราไม่ทราบ $y(t)$ เราจึงไม่อาจใช้ (2) ได้โดยตรง อย่างไรก็ดี เมื่อเรามีค่าของ $y_j \approx y(t_j)$ และ $f_j \approx y'(t_j)$ เราอาจใช้ทั้ง 2 ค่า ใน Quadrature formula สำหรับอินทิกรัลใน (2) เพื่อที่จะหา $y_{j+1} \approx y(t_{j+1})$ ผลจะได้สูตรในรูป

$$\boxed{y_{j+1} = y_j + \underbrace{h\,\Phi(h, t_j, y_j, f_j \text{ และอาจมี } y\text{'s และ } f(t,y)\text{'s อื่น ๆ อีก})}_{O(h^{n+1}) \text{ formula สำหรับอินทิเกรท } f(t,y) \text{ on } [t_j, t_{j+1}]}} \qquad \ldots(3)$$

การใช้ $(n+1)$st-order quadrature formula ใน (3) จะให้ nth-order method

8.3A วิธีตัวทำนาย-ตัวแก้อันดับที่สี่ของอาดามส์ (Adams Predictor-Corrector Hethod (APC4))

จากการให้ $h\Phi$ ใน (3) เป็น $O(h^5)$ Adams formulas [(17) และ (18) ของหัวข้อ 7.3B] เราจะได้

ตัวทำนายของอาดามส์ (Adams Predictor)

$$p_{j+1} = y_j + (h/24)[-9f_{j-3} + 37f_{j-2} - 59f_{j-1} + 55f_j] \qquad \ldots(4a)$$

$$e_{j+1}[h]_p = y(t_{j+1}) - p_{j+1} = (251/720)y^{(v)}(\lambda_p)h^5$$

โดยที่ $t_{j-3} <= \lambda_p <= t_{j+1}$ $\qquad \ldots(4b)$

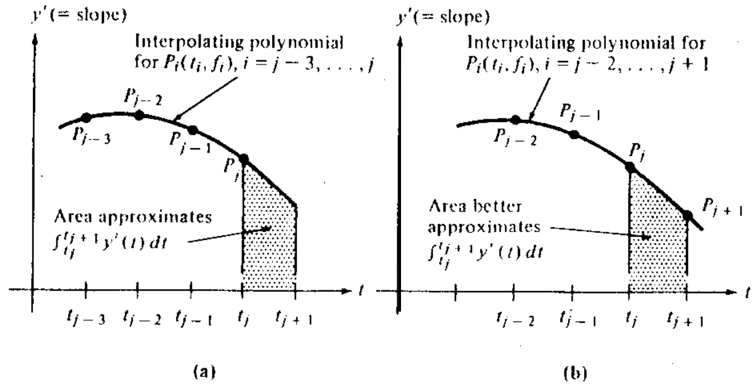
ตัวแก้ของอาดามส์ (Adams Corrector)

$$c_{j+1} = y_j + (h/24)[f_{j-2} - 5f_{j-1} + 19f_j + 9f_{j+1}] \qquad \ldots(5a)$$

$$e_{j+1}[h]_c = y(t_{j+1}) - c_{j+1} = (-19/720)y^{(v)}(\lambda_c)h^5$$

โดยที่ $t_{j-2} <= \lambda_c <= t_{j+1}$ $\qquad \ldots(5b)$

รูป 8.3-2 แสดงความหมายทางกราฟฟิคของ (4a) และ (5a)



รูป 8.3-2 (a) วิธีตัวทำนายของอาดามส์ (b) วิธีตัวแก้ของอาดามส์

$\delta_{j+1} = (-19/270)[c_{j+1} - p_{j+1}]$ เป็นตัวประมาณค่าที่คำนวณได้ของ $e_{j+1}[h]_c$

...(6)

ถ้า δ_{j+1} ชี้ว่า c_{j+1} ไม่แม่นยำเพียงพอ เราอาจจะคำนวณ c_{j+1} ใหม่ โดยใช้ $f(t_{j+1}, c_{j+1})$ เพื่อเป็นค่าประมาณของ f_{j+1} ที่แม่นยำกว่าวิธีตัวทำนาย-ตัวแก้ ซึ่งทำเช่นนี้โดยใช้ (4) และ (5) ถูกเรียกว่า **วิธีของอาดามส์อันดับสี่** (fourth-order Adam method **(APC4))** หรืออาจเรียกว่า **วิธีอาดามส์-แบชฟอร์ท** (Adam-Bashforth method) หรือ **วิธีอาดามส์-มูลตัน (Adams-Moulton** method)

<u>ตัวอย่าง</u> จงใช้ APC4 on [2,3], h=0.1 สำหรับ IVP

$y' = -ty^2$, $y(2) = 1$ [ผลเฉลยแม่นตรงคือ $2/(t^2-2)$]

พยายามหาคำตอบถึงประมาณ 4s และเมื่อเริ่มต้นวิธีการ ให้ใช้ค่าที่แท้จริง (to 7s) ต่อไปนี้

$t_0 = 2.0$: $y_0 = y(2.0) = 1.0000000$, $f_0 = -t_0y_0^2 = -2.000000$

$t_1 = 2.3$: $y_1 = y(2.1) \doteq 0.8296755$, $f_1 = -t_1y_1^2 \doteq -1.446256$

$t_2 = 2.2$: $y_2 = y(2.2) \doteq 0.7042254$, $f_2 = -t_2y_2^2 \doteq -1.091053$

$t_3 = 2.3$: $y_3 = y(2.3) \doteq 0.6079027$, $f_3 = -t_3y_3^2 \doteq -0.6499552$

Solution

$j = 3$: $p_4 = y_3 + (h/24)\{-9f_0 + 37f_1 - 59f_2 + 55f_3\}$

$\doteq 0.5333741$

$y_4 = y_3 + (h/24)\{f_1 - 5f_2 + 19f_3 - 9(-t_4 p_4^2)\}$

$\doteq 0.5317149$ $(=c_4)$

$\delta_4 = (-19/270)(y_4 - p_4) \doteq 0.0001144$

เนื่องจากว่า δ_4 แสดงว่าอาจมีความไม่แม่นยำในทศนิยมตำแหน่งที่ 4 (ซึ่งก็คือตัวเลขนัยสำคัญตัวที่ 4) ของ y_4 เราใช้ y_4 เป็นค่าที่ปรับปรุงแล้วของ p_4 เพื่อหาค่าที่ปรับปรุงของ y_4 ดังนี้

$y_4 = y_3 + (h/24)\{f_1 - 5f_2 + 19f_3 - 9[-t_4(0.5317149)^2]\} \doteq 0.5318739$

และ $\delta_4 = (-19/270)(0.5318739 - 0.5317149) \doteq -0.0000112$

ซึ่งแสดงว่า y_4 ควรจะมีความแม่นยำประมาณ 5s

$j = 4$: $f_4 = f(t_4, y_4) = -(2.4)(0.5318739)^2 \doteq -0.6789358$

$p_5 = y_4 + (h/24)(-9f_1 + 37f_2 - 59f_3 + 55f_4) \doteq 0.4712642$

$y_5 = y_4 + (h/24)[f_2 - 5f_3 + 19f_4 - 9[-t_5 p_5^2)]$

$\doteq 08.4704654$ $(=c_5)$

$\delta_5 = (-19/270)(y_5 - p_5) \doteq 0.0000562$

ดังเช่นที่ผ่านมา δ_5 ชี้ให้เห็นว่าตัวเลขนัยสำคัญตัวที่ 4 ของ y_5 อาจจะขาดความแม่นยำ ดังนั้นเราจะปรับปรุง y_5 ดังนี้

$y_5 = y_4 + (h/24)\{f_2 - 5f_3 + 19f_4 - 9[-t_5(0.4704654)^2]\} \doteq 0.4705358$

และ $\delta_5 = (-19/270)(0.4705358 - 0.4704654) \doteq -0.0000050$

ซึ่งแสดงว่า y_5 นั้นควรจะมีความแม่นยำประมาณ 5s

ในตาราง 8.3-1 แสดงว่าความจริง y_4 และ y_5 มีความแม่นยำเพียงประมาณ 4s นอกจากนั้นเราได้ **corrected** y_4 ถึง 2 ครั้ง (โดยใช้ 0.531874 เป็นค่า p_4) ผลที่ได้คือ y_4 นั้นมีความแม่นยำน้อยกว่า 0.531874 โดยทั่ว ๆ ไปจากประสบการณ์แสดงให้เห็นว่า:

The values obtained by performing at most one correction of Y_{j+1} are as likely to be accurate as those obtained using any general strategy for iterating the corrector formula. If more than one correction appears necessary, decrease the stepsize.

ตาราง 8.3-1

ค่าจากสูตร APC4 และ RK4 สำหรับ $y' = -ty^2$, $y(2) = 1$ ($h = 0.1$)

		using *APC4* (h = 0.1)		*Using RK4 (h = 0.1)*	
t_j	*Exact* $y(t_j)$	y_j	$E_j[h]$	y_j	$E_j[h]$
t_0 = 2.0	1.000000	Exact	—	Exact	—
2.1	0.829876	Exact	—	0.829885	−0.000009
2.2	0.704225	Exact	—	0.704237	−0.000012
2.3	0.607903	Exact	—	0.607914	-0.000011
2.4	0.531915	0.531874	0.000041	0.531924	−0.000009
2.5	0.470588	0.470536	0.000052	0.470596	−0.000008
2.6	0.420168	0.420114	0.000054	0.420175	−0.000007
2.7	0.378072	0.378020	0.000052	0.378078	−0.000006
2.8	0.342466	0.342419	0.000047	0.342471	−0.000005
2.9	0.312012	0.311971	0.000041	0.312017	−0.000005
t_F = 3.0	0.285714	0.285674	0.000040	0.285718	−0.000004

ในตาราง 8.3-1 ทำการคำนวณ $f(t,y)$ 3 ครั้ง [ครั้งแรกที่ t_j, y_j) เพื่อหา p_{j+1} ครั้งต่อไปที่ (t_{j+1}, p_{j+1}) เพื่อหา y_{j+1} และอีกครั้งที่ (t_{j+1}, y_{j+1}) เพื่อหา corrected y_{j+1}] ในทุกขั้นตอนยกเว้นขั้นสุดท้าย ซึ่งทำการคำนวณเพียง 2 ครั้ง สำหรับปัญหานี้ APC4 ใช้ stepsize เล็กกว่า h = 0.1 เพื่อให้ได้ความแม่นยำ 5s h = 0.1 เหมาะสมสำหรับ RK4 เพราะว่า $f(t,y) = -ty^2$ คือ polynomial of degree <= 4 ใน t และ y APC4 จะมีความแม่นยำมากกว่า RK4 เมื่อ y(t) คือ polynomial of degree <= 4 ใน t

รูป 8.3-3 คือ APC4 algorithm เรามักกำหนด MaxIt ให้เท่ากับ 2 วิธีที่จะถูกใช้เพื่อเริ่มต้นวิธีตัวทำนาย-ตัวแก้นั้นควรจะมีอันดับเดียวกับวิธีตัวทำนาย-ตัวแก้เองเป็นอย่างน้อย RK4 มักจะถูกใช้ในการเริ่มต้น APC4

Algorithm: APC4 (Adams Fourth-Order Predictor-Corrector Method)

Purpose: To solve, on the interval $[t_0, t_F]$, the IVP

$$y' = f(t, y), \; y = y_0 \text{ when } t = t_0$$

using a specified number of steps to obtain *NumSig* significant digit accuracy. The three preceding slope values f_{j-1}, f_{j-2}, and f_{j-3} will be stored as simply f_{-1}, f_{-2}, and f_{-3}.

{*initialize*}
GET t_0, t_F, y_0, (parameters of IVP}
NumberOfSteps, [from t_0 to t_F; at thejth step, $t_j = t_0 + jh$}
NumSig, *MaxIt* (maximum number of iterations of corrector)
$h \leftarrow (t_F - t_0)/NumberOfSteps$; $RelTol \leftarrow 10^{-NumSig}$
$t \leftarrow t_0$; $Y \leftarrow y_0$ {(t, y) is the current (t_j, y_j).}
DO FOR $j = 0$ TO 2 (Initialize f_{-3}, f_{-2}, f_{-1}.}
BEGIN
$f_{j-3} \leftarrow f(t, y)$
Use a self-starting method to get *Ynew* {*Ynew* approximates $y(t + h)$.}
$t \leftarrow t + h$; $y \leftarrow Ynew$
OUTPUT $(j + 1, t, y, f_{j-3})$
END
{Now $t = t_3$, $y = y_3$, and f_{-3}, f_{-2}, f_-, are slope values at t_0, t_1, t_2 respectively.)
$f \leftarrow f(t, Y)$ {f is f_j, the slope at (t_j, y_j).}

{*iterate*}
DO FOR $j = 3$ TO ($NumberOfSteps - 1$)
BEGIN
[predict] $p \leftarrow Y - \dfrac{h}{24}[-9f_{-3} + 37f_{-2} \quad 59f_{-1} + 55f]$
DO FOR $k = 1$ TO *MaxIt* UNTIL **termination test** is satisfied
BEGIN
(correct] $c \leftarrow y + \dfrac{h}{24}[f_{-2} - 5f_{-1} + 19f + 9f(t + h, p)]$
$Delta \leftarrow -\frac{19}{270}(c - p)$ (estimate of $e_c[h]$}
$p \leftarrow c$ (prepare for another iteration)
[termination test: $|Delta| \leq RelTol * |c|$}
END

```
   IF termination test was not satisfied
     THEN OUTPUT (Smaller stepsize may be needed for desired accuracy.)
   (update)
   t ← t + h;   y ← c
   f-3 ← f-2; f-2 ← f-1; f-1 ← f; f ← f(t, y)
   OUTPUT (j + 1, t, y, f)
   END

IF termination test was satisfied at each step
   THEN OUTPUT (They values should be accurate to NumSig significant digits.)
```

รูป 8.3-3 รหัสเทียมสำหรับขั้นตอนวิธี APC4

8.3B <u>การควบคุม Stepsize ของวิธีตัวทำนาย-ตัวแก้ (Predictor-Corrector (PC) Methods)</u>

เราอาจใช้ δ_{j+1} เพื่อควบคุม stepsize ทำนองเดียวกับวิธีที่ใช้ค่าประมาณความคลาดเคลื่อน ในรูป 8.2-3 (หัวข้อ 8.2C) อย่างไรก็ดีจะต้องเริ่มวิธีการใหม่ทุกครั้งที่ h เปลี่ยนค่า เราไม่ควรเปลี่ยนแปลงค่า h บ่อยเกินไป วิธีการที่แนะนำสำหรับวิธีอันดับที่ n คือ

Ratio <-- δ_{j+1} /h
IF Ratio > BigRatio หรือ Ratio < SmallRatio ...(7)
THEN h <-- h[DesiredRatio/Ratio]$^{1/n}$

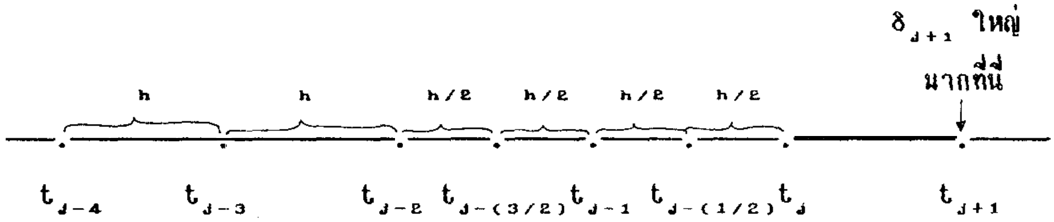
ใน (7) DesiredRatio = C*10^{-s} เมื่อ s คือจำนวนตัวเลขนัยสำคัญที่ต้องการ ผลของ (7) คือเพื่อลด h เมื่อ Ratio ใหญ่เกินไปจนยอมรับไม่ได้ และเพื่อเพิ่ม h เมื่อ Ratio แสดงว่า h นั้นไม่จำเป็นต้องมีค่าเล็ก

ถ้าเวลาหรือค่าใช้จ่ายสำหรับการเริ่มต้นวิธีตัวทำนาย-ตัวแก้อันดับที่สี่ นั้นเป็นแฟกเตอร์ซึ่งวิกฤต เราอาจประหยัดโดยละการบวก f_{j-4} เข้ากับ f_i's ในการหา y_{j+1} และอาจเพิ่ม h เป็น 2 เท่าหรือลด h ลงครึ่งหนึ่งดังนี้

HALVING H (ลดลงครึ่งหนึ่ง): ทำ interpolate quartic กับจุด 5 จุดคือ $(t_{j-4}, f_{j-4}), \ldots, (t_j, y_j)$ เพื่อหาค่า

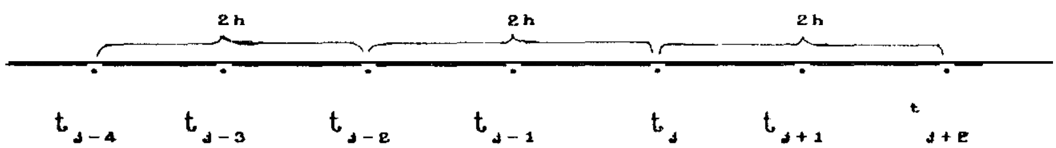
$$f_{j-(1/2)} = (1/128)[-5f_{j-4} + 28f_{j-3} - 70f_{j-2} + 140f_{j-1} + 35f_j] \quad \ldots(8a)$$

$$f_{j-(3/2)} = (1/64)[3f_{j-4} - 16f_{j-3} + 54f_{j-2} + 24f_{j-1} - f_j] \quad \ldots(8b)$$



แล้วทำต่อจาก (t_j, y_j) ด้วย $h/2$

DOUBLING H (เพิ่มสองเท่า): ทำอีก 1 ขั้นเพื่อให้ได้ t_{j+2} แล้วใช้ f_{j-4}, f_{j-2}, f_j และ f_{j+2} เพื่อทำต่อจาก (t_{j+2}, y_{j+2}) ด้วย $2h$



แบบฝึกหัดบทที่ 8

โจทย์ข้อ 8.1-8.5 จะต้องใช้ IVP's ที่กำหนดให้ต่อไปนี้

(A) $y' = (y/t) - 2,\ y(1) = 2;\ h = 0.1$

[ผลเฉลยแม่นตรง: $y(t) = 2t(1-\ln t)$]

(B) $y' = (1/2)[(y/t)+(t/y)],\ y(1) = 3;\ h = 0.2$

[ผลเฉลยแม่นตรง: $y(t) = t^2\ t\ 8tl$

(C) $y' = y\ ,\ y(0) = 1;\ h = 0.2$

[ผลเฉลยแม่นตรง: $y(t) = (t\ t\ 2)^2/4$]

8.1 จงแสดงว่า $y(t)$'s ใน (A)-(C) คือผลเฉลยของ IVP's ที่กำหนดให้

8.2 (a) สำหรับ (A)-(C) จงใช้วิธีของออยเลอร์เพื่อทำ

(i) 2 h-steps เพื่อหาถึง $t = t_o + 2h$

(ii) 4 (h/2)-steps เพื่อหาถึง $t = t_o + 2h$

(b) ค่าประมาณ $y(t_o + h)$ และ $y(t_o + 2h)$ ที่ได้จาก (a) นั้นยืนยันหรือไม่ว่าวิธีของออยเลอร์ นั้นคือวิธีอันดับหนึ่ง (นั่นคือ $E(h/2) \approx E(h)/2$) ถ้าเป็นดังกล่าวจงใช้สูตรของริชาร์ดสัน เพื่อหาค่าประมาณที่ถูกปรับปรุงแล้วของ $y(t_o + 2h)$ และอภิปรายถึงความแม่นยำที่ได้รับด้วย

8.3 สำหรับ (A)-(C) จงใช้วิธีของเทย์เลอร์อันดับสอง เพื่อทำ

(i) 1 h-step เพื่อหาถึง $t = t_o + h$

(ii) 2 (h/2)-steps เพื่อหาถึง $t = t_o + h$

8.4 สำหรับ (A)-(C) จงทำซ้ำข้อ 8.3 สำหรับวิธีของออยเลอร์ที่ปรับปรุงแล้ว (Modified Euler method)

8.5 สำหรับ (A)-(C) จงทำซ้ำข้อ 8.3 สำหรับวิธีของฮวน (Huen's method)

8.6* สำหรับ (A)-(C) จงทำ 2 h-steps ของ RK4 จงเปรียบเทียบความแม่นยำที่ $t_o + h$ กับค่าอย่างเดียวกันที่ได้จากวิธีอื่น ๆ

8.7 สำหรับ (A)-(C) จงทำ 2 h-steps ของ RKF4 โดยใช้ Rmax = h^4 และค่า h จาก (A)-(C) จงเปรียบเทียบความแม่นยำที่ $t_o + h$ กับค่าอย่างเดียวกันที่ได้จากวิธีอื่น ๆ

Answers:

	Approx. of	(a)(i) Using h	(ii) Using h/2	(b) Improved
8.2 (A)	y(1.1)	2.00000	1.99524	1.33048
	y(1.2)	1.98182	1.97228	1.36274
(B)	y(1.2)	3.33333	3.32797	3.32261
	y(1.4)	3.64711	3.63735	3.62759
(C)	y(0.2)	1.20000	1.20488	1.20376
	y(0.4)	1.41909	1.42931	1.43353
8.4 (A)	y(1.1)	1.33048	1.33036	1.33032
(B)	y(1.2)	3.32262	3.32264	3.32263
(C)	y(0.2)	1.20976	1.20994	1.21000
8.5 (A)	y(1.1)	1.33031	1.33047	1.99032
(B)	y(1.2)	3.32356	3.32288	3.32265
(C)	y(0.2)	1.20955	1.20988	1.20999

8.6 y_1 = 1.93032 for (A); 3.32265 for (B); 1.21000 for (C).

y_2 = 1.36243 for (A); 3.62767 for (B); 1.44000 for (C).

8.7 (A) y_1 = 1.93032 ≈ y(1.1); y_2 = 1.83606 ≈ y(1.42858)

[h=0. 326561.

(B) y_1 = 3.32265 ≈ y(1.2); y_2 = 4.47216 ≈ y(2.0) [h=0.8].

(C) y_1 = 1.21000 ≈ yiO.2); y_2 = 2.25001 ≈ y(1.0) [h=0.8].