Runge-Kutta Methods.

The Runge-Kutta methods improves on Euler' s method for solving the Initial Value Problem (IVP) $y' = f(t, y)$, $y(t_0) = y_0$, without requiring the calculation of higher order derivatives.

According to this theory, the general expression for the explicit methods of *s* stages of Runge-Kutta is:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

$$k_i = f\left(t_n + c_i h,\ y_n + h \sum_{j=1}^{s} a_{ij} k_j\right)$$

where $a_{ij} = 0$ to $j \geq i$ and $\sum_{j=1}^{s} a_{ij} = c_i$.

A classic Runge-Kutta method of second order with 2-stages has the following Butcher tableau:

0	0	.
c_2	c_2	0
.	b_1	b_2

where the coefficients of the diagram verify the equation system:

$$b_1 + b_2 = 1$$
$$b_2\, c_2 = \frac{1}{2}$$

There is, therefore, an infinite family of second order Runge-Kutta's methods. The most commonly used are:

a) Euler' s modified method, which corresponds to $b_1 = 0$, $b_2 = 1$, $c_2 = \frac{1}{2}$. Its expression is $y_{n+1} = y_n + h\, k_2$ with $k_1 = f(t_n, y_n)$ and $k_2 = f\left(t_n + \frac{h}{2},\ y_n + \frac{h\, k_1}{2}\right)$.

Next, two ***Mathematica*** procedures, **eulermod** and **eulermodgraf**, will be programmed. These allow the calculation of tables of values and graphical representation of the approximate solution obtained using the modified Euler method:

```
eulermod[f_, h_, ini_, a_, b_] :=
Module[{yrk, t, y, rktable1, c},
 c = (b - a) / h;
 yrk[0] = ini;
 t[n_] := a + n h;
 yrk[n_] :=
  Module[{k1, k2},
   k1 = f[t[n - 1], yrk[n - 1]];
   k2 = f[t[n - 1] + h/2, yrk[n - 1] + h/2 k1];
   yrk[n] = yrk[n - 1] + h k2];
 rktable1 = Table[yrk[i], {i, 0, c}];
Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
eulermodgraf[f_, h_, ini_, a_, b_] := Module[{yrk, t, y, rktable1, c}, c = (b - a)/h;
  yrk[0] = ini; t[n_] := a + n h; yrk[n_] := Module[{k1, k2}, k1 = f[t[n - 1], yrk[n - 1]];
    k2 = f[t[n - 1] + h/2, yrk[n - 1] + h k1/2]; yrk[n] = yrk[n - 1] + h k2];
  rktable1 = Table[yrk[i], {i, 0, c}]; ListPlot[Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}],
   Joined → True, PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable1[[c + 1]]]]
```

where **f** is the function associated with the equation differential, **h** is the step length, **ini** is the value of the initial condition and **a** and **b** are the boundaries of the interval.

b) The improved Euler's method, which corresponds to $b_1 = \frac{1}{2}$, $b_2 = \frac{1}{2}$, $c_2 = 1$. Its expression is $y_{n+1} = y_n + \frac{h(k_1+k_2)}{2}$ with $k_1 = f(t_n, y_n)$ and $k_2 = f(t_n + h, y_n + h k_1)$.

Nex,t two ***Mathematica*** procedures, **mejoreuler** and **mejoreulergraf**, are programmed. These calculate a table of values and provide a graphical representation of the solution using the improved Euler method:

```
mejoreuler[f_, h_, ini_, a_, b_] :=
Module[{yrk, t, y, rktable1, c},
 c = (b - a) / h;
 yrk[0] = ini;
 t[n_] := a + n h;
 yrk[n_] :=
  Module[{k1, k2},
   k1 = f[t[n - 1], yrk[n - 1]];
   k2 = f[t[n - 1] + h, yrk[n - 1] + h k1];
   yrk[n] = yrk[n - 1] + (h / 2) (k1 + k2)];
 rktable1 = Table[yrk[i], {i, 0, c}];
Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
mejoreulergraf[f_, h_, ini_, a_, b_] := Module[{yrk, t, y, rktable1, c}, c = (b - a)/h;
  yrk[0] = ini; t[n_] := a + n h; yrk[n_] := Module[{k1, k2}, k1 = f[t[n - 1], yrk[n - 1]];
    k2 = f[t[n - 1] + h, yrk[n - 1] + h k1]; yrk[n] = yrk[n - 1] + 1/2 h (k1 + k2)];
  rktable1 = Table[yrk[i], {i, 0, c}]; ListPlot[Table[{t[i], rktable1[[i + 1]]}, {i, 0, c}],
   Joined → True, PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable1[[c + 1]]]]
```

where **f** is the function associated with the equation differential, **h** is the step length, **ini** is the value of the initial condition and **a** and **b** are the boundaries of the interval.

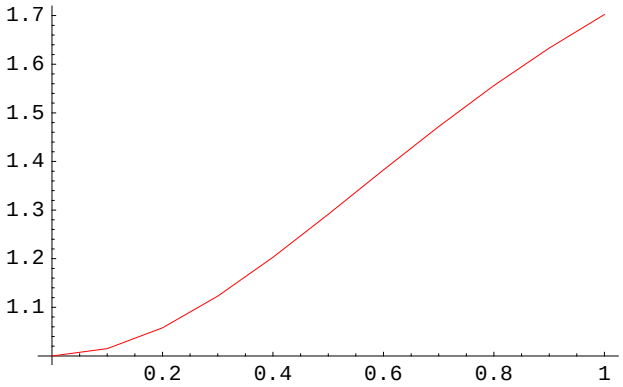
Both methods are used to solve the IVP $y' = -t\,y + \frac{4t}{y}$, $y(0)=1$ on the interval [0,1], with step length 0.1.

```
f[t_, y_] = -t y + 4 t/y;
```

```
eulermod[f, 0.1, 1, 0, 1]
```

```
0       1
0.1     1.015
0.2     1.05783
0.3     1.12286
0.4     1.20303
0.5     1.29151
0.6     1.38258
0.7     1.47185
0.8     1.55615
0.9     1.63337
1.      1.70225
```

```
eulermodgraf[f, 0.1, 1, 0, 1]
```

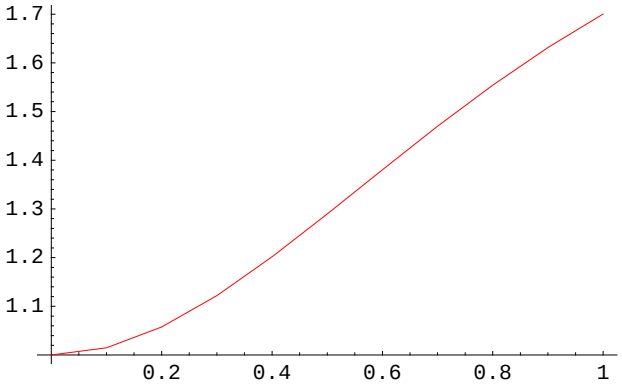



```
y[1.]=1.70225
```

```
mejoreuler[f, 0.1, 1, 0, 1]
```

```
0       1
0.1     1.015
0.2     1.05749
0.3     1.12202
0.4     1.20169
0.5     1.28977
0.6     1.38058
0.7     1.46972
0.8     1.55398
0.9     1.63123
1.      1.70021
```

```
mejoreulergraf[f, 0.1, 1, 0, 1]
```




```
y[1.]=1.70021
```

The previous chapter on one step methods shows that the exact solution of this IVP with $t = 1$ is 1.70187. Therefore, in this case, the approximation obtained by the first Runge-Kutta' smethod is better that the approximation obtained with the second one.

A classic Runge-Kutta's method of third order with 3-stages has the following Butcher tableau

0	0	.	.
c_2	c_2	0	.
c_3	$c_3 - a_{32}$	a_{32}	0
.	b_1	b_2	b_3

where the coefficients of the diagram verify the equation system:

$$b_1 + b_2 + b_3 = 1$$
$$b_2\, c_2 + b_3\, c_3 = \frac{1}{2}$$
$$b_2\, {c_2}^2 + b_3\, {c_3}^2 = \frac{1}{3}$$
$$b_3\, c_2\, a_{32} = \frac{1}{6}$$

The equations are over determined, so consequently there are an infinite number of third order Runge-Kutta's methods. The most frequently used is:

0	0	.	.
$\frac{1}{2}$	$\frac{1}{2}$	0	.
1	−1	2	0
.	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

and its expression is $y_{n+1} = y_n + \frac{h(k_1+4k_2+k_3)}{6}$ with $k_1 = f(t_n, y_n)$, $k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h k_1}{2}\right)$ and $k_3 = f(t_n + h, y_n + 2h k_2 - h k_1)$.

Next, two ***Mathematica*** procedures, **Runge3** and **Runge3graf**, are programmed. These calculate a table of values and provide a graphical representation of the solution using the third order Runge-Kutta method:

```
Runge3[f_, h_, ini_, a_, b_] :=
Module[{yrk3, t, rktable3, c},
 c = (b - a) / h;
 yrk3[0] = ini;
 t[n_] := a + n h;
 yrk3[n_] :=
  Module[{k1, k2, k3},
   k1 = f[t[n - 1], yrk3[n - 1]];
   k2 = f[t[n - 1] + h / 2, yrk3[n - 1] + (h / 2) k1];
   k3 = f[t[n - 1] + h, yrk3[n - 1] - h k1 + 2 h k2];
  yrk3[n] = yrk3[n - 1] + h ((1 / 6) k1 + (2 / 3) k2 + (1 / 6) k3)];
 rktable3 = Table[yrk3[i], {i, 0, c}];
Table[{t[i], rktable3[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
Runge3graf[f_, h_, ini_, a_, b_] :=
 Module[{yrk3, t, rktable3, c}, c = (b - a)/h; yrk3[0] = ini; t[n_] := a + n h; yrk3[n_] :=
   Module[{k1, k2, k3}, k1 = f[t[n - 1], yrk3[n - 1]]; k2 = f[t[n - 1] + h/2, yrk3[n - 1] + (h k1)/2];
    k3 = f[t[n - 1] + h, yrk3[n - 1] - h k1 + 2 h k2]; yrk3[n] = yrk3[n - 1] + h (k1/6 + (2 k2)/3 + k3/6)];
  rktable3 = Table[yrk3[i], {i, 0, c}]; ListPlot[Table[{t[i], rktable3[[i + 1]]}, {i, 0, c}],
   Joined → True, PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable3[[c + 1]]]]
```

where **f** is the function associated with the equation differential, **h** is the step length, **ini** is the value of the initial condition and **a** and **b** are the boundaries of the interval.

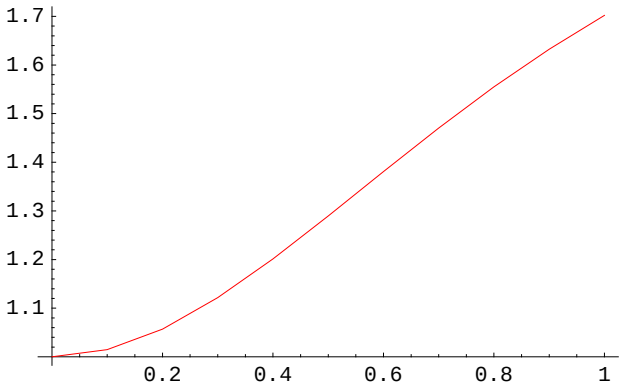
This method is used to solve the IVP $y' = -t\,y + \frac{4t}{y}$, *y(0)=1* on the interval [0,1], with step length 0.1.

```
f[t_, y_] = -t y + 4 t / y;
```

```
Runge3[f, 0.1, 1, 0, 1]
```

```
0       1
0.1     1.01476
0.2     1.05708
0.3     1.12157
0.4     1.20135
0.5     1.28967
0.6     1.38082
0.7     1.47033
0.8     1.55497
0.9     1.63259
1.      1.70187
```

```
Runge3graf[f, 0.1, 1, 0, 1]
```




```
y[1.]=1.70187
```

The most commonly used fourth order Runge-Kutta is represented by the following Butcher tableau:

0	0			
$\frac{1}{2}$	$\frac{1}{2}$	0		
$\frac{1}{2}$	0	$\frac{1}{2}$	0	
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

and its expression is

$$y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

$$K_1 = f(t_n, y_n)$$

$$K_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right)$$

$$K_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}K_2\right)$$

$$K_4 = f(t_n + h, y_n + hK_3)$$

Next, two ***Mathematica*** procedures, **Runge4** and **Runge4graf**, are programmed. These produce a table of values and give a graphical representation of the solution using the fourth order Runge-Kutta method:

```
Runge4[f_, h_, ini_, a_, b_] :=
Module[{yrk4, t, rktable4, c},
 c = (b - a) / h;
 yrk4[0] = ini;
 t[n_] := a + n h;
 yrk4[n_] :=
  Module[{k1, k2, k3, k4},
   k1 = f[t[n - 1], yrk4[n - 1]];
   k2 = f[t[n - 1] + h / 2, yrk4[n - 1] + (h / 2) k1];
   k3 = f[t[n - 1] + h / 2, yrk4[n - 1] + (h / 2) k2];
   k4 = f[t[n - 1] + h, yrk4[n - 1] + h k3];
   yrk4[n] =
    yrk4[n - 1] +
     (h / 6) (k1 + 2 k2 + 2 k3 + k4)];
 rktable4 = Table[yrk4[i], {i, 0, c}];
Table[{t[i], rktable4[[i + 1]]}, {i, 0, c}] // TableForm]
```

```
Runge4graf[f_, h_, ini_, a_, b_] := Module[{yrk4, t, rktable4, c}, c = (b - a)/h; yrk4[0] = ini;
  t[n_] := a + n h; yrk4[n_] := Module[{k1, k2, k3, k4}, k1 = f[t[n - 1], yrk4[n - 1]];
    k2 = f[t[n - 1] + h/2, yrk4[n - 1] + h k1/2]; k3 = f[t[n - 1] + h/2, yrk4[n - 1] + h k2/2];
    k4 = f[t[n - 1] + h, yrk4[n - 1] + h k3]; yrk4[n] = yrk4[n - 1] + 1/6 h (k1 + 2 k2 + 2 k3 + k4)];
  rktable4 = Table[yrk4[i], {i, 0, c}]; ListPlot[Table[{t[i], rktable4[[i + 1]]}, {i, 0, c}],
   Joined → True, PlotStyle → {RGBColor[1, 0, 0]}, PlotRange → All];
  Print["y[", t[c], "]=", rktable4[[c + 1]]]]
```

where **f** is the function associated with the equation differential, **h** is the step length, **ini** is the value of the initial condition and **a** and **b** are the boundaries of the interval.

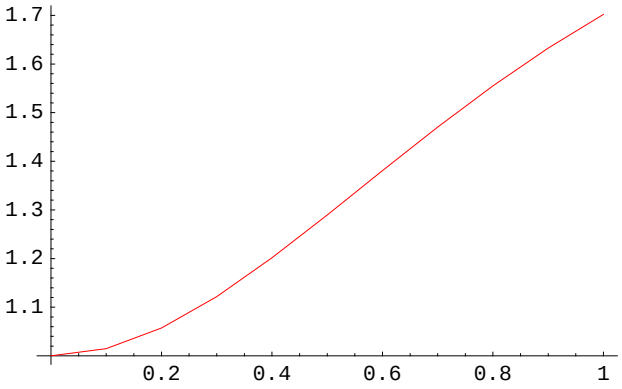
This method is used to solve the IVP $y' = -t\,y + \frac{4t}{y}$, *y(0)=1* on the interval [0,1], with step length 0.1.

```
f[t_, y_] = -t y + 4 t/y;
```

```
Runge4[f, 0.1, 1, 0, 1]
```

```
0       1
0.1     1.01482
0.2     1.05718
0.3     1.1217
0.4     1.20149
0.5     1.28981
0.6     1.38093
0.7     1.47042
0.8     1.55503
0.9     1.63261
1.      1.70187
```

```
Runge4graf[f, 0.1, 1, 0, 1]
```



y[1.]=1.70187

Example 1.

Using a fourth order Runge-Kutta method solve the IVP $y' = \frac{y^2-3t^2-2ty}{t^2+2ty}$, *y(1)=2, on the interval [1,2] with h=0.1*

Solution

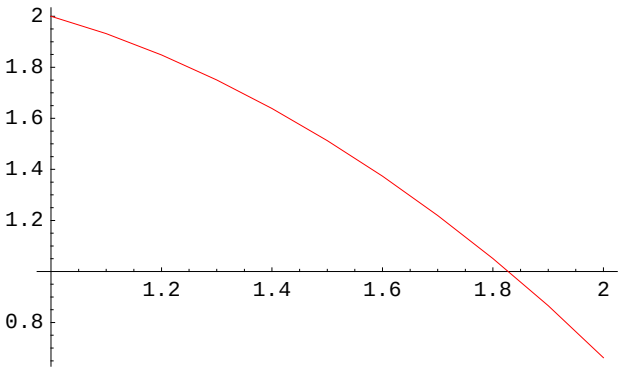
The function associated with the IVP is defined and the numerical method is applied:

```
f[t_, y_] := (y^2 - 3 t^2 - 2 t y)/(t^2 + 2 t y)
```

```
Runge4[f, 0.1, 2, 1, 2]
```

```
1       2
1.1     1.93191
1.2     1.84842
1.3     1.75041
1.4     1.63842
1.5     1.5127
1.6     1.37319
1.7     1.21949
1.8     1.05082
1.9     0.865842
2.      0.662386
```

```
Runge4graf[f, 0.1, 2, 1, 2]
```



y[2.]=0.662386

Example 2.

Starting with the IVP $y' = \sqrt{y} - \frac{20\, e^{-100(t-2)^2}}{\sqrt{\pi}}$, $y(1) = 1$, *obtain the approximate value of the solution at t=3, using the procedure Runge4 and with step length h=0.01. Analyze the graph produced, comparing it with that of the IVP* $y' = \sqrt{y}$, $y(1) = 1$.

Solution

It is important to see that ***Mathematica*** cannot solve this IVP:

```
DSolve[{y'[t] == Sqrt[y[t]] - 20 E^(-100 (t-2)^2)/Sqrt[π], y[1] == 1}, y[t], t]
```

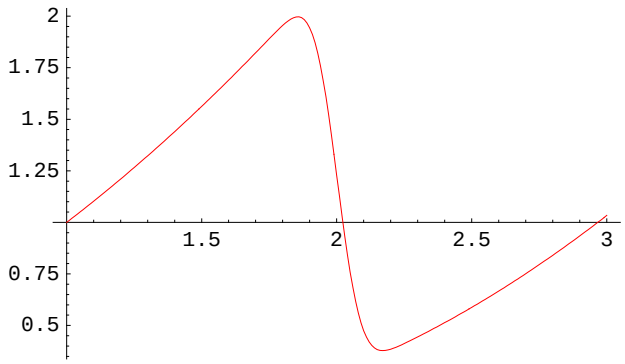
```
Solve::ifun :
 Inverse functions are being used by Solve, so some solutions may not be found;
   use Reduce for complete solution information. More...
```

```
DSolve[{y'[t] == -(20 E^(-100 (-2+t)^2)/Sqrt[π]) + Sqrt[y[t]], y[1] == 1}, y[t], t]
```

The function associated with the IVP is defined and the numerical method is applied

```
f[t_, y_] := Sqrt[y] - 20 E^(-100 (t-2)^2)/Sqrt[π]
```

```
Runge4graf[f, 0.01, 1, 3]
```

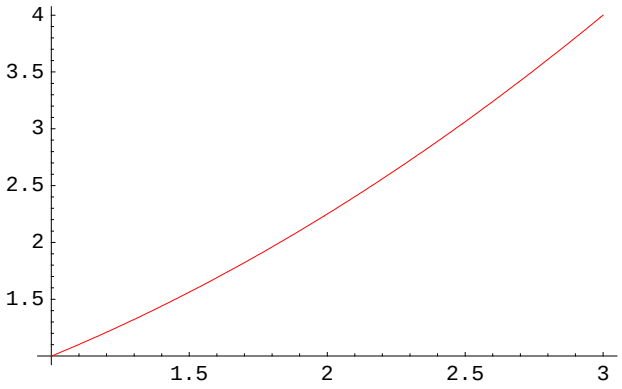



```
y[3.]=1.03349
```

Now, the second IVP is solved using the same procedure

```
f[t_, y_] := Sqrt[y]
```

```
Runge4graf[f, 0.01, 1, 3]
```




```
y[3.]=4.
```

The graphs coincide until just before $t = 2$, then diverge rapidly thereafter. The abrupt descent in the graph of the first IVP solution is because the impulse $\frac{20\, e^{-100\,(t-2)^2}}{\sqrt{\pi}}$ modifies the behaviour of the solution. The impulse is represented at follow:

```
Plot[20 e^-100 (t-2)^2 / Sqrt[π], {t, 1, 4}, PlotRange → All]
```