Numerical Methods for Differential Equations

Contents

Rectangular Rule

The approximation uses Euler's idea of replacing the integrand by a constant. The value of the integral is approximately the area of a rectangle of width $b - a$ and height $F(a)$.

$$\int_a^b F(x)dx \approx (b-a)F(a)$$

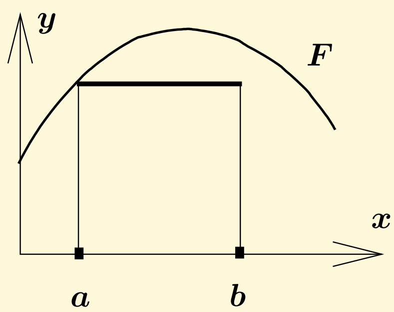


Figure 1. Rectangular Rule

Trapezoidal Rule

The rule replaces the integrand $F(x)$ by a linear function $L(x)$ which connects the planar points $(a, F(a))$, $(b, F(b))$. The value of the integral is approximately the area under the curve L, which is the area of a trapezoid.

$$\int_a^b F(x)dx \approx \frac{b-a}{2}\left(F(a) + F(b)\right)$$

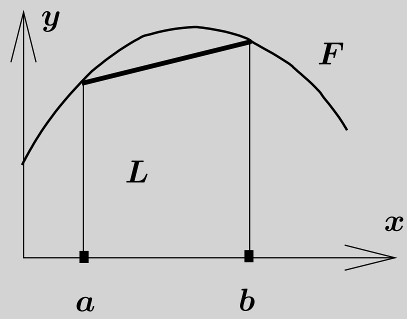


Figure 2. Trapezoidal Rule

Simpson's Rule

The rule replaces the integrand $F(x)$ by a quadratic polynomial $Q(x)$ which connects the planar points $(a, F(a))$, $((a+b)/2, F((a+b)/2))$, $(b, F(b))$. Then the integral of F is approximately the area under the quadratic curve Q.

$$\int_a^b F(x)dx \approx (b-a)\left(\frac{F(a) + 4F\left(\frac{a+b}{2}\right) + F(b)}{6}\right)$$

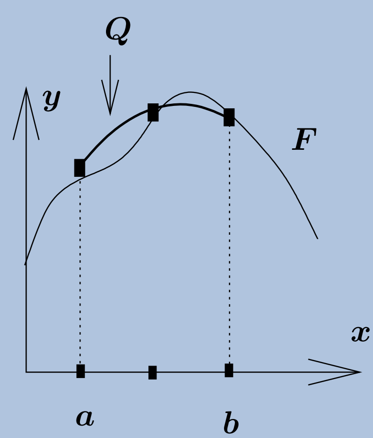
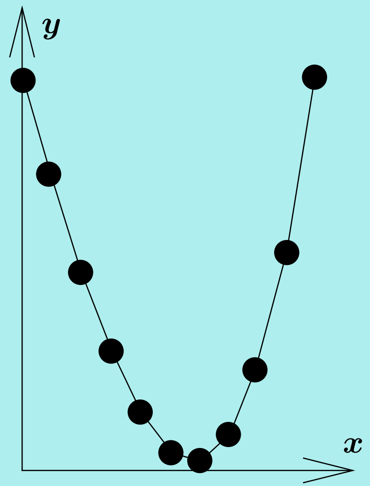


Figure 3. Simpson's Rule

How to make a connect-the-dots graphic

x	y
0.0	2.000
0.1	1.901
0.2	1.808
0.3	1.727
0.4	1.664
0.5	1.625

x	y
0.6	1.616
0.7	1.643
0.8	1.712
0.9	1.829
1.0	2.000



The table consists of xy-values for $y = x^3 - x + 2$. The graphic represents the table's rows, which are pairs (x, y), as *dots*. Joined dots make the *connect-the-dots* graphic.

Maple code

A connect-the-dots graphic can be made in `maple` by supplying a list L of pairs to be connected. An example:

```
L:=[[1,3],[2,1],[3,5]]:  plot(L);
```

Numerical Methods for $y' = F(x)$

Quadrature applies to give $y(x) = y_0 + \int_{x_0}^{x} F(x)dx$. Numerical solution methods amount to approximating the integral on the right by Rectangular, Trapezoidal and Simpson methods.

The methods replace the exact value of an integral $\int_{x_0}^{x_0+h} F(x)dx$ by a numerical approximation value which is useful for graphics when h is small. Larger intervals are broken into smaller intervals of length h, then the approximation is applied.

Table 1. Three numerical integration methods.

`Rect`	$Y = y_0 + hF(x_0)$
`Trap`	$Y = y_0 + \frac{h}{2}(F(x_0) + F(x_0 + h))$
`Simp`	$Y = y_0 + \frac{h}{6}(F(x_0) + 4F(x_0 + h/2) + F(x_0 + h))$

Maple code for the Rectangular and Trapezoid Rules

```
# Rectangular algorithm
# Group 1, initialize.
F:=x->evalf(cos(x) + 2*x):
x0:=0:y0:=0:h:=0.1*Pi:
Dots1:=[x0,y0]:

# Group 2, repeat 10 times
Y:=y0+h*F(x0):
x0:=x0+h:y0:=evalf(Y):
Dots1:=Dots1,[x0,y0];

# Group 3, plot.
plot([Dots1]);
```

```
# Trapezoidal algorithm
# Group 1, initialize.
F:=x->evalf(cos(x) + 2*x):
x0:=0:y0:=0:h:=0.1*Pi:
Dots2:=[x0,y0]:

# Group 2, repeat 10 times
Y:=y0+h*(F(x0)+F(x0+h))/2:
x0:=x0+h:y0:=evalf(Y):
Dots2:=Dots2,[x0,y0];

# Group 3, plot.
plot([Dots2]);
```

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/rect-algorithm.pdf

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/trap-algorithm.pdf

Maple code for Rectangular and Simpson Rules

```
# Rectangular algorithm
# Group 1, initialize.
F:=x->evalf(exp(-x*x)):
x0:=0:y0:=0:h:=0.1:
Dots1:=[x0,y0]:

# Group 2, repeat 10 times
Y:=evalf(y0+h*F(x0)):
x0:=x0+h:y0:=Y:
Dots1:=Dots1,[x0,y0];


# Group 3, plot.
plot([Dots1]);
```

```
# Simpson algorithm
# Group 1, initialize.
F:=x->evalf(exp(-x*x)):
x0:=0:y0:=0:h:=0.1:
Dots3:=[x0,y0]:

# Group 2, repeat 10 times
Y:=evalf(y0+h*(F(x0)+
   4*F(x0+h/2)+F(x0+h))/6):
x0:=x0+h:y0:=Y:
Dots3:=Dots3,[x0,y0];

# Group 3, plot.
plot([Dots3]);
```

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/rect2-algorithm.pdf

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/simp-algorithm.pdf

Numerical Methods for $y' = f(x, y)$

The methods replace the exact value of

$$y(x_0 + h) = y_0 + \int_{x_0}^{x_0+h} f(x, y(x))dx$$

by a numerical approximation Y. The value is useful for graphics when h is small.

Table 2. Three numerical methods for $y' = f(x, y)$.

`Euler`	$Y = y_0 + hf(x_0, y_0)$
`Heun`	$y_1 = y_0 + hf(x_0, y_0)$
	$Y = y_0 + \dfrac{h}{2}(f(x_0, y_0) + f(x_0 + h, y_1))$
`RK4`	$k_1 = hf(x_0, y_0)$
	$k_2 = hf(x_0 + h/2, y_0 + k_1/2)$
	$k_3 = hf(x_0 + h/2, y_0 + k_2/2)$
	$k_4 = hf(x_0 + h, y_0 + k_3)$
	$Y = y_0 + \dfrac{k_1 + 2k_2 + 2k_3 + k_4}{6}$

Maple code for Euler and Heun methods

```
# Euler algorithm
# Group 1, initialize.
f:=(x,y)->-y+1-x:
x0:=0:y0:=3:h:=0.1:L:=[x0,y0]:

# Group 2, repeat 10 times
Y:=y0+h*f(x0,y0):
euler-algorithm.pdf
x0:=x0+h:y0:=Y:L:=L,[x0,y0];


# Group 3, plot.
plot([L]);
```

```
# Heun algorithm
# Group 1, initialize.
f:=(x,y)->-y+1-x:
x0:=0:y0:=3:h:=0.1:L:=[x0,y0]:

# Group 2, repeat 10 times
Y:=y0+h*f(x0,y0):
Y:=y0+h*(f(x0,y0)+f(x0+h,Y))/2:
x0:=x0+h:y0:=Y:L:=L,[x0,y0];

# Group 3, plot.
plot([L]);
```

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/euler-algorithm.pdf

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/heun-algorithm.pdf

Maple code for Heun and RK4 methods

```
# Heun algorithm
# Group 1, initialize.
f:=(x,y)->-y+1-x:
x0:=0:y0:=3:h:=0.1:L:=[x0,y0]:

# Group 2, repeat 10 times
Y:=y0+h*f(x0,y0):
Y:=y0+h*(f(x0,y0)+f(x0+h,Y))/2:
x0:=x0+h:y0:=Y:L:=L,[x0,y0];

# Group 3, plot.
plot([L]);
```

```
# RK4 algorithm
# Group 1, initialize.
f:=(x,y)->-y+1-x:
x0:=0:y0:=3:h:=0.1:L:=[x0,y0]:

# Group 2, repeat 10 times.
k1:=h*f(x0,y0):
k2:=h*f(x0+h/2,y0+k1/2):
k3:=h*f(x0+h/2,y0+k2/2):
k4:=h*f(x0+h,y0+k3):
Y:=y0+(k1+2*k2+2*k3+k4)/6:
x0:=x0+h:y0:=Y:L:=L,[x0,y0];

# Group 3, plot.
plot([L]);
```

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/heun-algorithm.pdf

http://www.math.utah.edu/~gustafso/s2016/2280/pdf/rk4-algorithm.pdf

Numerical Algorithms: Planar Case

Notation. Let t_0, x_0, y_0 denote the entries of the dot table on a particular line. Let h be the increment for the dot table and let $t_0 + h$, x, y stand for the dot table entries on the next line.

Planar Euler Method

$$\begin{aligned} x &= x_0 + hf(t_0, x_0, y_0), \\ y &= y_0 + hg(t_0, x_0, y_0). \end{aligned}$$

Planar Heun Method

$$\begin{aligned} x_1 &= x_0 + hf(t_0, x_0, y_0), \\ y_1 &= y_0 + hg(t_0, x_0, y_0), \\ x &= x_0 + h(f(t_0, x_0, y_0) + f(t_0 + h, x_1, y_1))/2 \\ y &= y_0 + h(g(t_0, x_0, y_0) + g(t_0 + h, x_1, y_1))/2. \end{aligned}$$

Planar RK4 Method

$$
\begin{aligned}
k_1 &= hf(t_0, x_0, y_0), \\
m_1 &= hg(t_0, x_0, y_0), \\
k_2 &= hf(t_0 + h/2, x_0 + k_1/2, y_0 + m_1/2), \\
m_2 &= hg(t_0 + h/2, x_0 + k_1/2, y_0 + m_1/2), \\
k_3 &= hf(t_0 + h/2, x_0 + k_2/2, y_0 + m_2/2), \\
m_3 &= hg(t_0 + h/2, x_0 + k_2/2, y_0 + m_2/2), \\
k_4 &= hf(t_0 + h, x_0 + k_3, y_0 + m_3), \\
m_4 &= hg(t_0 + h, x_0 + k_3, y_0 + m_3), \\
x &= x_0 + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\
y &= y_0 + \frac{1}{6}\left(m_1 + 2m_2 + 2m_3 + m_4\right).
\end{aligned}
$$

Numerical Algorithms: General Case

Consider a vector initial value problem

$$\vec{\mathbf{u}}'(t) = \vec{\mathbf{F}}(t, \vec{\mathbf{u}}(t)), \quad \vec{\mathbf{u}}(t_0) = \vec{\mathbf{u}}_0.$$

Vector Euler Method

$$\vec{\mathbf{u}} = \vec{\mathbf{u}}_0 + h\vec{\mathbf{F}}(t_0, \vec{\mathbf{u}}_0)$$

Vector Heun Method

$$\vec{\mathbf{w}} = \vec{\mathbf{u}}_0 + h\vec{\mathbf{F}}(t_0, \vec{\mathbf{u}}_0),$$

$$\vec{\mathbf{u}} = \vec{\mathbf{u}}_0 + \frac{h}{2}\left(\vec{\mathbf{F}}(t_0, \vec{\mathbf{u}}_0) + \vec{\mathbf{F}}(t_0 + h, \vec{\mathbf{w}})\right)$$

Vector RK4 Method

$$\begin{aligned}
\vec{\mathbf{k}}_1 &= h\vec{\mathbf{F}}(t_0, \vec{\mathbf{u}}_0), \\
\vec{\mathbf{k}}_2 &= h\vec{\mathbf{F}}(t_0 + h/2, \vec{\mathbf{u}}_0 + \vec{\mathbf{k}}_1/2), \\
\vec{\mathbf{k}}_3 &= h\vec{\mathbf{F}}(t_0 + h/2, \vec{\mathbf{u}}_0 + \vec{\mathbf{k}}_2/2), \\
\vec{\mathbf{k}}_4 &= h\vec{\mathbf{F}}(t_0 + h, \vec{\mathbf{u}}_0 + \vec{\mathbf{k}}_3), \\
\vec{\mathbf{u}} &= \vec{\mathbf{u}}_0 + \frac{1}{6}\left(\vec{\mathbf{k}}_1 + 2\vec{\mathbf{k}}_2 + 2\vec{\mathbf{k}}_3 + \vec{\mathbf{k}}_4\right).
\end{aligned}$$