Approximating Solutions of Differential Equations

Semi-analitical solutions

By a semi-analitical method we understand the method when an exact solution $y(x)$ is approximated by another function $\bar{y}(x)$, We present two methods: **Picard Iteration Method** and **Taylor Series Method**.

Iteration method

Let consider the following IVP

$$\begin{cases} y' = f(x,y) \\ y(x_0) = y^0 \end{cases}$$

which is equivalent with the following Volterra integral equation

$$y(x) = y^0 + \int_{x_0}^{x} f(s, y(s))ds$$

Starting from this integral equation we can construct the Picard sequence (succesive approximation sequence)

$$y_{n+1}(x) = y^0 + \int_{x_0}^{x} f(s, y_n(s))ds$$

for a starting function $y_0(x)$ choosen. The n-th interate will be the approximating solution.

Example 1.

$$\begin{cases} y' = y \\ y(0) = 1 \end{cases}$$

The Picard sequence has the following form in this case is

$$y_{n+1}(x) = \int_{x_0}^{x} (y_n(s))ds$$

Let's calculate the Picard sequence for $y_0(x) \equiv 1$

```
> restart:
> f:=(x,y)->y;
```

$$f := (x, y) \to y$$

```
> y[0]:=x->1;
```

$$y_0 := 1$$

```
> n:=5;
```

$$n := 5$$

```
>
> for i from 1 to n do
y[i]:=unapply(simplify((1+int(f(s,y[i-1](s)),s=0..x))),x)
od;
```

$$y_1 := x \to 1 + x$$

$$y_2 := x \to 1 + x + \frac{1}{2}x^2$$

$$y_3 := x \to 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3$$

$$y_4 := x \to 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4$$

$$y_5 := x \to 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5$$

Therefore the n -th iterate is an approximation of exact solution.
Let's calculate the exact solution

```
> diff_eq:=diff(yy(x),x)=yy(x);
```

$$diff\_eq := \frac{d}{dx}\,yy(x) = yy(x)$$

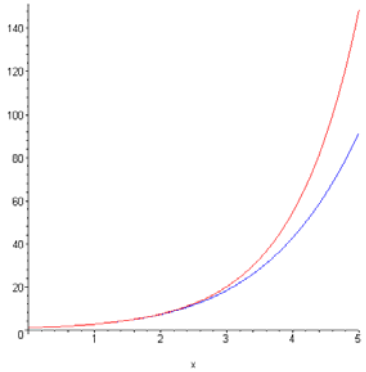
```
> in_c:=yy(0)=1;
```

$$in\_c := yy(0) = 1$$

```
> y_ex:=unapply(rhs(dsolve({diff_eq,in_c},yy(x))),x);
```

$$y\_ex := \exp$$

```
> plot([y_ex(x),y[n](x)],x=0..5,color=[red,blue]);
```



In the graph we can see the difference between the exact solution **y_ex(x)** and the n-th interate, **y[n](x)**.

Taylor Series Method

This method consist in finding an approximating solution as a Taylor expansion around the point x_0

$$\bar{y}(x) = y(x_0) + \frac{y'(x_0)}{1!}(x - x_0) + \ldots + \frac{y^{(n)}(x_0)}{n!}(x - x_0)^n$$

thus we need to find the values of the derivatives of the unkown function y in x_0, for this we will use the differential equation.
We denote by

$$a_j = \frac{y^{(j)}(x_0)}{j!}, \quad j = 0 .. n$$

```
> restart;
```

The procedure **derivs** computes the derivatives of the function $f(x, y(x))$ with respect to x in whichiti is made the substitution $y'(x) = f(x, y(x))$ and it has the inputs the function f and the derivation order n .

```
> derivs:=proc(f,n)
option remember;
if n=1 then f(x,y(x))
else
  subs(diff(y(x),x)=f(x,y(x)),diff(derivs(f,n-1),x))
fi;
end:
```

The procedure **coef** computes the values of the coefficients a_j , j=0..n. The inputs for this procedure are the function f , the order n of the Taylor polynomial, the point x_0 and the value y^0 .

```
> coef:=proc(f,n,x0,y0)
 local d;
if n=0 then y0
else
 d:=unapply(derivs(f,n),x):
 subs(y(x0)=y0,d(x0))
fi;
end:
```

Let's apply this method for the IVP for Example1

```
> f:=(x,y)->y;
```

$$f := (x, y) \to y$$

```
> x0:=0;y0:=1;
```

$$x0 := 0$$

$$y0 := 1$$

```
> n:=5;
```

$$n := 5$$

```
> for j from 0 to n do
 a[j]:=1/j!*coef(f,j,x0,y0)
od;
```

$$a_0 := 1$$

$$a_1 := 1$$

$$a_2 := \frac{1}{2}$$

$$a_3 := \frac{1}{6}$$

$$a_4 := \frac{1}{24}$$

$$a_5 := \frac{1}{120}$$

```
> y1:=sum('a[j]*(x-x0)^j','j'=0..n);
```

$$y1 := 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5$$

```
> y_app:=unapply(y1,x);
```

$$y\_app := x \to 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5$$

Therefore the **y_app** is an approximation of the exact solution.
Let's calculate the exact solution

```
> d_eq:=diff(yy(x),x)=yy(x);
```

$$d\_eq := \frac{d}{dx}\mathrm{yy}(x) = \mathrm{yy}(x)$$

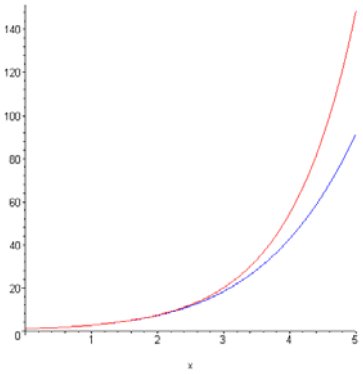
```
> in_c:=yy(0)=1;
```

$$in\_c := \mathrm{yy}(0) = 1$$

```
> y_ex:=unapply(rhs(dsolve({d_eq,in_c},yy(x))),x);
```

$$y\_ex := \exp$$

```
> plot([y_ex(x),y_app(x)],x=0..5,color=[red,blue]);
```



Numerical methods

A numerical method approximate the value of exact solution y(x) in the points $a = x_0, \ldots, x_n = b$ from the interval [a;b] with $y_0, \ldots, y_n$

$y(x_k) \approx y_k$, $k = 0 \,..\, n$

Euler method

In Euler method y_k are calculated using the formula

$y_{k+1} = y_k + h \cdot f(x_k, y_k)$

where h is the step

$$h = \frac{b-a}{n}$$

$x_{k+1} = x_k + h$

Let's apply this method for our Example 1 on interval [0;5]. First we calculate the exact solution

```
> restart;
> d_eq:=diff(yy(x),x)=yy(x);
```

$$d\_eq := \frac{d}{dx}\,\mathrm{yy}(x) = \mathrm{yy}(x)$$

```
> in_c:=yy(0)=1;
```

$$in\_c := \mathrm{yy}(0) = 1$$

```
> y_ex:=unapply(rhs(dsolve({d_eq,in_c},yy(x))),x);
```

$$y\_ex := \exp$$

```
> x0:=0;y0:=1;a:=0;b:=5;n:=10;
```

$$x0 := 0$$

$$y0 := 1$$

$$a := 0$$

$$b := 5$$

$$n := 10$$

```
> f:=(x,y)->y;
```

$$f := (x, y) \to y$$

```
> h:=evalf((b-a)/n);
```

$$h := 0.5000000000$$

```
> x[0]:=x0;y[0]:=y0;
```

$$x_0 := 0$$

$$y_0 := 1$$

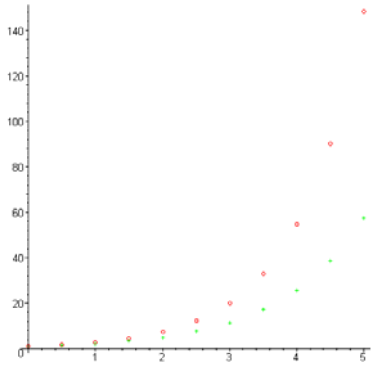
```
> for k from 0 to n-1 do
 x[k+1]:=evalf(x[k]+h):
 y[k+1]:=evalf(y[k]+h*f(x[k],y[k])):
od:
> exact_val:=[ x[j], y_ex(x[j])]$j=0..n;
```

$exact\_val := [0, 1], [0.5000000000, 1.648721271], [1.000000000, 2.718281828],$
$[1.500000000, 4.481689070], [2.000000000, 7.389056099],$
$[2.500000000, 12.18249396], [3.000000000, 20.08553692],$
$[3.500000000, 33.11545196], [4.000000000, 54.59815003],$
$[4.500000000, 90.01713130], [5.000000000, 148.4131591]$

```
> app_val:=[ x[j], y[j]]$j=0..n;;
```

$app\_val := [0, 1], [0.5000000000, 1.500000000], [1.000000000, 2.250000000],$
$[1.500000000, 3.375000000], [2.000000000, 5.062500000],$
$[2.500000000, 7.593750000], [3.000000000, 11.39062500],$
$[3.500000000, 17.08593750], [4.000000000, 25.62890625],$
$[4.500000000, 38.44335937], [5.000000000, 57.66503905]$

```
> plot([[exact_val],[app_val]],style=point,symbol=[circle,cross]);
```



Exercise: *Try to increase* n *, for example take* $n = 100$.

Runge-Kutta Method

We will implement Runge-Kutta method of order 4. The sequence y_k is calculated using the formula

$$y_{k+1} = y_k + h \cdot \left(\frac{1}{6} K_1^k + \frac{2}{6} K_2^k + \frac{2}{6} K_3^k + \frac{1}{6} K_4^k \right)$$

where

$$K_1^k = f(x_k, y_k)$$

$$K_2^k = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} K_1^k \right)$$

$$K_3^k = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} K_2^k \right)$$

$$K_4^k = f\left(x_k + h, y_k + h \cdot K_3^k \right)$$

```
> restart;
> d_eq:=diff(yy(x),x)=yy(x);
```

$$d\_eq := \frac{d}{dx} \mathrm{yy}(x) = \mathrm{yy}(x)$$

```
> in_c:=yy(0)=1;
```

$$in\_c := \mathrm{yy}(0) = 1$$

```
> y_ex:=unapply(rhs(dsolve({d_eq,in_c},yy(x))),x);
```

$$y\_ex := \exp$$

```
> x0:=0;y0:=1;a:=0;b:=5;n:=10;
```

$$x0 := 0$$

$$y0 := 1$$

$$a := 0$$

$$b := 5$$

$$n := 10$$

```
> f:=(x,y)->y;
```

$$f := (x, y) \to y$$

```
> h:=evalf((b-a)/n);
```

$$h := 0.5000000000$$

```
> x[0]:=x0;y[0]:=y0;
```

$$x_0 := 0$$

$$y_0 := 1$$

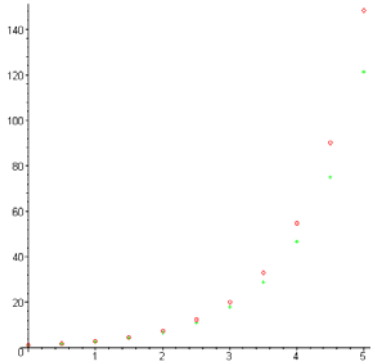
```
> for k from 0 to n-1 do
 x[k+1]:=evalf(x[k]+h):
 K1:=f(x[k],y[k]):
 K2:=f(x[k]+h/2,y[k]+h/2*K1):
 K3:=f(x[k]+h/2,y[k]+h/2*K2/2):
 K4:=f(x[k]+h,y[k]+h*K3):
 y[k+1]:=evalf(y[k]+h*(1/6*K1+2/6*K2+2/6*K3+1/6*K4)):
od:
> exact_val:=[ x[j], y_ex(x[j])]$j=0..n;
```

exact_val := [0, 1], [0.5000000000, 1.648721271], [1.000000000, 2.718281828],
[1.500000000, 4.481689070], [2.000000000, 7.389056099],
[2.500000000, 12.18249396], [3.000000000, 20.08553692],
[3.500000000, 33.11545196], [4.000000000, 54.59815003],
[4.500000000, 90.01713130], [5.000000000, 148.4131591]

```
> app_val:=[ x[j], y[j]]$j=0..n;;
```

app_val := [0, 1], [0.5000000000, 1.615885416], [1.000000000, 2.611085679],
[1.500000000, 4.219215270], [2.000000000, 6.817768424],
[2.500000000, 11.01673257], [3.000000000, 17.80177750],
[3.500000000, 28.76563266], [4.000000000, 46.48196631],
[4.500000000, 75.10953149], [5.000000000, 121.3683966]

```
> plot([[exact_val],[app_val]],style=point,symbol=[circle,cross]);
```



The Second-Order Boundary Value Problem

```
> restart;with(DEtools):
```

The second-order boundary value problem (BVP) consists of the second-order ODE

$$F(y(x), y'(x), y''(x), x) = 0$$

and appropriate conditions at the two ends of an interval [a, b]. Foe the second order differential equation we can consider different types of boundary conditions, such as

- nonhomogeneous *Dirichlet* conditions would be

$$y(a) = A,\ y(b) = B$$

- nonhomogeneous *Neumann* conditions would be

$$y'(a) = \alpha\ ,\ y'(b) = \beta$$

- nonhomogeneous *Robin* conditions would be

$$y(a) + \kappa_a\ y'(a) = \lambda_a$$

$$y(b) + \kappa_b\ y'(b) = \lambda_b$$

- most general linear boundary conditions for a second-order BVP would be

$$a_{11}\ y(a) + a_{12}\ y'(a) + b_{11}\ y(b) + b_{12}\ y'(b) = B_1$$

$$a_{21}\ y(a) + a_{22}\ y'(a) + b_{21}\ y(b) + b_{22}\ y'(b) = B_2$$

Not every BVP has a solution, and not every BVP that has a solution has a unique solution. Hence, a BVP can have no solution, one solution, or an infinite number of solutions.

The Shooting Method

The basic idea behind the shooting method is to convert a boundary value problem (BVP) into an initial value problem (IVP). It is easily motivated by an examination of a field of solutions that satisfy the left-hand boundary condition. For example, let's consider the following BVP

$$\left(\frac{d^2}{dx^2}\,y(x)\right) + 4\left(\frac{d}{dx}\,y(x)\right) + 4\,y(x) = 9\,\mathbf{e}^x$$

$$y(0) = 2$$

$$y(\ln(2)) = 0$$

The exact solution can be found using **dsolve**:

```
> d_eq:=diff(y(x),x$2)+4*diff(y(x),x)+4*y(x)=9*exp(x);
```

$$d\_eq := \left(\frac{d^2}{dx^2}\,y(x)\right) + 4\left(\frac{d}{dx}\,y(x)\right) + 4\,y(x) = 9\,\mathbf{e}^x$$

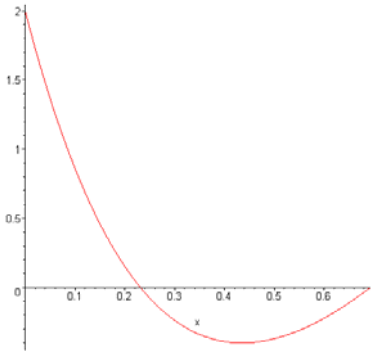
```
> bc:=y(0)=2,y(ln(2))=0;
```

$$bc := y(0) = 2,\ y(\ln(2)) = 0$$

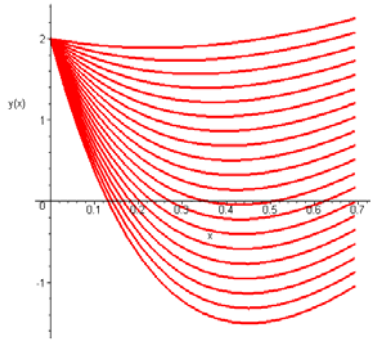
```
> y_sol:=unapply(rhs(dsolve({d_eq,bc},y(x))),x);
```

$$y\_sol := x \rightarrow \mathbf{e}^{(-2x)} - \frac{9\,\mathbf{e}^{(-2x)}\,x}{\ln(2)} + \mathbf{e}^x$$

```
> plot(y_sol(x),x=0..ln(2));
```



```
> DEplot(d_eq, y(x), x=0..ln(2), [seq([y(0)=2,D(y)(0)=-k],k=1..20)],
arrows=none, linecolor=red);
```



Not every solution starting at (0, 2) passes through (ln(2), 0). In fact, only one solution passes through both points. The other solutions have different slopes at x = 0. Just the one solution of the BVP passing through both points has the "right" initial slope to rise from (0, 2) and fall to (ln(2), 0). Hence, the solution of the BVP must be "launched" from the initial point with the correct slope in order for it to reach the target point at the right end of the interval. The similarity to launching a projectile leads to the name *Shooting Method* for the numeric solution of the BVP.

The shooting method is quite general. Here it will be demonstrated only for two second-order linear two-point boundary value problems of the form

$$\left(\frac{d^2}{dx^2}\,y(x)\right) + p(x)\left(\frac{d}{dx}\,y(x)\right) + q(x)\,y(x) = f(x)$$
$$y(a) = r$$
$$y(b) = s$$

The IVP that will be used as the basis for the shooting method is

$$\left(\frac{d^2}{dx^2}\,y(x)\right) + p(x)\left(\frac{d}{dx}\,y(x)\right) + q(x)\,y(x) = f(x)$$
$$y(a) = r$$
$$y'(a) = \alpha$$

where α is the unknown parameter. Call the solution to this IVP y^{α}. The goal is to find α such that $y^{\alpha}(b) = s$

```
> a:=0;b:=ln(2);
```

$$a := 0$$

$$b := \ln(2)$$

```
> ic1 := y(a)=2, D(y)(a)=alpha;
```

$$ic1 := y(0) = 2, D(y)(0) = \alpha$$

```
> y_alpha:=unapply(rhs(dsolve({d_eq,ic1},y(x))),x,alpha);
```

$$y\_alpha := (x, \alpha) \rightarrow \mathbf{e}^{(-2x)} + \mathbf{e}^{(-2x)}\, x\, (\alpha + 1) + \mathbf{e}^{x}$$

Now, we have to find the value of α such that $y\_alpha(\ln(2), \alpha) = 0$.

```
> eq:=y_alpha(ln(2),alpha) = 0;
```

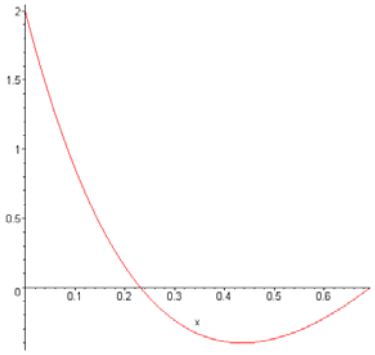
$$eq := \frac{9}{4} + \frac{1}{4}\ln(2)\,(1 + \alpha) = 0$$

```
> alpha1:=solve(eq,alpha);
```

$$\alpha 1 := -\frac{9 + \ln(2)}{\ln(2)}$$

Let's compare the exact solution and the approximating solution from the shooting method

```
> plot([y_sol(x),y_alpha(x,alpha1)],x=a..b,color=[red,blue]);
```



In this case, we found the exact slope for the shooting solution such that it coincides with the exact solution of the BVP. Not always the algebraic equation in α is so simple (in this case is linear), in general, this equation is nonlinear, so, we have to solve numericaly this equation and the solution will be an approximating value for the good slope, therefore appears some differences between the shooting solution and the exact solution of the BVP.