NOTES_ODE

April 10, 2019

TF2202 Teknik Komputasi - Persamaan Diferensial Biasa
Fadjar Fathurrahman

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
```

```
In [2]: from IPython.display import set_matplotlib_formats
        set_matplotlib_formats("svg")
        %matplotlib inline
```

```
In [3]: import matplotlib
        matplotlib.style.use("dark_background")
```

```
In [4]: import scipy.integrate
```

Dalam notebook ini akan diberikan implementasi sederhana dari beberapa metode yang dapat digunakan untuk menyelesaikan persamaan diferensial biasa.

Bentuk umum dari persamaan diferensial biasa orde 1 adalah:

$$y' = \frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$$

Solusi dari persamaan ini memiliki satu konstanta sembarang. Konstanta ini dapat ditentukan nilainya jika diberikan nilai y pada suatu titik, misalnya $y(a) = \alpha$.

Persamaan diferensial biasa orde n dapat dituliskan sebagai:

$$y^{(n)} = \frac{\mathrm{d}^n y}{\mathrm{d}x^n} = f\left(x, y, y', \ldots, y^{(n-1)}\right)$$

Dengan menggunakan notasi $y_0 = y$, $y_1 = y'$, $y_2 = y''$, ..., $y_{n-1} = y^{(n-1)}$, persamaan diferensial biasa orde n dapat diubah menjadi persamaan diferensial oder satu sebagai berikut:

$$\begin{aligned}
y_0' &= y_1 \\
y_1' &= y_2 \\
y_2' &= y_3 \\
\cdots &= \cdots \\
y_n' &= f\left(x, y_0, y_1, \ldots, y_{n-1}\right)
\end{aligned}$$

1 Metode Euler

In [38]:
```
def ode_euler(f, xi, xf, y0, N, verbose=False):

    h = (xf - xi)/N

    if verbose:
        print("ode_euler h = %18.10f\n" % (h))

    xsol = xi + np.arange(0,N+1)*h

    # orde dari ODE ditentukan dari jumlah syarat awal yang diberikan
    Ndim = len(y0)

    # array untuk solusi
    ysol = np.zeros((N+1,Ndim))

    # aplikasi syarat awal
    ysol[0,:] = y0[:]

    for i in range(N):
        ysol[i+1,:] = ysol[i,:] + h*f(xsol[i], ysol[i,:])

    return xsol, ysol
```

1.0.1 Contoh 1

Cari solusi dari persamaan diferensial:

$$y' + 4y = x^2$$

pada rentang x $[0, 1]$ dengan syarat awal $y(0) = 1$. Bandingkan hasil yang diperoleh dengan solusi analitik:

$$y = \frac{31}{32}e^{-4x} + \frac{1}{4}x^2 - \frac{1}{8}x + \frac{1}{32}$$

Ubah persamaan menjadi bentuk standar $y' = f(x, y)$

$$y' = x^2 - 4y$$

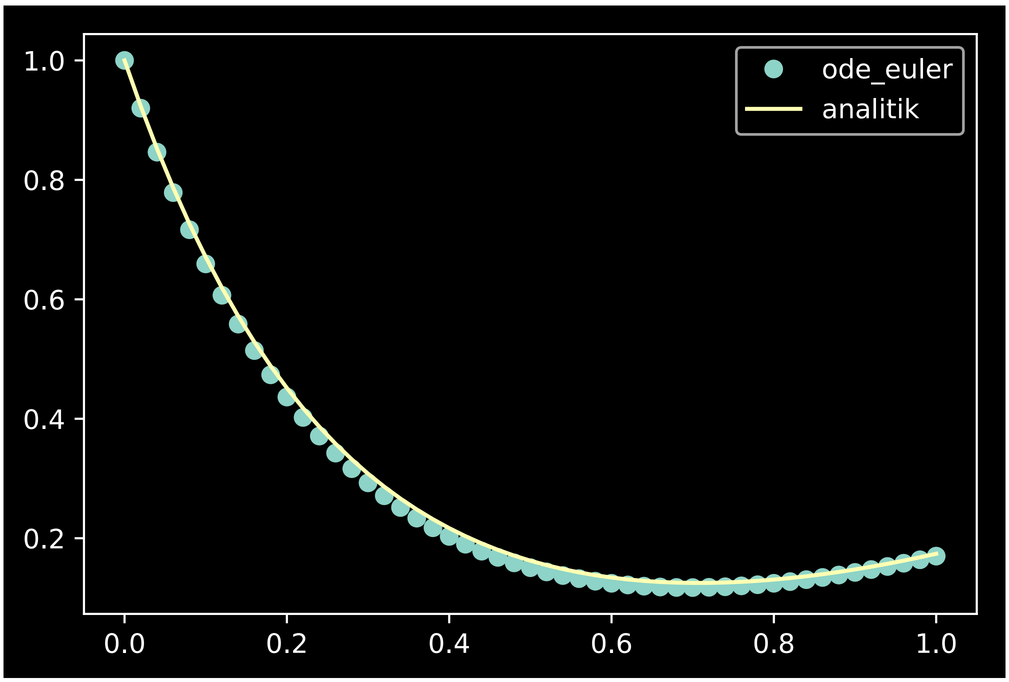
sehingga $f(x, y) = x^2 - 4y$

In [77]:
```
def my_ode_problem_01(x,y):
    return x**2 - 4*y

xi = 0.0
xf = 1.0
y0 = [1.0]  # syarat batas dalam bentuk list
Ninterval = 50
xsol, ysol = ode_euler(my_ode_problem_01, xi, xf, y0, Ninterval, verbose=True)
```

```
ode_euler h =       0.0200000000
```

```
In [79]: def analytic_sol_01(x):
             return (31.0/32.0)*np.exp(-4*x) + x**2/4 - x/8 + 1.0/32.0
```

```
In [81]: plt.clf()
         plt.plot(xsol, ysol[:,0], label="ode_euler", marker="o", linewidth=0)
         plt.plot(xsol, analytic_sol_01(xsol), label="analitik")
         plt.legend();
```



1.0.2 Contoh 2

Selesaikan persamaan diferensial berikut:

$$y'' = -0.1y' - x$$

pada $x = 0$ sampai dengan 2 dengan syarat awal $y(0) = 0$ dan $y'(0) = 1$ dengan menggunakan ukuran langkah $h = 0.05$.

Bandingkan dengan solusi analitik:

$$y = 100x - 5x^2 + 990\left(e^{-0.1x} - 1\right)$$

```
In [70]: def my_ode_problem_02(x, y):
             f = np.zeros(2)
```

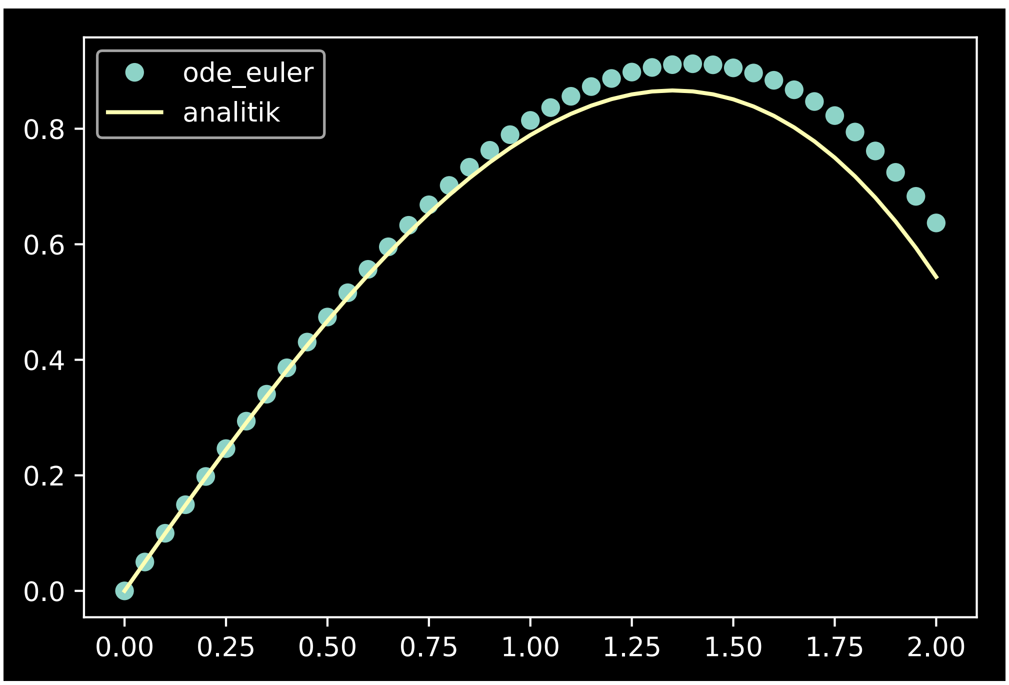
```
    f[0] = y[1]
    f[1] = -0.1*y[1] - x
    return f
```

```
In [71]: def analytic_sol_02(x):
             return 100*x - 5*x**2 + 990*(np.exp(-0.1*x) -1)
```

```
In [72]: xi = 0.0
         xf = 2.0
         y0 = [0.0, 1.0]
         h = 0.05
         Ninterval = int( (xf - xi)/h )
         xsol, ysol = ode_euler(my_ode_problem_02, xi, xf, y0, Ninterval, verbose=True)
```

```
ode_euler h =       0.0500000000
```

```
In [73]: plt.clf()
         plt.plot(xsol, ysol[:,0], label="ode_euler", marker="o", linewidth=0)
         plt.plot(xsol, analytic_sol_02(xsol), label="analitik")
         plt.legend();
```



2 Metode Runge-Kutta orde 4

```
In [69]: def ode_RK4(f, xi, xf, y0, N, verbose=False):
```

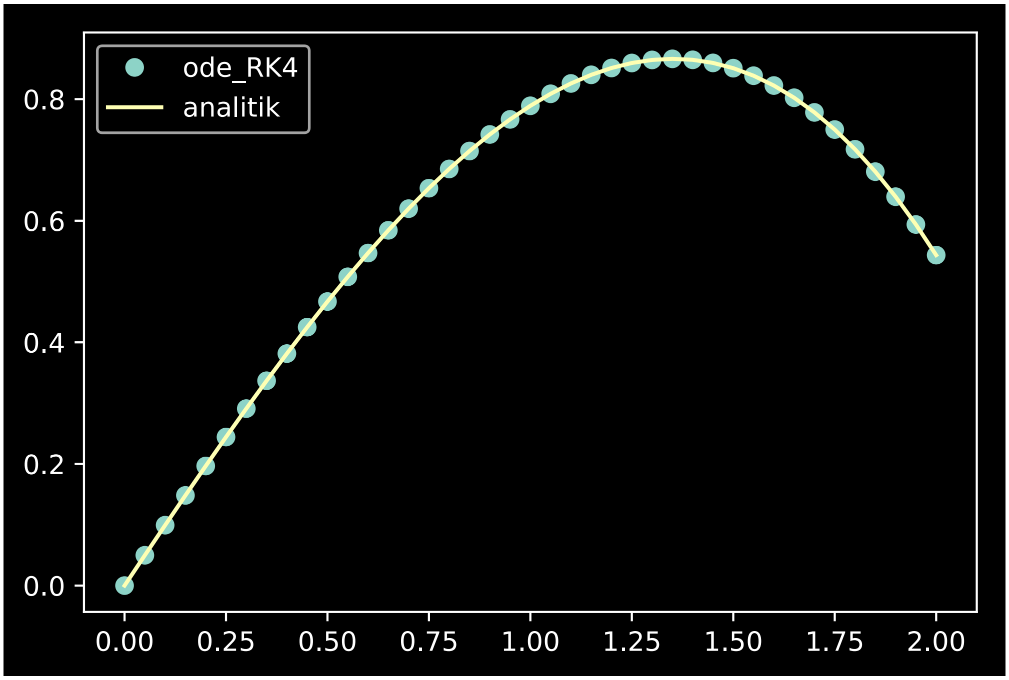
```
    h = (xf - xi)/N

    if verbose:
        print("ode_RK4 h = %18.10f\n" % (h))

    xsol = xi + np.arange(0,N+1)*h

    # orde dari ODE ditentukan dari jumlah syarat awal yang diberikan
    Ndim = len(y0)

    # array untuk solusi
    ysol = np.zeros((N+1,Ndim))

    # aplikasi syarat awal
    ysol[0,:] = y0[:]

    for i in range(N):
        f1 = h*f( xsol[i], ysol[i,:] )
        f2 = h*f( xsol[i] + h/2, ysol[i,:] + f1/2 )
        f3 = h*f( xsol[i] + h/2, ysol[i,:] + f2/2 )
        f4 = h*f( xsol[i] + h, ysol[i,:] + f3 )
        ysol[i+1,:] = ysol[i,:] + (f1 + 2*(f2 + f3) + f4)/6

    return xsol, ysol
```

2.0.1 Contoh

In [74]:
```
xi = 0.0
xf = 2.0
y0 = [0.0, 1.0]
h = 0.05
Ninterval = int( (xf - xi)/h )
xsol, ysol = ode_RK4(my_ode_problem_02, xi, xf, y0, Ninterval, verbose=True)
```

```
ode_RK4 h =       0.0500000000
```

In [75]:
```
plt.clf()
plt.plot(xsol, ysol[:,0], label="ode_RK4", marker="o", linewidth=0)
plt.plot(xsol, analytic_sol_02(xsol), label="analitik")
plt.legend();
```


ode_RK4
analitik
0.0
0.2
0.4
0.6
0.8
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00