

1. Η `sp.dsolve()` εντολή Python χρησιμοποιείται για να λύσει Συνήθειες Δ.Ε. *ordinary differential equations (ODEs)* χρησιμοποιώντας την *sympy library* της Python για συμβολικά Μαθηματικά. Η συνάρτηση `dsolve()` είναι ένα πανίσχυρο εργαλείο για επίλυση ODEs και μπορεί να επιλύσει ένα μεγάλο εύρος Δ.Ε..

Όμως:

Η `dsolve()` χρησιμοποιείται για να παρέχει συμβολικές (γενικές) λύσεις και δεν επιλύει αριθμητικά την Δ.Ε. Για Αριθμητικές επιλύσεις θα πρέπει να χρησιμοποιήσετε την **`scipy.integrate`**

Αν η `dsolve()` δεν μπορεί να βρει μια λύση κλειστής μορφής θα επιστρέψει ένα αποτέλεσμα χωρίς τιμή και θα πρέπει να χρησιμοποιήσετε αριθμητικές μεθόδους.

`sp.dsolve()`

**`sp.dsolve(eq, func, ics)`**

- **eq**: Η Δ.Ε. προς επίλυση  $y(x)$  Πρέπει να είναι μια έκφραση *sympy* που αναφέρεται σε μια ODE.
- **func** (optional): Η συνάρτηση της ODE, που αντιστοιχεί στην  $y(x)$ . Αυτή είναι απαραίτητη αν η ODE είναι συγκεκριμένη συνάρτηση. Αν την παραβλέψω η `dsolve()` προσπαθεί να ανιχνεύσει την συνάρτηση.
- **ics** (optional): Αρχικές συνθήκες για επίλυση της ODE. Παράδειγμα,  $\{y(0): 1\}$  που δείχνει ότι  $y(0) = 1$ . Αν δεν υπάρχουν αρχικές συνθήκες η `dsolve()` επιστρέφει μια γενική λύση.

The Python command `eq = sp.Eq(y(x).diff(x), y(x))` uses the SymPy library in Python to create an equation involving a function and its derivative

Πχ Λύσε την  $dy/dx=y$

```
import sympy as sp
# Ορισμός συμβόλων
x = sp.symbols('x')
y = sp.Function('y')
# Ορισμός ODE Εξίσωσης
eq = sp.Eq(y(x).diff(x), y(x))
# Λύση ODE
solution = sp.dsolve(eq)
print(solution)
```

**Αποτέλεσμα!**

Eq(y(x), C1\*exp(x)) Γενική λύση  $y(x)=C_1 \cdot e^x$  όπου C1 σταθερά.

**Ας λύσω την ίδια με αρχική συνθήκη  $y(0)=1$ :!!!!!!!!!!!!**

```
# Λύση ODE
import sympy as sp
# Ορισμός συμβόλων
x = sp.symbols('x')
y = sp.Function('y')
# Ορισμός ODE Εξίσωσης
eq = sp.Eq(y(x).diff(x), y(x))
# Λύση ODE με Αρχικές συνθήκες όπου η σταθερά C =1
ics={y(0):1}
solution = sp.dsolve(eq, ics=ics)
```

```
print(solution)
```

**Output:**

$\text{Eq}(y(x), \exp(x))$

$y(x) = e^x$

## Επίλυση Διαφορικής Εξίσωσης ΔΕΥΤΕΡΟΥ ΒΑΘΜΟΥ

$$\underline{d^2y/dx^2 + y = 0}$$

Κώδικας Python

```
import sympy as sp
```

```
x = sp.symbols('x')
```

```
y = sp.Function('y')
```

```
# Ορισμός ODE Εξίσωσης
```

```
eq2 = sp.Eq(y(x).diff(x, x) + y(x), 0)
```

```
# Επίλυση της ODE
```

```
solution2 = sp.dsolve(eq2)
```

```
print(solution2)
```

**Αποτέλεσμα!**

$\text{Eq}(y(x), C_1 \cos(x) + C_2 \sin(x))$

Δηλαδή η Γενική επίλυση:  $y(x) = C_1 \cos(x) + C_2 \sin(x)$ ,  
where  $C_1$  and  $C_2$  are constants of integration.

ΜΕΡΟΣ Β.

$$Y' + 3.y = x^2 \text{ και } y(0) = 1$$

```
import sympy as sp
x=sp.symbols('x')
y= sp.Func-tion('y')(x)
equation=sp.Eq(y.diff(x)+3*y, x**2)
solution=sp.solve(equation, y, ics={y.subs(x,0):1})
print(solution)
/# Βρες την τιμή του y(0.05)
y_at_0_05=solution.rhs.subs(x, 0.05)
print("y(0.05) =", y_at_0_05)
```

## ΣΧΟΛΙΟ!

***solution.rhs:*** Στην SymPy, μια εξίσωση αναπαρίσταται ως αντικείμενο Ισότητας με δύο συνιστώσες: Το *left-hand side (lhs)* και το *right-hand side (rhs)*. Κάθε πρόσβαση στο *solution.rhs* μας δίνει το *right-hand side* της εξίσωσης. Πχ αν η λύση είναι  $Eq(f(x), 0)$ , τότε το *solution.rhs* θα είναι το 0.

## Η λειτουργία `ivp`

Μπορεί να λύσει προβλήματα αρχικής τιμής και ΜΕΡΙΚΕΣ ΔΙΑΦΟΡΙΚΕΣ ΕΞΙΣΩΣΕΙΣ

Ένα Πρόβλημα αρχικής τιμής (IVP) για μια διαφορική εξίσωση περιλαμβάνει την επίλυση της διαφορικής εξίσωσης με ένα δεδομένο σύνολο αρχικών συνθηκών σε ένα συγκεκριμένο σημείο. Ο στόχος είναι να βρεθεί μια συνάρτηση που να ικανοποιεί τη διαφορική εξίσωση και επίσης να πληροί τις αρχικές συνθήκες

`solve_ivp` είναι μια συνάρτηση από το `scipy.integrate module` της Python, που λύνει προβλήματα αρχικής τιμής ... `initial value problem`". Είναι ένα ευέλικτο εργαλείο που χρησιμοποιείται για την αριθμητική επίλυση συνηθισμένων διαφορικών εξισώσεων (ODEs), δεδομένων των αρχικών συνθηκών και ενός χρονικού διαστήματος.

Πχ  $dy/dx=f(x,y)$  και  $y(x_0)=y_0$  (αρχική συνθήκη)

ΒΗΜΑ 1 Λύνω την ΔΕ  $\int dx/dy \quad dx = \int 3x^2 dx = x^3 + C$

ΒΗΜΑ 2 Εφαρμόζω τις αρχικές συνθήκες

Ξέρω ότι  $y(0)=4$  Βάζω  $x=0$  και  $y(0)=4$  στη λύση:

$$4=0^3+C \quad \text{Άρα } C=4$$

ΒΗΜΑ 3 Γράφω την τελική IVP λύση

$$y(x)=x^3+4$$

## ΠΑΡΑΔΕΙΓΜΑ:

**Λύσε την 1<sup>η</sup> τάξεως ODE Με αρχικές Τιμές Initial Values (Initial conditions) Που περιγράφουν την κατάσταση σε μια αρχική στιγμή. Αρχική συνθήκη: Αυτή είναι η τιμή της συνάρτησης (ή των παραγώγων της) σε ένα συγκεκριμένο σημείο, συνήθως στο  $t=0$  ή κάποιο άλλο σημείο εκκίνησης. Θα λύσω την ODE:**

$$dy/dt = -2y + 1$$

Με αρχική συνθήκη  $y(0)=0$ , στο χρονικό διάστημα  $t=[0,5]$ .

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Ορισμός συνάρτησης ΔΕ dy/dt = -2y + 1
def model(t, y):
    return -2 * y + 1

# Αρχική συνθήκη
y0 = [0]

# Χρονικό Διάστημα για την λύση (Από t=0 to t=5)
t_span = (0, 5)

# Καθορίστε τα σημεία στα οποία θέλουμε να αξιολογήσουμε τη λύση
t_eval = np.linspace(0, 5, 100)

# Λύση της Διαφορικής Εξίσωσης
sol = solve_ivp(model, t_span, y0, t_eval=t_eval)

# Γραφική παράσταση
plt.plot(sol.t, sol.y[0], label='y(t)')
plt.xlabel('Time t')
plt.ylabel('y(t)')
plt.title('Solution to dy/dt = -2y + 1')
plt.legend()
plt.grid(True)
plt.show()
```

- **Define the ODE:** The `model` function represents the equation  $\frac{dy}{dt} = -2y + 1$ .
- **Initial Condition:** We set the initial condition as  $y(0) = 0$  with `y0 = [0]`.
- **Time Span:** We want to solve the equation from  $t = 0$  to  $t = 5$ , so we set `t_span = (0, 5)`.
- **Time Points:** We specify the points where we want to evaluate the solution with `t_eval = np.linspace(0, 5, 100)`, which gives 100 points between 0 and 5.
- **Call `solve_ivp`:** This function solves the ODE, and the solution is stored in `sol`.
- **Plot the Results:** We plot `sol.t` (time points) vs. `sol.y[0]` (solution values).