Oppdatert klasse med input og output beskrivelse

In [ ]:

```
class Analyse:
    def __init__(self, data, T):
        """
        Input: data array og total analysens tiden T
        Output: ingenting fra init, men se på hversubklasse
        """
        self.T = T
        self.N = data.shape[0]
        self.f_s = self.N/self.T
        self.dt = 1/self.f_s

        print(f"Data med {self.N} samplingspunkter er mottat...")
        print(f"Samplingsfrekvens er på: {self.f_s}Hz, data som har frekvens
        print(f"Varighet av data er på {self.T} sekunder")

        self.t = np.linspace(0,self.T,self.N)
        self.x_n = data


    def sampled_signal(self):
        """
        input: nan
        output: plot med original signal
        """
        plt.figure(figsize=([15,5]))
        plt.plot(self.t, self.x_n)
        plt.grid(True)
        plt.xlabel("tid [s]")
        plt.ylabel("amplitude")
        plt.title("Plot of sampled signal")
        plt.show()

    def fourier_transform(self, xmin, xmax):
        """
        input:
            xmin: hvorfra vil du se plottet ditt i x-akse
            xmax: til hvor vil du se plottet ditt i x-akse

        output:
            plot av Fourier transformert signal i frekvensdomenet
        """
        self.x_k = np.fft.fft(self.x_n)
        self.freq = np.fft.fftfreq(self.N, self.dt)
        plt.figure(figsize=([15,5]))
        plt.plot(self.freq, abs(self.x_k))
        plt.grid(True)
        plt.xlabel("frekvens [f]")
        plt.ylabel("amplitude")
        plt.title("Plot of fourier transform")
        plt.xlim(xmin=xmin, xmax=xmax)
        plt.show()

        return self.x_k
    def wavelet_analyse(self,tmin, tmax, fmin, fmax, steps):
        """
        input:
            tmin: fra hvilke tider vil du analysere
            tmax: til hvilke tider vil du analysere
```

```
            fmin: fra hvilke frekvenser vil du analysere
            fmax: til hvilke frekvenser vil du analysere
            steps: hvor mange steger mellom tider og frekvenser du vil analys
                   blir

            output: en 2d matrise og plott av denne matrisen
        """
        self.steps = steps
        def wavelet(f_a, K, t_k, t_n):
            C = 0.798*2*np.pi*f_a/(self.f_s*K)
            eksp1 = -1j*2*np.pi*f_a*(t_n - t_k)
            eksp2 = -K**2
            eksp3 = (-2*np.pi*f_a**2)*((t_n - t_k)**2)/(2*K)**2
            return C*(np.exp(eksp1) - np.exp(eksp2))*np.exp(eksp3)

        self.t_ = np.linspace(tmin,tmax,self.steps)
        self.f_ = np.linspace(fmin,fmax,self.steps)
        T,F = np.meshgrid(self.t_,self.f_)
        K = 10

        sum = 0
        for i in range(self.N):
            sum = sum + np.conj(self.x_n[i]*(wavelet(F, K, T, self.t[i])))
        self.Z = abs(sum)


        plt.figure(figsize=([10,7]))
        plt.contourf(T,F,self.Z)
        plt.grid(True)
        plt.colorbar()
        plt.xlabel("tid [s]")
        plt.ylabel("Frekvens [Hz]")
        plt.title("2D plot av wavelettransform")
        plt.show()

    def hastighet(self,f_o, c):
        """
        input:
            f_o: opprinnelig frekvens som kilden sender ut
            c: opprinnelig signal hastighet
        output: plot av hastighetskurven
        """
        frek = np.zeros(self.steps)
        tid = self.t_
        vel = np.zeros(self.steps)

        def v(f_k):
            return (c * ((f_k/f_o) - 1))

        for i in range(self.steps):
            index = np.where(self.Z == np.amax(self.Z[:,i]))
            monoindex = index[0]
            frek[i] = self.f_[monoindex]
            vel[i] = v(frek[i])
        plt.figure(figsize=([10,7]))
        plt.plot(tid, vel)
        plt.plot(tid, vel, 'o')
        plt.grid(True)
        plt.xlabel("tid [s]")
        plt.ylabel("Hastighet [m/s]")
        plt.title("Hastighetsplot")
        plt.show()
```