NTNU

Norwegian University of Science and Technology

Department of mechanical and industrial engineering

TMM4175 Polymer Composites

AS2-1 - Micromechanical finite element analysis of a hexagonal arrangement of carbon fibers

Authors
Alberto Da Rold
Vegard G. Jervell
Erwan Luguern
Cyriaque Mas

Supervisor
Nils-Petter Vedvik

February 26, 2021

Contents

1 Introduction

In this report, a hexagonally packed carbon fiber composite with an epoxy matrix will be studied. The effect of the volume fraction the transverse modulus of the carbon fibers on the composite will be studied and discussed. Finally, some simple micro mechanical models will be compared to the finite element analysis (FEA) and the Halpin-Tsai model will be fit to the results from the FEA.

Finite element analysis is done in Abaqus. The materials were generated using the material properties displayed in Table 1.1. The carbon fibers are assumed to be transversely isotropic, while the epoxy matrix is assumed to be isotropic. These assumptions imply that the following relations are valid.

$$
\begin{aligned}
E_{2f} &= E_{3f} & E_{ijm} &= E_m \\
\nu_{12f} &= \nu_{13f} & \nu_{ijm} &= \nu_m \\
G_{12f} &= G_{13f} & G_{ijm} &= G_m = \frac{E_m}{2(1+\nu_m)} \\
G_{23f} &= \frac{E_{2f}}{2(1+\nu_{23f})}
\end{aligned}
$$

E_{1f} [MPa]	E_{2f} [MPa]	ν_{12f} [-]	ν_{23f} [-]	G_{12f} [MPa]	E_m [MPa]	ν_m [-]
233 000	15 000	0.2	0.35	9000	4100	0.37

Table 1.1: Material properties

2 Theory

2.1 The unit cell

The area of a hexagonal cell with sides of length a, as shown in Figure 2.1 is given by

$$
\begin{aligned}
A &= 6\frac{a^2}{2}\sin\left(\frac{\sqrt{3}}{2}\right) \\
A &= \frac{3\sqrt{3}a^2}{2}
\end{aligned}
\tag{2.1}
$$

The area of fibers in the unit cell is given by

$$A_f = 3\pi r_f^2 \tag{2.2}$$

Thereby the volume fraction of fibers in the composite can be related to the unit cell parameter a and the radius of the fibers by

$$V_f = \frac{A_f}{A} = \frac{2\pi r_f^2}{\sqrt{3}a^2}. \tag{2.3}$$

The dimensions of the tetragonal unit cell can be related to the dimensions of the hexagonal cell by

$$
\begin{aligned}
a_2 &= 2a\cos\left(\frac{\pi}{6}\right) = a\sqrt{3} \\
a_3 &= a
\end{aligned}
\tag{2.4}
$$

For all simulations, the parameter a was kept constant and equal to unity, while the fiber radius and unit cell depth was varied.

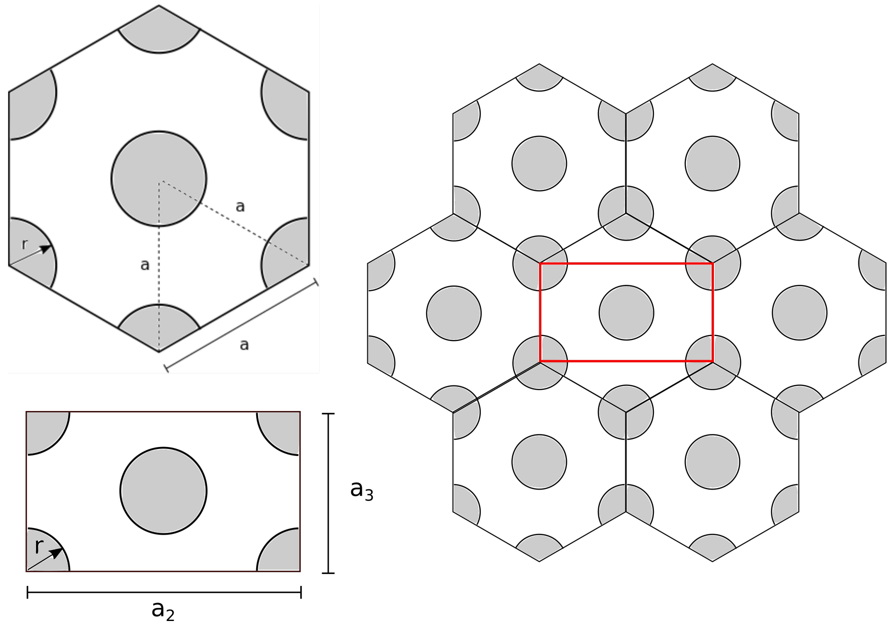

Figure 2.1: Unit cell of a hexagonal fiber composite

2.2 Simple micro-mechanical models

When a load is applied to a UDFC in the fiber direction, hereby denoted the 1 direction, the fiber strain will be equal to the matrix strain which will equal the composite strain. Additionally, stress on the fibers can be assumed to be proportional to the area fraction of the fibers, which is equal to the volume fraction of the fibers. From this, an expression relating the material moduli and volume fractions to the composite modulus can be derived.

$$\begin{aligned} \sigma_1 &= V_f \sigma_{f,1} + V_m \sigma_{m,1} \\ E_1 \epsilon_1 &= V_f E_{1,f} \epsilon_{1,f} + V_m E_{m,1} \epsilon_{m,1} \\ E_1 &= V_f E_{1,f} + V_m E_{1,m} \end{aligned} \tag{2.5}$$

Conversely, for a load in the transverse direction, it may be assumed that the strain is equal to the volume weighted average of the fiber strain and matrix strain, with the stress experienced by the fiber and matrix being equal. By these assumptions, an expression relating the material properties and volume fractions to the composite properties in the transverse directions can be derived. Denoting $\phi_2 = \phi_3 = \phi_t$ for properties ϕ in the transverse direction of the composite

$$\begin{aligned} \epsilon_t &= V_f \epsilon_f + V_m \epsilon_m \\ \frac{\sigma_t}{E_t} &= V_f \frac{\sigma_{f,t}}{E_{f,t}} + V_m \frac{\sigma_{m,t}}{E_{m,t}} \\ \frac{1}{E_t} &= \frac{V_f}{E_{f,t}} + \frac{V_m}{E_{m,t}} \end{aligned} \tag{2.6}$$

From the expression in equations (2.5) and (2.6) and the corresponding assumptions, one can derive an expression for the Poisson ratio of the composite,

$$\nu_{12} = V_f \nu_{12f} + V_m \nu_m. \tag{2.7}$$

2.3 The Halpin-Tsai model

The Halpin-Tsai model is a semi-empirical model that is used to approximate the transverse modulus and shear modulus of a transversely orthotropic material. The model is derived by the self-consistent field approach and contains a fitting parameter ξ. The transverse modulus is approximated as

$$E_2 = E_m \frac{1 + \xi_1 \eta_1 V_f}{1 - \eta_1 V_f}, \qquad \eta_1 = \frac{E_{2,f} - E_m}{E_{2,f} + \xi_1 E_m}. \tag{2.8}$$

The shear modulus G_{12} is approximated as

$$G_{12} = G_m \frac{1 + \xi_2 \eta_2 V_f}{1 - \eta_2 V_f}, \qquad \eta_2 = \frac{G_{12,f} - G_m}{G_{12,f} + \xi_2 G_m} \tag{2.9}$$

The parameters ξ_1 and ξ_2 can be fitted to experimental data or data from a finite element analysis by non-linear regression.

The problem with this model is that it's a semi-empirical mode. Simulation or experimentation are therefore required to determine the ξ-parameter. Additionally, the value of the parameter has little physical interpretation outside of parametrising the increased strength of the composite resulting from the interaction between the matrix and fibers.

3 Load tests

For all load tests, the unit cell described in section 2.1 was used, together with the material constants given in Table 1.1. Specific parameters for each simulation are found in appendix A.

3.1 Mesh size convergence

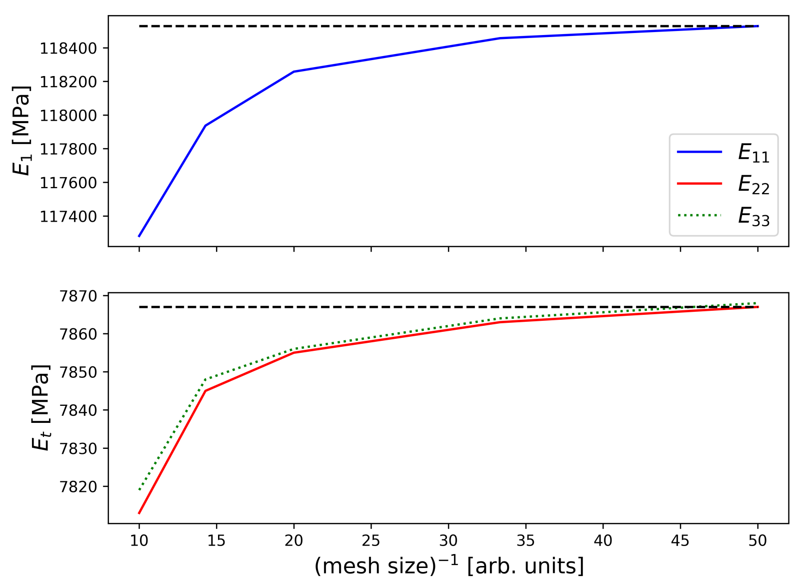

Figure 3.1: Calculated properties as a function of inverse mesh size.

In this section we want to see for our model how the precision of the mesh influence the precision of the results of the simulation. The mesh used for the load test simulations was gradually refined. The composite moduli computed from the load test results converged around a mesh size of 0.02 (arb. units) as shown in Figure 3.1.

As we can see, further that an certain point, the results doesn't change anymore. Only the simulation time increase, not the precision. For the rest of this assignment we will use the mesh size of the convergence point determine here. A relatively coarse mesh may have a minor influence on the results.

3.2 Variation of the model depending on E_{2f}

The transverse modulus of the fiber, E_{2f} is estimated and associated with significant uncertainty. The theoretical variation was be computed from equation (2.6). Additionally, E_{2f} was varied in the finite element model, the results are displayed in Figure 3.2. As expected, the variation is more prominent at higher volume fractions of fiber, it is clear that the analytical model under-predicts the variation. Especially for a high volume fraction of fiber one can observe that the response of the composite transverse modulus to a change in the transverse fiber modulus is highly non-linear, and increases rapidly as the pertubation surpasses $\approx \pm 2\%$.

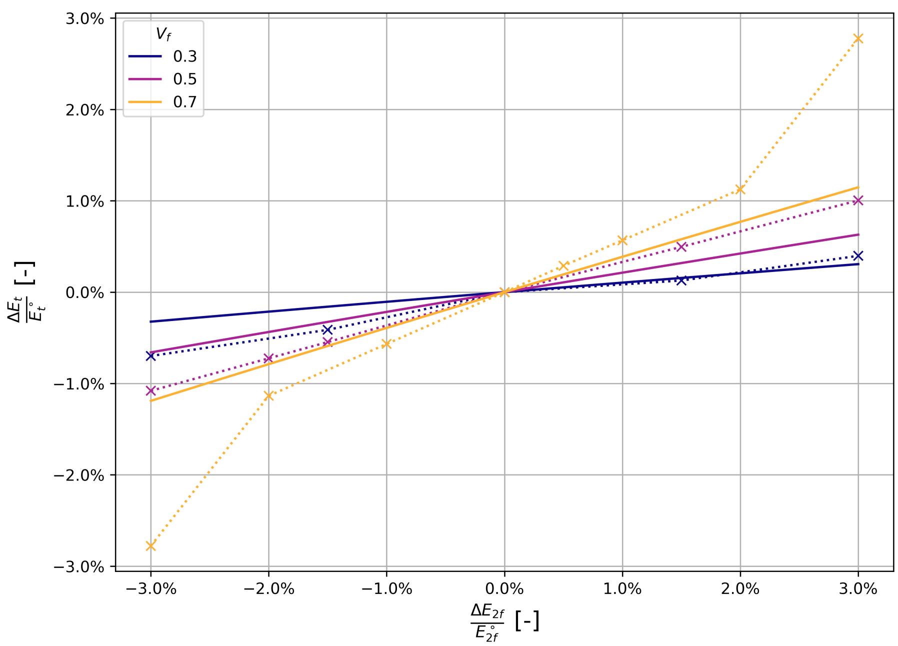

Figure 3.2: Relative change in the transverse modulus as a function of relative change in the transverse modulus of the fiber. Solid lines show theoretical variation, dashed lines show results from FEA. E_t° indicates the composite transverse modulus at $E_{2f} = E_{2f}^\circ = 15\,000\,\text{MPa}$

3.3 Simulation results

Now we can add on the graphics the results obtain with the Abaqus simulation, load testing simulated in Abaqus with a mesh size of $\frac{1}{50}$ was compared to theoretical values computed as described in section. Our result with the FEA model are displayed in Figure 3.4.

As we can see, the simulation fit really well the longitudinal modulus, on the contrary of the transverse modulus where the error is large. Therefore, another model is required if we want to use theoretical calculus.

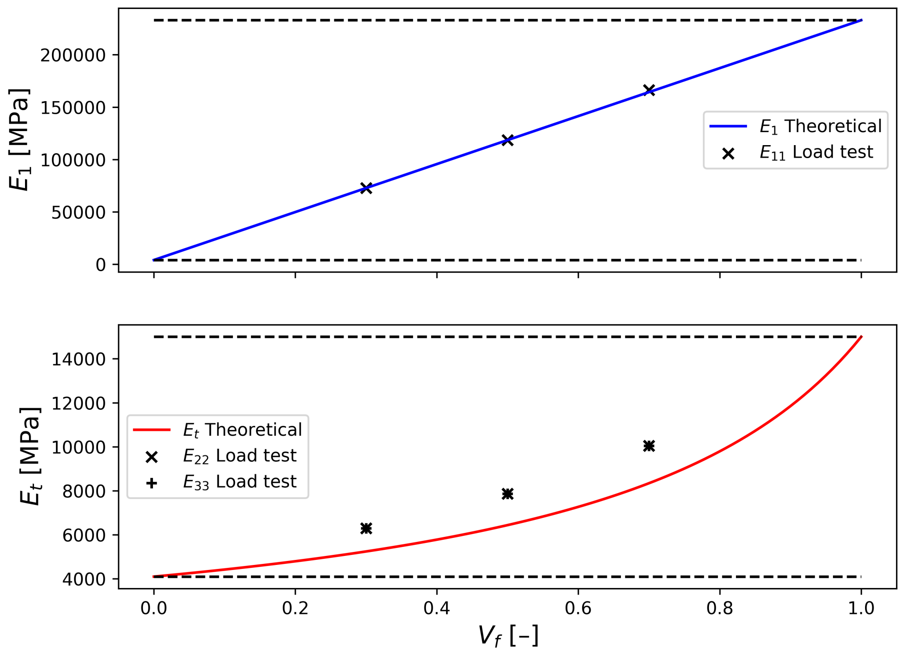

Figure 3.3: Comparison of composite moduli calculated by the theoretical models in section 2.2 and by FEA.

The comparison shown in Figure 3.4, shows that the simple micro-mechanical model is not accurate to the real value of the transverse modulus. Another model is required.

3.4 The Halpin-Tsai model

Using these results, we can refine our theoretical model with the Halpin-Tsai semi-empirical method. Now we want to try to find the right ξ_1 value with the results of our simulation. The fit resulting from a non-linear least squares regression computed using `scipy.optimize.curve_fit()` is shown in Figure 3.4

We have now a model that fit with a few error the simulation. This model could be use for different engineering application or for further calculus.

4 Conclusion

During this assignment we have worked for the first time on Abaqus and try to compare the theoretical model against the simulation software. To conclude, theses are our main observation during this work:

The micro-mechanical model, for the longitudinal modulus, fit well to the Abaqus simulation. It seems reasonable to think that this fact is correct for every composite sharing similar geometry. On the contrary, the transverse modulus have a poor accuracy with the micro-mechanical model. Another model is necessary and this is why we have tested the Halpin-Tsai model. This one allow a precision good enough for multiple engineering case.

In the same time, the mesh precision doesn't influence too much the accuracy of the results. Obviously, a really rough mesh will give false results. However, as our results prove, a good parameter for the mesh volume fraction will allow a low computation time with a good precision.

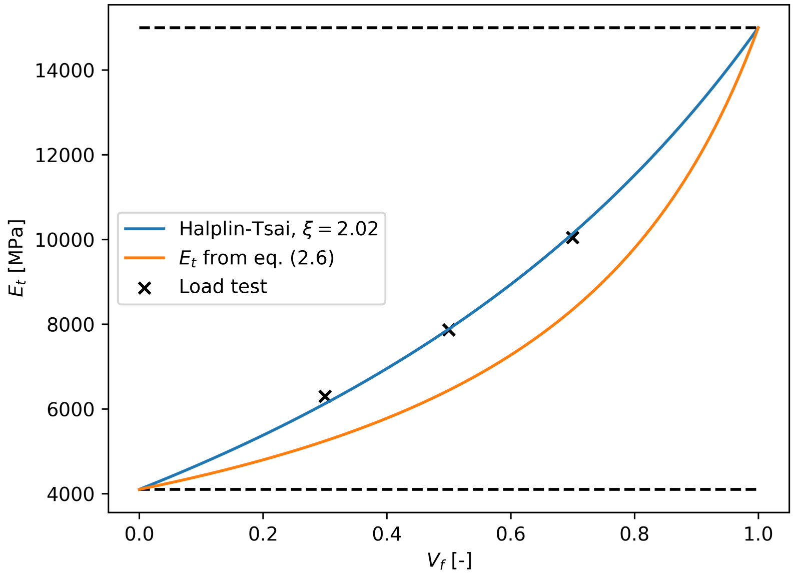

Figure 3.4: Comparison of composite moduli calculated by the theoretical models in section 2.2 and by FEA.

To further expand on this work, it would be interesting to compare our transverse modulus results with some other model, like the Puck or the Ekval model.

A Load test parameters

Mesh	Depth	Displacement	V_f	r_f
0.1	0.1	0.1	0.5	0.3713
0.07	0.07	0.07	0.5	0.3713
0.05	0.05	0.05	0.5	0.3713
0.03	0.03	0.03	0.5	0.3713
0.02	0.02	0.02	0.5	0.3713

Table A.1: Parameters for mesh convergence analysis. All parameters are given as relative sizes.

Mesh	Depth	Displacement	V_f	r_f
0.02	0.02	0.02	0.3	0.2876
0.02	0.02	0.02	0.5	0.3713
0.02	0.02	0.02	0.7	0.4392

Table A.2: Parameters used in load testing.

$V_f = 0.3$		$V_f = 0.5$		$V_f = 0.7$	
E_{2f}	E_{22}	E_{2f}	E_{22}	E_{2f}	E_{22}
15450	6322	14700	7810	15075	10073
14550	6253	15450	7946	15150	10101
15225	6305	14550	7782	14850	9987
14775	6271	15225	7906	15300	10157
		14775	7824	15450	10323
				14700	9930
				14550	9765

Table A.3: E_{2f} values used in pertubation tests, and corresponding computed E_{22} values. Mesh, displacement and fiber radius were equivalent to those found in Table A.2.

B Python code

All plots presented were generated by the code presented in this section. The excel files referenced in the code contain the data displayed in section A.

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.optimize as opt
import matplotlib.cm as cm
import matplotlib.ticker as mtick

def plot_moduli():
    E1f = 233000
    E2f = 15000
    Em = 4100

    V_f = np.linspace(0, 1, 100)
    V_m = 1 - V_f

    E1 = V_f * E1f + V_m * Em
    Et = 1/(V_f/E2f + V_m/Em)

    data = pd.read_excel('properties.xlsx')
    fracs = np.array(data['fracs'].tolist()) * 1e-2

    fig, ax = plt.subplots(2,1, figsize=(8,6), sharex=True)

    twin = ax[1]
    ax = ax[0]
    ax.plot(V_f, E1, label=r'$E_1$ Theoretical', color='blue')
    ax.scatter(fracs,data['E11'], marker='x', color='black', label=r'$E_{11}$ Load test')
    twin.plot(V_f, Et, label=r'$E_t$ Theoretical', color='red')

    ax.plot(V_f, [E1f for f in V_f], color='black', linestyle='--')
    ax.plot(V_f, [Em for f in V_f], color='black', linestyle='--')

    twin.scatter(fracs, data['E22'], marker='x', color='black', label=r'$E_{22}$ Load test')
    twin.scatter(fracs, data['E33'], marker='+', color='black', label=r'$E_{33}$ Load test')
    twin.plot(V_f, [E2f for f in V_f], color='black', linestyle='--')
    twin.plot(V_f, [Em for f in V_f], color='black', linestyle='--')

    ax.legend()
    twin.legend()
    twin.set_xlabel(r'$V_f$ [-]', fontsize=14)
    ax.set_ylabel(r'$E_1$ [MPa]', fontsize=14)
    twin.set_ylabel(r'$E_t$ [MPa]', fontsize=14)

    plt.suptitle('Composite moduli')
    plt.savefig('theoretical_moduli', dpi=600)
    plt.show()

def mesh_convergence():
    data = pd.read_excel('mesh.xlsx')
    sizes = np.array(data['mesh'].to_list())

    fig, axs = plt.subplots(2,1, figsize=(8,6), sharex=True)
    leg1, = axs[0].plot(1/sizes, data['E11'], color='blue', label=r'$E_{11}$')
    axs[0].plot(1/sizes, [data['E11'].to_list()[-1] for s in sizes], color='black', linestyle='--')

    leg2, = axs[1].plot(1/sizes, data['E22'], color='red', label=r'$E_{22}$')
    leg3, = axs[1].plot(1 / sizes, data['E33'], color='green', linestyle=':', label=r'$E_{33}$')
    axs[1].plot(1 / sizes, [data['E22'].to_list()[-1] for s in sizes], color='black', linestyle='--')

    axs[0].set_ylabel(r'$E_1$ [MPa]', fontsize=14)
```

```python
    axs[1].set_ylabel(r'$E_t$ [MPa]', fontsize=14)

    axs[1].set_xlabel(r'(mesh size)$^{-1}$ [arb. units]', fontsize=14)
    axs[0].legend(handles = [leg1, leg2, leg3], fontsize=14)
    plt.savefig('convergence', dpi=600)
    plt.show()

def halplin_tsai_Et(Vf, ksi):
    E2f = 15000
    Em = 4100
    eta = (E2f - Em) / (E2f + ksi * Em)

    E2 = Em * (1 + ksi * eta * Vf) / (1 - eta * Vf)

    return E2

def halplin_tsai_G(Vf, ksi):
    G2f = 9000
    Em = 4100
    nu_m = 0.37
    Gm = Em/(2 * (1 + nu_m))
    eta = (G2f - Gm) / (G2f + ksi * Gm)

    G2 = Gm * (1 + ksi * eta * Vf) / (1 - eta * Vf)

    return G2

def fit_halplin_tsai():
    data = pd.read_excel('properties.xlsx')
    xdata = np.array(data['fracs'].to_list()) * 1e-2
    ydata = np.array(data['E22'].to_list())
    y2data = np.array(data['E33'].to_list())

    xdata = np.concatenate((xdata, xdata))
    ydata = np.concatenate((ydata, y2data))
    coeff = opt.curve_fit(halplin_tsai_Et, xdata, ydata, p0=[0.5])
    Vf = np.linspace(0, 1, 100)

    Vm = 1 - Vf
    E2f = 15000
    Em = 4100
    Et_model = 1/(Vf/E2f + Vm/Em)

    plt.scatter(xdata, ydata, marker = 'x', color='black', label='Load test')
    plt.plot(Vf, halplin_tsai_Et(Vf, coeff[0]), label=r'Halplin-Tsai, $\xi = $'+str(round(coeff[0][0],3)))
    plt.plot(Vf, Et_model, label=r'$E_t$ from eq. (2.6)')
    plt.hlines([E2f, Em], 0, 1, colors='black', linestyles='--')
    plt.legend(loc='center left')
    plt.ylabel(r'$E_t$ [MPa]')
    plt.xlabel(r'$V_f$ [-]')
    plt.savefig('halplin_tsai', dpi=600)
    plt.show()

def pertubations():
    cmap = cm.get_cmap('plasma')
    fig, ax = plt.subplots(figsize=(9, 6.5))
    plt.sca(ax)

    data = pd.read_excel('pertubation.xlsx')
    color_scaler = lambda V: 0.9*V/(0.4) - 0.3/0.4

    Vf_list = np.array([0.3,0.5,0.7])
    p_list = np.linspace(-450, 450, 100)
    E12f_0 = 15000
    E2f = E12f_0 + p_list
```

```python
Em = 4100
for Vf in Vf_list:

    data_E2f = np.array(data['E2F_'+str(int(Vf*100))].to_list())
    data_Et = np.array(data['Et_'+str(int(Vf*100))].to_list())

    data_dE2f = (data_E2f - data_E2f[0])/data_E2f[0]
    data_dEt = (data_Et - data_Et[0])/data_Et[0]

    data_dE2f.sort()
    data_dEt.sort()

    Vm = 1 - Vf
    Et_0 = 1/(Vf/E12f_0 + Vm/Em)
    Et = 1/(Vf/E2f + Vm/Em)

    dEt = (Et - Et_0)/Et_0
    dE2f = p_list/E12f_0

    plt.plot(dE2f, dEt, color=cmap(color_scaler(Vf)), label=Vf)
    plt.plot(data_dE2f, data_dEt, color=cmap(color_scaler(Vf)), linestyle = ':', marker='x')

plt.grid()
plt.xlabel(r'$\frac{\Delta E_{2f}}{E_{2f}^\circ}$ [-]', fontsize=15)
plt.ylabel(r'$\frac{\Delta E_{t}}{E_{t}^\circ}$ [-]', fontsize=15)
plt.legend(title=r'$V_f$')
ax.xaxis.set_major_formatter(mtick.PercentFormatter(1, decimals=1))
ax.yaxis.set_major_formatter(mtick.PercentFormatter(1, decimals=1))
plt.savefig('pertubations', dpi=600)
plt.show()
```