Documentation for `viscosity_curve_195C_recalibrated.py`

Contents

1 Overview

The script `viscosity_curve_195C_recalibrated.py` implements a Krieger–Dougherty (KD)-based transfer methodology to predict the viscosity curve of a sand–plastic composite (SPC) at 195 °C from measured wood–plastic composite (WPC) data at the same temperature (50 wt% wood). Specifically, it:

1. Loads WPC data (shear_rate vs. η_{wpc}) at 195 °C.
2. Computes the volume fraction of wood (ϕ_{wpc}) from 50 wt% wood using densities.
3. Normalizes WPC viscosity to obtain relative viscosity ($\eta_{r,\text{wpc}}$) and fits a KD model to determine WPC's η_{int} and ϕ_m.
4. Back-calculates a "pure-polymer" viscosity curve by dividing out the KD factor.
5. Fits a power-law ($\eta = K\,\dot{\gamma}^{n-1}$) to that back-calculated polymer data.
6. Transfers to sand by reusing the same volume fraction ($\phi_{\text{spc}} = \phi_{\text{wpc}}$) and applying sand KD parameters ($\eta_{\text{int,sand}}$, $\phi_{m,\text{sand}}$).
7. Predicts absolute SPC viscosity (η_{spc}) by multiplying the KD factor for sand by the fitted polymer's power-law.
8. Exports results to CSV files and creates a log–log comparison plot (WPC vs. SPC).

The end result is a realistic SPC viscosity curve (Pa·s vs. shear rate) at 195 °C, anchored to actual WPC data and consistent polymer modeling.

2 Background & Modeling Approach

2.1 Krieger–Dougherty (KD) Model

The KD model is a semi-empirical formula for relative viscosity $\eta_r(\phi)$ of a particle-filled polymer as a function of particle volume fraction ϕ:

$$\eta_r(\phi) \;=\; \left(1 - \tfrac{\phi}{\phi_m}\right)^{-\,\eta_{\text{int}}\,\phi_m},$$

where

- η_{int} = intrinsic viscosity (how strongly particles thicken the melt),
- ϕ_m = maximum packing fraction (theoretical limit as $\phi \to \phi_m$).

KD yields a constant relative viscosity (no $\dot{\gamma}$ dependency) for fixed ϕ. To obtain *absolute* viscosity at varying shear rates, one multiplies $\eta_r(\phi)$ by a shear-rate-dependent base polymer viscosity $\eta_{\text{polymer}}(\dot{\gamma})$.

2.2 Transfer from WPC to SPC

1. Measure WPC (wood–polymer) at 50 wt% wood → compute ϕ_{wpc}.
2. Fit KD to $(\phi_{\text{wpc}}, \eta_{r,\text{wpc}})$ to find $(\eta_{\text{int,wpc}}, \phi_{m,\text{wpc}})$.
3. "Back-calculate" the polymer's $\eta_{\text{polymer}}(\dot{\gamma})$ at 195 °C by dividing out

$$\text{KD\_factor}_{\text{WPC}} = \text{KD}\big(\phi_{\text{wpc}}; \eta_{\text{int,wpc}}, \phi_{m,\text{wpc}}\big).$$

4. Fit a power-law to the back-calculated polymer data:

$$\eta_{\text{polymer}}(\dot{\gamma}) = K_{\text{poly}}\,\dot{\gamma}^{\,n_{\text{poly}}-1}.$$

5. For sand, choose KD parameters $(\eta_{\text{int,sand}}, \phi_{m,\text{sand}})$, reuse $\phi = \phi_{\text{wpc}}$, compute

$$\text{KD\_factor}_{\text{SPC}} = \text{KD}\big(\phi_{\text{spc}}; \eta_{\text{int,sand}}, \phi_{m,\text{sand}}\big).$$

6. Predict SPC absolute viscosity:

$$\eta_{\text{spc}}(\dot{\gamma}) = \underbrace{\text{KD\_factor}_{\text{SPC}}}_{\text{constant in }\dot{\gamma}} \times \underbrace{K_{\text{poly}}\,\dot{\gamma}^{\,n_{\text{poly}}-1}}_{\text{fitted polymer curve}}.$$

7. Compare WPC (original) vs. SPC (predicted) on a log–log plot.

—

3 Dependencies

- **Python 3.7+** (tested on 3.8/3.9)
- **Pandas** (1.1.0)
- **NumPy** (1.18.0)
- **SciPy** (1.4.0) – for `scipy.optimize.leastsq`
- **Matplotlib** (3.2.0)

Install via `pip`:

```
pip install pandas numpy scipy matplotlib
```

—

4 File Structure

```
/project-folder

 wpc_viscosity.csv                     # Input: WPC data at 195 °C (no header)
 viscosity_curve_195C_recalibrated.py # This Python script

 Output files produced by the script:
 polymer_fit_195C.csv                  # Back-calculated polymer curve
 spc_viscosity_prediction_195C.csv     # Predicted SPC data
 spc_prediction_195C.png               # Plot comparing WPC vs. SPC curves
```

`wpc_viscosity.csv`: A two-column, headerless CSV containing:

1. `shear_rate` [1/s]
2. `eta_wpc` [Pa·s]

Measurements are taken at 195 °C, with exactly 50 wt% wood flour in the polymer matrix.

`viscosity_curve_195C_recalibrated.py`: Main script that loads `wpc_viscosity.csv`, performs KD fitting, back-calculates polymer, fits power-law, transfers to sand KD, predicts SPC, exports CSVs, and plots.

—

5 Step-by-Step Workflow

5.1 Loading WPC Data & Computing ϕ_{wpc}

```
# 3.1 Read in the WPC data (195 °C, 50 wt% wood). No header.
df = pd.read_csv(DATA_CSV, header=None)
df.columns = ["shear_rate", "eta_wpc"]
```

- **Purpose:** Load wood–composite data with two columns (no header).
- **Result:** DataFrame `df` with `shear_rate` and `eta_wpc`.

```
# 3.2 Compute _wpc from 50 wt% wood:
phi_wpc_value = wt_to_vol_frac(WT_FRAC_WPC, RHO_WOOD, RHO_PE)
df["phi_wpc"] = phi_wpc_value
```

- **Compute ϕ_{wpc}:**

$$\phi_{\text{wpc}} = \frac{(WT\_FRAC\_WPC/100)/\rho_{\text{wood}}}{(WT\_FRAC\_WPC/100)/\rho_{\text{wood}} \;+\; (1 - WT\_FRAC\_WPC/100)/\rho_{\text{polymer}}}.$$

- **Assign:** A constant column `phi_wpc` ≈ 0.41 in every row.

```
# 3.3 Compute relative viscosity of WPC: _r_wpc = _wpc /
eta_matrix0 = df["eta_wpc"].iloc[0]
df["eta_r_wpc"] = df["eta_wpc"] / eta_matrix0
```

- **Assume:** The first (lowest shear rate) `eta_wpc` approximates zero-shear polymer viscosity (η_0).
- **Compute:** `eta_r_wpc[i]` = `eta_wpc[i]` / `eta_matrix0`.

—

5.2 Fitting Krieger–Dougherty (KD) to WPC

```
eta_int_wpc, phi_m_wpc = fit_kd(df["phi_wpc"].values, df["eta_r_wpc"].values)
```

- **Function:** `fit_kd` minimizes

$$\sum_i \big(\ln(\eta_{r,\mathrm{wpc},i}) - \ln\big(\mathrm{KD}(\phi_{\mathrm{wpc},i}; [\eta_{\mathrm{int}}, \phi_m])\big)\big)^2.$$

- **Output:**
  - `eta_int_wpc` = fitted intrinsic viscosity for wood filler in polymer.
  - `phi_m_wpc` = fitted maximum packing fraction for wood.

```
# Check _m_wpc > _wpc. If not, bump upward:
if phi_m_wpc <= phi_wpc_value:
    phi_m_wpc = phi_wpc_value + 1e-3
```

- **Ensure:** $\phi_{m,\mathrm{wpc}} > \phi_{\mathrm{wpc}}$ (otherwise KD argument is invalid).

```
# 4.1 Compute KD factor for WPC at _wpc:
df["kd_wpc"] = krieger_dougherty_safe(
    np.array([phi_wpc_value]), eta_int_wpc, phi_m_wpc
)[0]
print(f"KD factor for WPC: _r_wpc  {df['kd_wpc'].iloc[0]:.4f}")
```

- **kd_wpc** = $\big(1 - \phi_{\mathrm{wpc}}/\phi_{m,\mathrm{wpc}}\big)^{-\eta_{\mathrm{int,wpc}}\,\phi_{m,\mathrm{wpc}}}$.
- **Should match (approximately)** $\eta_{r,\mathrm{wpc}}$ averaged across data.

—

5.3 Back-Calculating the Pure-Polymer Viscosity

```
# 5.1 Compute: _polymer_data[i] = _wpc[i] / kd_wpc
df["eta_polymer_data"] = df["eta_wpc"] / df["kd_wpc"]
```

- **Rationale:**

$$\eta_{\mathrm{wpc},i} = \eta_{\mathrm{KD,wpc}} \times \eta_{\mathrm{polymer}}(\dot{\gamma}_i).$$

  Therefore,

$$\eta_{\mathrm{polymer}}(\dot{\gamma}_i) = \frac{\eta_{\mathrm{wpc},i}}{\eta_{\mathrm{KD,wpc}}}.$$

- **Result:** A back-calculated polymer viscosity curve at 195 °C.

- **Save:**

```
df_poly = df[["shear_rate", "eta_polymer_data"]]
df_poly.to_csv("polymer_fit_195C.csv", index=False)
```

Produces a two-column CSV (`polymer_fit_195C.csv`) containing `shear_rate` vs. `eta_polymer_data`.

—

5.4 Fitting a Power-Law to the Polymer

```
K_poly, n_poly = fit_power_law(df["shear_rate"].values, df["eta_polymer_data"].values)
```

- **Model:**
$$\eta_{\text{polymer}}(\dot\gamma) = K\,\dot\gamma^{\,n-1}.$$
- **Method:** Linear regression on $\ln\eta$ vs. $\ln\dot\gamma$:
$$\ln\eta = \ln K + (n-1)\ln\dot\gamma$$
$\rightarrow$ slope $= (n-1)$, intercept $= \ln K$.
- **Output:**
  - `K_poly` = consistency index (Pa·s^n).
  - `n_poly` = flow index ($n < 1$ for shear-thinning).

—

5.5 Transferring to Sand: Computing ϕ_{spc} & $\eta_{r,\text{spc}}$

```
if FORCE_PHI_SPC:
    df["phi_spc"] = 0.50  # 50 vol% sand
else:
    df["phi_spc"] = df["phi_wpc"]  # reuse _wpc  0.41
```

- **Option A (force $\phi_{\text{spc}} = 0.50$)** vs. **Option B (reuse ϕ_{wpc})**.
- For a "strict KD transfer," use Option B (same volume fraction).

```
df["eta_r_spc"] = krieger_dougherty_safe(
    df["phi_spc"].values, ETA_INT_SAND, PHI_M_SAND
)
```

- **Compute:**
$$\eta_{r,\text{spc}} = \big(1 - \phi_{\text{spc}}/\phi_{m,\text{sand}}\big)^{-\,\eta_{\text{int,sand}}\,\phi_{m,\text{sand}}}.$$
- This is a *single constant* across all shear rates.

—

5.6 Predicting SPC Viscosity & Exporting Results

```
# 7.1 Define fitted polymer model:
def polymer_viscosity_fit(gamma_dot: float) -> float:
    return K_poly * (gamma_dot ** (n_poly - 1))

# 7.2 Compute absolute SPC: _spc = _r_spc * _polymer_fit()
df["eta_spc"] = df["eta_r_spc"] * df["shear_rate"].apply(polymer_viscosity_fit)

# 7.3 Export predicted SPC data:
df_out = df[["shear_rate", "phi_spc", "eta_spc"]]
df_out.to_csv("spc_viscosity_prediction_195C.csv", index=False)
```

- ‘**eta**$_{spc}$‘$is the final predicted sand-composite viscosity at 195C, for each shear rate.$**Export:**$Creates$`spc_viscosity_`
`shear_rate`, `phi_spc`, `eta_spc`

—

5.7 Plotting & Visualization

- `plt.figure(figsize=(6, 4))`

```
# (a) Original WPC data
plt.loglog(df["shear_rate"], df["eta_wpc"], marker="o", linestyle="-",
           label="WPC (50 wt% wood) @195 °C")

# (b) Predicted SPC data
plt.loglog(df["shear_rate"], df["eta_spc"], marker="s", linestyle="--",
           label=f"SPC (={df['phi_spc'].iloc[0]:.2f}) @195 °C")

plt.xlabel("Shear Rate [1/s]")
plt.ylabel("Viscosity [Pa·s]")
plt.title("195 °C: WPC @50 wt% vs. Predicted SPC @ same ")
plt.grid(which="both", ls="--", alpha=0.3)
plt.legend()
plt.tight_layout()

output_png = "spc_prediction_195C.png"
plt.savefig(output_png, dpi=300)
plt.show()
```

- **Blue circles:** Original WPC data at 195 °C.
- **Orange squares:** Predicted SPC data (at same ϕ) at 195 °C.
- Both curves share the same shear-thinning slope (since the polymer model is reused), but are vertically offset by KD factors.

—

6 Function Reference

Below are the helper functions, their purpose, inputs, and outputs.

6.1 wt_to_vol_frac

```
def wt_to_vol_frac(wt_frac: float, rho_filler: float, rho_matrix: float) -> float:
```

- **Purpose:** Convert a filler weight fraction (wt%) into a *volume fraction* ϕ, given filler density and matrix density.
- **Arguments:**

– $\texttt{wt}_f rac(float) : Weightfractionoffiller(inpercent, e.g.$ 50.0).

- **Returns:** `phi` (float): Resulting volume fraction (unitless, between 0 and 1).
- **Equation:**

$$w = \frac{\text{wt\_frac}}{100}, \quad \phi = \frac{\frac{w}{\rho_{\text{filler}}}}{\frac{w}{\rho_{\text{filler}}} + \frac{1-w}{\rho_{\text{matrix}}}}.$$

6.2 krieger_dougherty_safe

```
def krieger_dougherty_safe(phi: np.ndarray, eta_int: float, phi_m: float) -> np.ndarray:
```

- **Purpose:** Compute the Krieger–Dougherty relative viscosity $\eta_r(\phi)$ safely, ensuring no invalid power raises if $\phi \geq \phi_m$.
- **Arguments:**
  - `phi` (np.ndarray): Array of volume fractions ϕ.
  - $\texttt{eta}_i nt(float) : Intrinsicviscosityparameter(\eta_{\text{int}}$).
  - $\texttt{phi}_m(float) : Maximumpackingfraction(\phi_m$).
- **Returns:** $\texttt{eta}_r(np.ndarray) : Arrayofrelativeviscositiesforeach\phi$, computed as

$$\eta_r(\phi) = \left(\max(1 - \phi/\phi_m, \varepsilon)\right)^{-\eta_{\text{int}}\,\phi_m},$$

  where $\varepsilon = 10^{-12}$ is a small clip to avoid zero or negative base.

- Implementation Details:
  - Calculate base = 1.0 - (phi / $\text{phi}_m).Clip$ base $toaminimumof$ 1e-12($sothat$(1 - $\phi/\phi_m) \leq$ 0 becomes 1e-12).
  - Compute base_clipped**(-eta_int * phi_m).

6.3 fit_kd

```
def fit_kd(phi_array: np.ndarray, eta_r_array: np.ndarray) -> tuple:
```

- **Purpose:** Fit KD parameters $(\eta_{\text{int}}, \phi_m)$ to experimental data $(\phi_i, \eta_{r,i})$ by minimizing the sum of squared residuals in log-space:

$$\text{residual}_i(\eta_{\text{int}}, \phi_m) = \ln(\eta_{r,i}) - \ln\big(\eta_{r,\text{KD}}(\phi_i; \eta_{\text{int}}, \phi_m)\big).$$

- **Arguments:**
  - `phi`$_array(np.ndarray) : 1Darrayofvolumefractions\phi_i$.
  - `eta`$_rarray(np.ndarray) : 1Darrayofmeasuredrelativeviscosities\eta_{r,i}$.
- **Returns:** $(\eta_{\text{int}}\_\text{fit}, \phi_m\_\text{fit})$ – Fitted intrinsic viscosity and maximum packing fraction.
- **Method:**
  - Defines a residual function in log-space.
  - Uses `scipy.optimize.leastsq` with initial guess [6.0, 0.30].
  - Returns the optimized parameters.

6.4 fit_power_law

```
def fit_power_law(shear_rates: np.ndarray, viscosities: np.ndarray) -> tuple:
```

- **Purpose:** Fit a power-law model $\eta(\dot{\gamma}) = K\,\dot{\gamma}^{\,n-1}$ to a set of $(\dot{\gamma}_i, \eta_i)$ data.
- **Arguments:**
  - `shear_rates` (np.ndarray): 1D array of shear rates $\dot{\gamma}_i$.
  - `viscosities` (np.ndarray): 1D array of viscosities η_i.
- **Returns:** $(K_{\text{fit}}, n_{\text{fit}})$ – Fitted power-law constants:

$$\ln\eta = \ln K + (n-1)\ln\dot{\gamma} \quad\Longrightarrow\quad m = n-1, \quad b = \ln K.$$

- **Method:** Perform a linear regression on $(\ln\eta, \ln\dot{\gamma})$ to extract slope m and intercept b. Then $n = m + 1$, $K = e^b$.

6.5 polymer_viscosity_fit

```
def polymer_viscosity_fit(gamma_dot: float) -> float:
    return K_poly * (gamma_dot ** (n_poly - 1))
```

- **Purpose:** Evaluate the fitted power-law polymer model at a given shear rate:

$$\eta_{\text{polymer}}(\dot{\gamma}) = K_{\text{poly}}\,\dot{\gamma}^{\,n_{\text{poly}}-1}.$$

- **Arguments:** `gamma_dot` (float): Shear rate $\dot{\gamma}$.
- **Returns:** `_polymer` (float): Calculated pure-polymer viscosity at the given $\dot{\gamma}$.
- **Note:** Uses the global variables `K_poly` and `n_poly` determined by `fit_power_law`.

—

7 How to Run

1. **Ensure all dependencies are installed:**

```
pip install pandas numpy scipy matplotlib
```

2. **Place:**

   - `wpc_viscosity.csv` (50 wt% wood WPC data at 195 °C, no header)
   - `viscosity_curve_195C_recalibrated.py`

   in the same directory.

3. **Run** the script:

```
python viscosity_curve_195C_recalibrated.py
```

4. **Observe console output:**

   - Fitted KD parameters for WPC ($\eta_{\text{int,wpc}}, \phi_{m,\text{wpc}}$).
   - Back-calculated KD factor for WPC.
   - Fitted polymer power-law constants ($K_{\text{poly}}, n_{\text{poly}}$).
   - KD factor for SPC ($\eta_{r,\text{spc}}$).
   - Confirmation of exported CSV files and saved PNG plot.

5. **Check generated files:**

   - `polymer_fit_195C.csv`
   - `spc_viscosity_prediction_195C.csv`
   - `spc_prediction_195C.png`

—

8 Output Files & Their Contents

8.1 `polymer_fit_195C.csv`

- Columns:

  `shear_rate`, `eta_polymer_data`

- This is the "back-calculated" pure-polymer viscosity curve at 195 °C.
- Obtained by dividing the measured WPC viscosities by the KD factor for wood at ϕ_{wpc}.

8.2 `spc_viscosity_prediction_195C.csv`

- Columns:

  `shear_rate`, `phi_spc`, `eta_spc`

- Predictions for the sand–polymer composite at 195 °C and $\phi_{\mathrm{spc}} = \phi_{\mathrm{wpc}}$ (0.41).
- `eta_spc` is the KD factor for sand at ϕ_{spc} multiplied by the fitted polymer power-law.

8.3 `spc_prediction_195C.png`

- A log–log plot comparing:
  - **WPC (195 °C, 50 wt% wood)** – blue circles.
  - **Predicted SPC (195 °C, $\phi = 0.41$)** – orange squares.
- Both curves share the same shear-thinning slope (since the polymer model is reused), but are vertically offset by KD factors.

—

9 Customization & Parameter Tuning

- **Force exactly 50 vol% sand:** Set `FORCE_PHI_SPC = True` near the top of the script. Then $\phi_{\mathrm{spc}} = 0.50$ instead of reusing ϕ_{wpc}.
- **Use different sand KD parameters:** Adjust `ETA_INT_SAND` and `PHI_M_SAND`. For example:

$$\texttt{ETA\_INT\_SAND} = 2.5, \quad \texttt{PHI\_M\_SAND} = 0.64.$$

- **Use a different polymer model:** Modify `polymer_viscosity_fit` or replace the power-law entirely. If you have a Carreau or Cross equation, implement it instead. If you have a separate neat-polymer CSV at 195 °C, you can skip Section 5.4 and directly interpolate that data as $\eta_{\mathrm{neat}}(\dot{\gamma})$.
- **Change WPC weight fraction:** If your wood composite is not exactly 50 wt% but some other percentage, modify:

$$\texttt{WT\_FRAC\_WPC} =< \text{your new wt\%} >$$

  The script will recalculate ϕ_{wpc} accordingly.

- **Run at a different temperature:** You need a new CSV of WPC data at that temperature (no header). Update `DATA_CSV` to point to that file, and rename output files accordingly (e.g. `viscosity_curve_185C.py`, output to `polymer_fit_185C.csv`, etc.). Ensure you choose KD parameters ($\eta_{\mathrm{int,sand}}, \phi_{m,\mathrm{sand}}$) appropriate for that polymer at that temperature, if they differ significantly.

—

10 Troubleshooting & Common Pitfalls

1. **Invalid Value in Power Warning** If you see `RuntimeWarning: invalid value encountered in power`, it means at some point $\phi \geq \phi_m$ was passed into the KD exponent.

   - Check that `phi_wpc < phi_m_wpc` after fitting; the script automatically bumps $\phi_{m,\mathrm{wpc}}$ by 10^{-3} if necessary.
   - Check that `phi_spc < phi_m_sand`; if you force $\phi_{\mathrm{spc}} = 0.50$ but choose $\phi_{m,\mathrm{sand}} < 0.50$, KD will produce invalid values. Ensure $\phi_{m,\mathrm{sand}} > \phi_{\mathrm{spc}}$.

2. **Poor Power-Law Fit for Polymer**

   - The back-calculated polymer data (`eta_polymer_data`) should form a roughly straight line on a log–log plot vs. `shear_rate`.
   - If it is very noisy or non-linear, consider using a Carreau or Cross model instead of a simple power-law.
   - Check if any measured WPC data points are erroneous or have experimental slip issues.

3. **Output SPC Curve Looks Too High or Low**

   - Revisit `ETA_INT_SAND` and `PHI_M_SAND`. Slight adjustments (e.g. $\eta_{\mathrm{int,sand}} = 2.5$ or $\phi_{m,\mathrm{sand}} = 0.60$) can shift the KD factor by 10–20%.
   - Verify that the base polymer fit (`K_poly`, `n_poly`) is reasonable: e.g. at 195 °C, typical polyolefin K_{poly} might be on the order of $10^2 - 10^3$ Pa·s^n, and $n_{\mathrm{poly}} \approx 0.4 - 0.8$.

4. **Mismatched Shear Rates Between WPC and Neat Data**

   - In this script, we assume WPC data is at a set of discrete shear rates, and the pure-polymer fit is derived from the same data via the KD factor.
   - If you instead have a separate pure-polymer CSV with different shear rates, you must interpolate the pure-polymer data at the WPC shear rates (or resample to a common grid). Otherwise, `eta_spc = eta_r_spc * eta_neat()` may mismatch.

5. **Plot Is Empty or Not Showing Points**

   - Ensure `df["shear_rate"]` is strictly positive (no zeros). A zero shear rate will cause log–log plotting to fail.
   - If `df["eta_spc"]` or `df["eta_wpc"]` contain NaNs (due to invalid KD calls), investigate upstream points where $\phi \geq \phi_m$ and correct them.

End of Documentation