From Energy-Fan Percentiles to $x_{\max}(E_{cs})$ via Swebrec Geometry

sykim

September 11, 2025

Notation

$P(x)$	cumulative passing at size x
x_P	percentile size at $P \times 100\%$ passing (e.g., x_{50})
$x_{\max}$	Swebrec upper bound (top size)
D	normalization length in the fan (here $D = x_{\max}$)
E_{cs}	specific energy (un-normalized)
E'	normalized energy, $E' = E_{cs}\,(D/25)^{\lambda}$
b	Swebrec shape parameter
A	$A \equiv \ln \dfrac{x_{\max}}{x_{50}}$
C_P, α_P	fan coefficients on a given branch: $\dfrac{x_P}{D} = C_P\, E'^{-\alpha_P}$

1 Initial derivation

(1) Started from the 3-parameter Swebrec:

$$P(x) = \frac{1}{1 + \left(\frac{\ln(x_{\max}/x)}{\ln(x_{\max}/x_{50})}\right)^b}. \tag{1}$$

Evaluated at $x = x_{20}$ to get a relation such as $\ln(x_{\max}/x_{20})/\ln(x_{\max}/x_{50}) = 4^{1/b}$.

(2) Wrote the (double) energy-fan laws with $D = x_{\max}$ as

$$\frac{x_P}{x_{\max}} = C_P\left[E_{cs}(x_{\max}/25)^{\lambda}\right]^{-\alpha_P}. \tag{2}$$

(3) Substituted Eq. (2) into the log-ratio above, replacing, e.g., $\ln(x_{\max}/x_{20})$ with $-\ln C_{20} + \alpha_{20}\ln E'$.

(4) Solved the resulting $\big(\alpha_{20}\ln E' - \ln C_{20}\big)/\big(\alpha_{50}\ln E' - \ln C_{50}\big) = 4^{1/b}$ for $\ln E'$ as a constant, and then used $E' = E_{cs}(x_{\max}/25)^{\lambda}$ to infer $x_{\max} \propto E_{cs}^{-1/\lambda}$.

2 Issue with the initial derivation

E' is forced to be a constant. In fact, evaluating at $x = x_{20}$ gives $E' \approx 0.378$ for a set of parameters provided by Ouchterlony et al. doi:10.3390/min11111262 for a high-energy branch, whereas evaluating at $x = x_{80}$ with $\big(\alpha_{80}\ln E' - \ln C_{80}\big)/\big(\alpha_{50}\ln E' - \ln C_{50}\big) = 0.25^{1/b}$ gives $E' \approx 0.294$.

3 Alternative derivation

On a given branch (low or high energy), the double fan is

$$\frac{x_P}{D} = C_P\, E'^{-\alpha_P}, \qquad E' = E_{cs}\left(\frac{D}{25}\right)^{\lambda}. \tag{3}$$

With fixed D, we have

$$\ln\frac{x_{80}}{x_{50}} = \ln\frac{C_{80}}{C_{50}} - (\alpha_{80}-\alpha_{50})\ln E', \tag{4}$$

$$\ln\frac{x_{20}}{x_{50}} = \ln\frac{C_{20}}{C_{50}} - (\alpha_{20}-\alpha_{50})\ln E'. \tag{5}$$

From Eq. (1), evaluating at x_{80} and x_{20} gives

$$r(E') \equiv -\frac{\ln(x_{80}/x_{50})}{\ln(x_{20}/x_{50})} = \frac{1-0.25^{1/b(E')}}{4^{1/b(E')}-1} = 0.25^{1/b(E')}. \tag{6}$$

Thus $b(E') = \dfrac{\ln 0.25}{\ln r(E')}$. Then

$$A(E') = \ln\frac{x_{\max}}{x_{50}} = -\frac{\ln(x_{80}/x_{50})}{0.25^{1/b(E')}-1} = -\frac{\ln(x_{80}/x_{50})}{r(E')-1}. \tag{7}$$

At large E', intercepts in the fan are negligible so

$$r(E') \to r_\infty = \frac{\alpha_{50}-\alpha_{80}}{\alpha_{20}-\alpha_{50}}, \qquad b(E') \to b_\infty = \frac{\ln 0.25}{\ln r_\infty} = \text{const.} \tag{8}$$

Then Eq. (7) is affine in $\ln E'$:

$$A(E') = a_0 + a_1 \ln E', \qquad a_1 = \frac{\alpha_{80}-\alpha_{50}}{r_\infty - 1}. \tag{9}$$

Therefore $x_{\max}(E') = x_{50}(E')\, e^{A(E')}$ is a power law,

$$x_{\max}(E') = K\, E'^{-\beta}, \qquad \beta = \alpha_{50} - a_1 = \alpha_{50} + \frac{\alpha_{80}-\alpha_{50}}{1-r_\infty}. \tag{10}$$

With (10), $D = x_{\max}$, and $E' = E_{cs}(x_{\max}/25)^{\lambda}$,

$$x_{\max}^{1+\lambda\beta} = K\, 25^{\lambda\beta} E_{cs}^{-\beta} \;\Rightarrow\; \boxed{x_{\max}(E_{cs}) = K'\, E_{cs}^{-\gamma}\,,} \qquad \boxed{\gamma = \frac{\beta}{1+\lambda\beta}\,,} \tag{11}$$

where K' is a calibration constant set by one high-energy datapoint.

With high-energy branch in Faramarzi-Ore 1: With $\alpha_{80} = 0.6643$, $\alpha_{50} = 0.8817$, $\alpha_{20} = 1.3452$, $\lambda = 0.5142$:

$$r_\infty = \frac{0.8817-0.6643}{1.3452-0.8817} \approx 0.469, \qquad b_\infty \approx 1.83,$$

$$\beta = 0.8817 + \frac{0.6643-0.8817}{1-0.469} \approx 0.472,$$

$$\gamma = \frac{0.472}{1+0.5142\times 0.472} \approx 0.380.$$

Hence $x_{\max}(E_{cs}) \approx K'\, E_{cs}^{-0.380}$ on the high-energy branch. The prefactor K' can be set from one high-energy calibration pair $(E_{cs,0}, x_{\max,0})$.

4 Numerical approach and code

The general (piecewise) procedure:

(1) Fit fan laws on each branch for $P \in \{20, 50, 80\}$: $x_P/D = C_{P,\text{branch}}\, E'^{-\alpha_{P,\text{branch}}}$.

(2) For any E', compute $x_{20}/x_{\max}(E'), x_{50}/x_{\max}(E'), x_{80}/x_{\max}(E')$.

(3) Compute $r(E')$ from Eq. (6); invert to get $b(E')$.

(4) Compute $A(E')$ via Eq. (7); set $x_{\max}(E') = x_{50}(E')\, e^{A(E')}$.

(5) If $D = x_{\max}$, solve the fixed point $x = x_{\max}\big(E_{cs}(x_{\max}/25)^{\lambda}\big)$ by iteration in x.

Reference Python

```
import math
import numpy as np
import csv

# ------------------------------
# 1) Fan fits: x_P(E) = C_P * E^(-alpha_P)  (HIGH-energy branch)
C20, a20 = 0.05225, 1.3452
C50, a50 = 0.177,   0.8817
C80, a80 = 0.3137,  0.6643

def xP_over_D(E, C, a):  # percentile at energy E
    return C * (E ** (-a))

# ------------------------------
# 2) Invert r -> b  (r = (1 - 0.25^(1/b)) / (4^(1/b) - 1))
def b_from_r(r, tol=1e-10, b_lo=0.1, b_hi=20.0):
    def r_of_b(b):
        t1 = 0.25 ** (1.0 / b)
        t2 = 4.0 ** (1.0 / b)
        return (1.0 - t1) / (t2 - 1.0)
    r_lo, r_hi = r_of_b(b_lo), r_of_b(b_hi)
    if not (min(r_lo, r_hi) <= r <= max(r_lo, r_hi)):
        r = max(min(r, max(r_lo, r_hi)), min(r_lo, r_hi))
    for _ in range(80):
        b_mid = 0.5 * (b_lo + b_hi)
        r_mid = r_of_b(b_mid)
        if (r_mid - r) == 0 or abs(b_hi - b_lo) < 1e-12:
            return b_mid
        if (r_mid < r) == (r_lo < r):
            b_lo, r_lo = b_mid, r_mid
        else:
            b_hi, r_hi = b_mid, r_mid
    return 0.5 * (b_lo + b_hi)

# ------------------------------
# 3) Map energy -> x_max via Swebrec geometry with x20, x50, x80
def xmax_from_E(E):
    x20 = xP_over_D(E, C20, a20)
    x50 = xP_over_D(E, C50, a50)
    x80 = xP_over_D(E, C80, a80)
    l8050 = math.log(x80 / x50)
    l2050 = math.log(x20 / x50)
    r = -l8050 / l2050
```

```
    b = b_from_r(r)
    t = (0.25) ** (1.0 / b)
    A = -18050 / (t - 1.0)
    return x50 * math.exp(A)

# -----------------------------
# 4) Solve implicit x_max = F(E_cs * (x_max/BASE)^lambda)
def xmax_from_Ecs(Ecs, lam, BASE=25.0, x0=None, iters=50, tol=1e-10):
    if x0 is None:
        x0 = 0.5 * BASE
    y = math.log(x0)
    for _ in range(iters):
        E = Ecs * (math.exp(y) / BASE) ** lam
        x_new = xmax_from_E(E)
        y_new = math.log(x_new)
        if abs(y_new - y) < tol:
            y = y_new
            break
        y = y_new
    return math.exp(y)

# -----------------------------
# 5) Utility: energy needed to hit a target top size
def Ecs_for_target_xmax(x_target, lam, BASE=25.0):
    def H(Ecs):
        E = Ecs * (x_target / BASE) ** lam
        return xmax_from_E(E) - x_target
    lo, hi = 1e-6, 1e6
    # bracket
    hlo, hhi = H(lo), H(hi)
    expand = 0
    while hlo * hhi > 0 and hi <= 1e30:
        lo /= 10.0
        hi *= 10.0
        hlo, hhi = H(lo), H(hi)
        expand += 1
        if expand > 200:
            raise RuntimeError("Could not bracket root; check 
               coefficients/units.")
    for _ in range(80):
        mid = math.sqrt(lo * hi)  # bisection in log-space
        hmid = H(mid)
        if abs(hmid) < 1e-12:
            return mid
        if hlo * hmid < 0:
            hi, hhi = mid, hmid
        else:
            lo, hlo = mid, hmid
    return math.sqrt(lo * hi)

# -----------------------------
# 6) Generate & write x_max vs E_cs table
if __name__ == "__main__":
    # PARAMETERS (edit these):
    lam = 0.5142         # fragmentation-size coupling exponent
    BASE = 25.0          # same units as x (e.g., mm); must match fan-fit
        units
    npts = 101           # how many samples along E_cs
```

```python
Ecs_min, Ecs_max = 1e+0, 1e+5  # sweep range for E_cs

Ecs_grid = np.logspace(np.log10(Ecs_min), np.log10(Ecs_max), npts)
xmax_vals = [xmax_from_Ecs(Ecs, lam=lam, BASE=BASE) for Ecs in Ecs_grid]

# Write CSV
out_csv = "xmax_vs_Ecs.csv"
with open(out_csv, "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["E_cs", "x_max"])
    for Ecs, x in zip(Ecs_grid, xmax_vals):
        writer.writerow([f"{Ecs:.12g}", f"{x:.12g}"])

print(f"Wrote␣{out_csv}␣with␣{len(Ecs_grid)}␣rows.")

# Quick sanity print
for i in [0, len(Ecs_grid)//2, -1]:
    print(f"E_cs={Ecs_grid[i]:.4g}␣->␣x_max={xmax_vals[i]:.6g}")

# Plot
try:
    import matplotlib.pyplot as plt
    plt.figure()
    plt.loglog(Ecs_grid, xmax_vals)
    plt.xlabel(r"$E_\text{cs}$␣(kWh/t)")
    plt.ylabel(r"$x_{\max}$␣(mm)")
    plt.xlim(1, 1e5)
    plt.ylim(0.01, 1)
    plt.title(r"$x_{\max}$␣vs␣$E_\text{cs}$")
    plt.grid(True, which="both", ls=":")
    plt.show()
except Exception as e:
    print("Error:", e)
```