A Closed-Form, Accurate Formulation of Chromatographic Peak Shapes with Spectrum-Modeling Applications in Chromatogram Modeling

Author: Amy Kukleva
Version: 21 Aug 2025

Abstract

Chromatographic peaks are often treated as Gaussian under ideal, linear partitioning, yet real HPLC/GC peaks exhibit asymmetry (tailing/fronting) and baseline drift. We present a closed-form, computationally efficient model that is chemically justified and easy to tune: the exponentially modified Gaussian (EMG), i.e., a Gaussian band convolved with a first-order exponential delay. We define variables in analytical-chemistry terms but write equations for computational scientists. Each component peak is parameterized by retention time t_R , Gaussian dispersion σ , an exponential time constant τ governing asymmetry ($\tau > 0$ tailing, optional mirrored case for fronting), and area A proportional to injected mass. Mixture chromatograms are a sum of peaks plus a baseline model $B(t)$. We detail baseline estimation/removal (polynomial, asymmetric least squares, morphological/wavelet options) and deconvolution of overlapping peaks via constrained least-squares on EMG parameters. The model is unified for HPLC and GC; differences are reflected in parameter values, not functional form. Python code is provided for simulation, plotting, and baseline correction. The result is visually realistic and quantitatively faithful for calibration and algorithm testing.

Nomenclature (variables/parameters)

Chromatographic:
t : time (min or s).
t_0 : void (hold-up) time; elution time of an unretained compound.
t_R : retention time (peak center parameter in the model).
k' : retention (capacity) factor.
K_D : distribution (partition) coefficient, stationary vs mobile phase.
V_s, V_m : stationary and mobile phase volumes in the column; $\phi = V_s / V_m$.
N : number of theoretical plates (efficiency).
A : peak area (proportional to injected moles n times detector response factor).
h : peak height (not a primary parameter in our formulation; derived from A, σ, τ).

Peak-shape:
σ: Gaussian standard deviation in time (band dispersion).

τ : exponential time constant of tailing ($\tau > 0$: tailing; mirrored construction for fronting).
$\mathrm{erfc}(\cdot)$: complementary error function.
$B(t)$: baseline function (e.g., constant/linear/polynomial/ALS estimate).

Computational:
Grid $\{t_k\}_{k=1}^{M}$ for evaluation; $\mathbf{y}$ observed/simulated signal; θ parameter vector(s) for one or multiple peaks.

Background: retention and Gaussian band broadening

Under linear chromatography (constant K_D , linear adsorption isotherm) the retention factor is $k' = K_D\phi$ with $\phi = V_s / V_m$. The retention time is

$$t_R = t_0(1 + k') = t_0(1 + K_D\phi). \tag{1}$$

The ideal band exiting a high-efficiency column is approximately Gaussian; denoting the Gaussian standard deviation in time by σ, the ideal (symmetric) peak with **area** A is

$$I_G(t; t_R, \sigma, A) = \frac{A}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(t - t_R)^2}{2\sigma^2}\right). \tag{2}$$

σ subsumes column efficiency as well as any lumped, on-column dispersion.

A closed-form asymmetric peak: the EMG (semi-Gaussian long tail)

Real peaks often tail (or less commonly front). A physically justified, closed-form model is the **exponentially modified Gaussian (EMG)**: a Gaussian convolved with a one-sided exponential (first-order delay). Using **area** A as the size parameter (so integrals are invariant under asymmetry), the EMG intensity is

$$I_{\mathrm{EMG}}(t; t_R, \sigma, \tau, A) = \frac{A}{2\tau} \exp\left(\frac{\sigma^2}{2\tau^2} - \frac{t - t_R}{\tau}\right) \mathrm{erfc}\left(\frac{1}{\sqrt{2}}\left(\frac{\sigma}{\tau} - \frac{t - t_R}{\sigma}\right)\right), \qquad (\tau > 0). \tag{3}$$

Properties. (i) $\int_{-\infty}^{\infty} I_{\mathrm{EMG}}(t)\, dt = A$. (ii) As $\tau \to 0^+$, $I_{\mathrm{EMG}} \to I_G$ (Gaussian limit). (iii) Increasing τ/σ increases right-skew (tailing).

Fronting. For fronting, either reflect time about t_R :

$$I_{\mathrm{front}}(t) = I_{\mathrm{EMG}}(2t_R - t;\, t_R, \sigma, \tau_f, A), \tag{4}$$

or allow a signed τ in Eq. (3) with care to maintain non-negativity (the reflected form is numerically robust).

Alternative semi-Gaussian (bi-Gaussian) option. In some workflows, a piecewise Gaussian is convenient:

$$I_{\text{biG}}(t; t_R, \sigma_l, \sigma_r, A) = \frac{A}{Z} \times \begin{cases} \exp\left(-\dfrac{(t-t_R)^2}{2\sigma_l^2}\right), & t \le t_R, \\ \exp\left(-\dfrac{(t-t_R)^2}{2\sigma_r^2}\right), & t > t_R, \end{cases} \tag{5}$$

with normalization $Z = \sigma_l\sqrt{2\pi}/2 + \sigma_r\sqrt{2\pi}/2$ to ensure area A. Eq. (5) is differentiable everywhere except t_R; Eq. (3) is smooth and generally preferred.

Mixtures, baseline, and total signal

A multi-component chromatogram is a sum of peaks plus baseline:

$$y(t) = \sum_{i=1}^{n} I_i(t; t_{R,i}, \sigma_i, \tau_i, A_i) + B(t). \tag{6}$$

Choose any analytic $B(t)$: constant, linear $a + bt$, polynomial, spline, or an estimated baseline (see below). Noise can be added post hoc for realism.

Baseline detection and removal

Baselines (bleed, gradients, drift) bias area estimates. Common, effective strategies:

1. **Low-order polynomial/spline fit** to baseline windows (iteratively masking peaks).
2. **Asymmetric least squares (ALS)** baseline (Eilers–Boelens). Given samples $\{(t_k, y_k)\}$, seek $\mathbf{b}$ minimizing

$$\min_{\mathbf{b}} \sum_k w_k (y_k - b_k)^2 + \lambda \|D^2 \mathbf{b}\|_2^2, \tag{7}$$

   where D^2 is the discrete second-difference operator (smoothness), $\lambda > 0$ controls smoothness, and weights w_k update asymmetrically: $w_k = p$ if $y_k > b_k$ (down-weight peaks), else $w_k = 1 - p$, with $p \in (0, 1)$ (typically $p \approx 0.001$–0.05). Iterate weight updates to convergence. Resulting $\mathbf{b}$ is subtracted to yield baseline-corrected signal.
3. **Morphological (top-hat)** or **wavelet** filtering to separate low-frequency baseline from peaks.

ALS is robust, fast, and parameter-light; we provide code.

Overlapping peaks and integration

For partially resolved peaks, fit a sum of EMGs to the observed segment:

$$\min_{\theta} \sum_{k \in W} \left[y_k - \sum_{j=1}^{m} I_{\text{EMG}}(t_k; t_{R,j}, \sigma_j, \tau_j, A_j) \right]^2, \tag{8}$$

subject to $A_j \geq 0$, $\sigma_j > 0$, $\tau_j \geq 0$ (or mirrored fronting option). The fitted A_j give unbiased areas even without baseline separation, provided the shape family is appropriate.

HPLC vs GC

The same functional family (Eq. 3) fits both. Technique-specific effects appear via parameters: GC often has smaller τ (more symmetric peaks on well-deactivated columns) and temperature-dependent σ; HPLC tailing may reflect secondary interactions (silanol, ion-exchange), increasing τ. Retention mapping uses Eq. (1); for simulation convenience one may specify t_R directly and keep K_D implicit.

Implementation (Python)

Below is a compact library for simulation, plotting, ALS baseline, and optional bi-Gaussian/fronting support. All times use consistent units.

```
import numpy as np
from math import erfc
import matplotlib.pyplot as plt

# ---------------------- Peak models ----------------------

def emg(t, tR, sigma, tau, area):
    """Exponentially Modified Gaussian (EMG) with area normalization.
  t   : array-like time axis
  tR  : retention time (float)
  sigma: Gaussian std (float, >0)
  tau : tail time constant (float, >0 for tailing)
  area: total area under the peak (float, >=0)
  Returns y(t) with integral = area.
  """
    t = np.asarray(t)
    if tau == 0:
        # Gaussian limit
        return (area / (sigma * np.sqrt(2*np.pi))) * np.exp(-0.5 * ((t - tR)/
sigma)**2)
    z = (sigma / tau - (t - tR) / sigma) / np.sqrt(2.0)
    pref = (area / (2.0 * tau))
    expo = np.exp(0.5 * (sigma / tau)**2 - (t - tR)/tau)
    return pref * expo * np.vectorize(erfc)(z)


def emg_fronting(t, tR, sigma, tau_f, area):
    """Fronting via reflection: evaluate a tailing EMG reflected about tR."""
```

```
        t = np.asarray(t)
        return emg(2.0*tR - t, tR, sigma, tau_f, area)


def bigaussian(t, tR, sigma_left, sigma_right, area):
        """Piecewise Gaussian (semi-Gaussian) with different sigmas left/right;
area-normalized."""
        t = np.asarray(t)
        left = t <= tR
        right = ~left
        # unnormalized halves
        y = np.zeros_like(t, dtype=float)
        y[left] = np.exp(-0.5 * ((t[left]-tR)/sigma_left)**2)
        y[right] = np.exp(-0.5 * ((t[right]-tR)/sigma_right)**2)
        # normalization constant so that integral equals area
        Z = 0.5*np.sqrt(2*np.pi)*(sigma_left + sigma_right)
        return area * (y / Z)

# ---------------------- Baseline (ALS) -------------------

def baseline_als(y, lam=1e5, p=0.01, niter=10):
        """Asymmetric Least Squares baseline (Eilers–Boelens).
    y: 1D numpy array
    lam: smoothness (higher = smoother)
    p: asymmetry (0<p<1); small p pushes baseline under peaks
    niter: iterations of weight update
    Returns baseline b of same shape as y.
    """
        y = np.asarray(y, dtype=float)
        L = len(y)
        # Second difference operator D2 via sparse-like banded arrays
        # We build bands for solving (W + lam * D2^T D2) b = W y
        # Implemented with simple tridiagonal construction for D2^T D2
        D2 = np.zeros((L,))             # placeholder not used directly
        # Tridiagonal for D2^T D2 has bands: [1, -2, 1] convolved twice
        main = np.zeros(L)
        off1 = np.zeros(L-1)
        off2 = np.zeros(L-2)
        # Build D2^T D2 main and off-diagonals
        # See derivation: T = D2^T D2 yields five-diagonal matrix with coefficients
[1, -4, 6, -4, 1]
        T_main = np.full(L, 6.0)
        T_off1 = np.full(L-1, -4.0)
        T_off2 = np.full(L-2, 1.0)
        # Boundary adjustments
        T_main[0] = 1.0; T_main[1] = 5.0; T_main[-2] = 5.0; T_main[-1] = 1.0
        T_off1[0] = -2.0; T_off1[-1] = -2.0
        # Iterative reweighting
```

```
        w = np.ones(L)
        b = np.copy(y)
        for _ in range(niter):
                W_main = w.copy()
                # Solve (W + lam*T) b = W*y  via banded solver (here naive O(L^2) for
clarity)
                # Assemble five-diagonal matrix A
                A = np.zeros((L, L))
                # Fill W
                np.fill_diagonal(A, W_main)
                # Add lam*T
                np.fill_diagonal(A, A.diagonal() + lam*T_main)
                for i in range(L-1):
                        A[i, i+1] += lam*T_off1[i]
                        A[i+1, i] += lam*T_off1[i]
                for i in range(L-2):
                        A[i, i+2] += lam*T_off2[i]
                        A[i+2, i] += lam*T_off2[i]
                b = np.linalg.solve(A, W_main * y)
                # Update weights
                w = p * (y > b) + (1-p) * (y <= b)
        return b

# ---------------------- Simulation & plotting -----------

def simulate(t, peaks, baseline=None):
        """Simulate y(t) = sum(peaks) + baseline.
    peaks: list of dicts with keys {model, params}
          model in {"emg", "emg_front", "bigauss"}
          params: tuple matching the model signature
    baseline: callable(t) -> array or None
    """
        t = np.asarray(t)
        y = np.zeros_like(t, dtype=float)
        for pk in peaks:
                m = pk["model"].lower()
                if m == "emg":
                        tR, sigma, tau, area = pk["params"]
                        y += emg(t, tR, sigma, tau, area)
                elif m == "emg_front":
                        tR, sigma, tau_f, area = pk["params"]
                        y += emg_fronting(t, tR, sigma, tau_f, area)
                elif m == "bigauss":
                        tR, sl, sr, area = pk["params"]
                        y += bigaussian(t, tR, sl, sr, area)
                else:
                        raise ValueError(f"unknown model {m}")
        if baseline is not None:
```

```
        y = y + baseline(t)
    return y


def plot_chrom(t, y, yb=None, title=None):
    plt.figure(figsize=(9, 4))
    plt.plot(t, y, lw=1.5)
    if yb is not None:
        plt.plot(t, yb, lw=1.0)
        plt.legend(["signal", "baseline"])
    plt.xlabel("time")
    plt.ylabel("intensity")
    if title:
        plt.title(title)
    plt.tight_layout()
    plt.show()
```

Example use.

```
# Time axis
T = np.linspace(0, 6, 2001)
# Baseline (linear drift)
B = lambda t: 0.03 * t
# Peaks (mix of Gaussian, tailing EMG, and an overlapping neighbor)
peaks = [
    {"model": "emg",        "params": (1.0, 0.060, 0.000, 0.80)},    # Gaussian
(tau=0)
    {"model": "emg",        "params": (2.0, 0.055, 0.000, 0.90)},    # Gaussian
    {"model": "emg",        "params": (4.00, 0.050, 0.150, 1.20)},    # tailing
    {"model": "emg",        "params": (4.35, 0.050, 0.000, 0.35)},    # small
overlap
    {"model": "bigauss",    "params": (5.50, 0.045, 0.060, 0.70)},    # semi-
Gaussian
]
raw = simulate(T, peaks, baseline=B)
base = B(T)
corrected = raw - base

plot_chrom(T, raw, base, title="Raw chromatogram with baseline")
plot_chrom(T, corrected, title="Baseline-corrected chromatogram")

# Estimate baseline by ALS from raw (no prior B)
estB = baseline_als(raw, lam=2e5, p=0.01, niter=8)
plot_chrom(T, raw, estB, title="ALS baseline estimate")
```

Notes. (i) Areas A_i may be set from injected mass times response factor. (ii) For quantitative tests, integrate corrected peaks with trapezoidal rules or fit EMG parameters by least squares for overlapping regions.

Validation notes

- **Gaussian limit:** with $\tau = 0$, Eq. (3) reduces to Eq. (2) (code implements this branch explicitly).
- **Area invariance:** EMG area equals A by construction; numerical integration of `emg` recovers A to within quadrature tolerance.
- **Asymmetry control:** τ/σ serves as a dimensionless tailing knob. Typical moderate tailing corresponds to $\tau/\sigma \approx 1$–2.
- **Fronting:** use `emg_fronting` with $\tau_f > 0$ to mirror the tail to the leading side.

Discussion

Why EMG? It has a clear physical interpretation (dispersive Gaussian + kinetic delay), smooth closed form (analytical integrals/derivatives), and excellent empirical fit across HPLC/GC scenarios. When peaks are severely non-ideal (double-tails, adsorption heterogeneity), mixtures of EMGs or the bi-Gaussian (Eq. 5) can be used with minimal overhead.

HPLC vs GC differences. Differences predominantly affect parameter ranges: GC often entails smaller τ (better deactivation), σ scales with temperature/flow; HPLC tailing reflects secondary interactions (pH, silanol, ion exchange). The same equations apply.

Baseline strategy. ALS (Eq. 7) is a strong default; polynomial/spline fits to hand-picked baseline windows are effective when baseline is simple. Morphological or wavelet methods handle structured backgrounds.

Deconvolution. Eq. (8) enables area-faithful integration for overlapped peaks. Provide tight bounds and good initials (from apex/time windows); fix τ if necessary for stability.

Conclusion

A compact, closed-form model based on EMG peaks (with optional mirrored and bi-Gaussian variants) accurately reproduces realistic HPLC/GC chromatograms, while remaining fast and tunable. It unifies HPLC and GC under one functional form, supports baseline estimation and overlap deconvolution, and maps cleanly to physical parameters (t_R, σ, τ, A). The provided Python layer is ready to plug into simulators and data systems for calibration, method development, and algorithm testing.

References

[1] J.C. Giddings, *Dynamics of Chromatography*, Marcel Dekker, 1965.
[2] L.R. Snyder, J.J. Kirkland, J.W. Dolan, *Introduction to Modern Liquid Chromatography*, 3rd ed., Wiley, 2010.
[3] V.B. Di Marco, G.G. Bombi, "Mathematical functions for the representation of chromatographic peaks," *J. Chromatogr. A* **931** (2001) 1–30.
[4] J.P. Foley, J.G. Dorsey, "Equations for chromatographic peak asymmetry," *Anal. Chem.* **56** (1984) 517–520.
[5] P.H.C. Eilers, H.F.M. Boelens, "Baseline correction with asymmetric least squares smoothing," 2005 (preprint/tech note, widely cited in spectral/chromatographic baseline correction).
[6] A. Felinger, *Data Analysis and Signal Processing in Chromatography*, Elsevier, 1998.
[7] M.F. Wahab, D.W. Armstrong, D.C. Patel, "Evaluation of chromatographic peak asymmetry," *LCGC Europe* **30** (2017) 670–678.