HΣ-GMA : High-Fidelity Generative Monte Carlo Approximation

DigWise Technology Corporation, Ltd.

1. 數據分佈解析：機率密度 (PDF)、累積密度 (CDF) 與分位數函數 (PPF)

在數據科學與統計分析中，理解數據分佈至關重要。除了均值與變異性，更需關注數據在不同數值範圍內的分佈，通常透過機率密度函數（PDF）與累積密度函數（CDF）描述。PDF 表示某數值的相對可能性，例如正態分佈的數據多集中於均值附近，而 CDF 則顯示小於或等於某數值的累積機率，並呈單調遞增特性。為了直觀理解 PDF 與 CDF，我們從多維矽晶特徵中選取一個維度（降維），並以帶有隨機噪聲的正弦波數據來模擬其分佈，如 Example 1 所示。進一步地，透過線性縮放（Normalization）將值域映射至 [100, 200]，這在數據預處理、特徵縮放與標準化中常見，有助於維持相對分佈結構，統一數據尺度。

Example 1 KDE, PDF and CDF

```
# KDE, PDF and CDF
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats

x = np.linspace(0,10,1000)
y = np.sin(x) + np.random.randn(len(x))*0.2
y = np.interp(y, (y.min(), y.max()), (100, 200)) # scale to [100, 200]
t = np.linspace(y.min(), y.max(), 100) # range of y
pdf = stats.gaussian_kde(y, bw_method=0.1)(t)
cdf = np.cumsum(pdf)/pdf.sum()

# PDF, KDE and CDF
plt.figure(figsize=(10,5))
ax1 = plt.subplot(121)
ax1.scatter(x, y, alpha=0.5)
ax1.set_ylabel('Value')
ax1.set_xlabel('Position')
ax2 = plt.subplot(122)
ax2.hist(y, bins=20, alpha=0.4, density=True, label='Histogram')
ax2.plot(t, pdf, lw=2, alpha=0.4, c='k', label='PDF(KDE)')
ax2.set_xlabel('Value')
ax2.set_ylabel('Density')
ax2.set_ylim(0)
ax2.legend(loc='upper left')
ax3 = plt.twinx(ax2)
ax3.plot(t, cdf, lw=5, alpha=0.4, c='g', label='CDF(PDF(KDE))')
ax3.set_ylabel('CDF')
ax3.set_ylim(0)
ax3.legend(loc='lower right')
plt.suptitle(f'{len(x):,} samples')
plt.tight_layout()
```

如圖 Figure 1 所示，由於直方圖（Histogram）將數據離散化，為了更平滑地近似數據分佈，我們使用核密度估計（KDE）來估算概率密度函數（PDF，淺黑色直方圖的輪廓線），並以累積概率曲線呈現 CDF（淺綠色線），以判斷數據的集中程度與累積概率。KDE 作為非參數估計方法，透過核函數提供更連續且精確的密度估計，幫助深入理解數據分佈。

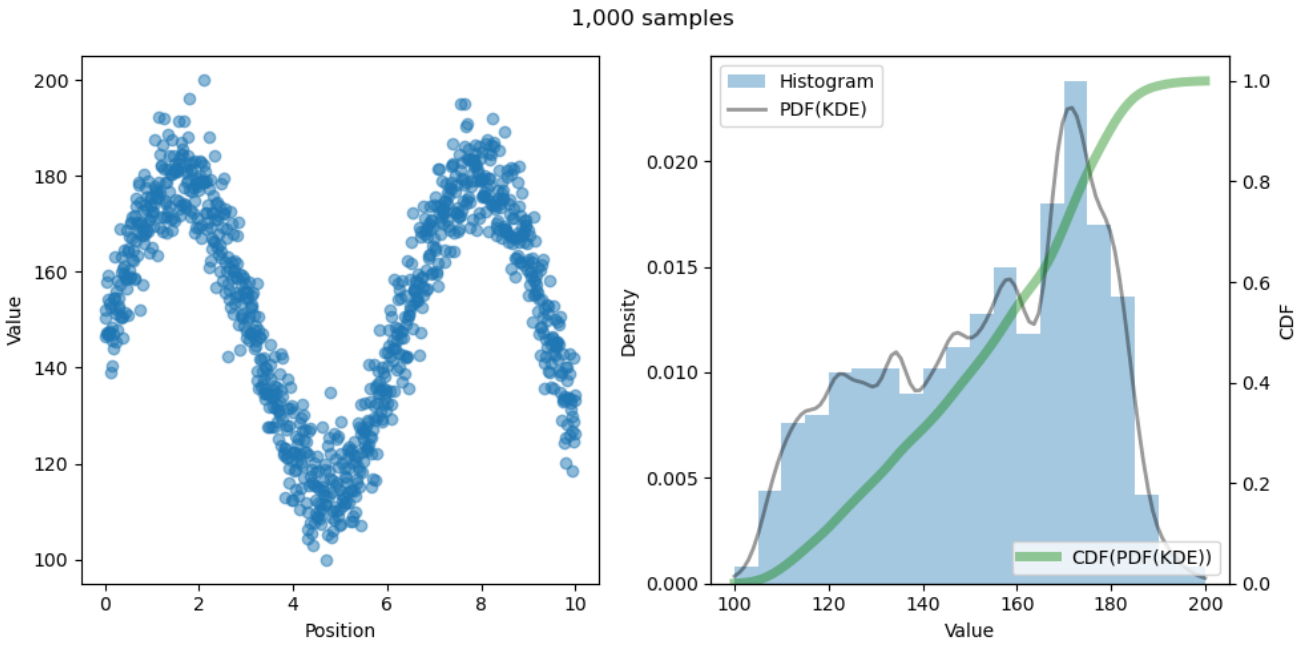

Figure 1 隨機噪聲正弦波數據的分佈及其 PDF 與 CDF

除了直方圖與 KDE，經驗累積分佈函數（ECDF）透過排序樣本並計算累積頻率，能精確呈現數據全貌，且具連續性。當數據量足夠大時，ECDF 可穩定反映真實分佈，適用於大規模數據分析，並可結合其他方法提供更全面的理解。

PPF（Percent-Point Function），即分位數函數，是 CDF 的反函數，可根據給定機率（如 10%、50%、90%）返回對應的數據點。透過 np.interp 進行反查詢計算 PPF，我們可以從 CDF（或 ECDF）對應特定機率百分位（如 10%、50%、90%）獲取數值，如圖 Figure 2 所示。這一過程可通過 Example 2 程式碼來實現。

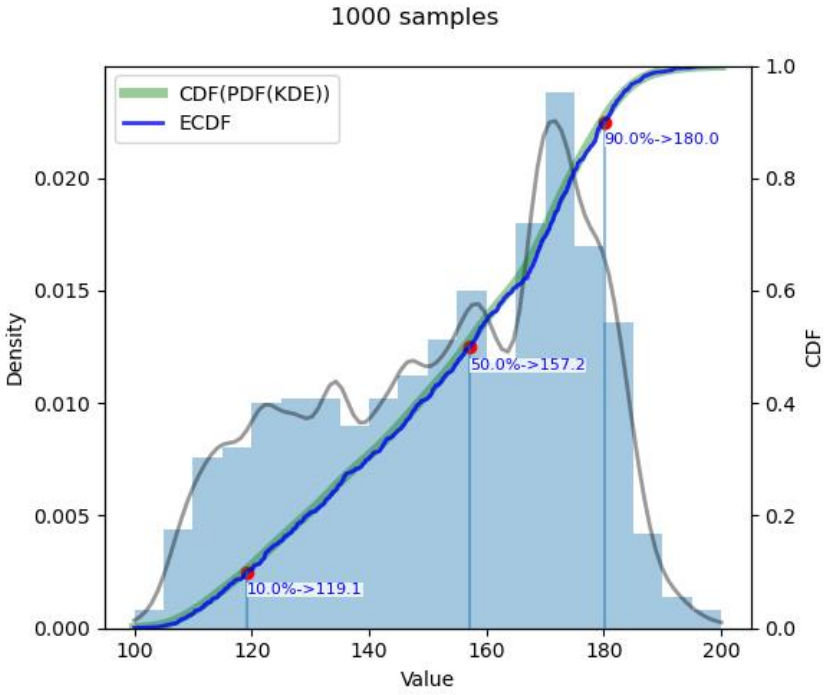

Figure 2 CDF, ECDF and PPF

Example 2 PPF: Retrieve Data Value from CDF Probability

```
# Empirical CDF directly computes percentiles based on sorted y
ecdf = (stats.rankdata(sorted(y), method='average')-0.5) / len(y)

percentiles = [0.1, 0.5, 0.9] # 10%, 50%, 90%
#levels = np.interp(percentiles, cdf, t)
levels = np.interp(percentiles, ecdf, sorted(y))

plt.figure(figsize=(6,5))
ax1 = plt.subplot(111)
ax1.hist(y, bins=20, alpha=0.4, density=True, label='Histogram')
ax1.plot(t, pdf, lw=2, alpha=0.4, c='k', label='PDF(KDE)')
ax1.set_xlabel('Value')
ax1.set_ylabel('Density')
ax1.set_ylim(0)

ax2 = plt.twinx(ax1)
ax2.plot(t, cdf, lw=5, alpha=0.4, c='g', label='CDF(PDF(KDE))')
ax2.plot(sorted(y), ecdf, lw=2, alpha=0.8, c='b', label='ECDF')
ax2.scatter(levels, percentiles, c='r')
for l,p in zip(levels,percentiles):
    ax2.text(l, p-0.04, f'{p*100:.1f}%->{l:.1f}', fontsize=8, c='b',
        bbox={'facecolor':'w', 'edgecolor':'none', 'pad':0, 'alpha':0.8})
    ax2.axvline(l, ymin=0, ymax=p, alpha=0.5)
ax2.set_ylabel('CDF')
ax2.set_ylim(0, 1)
ax2.legend(loc='upper left')
plt.suptitle(f'{len(x)} samples')
plt.tight_layout()
```

2. High-Sigma 邊界的不確定性：數據稀疏對極端值估計的影響

如圖 Figure 3 所示，當一維數據分佈（v1）偏離正態分佈時，基於高斯分佈的 3σ邊界（紅色邊界）與基於原始數據的排列所得到的分位數（quantile）3σ邊界（藍色邊界）可能存在顯著偏差。此時，我們需要依賴真實的累積分佈函數（CDF）並通過 PPF 反推對應的數據邊界。

當數據採樣點不足時（例如，從 600 點降至 300 點時，v2），高 Sigma 邊界的可靠性會降低。極端值的估計依賴於足夠的樣本數，當樣本數不足時，CDF 的高尾部變得稀疏，導致 PPF 在高 Sigma 區域的計算不穩定，從而影響統計推斷的準確性（如橙色邊界所示）。在這種情況下，傳統基於正態分佈的邊界可能顯示出偏差。

當採樣點不足時，傳統基於正態分佈的邊界可能顯示偏差，此時可使用 KDE（核密度估計）來估算 CDF 並反推邊界，提供較為平滑的處理方式。雖然 KDE 能捕捉數據的真實分佈並在極端值區域保持一致性，但在樣本數量極為稀疏的情況下，其可靠性仍需謹慎評估。

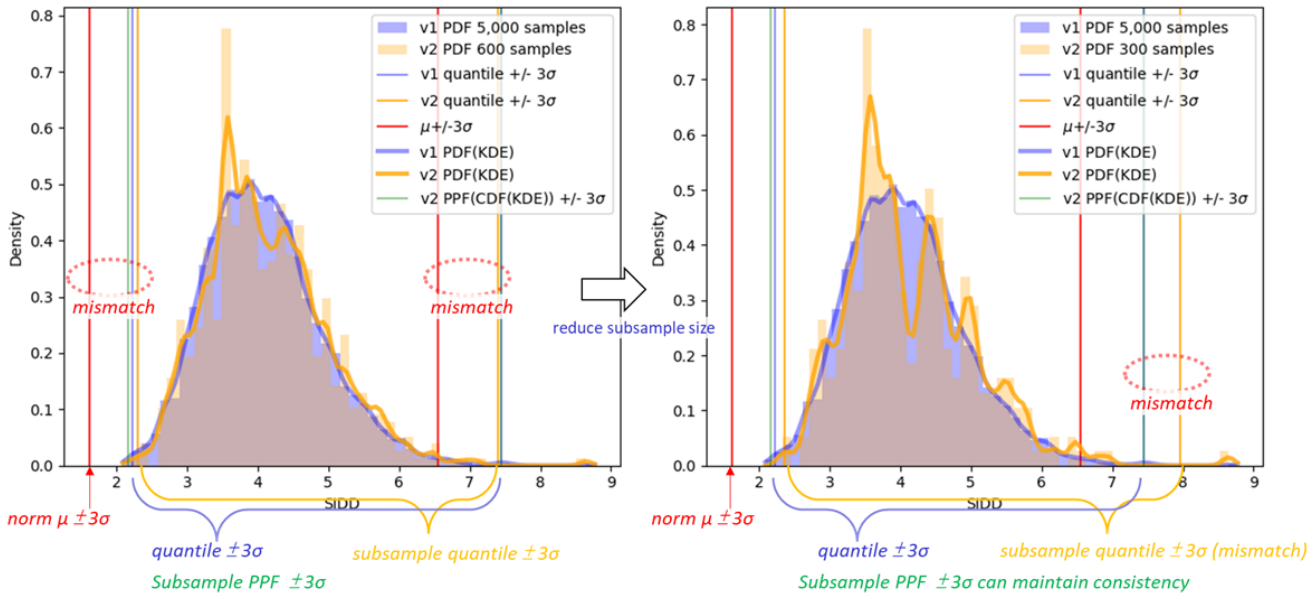

Figure 3 數據稀疏導致 High-Sigma 邊界不確定性

Example 3 這段程式說明在樣本數不足（例如從大量資料中隨機取出 300 筆子樣本）時，對高 σ（如 3σ）邊界的估計會出現明顯偏差。儘管原始資料的機率分佈（如對數常態）較為穩定，但

在極端值區域，稀疏的數據點會使估計出的邊界與原始分佈產生明顯差異，凸顯出在高 σ 區間，稀疏性對極值預測的敏感性與不確定性。

當樣本數不足時，傳統以正態分佈假設的 mean ± 3σ 邊界更顯不可靠。即使進行高 σ 模擬或外插，所推估的邊界亦難以反映真實分佈，特別是在數據呈偏態或長尾特性時，易低估極端值風險。相較之下，透過分位數估算或 KDE 重建 CDF 雖仍受限於樣本量，但能在不依賴特定分佈假設下，提供較具信心水平與彈性的邊界推估，並降低系統性偏差風險。

Example 3 Uncertainty in High-Sigma Bounds

```
# The impact of data sparsity on extreme value estimation
def sigma_percentage(sigma):
    '''return left and right % boundary based on the specified sigma value'''
    return (stats.norm.cdf((-sigma, sigma))*100)

subn = 300 # from 600 to 300
v1 = np.random.lognormal(mean=0.0, sigma=0.2, size=5000)*4  # raw data
v2 = np.random.choice(v1, size=subn)  # subsample

q1 = np.quantile(v1, q=sigma_percentage(3)/100) # 3σ quantile of raw data v1
q2 = np.quantile(v2, q=sigma_percentage(3)/100) # 3σ quantile of subsample v2

plt.figure(figsize=(6,5))
ax1 = plt.subplot(111)
ax1.hist(v1, bins=50, alpha=0.3, density=True, color='b', label=f'v1 PDF {len(v1):,} samples')
ax1.hist(v2, bins=50, alpha=0.3, density=True, color='orange', label=f'v2 PDF {len(v2):,} samples')
ax1.axvline(q1[0], alpha=0.4, c='b')
ax1.axvline(q1[1], alpha=0.4, c='b', label='v1 quantile +/- 3$\sigma$')
ax1.axvline(q2[0], alpha=0.8, c='orange')
ax1.axvline(q2[1], alpha=0.8, c='orange', label='v2 quantile +/- 3$\sigma$')
ax1.axvline(v1.mean()-3*v1.std(), alpha=0.8, c='r')
ax1.axvline(v1.mean()+3*v1.std(), alpha=0.8, c='r', label='$\mu$+/-3$\sigma$')

# KDE, CDF, PPF
t = np.linspace(v1.min(), v1.max(), 100)
k1 = stats.gaussian_kde(v1, bw_method=0.1)(t) # PDF of raw data v1
k2 = stats.gaussian_kde(v2, bw_method=0.1)(t) # PDF of subsample v2
c1 = np.cumsum(k1)/k1.sum() # CDF of v1
c2 = np.cumsum(k1)/k2.sum() # CDF of v2
p1 = np.interp(sigma_percentage(3)/100, c1, t) # v1 PPF
p2 = np.interp(sigma_percentage(3)/100, c2, t) # v2 PPF

ax1.plot(t, k1, lw=3, alpha=0.4, c='b', label='v1 PDF(KDE)')
ax1.plot(t, k2, lw=3, alpha=0.8, c='orange', label='v2 PDF(KDE)')
ax1.axvline(p2[0], alpha=0.4, c='g')
ax1.axvline(p2[1], alpha=0.4, c='g', label='v2 PPF(CDF(KDE)) +/- 3$\sigma$')
ax1.set_xlabel('SIDD')
ax1.set_ylabel('Density')
ax1.set_ylim(0)
ax1.legend()
plt.tight_layout()
```

3. 合成虛擬晶圓數據

晶圓層級的結構性（非隨機）均勻性對數據的宏觀影響，使得真實數據通常不僅呈現偏態或簡單的對數正態分佈。首先，建立虛擬晶圓座標並保留位於晶圓中心半徑內的數據。Example 4 展示了虛擬晶圓座標的生成，晶圓尺寸為 65×65 個樣本。

volcanoTiltGaussian 函式利用半徑、相對位移等參數，通過多個高斯核合成虛擬矽晶圓數據，模擬並捕捉 RO（環形振盪器，Ring Oscillator）和 SIDD（靜態 IDDQ）數據分佈的特徵。調整這些參數，我們可以生成反映不同晶圓廠製程變異性的數據，展現各種均勻性風味（uniformity）。同時，我們將數據進行正規化處理，使其符合近似標準正態分佈（~N(0, 1)），以便於後續處理。

Example 4 Synthetic Virtual Wafer Silicon

```
# Generative virtual silicon data for simulation
import pandas as pd

def genWaferXY(w=65, h=65):
    ix, iy = np.linspace(1, w, w), np.linspace(1, h, h)
    gx, gy = np.meshgrid(ix, iy)
    cx, cy = np.ceil(w/2).astype(int), np.ceil(h/2).astype(int)
    cr = min(cx, cy)
    r = np.sqrt((gx-cx)**2 + (gy-cy)**2)
    mask = (r <= cr)
    x, y, r = gx[mask], gy[mask], r[mask]
    return x, y, r

def volcanoTiltGaussian(x, y, r, h1=1, s1=15, h2=1, s2=5, tilt_factor=0.2, tilt_angle=0):
    g1 = h1 * np.exp(-r**2/(2*s1**2))
    g2 = h2 * np.exp(-r**2/(2*s2**2))
    z = g1 - 0.5*g2
    theta = np.arctan2(x, y)  # map xy to [-π, π]
    z = z + tilt_factor * np.tanh(theta-tilt_angle)
    return np.interp(z, [z.min(), z.max()], [0, 1])

def logNormScale(u, s): # normal to log-normal
    log_u = np.log(u**2 / np.sqrt(s**2 + u**2))
    log_s = np.sqrt(np.log(1 + s**2 / u**2))
    return log_u, log_s

# generate a base wafer surface with Gaussian noise
x, y, r = genWaferXY()

# the uniformity signature of different fabs
#z0 = volcanoTiltGaussian(x, y, r, h1=0.5, s1=3, h2=2, s2=40, tilt_factor=-0.5) # T-like
z0 = volcanoTiltGaussian(x, y, r, h1=1.0, s1=15, h2=1, s2=5, tilt_factor=-0.5) # S-like
ϵ0 = 1/z0.std()*(z0-z0.mean()) # z0 to ~N(0,1)
```

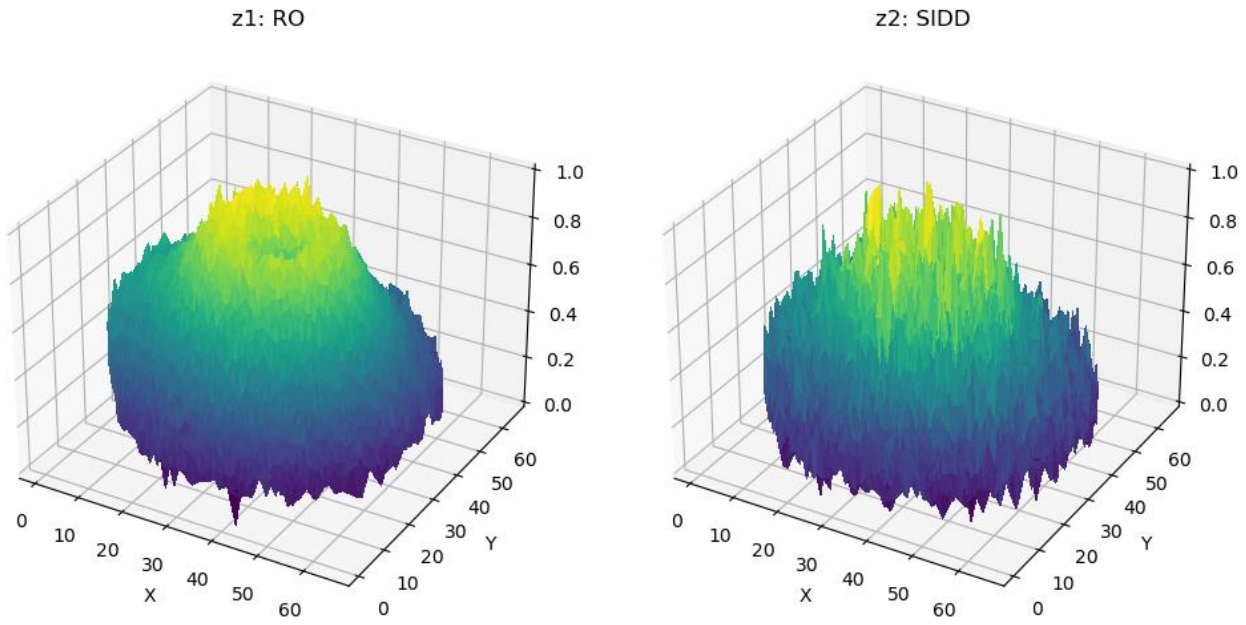

Figure 4 Synthetic Virtual Silicon Wafer

如圖 Figure 4，晶圓層級的數據曲面可能呈現類似傾斜火山錐的結構。其中，RO（環形振盪器）符合偏態高斯分佈，而 SIDD（靜態 IDDQ）則呈現對數常態分佈。此外，RO 與 SIDD 之間具有高度相關性（皮爾森相關係數 ρ = 0.9）。

在 Example 5 中，我們首先為 RO 添加高斯噪聲，使其符合標準正態分佈 $N(0,1)$，並進行標準化。接著，通過條件高斯建模（conditional Gaussian modeling）生成與目標相關性的 log(SIDD) 數據。隨後，根據 SIDD 實際量測的均值 mu 和標準差 sigma，使用 logNormScale(mu, sigma) 計算對數空間中的均值 log_mu 和標準差 log_s，並將生成的數據轉換回對數正態分佈，最後映射至 [0, 1] 以便比較。結果如圖 Figure 5 所示。

Example 5 Generating RO & SIDD Distributions

```
# add Gaussian noise to the feature surface (RO ~ N(0,1))
ϵ1 = np.random.normal(0, 1, len(z0)) # noise ~N(0,1)
z1 = (ϵ0 + 0.2*ϵ1) # RO ~N(0,1)
z1 = z0.std()*z1 + z0.mean() # back to z0 scale
z1 = np.interp(z1, [z1.min(), z1.max()], [0, 1]) # to [0, 1] for comparison

# generate SIDD ~logN(0,1) with a correlation of 0.9 with z1
ρ  = 0.9
ϵ1 = 1/z1.std()*(z1-z1.mean()) # z1 to ~N(0,1)
ϵ2 = np.random.normal(0, 1, len(z1)) # noise ~N(0,1)
```

```
# conditional Gaussian standarization
z2 = ρ*ϵ1 + np.sqrt(1-ρ**2)*ϵ2 # log(SIDD) ~N(0,1)

mu, sigma = 0.5, 0.1 # measured mu and sigma of SIDD
log_u, log_s = logNormScale(mu, sigma) # retarget to log-normal scale
z2 = np.exp(log_s*z2 + log_u) # to z2 scale ~logN(u2, s2)
z2 = np.interp(z2, [z2.min(), z2.max()], [0, 1]) # to SIDD scale

d = pd.DataFrame(np.c_[z1,z2], columns=['ROu','SIDD'])

# feature surface
plt.figure(figsize=(10,5))
ax1 = plt.subplot(121, projection='3d', title='z1')
ax1.plot_trisurf(x, y, z1, cmap='viridis', antialiased=False,linewidth=0,edgecolors='none')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax2 = plt.subplot(122, projection='3d', title='z2')
ax2.plot_trisurf(x, y, z2, cmap='viridis', antialiased=False,linewidth=0,edgecolors='none')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
plt.tight_layout(rect=(0,0,1,0.95))

# scatter & histogram
plt.figure(figsize=(10,5))
ax1 = plt.subplot(121)
ax1.scatter(z1, z2, alpha=0.5)
ax1.set_xlabel('z1: RO')
ax1.set_ylabel('z2: SIDD')
ax2 = plt.subplot(122)
ax2.hist(z1, bins=30, density=True, alpha=0.4, label='z1: RO')
ax2.hist(z2, bins=30, density=True, alpha=0.4, label='z2: SIDD')
ax2.legend()
plt.tight_layout()
```

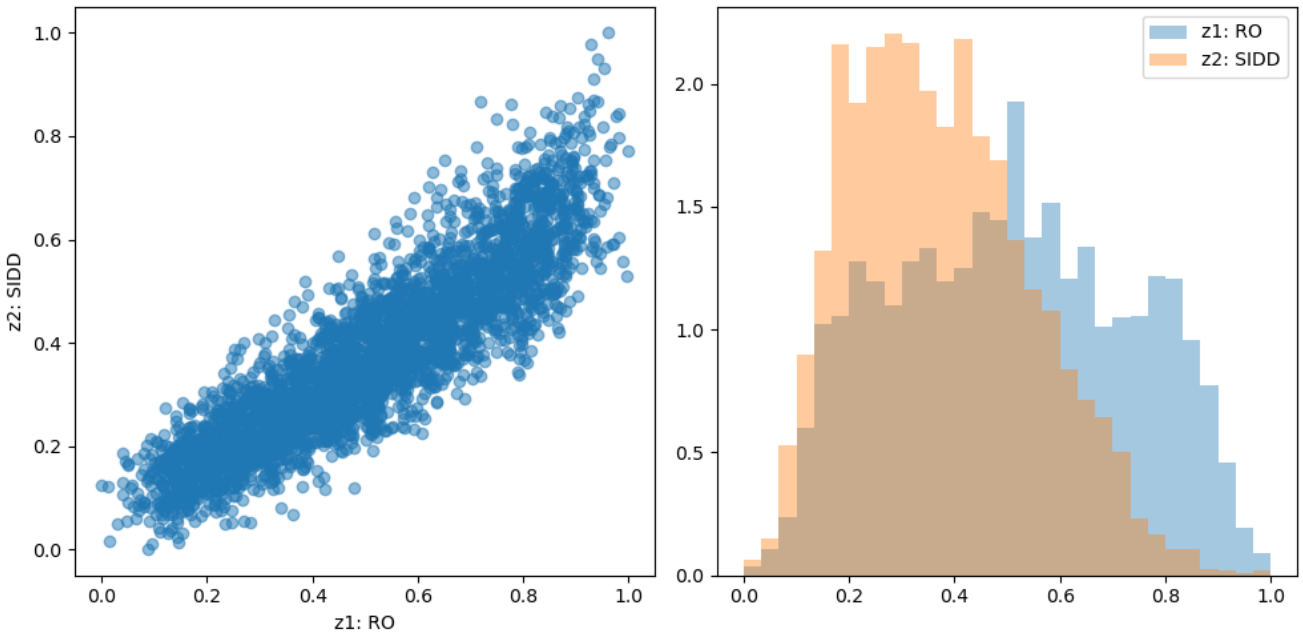

Figure 5 Scatter Plot and Distribution of RO and SIDD

Example 6 透過 2D 散點圖呈現了在假設各維度為高斯分佈的情況下，基於 3σ 邊界以及 quantile 3σ 邊界的比較，並在高維空間中繪製對應的等高線。儘管在進行高 sigma 仿真時，這些邊界的實際意義可能會有所減弱，特別是當資料偏態或具有長尾特性時，這些邊界依然提供了不同方法下對極端值的估算，並反映了在不同假設下，對於高維數據的風險邊界的不同評估。

Example 6 2D Feature Density and 3σ Boundary

```
# Individual Gaussian 3σ – Pessimistic Risk
def featureKDE(v, res=100):
    kde = stats.gaussian_kde(v)
    t = np.linspace(min(v)-v.std(),max(v)+v.std(),res)
    p = kde(t)
    return t,p

def featureKDE2D(x,y, res=50):
    kde = stats.gaussian_kde(np.vstack([x, y]))
    ix,iy = np.linspace(min(x),max(x),res), np.linspace(min(y),max(y),res)
    gx,gy = np.meshgrid(ix,iy)
    tx,ty = map(np.ravel,[gx,gy])
    p = kde(np.vstack([tx,ty]))
    return gx,gy,p.reshape(gx.shape)

def featureScatterCDF2D(dt,fx,fy): # XY CDF 2D
    x,y = abs(dt[[fx,fy]].values.T)
    percentiles = sigma_percentage(3)/100

    tx,px = featureKDE(x,res=100) # X PDF(ROu)
    ty,py = featureKDE(y,res=100) # Y PDF(SIDD)
    bx = np.quantile(x, q=sigma_percentage(3)/100)
    by = np.quantile(y, q=sigma_percentage(3)/100)

    gx,gy,pz = featureKDE2D(x,y, res=50) # XY PDF
    cdf = np.cumsum(sorted(pz.ravel()))/pz.sum()
    levels = np.interp(percentiles, cdf, sorted(pz.ravel()))  # map % to density level
    ctags = {l:f'{p*100:.2f}%' for l,p in zip(levels,percentiles)}

    fig,axes = plt.subplots(2,2,figsize=(6,6),
        gridspec_kw={'width_ratios':[4,1],'height_ratios': [4,1]})

    # ROu & SIDD 2D scatter and contour
    axes[0,0].scatter(x,y,alpha=0.5,s=5,label=f'{len(x):,} samples')
    axes[0,0].axvline(bx[0],c='b',alpha=0.3)
    axes[0,0].axvline(bx[1],c='b',alpha=0.3,label=f'{fx} quantile 3$\sigma$={bx}')
    axes[0,0].axhline(by[0],c='r',alpha=0.3)
    axes[0,0].axhline(by[1],c='r',alpha=0.3,label=f'{fy} quantile 3$\sigma$={by}')
    axes[0,0].set_xlabel(fx)
    axes[0,0].set_ylabel(fy)
    axes[0,0].grid(alpha=0.4)

    # shaded region below the -3σ boundary
    axes[0, 0].contourf(gx, gy, pz, levels=[0, levels[0]], colors=['gray', None], alpha=0.2)

    cs = axes[0,0].contour(gx,gy,pz, levels=levels, colors=['r','b','g'], alpha=1)
    cl = plt.clabel(cs, inline=True, fontsize=10, colors=['r','b','g'], fmt=ctags)
    [txt.set_bbox(dict(facecolor='white', edgecolor='none', pad=1, alpha=0.6)) for txt in cl]
```

```
    # ROu distribution
    axes[1,0].fill_between(tx,px,0, lw=0, alpha=0.3)
    axes[1,0].axvline(bx[0],c='b',alpha=0.3)
    axes[1,0].axvline(bx[1],c='b',alpha=0.3)

    # SIDD distribution
    axes[0,1].fill_betweenx(ty,0,py, lw=0, alpha=0.3)
    axes[0,1].axhline(by[0],c='r',alpha=0.3)
    axes[0,1].axhline(by[1],c='r',alpha=0.3)

    axes[0,1].set_ylim(axes[0,0].get_ylim())  # Match Y-axis with main plot
    axes[1,0].set_xlim(axes[0,0].get_xlim())  # Match X-axis with main plot
    axes[0,0].legend()
    axes[1,1].set_visible(False)
    plt.suptitle(f'3σ Contour, {len(d):,} samples')
    plt.tight_layout()

d = pd.DataFrame(np.c_[z1,z2], columns=['ROu','SIDD'])
featureScatterCDF2D(d, 'ROu', 'SIDD')
```

如圖 Figure 6 所示，我們透過虛擬晶圓數據比較高維數據的聯合分佈與假設每個維度獨立且符合常態分佈時所得到的高 sigma 邊界之間的差異。由於採樣不足或基於正態分佈假設的偏差，二維數據降維後得到的機率密度邊界可能不準確。當將這一問題擴展到二維散點密度，甚至更高維度時，誤差和差異變得更加明顯。圖中顯示，假設各維度獨立高斯分佈的隨機模擬可能涵蓋實際不存在的區域（灰色區域），這樣得到的高 sigma 邊界不僅消耗大量計算資源，並不可靠且過於悲觀。

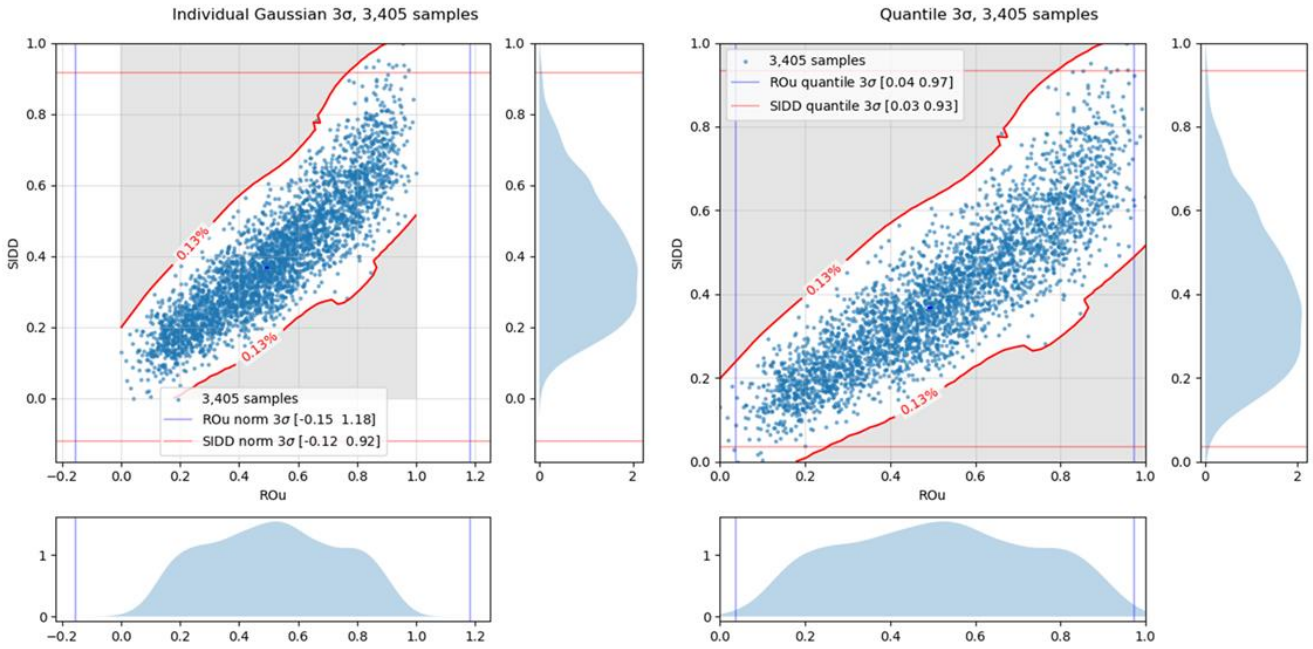

Figure 6 Individual Gaussian and Quantile-based 3σ Boundaries

4. Generative Monte Carlo Approximation (GMA)

在樣本數受限的情況下，Generative Monte Carlo Approximation (GMA) 提供一種有效估算高 sigma 邊界的方法。透過少量原始資料（例如僅 3,405 筆實際測量樣本），利用 Gaussian Mixture Model（GMM）進行生成式學習，可在合理的計算資源下擴增出數萬筆樣本（如 100K parts，去除超出 [0, 1] 範圍），以模擬高維特徵空間的完整機率分佈，如 Example 7 所示。圖 Figure 7 進一步將 -3σ 邊界的差異進行視覺化呈現。

Example 7 Synthetic Sample Generation via GMM

```
# Gaussian Mixture Model (GMM)
from sklearn.mixture import GaussianMixture

def GMM(v, n_samples=1000, n_components=5, scale=1e2):
    gmm = GaussianMixture(n_components=n_components, covariance_type='full', random_state=0)
    gmm.fit(v*scale)
    dv, _ = gmm.sample(n_samples=n_samples) # generator
    return dv/scale

dt = pd.DataFrame(np.c_[z1,z2], columns=['ROu','SIDD']) # raw data

# generative samples based on GMM
samples = GMM(dt.values, n_samples=100000, scale=1e3) # 100K synthetic samples
dv = pd.DataFrame(samples, columns=['ROu','SIDD']) # convert to dataframe
```

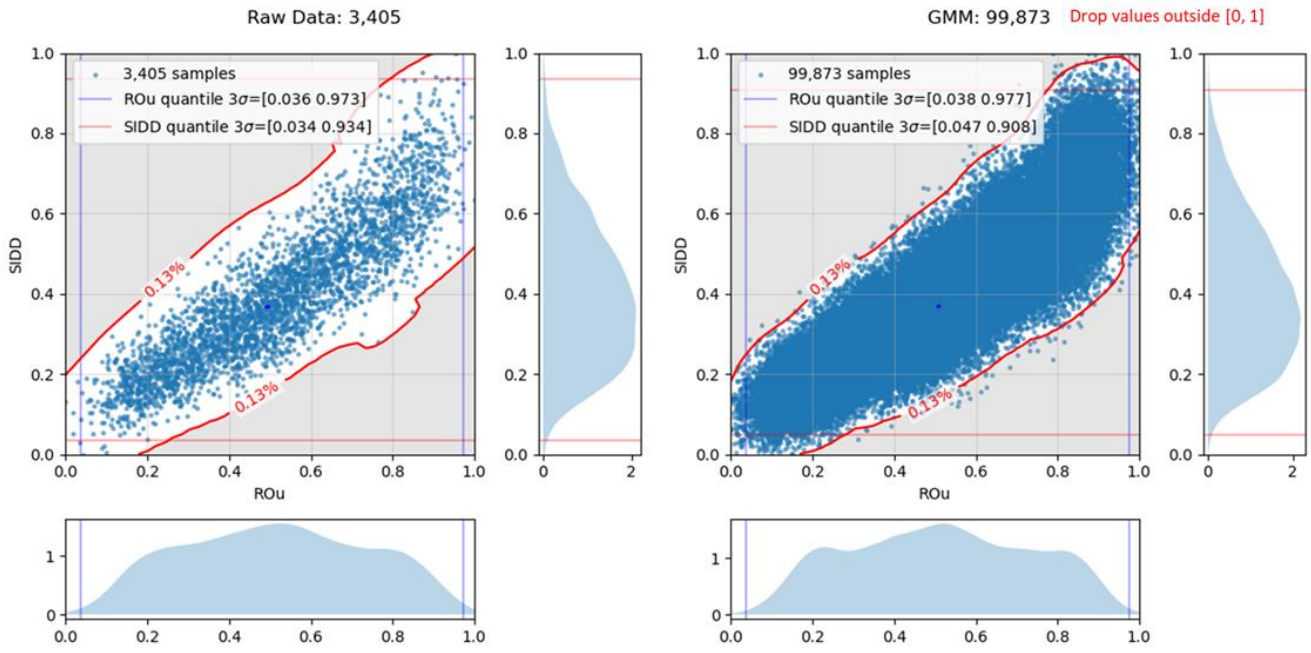

Figure 7 Visualizing -3σ Feature Contours: Raw vs. GMM 10K Synthetic

進一步地，Example 8 透過高維特徵的 2D 散點密度輪廓，推導出更具代表性的 high-sigma 區域邊界。與傳統基於維度獨立性與高斯假設的高 sigma 外插結果相比，GMA 在高維空間中的估計不僅更符合實際分佈，還能保留潛在的非對稱性和尾部風險，實現更具成本效益且信心水準較高的可靠性預測。

Example 8 Generative Modeling for Rare Event Estimation

```
# Generative Modeling for Rare Event Estimation
def gmmDensityContour(dt, num=10000, n_components=5, sigmas=[3]):
    samples = GMM(dt.values, n_samples=num, n_components=n_components, scale=1e3)
    dv = pd.DataFrame(samples, columns=['ROu','SIDD']) # raw data
    dv = dv[ dv['ROu'].between(0,1) & dv['SIDD'].between(0,1) ] # drop outliers

    pL = sorted([sigma_percentage(v)[0]*2/100 for v in sigmas])

    plt.figure(figsize=(6,6))
    ax = plt.subplot(111)
    ax.scatter(*dt.values.T, s=5, c='b', alpha=0.4, label=f'raw data: {len(dt):,}', zorder=1)
    ax.scatter(*dv.values.T, s=2, c='r', alpha=0.2, label=f'synthetic: {len(dv):,}', zorder=0)

    for d,tag,c in [(dt,'raw data',('b','g')), (dv,'synthetic',('r','orange'))]:
        x,y = d.values.T
        gx,gy,pz = featureKDE2D(x,y, res=50) # XY PDF
        cdf = np.cumsum(sorted(pz.ravel()))/pz.sum()
        levels = np.interp(pL, cdf, sorted(pz.ravel()))
        ctags = {l:f'{s}$\sigma$' for l,s in zip(levels,sigmas)}
        cs = ax.contour(gx, gy, pz, levels=levels, colors=c, alpha=0.6, linewidths=3)
        cl = plt.clabel(cs, inline=True, fontsize=10, colors=c, fmt=ctags)
        [txt.set_bbox(dict(facecolor='white', edgecolor='none', pad=1, alpha=0.9)) for txt in cl]

    ax.set_xlabel('ROu')
    ax.set_ylabel('SIDD')
    ax.set_xlim([0,1])
    ax.set_ylim([0,1])
    plt.legend()
    plt.suptitle(f'Feature Density Contour (GMM: {dv.shape[0]:,} samples), sigmas: {sigmas}')
    plt.tight_layout()

dt = pd.DataFrame(np.c_[z1,z2], columns=['ROu','SIDD']) # raw data

# high-sigma boundary evaluation
gmmDensityContour(dt, 100000, sigmas=[3])
gmmDensityContour(dt, 100000, sigmas=[4.5])
```

圖 Figure 8 展示了利用 GMM 模型對高維 (2D) 特徵進行建模，生成 100K 部件並去除超出 [0, 1] 範圍的資料後進行的 high-sigma 邊界分析。

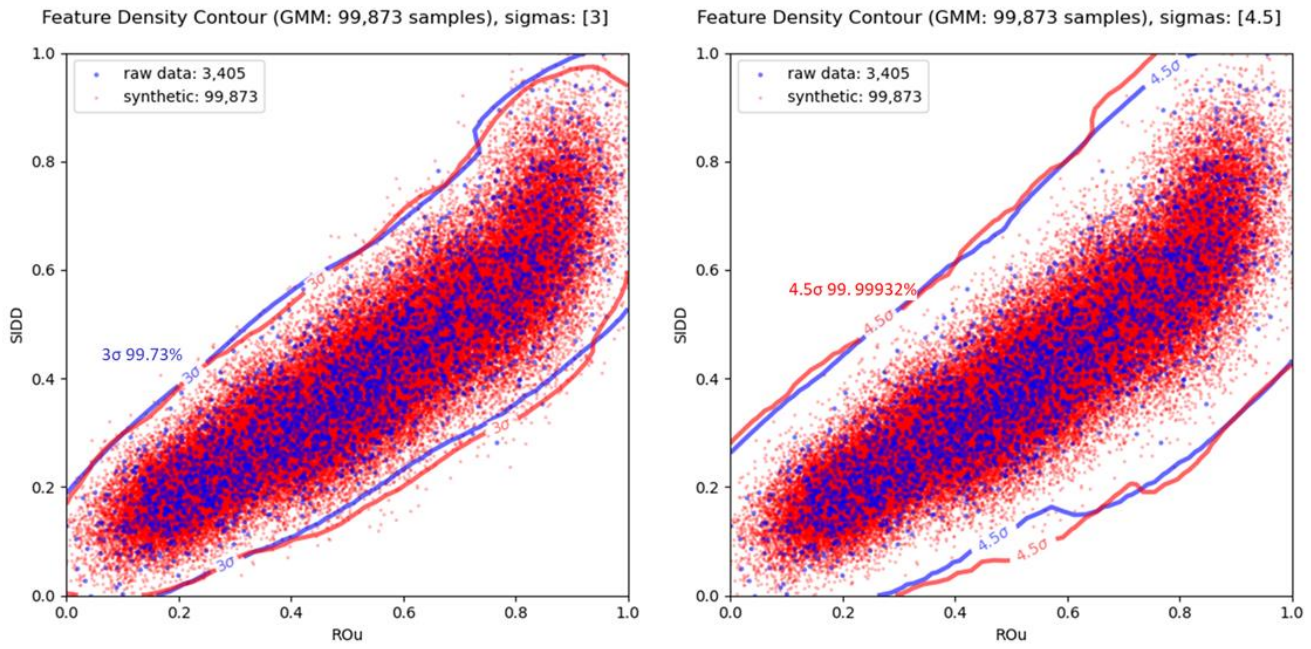

Figure 8 Generative Modeling for Rare Event Estimation

綜合上述方法，Generative Monte Carlo Approximation (GMA) 提供了一個強大且具成本效益的工具，尤其適用於樣本數有限的情境下。透過 GMM 模型生成大量合成數據，GMA 能夠有效地在高維空間中捕捉資料的分佈特性，並推導出更準確的 high-sigma 邊界，相比傳統的高斯假設或過度依賴降維方法，能更真實地反映資料的非對稱性與尾部風險。在實際應用中，這不僅能提高預測的準確度與可靠性，還能顯著降低計算資源的消耗，使得 GMA 成為高維資料分析中不可或缺的有效工具。這樣的發展為各類高維數據處理及風險預測模型提供了更為穩健且高效的替代方案，未來有望在更多領域發揮其潛力。