Supporting Information For:

Solid-State Nanopore Sizing for cfDNA Sample Quality Control in Point-of-Need Sequencing

Muhammad Asad Ullah Khalid[1], Md. Ahasan Ahamed[1,2], Anthony J. Politza[3], and Weihua Guan[1*]

[1]Department of Intelligent Systems Engineering, Indiana University, Bloomington, Indiana 47408, United States

[2]Department of Electrical Engineering, Pennsylvania State University, University Park, Pennsylvania 16802, United States

[3]Department of Biomedical Engineering, Pennsylvania State University, University Park, Pennsylvania 16802, United States

* Corresponding Author's Email: guanw@iu.edu

Contents

Supplementary Figures ---------- 02

Supplementary Code ---------- 07

S1. Supplementary Figures

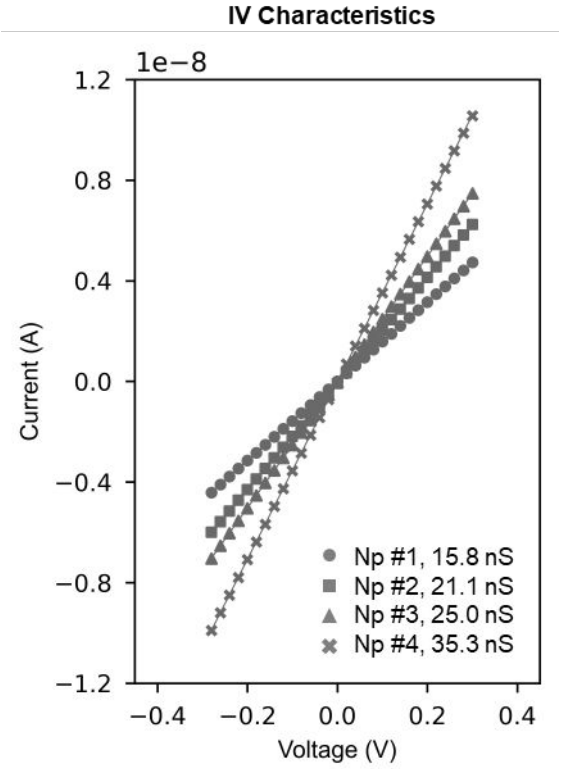


Figure S1. IV curves of four nanopores with different diameters.

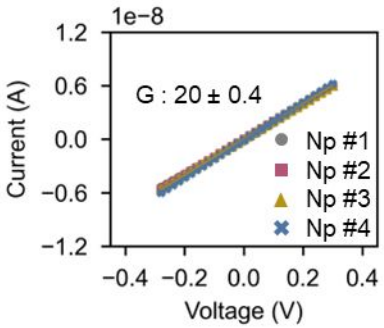


Figure S2. IV curves show negligible variations in open pore conductance values and hence estimated pore diameters (9.7 ± 0.8 nm).

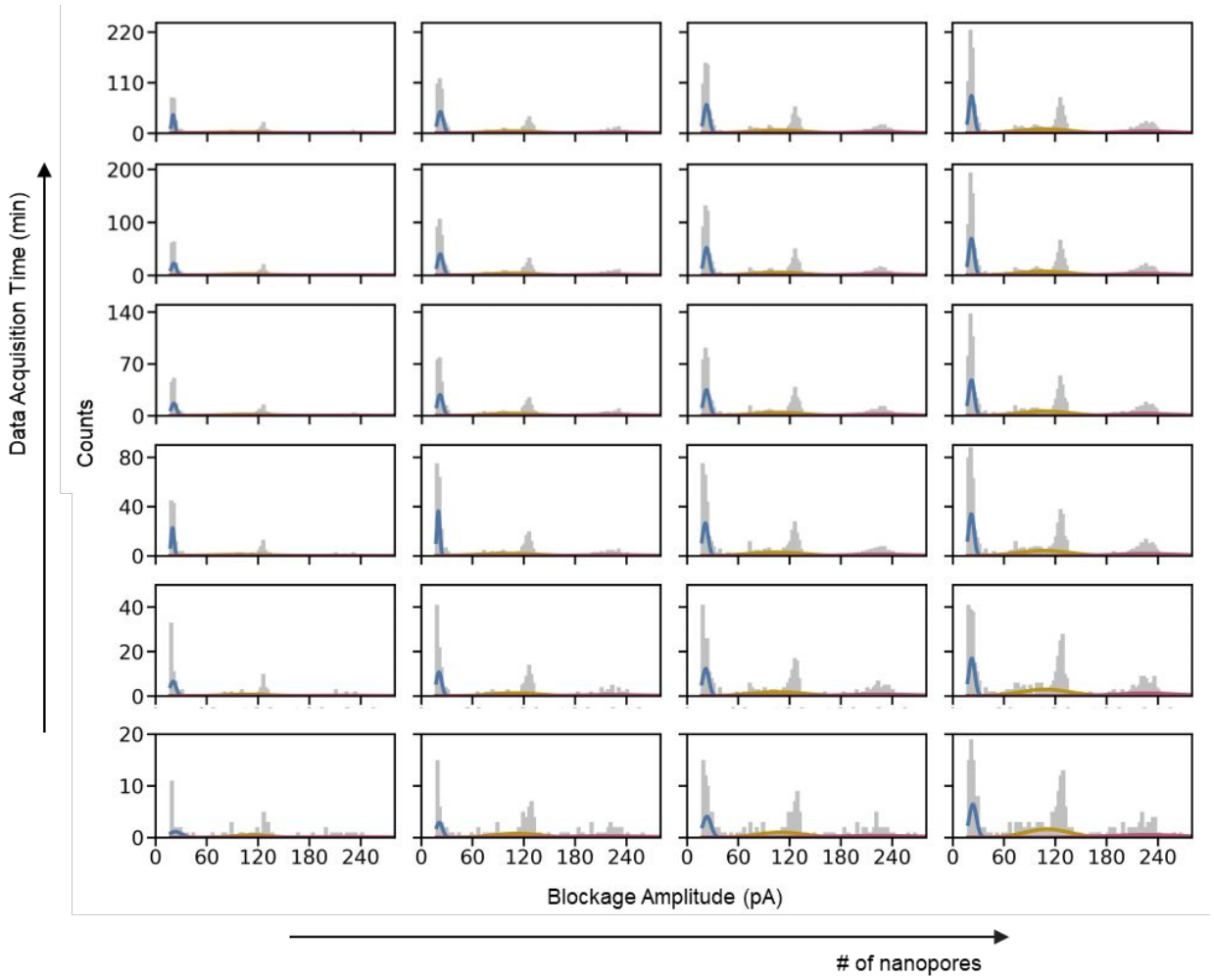


Figure S3. Counts' histograms and multi-gaussian fittings for twenty-four test combinations of data acquisition time (5, 10, 15, 20, 25, and 30 min) and number of nanopores (1, 2, 3, and 4).

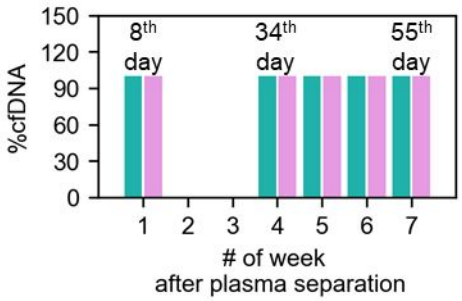


Figure S4. %cfDNA remained unaffected even after ~7 weeks post-plasma separation from blood sample.

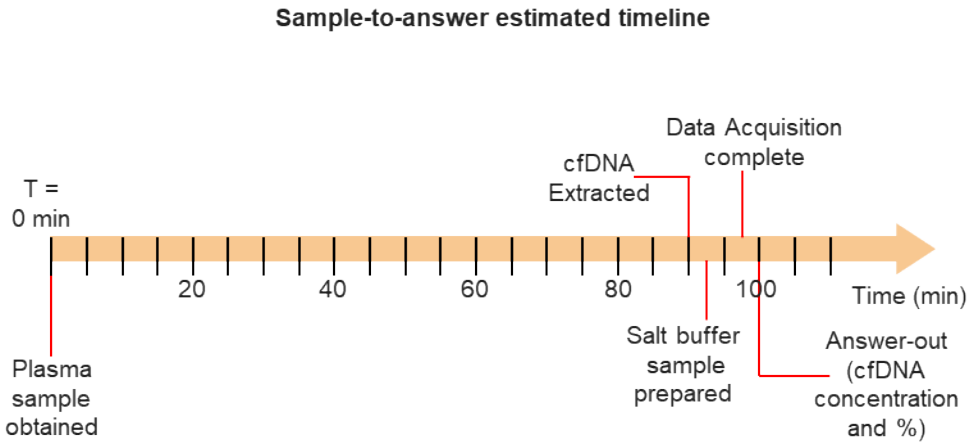


Figure S5. An estimated timeline for sample-to-answer workflow of nanopore based sample QC.

S2. Supplementary Code

Python script for plotting counts histograms of peak blockage amplitudes and respective multi-gaussian fitting.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.mixture import GaussianMixture

def plot_histogram_with_gaussian_mixture(file_path, sheet_name, num_components=3, bin_size=2.5,
                                         figsize=(1.98, 1.17), dpi=600, xlabel='Blockage Amplitude (pA)', ylabel='Counts',
                                         font_family='Arial', font_size=8, bar_color='silver',
                                         bar_width=5, fit_colors=['#B79315', '#4E75A3', '#B55475'],
                                         xlim=None, ylim=None, xticks=None, yticks=None, xtick_rotation=0, ytick_rotation=0, tick_pad=1):

# Load data and fit GMM
    df = pd.read_excel(file_path, sheet_name=sheet_name)
    data = df.iloc[:, 0]
    hist_values, bin_edges = np.histogram(data, bins=np.arange(min(data), max(data) + bin_size, bin_size))
    bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2
    gmm = GaussianMixture(n_components=num_components)
    gmm.fit(data.values.reshape(-1, 1))

# Generate points for the fitted Gaussians
    x_fit = np.linspace(min(data), max(data), 1000)
    logprob = gmm.score_samples(x_fit.reshape(-1, 1))
    responsibilities = gmm.predict_proba(x_fit.reshape(-1, 1))

# Determine the actual number of fitted components
```

```
    n_fitted_components = responsibilities.shape[1]

# Get labels (predictions) for each data point
    labels = gmm.predict(data.values.reshape(-1, 1))

# Calculate individual counts under each Gaussian
    counts = []
    for i in range(n_fitted_components):
        component_data = data[labels == i]
        counts.append(len(component_data))

# Plot histogram and Gaussians
    fig, ax = plt.subplots(figsize=figsize, dpi=dpi)
    ax.bar(bin_centers, hist_values, width=bar_width, color=bar_color, label='Histogram')
    for i in range(n_fitted_components):
        pdf = np.exp(logprob) * responsibilities[:, i]
        ax.plot(x_fit, pdf * len(data), color=fit_colors[i % len(fit_colors)], label=f'Gaussian {i+1}')

# Formatting
    if xlim:
        ax.set_xlim(xlim)
    if ylim:
        ax.set_ylim(ylim)
    if xticks is not None:
        ax.set_xticks(xticks)
    if yticks is not None:
        ax.set_yticks(yticks)
    ax.tick_params(axis='x', rotation=xtick_rotation, labelsize=font_size, pad=tick_pad)
    ax.tick_params(axis='y', rotation=ytick_rotation, labelsize=font_size, pad=tick_pad)
    ax.set_xlabel(xlabel, fontfamily=font_family, fontsize=font_size)
```

```
    ax.set_ylabel(ylabel, fontfamily=font_family, fontsize=font_size)
    plt.tight_layout()

# Output means, standard deviations, and counts of fitted Gaussians
    print("Fitted Gaussian parameters:")
    for i in range(n_fitted_components):
        print(f"Gaussian {i+1}:")
        print(f"  Mean: {gmm.means_[i][0]:.2f}")
        print(f"  Std Dev: {np.sqrt(gmm.covariances_[i][0][0]):.2f}")
        print(f"  Count: {counts[i]}")
    plt.show()

# Adjust file path and parameters as needed
file_path = r"File_Path "
sheet_name = 'Sheet1'
plot_histogram_with_gaussian_mixture(file_path, sheet_name, num_components=3,
                                     xlim=(0, 280), ylim=(0, 300),
                                     xticks=[0, 60, 120, 180, 240], yticks=[0, 100, 200, 300])
```