APPLICATION NOTE

Demystifying DGE-AUC Part 6: Utilizing multiwavelength data

Akash Bhattacharya
Beckman Coulter Life Sciences, 4510 Byrd Dr, Loveland, CO 80538

Abstract

This application note covers the use of multiwavelength (MWL) absorbance data in the context of density gradient equilibrium analytical ultracentrifugation or DGE-AUC. The data presented in this case study is on a system of drug-loaded (liposomal Doxorubicin, commercial name “Doxoves®”) and control liposomes[1]. The data is analyzed using a MATLAB script, which is included as an appendix. Data analysis includes some of the user driven operations shown in previous installments of this series: such as data truncation, noise filtering, etc.

Introduction

Background

The first application notes in this series (Demystifying DGE-AUC Part 1: Back to the Basics), introduced the technique of density gradient equilibrium analytical ultracentrifugation or DGE-AUC and compared it to the currently used techniques of Sedimentation Velocity (SV-AUC) and Sedimentation Equilibrium (SE-AUC)[2]. The second application note dove into details of the physics behind the formation of density gradients and the relationship of analytical and preparative experiments in this context (Demystifying DGE-AUC Part 2: Physical Principles and Links to Preparative DGUC). The third application note (Demystifying DGE-AUC Part 3: Sample Preparation) discussed gradient-forming materials (GFMs) in detail, showing how to prepare samples for DGE-AUC (including providing an easy to use Excel guide) and provided a guide to set up DGE-AUC experiments on the Optima AUC (including a table of data collection parameters). The fourth application note discussed data analysis using the Origin software package (Demystifying DGE-AUC Part 4: Fundamentals of Data Analysis) while the fifth application note covered various steps taken to optimize data quality (Demystifying DGE-AUC Part 5: Optimizing data quality).

Case Study: Liposomes

This application note will focus on a case study using liposomes[3-6]. Liposomes are a family of spherical, artificial, nanoparticle drug carrier systems composed of lipid bilayers[3-5]. The samples used are based on commercially available research grade (nonmedical) liposomal Doxorubicin. A schematic is shown in Figure 1. These are similar to the commercial medical grade product Doxil, a chemotherapy drug which encapsulates the active drug (Doxorubicin) in liposomes to prevent immune degradation of the drug and improve chances of it reaching the target cancerous tissue[7-9]. The context of this case study is an attempt to quantify the loading fraction of these liposomes. This is a similar question to finding the loading fraction of gene therapy vectors such as Adeno-Associated Virus (AAV)[10]. However, there are some interesting challenges as well as opportunities in this novel system.

The challenge comes from the stability (or lack of) of liposomes and other similar nanoparticle drug carriers compared to the robustness of AAV capsids[11,12]. DGE-AUC experiments on AAV capsids can be conducted using CsCl – the same GFM that is used in preparative DGUC experiments[13]. Liposomes are not purified using DGUC[14-16], therefore, we do not necessarily have a starting point for choosing an appropriate GFM (see the workflow in Demystifying DGE-AUC II: On gradient-forming materials). This will be addressed in the results section.

The opportunity comes from the UV-vis spectra of drug-loaded liposomes, wherein the drug molecule itself is also a chromophore. Thus, the drug-loaded liposomes will have additional absorption peaks beyond those which are also present in empty, or control liposomes. These additional absorption peaks can be used for detection of the drug molecule directly in AUC experiments. This is not equivalent to the true multiwavelength analysis which is carried out for AUC data collected for AAV systems[10]. However, this type of analysis does take advantage of multiwavelength data to assign identities to different peaks, even if the final quantitation of loading fraction is based entirely on single wavelength DGE-AUC scans.

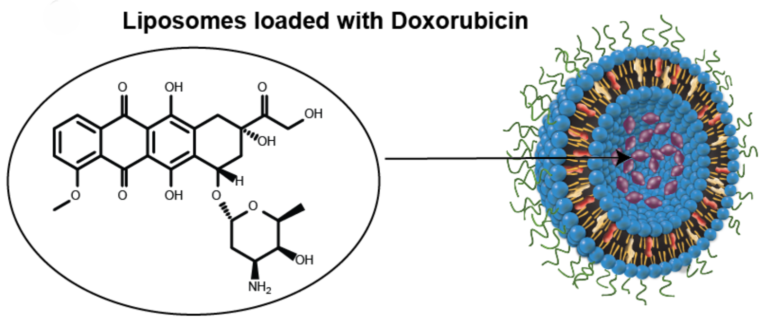

Figure 1. Schematic of Doxoves.

Analysis software

The analysis steps shown in this case study were all carried out using MATLAB. The MATLAB script is included as an appendix.

Materials and Methods

The liposome samples were sourced from FormuMax as shown in Table 1:

Liposome	Vendor	Name	Cat #
Doxoves®	FormuMax	Doxoves®: liposomal Doxorubicin	F30204B-D2
Control Liposomes	FormuMax	Plain control ammonium sulphate liposomes	F30204B-C2

Table 1. Liposome sample sourcing information.

UV-Vis spectra of liposomes were recorded on Thermo Fisher Scientific Nanodrop 2000c or Nanodrop One instruments using 10 mm path length equivalent mode. Reference blanks were recorded using PBS buffer at pH 7.4.

Analysis Methodology

The data was analyzed using MATLAB. Details about the code and its functions are covered in the Results section.

Results

UV-Vis Spectra of liposomes

Figure 2 below shows the UV-vis spectra recorded for drug-loaded Doxoves®. It can be seen that both drug-loaded, and control liposomes show substantial absorption at the low UV range (< 300 nm). However, it is the drug-loaded liposomes (Doxoves®), which have an absorption peak at 490 nm, while control/empty liposomes do not. Thus, both liposomes can be detected at 230 nm (our wavelength of choice for many such nanoscale systems, including AAV[17,18]), while drug-loaded liposomes can also be selectively detected at 490 nm.

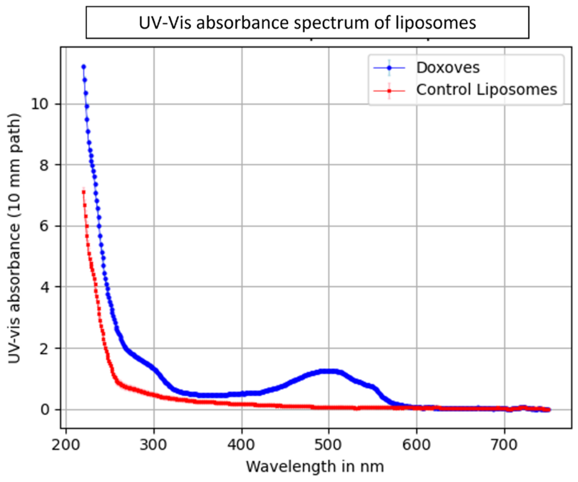

Figure 2. UV-vis Absorption spectra of drug-loaded liposomes (Doxoves®) in blue trace and empty/control liposomes in red trace.

Trial Single Wavelength (SWL) DGE-AUC of control liposomes in sucrose

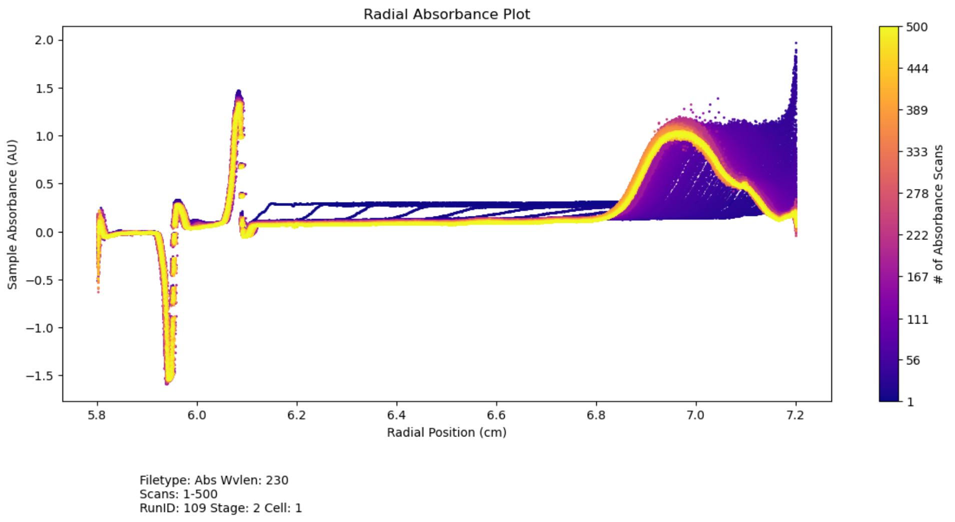

Figure 3. Trial DGE-AUC experiment on control/empty liposomes in sucrose.

One of the optimization steps in a DGE-AUC experiment is to determine the optimal starting concentration/density of the gradient-forming material (GFM). This process is discussed in the previous installment of this technical series: “Demystifying DGE-AUC Part 5: Optimizing data quality”. In Figure 3, we show the results of conducting a DGE-AUC experiment on a sample of control/empty liposomes in a PBS buffer containing 10% sucrose. The relevant parameters are shown below in Table 2.

Sample	Control/Empty Liposomes
Dilution	100x
Buffer	PBS, pH 7.4
Sucrose % (w/w)	6.5
Temperature (°C)	20
Rotor	ANTi50
Speed (krpm)	40
Wavelength (nm)	230
Scan Interval (sec)	600
# of Scans	500

Table 2. Experimental parameters for trial DGE-AUC experiment on control/empty liposomes in sucrose.

The following observations can be made from this plot:

1. The control/empty liposomes accumulate as a peak at the right end of the column, near 7.0 cm. This suggests that the density of these liposomes is greater than that of 6.5% w/w sucrose.

2. The main liposome peak appears to have a small right shoulder – suggesting heterogeneity in the formulation of these liposomes.

This experimental dataset is shown as an example of the optimization carried out to determine that the best starting concentration of GFM was ~ 10% w/v sucrose, as discussed in the next section.

Multiwavelength (MWL) DGE-AUC of drug-loaded liposomes (Doxoves®) and control liposomes in sucrose

In the following case study, we have performed DGE-AUC experiments on liposomes. The following samples were tested:

Sample 1: Control liposomes diluted 1:100 from stock in PBS with 10% sucrose.

Sample 2: Control liposomes diluted 1:100 from stock in PBS with 10% sucrose + Doxoves® (liposomes loaded with Doxorubicin), diluted 1:250 from stock in PBS with 10% sucrose.

Experiment conditions were as follows:

Sample	Control/Empty Liposomes	Control/Empty Liposomes + Drug-loaded Doxoves®
Rotor Cell Position	1	5
Dilution	100x	100x (Control) + 250x (Doxoves®)
Buffer	PBS, pH 7.4	PBS, pH 7.4
Sucrose % (w/w)	10	10
Temperature (°C)	20	20
Rotor	ANTi50	ANTi50
Speed (krpm)	40	40
Wavelength (nm)	230, 490	230, 490
Scan Interval (sec)	600	600
# of Scans	250	250

Table 3. Experimental parameters for DGE-AUC experiment on liposomes in sucrose.

The overall experiment duration was ~ 42 hrs. The last 5 scans (corresponding to the last 50 minutes) were used for DGE analysis. The raw data corresponding to all 250 scans for both cell positions 1 (empty liposomes) and 5 (mixture of empty liposomes and drug-loaded Doxoves®) is shown below in Figure 4.

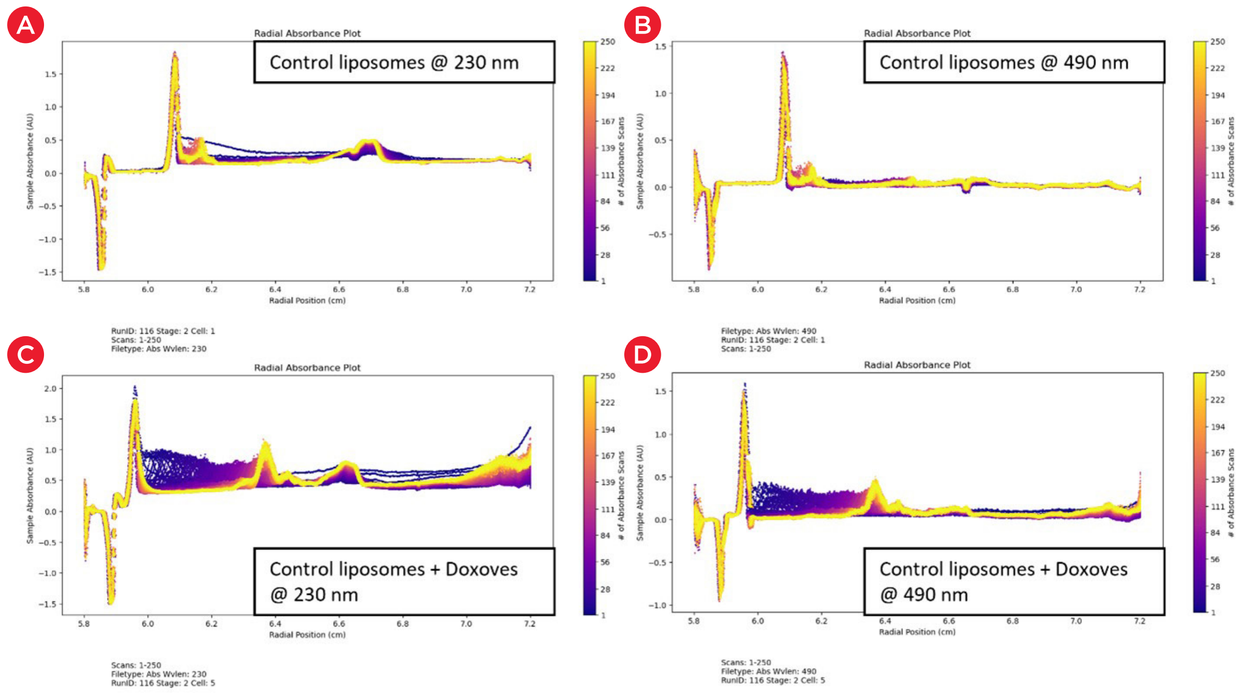

Figure 4. DGE-AUC on liposomes in sucrose (all 250 scans). **A**: Control liposomes at 230 nm, **B**: Control liposomes at 490 nm, **C**: Control liposomes + Doxoves® at 230 nm, **D**: Control liposomes + Doxoves® at 490 nm.

Figure 4 above shows all the absorbance scans collected for both samples and both wavelengths. The following observations can be made:

1. Sample 1 with the control liposomes show a broad and irregularly shaped peak from 6.5 to 6.8 cm with a leading tail at 230 nm (panel A). This broad peak shows signal accumulation over time. There is also a signal artifact at 6.2 cm, which we attribute to a window imperfection (owing to the fact that this radial position does not show signal accumulation over time).
2. A corresponding peak is not seen for control liposomes at 490 nm (panel B).
3. Sample 2 with the mixture of control liposomes and Doxoves® shows a peak between 6.3 to 6.5 cm and another peak between 6.5 to 6.7 cm at 230 nm (panel C). Both are identified as sample peaks due to the visible accumulation of signal over time.
4. Sample 2 shows only the first peak from 6.3 to 6.5 cm at 490 nm, but not the second peak.
5. The baseline is irregular, and the overall data quality is very noisy for both wavelengths and both samples

As mentioned, the last 5 scans were sampled from this experiment and plotted as shown below in Figure 5. Both samples appear to have attained equilibrium and we may proceed to peak analysis.

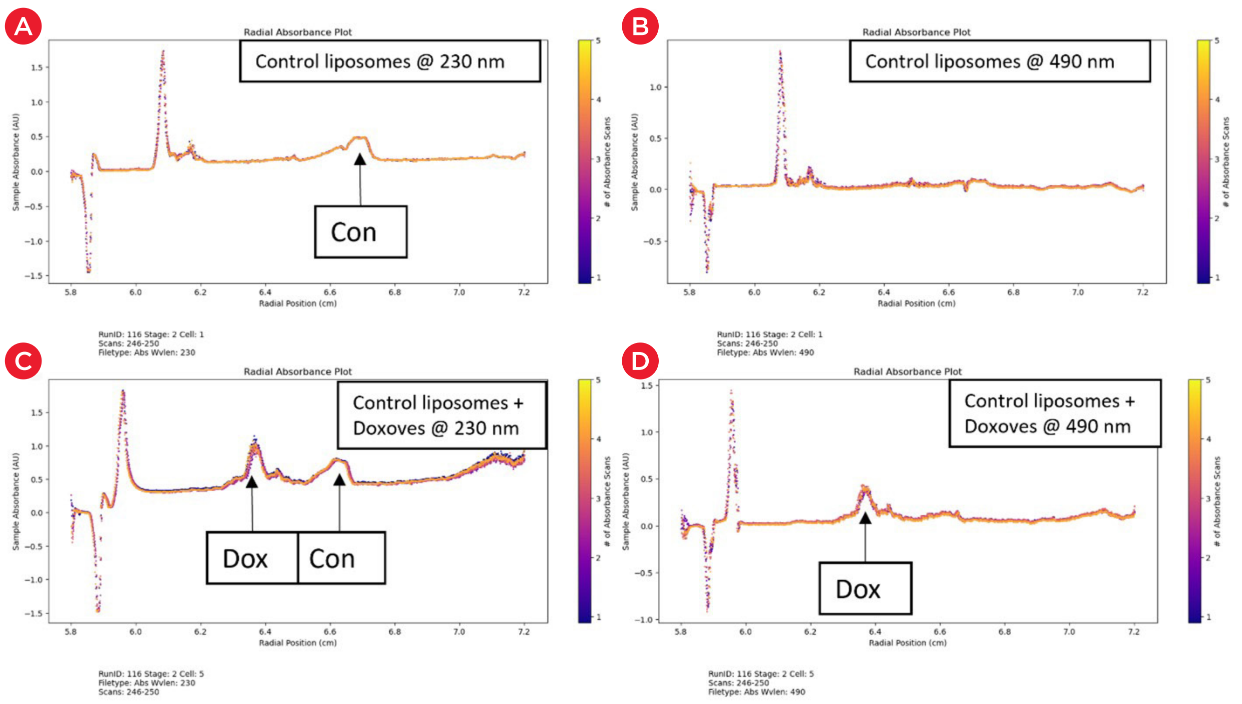

Figure 5. DGE-AUC on liposomes in sucrose (last 5 scans). **A**: Control liposomes at 230 nm, **B**: Control liposomes at 490 nm, **C**: Control liposomes + Doxoves® at 230 nm, **D**: Control liposomes + Doxoves® at 490 nm.

We know from the UV-vis spectra of control liposomes and Doxoves® **(shown in Fig 1)** that control liposomes absorb at 230 nm, but not at 490 nm, while Doxoves® absorb at both wavelengths. By comparison between panel A and panel C of Fig 3 and 4, we identify the first peak in panel C (between 6.5 to 6.7 cm) as corresponding to control liposomes. This is equivalent to the peak between 6.5 and 6.8 cm in panel A and is notably absent in panel B (at 490 nm). The control liposome peak is marked "Con" in panel A and C. Therefore, it follows that the second peak in panel C (between 6.3 and 6.5 cm) corresponds to Doxoves®. This peak is also seen in panel D (at 490 nm). The Doxoves® peak is marked "Dox" in panels C and D. This result is summarized in the table below:

	Left peak	Right peak
Control Liposomes @ 230 nm	N	Y
Control Liposomes @ 490 nm	N	N
Control Liposomes + Doxoves® @ 230 nm	Y	Y
Control Liposomes + Doxoves® @ 490 nm	Y	N

Table 4. Interpreting MWL-DGE-AUC to assign species identities to different peaks.

Therefore, it is sufficient to simply use the 230 nm scan data and integrate between the radii defined for these two peaks. This will determine the ratio of area-under-curve for the absorbance scan data at 230 nm corresponding to the two peaks. This ratio will lead to the loading fraction. This is demonstrated in the following section.

Data Analysis with MATLAB

Scans 246 to 250 of Absorbance data collected at 230 nm for the sample containing mixture of empty and drug-loaded liposomes were analyzed using the following steps:

1. Scans 246 to 250 were averaged. This was done using a Python script but can also be done using Excel.

2. The averaged scans were loaded in a spreadsheet in 2-column X-Y format (X=radius, Y=signal) as shown below (the radius column runs from 5.8 to 7.2 cm in increments of 0.001 cm):

X	Y
5.8	0.083159
5.801	-0.0245376
5.802	-0.11121722
5.803	-0.22363808
5.804	-0.16656466
5.805	-0.2013764
5.806	-0.1447744
5.807	-0.11111026
5.808	-0.06342976
5.809	-0.02309734

Table 5. Averaged absorbance scan data ready for integration.

3. This spreadsheet was loaded into MATLAB. The MATLAB script asks the user to define the minimum and maximum radial positions to truncate the data.

4. The input data is then smoothed to eliminate signal noise. Three options are provided to the user for smoothing:

   a. Moving Average

   b. Weighted Moving Average

   c. Polynomial

5. After smoothing, the script then performs baseline correction on the scan data and generates a diagnostic plot.

6. It then prompts the user to input the number of integration windows (in this case = 2) and define their limits (in this case: 6.3 to 6.5 cm and 6.5 to 6.7 cm).

7. The script finishes by calculating the absolute and percentage area under curve for each peak window. The results are saved in an output Excel file as shown below:

Window Number	Start Point	End Point	Area	% of Total Area	% of Sum Areas
Window 1	6.3	6.5	0.053364674	48.0345302	53.43711783
Window 2	6.5	6.7	0.046499758	41.85529199	46.56288217
Sum of Window Areas			0.099864432		

Table 6. Peak integration results for DGE-AUC on control liposomes + Doxoves® at 230 nm.

8. The plots are saved in png (scalar) and pdf (vector) format. The script can be modified to save in other formats as well (such as .svg, .jpg, etc).

Loading Fraction

Since we have already identified the left peak (6.3 to 6.5 cm) in Figure 5, panel C as corresponding to drug-loaded Doxoves® and the right peak (6.5 to 6.7 cm) as corresponding to empty liposomes, therefore in this sample, the loading fraction is the % of Sum Areas for window #1. This is ~ 53.44%.

The results of the MATLAB analysis are shown below in Figure 6. Panel A shows the raw data after truncation in purple trace, after smoothing with a moving average protocol (using a 10-point window) in cyan dots, the linear baseline in red dashes and the data after truncation, smoothing and baseline correction in green line trace. Panel B shows the two integration windows applied to the corrected data.

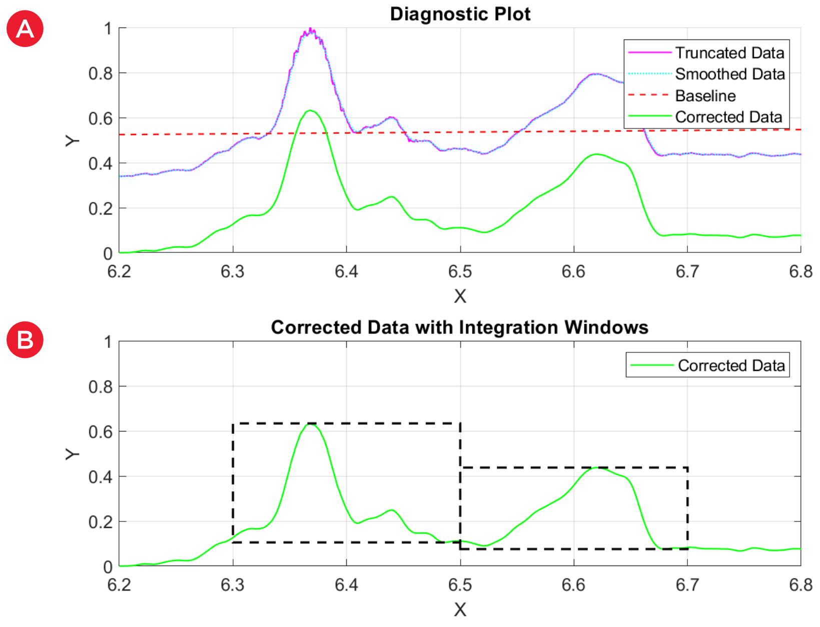

Figure 6. Peak integration for DGE-AUC at 230 nm of Doxoves + control liposomes in sucrose. **A**: Raw (truncated) data, smoothed data, baseline, data after baseline correction, **B**: peak analysis using baseline corrected data.

Discussion

The application of DGE-AUC to this system of liposomes allows the opportunity to use multiwavelength absorption data. Note that in this case study, we have used the juxtaposition of 230 nm and 490 nm absorption data to ascribe an identity to each of two peaks. The actual integration analysis was carried out exclusively on 230 nm data. Thus, this analysis should rightfully be thought of as **multiwavelength-informed peak selection**, as opposed to true multiwavelength analysis. The latter utilizes wavelength specific extinction coefficients of different participating species to extract relative populations. This is shown in[10], but in the context of SV-AUC experiments for AAV samples.

A point that has been made previously in this technical series is that DGE-AUC data can be visually interpreted and usually analyzed with commonly used graphing software like Origin, GraphPad Prism, SigmaPlot, Igor Pro® or even Excel. In this study, we have chosen to demonstrate the analysis of DGE-AUC data using MATLAB. The script used herein has been included as an appendix.

Conclusions

This application note should serve as a guide for any AUC user interested in exploring DGE-AUC with the help of scripting languages like MATLAB. The functions of data transformation – which include smoothing, baseline correction, etc. – can all be easily implemented using MATLAB and converted to other languages (such as GNU Octave) with minimal effort or Python/R with minor effort. The advantage with using scripts is, of course the automation of the analysis workflow. Once the script is configured for a specific sample (by defining data range and integration windows), it can be adapted to batch process multiple datasets for this sample type with little manual intervention.

Appendix: MATLAB script

```
%README
%{
---------------------------------------------------------
Author: Akash Bhattacharya
---------------------------------------------------------
What and Why?
This script will take 2-column population distribution data with some independent variable on the
X-axis (1st col) and some dependant variable (call it signal or population)
on the Y-axis (2nd col), and do the following as per user definitions:

1.  Perform data truncation
2.  Smooth truncated data using a moving window
3.  Fit a linear baseline
4.  Perform integration within user defined windows
5.  Save output data at each stage to Excel file
6.  Plot output
----------------------------------------------------------
Where to use?
This script can be used with any population distribution data
which has the structure: (parameter, population).
In Biophysics, this may be seen in chromatograms, for instance.
This particular implementation is written for Density Gradient Equilibrium
Analytical Ultracentrifugation (DGE-AUC) plots.
----------------------------------------------------------
Citation for DGE-AUC:
Sternisha SM, Wilson AD, Bouda E, Bhattacharya A, VerHeul R.
Optimizing high-throughput viral vector characterization with
density gradient equilibrium analytical ultracentrifugation.
Eur Biophys J. 2023 Jul;52(4-5):387-392. doi: 10.1007/s00249-023-01654-z.
Epub 2023 May 2. PMID: 37130969; PMCID: PMC10444642.
----------------------------------------------------------
What does input data look like?
Input data is provided as .xlsx files.
Format is 2 column
Example:
X-avg   Y-avg
5.811147619     0.057439952
5.81217619      0.057959729
5.813133333     0.047285886
5.814114286     0.053657486
5.815142857     0.048678086
....
-------------------------------------------------------------
Step by step :

1.  Take an AUC scan (or an average of scans), plot the full range.
2.  Truncate the dataset according to user defined xmin and xmax values.
3.  Smooth the truncated data.
4.  Correct for linear baseline
5.  Plot the truncated dataset, data after smoothing, baseline and data after baseline correction.
6.  Pick the number of integration windows. Define their limits.
7.  Calculate area-under-curve using trapezoidal integrals for all windows.
8.  Plot output, plot integration windows.
9.  Write report to Excel.

Note: Make sure to adjust the legend locations, labels, and other parameters as needed for your
specific data.
%}
%--------------------------------------------------------------------------
% Part 0: Initialize all

% Initialization for Clean and Controlled Execution
clear; clc; close all; %Clear variables, console and close figures

% Reset default settings for figures to ensure consistent layout/size
set(0, 'DefaultFigureWindowStyle', 'normal');
set(0, 'DefaultLineLineWidth', 1);
set(0, 'DefaultAxesFontSize', 12);

% End Part 0: Initialize all
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Part 1: User Inputs

% Set a default directory as a fallback option
defaultDirectory = <Specify yourself>;

% Allow the user to select a file with a GUI, starting at the default directory
[file_name, file_path] = uigetfile(fullfile(defaultDirectory, '*.xlsx'), 'Select Data File');
if isequal(file_name, 0) || isequal(file_path, 0)
    error('No file was selected. Exiting the script.');
else
    fullPath = fullfile(file_path, file_name);
    disp(['File selected: ', fullPath]);
end

% Use the file path as the data directory
DataDirectory = file_path;
disp(['Using data directory: ', DataDirectory]);

% Check if the selected file exists and is accessible
if ~exist(fullPath, 'file')
    error('Selected file does not exist or cannot be accessed.');
end
```

```
% Check if the file exists
if ~exist(fullPath, 'file')  % Using fullPath for clarity
    error('File not found: %s', fullPath);
end
% Allow the user to specify the sheet name
defaultSheetName = 'Averaged Scan Data';  % Default sheet name
prompt = sprintf('Enter sheet name (default: %s): ', defaultSheetName);
sheet_name = input(prompt, 's');
if isempty(sheet_name)
    sheet_name = defaultSheetName;
end

% Try to load data from the specified sheet in the Excel file
try
    data = readmatrix(fullPath, 'Sheet', sheet_name);
catch
    error('Failed to read the specified sheet: %s from the file: %s. Ensure the sheet name is correct and
the file is not corrupted.', sheet_name, fullPath);
end

% Check if data is loaded properly
if isempty(data)
    error('No data loaded from the sheet: %s in the file: %s. Check if the sheet is empty.', sheet_name,
fullPath);
end

% Construct Output Filename Based on Input Filename and Current Date-Time

% Extract the base name of the input file without extension
[~, inputBaseName, ~] = fileparts(fullPath);

% Get current date and time formatted as YYYYMMDD-HHMM
dateTimeNow = datestr(now, 'yyyymmdd-HHMM');

% Construct the output filename
outputFileName = sprintf('%s-output-%s.xlsx', inputBaseName, dateTimeNow);

% Define the full output file path in the same directory as the input
outputFilePath = fullfile(DataDirectory, outputFileName);

% Display the intended output file path (optional, for verification)
disp(['Output will be saved as: ', outputFilePath]);

% User defined output Verbosity flag (set to 1 to enable printing, 0 to disable)
defaultVerbosity = 1; % Default verbosity is enabled
verbosity = input(sprintf('Set verbosity (1=enabled, 0=disabled, default: %d): ', defaultVerbosity), 's');
verbosity = str2double(verbosity);
if isnan(verbosity) || isempty(verbosity)
    verbosity = defaultVerbosity;
end
```

```
% End Part 1: User Inputs
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Part 2: Data Preparation

% Default column indices for x and y data
defaultXCol = 1;
defaultYCol = 2;
x_col = input(sprintf('Enter column number for X data (default: %d): ', defaultXCol), 's');
y_col = input(sprintf('Enter column number for Y data (default: %d): ', defaultYCol), 's');
x_col = str2double(x_col);
y_col = str2double(y_col);
if isnan(x_col) || isempty(x_col)
    x_col = defaultXCol;
end
if isnan(y_col) || isempty(y_col)
    y_col = defaultYCol;
end

% User-defined xmin and xmax values
defaultXmin = 6.2;
defaultXmax = 7.0;
xmin = input(sprintf('Enter minimum x value (default: %g): ', defaultXmin), 's');
xmax = input(sprintf('Enter maximum x value (default: %g): ', defaultXmax), 's');
xmin = str2double(xmin);
xmax = str2double(xmax);
if isnan(xmin) || isempty(xmin)
    xmin = defaultXmin;
end
if isnan(xmax) || isempty(xmax)
    xmax = defaultXmax;
end
if xmin >= xmax
    error('xmin must be less than xmax.');
end

% Ensure the specified columns exist in the data
if size(data, 2) < max(x_col, y_col)
    error('The data does not contain the specified columns %d and %d. Ensure that the sheet has at least
%d columns.', x_col, y_col, max(x_col, y_col));
end

% Extract x and y columns
x = data(:, x_col);
y = data(:, y_col);

% Check if x and y data are not empty
if isempty(x) || isempty(y)
    error('The specified column(s) for X (%d) or Y (%d) are empty in the sheet: %s. Ensure the columns
contain data.', x_col, y_col, sheet_name);
end
```

```
% End Part 2: Data Preparation
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% Part 3: Data Processing - Alternative Smoothing Methods

% User choice for smoothing method
disp('Select a smoothing method:');
disp('1: Moving Average');
disp('2: Weighted Moving Average');
disp('3: Polynomial Smoothing (order = 2)');
userInput = input('Enter your choice (1-3, default: 1): ', 's');  % Prompt user, showing the default

% Check if input is empty and set default to 1
if isempty(userInput)
    smoothingChoice = 1;  % Default to 'Moving Average'
else
    smoothingChoice = str2double(userInput);  % Convert input to numeric
    if isnan(smoothingChoice) || ~ismember(smoothingChoice, [1, 2, 3])
        error('Invalid input. Please enter 1, 2, or 3.');  % Error if input is not 1, 2, or 3
    end
end

% Truncate data based on user-defined xmin and xmax values
truncated_indices = (x >= xmin) & (x <= xmax);
truncated_x = x(truncated_indices);
truncated_y = y(truncated_indices);

if isempty(truncated_x)
    error('No data within the specified range [%g, %g]. Please adjust your xmin and xmax values.', xmin,
xmax);
end

% Initialize default parameters
defaultWindowSize = 3;  % Default window size for smoothing

% User input for window size with validation and default handling
prompt = sprintf('Enter window size for smoothing (default: %d): ', defaultWindowSize);
userInput = input(prompt, 's');
windowSize = str2double(userInput);

% Validate window size input
if isnan(windowSize) || windowSize <= 0
    % Set windowSize to either a fixed default or a dynamic portion of your data length
    windowSize = min(defaultWindowSize, length(truncated_y));  % Ensure it's not more than available
points
elseif windowSize > length(truncated_y)
    % If window size is greater than data points available, reset to maximum available
    windowSize = length(truncated_y);
    fprintf('Window size set to maximum available points: %d\n', windowSize);
end

% Display the used window size if verbosity is enabled
if verbosity
    disp(['Using window size: ', num2str(windowSize)]);
end
```

```
% Padding data for smoothing to minimize edge effects
pad_size = windowSize;  % Padding size equal to the window size
y_padded = [repmat(truncated_y(1), pad_size, 1); truncated_y; repmat(truncated_y(end), pad_size, 1)];

% Apply selected smoothing method
switch smoothingChoice
    case 1
        % Moving Average Smoothing
        smoothed_y_padded = movmean(y_padded, windowSize);
        % Remove padding
        smoothed_y = smoothed_y_padded(pad_size+1:end-pad_size);
        method = 'Moving Average';

    case 2
        % Weighted Moving Average
        weights = linspace(1, 2, windowSize);  % Linearly increasing weights
        weights = weights / sum(weights);  % Normalize weights
        smoothed_y_padded = conv(y_padded, weights, 'same');
        % Remove padding
        smoothed_y = smoothed_y_padded(pad_size+1:end-pad_size);
        method = 'Weighted Moving Average';

    case 3
        % Polynomial Smoothing (Custom Implementation)
        smoothed_y = polySmooth(truncated_x, truncated_y, windowSize, 2);  % Polynomial order is 2
        method = 'Polynomial Smoothing';
        smoothed_y_padded = []; % This method is not padded in this example
    otherwise
        error('Invalid choice. Please select 1, 2, or 3.');
end

% Fit a linear baseline to the smoothed data
p = polyfit(truncated_x, smoothed_y, 1);  % Fit a linear polynomial
baseline = polyval(p, truncated_x);       % Evaluate the polynomial at the x values

% Subtract the baseline from the smoothed data
corrected_y = smoothed_y - baseline;

% Adjust the entire dataset if the minimum value is less than zero
if min(corrected_y) < 0
    corrected_y = corrected_y - min(corrected_y);
end
```

```
% Optionally, plot the original, padded smoothed, smoothed, and baseline-corrected data for
comparison
if verbosity
    figure;
    hold on;
    plot(truncated_x, truncated_y, 'm-', 'LineWidth', 1,'DisplayName', 'Truncated Data');
    plot(truncated_x, smoothed_y, 'c:', 'LineWidth', 1,'DisplayName', 'Smoothed Data');
    plot(truncated_x, baseline, 'r--', 'LineWidth', 1,'DisplayName', 'Fitted Baseline');
    plot(truncated_x, corrected_y, 'g-', 'LineWidth', 1, 'DisplayName', 'Baseline Corrected Data');
    title('Data Smoothing and Baseline Correction');
    xlabel('X');
    ylabel('Y');
    legend show;
    grid on;
    hold off;
end

% End Part 3: Data Processing
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Part 4: Integration and Area Under the Curve Calculation

% Ask the user how many integration windows they want to define
nWindows = input('Enter the number of integration windows: ');
integrationWindows = zeros(nWindows, 2);  % Initialize matrix to store windows

% Collect window definitions from the user
for i = 1:nWindows
    integrationWindows(i, :) = input(sprintf('Enter start and end points for window %d (e.g., [start end]): ', i));
end

% Initialize array to store areas for each window plus one extra for the full range
areas = zeros(1, nWindows + 1);

% Calculate the area under the curve for the entire range of truncated_x
areas(1) = trapz(truncated_x, corrected_y);  % Save it in the first slot of the areas array

% Initialize sum of areas for user-defined windows
sumAreas = 0;

% Calculate the area under the curve for each defined window
for i = 1:nWindows
    % Find indices of x within the integration window
    windowIndices = truncated_x >= integrationWindows(i, 1) & truncated_x <= integrationWindows(i, 2);
    % Extract the subset of x and y data within the window
    windowX = truncated_x(windowIndices);
    windowY = corrected_y(windowIndices);
    % Calculate the area using the trapezoidal rule
    areas(i + 1) = trapz(windowX, windowY);  % Save in subsequent slots of the areas array
    % Accumulate the area sums
    sumAreas = sumAreas + areas(i + 1);

end
```

```
% Calculate percentages
percentOfTotal = areas(2:end) / areas(1) * 100;  % Each window's area as a % of the total area
percentOfSumAreas = areas(2:end) / sumAreas * 100;  % Each window's area as a % of the sum of all
windows

% Display the results
disp('Area under the entire curve:');
fprintf('Total area: %g\n', areas(1));
disp('Areas and percentages under the curve for each defined window:');
for i = 1:nWindows
    fprintf('Window %d (from %g to %g): Area = %g, %% of Total = %g%%, %% of Sum Areas = %g%%\n', ...
            i, integrationWindows(i, 1), integrationWindows(i, 2), areas(i + 1), percentOfTotal(i),
percentOfSumAreas(i));
end
disp('Sum of areas for all defined windows:');
fprintf('Sum of areas: %g\n', sumAreas);

% End Part 4: Integration and Area Under the Curve Calculation
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Part 5: Export Results to Excel with Headers

% Define output file path using the dynamic filename constructed earlier
outputFilePath = fullfile(DataDirectory, outputFileName);

% Prepare the data for the "Raw Data" sheet with headers
raw_headers = {'X', 'Y'};

export_raw_data = [raw_headers; num2cell(NaN(max(length(x), length(y)), 2))];  % Initialize with NaNs
to accommodate different lengths
export_raw_data(2:end, 1) = num2cell(x);  % Raw data x
export_raw_data(2:end, 2) = num2cell(y);  % Raw data y

% Write Raw Data to Excel file using writecell
writecell(export_raw_data, outputFilePath, 'Sheet', 'Raw Data', 'Range', 'A1');

% Prepare the data for the "Processed Data" sheet with headers
processed_headers = {'Trunc(X)', 'Trunc(Y)', 'Smooth(Y)', 'Baseline', 'Corr(Y)'};
max_length = length(truncated_x);  % All processed arrays should be aligned in length
export_processed_data = [processed_headers; num2cell(NaN(max_length, 5))];  % Initialize with NaNs
export_processed_data(2:end, 1) = num2cell(truncated_x);  % Truncated data x
export_processed_data(2:end, 2) = num2cell(truncated_y);  % Truncated data y
export_processed_data(2:end, 3) = num2cell(smoothed_y);   % Smoothed y
export_processed_data(2:end, 4) = num2cell(baseline);     % Baseline
export_processed_data(2:end, 5) = num2cell(corrected_y);  % Corrected y

% Write Processed Data to Excel file using writecell
writecell(export_processed_data, outputFilePath, 'Sheet', 'Processed Data', 'Range', 'A1');

% Display completion message
disp(['Data successfully written to ', outputFilePath]);
```

```
% End Part 5: Export Results to Excel with Headers
%---------------------------------------------------------------------------

%---------------------------------------------------------------------------
% Part 6: Export Integration Results to Excel

% Initialize the cell array for storing integration results
results_header = {'Window Number', 'Start Point', 'End Point', 'Area', '% of Total Area', '% of Sum Areas'};
integration_results = cell(nWindows + 2, length(results_header));  % +2 for total and sum areas rows

% Fill in the header
integration_results(1, :) = results_header;

% Fill in the data for each window
for i = 1:nWindows
    integration_results{i + 1, 1} = sprintf('Window %d', i);
    integration_results{i + 1, 2} = integrationWindows(i, 1);
    integration_results{i + 1, 3} = integrationWindows(i, 2);
    integration_results{i + 1, 4} = areas(i + 1);
    integration_results{i + 1, 5} = percentOfTotal(i);
    integration_results{i + 1, 6} = percentOfSumAreas(i);
end

% Add total area and sum of areas
integration_results{nWindows + 2, 1} = 'Total Area';
integration_results{nWindows + 2, 4} = areas(1);
integration_results{nWindows + 2, 1} = 'Sum of Window Areas';
integration_results{nWindows + 2, 4} = sumAreas;

% Write integration results to Excel file
writecell(integration_results, outputFilePath, 'Sheet', 'Integration Results', 'Range', 'A1');

% Display completion message
disp(['Integration results successfully written to ', outputFilePath]);

% End Part 6: Export Integration Results to Excel
%---------------------------------------------------------------------------

%---------------------------------------------------------------------------
% Part 7: Plot Results and Save Figure

% Create figure
fig = figure;
set(fig, 'Position', [100, 100, 800, 600]);  % Set figure size

%---------------------------------------------------------------------------
% Ask for Y-axis Limits

% Initialize default y-axis limits based on data range
defaultYMin = min(corrected_y);
defaultYMax = max(corrected_y);

% Prompt user for custom y-axis limits
yMinInput = input(sprintf('Enter minimum Y-axis limit (default: %g): ', defaultYMin), 's');
yMaxInput = input(sprintf('Enter maximum Y-axis limit (default: %g): ', defaultYMax), 's');
```

```
% Convert inputs to numbers and apply defaults if necessary
yMin = str2double(yMinInput);
yMax = str2double(yMaxInput);
if isnan(yMin) || isempty(yMin)
    yMin = defaultYMin;  % Use default if input is invalid or empty
end
if isnan(yMax) || isempty(yMax)
    yMax = defaultYMax;  % Use default if input is invalid or empty
end

% Ensure valid range
if yMin >= yMax
    error('Minimum Y-axis limit must be less than maximum Y-axis limit.');
end

% Display set limits
disp(['Y-axis limits set to: Min = ', num2str(yMin), ', Max = ', num2str(yMax)]);

%----------------------------------------------------------------------------

% Subplot 1: Diagnostic plots
subplot(2, 1, 1);
hold on;
plot(truncated_x, truncated_y, 'm-', 'DisplayName', 'Truncated Data');
plot(truncated_x, smoothed_y, 'c:', 'DisplayName', 'Smoothed Data');
plot(truncated_x, baseline, 'r--', 'DisplayName', 'Baseline');
plot(truncated_x, corrected_y, 'g-', 'DisplayName', 'Corrected Data');
title('Diagnostic Plot');
xlabel('X');
ylabel('Y');
ylim([yMin, yMax]);  % Apply custom y-axis limits
legend show;
grid on;
hold off;

% Subplot 2: Corrected data with integration windows
subplot(2, 1, 2);
plot(truncated_x, corrected_y, 'g-', 'DisplayName', 'Corrected Data'); % Plot corrected data
title('Corrected Data with Integration Windows');
xlabel('X');
ylabel('Y');
ylim([yMin, yMax]);  % Apply custom y-axis limits
hold on;
% Array for legend entries
legendInfo = {'Corrected Data'}; % Initialize with corrected data
```

```
% Add rectangles for integration windows and prepare legend entries
for i = 1:nWindows
    idx = truncated_x >= integrationWindows(i, 1) & truncated_x <= integrationWindows(i, 2);
    x_rect = truncated_x(idx);
    if ~isempty(x_rect)
        rectangle('Position', [x_rect(1), min(corrected_y(idx)), x_rect(end) - x_rect(1),
max(corrected_y(idx)) - min(corrected_y(idx))], ...
                  'EdgeColor', 'k', 'LineWidth', 1.5, 'LineStyle', '--');
        % Add a descriptive entry for the legend
        legendInfo{end+1} = sprintf('Window %d', i);
    end
end
legend(legendInfo); % Display legend with descriptive labels
grid on;
hold off;

% Generate output filenames for the figure
dateTimeNow = datestr(now, 'yyyymmdd-HHMM');
figBaseName = sprintf('%s-fig-%s', inputBaseName, dateTimeNow);
figPDFPath = fullfile(DataDirectory, [figBaseName, '.pdf']);
figPNGPath = fullfile(DataDirectory, [figBaseName, '.png']);

% Save figure
saveas(fig, figPDFPath, 'pdf'); % Save as PDF
saveas(fig, figPNGPath, 'png'); % Save as PNG

% Display completion message
disp(['Figure saved as PDF: ', figPDFPath]);
disp(['Figure saved as PNG: ', figPNGPath]);

% End Part 7: Plot Results and Save Figure
%---------------------------------------------------------------------------

%---------------------------------------------------------------------------

% Custom function for Polynomial Smoothing
function smoothed = polySmooth(x, y, windowSize, polyOrder)
    smoothed = zeros(size(y));
    halfWin = floor(windowSize / 2);
    for i = 1:length(y)
        lb = max(1, i - halfWin);
        ub = min(length(y), i + halfWin);
        p = polyfit(x(lb:ub), y(lb:ub), polyOrder);
        smoothed(i) = polyval(p, x(i));
    end
end
```

References

1. Inc., F. S. (FormuMax Scientific Inc.).

2. Cole, J. L., Lary, J. W., P Moody, T. & Laue, T. M. Analytical ultracentrifugation: sedimentation velocity and sedimentation equilibrium. Methods Cell Biol 84, 143-179 (2008). **https://doi.org/10.1016/S0091-679X(07)84006-4**

3. Fulton, M. D. & Najahi-Missaoui, W. Liposomes in Cancer Therapy: How Did We Start and Where Are We Now. Int J Mol Sci 24, 6615 (2023). **https://doi.org/10.3390/ijms24076615**

4. Guimarães, D., Cavaco-Paulo, A. & Nogueira, E. Design of liposomes as drug delivery system for therapeutic applications. Int J Pharm 601, 120571 (2021). **https://doi.org/10.1016/j.ijpharm.2021.120571**

5. Liu, P., Chen, G. & Zhang, J. A Review of Liposomes as a Drug Delivery System: Current Status of Approved Products, Regulatory Environments, and Future Perspectives. Molecules 27, 1372 (2022). **https://doi.org/10.3390/molecules27041372**

6. Sheoran, S., Arora, S., Samsonraj, R., Govindaiah, P. & Vuree, S. Lipid-based nanoparticles for treatment of cancer. Heliyon 8, e09403 (2022). **https://doi.org/10.1016/j.heliyon.2022.e09403**

7. Muggia, F. M. Liposomal encapsulated anthracyclines: new therapeutic horizons. Curr Oncol Rep 3, 156-162 (2001). **https://doi.org/10.1007/s11912-001-0016-5**

8. Porche, D. J. Liposomal doxorubicin (Doxil). J Assoc Nurses AIDS Care 7, 55-59 (1996). **https://doi.org/10.1016/S1055-3290(96)80016-1**

9. Rivankar, S. An overview of doxorubicin formulations in cancer therapy. J Cancer Res Ther 10, 853-858 (2014). https://doi.org/10.4103/0973-1482.139267

10. Richter, K. et al. Purity and DNA content of AAV capsids assessed by analytical ultracentrifugation and orthogonal biophysical techniques. Eur J Pharm Biopharm 189, 68-83 (2023). **https://doi.org/10.1016/j.ejpb.2023.05.011**

11. Sainaga Jyothi, V. G. S. et al. Stability characterization for pharmaceutical liposome product development with focus on regulatory considerations: An update. Int J Pharm 624, 122022 (2022). **https://doi.org/10.1016/j.ijpharm.2022.122022**

12. Bennett, A. et al. Thermal Stability as a Determinant of AAV Serotype Identity. Mol Ther Methods Clin Dev 6, 171-182 (2017). **https://doi.org/10.1016/j.omtm.2017.07.003**

13. Su, Q., Sena-Esteves, M. & Gao, G. Purification of Recombinant Adeno-Associated Viruses (rAAVs) by Cesium Chloride Gradient Sedimentation. Cold Spring Harb Protoc 2020, 095604 (2020). **https://doi.org/10.1101/pdb.prot095604**

14. Has, C. & Sunthar, P. A comprehensive review on recent preparation techniques of liposomes. J Liposome Res 30, 336-365 (2020). **https://doi.org/10.1080/08982104.2019.1668010**

15. Lombardo, D. & Kiselev, M. A. Methods of Liposomes Preparation: Formation and Control Factors of Versatile Nanocarriers for Biomedical and Nanomedicine Application. Pharmaceutics 14 (2022). **https://doi.org/10.3390/pharmaceutics14030543**

16. Patil, Y. P. & Jadhav, S. Novel methods for liposome preparation. Chem Phys Lipids 177, 8-18 (2014). **https://doi.org/10.1016/j.chemphyslip.2013.10.011**

17. Kattawar, G. W. & Plass, G. N. Electromagnetic scattering from absorbing spheres. Appl Opt 6, 1377-1382 (1967). https://doi.org/10.1364/AO.6.001377

18. Plass, G. N. Mie scattering and absorption cross sections for absorbing particles. Appl Opt 5, 279-285 (1966). https://doi.org/10.1364/AO.5.000279

Legal Notices

Several third-party software vendors supply programs for the analyses of the raw data generated by the Beckman Coulter Inc. ("Beckman") Analytical Ultracentrifuge. Third-party analysis software has not been validated by Beckman for use with the Beckman Analytical Ultracentrifuge. Training conducted by Beckman on the third-party software, does not imply a recommendation for use or the suitability of the third-party software use by the customer. Beckman does not endorse any third-party analyses software. Beckman warranty and/or performance guarantee that may be applicable or are provided by Beckman for Beckman Analytical Ultracentrifuge do not apply to any third-party software.

Copyright, License and Terms of Use Disclaimers are documented for the below software on their respective pages. For example:

a. SEDFIT: **https://sedfitsedphat.nibib.nih.gov/software/**

b. OriginLab: **https://www.originlab.com/index.aspx?go=Company/TermsOfUse**

c. UltraScan: **http://ultrascan.aucsolutions.com/license.php**

d. Microsoft: **https://www.microsoft.com/en-us/useterms/**

e. Gussi: **https://www.utsouthwestern.edu/labs/mbr/software/**

f. Python: **https://www.python.org/**

g. Matplotlib: **https://matplotlib.org/**

h. NumPy: **https://numpy.org/**

i. SciPy: **https://scipy.org/**

j. MATLAB: **https://www.mathworks.com/products/matlab.html**

Beckman Coulter makes no warranties of any kind whatsoever express or implied, with respect to this protocol, including but not limited to warranties of fitness for a particular purpose or merchantability or that the protocol is non-infringing. All warranties are expressly disclaimed. Your use of the method is solely at your own risk, without recourse to Beckman Coulter. Not intended or validated for use in the diagnosis of disease or other conditions. This protocol is for demonstration only and is not validated by Beckman Coulter.

All products and services identified, unless noted as for in-vitro diagnostic (IVD) use, are for research use and not intended or validated for use in the diagnosis of disease or other conditions.

©2025 Beckman Coulter, Inc. All rights reserved. Beckman Coulter, the Stylized Logo, and Beckman Coulter product and service marks mentioned herein are trademarks or registered trademarks of Beckman Coulter, Inc. in the United States and other countries. All other trademarks are the property of their respective owners.

BECKMAN COULTER Life Sciences

For Beckman Coulter's worldwide office locations and phone numbers, please visit Contact Us at beckman.com

2024-GBL-EN-106996-v1

danaher