```
% Purpose: This script simulates the response of multiple nanobiosensors to a time-dependent biomarker concentration profile. The nanobiosensor characteristics are

% Parameters
numPoints = 100;
time = linspace(0, 10, numPoints);

% Time-dependent Biomarker Concentration Profile
biomarkerConcentrationProfile = 30 * sin(time) + 50;  % Adjusted for a more dynamic profile

% Nanobiosensor Characteristics
nanobiosensors = struct('type', {'affinity', 'catalytic'}, 'nanomaterial', {'gold', 'silver'});

% Adjusted Binding Affinities, Baselines, and Amplitudes for Notable Difference
nanobiosensors(1).bindingAffinity = 0.1;  % Adjusted for gold nanobiosensor
nanobiosensors(1).baseline = 1.5;
nanobiosensors(1).amplitude = 6;

nanobiosensors(2).bindingAffinity = 0.05;  % Adjusted for silver nanobiosensor
nanobiosensors(2).baseline = 2.5;
nanobiosensors(2).amplitude = 8;

% Preallocate arrays for performance metrics
sensitivities = zeros(length(nanobiosensors), numPoints);
specificities = zeros(length(nanobiosensors), numPoints);

% Simulate Nanobiosensors and Analyze Performance Metrics
try
    figure;
    for i = 1:length(nanobiosensors)
        [bindingCurve, signalChange, noisySignal] = simulateNanobiosensor(time, biomarkerConcentrationProfile, nanobiosensors(i));

        % Vary the threshold to evaluate sensitivity and specificity
        thresholds = linspace(min(signalChange), max(signalChange), 100);

        for t = 1:length(thresholds)
            threshold = thresholds(t);
            truePositive = sum(signalChange > threshold & biomarkerConcentrationProfile > threshold);
            trueNegative = sum(signalChange <= threshold & biomarkerConcentrationProfile <= threshold);
            falsePositive = sum(signalChange > threshold & biomarkerConcentrationProfile <= threshold);
            falseNegative = sum(signalChange <= threshold & biomarkerConcentrationProfile > threshold);

            % Calculate sensitivity and specificity at each threshold
            sensitivities(i, t) = truePositive / (truePositive + falseNegative);
            specificities(i, t) = trueNegative / (trueNegative + falsePositive);
        end

        % Plot results and nanobiosensor response
        subplot(3, 1, 1);
        plot(time, biomarkerConcentrationProfile, 'LineWidth', 2);
        xlabel('Time (arbitrary units)');
        ylabel('Biomarker Concentration');
        title('Time-dependent Biomarker Concentration Profile');

        subplot(3, 1, 2);
        plot(time, bindingCurve, 'LineWidth', 2, 'DisplayName', nanobiosensors(i).nanomaterial);
        hold on;
        xlabel('Time (arbitrary units)');
        ylabel('Binding Curve');
        title('Biomarker Binding Curve');
        legend('Location', 'Best');

        subplot(3, 1, 3);
        plot(time, signalChange, 'LineWidth', 2, 'DisplayName', nanobiosensors(i).nanomaterial);
        hold on;
        xlabel('Time (arbitrary units)');
        ylabel('Signal Intensity');
        title('Nanobiosensor Response to Biomarker');
        legend('Location', 'Best');
    end

    % Plot performance metrics
    figure;
    subplot(2, 1, 1);
    for i = 1:length(nanobiosensors)
        plot(time, sensitivities(i, :), 'LineWidth', 2, 'DisplayName', nanobiosensors(i).nanomaterial);
        hold on;
    end
    xlabel('Time (arbitrary units)');
    ylabel('Sensitivity');
    title('Sensitivity Over Time');
    legend('Location', 'Best');
    grid on;

    subplot(2, 1, 2);
    for i = 1:length(nanobiosensors)
        plot(time, specificities(i, :), 'LineWidth', 2, 'DisplayName', nanobiosensors(i).nanomaterial);
        hold on;
    end
    xlabel('Time (arbitrary units)');
```

```
    ylabel('Specificity');
    title('Specificity Over Time');
    legend('Location', 'Best');
    grid on;

catch ex
    fprintf('Error in simulating nanobiosensors: %s\n', ex.message);
    disp(ex.getReport());
end

% Supporting Functions
% ----------------------

function [bindingCurve, signalChange, noisySignal] = simulateNanobiosensor(time, concentrationProfile, nanobiosensor)
    try
        bindingCurve = sigmoid(concentrationProfile, nanobiosensor.bindingAffinity);

        % Adjust baseline and amplitude for each nanobiosensor type
        baseline = nanobiosensor.baseline;
        amplitude = nanobiosensor.amplitude;

        signalChange = calculateSignalChange(concentrationProfile, bindingCurve, baseline, amplitude);

        noiseLevel = 0.2;
        noisySignal = addNoise(signalChange, noiseLevel);
    catch ex
        fprintf('Error in simulating nanobiosensor components: %s\n', ex.message);
        bindingCurve = [];
        signalChange = [];
        noisySignal = [];
    end
end

function curve = sigmoid(x, k)
    curve = 1 ./ (1 + exp(-k * (x - mean(x))));
end

function signalChange = calculateSignalChange(concentration, bindingCurve, baseline, amplitude)
    signalChange = baseline + amplitude * bindingCurve;
end

function noisySignal = addNoise(signal, noiseLevel)
    noisySignal = signal + noiseLevel * randn(size(signal));
end

% End of code
```

Published with MATLAB® R2021a