Supporting Information

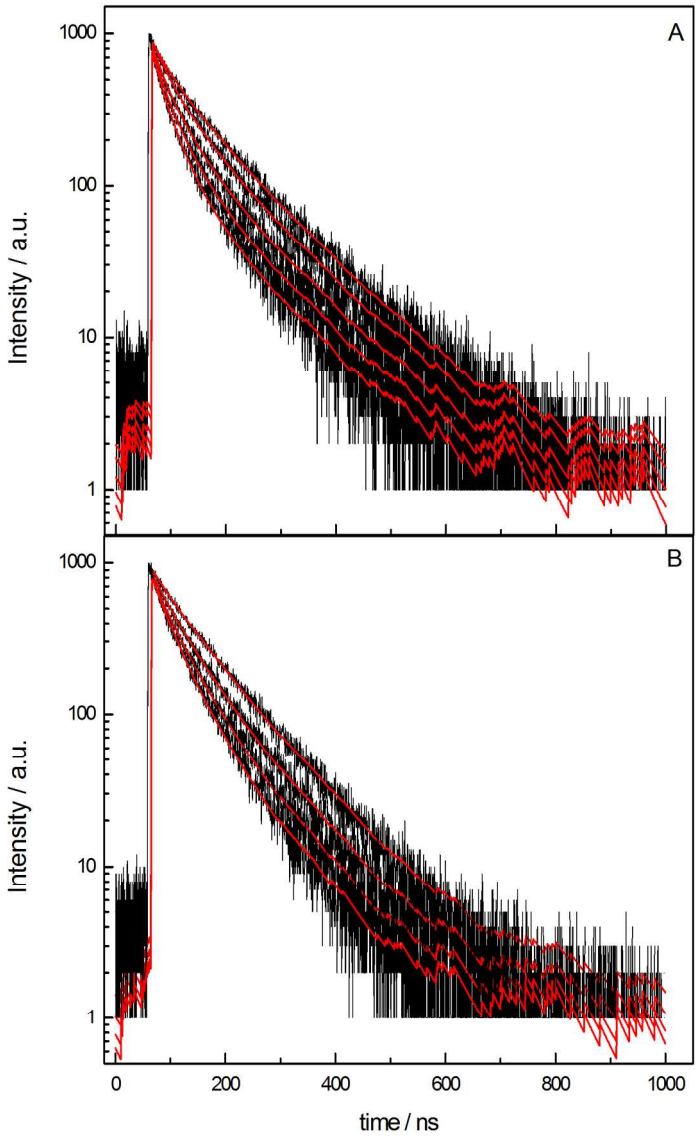

Figure 1. Intensity as a function of time plot from TRFQ of two [DTATf] samples in the presence of different quencher concentration. Black lines are the experimental values and red lines are the fit obtained as described in the manuscript, with the code available here. The blue line is the pulse .A) [DTATf] = 0.02 M; B) [DTATf] = 0.1 M. In Figure A, the [quencher] in 10^{-4} mol.L^{-1}, from top to bottom, are: 0.5; 1.0; 1.5; 2.1; 2.6, and, in B, are: 2.1; 4.1; 6.0; 8;2.

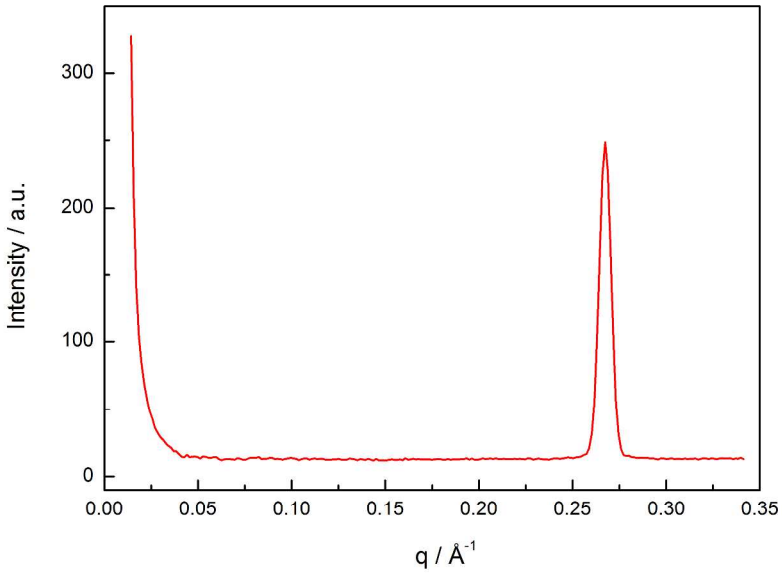

Figure 2. SAXS scattering curve of solid DTATf, at 298 K, showing a narrow peak at q ~ 0.27 $Å^{-1}$.

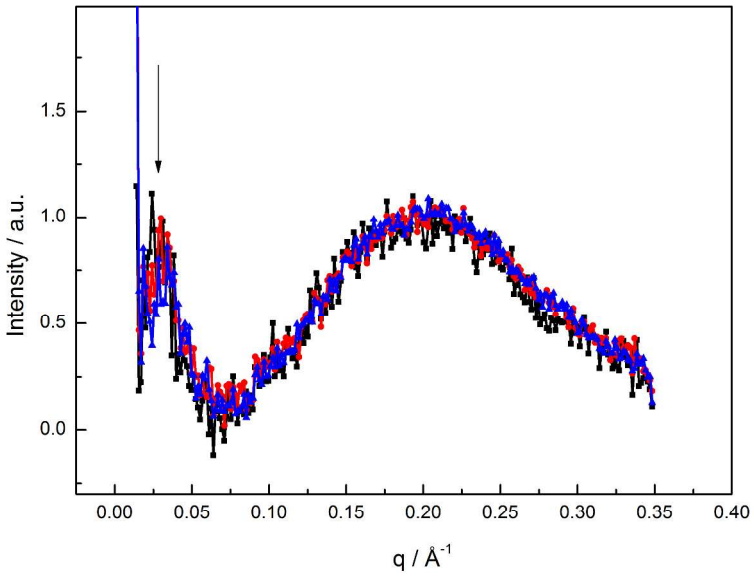

Figure 3. SAXS curve obtained for aqueous DTATf at different concentrations: (black, ■) 0.03 M; (red, ●) 0.08 M; (blue, ▲) 0.1 M. The arrow points to the interaction peak, which varied its position with [DTATf].

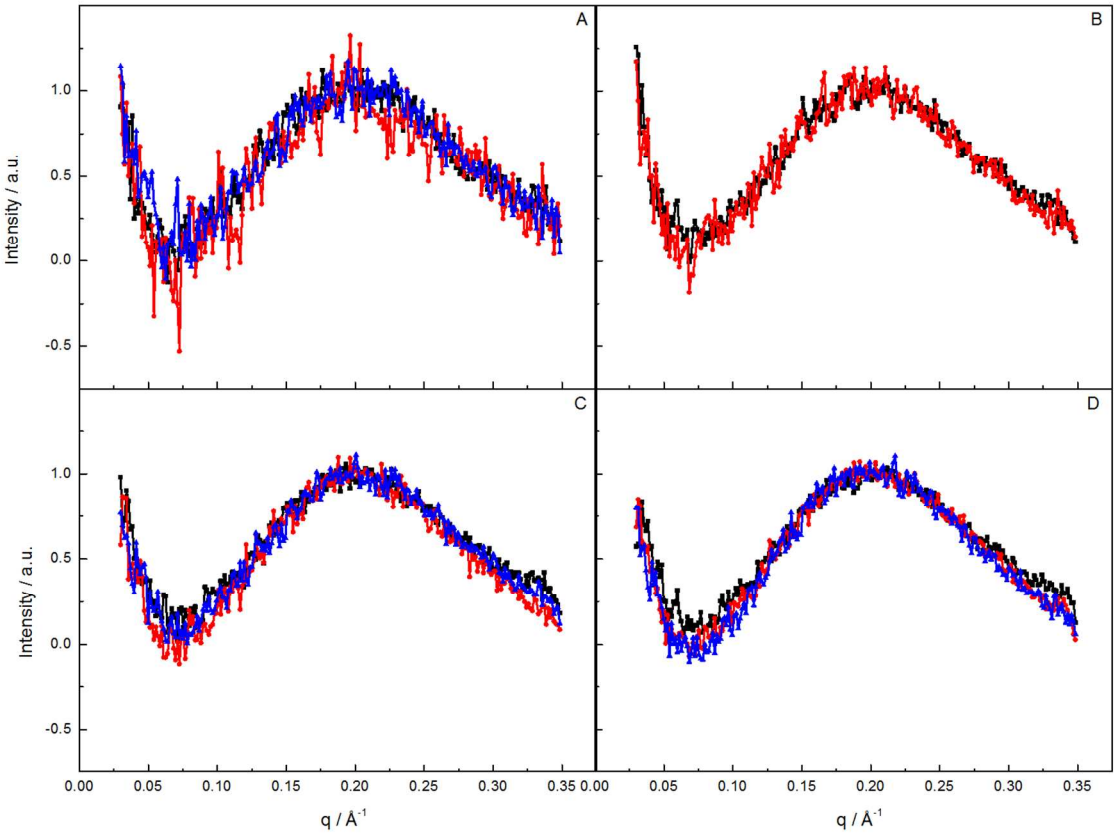

Figure 4. SAXS curves obtained for different [DTATf] in water and NaTf aqueous solutions: A) [DTATf] = 0.03 M; [DTATf] = 0.06 M; [DTATf] = 0.08 M; [DTATf] = 0.1 M. in all spectra: (black, ■) water; (red, ●) [NaTf] = 0.0075 M; (blue, ▲) [NaTf] = 0.03 M. The noise level decreased with [DTATf]. The curves of pure DTATf at different concentrations are nearly identical to respective curves of the surfactant in the presence of NaTf.

Table 1. T_1 and T_2 values for three DTAX 0.1 M samples.

H's	DTATf		DTAB		DTAMs	
	T_1	T_2	T_1	T_2	T_1	T_2
HP	0.907	0.090	1.056	0.800	1.056	0.800
H1	0.880	-	1.065	-	1.065	-
H2	0.883	-	1.018	-	1.018	-
H3	-	-	0.999	0.160	0.999	0.160
HG	1.060	0.051	1.163	0.160	1.163	0.160
H12	1.516	0.160	2.140	0.510	2.140	0.510

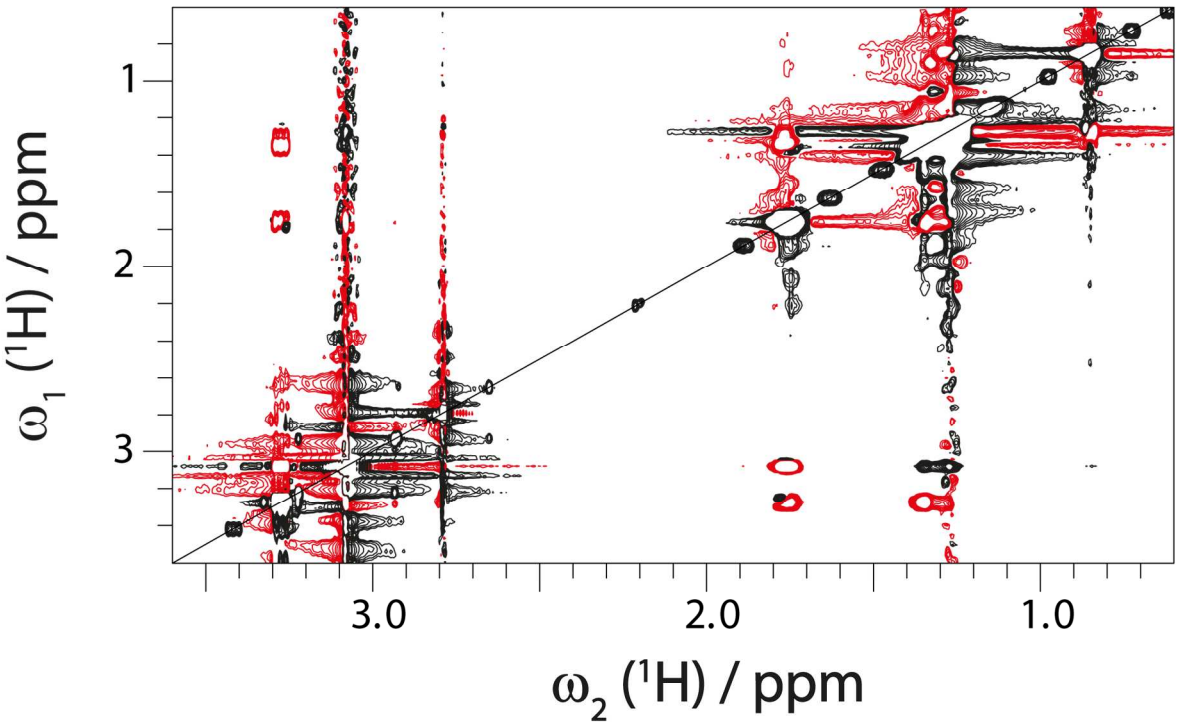

Figure 5. 2D 1H-1H NOE spectra of [DTAMs] = 0.02 M (cmc of DTAMs is 0.024 M at 318 K). Note to the fact that the NOE peak between H12 and HP shows a very small intensity, almost present only at the level of the noise.

Source code for treating TRFQ data of micellar systems. Programed for usage in Origin 8.5 or newer.

[General Information]

Function Name = Aggregationnumber; name show up in the select function dialog

Brief Description = Deconvolutes IRF from fluorescence decay response (Turro)

Function Type = User-Defined; source type, built-in, user-defined, external dll

Function Form = Equations; function form;

Function Source = N/A

Number Of Parameters = 4

Number Of Independent Variables = 1

Number Of Dependent Variables = 1

FunctionPrev = ExpGro1

Analytical Derivatives for User-Defined = 0

[Fitting Parameters]

Naming Method = user-defined

Names = I0,tau0i,A1,kq

Lower Bounds = --(I, Off),0(I, Off),0(I, Off),--(I, Off)

Upper Bounds = --(I, Off),--(I, Off),--(I, Off),--(I, Off)

Meanings = I(0) after IRF deconv,tau(0),Q*Nagg/C-CMC,quenching constant

Initial Values = 1(V),100(V),0.3(V),0.1(V)

Number Of Significant Digits = 0,0,0,0

Unit = ,,,

[Independent Variables]

x =

[Dependent Variables]

y =

[Constants]

[Formula]

```
Worksheet wks = Project.ActiveLayer();
	NLFitContext *pCtxt = Project.GetNLFitContext();
	if ( pCtxt )
	{
		// Vector for the output signal in each iteration.
		static vector vSignal;
		// If parameters were updated, we will recalculate the convolution result.
		BOOL bIsNewParamValues = pCtxt->IsNewParamValues();
		if ( bIsNewParamValues )
		{
			// Read sampling and response data from worksheet.
			Dataset dsSampling(wks, 0);
			Dataset dsResponse(wks, 2);
			int iSize = dsSampling.GetSize();

			vector vResponse, vSample;

			vResponse = dsResponse;
			vSample = dsSampling;

			vSignal.SetSize(iSize);
			vResponse.SetSize(iSize);
			vSample.SetSize(iSize);

			// Compute the exponential decay curve
			vSignal = I0 * exp( (-vSample/tau0i) -A1 * (1 - exp(-kq * vSample)));
			// Perform convolution
```

```
                int iRet = fft_fft_convolution(iSize, vSignal, vResponse);

        }

        NLSFCURRINFO    stCurrInfo;
        pCtxt->GetFitCurrInfo(&stCurrInfo);
        // Get the data index for the iteration
        int nCurrentIndex = stCurrInfo.nCurrDataIndex;
        // Get the evaluated y value
        y = vSignal[nCurrentIndex];
        // For compile the function, since we haven't use x here.
        x;
    }
```

[Constraints]

[Parameters Initialization]

[Initializations]

[After Fitting]

[Controls]

General Linear Constraints = 0

Initialization Scripts = 0

Scripts After Fitting = 0

Compile On Param Change Script = 1

Enable Parameters Initialization = 0

[Origin C Function Header]

```
#pragma warning(error : 15618)
#include <origin.h>
// Header files need to be included
#include <ONLSF.H>
#include <fft_utils.h>

// Add your special include files here.
// For example, if you want to fit with functions from the NAG library,
// add the header file for the NAG functions here.

// Add code here for other Origin C functions that you want to define in this file,
// and access in your fitting function.

// You can access C functions defined in other files, if those files are loaded and compiled
// in your workspace, and the functions have been prototyped in a header file that you have
// included above.
```

// You can access NLSF object methods and properties directly in your function code.

// You should follow C-language syntax in defining your function.

// For instance, if your parameter name is P1, you cannot use p1 in your function code.

// When using fractions, remember that integer division such as 1/2 is equal to 0, and not 0.5

// Use 0.5 or 1/2.0 to get the correct value.

// For more information and examples, please refer to the "User-Defined Fitting Function"

// section of the Origin Help file.

[Origin C Parameter Initialization Header]

[References]

[Compile Function]

Compile Parameters Initialization = 0

Compile = 1

[Derived Parameter Settings]

Names =

Meanings =

Unit =

[QuickCheck]

x=1

I0=1

tau0i=100

A1=0.3

kq=0.1