SUPPORTING INFORMATION FOR:

A Sweet Introduction to the Mathematical Analysis of Time-Resolved Spectra and Complex Kinetic Mechanisms: The Chameleon Reaction Revisited

Ricardo J. Fernández-Terán,[a]* Estefanía Sucre-Rosales,[b]
Lorenzo Echevarria,[bc] and Florencio E. Hernández[bd]

[a] *Department of Chemistry, University of Sheffield. Sheffield S3 7HF, United Kingdom.*

[b] *Department of Chemistry, University of Central Florida, Orlando, Florida 32816, United States.*

[c] *Departamento de Química, Universidad Simón Bolívar, Caracas 1080-A, AP 89000, Venezuela.*

[d] *CREOL/The College of Optics and Photonics, University of Central Florida, Orlando, Florida 32816, United States.*

**E-mail:* Ricardo.Fernandez@sheffield.ac.uk and Ricardo.FernandezTeran@gmail.com

Contents

Page

1. Safety Precautions S3

2. Overall Description of the MATLAB Code Provided with this Manuscript S3

3. Extended Discussion of First Order Kinetics Using Matrix Methods and the Matrix Exponential. S4

4. The Moore–Penrose Pseudoinverse: Least-Squares Fit S6

5. Description of the MATLAB Code `kineticsKmat_simu` S8

6. Description of the MATLAB Code `kineticsGEN_simu` S10

7. Description of the MATLAB Code `TRSpec_Bilinear_Simu` S13

8. Description of the MATLAB Code `TRSpec_Bilinear_FitDemo` S15

9. The Instrument Response Function (IRF) S16

10. Description of the MATLAB Code `IRFconvol`. S18

11. Description of the MATLAB Code `IRF_Kmat`. S19

12. Description of the MATLAB Code `TRSpec_Bilinear_FitData` S20

13. Discussion of the Kinetic Model and Fit Procedure. S21

14. Interpretation of Difference Spectra: Correlation with Transient Absorption Spectroscopy S22

15. Additional Figures. S24

16. References and Notes S25

1. Safety Precautions

$KMnO_4$ is a strong oxidizer, and NaOH is a corrosive base. Contact of either with skin and eyes may lead to irritation, can cause chemical burns, and may be harmful if swallowed or inhaled. Standard personal protective equipment (PPE), including a laboratory coat, safety goggles and nitrile gloves should be worn when handling these chemicals and their solutions. When in contact with the skin, $KMnO_4$ may produce brown stains due to the formation of MnO_2, which should disappear on their own.

The NaOH, sugar solutions, aqueous $KMnO_4$ solutions, and the reaction mixtures must be disposed in accordance with local regulations. It is important to recall that $KMnO_4$ is toxic to aquatic life.

2. Overall Description of the MATLAB Code Provided with this Manuscript

The code files provided with this Manuscript can be used to model, simulate and fit spectrokinetic data using either a coupled network of first-order reactions, or a generalised kinetic scheme described by a system of ODEs. The following pages describe each of the routines in relative detail.

Installation of the provided code files is straightforward: the user must download the ZIP file containing all MATLAB `.m` files, extract it to a folder and add this folder to the MATLAB path. Alternatively, the user can change the working directory to where these files were extracted and run them from there.

Running the code requires the files provided and the 'Optimization', 'Statistics and Machine Learning' and 'Curve Fitting' Toolboxes from MATLAB/MathWorks. The code was tested and developed in MATLAB, version R2021b.

3. Extended Discussion of First Order Kinetics Using Matrix Methods and the Matrix Exponential

Exponentiation of scalars [$\exp(x)$] is a familiar operation for many students, as it is introduced in elementary mathematics courses. The exponential of a matrix, in contrast, is very rarely discussed in introductory linear algebra courses.

For matrices, it is important to distinguish a matrix exponential [$\boldsymbol{P} = \mathrm{expm}(\boldsymbol{K})$] from an element-wise exponentiation [$\{\boldsymbol{Q}\}_{ij} = \exp(\{\boldsymbol{A}\}_{ij})$]. The former is a matrix function on square matrices, while the latter can be applied to any matrix. It follows that, in general, $\boldsymbol{P} \neq \boldsymbol{Q}$. Since the matrix exponential is the natural solution to a first order coupled differential equation, it is worthwhile mentioning two ways to calculate it and an alternative method.

First, since the exponential function can be expressed as a power series, we can write it as a Taylor series considering a matrix argument:

$$\mathrm{expm}(\boldsymbol{K}) = \sum_{n=0}^{\infty} \frac{1}{n!} \boldsymbol{K}^n \ , \quad \text{with} \quad \boldsymbol{K}^0 \equiv \mathbb{I}_{N\times N} \ , \tag{s1}$$

where $\mathbb{I}_{N\times N}$ is the identity matrix. For numerical calculations, the series in eq. s1 can be truncated after a given number of terms depending on the desired accuracy.[1]

In a second method, if $\boldsymbol{K}$ is diagonalisable, there exists a matrix $\boldsymbol{U}$ and a diagonal matrix $\boldsymbol{\Lambda}$ such that:

$$\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} \ , \tag{s2}$$

which leads to:[2]

$$\mathrm{expm}(\boldsymbol{K}) = \boldsymbol{U}\,\mathrm{expm}(\boldsymbol{\Lambda})\boldsymbol{U}^{-1} \tag{s3}$$

For a diagonal matrix, the matrix exponential is equal to the (conventional) exponential of the elements of the diagonal.[2] The interested reader can consult refs. 2–4 for further details.

The eigenvalue/eigenvector method described by Berberan-Santos and Martinho[5] presents an alternative to the calculation of the matrix exponential, and is summarised in the following.

If all eigenvalues (λ_i) of $\boldsymbol{K}$ are different ($\lambda_i \neq \lambda_j \ \forall i \neq j$), then there exist coefficients α_i and vectors $\vec{V}_i^0$ such that:

$$\boldsymbol{C}(t) = \sum_{i=1}^{N} \alpha_i \vec{V}_i^0 \exp(\lambda_i t) \tag{s4}$$

Let us define a vector $\vec{\alpha}$ containing the scaling factors, a vector $\vec{\Theta}(t)$ containing the scaled exponential functions, and a vector $\vec{\lambda}$ containing the eigenvalues of $\boldsymbol{K}$. Then, we can write:

$$\vec{\Theta}(t) \equiv \vec{\alpha} \times \exp(\vec{\lambda}t) = \begin{bmatrix} \alpha_1 \exp(\lambda_1 t) \\ \alpha_2 \exp(\lambda_2 t) \\ \vdots \\ \alpha_N \exp(\lambda_N t) \end{bmatrix} , \tag{s5}$$

where we use $\times$ to indicate element-wise multiplication of the vector of scaling coefficients and the vector of exponentials (the `.*` operation in MATLAB), while the dot represents the standard matrix multiplication (the `*` operator).

With that definition, the concentration matrix $\boldsymbol{C}(t)$ becomes simply:

$$\boldsymbol{C}(t) = \left(\boldsymbol{V} \cdot \left[\vec{\alpha} \times \exp(\vec{\lambda}t)\right]\right)^{\mathrm{T}} = \left[\boldsymbol{V} \cdot \vec{\Theta}(t)\right]^{\mathrm{T}} , \tag{s6}$$

where $\boldsymbol{V}$ is a matrix containing the eigenvectors of $\boldsymbol{K}$ as column vectors. $\vec{\alpha}$ can be calculated from:

$$\vec{\alpha} = \boldsymbol{V}^{-1} \cdot \vec{C}(0) , \tag{s7}$$

where $\vec{C}(0)$ is the vector of starting concentrations of each species.

4. The Moore–Penrose Pseudoinverse: Least-Squares Fit

As stated in the manuscript, our guess for the time-resolved spectral dataset originates from a known concentration matrix (dependent on a set of kinetic parameters, β) and an (unknown) spectral matrix:

$$\boldsymbol{D}_{\text{fit}}(t, \lambda; \beta) = \boldsymbol{C}(t; \beta) \cdot \hat{\boldsymbol{S}}(\lambda) \tag{s8}$$

The problem in eq. s8 could be analytically solved if, for example, $\boldsymbol{C}(t; \beta)$ was a square invertible matrix.

As is shown in elementary linear algebra courses, the matrix equation $\boldsymbol{AX} = \boldsymbol{B}$ can be solved for $\boldsymbol{X}$ if $\boldsymbol{A}$ is invertible, the solution being $\boldsymbol{X} = \boldsymbol{A}^{-1}\mathbf{B}$.

If $\boldsymbol{A}$ is square but not invertible, or if it is a rectangular matrix—as is normally the case for $\boldsymbol{C}(t)$—then we can define a unique matrix $\boldsymbol{A}^{+}$ which satisfies the Moore–Penrose conditions:[6]

$$\boldsymbol{A}\boldsymbol{A}^{+}\boldsymbol{A} = \boldsymbol{A} \tag{s9a}$$

$$\boldsymbol{A}^{+}\boldsymbol{A}\boldsymbol{A}^{+} = \boldsymbol{A}^{+} \tag{s9b}$$

$$(\boldsymbol{A}\boldsymbol{A}^{+})^{*} = \boldsymbol{A}\boldsymbol{A}^{+} \tag{s9c}$$

$$(\boldsymbol{A}^{+}\boldsymbol{A})^{*} = \boldsymbol{A}^{+}\boldsymbol{A} \tag{s9d}$$

where $(\cdots)^{*}$ denotes the conjugate transpose of a matrix. If $\boldsymbol{A}$ is invertible, then $\boldsymbol{A}^{+} = \boldsymbol{A}^{-1}$. If it is non-invertible but has linearly independent rows (eq. s10a) or columns (eq. s10b), then $\boldsymbol{A}^{+}$ can be computed as:[6]

$$\boldsymbol{A}^{+} = (\boldsymbol{A}^{*}\boldsymbol{A})^{-1}\boldsymbol{A}^{*}\ , \quad \text{thus} \quad \boldsymbol{A}^{+}\boldsymbol{A} = \mathbb{I} \tag{s10a}$$

$$\boldsymbol{A}^{+} = \boldsymbol{A}^{*}(\boldsymbol{A}\boldsymbol{A}^{*})^{-1}\ , \quad \text{thus} \quad \boldsymbol{A}\boldsymbol{A}^{+} = \mathbb{I} \tag{s10b}$$

We can use the Moore–Penrose pseudoinverse of $\boldsymbol{C}(t; \beta)$ to calculate $\hat{\boldsymbol{S}}(\lambda)$, thus solving the problem from eq. s8. It can be shown that $\boldsymbol{A}^{+}$ leads to the least-squares fit of a linear system whenever an exact solution does not exist (i.e. $\boldsymbol{A}$ is singular or rectangular).[6]

In simple terms, the Moore–Penrose pseudoinverse allows us to find the *best possible* solution for a *linear* system where the spectral matrix is our unknown, and we have a known concentration matrix and experimental time-resolved spectra. If the concentration matrix does not represent the experiment in a meaningful way, or if the bilinearity assumption breaks down, the quality of the determined spectral matrix—and hence the quality of $\boldsymbol{D}_{\text{fit}}(t, \lambda; \beta)$—will be unsatisfactory, and large/structured residuals will be observed [i.e. $\boldsymbol{D}_e - \boldsymbol{D}_{\text{fit}}(t, \lambda; \beta)$].

The Moore–Penrose pseudoinverse is related to the singular-value decomposition (SVD) by:[3]

$$\boldsymbol{D}^{+} = \boldsymbol{V}\boldsymbol{\Sigma}^{+}\boldsymbol{U}^{\mathrm{T}} \ , \tag{s11}$$

where $\{\boldsymbol{\Sigma}^{+}\}_{ij} = 1/\{\boldsymbol{\Sigma}\}_{ij}$ for every non-zero diagonal element. This relation allows for an alternative way to calculate the Moore–Penrose pseudoinverse of a matrix from its SVD representation.

5. Description of the MATLAB Code `kineticsKmat_simu`

The syntax of this function is `Ct = kineticsKmat_simu(t,K,C0,MakePlot,NameArray)`.

The function `kineticsKmat_simu` returns a time-dependent concentration matrix, $\boldsymbol{C}(t)$, of size $n_t \times N$ from a given time vector (t, with n_t time points), initial concentration vector $[\boldsymbol{C}(0)]$, and a rate matrix ($\boldsymbol{K}$). Note that the time vector must be positive for a physically meaningful description of the kinetic system. An additional boolean (true/false) input, `MakePlot`, determines whether a concentration vs. time plot will be made. The last input, `NameArray` contains the names of all chemical species in order to make a more meaningful legend. If an empty argument (`[]` in MATLAB) is given, the legend will simply state 'Species n' for each component.

For example, let us consider the system described by eq. s12:

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \underset{k_{2b}}{\overset{k_{2f}}{\rightleftharpoons}} \mathrm{X} \xrightarrow{k_3} \mathrm{P} \tag{s12}$$

This would lead to a rate matrix ($\boldsymbol{K}$) of the form:

$$\boldsymbol{K} = \begin{pmatrix} -k_1 & 0 & 0 & 0 \\ k_1 & -k_{2f} & k_{2b} & 0 \\ 0 & k_{2f} & -k_{2b} - k_3 & 0 \\ 0 & 0 & k_3 & 0 \end{pmatrix} \tag{s13}$$

The result of calling `kineticsKmat_simu` are shown in Figure **S1**, with the parameter values as described in the figure caption.

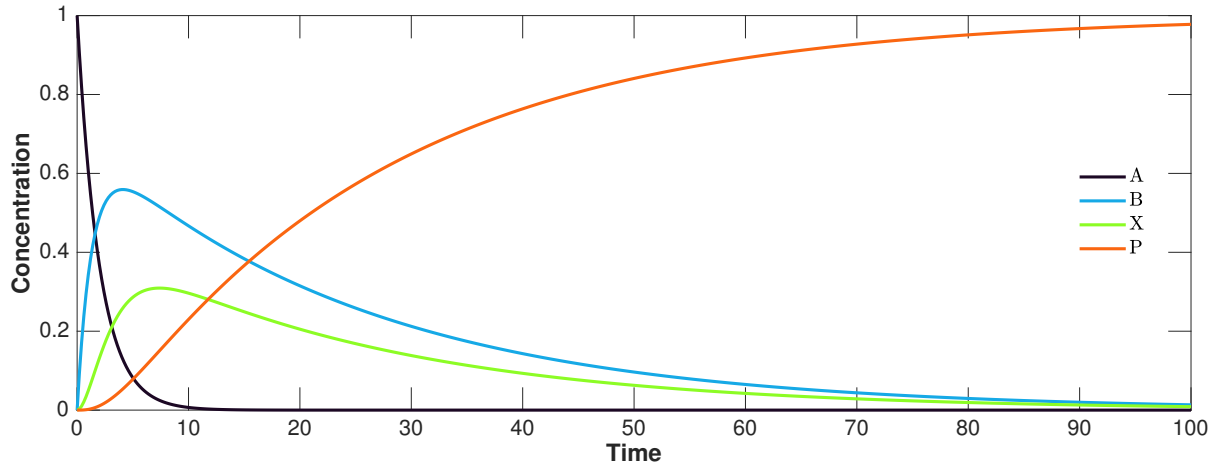

Figure S1. Simulated concentration profiles for the system described by eq. s12, with $k_1 = 0.5$, $k_{2f} = 0.3$, $k_{2b} = 0.4$, $k_3 = 0.1$, and $\boldsymbol{C}(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. Time and concentration units are arbitrary.

The following code can be used to reproduce Figure **S1**:

```
k1  = 0.5;
k2f = 0.3;
k2b = 0.4;
k3  = 0.1;

K   = [-k1  0       0         0
       k1   -k2f    k2b       0
       0    k2f     -k3-k2b   0
       0    0       k3        0];

t   = linspace(0,100,500)';
C0  = [1 0 0 0]';

MakePlot = 1;
SpeciesNames = ["A";"B";"X";"P"];

Ct = kineticsKmat_simu(t,K,C0,MakePlot,SpeciesNames);
```

The code can be easily modified and extended to chemical systems of coupled first-order reactions of arbitrary complexity simply by changing the functional form of $\boldsymbol{K}$.

In our MATLAB implementation, the vectors and matrices have the following sizes ($\rightarrow$ indicates the number of rows $\times$ columns): $t \rightarrow 1 \times n_t$; $\vec{\alpha}$ and $\vec{\lambda} \rightarrow N \times 1$; $\exp(\vec{\lambda}t) \rightarrow N \times n_t$; and $\boldsymbol{C}(t) \rightarrow n_t \times N$.

The eigenvalue/eigenvector method described in section 3 greatly simplifies the calculation of the concentration matrix, and can significantly speed up the calculations. The rate matrix only needs to be diagonalised once, after which the solutions become a sum of exponentials (as given by eq. s5), rather than the more involved calculation of the matrix exponential for each time step.

We found that this speed up becomes more significant the larger the number of time points (n_t), scaling roughly with $\sqrt{n_t}$ in the range $1 \leq n_t \leq 10^5$. In our test runs (using the same system as for Figure **S1**), the maximum speed-up factor was ~ 1800, for $n_t \approx 3 \times 10^4$. This can quickly become critical when fitting large datasets. A hard-coded option, `EigenVal` controls whether the eigenvalue/eigenvector (`EigenVal=1`) or the matrix exponential method (`EigenVal=0`) will be used for the actual calculation.

6. Description of the MATLAB Code `kineticsGEN_simu`

The syntax of this function is `Ct = kineticsGEN_simu(t,k,C0,ODEfunc,MakePlot,NameArray)`

In case more complex kinetics need to be modelled, we provide the function `kineticsGEN_simu`, which produces a similar output to `kineticsKmat_simu`, except that it can simulate chemical reactions of arbitrary complexity without being limited to first-order processes. This function is based on the code of our earlier implementation of kinetics for the Susceptible–Infected–Recovered viral spread model,[7] which can be used to introduce students to the simulation of more complex kinetics.

The `kineticsGEN_simu` function uses the numeric integration algorithms implemented in MATLAB to integrate the system of ordinary differential equations (ODEs) which are given as an input (`ODEfunc`), which is described in more detail in the following. We recommend consulting the appropriate MATLAB documentation to decide which integration method is more suitable to solve the ODE system under study. By default, our program will use the `ode23tb` function.

The input `ODEfunc` is a function handle to a function which returns an array of differentials respect to a concentration vector, linked together by rate constants. Both arguments need to be given as inputs: in the present case, the function is explicitly defined as `dTot = DiffEqs(C,k)`, where `C` is the concentration vector and `k` is a vector containing the corresponding rate constants. An in-depth example is presented below.

The additional inputs, `MakePlot` and `NameArray` have the same purpose and definition as above.

As an example of a more complex chemical reaction, let us consider the following reaction scheme:

$$\begin{cases} \mathrm{A} \xrightarrow{k_1} \mathrm{B} \underset{k_3}{\overset{k_2}{\rightleftharpoons}} \mathrm{X} & \text{(s14a)} \\ \mathrm{B} \xrightarrow{k_4} \mathrm{Y} & \text{(s14b)} \\ \mathrm{X} + \mathrm{Y} \xrightarrow{k_5} \mathrm{P} & \text{(s14c)} \\ 2\,\mathrm{P} \xrightarrow{k_6} \mathrm{P}_2 & \text{(s14d)} \end{cases}$$

Which leads to the following system of coupled differential equations:

$$\begin{cases} \dfrac{d[\mathrm{A}]}{dt} = -k_1[\mathrm{A}] & \text{(s15a)} \\ \dfrac{d[\mathrm{B}]}{dt} = +k_1[\mathrm{A}] - (k_2 + k_4)[\mathrm{B}] + k_3[\mathrm{X}] & \text{(s15b)} \\ \dfrac{d[\mathrm{X}]}{dt} = +k_2[\mathrm{B}] - k_3[\mathrm{X}] - k_5[\mathrm{X}][\mathrm{Y}] & \text{(s15c)} \\ \dfrac{d[\mathrm{Y}]}{dt} = +k_4[\mathrm{B}] - k_5[\mathrm{X}][\mathrm{Y}] & \text{(s15d)} \\ \dfrac{d[\mathrm{P}]}{dt} = +k_5[\mathrm{X}][\mathrm{Y}] - k_6[\mathrm{P}]^2 & \text{(s15e)} \\ \dfrac{d[\mathrm{P_2}]}{dt} = +k_6[\mathrm{P}]^2 & \text{(s15f)} \end{cases}$$

This system is considerably more complex and involves second-order kinetic terms (i.e. [X][Y] and $[\mathrm{P}]^2$), hence the matrix method discussed in the manuscript and applicable to the previous example (eq. s12) will no longer yield an analytical solution, as now the rate matrix would be time-dependent. Consequently, the system cannot be integrated analytically in every case, motivating the need for numerical integration.[5]

The result of calling `kineticsGEN_simu` for the system described in eq. s15 are shown in Figure **S2**, with the parameter values as described in the figure caption.

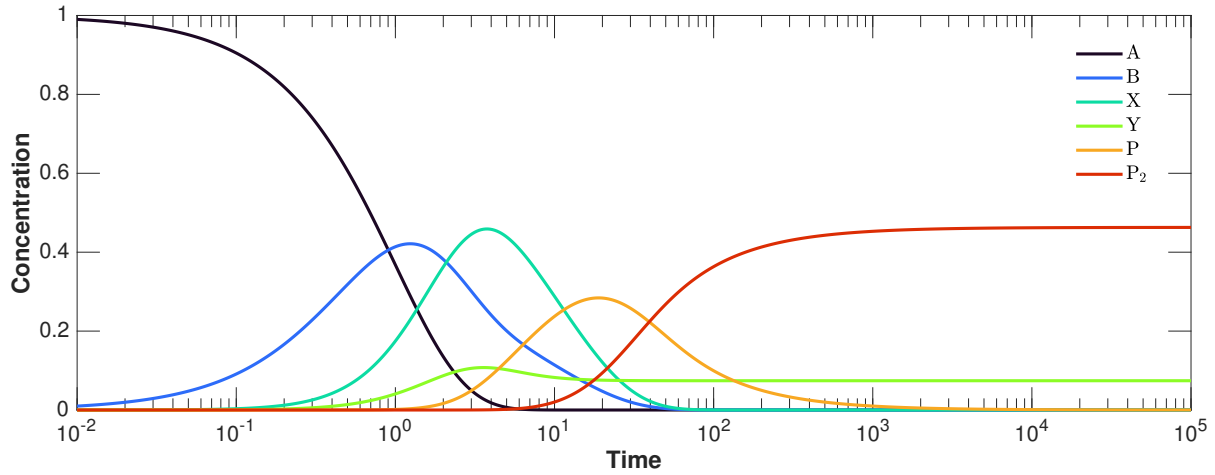

Figure S2. Simulated concentration profiles for the system described by eq. s14, with $k_1 = 1$, $k_2 = 0.7$, $k_3 = 0.3$, $k_4 = 0.15$, $k_5 = 0.8$, $k_6 = 0.1$, and $\boldsymbol{C}(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. Time and concentration units are arbitrary. The time axis in logarithmic scale highlights the processes occurring across many different timescales.

The following code can be used to reproduce Figure **S2** (Note that function definitions (i.e. `DiffEqs`) must appear at the end of the script/function, as per MATLAB directives):

```
k   = [1 0.7 0.3 0.15 0.8 0.1];
t   = [0 logspace(-2,5,500)]';
C0  = [1 0 0 0 0 0]';
MakePlot     = 1;
SpeciesNames = ["A";"B";"X";"Y";"P";"P$_{2}$"];
ODEfunc = @DiffEqs;
kineticsGEN_simu(t,k,C0,ODEfunc,MakePlot,SpeciesNames);

% Define the system of differential equations to solve
function dTot = DiffEqs(C,k)
    % Read species from C vector
    A   = C(1);
    B   = C(2);
    X   = C(3);
    Y   = C(4);
    P   = C(5);
    P2  = C(6);

    % Define the ODE system
    dA  = -k(1)*A;
    dB  = +k(1)*A - (k(2)+k(4))*B + k(3)*X;
    dX  = +k(2)*B - k(3)*X - k(5)*X*Y;
    dY  = +k(4)*B - k(5)*X*Y;
    dP  = +k(5)*X*Y - k(6)*P.^2;
    dP2 = +k(6)*P.^2;

    dTot= [dA;dB;dX;dY;dP;dP2];
end
```

Even more complex models describing the time dependence of the concentrations of chemical species have been derived, for example, in the case of diffusion-assisted electron transfer in solution, where classical kinetics do not hold.[8]

7. Description of the MATLAB Code `TRSpec_Bilinear_Simu`

The syntax of this function is `[WL,t,D,S] = TRSpec_Bilinear_Simu(MakePlots,NoisePercent)`.

The `TRSpec_Bilinear_Simu` function will generate synthetic time-resolved data from a given concentration matrix, $\boldsymbol{C}(t)$, and species associated spectra (SAS), $\boldsymbol{S}(\lambda)$. It will also give the time (`t`) and wavelength (`WL`) vectors as outputs, as well as the *theoretical* noise-free SAS (`S`).

In the first step, the spectral matrix is constructed from a series of Gaussian functions with different parameters. Then, a concentration matrix is simulated by invoking the `kineticsKmat_simu` function; and finally, time-resolved spectra are constructed using the definition given in the manuscript, $\boldsymbol{D}(t, \lambda) = \boldsymbol{C}(t) \cdot \boldsymbol{S}(\lambda)$.

Noisy datasets can be simulated by setting a non-zero value to the `NoisePercent` input argument. This will add randomly generated noise with an amplitude proportional to the maximum of the absolute scale of the dataset (with random numbers drawn uniformly from a normal distribution with zero mean and unit variance, using the `randn` MATLAB integrated function).

A series of plotting commands (controlled by the input argument, `MakePlots`) will generate the following figures:

1. Kinetic profiles of the species (plot obtained from `kineticsKmat_simu`, see Figure **S1**).

2. Contour plot of the entire time-resolved spectral dataset (Figure **S3A**).

3. Spectral traces as a function of time ("transient spectra").

4. Kinetic traces at selected wavelengths (Figure **S3B**).

Calling the provided script `TRSpec_Bilinear_Simu` with `MakePlots=1` as input argument will directly reproduce the components of Figure **S3**:

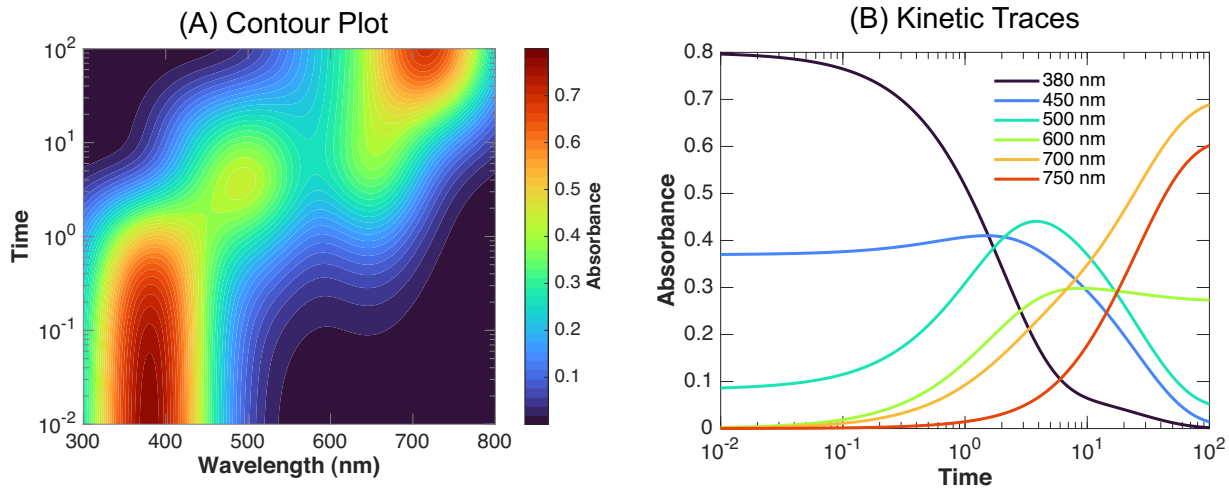

Figure S3. (A) Contour plot and (B) Kinetic traces along selected wavelengths for a simulated dataset. Kinetic parameters are identical to those of Figure **S1**.

8. Description of the MATLAB Code `TRSpec_Bilinear_FitDemo`

The `TRSpec_Bilinear_FitDemo` script contains a simulation which can be used to test the robustness of a given kinetic model. In brief, this script will repeatedly run the `TRSpec_Bilinear_Simu` script. If `NoisePercent` is set to a non-zero value, this will generate a new dataset every time the script is run. Each of the 'noisy' datasets will be fitted based on the kinetic model, and given starting parameters `p0`, with their corresponding lower/upper bounds (`LB` and `UB`, respectively).

Students and lecturers are encouraged to discuss the results obtained from these runs, to determine whether a given kinetic model is more susceptible to the effects of noise. As has been shown by Vauthey and co-workers,[9] the larger the difference in the rate constants, the more reliable the fits, also in the presence of larger amounts of noise. This activity per se represents a valuable module that can be incorporated into a (pre-/post-)laboratory discussion, or as part of a computer experiment where different models and ratios of kinetic constants and/or spectral features can be tested.

It is of fundamental importance that the distinction between the *best fitting* and a *physically meaningful* model is well understood, as it is the basis of a reasonable and sound analysis of a time-resolved spectral dataset, from which new physical insight may be obtained.

9. The Instrument Response Function (IRF)

To correctly take into account the measurement conditions, an instrument response function (IRF) needs to be introduced into the kinetic model. The IRF limits the fastest response that is observable in the experiment,[10] and can significantly alter the observed dynamics of events whose rate constants are similar to the time extent of the IRF (Figure **S4**).

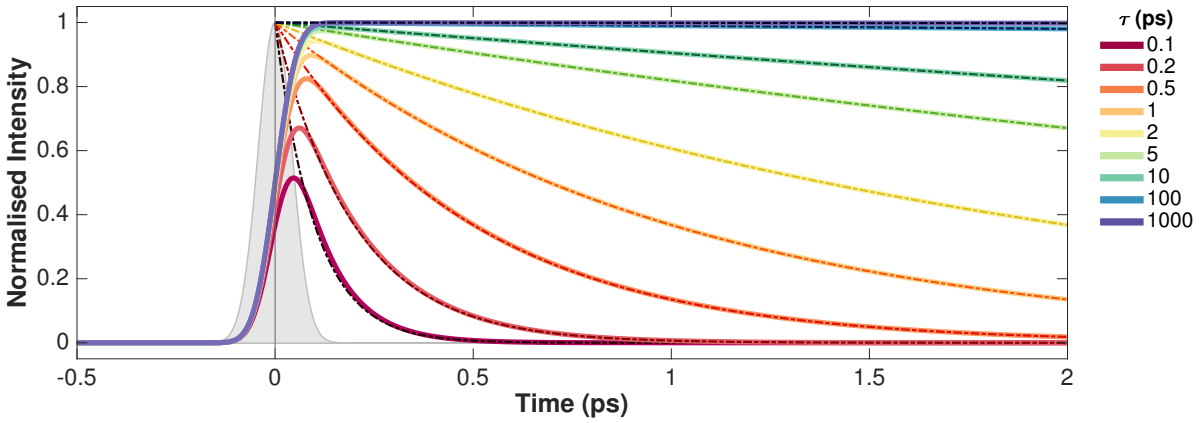

Figure S4. Illustration of the effect of the IRF on the observed kinetics for exponential decays of different lifetimes [$\exp(-t/\tau)$]. The IRF is shown as a shaded area, the instantaneous (δ-convolved) responses are shown with dash-dotted lines and the convolved responses are shown with solid lines. The IRF was modelled as a Gaussian with FWHM = 100 fs.

For chemical reactions, the IRF is typically the maximum between the mixing time of the solution and the interval between each consecutive measurement (i.e. the time it takes to record one spectrum). This is true for a reaction taking place in a stirred cuvette (e.g. our experimental conditions), or for stop-flow experiments (where two or more reagents are rapidly mixed in a specially designed cell to minimise the mixing time). The typical time resolution of a stopped flow system is in the order of 0.3–3 ms,[11,12] albeit they are usually operated in single-wavelength mode and hence recording time-resolved spectra would require several repetitions.

For a system that is excited with laser pulses, two limiting cases exist. In ultrafast experiments in the fs–ps timescales, the detection electronics do not have the time resolution to directly detect the spectra in real time. Instead, a short pulse acts as a probe/gate, and its arrival time is accurately controlled and varied in repeated experiments (e.g. with a delay stage). This is the case for pump–probe and fluorescence up-conversion spectroscopy, where the IRF corresponds to the convolution of the excitation (pump) and detection (probe/gate) pulses.[9,10] In these experiments, the theoretically achievable time resolution is in the order of tens to hundreds of femtoseconds.

The second limiting case corresponds to slower experiments, where spectra are either monitored in real time or via arrival time statistics. The former corresponds to time-resolved emission spectroscopy in the picosecond regime using a streak camera, and time-resolved absorption in the ns regime (flash photolysis); the latter corresponds to time-correlated single photon counting in the ns regime and slower. In both cases, the IRF corresponds to the convolution of the excitation (pump) pulse and the intrinsic response time of the detection electronics.[9,13]

When considering the IRF, the experimentally observed time-resolved signal, $S(t;\lambda)$, becomes:[9]

$$S(t;\lambda) = \mathrm{IRF}(t;\lambda) \otimes I(t;\lambda) \tag{s16}$$

$$= \int_{-\infty}^{+\infty} \mathrm{IRF}(t-\tau;\lambda) \cdot I(t;\lambda)\ d\tau \tag{s17}$$

where the $\otimes$ symbol represents the convolution operation between the IRF and the instantaneous response, $I(t;\lambda)$. In general, the IRF can be wavelength dependent.[9,13]

Calculation of the expected time-resolved signal obtained from a model can be performed both analytically (in certain cases) and numerically. For a Gaussian IRF centred at t_0 with known FWHM (Δ; defined as $\Delta = 2\sigma\sqrt{2\ln 2}$), convolved with a sum of exponential decays, we have:[9]

$$I(t,\lambda) = \sum_{i=1}^{N} A_i \cdot \frac{1}{2} \exp\left[-\frac{1}{\tau_i}\left(t - t_0 - \frac{\sigma^2}{2\tau_i}\right)\right] \cdot \left[1 + \mathrm{erf}\left(\frac{t - t_0 - \tau_i^{-1}\sigma^2}{\sigma\sqrt{2}}\right)\right] , \tag{s18}$$

Where τ_i are the lifetimes of each component. By combining eq. s18 with eq. s6, one can thus obtain an analytical expression for the convolution of a generalised first-order kinetic system with a Gaussian IRF, whenever all eigenvalues of $\boldsymbol{K}$ are different (see section 11 for more details). Formulas for more complex reaction schemes can be obtained by evaluating the integral in eq. s17, whenever a functional form of the kinetic response is known.

In the general case, however, an analytical formula cannot be derived, especially for complex kinetic systems. Rather than evaluating the integral in eq. s17 analytically, a numerical convolution can be performed instead. This also allows the use of an experimentally determined IRF instead of a simulated/approximate one, which may offer advantages or disadvantages depending on the technique and the experimental conditions.[10,14]

10. Description of the MATLAB Code `IRFconvol`

In the following, we will explain our MATLAB implementation of the numerical convolution between a Gaussian IRF and an arbitrary kinetic scheme, which can be imported as a function (e.g. `kineticsKmat_simu` and `kineticsGEN_simu`). The built-in functions `conv` and `conv2` can be used to perform the convolution operation between the two data arrays. In our implementation, we use `conv2` since it allows us to convolve all components of the concentration matrix simultaneously. This operation is trivial for regularly spaced data.

Irregularly spaced data needs to be interpolated from an equidistant grid spanning the entire time axis with a fine time resolution. This grid is constructed by taking the minimum spacing amongst the given time points (the `step` variable in the code). The number of additional points to be calculated becomes more significant for datasets spanning many orders of magnitude in time—where the time points are typically spaced in a logarithmic grid. Amongst the interpolation options available in MATLAB, we have chosen the '`spline`' method, which gave the smallest errors in test runs (i.e. difference between the numerical and analytical convolution for a first order system with the same parameters given for Figure **S1**), with a mean deviation of less than 0.1%.

In our implementation, we begin by considering a cut-off point after which the convolved response is identical that without the convolution. In other words, we define a cut-off time after which the effects of the IRF are negligible and the response is taken as the non-convolved data (to save computing time). This is defined in the code by the `endConv` and `cutoff` variables. The former has a default value of $1000 \times \Delta$, while the latter defaults to `endConv`-10Δ (where Δ is the FWHM of the Gaussian IRF). These parameters can be adjusted as needed. After these variables are correctly defined, the convolution is performed, and the data is interpolated on the original time axis (`t`). A final check is made to ensure that whenever $N = 1$ the data is output as a column vector, for compatibility with the other scripts (which expect the concentration matrix to be $n_t \times N$).

11. Description of the MATLAB Code `IRF_Kmat`

The syntax of this function is `[CConc,tIRF,simIRF,NConc] = IRF_Kmat(t,K,C0,w,t0)`.

The `IRF_Kmat` function will generate a concentration matrix $[\boldsymbol{C}(t)]$ from a model of coupled first order reactions, following the eigenvalue approach of `kineticsKmat_simu` (section 5), except that now the response is convolved with a Gaussian IRF of FWHM `w` and time-zero (`t0`). This is the function that should be used to fit experimental data whenever the response function is Gaussian, and works equally for an equidistant and non-equidistant time grid.

This function performs significantly faster (typically by a factor $\sim 10 - -20$) than `IRFconvol`, since the analytical equations are being used to directly calculate the convolved response. This approach is explained in more detail in the following. We have chosen the eigenvalue/eigenvector method because it presents a significant performance advantage, as discussed in the previous sections.

The approach closely follows the one presented in section 5, with a slight modification. In brief, by combining eq. s6 and eq. s18, we get:

$$\vec{\Psi}(t) = \frac{1}{2}\exp\left[\vec{\lambda}\left(t - t_0 + \vec{\lambda}\frac{\sigma^2}{2}\right)\right] \cdot \left[1 + \mathrm{erf}\left(\frac{t - t_0 + \vec{\lambda}\sigma^2}{\sigma\sqrt{2}}\right)\right] \ , \tag{s19}$$

where the FWHM (Δ) is related to σ by $\Delta = 2\sigma\sqrt{2\ln 2}$. Note the sign change respect to eq. s18, where τ_i (typically positive) are used instead of $\vec{\lambda}$—the latter is a vector containing all eigenvalues (typically negative) of the kinetic matrix, as defined previously. The function $\mathrm{erf}(x)$, also called Gauss error function, is defined as:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \tag{s20}$$

With this in mind, $\exp\vec{\lambda}t$ can be replaced for $\vec{\Psi}$ in eq. s5, leading to an IRF-convolved response $[\vec{\Theta}_{\mathrm{IRF}}(t)]$ of the form:

$$\vec{\Theta}_{\mathrm{IRF}}(t) \equiv \vec{\alpha} \times \vec{\Psi}(t) \tag{s21}$$

Finally, it follows that eq. s6 can be rewritten as:

$$\boldsymbol{C}(t) = \left[\boldsymbol{V} \cdot \vec{\Theta}_{\mathrm{IRF}}(t)\right]^{\mathrm{T}} \ , \tag{s22}$$

with the symbols $\cdot$ and $\times$ representing respectively the usual matrix-vector multiplication, and the element-wise multiplication (as described in section 5).

12. Description of the MATLAB Code `TRSpec_Bilinear_FitData`

The `TRSpec_Bilinear_FitData` script contains the central analysis routine of the manuscript. In its current implementation, it uses the `IRF_Kmat` function to calculate the concentration matrix, but the code can be easily adapted to use the `IRFconvol` function if needed (e.g. when a more complex mechanism is considered).

A time-resolved spectral dataset (`Dexp`; time along rows and pixels along columns), and the corresponding time (`t`) and wavelength (`WL`) column vectors need to be provided. These can be imported from experimental or simulated data.

In brief, the code will first ask the user for a data file, where the experimental dataset is contained. The fit function (`FitFunc`) must be defined (at the end of the script, as per MATLAB language specifications). This in turn requires either definition of a standard K matrix model (`Kmodel`, as a function of a vector of rate constants), implying the use of `IRF_Kmat`; or the implementation of a generalised kinetic mechanism via `kineticsGEN_simu` and `IRFconvol`.

Instead of using the Moore–Penrose pseudoinverse to calculate the spectra, we benefit from the native implementation of an efficient and accurate linear least squares routine in MATLAB, defined by the '\' operator (i.e. the system $\boldsymbol{A}\vec{x} = \boldsymbol{B}$ is solved for $\vec{x}$ by writing `x=A\B`). The interested reader can consult the MATLAB documentation for further details.

The fit options (as defined by `optimoptions`) can be tweaked to need, and the user must ensure that proper convergence criteria are set. After the fit is performed, a series of auxiliary code sections will plot the results (SAS/EAS, fitted dataset) and residuals (as a contour/2D plot).

Abundant comments in the code will help the reader decide which sections to change, and where to enter the corresponding parameters.

13. Discussion of the Kinetic Model and Fit Procedure

The kinetic model we have chosen does not consider the concentration of the sugars, since our reaction conditions allow us to approximate the reaction with a pseudo-first order mechanism. We take this to our advantage, since the sugars themselves do not have a characteristic spectroscopic signature in the UV-Vis, and hence their spectral contribution cannot be isolated.

In a typical time-resolved experiment, the concentration of all species is identically zero before time-zero. Immediately after the excitation pulse or trigger, a transient concentration of all species is produced. This transient concentration can decay back to a net zero concentration at $t \to \infty$, or lead to the formation of a photoproduct (which would have an 'infinite' lifetime).

In our case, however, the MnO_4^- species was present at $t < t_0$, hence we need to consider that while the initial concentrations are $C(0) = [1\ 0\ 0\ 0]^{\mathrm{T}}$, the concentration of species 1 at $t < t_0$ is also equal to 1. This is taken into account in the `IRFconvol_SP` routine, which sets the concentration of species 1 to 1 at times before t_0, to take into account the existence of the MnO_4^- species in the absolute spectra.

A standard fit procedure involves the minimisation of the square of the norm of the residuals, i.e. $\|\boldsymbol{D}_{\mathrm{fit}}(t,\lambda;\beta) - \boldsymbol{D}_e(t,\lambda)\|^2$. We have added a penalty for the non-negativity of the SAS by considering an additional factor:

$$\mathrm{SSRP} = \|\boldsymbol{D}_{\mathrm{fit}}(t,\lambda;\beta) - \boldsymbol{D}_e(t,\lambda)\|^2 + \gamma \cdot \mathrm{NNR}^2 \ , \tag{s23}$$

where SSRP is the sum of the squares of the residuals with a penalty function, NNR, defined as:

$$\mathrm{NNR} = \sum_{i,j} \{\boldsymbol{S}\}_{ij} \quad \text{such that} \quad \{\boldsymbol{S}\}_{ij} < 0 \ , \tag{s24}$$

and γ is a tuning factor (which we have taken as $\gamma = 1$) to account for the 'importance' of the NNR penalty. Simply put, NNR is the sum of all negative elements of the species-associated spectra matrix. Setting $\gamma = 0$ returns the traditional (unpenalised) least-squares.

14. Interpretation of Difference Spectra: Correlation with Transient Absorption Spectroscopy

As discussed in our manuscript, the species-associated spectra are positive for absolute absorption spectra. In transient photochemical experiments, where a chemical reaction is initiated by light, or when it is not possible to measure the spectrum of the 'initial conditions', a series of difference spectra are obtained instead. Difference spectra contain positive contributions (due to the absorption of excited-state species or photoproducts) and negative contributions (due to bleaching of the ground state and stimulated emission). In turn, the SAS of difference spectra (now called *SADS*, species-associated *difference* spectra) can contain a positive and a negative component. Note that, as before, a species will only manifest in the absolute/difference spectra whenever its extinction coefficient is non-zero.

With difference spectra, it is easier to directly visualise the zero delay, and the model can be the simplified to a standard kinetic model [i.e. with $C(t < 0) = 0$ for species 1]. Figure **S5** below illustrates a series of time-dependent difference spectra obtained for the glucose dataset (as a representative example), both as a contour plot and as a series of time-dependent spectra.

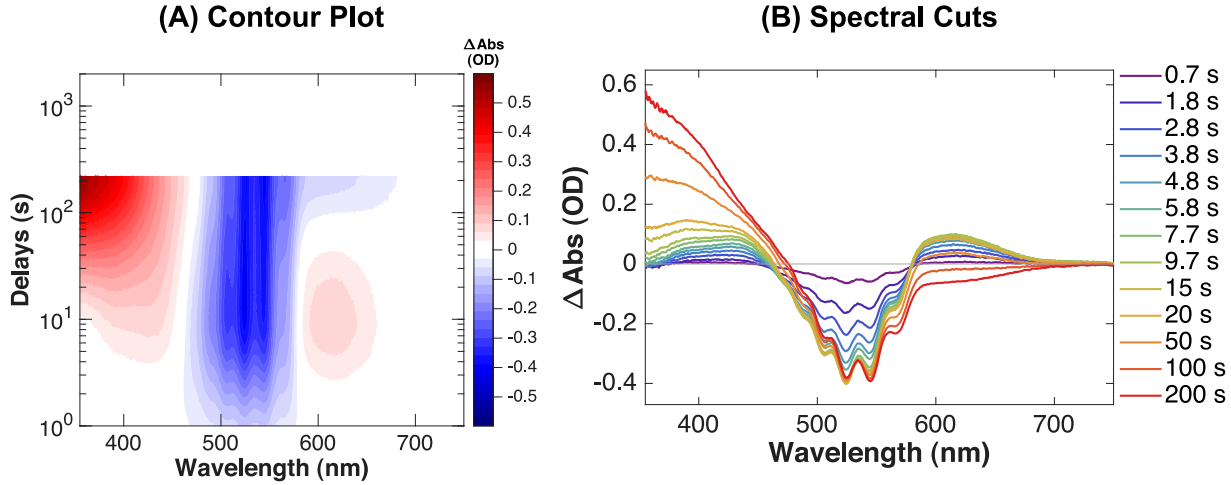

Figure S5. Difference time-resolved absorption spectra (ΔAbs) for the glucose dataset: (A) contour or 2D plot, and (B) spectral cuts at selected time points. The time axis has been shifted by the t_0 value obtained from the fit (2.1 s).

A caveat of difference spectra is that the ground-state and excited-state/photoproduct spectra can no longer be directly obtained from the difference spectra. Detailed knowledge of the excited fraction/ground-state spectrum—by measuring in a steady-state UV-Vis absorption and/or emission spectrometer (for a chemical reaction); or by taking into account the excitation parameters

of the setup (in the case of transient absorption)—can allow one to disentangle the positive and negative contributions to the observed difference signal.

As can be seen, the same main features that are evident in the absolute spectra can be observed here, namely the positive signals at ca. 400 and 600 nm at early times, and their shift to higher energies with time. The 'permanent' bleach observed corresponds to the irreversible loss of the MnO_4^- species, as described by the kinetic mechanism.

We believe that this discussion may ease the students' interpretation of difference spectra, serving as a primer into the domain of transient absorption spectroscopy in the ultrafast timescales.

15. Additional Figures

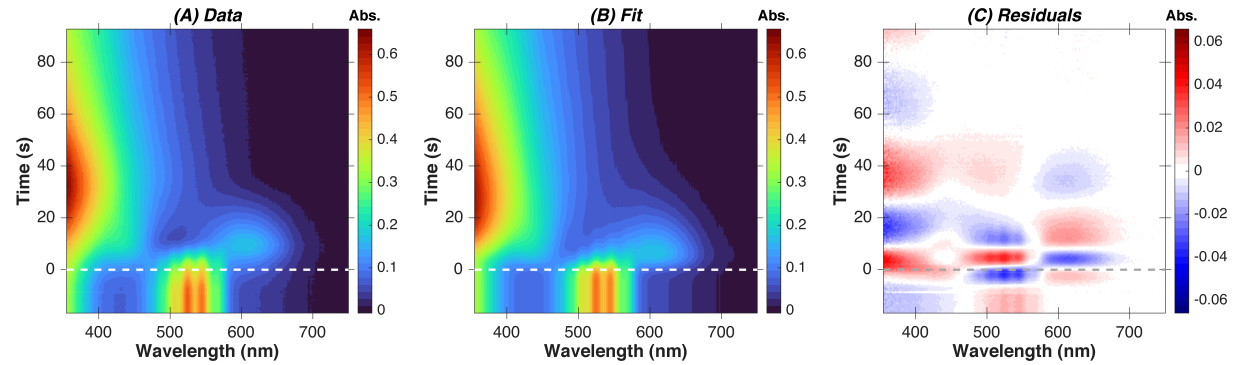

Figure S6. Data, fit, and residuals obtained for the oxidation of fructose with aqueous $KMnO_4$. The residual Z scale is expanded to 10% of the maximum absorbance.

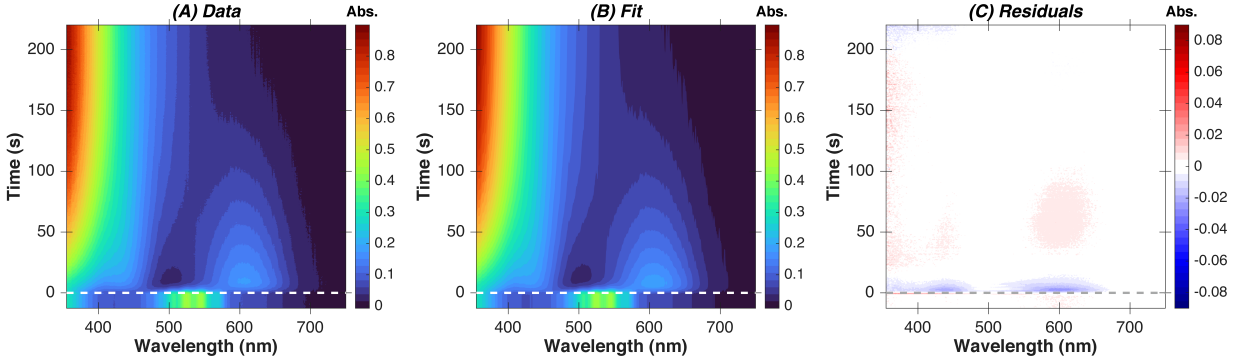

Figure S7. Data, fit, and residuals obtained for the oxidation of glucose with aqueous $KMnO_4$. The residual Z scale is expanded to 10% of the maximum absorbance.

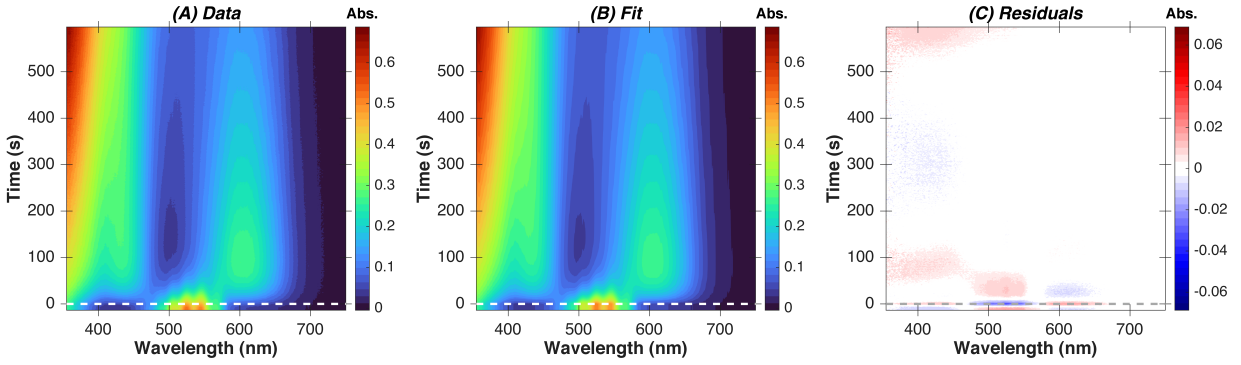

Figure S8. Data, fit, and residuals obtained for the oxidation of sucrose with aqueous $KMnO_4$. The data was truncated after 600 s. The residual Z scale is expanded to 10% of the maximum absorbance.

16. References and Notes

(1) In certain cases, the series will terminate after a finite number of terms. For example, if $\boldsymbol{K}$ is nilpotent, then for a certain $q > 0$, $\boldsymbol{K}^q = \mathbb{O}_{N\times N}$, after which all further terms are identically zero.

(2) R. Bellman, *Introduction to Matrix Analysis*, 2nd ed., Classics in Applied Mathematics (Society for Industrial and Applied Mathematics, Santa Monica, California, 1997).

(3) G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences (Johns Hopkins University Press, Baltimore, 2013).

(4) B. Hall, *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, 2nd ed., Graduate Texts in Mathematics (Springer International Publishing, London, 2015).

(5) M. N. Berberan-Santos and J. M. Martinho, The integration of kinetic rate equations by matrix methods, J. Chem. Educ. **67**, 375 (1990).

(6) A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, 2nd ed., CMS Books in Mathematics (Springer-Verlag, New York, 2003).

(7) E. Sucre-Rosales, R. Fernández-Terán, D. Carvajal, L. Echevarría, and F. E. Hernández, Experience-Based Learning Approach to Chemical Kinetics: Learning from the COVID-19 Pandemic, J. Chem. Educ. **97**, 2598 (2020).

(8) G. Angulo, A. Rosspeintner, B. Lang, and E. Vauthey, Optical transient absorption experiments reveal the failure of formal kinetics in diffusion assisted electron transfer reactions, Phys. Chem. Chem. Phys. **20**, 25531 (2018).

(9) J. S. Beckwith, C. A. Rumble, and E. Vauthey, Data analysis in transient electronic spectroscopy – an experimentalist's view, Int. Rev. Phys. Chem. **39**, 135 (2020).

(10) I. H. Van Stokkum, D. S. Larsen, and R. Van Grondelle, Global and target analysis of time-resolved spectra, Biochim. Biophys. Acta - Bioenerg. **1657**, 82 (2004).

(11) Q. H. Gibson and L. Milnes, Apparatus for rapid and sensitive spectrophotometry., Biochem. J. **91**, 161 (1964).

(12) R. V. Prigodich, A stopped-flow kinetics experiment for the physical chemistry laboratory using non-corrosive reagents, J. Chem. Educ. **91**, 2200 (2014).

(13) C. Slavov, H. Hartmann, and J. Wachtveitl, Implementation and evaluation of data analysis strategies for time-resolved optical spectroscopy, Anal. Chem. **87**, 2328 (2015).

(14) J. S. Beckwith, B. Lang, J. Grilj, and E. Vauthey, Ion-Pair Dynamics upon Photoinduced Electron Transfer Monitored by Pump-Pump-Probe Spectroscopy, J. Phys. Chem. Lett. **10**, 3688 (2019).