J. Electrochem. Sci. Eng. **8(2)** (2018) 171-181; DOI: http://dx.doi.org/10.5599/jese.507

Open Access : : ISSN 1847-9286

https://pub.iapchem.org/ojs/index.php/JESE

Original scientific paper

A short introduction to digital simulations in electrochemistry: simulating the Cottrell experiment in NI LabVIEW

Soma Vesztergom

Eötvös Loránd University, Department of Physical Chemistry, Pázmány Péter sétány 1/A, 1117 Budapest, Hungary

E-mail: vesztergom@chem.elte.hu; Tel.: +36-20-461-2429

Received: February 9, 2018; Revised: March 28, 2018; Accepted: March 28, 2018

Abstract

A brief introduction to the use of digital simulations in electrochemistry is given by a detailed description of the simulation of Cottrell's experiment in the LabVIEW programming language. A step-by-step approach is followed and different simulation techniques (explicit and implicit Euler, Runge–Kutta and Crank–Nicolson methods) are applied. The applied techniques are introduced and discussed on the basis of Padé approximants. The paper might be found useful by undergraduate and graduate students familiarizing themselves with the digital simulation of electrochemical problems, as well as by university lecturers involved with the teaching of theoretical electrochemistry.

Keywords

Explicit and implicit Euler method; Runge–Kutta methods; Crank–Nicolson method; Padé approximants.

Introduction

At the 6th Regional Symposium on Electrochemistry for South-East Europe (6th RSE–SEE), I gave a keynote lecture [1] on the kinetics of hydrogen evolution in mildly acidic solutions. Under these chemical conditions — at low to moderate pH values ($1 < pH < 4$) — the reduction of H^+ ions and that of water molecules both give a significant contribution to the measured cathodic current. In my talk, I presented digital simulations (and a simplified analytical model) that can describe polarization curves measured on rotating disk electrodes immersed into mildly acidic solutions.

Results presented in my talk at the 6th RSE–SEE were published [2] just a few weeks before the conference. Instead of further discussing these results, I chose to focus more in this paper on the applied method of digital simulation (DS).

DS presents a powerful tool that can be used for the qualitative understanding, as well as for the quantitative description of electrode processes. When carrying out DS, we create a numerical

model of the electrochemical system within a computer and allow this model to evolve according to a set of simple algebraic equations. These equations are derived from the physical laws that govern the electrochemical system under study. In effect, we carry out a simulation of the experiment with the aim to extract numeric representations of current, concentration profiles, potential transients, and so on [3]. The results of digital simulations can then be compared to measured quantities, so that the physical parameters of the system can be fine-tuned in subsequent simulations, until the measured data are reproduced at a desired accuracy.

DS thus allows the determination of kinetic parameters, such as charge transfer rates and diffusion coefficients, *etc*. DS can yield important quantitative information even for complex electrochemical systems where the often-complicated set of partial differential equations, describing the transformation and movement of material, can be solved in closed form with difficulty or not at all [3].

In this paper I give a very brief and practice-oriented introduction to digital simulations and their use in electrochemistry. Instead of reviewing the topic of digital simulations in detail — several good textbooks are dedicated to the subject and are recommended to the reader [3,4] — I choose a rather straightforward approach: that is, giving a step-by-step guide to the simulation of a simple electrochemical problem, that is known as Cottrell's experiment [5].

The paper is intended to be a starting point for undergraduate or graduate students, interested in digital simulations. Furthermore, the paper may also be found useful by university lecturers, as it contains a simpler than usual description and classification of different partial differential equation solving methods, based on Padé's approximants.

Experimental

Cottrell's experiment, first described in 1903 by F.G. Cottrell, is one of the oldest problems in electrochemistry that has a well-known analytical solution. The simulation of Cottrell's experiment is thus expedient, since the success of any simulation can directly be tested [6] by comparison to analytically obtained results.

In order to understand Cottrell's experiment, consider [7] a large planar electrode, of surface A, initially at rest, in contact with a semi-infinite layer of unstirred solution that contains some excess electrolyte and a small amount of a redox-active species R with a bulk concentration of c^{∞}. At the instant $t = 0$, the electrode potential is suddenly changed to a value at which the species R is oxidized to some product O in the one-electron reduction

$$R \rightarrow O + e^- \tag{1}$$

and the concentration of R, close to the electrode surface, is brought to essentially zero.

In this experiment, the reaction rate (and thus, the current) is determined by the rate of transport at which the species R is resupplied at the electrode surface. That is, the partial differential equation governing the system is that describing Fick's law,

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}. \tag{2}$$

Equation (2) is a second-order partial differential equation where c denotes the concentration and D the diffusion coefficient of the species R, while x and t denote the space and time variables, respectively. To obtain an analytical solution of Equation (2), we take two boundary conditions into consideration: *i.)* that the concentration of the species R is zero at the electrode surface at any $t > 0$ and *ii.)* that the concentration at infinite distance from the electrode surface never changes and remains equal to c^{∞}

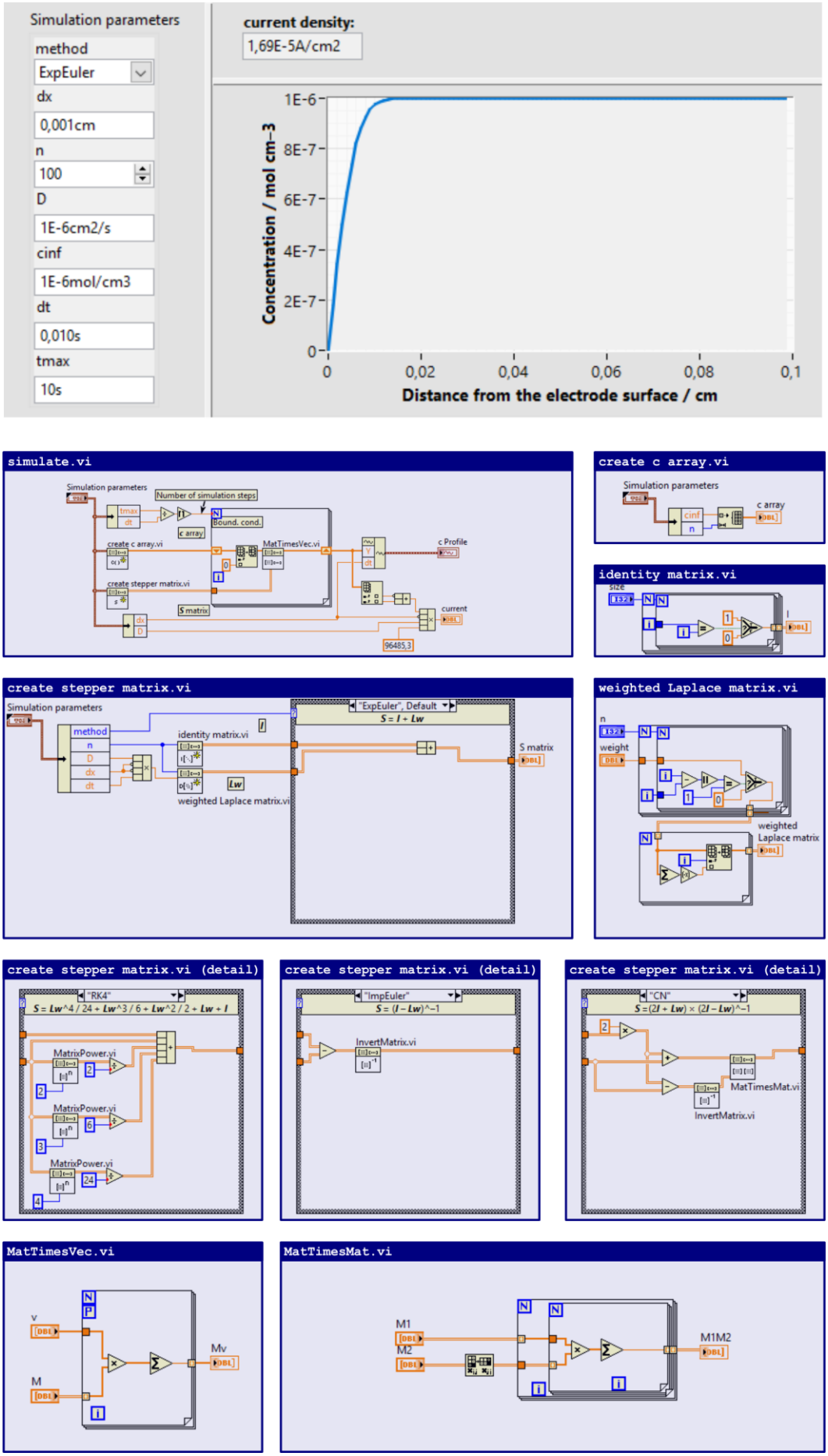

Figure 1. *The graphical user interface (top) of a simple LabVIEW program that simulates Cottrell's experiment. Snippets (code details) are shown in the blue boxes; see the text for explanation*

Taking the above boundary conditions into consideration leads to the solution:

$$c(x,t) = c^{\infty}\,\mathrm{erf}\left(\frac{x}{\sqrt{4Dt}}\right), \tag{3}$$

where the function erf(*z*) denotes Gauss's error function, defined by the integral:

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z \exp\left(-u^2\right)\mathrm{d}u.$$

The current density $j(t)$, passing the electrode surface at any time $t > 0$ is thus:

$$j(t) = F\,D\lim_{x\to 0}\left(\frac{\partial}{\partial x}c(x,t)\right) = F\,c^{\infty}\sqrt{\frac{D}{\pi t}}, \tag{4}$$

where F = 96485.3 C mol^{-1} is Faraday's constant.

Equation (3) makes it straightforward to calculate the concentration of species R at any given time $t > 0$ at a distance x measured from the electrode surface; while from Equation (4), the well-known Cottrell equation, the current density can be calculated for any $t > 0$. These analytical formulae will be used below for testing the results of digital simulations.

Simulation approaches

LabVIEW, developed by National Instruments, is an easy-to-use, interactive, graphical programming language that helps users write sophisticated programs and applications in a short amount of time without needing a computer science degree [8]. The popularity of LabVIEW continuously increases amongst students and scientific researchers, also in electrochemistry, mostly due to its versatility in measurement automation. However, LabVIEW also provides simple means for digital simulations and I therefore chose to present the basics of simulation algorithms in this programming language. Figure 1, discussed extensively below, shows the user interface of a program written for the simulation of Cottrell's experiment, as well as some details of the code. The main routine is that shown by the `simulate.vi` snippet.

In most digital simulation schemes, we consider the electrolyte solution in terms of discrete and small (Δx wide) volume elements, as illustrated by Figure 2. When we apply this approach to the simulation of Cottrell's experiment in NI LabVIEW, we create a 1-dimensional array of concentrations, containing n entries, which we denote by ***c***. At the start of the simulation process, all entries of the array ***c*** will be set equal to c^{∞} (the bulk concentration). The initialization of the ***c*** array in LabVIEW is shown by the `create c array.vi` snippet of Figure 1.

Each i entry of the array ***c*** (the index i can take values between 0 and $n-1$ in LabVIEW) corresponds to the concentration of the species R at a certain distance from the electrode surface, as shown by Figure 2. For accurate simulations, the Δx width of the control volumes must be small, thus to include a big enough space in the simulations, a sufficiently high number of the control volumes is required. (See Table 1 for numeric parameter values, used for the simulations presented in this paper.)

When carrying out digital simulations, we iteratively modify the entries of the ***c*** array, as shown by the `simulate.vi` snippet of Figure 1. Each iteration step represents a Δt time-step in which we replace the first element of ***c*** with zero (thereby realizing the boundary condition of the experiment) and then multiply the array (vector) ***c*** with a so-called stepper matrix, denoted here by ***S***. As the iteration proceeds, we continuously "update" the ***c*** array, which at the end of the iteration will turn to be a discrete representation of the concentration profile, corresponding to the time t.

Table 1. *Simulation parameters.*

Symbol	Meaning	Value
Δx	width of the control volumes	0.001 cm
n	number of the control volumes	100
D	diffusion coefficient	10^{-6} cm^2 s^{-1}
c^{∞}	bulk concentration	10^{-6} mol cm^{-3}
Δt	time-step	varied between 1 ms and 5 s

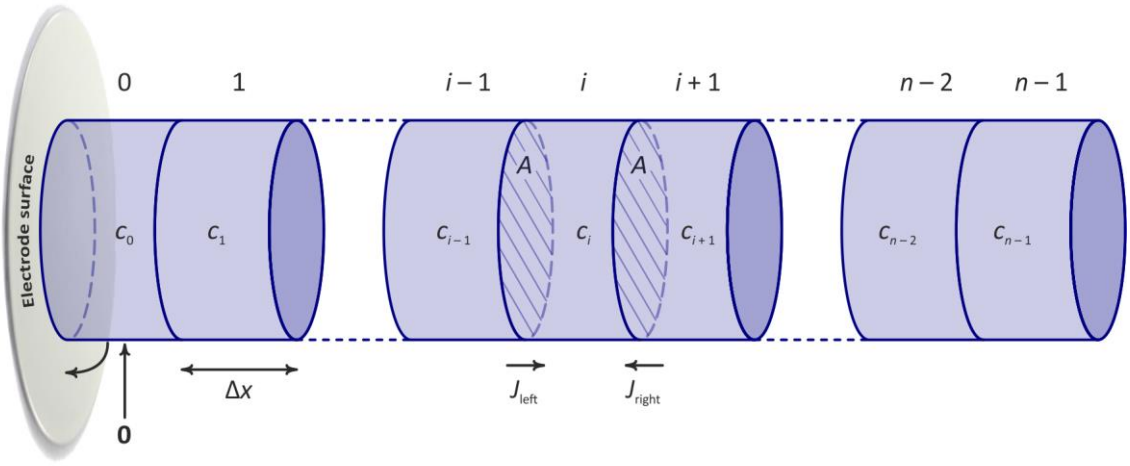

Figure 2. *Scheme for the digital simulation of Cottrell's experiment*

The multiplication of the ***c*** array with the stepper matrix ***S*** in each iteration step mimics the role of transport in the simulation process, thus the construction of the stepper matrix has a deep impact on both the accuracy and the speed of the simulation.

To understand the role of the stepper matrix ***S*** in digital simulations, let us consider Figure 2, and assume that in each control volume shown, the concentration values are different. Let us then try to estimate the changes of concentration in the control volume marked by the index i during a small time-step Δt. Fick's first law of diffusion allows us to calculate the J_{left} and J_{right} fluxes of material from the left-hand $(i-1)$ and the right-hand $(i+1)$ neighbours, directed to the i^{th} control volume as:

$$J_{\text{left}} = \frac{1}{A}\frac{\Delta n_{\text{left}}}{\Delta t} = -D\frac{c_i - c_{i-1}}{\Delta x} \tag{5a}$$

and

$$J_{\text{right}} = \frac{1}{A}\frac{\Delta n_{\text{right}}}{\Delta t} = -D\frac{c_i - c_{i+1}}{\Delta x}, \tag{5b}$$

where Δn_{left} and Δn_{right} denote the amount of substance arriving to the i^{th} cell from its left and right neighbours, respectively. In each time-step Δt, the amount of substance that enters the i^{th} cell is $\Delta n = \Delta n_{\text{left}} + \Delta n_{\text{right}}$, and taking into consideration the volume of the cell ($A\Delta x$), this results in a

$$\frac{\Delta c_i}{\Delta t} = \frac{\left(J_{\text{left}} + J_{\text{right}}\right)A}{A\Delta x} = D\frac{c_{i-1} - 2c_i + c_{i+1}}{\Delta x^2} \tag{6}$$

concentration change in the i^{th} control volume. We note that the fraction on the right-hand side of Equation (6) approximates the second spatial derivative of the concentration profile, and that Equation (6) is in fact a discretized version, with respect to both space and time, of Equation (2), Fick's second law.

Equation (6) dictates that in a time-step of Δt duration, the change of concentration in the i^{th} control volume depends on the concentrations in the i^{th}, $(i-1)^{\text{th}}$ and $(i+1)^{\text{th}}$ control volumes, as well as on the model parameters D, Δx and Δt. Equation (6) may be re-written in the form of a matrix-vector equation as:

$$\frac{\Delta \boldsymbol{c}}{\Delta t} = \frac{D}{\Delta x^2} \boldsymbol{L}\,\boldsymbol{c} \tag{7}$$

by defining the so-called Laplacian matrix $\boldsymbol{L}$, an $n \times n$ matrix as:

$$L_{i,k} = \begin{cases} 1 & \text{if } i = k \pm 1 \\ -\deg(i) & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}, \tag{8}$$

where $\deg(i)$ is the number of neighbours of the i^{th} control volume. For the simulation scheme shown in Figure 2, the Laplacian — a negative semi-definite matrix — takes the following form:

$$\boldsymbol{L} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}. \tag{9}$$

Note that Equation (7) contains a finite difference (with respect to time) on its left-hand side, approximating a time derivative. For infinitesimally small time-steps, Equation (7) may be re-written as:

$$\frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}t} = \frac{D}{\Delta x^2} \boldsymbol{L}\,\boldsymbol{c}, \tag{10}$$

which is a differential equation for the vector $\boldsymbol{c}$. Assuming that the concentration vector $\boldsymbol{c}^{(k)}$ is known at any k^{th} step of the simulation, the $\boldsymbol{c}^{(k+1)}$ vector in the next step, Δt time later, could be calculated by solving Equation (10) in the form:

$$\boldsymbol{c}^{(k+1)} = \exp\left(\boldsymbol{L}_{\text{w}}\right) \boldsymbol{c}^{(k)}, \tag{11}$$

where $\boldsymbol{L}_{\text{w}}$ denotes the so-called weighted Laplacian, defined as:

$$\boldsymbol{L}_{\text{w}} = \frac{D\,\Delta t}{\Delta x^2} \boldsymbol{L}. \tag{12}$$

The LabVIEW code used for the calculation of the weighted Laplacian is shown by the `weighted Laplace matrix.vi` snippet of Figure 1.

As can be seen in Equation (11), the "ideal" choice for the stepper matrix $\boldsymbol{S}$ we introduced before would be that $\boldsymbol{S} = \exp\left(\boldsymbol{L}_{\text{w}}\right)$; the multiplication of the concentration vector $\boldsymbol{c}$ in each simulation step with this matrix could model the time evolution of the system, leaving the resolution of spatial discretization the only factor that limits the accuracy of calculations.

The problem is, however, that the exponential of the weighted Laplacian is not known and can only be approximated. One feasible way of approximating the exponential of a matrix is to truncate its Taylor series at a given order; another, more accurate way, is to use its Padé approximant [9].

It is this latter approach which I will follow here, as many digital simulation techniques — such as the explicit and implicit Euler, the Runge–Kutta and the Crank–Nicolson methods [4] — may be interpreted as techniques relying on the use of different Padé approximants.

Given a function $f(x)$ and two integers $m \geq 0$ and $n \geq 0$, the Padé approximant of the function $f(x)$ of order $[m/n]$ is the rational function:

$$\mathrm{R}(x) = \frac{\sum_{j=0}^{m} a_j x^j}{\sum_{j=0}^{n} b_j x^j}, \tag{13}$$

which agrees with $f(x)$ to the highest possible order, which amounts to:

$$f(0) = \mathrm{R}(0) \tag{14a}$$

$$f'(0) = \mathrm{R}'(0) \tag{14b}$$

$$f''(0) = \mathrm{R}''(0) \tag{14c}$$

$$f^{(m+n)}(0) = \mathrm{R}^{(m+n)}(0) \tag{14d}$$

Equation (14) is a system of Equations, by solving which the parameters a_j and b_j of Equation (13) can be determined up to one degree of freedom; it is usually assumed that $b_0 = 1$.

In what follows, I will use $\mathrm{P}_{m,n}(x)$ to note the Padé approximant of the exponential function $\exp(x)$. Exact expressions for $\mathrm{P}_{m,n}(x)$ are tabulated in Table 2 for different values of m and n. Some Padé approximants to the exponential function are plotted, together with $\exp(x)$, in Figure 3 to show the "goodness" of these approximations.

Padé approximants, when applied to matrices instead of scalars, form the basis of many known simulation techniques, as will be discussed below.

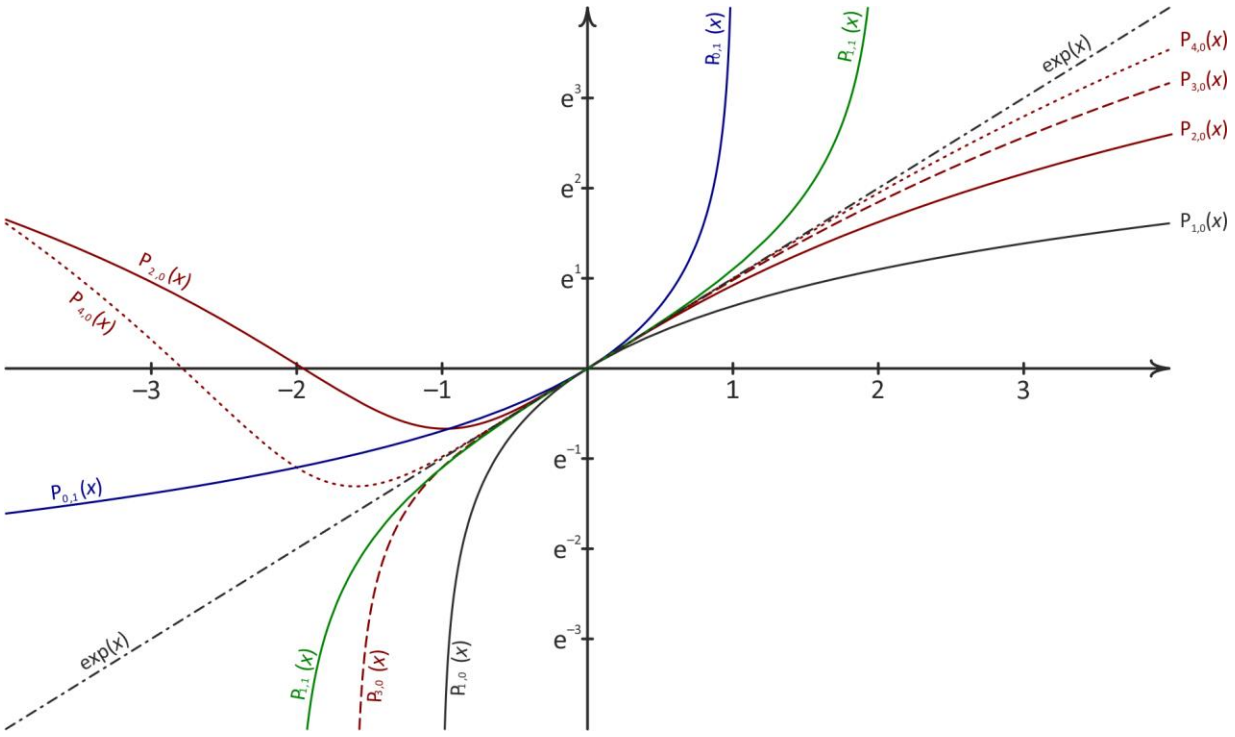

Figure 3. *Some Padé approximants* $\mathrm{P}_{m,n}(x)$ *of the function exp(x)*

Table 2. *Some Padé approximants* $P_{m,n}(x)$ *of the function exp(x)*

$m \backslash n$	**0**	**1**	**2**	**3**
0	1	$\frac{1}{-x+1}$	$\frac{1}{\frac{1}{2}x^2-x+1}$	$\frac{1}{-\frac{1}{6}x^3+\frac{1}{2}x^2-x+1}$
1	$x+1$	$\frac{\frac{1}{2}x+1}{-\frac{1}{2}x+1}$	$\frac{\frac{1}{3}x+1}{\frac{1}{6}x^2-\frac{2}{3}x+1}$	$\frac{\frac{1}{4}x+1}{-\frac{1}{24}x^3+\frac{1}{4}x^2-\frac{3}{4}x+1}$
2	$\frac{1}{2}x^2+x+1$	$\frac{\frac{1}{6}x^2+\frac{2}{3}x+1}{-\frac{1}{3}x+1}$	$\frac{\frac{1}{12}x^2+\frac{1}{2}x+1}{\frac{1}{12}x^2-\frac{1}{2}x+1}$	$\frac{\frac{1}{20}x^2+\frac{2}{5}x+1}{-\frac{1}{60}x^3+\frac{3}{20}x^2-\frac{3}{5}x+1}$
3	$\frac{1}{6}x^3+\frac{1}{2}x^2+x+1$	$\frac{\frac{1}{24}x^3+\frac{1}{4}x^2+\frac{3}{4}x+1}{-\frac{1}{4}x+1}$	$\frac{\frac{1}{60}x^3+\frac{3}{20}x^2+\frac{3}{5}x+1}{\frac{1}{20}x^2-\frac{2}{5}x+1}$	$\frac{\frac{1}{120}x^3+\frac{1}{10}x^2+\frac{1}{2}x+1}{-\frac{1}{120}x^3+\frac{1}{10}x^2-\frac{1}{2}x+1}$
4	$\frac{1}{24}x^4+\frac{1}{6}x^3+\frac{1}{2}x^2+x+1$	$\frac{\frac{1}{120}x^4+\frac{1}{15}x^3+\frac{3}{10}x^2+\frac{4}{5}x+1}{-\frac{1}{5}x+1}$	$\frac{\frac{1}{360}x^4+\frac{1}{30}x^3+\frac{1}{5}x^2+\frac{2}{3}x+1}{\frac{1}{30}x^2-\frac{1}{3}x+1}$	$\frac{\frac{1}{840}x^4+\frac{2}{105}x^3+\frac{1}{7}x^2+\frac{4}{7}x+1}{-\frac{1}{210}x^3+\frac{1}{14}x^2-\frac{3}{7}x+1}$

The $P_{1,0}$ *(explicit Euler) method*

The explicit Euler method is the most straightforward one applied for the solution of partial differential equations. It is based on the $P_{1,0}(x)$ Padé approximation

$$\exp\left(\boldsymbol{L}_{\mathrm{w}}\right) \approx \boldsymbol{I} + \boldsymbol{L}_{\mathrm{w}} = \boldsymbol{S}, \tag{15}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix (see the `identity matrix.vi` snippet of Figure 1). When plugged into Equation (11), it yields that

$$\boldsymbol{c}^{(k+1)} = \left(\boldsymbol{I} + \boldsymbol{L}_{\mathrm{w}}\right)\boldsymbol{c}^{(k)}. \tag{16}$$

The construction of the stepper matrix $\boldsymbol{S}$, used for simulations based on the explicit Euler method, is shown in the `create stepper matrix.vi` snippet of Figure 1.

Note that as $\Delta\boldsymbol{c} = \boldsymbol{c}^{(k+1)} - \boldsymbol{c}^{(k)}$, Equation (16) can also be directly obtained by rearranging Equation (10). The explicit Euler method can thus be interpreted as approximating the differential $\mathrm{d}\boldsymbol{c}/\mathrm{d}t$ in Equation (10) with the finite difference $\Delta\boldsymbol{c}/\Delta t$. Consequently, the explicit Euler method, a linear approximation, may only yield accurate results when a small enough time-step Δt is used. In-fact, when $\Delta t > \Delta x^2/2D$, the explicit Euler method fails to converge and gives erroneous results, as will be discussed later.

The $P_{2,0}$*,* $P_{3,0}$ *and* $P_{4,0}$ *(Runge–Kutta) methods*

The overall error of the explicit Euler method can be decreased by taking higher order terms into consideration. This can be achieved by using the $P_{2,0}$, $P_{3,0}$ and $P_{4,0}$ Padé approximants, which are in-fact Taylor series expansions of different orders to the exponential function. This forms the basis of the so-called 2nd, 3rd and 4th order Runge–Kutta methods [10]. While the explicit Euler method could be interpreted as a technique that takes the slope (but not the curvature) of the concentration *vs.* time dependence into account, Runge–Kutta methods aim to correct this error.

When applying Runge–Kutta methods, the stepper matrix $\boldsymbol{S}$ is constructed as

$$\exp\left(\boldsymbol{L}_{\mathrm{w}}\right) \approx \boldsymbol{I} + \boldsymbol{L}_{\mathrm{w}} + \frac{1}{2}\boldsymbol{L}_{\mathrm{w}}^2 = \boldsymbol{S} \qquad \text{for the 2nd order,} \tag{17}$$

$$\exp\left(\boldsymbol{L}_{\mathrm{w}}\right) \approx \boldsymbol{I} + \boldsymbol{L}_{\mathrm{w}} + \frac{1}{2}\boldsymbol{L}_{\mathrm{w}}^2 + \frac{1}{6}\boldsymbol{L}_{\mathrm{w}}^3 = \boldsymbol{S} \qquad \text{for the 3rd order,} \tag{18}$$

and

$$\exp\left(\boldsymbol{L}_{\mathrm{w}}\right) \approx \boldsymbol{I} + \boldsymbol{L}_{\mathrm{w}} + \frac{1}{2}\boldsymbol{L}_{\mathrm{w}}^2 + \frac{1}{6}\boldsymbol{L}_{\mathrm{w}}^3 + \frac{1}{24}\boldsymbol{L}_{\mathrm{w}}^4 = \boldsymbol{S} \qquad \text{for the 4th order method.} \tag{19}$$

The construction of the stepper matrix ***S***, used for simulations based on the 4th order Runge–Kutta method, is shown by the respective `create stepper matrix.vi (detail)` snippets of Figure 1. Note that — similarly to the Euler method discussed above — Runge–Kutta methods are still *explicit*; that is, in each iteration step, they calculate the state of a system at a later time from the state of the system at the current time.

The $P_{0,1}$ (implicit Euler) method

The implicit (also called backward [4]) Euler method differs largely from explicit methods. The method is *implicit*, meaning that it finds a solution for the state of the system at a later time by solving an equation that involves both the current state of the system and the later (yet unknown) state. The stepper matrix used by the implicit Euler method, as also shown by the respective `create stepper matrix.vi (detail)` snippet of Figure 1, is

$$\exp\left(\boldsymbol{L}_{\mathrm{w}}\right) \approx (\boldsymbol{I} - \boldsymbol{L}_{\mathrm{w}})^{-1} = \boldsymbol{S} \tag{20}$$

Note that due to the implicit nature of the method, the creation of the stepper matrix involves matrix inversion. This clearly requires some extra computation, yet implicit methods can become very useful for the solution of stiff problems where explicit methods require the application of impractically small Δ*t* values. As implicit methods are numerically stable, they can be applied with larger time-steps, and as usually the matrix inversion in Equation (20) has to be carried out only once (at the beginning of the iteration), implicit methods still require small computation times.

The $P_{1,1}$ (Crank–Nicolson) method

The method named after Crank and Nicolson [11] was developed for the numerical solution of partial differential equations, but it lies on the basis of the trapezium method of solving ordinary differential equations. The method is *semi-implicit* and is unconditionally stable; although when applied for stiff systems with large time-steps it may still be prone to spurious (decaying) oscillations. The stepper matrix used by the Crank–Nicolson method, as also shown by the respective `create stepper matrix.vi (detail)` snippet of Figure 1, is

$$\exp\left(\boldsymbol{L}_{\mathrm{w}}\right) \approx (\boldsymbol{I} + \tfrac{1}{2}\boldsymbol{L}_{\mathrm{w}})(\boldsymbol{I} - \tfrac{1}{2}\boldsymbol{L}_{\mathrm{w}})^{-1} = \boldsymbol{S}. \tag{21}$$

Comparison of the different simulation methods

As expected, applying Padé approximations of different order for the construction of the stepper matrix ***S*** yields solutions that also behave differently. The efficiency of the methods was tested by simulating concentration profiles corresponding to *t* = 5 s, using the parameter values of Table 1 and Δ*t* values ranging between 1 ms and 5 s. Each simulated concentration profile was compared to the theoretically expected one, Equation (3), and the square-root of the mean squared deviation was plotted as a function of the used time-step in Figure 4.

Figure 4 clearly shows a major difference with respect to the stability of fully explicit (explicit Euler and Runge–Kutta) and implicit (implicit Euler and Crank–Nicolson) methods; at about $\Delta t > \Delta x^2 / 2D$, all explicit methods begin to diverge (see also Figure 5). Compared to the explicit Euler, the 2nd order Runge–Kutta method brings a significant improvement of the error; this improvement is less significant for the 3rd and 4th order Runge–Kutta methods.

The stability of the implicit Euler and the Crank–Nicolson methods is better (see also Figure 5), although the error of these methods is also significant at larger time-steps.

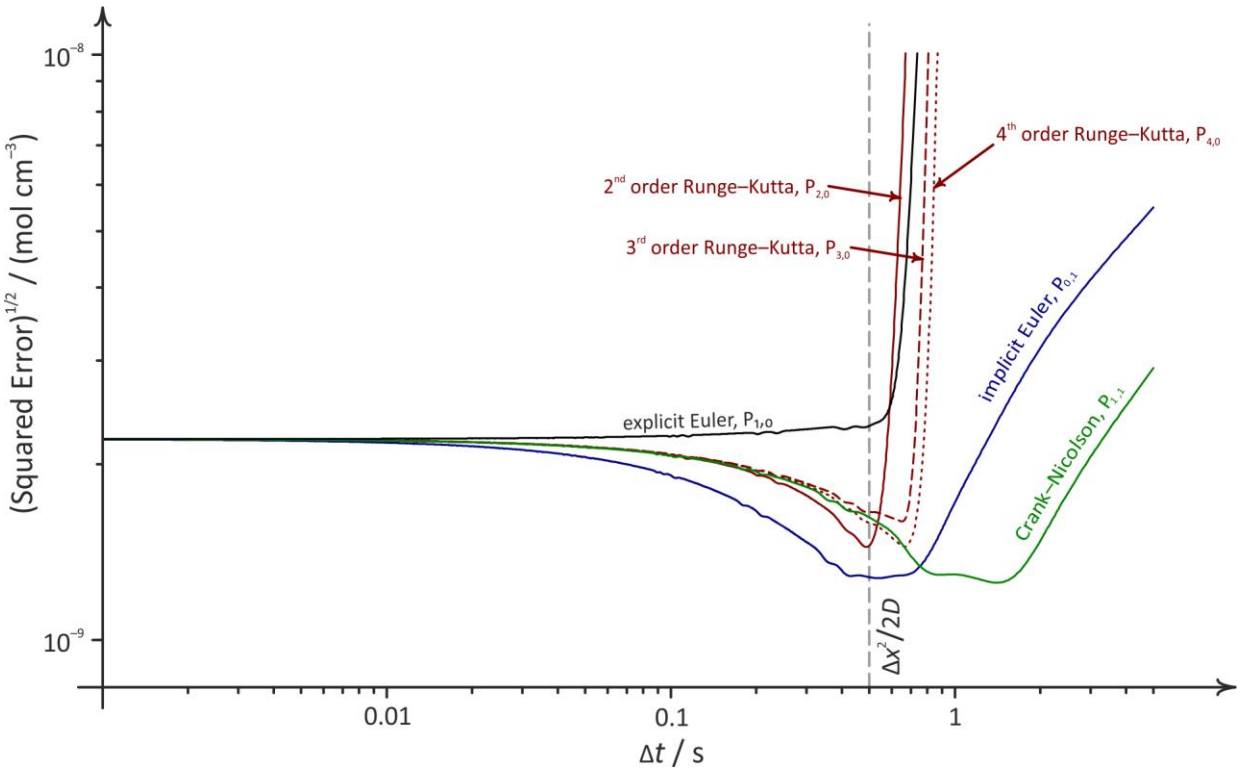

Figure 4. *The error of six different digital simulation methods, as a function of the applied time-step. Other simulation parameters are tabulated in Table 1*

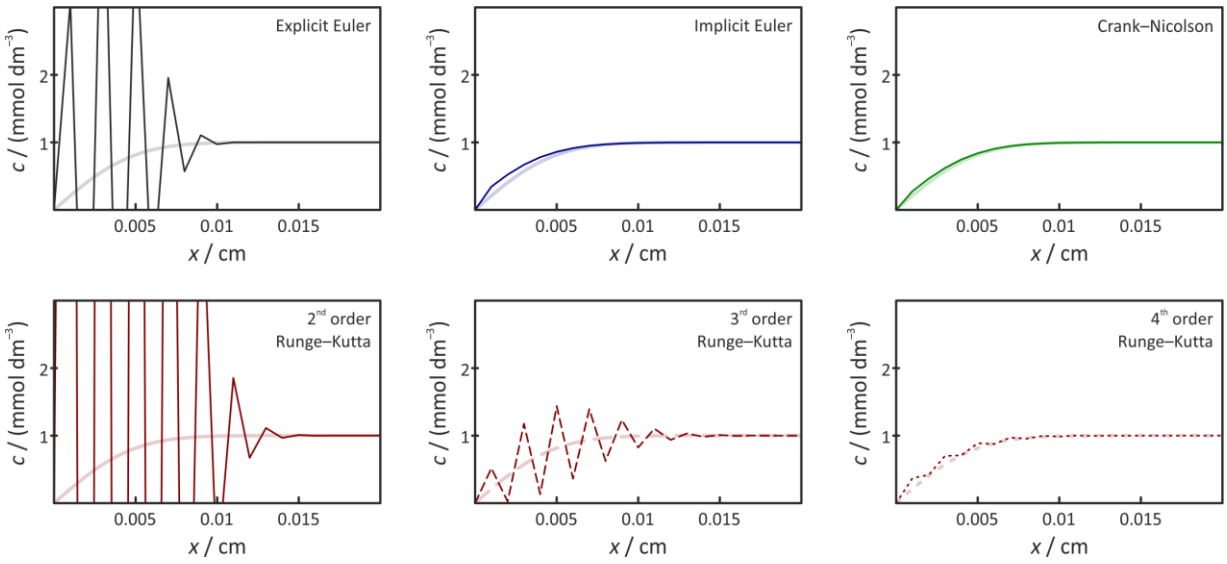

Figure 5. *Concentration profiles simulated using six different methods for t = 7 s, using a relatively large time-step (Δt = 700 ms). Faded thick curves show theoretical profiles, as calculated from Equation (3). For the values of other simulation parameters, see Table 1*

Summary

In this paper I attempted to describe numerical methods used for the digital simulation of a rather simple, however instructive electrochemical problem, Cotrell's experiment. The same techniques may also be used — by modifying the near-electrode boundary condition — for the simulation of more complicated electrochemical experiments, such as cyclic voltammetry. The described simulation strategies may also be extended to take into account homogeneous reactions [12], or effects related to ohmic drop [13,14]. The further discussion of more complicated systems is, however, beyond the scope of this paper: my only intention here was to

give a starting point for undergraduate or graduate students who decided to familiarize themselves with digital simulations. Accordingly, I attempted to describe basic numerical procedures in a simpler than usual manner, following a classification scheme based on Padé's approximants. Readers interested in the topic of digital simulation in more detail are referred to other sources [3,4].

Acknowledgements: *I would like to thank my dear friends and colleagues Éva Valkó and Norbert Barankai for their valuable suggestions with which they aided this work. Financial support from the National Research, Development and Innovation Office of Hungary (under grant PD124079) is gratefully acknowledged.*

References

[1] S. Vesztergom, V. Grozovski, G. G. Láng, P. Broekmann, 6th Regional Symposium on Electrochemistry for South-East Europe, On the Electrolysis of Dilute Solutions of Strong Acids, Balatonkenese, Hungary, 2017, p. 103 (KN103).
[2] V. Grozovski, S. Vesztergom, G. G. Láng, P. Broekmann, *Journal of the Electrochemical Society* **164** (2017) 3171–3178.
[3] A. J. Bard, L. R. Faulkner, *Electrochemical Methods: Fundamentals and Applications*, John Wiley & Sons, New York, 2001, pp. 785–807.
[4] D. Britz, J. Strutwolf, *Digital Simulation in Electrochemistry*, 4th Edition, Springer, Berlin, 2016.
[5] F. G. Cottrell, Zeitschrift für physikalische Chemie **42** (1903) 385–431.
[6] E. Kätelhön, R.G. Compton, *Analyst* **140** (2015) 2592–2598.
[7] A. J. Bard, Gy. Inzelt, F. Scholz (Eds), *Electrochemical Dictionary*, Springer, Berlin, 2008, p. 163.
[8] G. W. Johnson, R. Jennings, *LabVIEW Graphical Programming*, 4th Edition, McGraw-Hill, New York, 2006.
[9] G. A. Baker, P. Graves-Morris, *Padé Approximants*, Cambridge University Press, Cambridge, 1996.
[10] K. A. Atkinson, *An Introduction to Numerical Analysis,* John Wiley & Sons, New York, 1989.
[11] J. Crank, P. Nicolson, Proceedings of the Cambridge Philosophical Society, **43** (1947) 50–67.
[12] S. Vesztergom, N. Barankai, N. Kovács, M. Ujvári, Th. Wandlowski, G.G. Láng, *Acta Chimica Slovenica* **61** (2014) 223–232.
[13] S. Vesztergom, N. Barankai, N. Kovács, M. Ujvári, H. Siegenthaler, P. Broekmann, G. G. Láng, *Journal of Solid State Electrochemistry* **20** (2016) 3165–3177.
[14] S. Vesztergom, N. Barankai, N. Kovács, M. Ujvári, H. Siegenthaler, P. Broekmann, G. G. Láng, *Electrochemistry Communications* **68** (2016) 54–58.