Additional Examples

2018-06-22

This vignette provides examples that demonstrates the capabilities of the functions in the eChem package. In order to keep the size of this file small, the `R` code is included but not evaluated, which means that no visualizations appear in this file. You may copy and paste code from this file or the accompanying `Additional_Examples.R` source file into the `R` console to run the code and view the results.

```
# load the eChem package so that its functions are available
library(eChem)
```

Replicating Figures From Gosser's Textbook

As explained in the vignette `Computational Details`, the four `simulate` functions in the eChem package use the explicit finite difference computational method outlined in Gosser, D. K. *Cyclic Voltammetry Simulation and Analysis of Reaction Mechanisms*, VCH, New York, 1993. Gathered here are snippets of `R` code that replicate selected figures from Gosser's textbook.

Example 1: This replicates Gosser's test conditions for an E-only mechanism; see box on page 36 and Figure 2-7 and Figure 2-8. Note: The experimental settings identified by Gosser give the formal potential as 0 V; however, given the starting and the switching potentials and the cyclic voltammograms in the figures, the formal potential must be –0.25 V, which is the value used here.

```
# simulate the cyclic voltammogram
example1 = simulateCV(e.start = 0, e.switch = -0.5, e.form = -0.25,
                      mechanism = "E", scan.rate = 1, area = 0.01,
                      temp = 298, conc.bulk = 1e-3, n = 1, d = 1e-5,
                      alpha = 0.5, ko = 1, kcf = 0, kcr = 0)

# show plot of the applied potential; note that the y-axis here
# runs in the opposite direction than that shown in Gosser's Figure
# 2-7 with more negative potentials shown on the bottom
plotPotential(example1)

# show plot of the cyclic voltammogram
plotCV(list(example1))

# adjust the plotting window to display six plots in a 2 by 3 array
old.par = par(mfrow = c(2, 3))

# plot the diffusion profiles; note that the scale on the x-axis
# extends to greater distances from the electrode than does those
# shown in Gosser's Figure 2-8
plotDiffusion(example1, t = 0.225)
plotDiffusion(example1, t = 0.275)
plotDiffusion(example1, t = 0.380)
plotDiffusion(example1, t = 0.725)
plotDiffusion(example1, t = 0.775)
plotDiffusion(example1, t = 0.880)

# reestablish the original plot window
```

```
par(old.par)

# the plotGrid function provides a similar view of the diffusion
# profiles, but includes a cyclic voltammogram that indicates the
# specific time and potential for each diffusion profile
plotGrid(example1)

# the animate function shows how the cyclic voltammogram and the
# diffusion profiles change as the potential is scanned from
# e.start to e.switch and back to e.start; note that the files
# created by this function are saved to the current working
# directory
animateCV(example1, out_type = "html", out_name = "example1")
```

Example 2: This code replicates Figure 2-11 in Gosser's text, which shows the effect of the heterogeneous electron-transfer rate constant (k^o) on the results of a cyclic voltammetry experiment where the initial potential is the default value of 0.0 V, the switching potential is –0.8 V, and the redox couple's standard state formal potential is –0.4 V; note, Gosser does not give a value of k^o for reversible conditions, which we take here as 1 cm/s.

```
# simulate the cyclic voltammograms
example2a = simulateCV(ko = 1, e.switch = -0.8, e.form = -0.4)
example2b = simulateCV(ko = 0.01, e.switch = -0.8, e.form = -0.4)
example2c = simulateCV(ko = 0.001, e.switch = -0.8, e.form = -0.4)
example2d = simulateCV(ko = 0.0001, e.switch = -0.8, e.form = -0.4)

# overlay the four cyclic voltammograms and add a legend
plotCV(list(example2a, example2b, example2c, example2d),
       legend_text = c("ko = 1 cm/s", "ko = 0.01 cm/s",
                       "ko = 0.001 cm/s", "ko = 0.0001 cm/s"))
```

Example 3: This code replicates Gosser's Figure 2-12, which shows the effect of the transfer coefficient (alpha) on a cyclic voltammogram.

```
# simulate the cyclic voltammograms
example3a = simulateCV(ko = 0.0001, alpha = 0.75, e.start = 0.4,
                       e.switch = -1.2, e.form = -0.4)
example3b = simulateCV(ko = 0.0001, alpha = 0.50, e.start = 0.4,
                       e.switch = -1.2, e.form = -0.4)
example3c = simulateCV(ko = 0.0001, alpha = 0.25, e.start = 0.4,
                       e.switch = -1.2, e.form = -0.4)

# overlay the three cyclic voltammograms and add a legend
plotCV(list(example3a, example3b, example3c),
       legend_text = c("alpha = 0.75", "alpha = 0.50", "alpha = 0.25"))
```

Example 4: This code replicates Gosser's Figure 2-14, which shows how the forward chemical rate constant, $k_{c,f}$, affects the cyclic voltammogram for anEC mechanism when the reverse chemical rate constant, $k_{c,r}$, is 0; note that for the last example it is necessary to increase the number of time units to satisfy one of the computation's boundary conditions.

```
# simulate the cyclic voltammograms
example4a = simulateCV(mechanism = "EC", ko = 1, kcf = 0, kcr = 0)
example4b = simulateCV(mechanism = "EC", ko = 1, kcf = 2.5, kcr = 0)
example4c = simulateCV(mechanism = "EC", ko = 1, kcf = 100, kcr = 0)
example4d = simulateCV(mechanism = "EC", ko = 1, kcf = 1000, kcr = 0,
```

```
                        t.units = 4000)

# overlay the cyclic voltammograms and add a legend
plotCV(list(example4a, example4b, example4c, example4d),
       legend_text = c("kcf = 0", "kcf = 2.5", "kcf = 100", "kcf = 1000"))
```

Example 5: This code replicates Gosser's Figure 2-26, which shows the effect of the forward chemical rate constant, $k_{c,f}$, on a double-step chronoamperometry experiment with an EC mechanism. Gosser does not provide values for some parameters, such as the heterogeneous electron-transfer rate constant, k^o, the electrode's area, A, and the diffusion coefficient, D, each of which is set here at its default level; the resulting chronoamperograms, therefore, are similar to, but not identical to those in Figure 2-26. Note that the y-axis in Gosser's figure is in ms and that the y-axis in Gosser's figure is in mA instead of in s and in μA.

```
# simulate the chronoamperograms
example5a = simulateCA(mechanism = "EC", pulses = "double",
                       t.1 = 0.001, t.2 = 0.003, t.end = 0.005,
                       kcf = 50)
example5b = simulateCA(mechanism = "EC", pulses = "double",
                       t.1 = 0.001, t.2 = 0.003, t.end = 0.005,
                       kcf = 100, t.units = 3000)
example5c = simulateCA(mechanism = "EC", pulses = "double",
                       t.1 = 0.001, t.2 = 0.003, t.end = 0.005,
                       kcf = 500, t.units = 5000)

# overlay the chronoamperograms and add a legend
plotCA(list(example5a, example5b, example5c), scale = 0.2,
       legend_text = c("kcf = 50", "kcf = 100", "kcf = 500"),
       legend_position = "topright")
```

Example 6: This code replicates Gosser's Figure 5-6, which shows the cyclic voltamogram for the reduction of methlycobalamin by an EC mechanism. Note that the number of time units is set to 11,000 to satisfy one of the boundary conditions for the `simulateCV` function.

```
# simulate the cyclic voltammogram
example6 = simulateCV(e.start = -1, e.switch = -1.65, e.form = -1.529,
                conc.bulk = 0.002, ko = 0.012, alpha = 0.78,
                kcf = 580, kcr = 0, mechanism = "EC", area = 0.019,
                scan.rate = 0.3, d = 1.7e-6, t.units = 11000)

# display the cyclic voltammogram
plotCV(filenames = list(example6))
```

Annotating Simulations With Characteristic Values

Each of the four electrochemistry experiments has an `annotate` function that displays the result of a single simulation as a plot of current (`simulateCV`, `simulateLSV`, and `simulateCA`) or charge (`simulateCC`) vs. potential (`simulateCV` and `simulateLSV`) or time (`simulateCA` and `simulateCC`). The plot is then annotated with one or more characteristic values used for quantitative purposes or used to determine electrochemical and chemical reversibility.

Example 7: For a cyclic voltammetry experiment the characteristic values are the cathodic peak potential, $E_{p,c}$, the anodic peak potential, $E_{p,a}$, the difference between the peak potentials, ΔE, the average peak potential, E_{avg}, the cathodic peak current, $i_{p,c}$, the anodic peak current, $i_{p,a}$, and the peak current ratio, $|i_{p,c}|/|i_{p,a}|$. When a value cannot be calculated, then it is reported as "not measurable".

```
# simlulate the cyclic voltammograms; for the second example, the
# return peak is sufficiently distorted such that its peak
# current cannot be determined
example7a = simulateCV(ko = 1, e.switch = -0.8, e.form = -0.4)
example7b = simulateCV(ko = 0.0001, e.switch = -0.8, e.form = -0.4)

# plot the annotated cyclic voltammograms
annotateCV(example7a)
annotateCV(example7b)
```

Example8: For a linear sweep voltammetry experiment the characteristic values are the peak potential, E_p and the peak current, i_p. In the absence of stirring both values are included; when stirring is set to `slow`, `medium`, or `fast`, only the peak, or limiting current is returned.

```
# simlulate the linear sweep voltammograms; for the first example
# the solution is not stirred, but for the second example the
# stir rate is set to slow
example8a = simulateLSV(ko = 1, e.start = 0, e.end = -0.8,
                        e.form = -0.4, stir.rate = "off")
example8b = simulateLSV(ko = 1, e.start = 0, e.end = -0.8,
                        e.form = -0.4, stir.rate = "slow")

# plot the annotated linear sweep voltammograms
annotateLSV(example8a)
annotateLSV(example8b)

# show the applied potential and diffusion profiles
plotPotential(example8a)
plotGrid(example8a)
plotGrid(example8b)
```

Example 9: For a chronoamperometry experiment the characteristic value is the current, i, at a specified time after the application of each pulse. For a single pulse experiment there is one characteristic current; for a double pulse experiment there are two characteristic currents and a current ratio.

```
# simulate the chronoamperograms; the first example is for a
# single pulse experiment and the second example is for a double
# pulse experiment
example9a = simulateCA(pulses = "single", t.1 = 1, t.end = 3)
example9b = simulateCA(pulses = "double", t.1 = 1, t.2 = 2, t.end = 3)

# annotate the chronoamperograms; the scale.factor is set to 10
# so that it is easier to see how the currents are defined
annotateCA(example9a, time.delay = 0.25, scale.factor = 10)
annotateCA(example9b, time.delay = 0.25, scale.factor = 10)

# show the applied potential and diffusion profiles
plotPotential(example9a)
plotPotential(example9b)
plotGrid(example9a)
plotGrid(example9b)
```

Example 10: For a chronocoulometry experiment the characteristic value is the charge, Q, at a specified time after the application of each pulse. For a single pulse experiment there is one characteristic charge; for a double pulse experiment there are two characteristic charges and a charge ratio.

```
# simulate the chronocoulograms by wrapping simulateCC around a
# call to simulateCA; the first example is for a single pulse
# experiment and the second example is for a double pulse
# experiment
example10a = simulateCC(simulateCA(pulses = "single",
                                   t.1 = 1, t.end = 3))
example10b = simulateCC(simulateCA(pulses = "double",
                                   t.1 = 1, t.2 = 2, t.end = 3))

# annotate the chronoamperograms
annotateCC(example10a, time.delay = 0.25)
annotateCC(example10b, time.delay = 0.25)

# show the applied potential and diffusion profiles
plotPotential(example10a)
plotPotential(example10b)
plotGrid(example10a)
plotGrid(example10b)
```

Exploring the Properties of Electrochemical Experiments

Each of the `simulate` functions returns a lengthy list of data that is accessible using the syntax `object_name$data_name` where `object_name` identifies the object that contains the results of the simulation and `data_name` identifies the specific piece of data.

Example 11: This example shows how to verify the square root dependence of current on scan rate for a cyclic voltammetry experiment involving a redox system with an E-only mechanism. Note that the lines of code following the call to `simulateCV` demonstrates how to extract additional information from the objects that contains the result of a simulation, how to plot this data, and how to fit it to a linear model.

```
# simulate the cyclic voltammograms for seven different scan rates
example_cv11a = simulateCV(scan.rate = 0.01)
example_cv11b = simulateCV(scan.rate = 0.05)
example_cv11c = simulateCV(scan.rate = 0.1)
example_cv11d = simulateCV(scan.rate = 0.5)
example_cv11e = simulateCV(scan.rate = 1)
example_cv11f = simulateCV(scan.rate = 5)
example_cv11g = simulateCV(scan.rate = 10)

# examine an overlay of four simulations to see affect of scan rate
plotCV(filenames = list(example_cv11a, example_cv11c,
                        example_cv11e, example_cv11g),
       legend_text = c("scan rate = 0.01 V/s", "scan rate = 0.1 V/s",
                       "scan rate = 1 V/s", "scan rate = 10 V/s"))

# create vector of scan rates
scanrates = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)

# use the max function to find the peak current for each simluation
peakcurrents = c(max(example_cv11a$current), max(example_cv11b$current),
                 max(example_cv11c$current), max(example_cv11d$current),
                 max(example_cv11e$current), max(example_cv11f$current),
                 max(example_cv11g$current))
```

```
# plot the peak current vs. the square root of the scan rate
plot(x = sqrt(scanrates), y = peakcurrents,
     pch = 19, col = "blue",
     xlab = "(scan rate)^0.5",
     ylab = expression("peak current (", mu, "A)"))

# use the linear model function to complete a regression analysis
iv.lm = lm(peakcurrents ~ sqrt(scanrates))

# use the abline function to add the regression line to plot
abline(iv.lm, col = "blue", lty = 1)
```

Example 12: This example explores the relationship between the forward, k_f, and the reverse, k_r, heterogeneous electron-transfer rate constant as a function of the applied potential, and how that relationship is affected by the transfer coefficient, α.

```
# simulate cyclic voltammograms setting ko to 0.05 for all and
# setting alpha to 0.4, 0.5, and 0.6
example12a = simulateCV(ko = 0.05, alpha = 0.3)
example12b = simulateCV(ko = 0.05, alpha = 0.5)
example12c = simulateCV(ko = 0.05, alpha = 0.7)

# overlay the cyclic voltammograms and add a legend; note how the
# value of alpha affects the peak currents on the forward and the
# reverse scan
plotCV(filenames = list(example12a, example12b, example12c),
       legend_text = c("alpha = 0.4", "alpha = 0.5", "alpha = 0.6"))

# examine how the
par(mfrow = c(3, 1))
plot(example12a$potential, example12a$k_f, type = "l",
     xlab = "potential (V)", ylab = "rate constant (cm/s)",
     main = "alpha = 0.3")
lines(example12a$potential, example12a$k_b, lty = 2)
plot(example12b$potential, example12b$k_f, type = "l",
     xlab = "potential (V)", ylab = "rate constant (cm/s)",
     main = "alpha = 0.5")
lines(example12b$potential, example12b$k_b, lty = 2)
plot(example12c$potential, example12c$k_f, type = "l",
     xlab = "potential (V)", ylab = "rate constant (cm/s)",
     main = "alpha = 0.7")
lines(example12c$potential, example12c$k_b, lty = 2)
par(mfrow = c(1,1))
```

Modeling Experimental Data

One way to obtain a feel for how an electrochemical method's experimental variables affect the outcome of that an experiment is to try and model real or synthetic data using the appropriate `simulate` function. The two examples below demonstrate how to create reduced files for this purpose using raw data collected in the lab or using data generated using one of the `simulate` functions.

Example 13: This example uses raw data generated in lab for the cyclic voltammetry of ferrocene. The

example assumes that the original data is cleaned up and exported as a .csv file that consists of two columns, one of which has a header of `potential` and the other of which has a header of `current`. The ferrocene_data.csv file is included with the eChem package.

```
# create a data.frame of potentials and currents using the
# read.csv function; note that the system.file command wrapped
# inside the call to teh read.csv function is used to determine
# the path to the ferrocene_data.csv file
example13 = read.csv(system.file("extdata", "ferrocene_data.csv",
                                 package = "eChem"))

# examine the file's structure, noting that it consists of two
# variables, each with 1600 values, and that the potentials are
# in V and the currents are in A
str(example13)
head(example13)

# create a subsample of the full data by retaining every 20th
# value and adjust the current so it is reported in μA instead of
# in A (there is no need to adjust the units for the potential);
# the smaller data set will allow us to see individual points
# when using plotCV to examine the data
example13_potential = example13$potential[seq(1,1600,20)]
example13_current = example13$current[seq(1,1600,20)]*1e6

# create a reduced data file (as a list) with the structure
# required by plotCV and then examine the plot
example13_expt = list("expt" = "CV", "file_type" = "reduced",
                      "potential" = example13_potential,
                      "current" = example13_current)
plotCV(filename = list(example13_expt))

# use simulateCV to test a set of conditions and then overlay the
# simulated data and the experimental data; as first guess we use
# the initial potential of 0 V, the switching potential of 0.8 V,
# approximate the formal potential as 0.4 V, and leave all other
# values at their default levels
example13_sim = simulateCV(e.start = 0, e.switch = 0.8, e.form = 0.4)

# overlay the experimental data and the simulated data and
# evaluate the quality of the fit
plotCV(filename = list(example13_sim, example13_expt),
       legend_text = c("simulated", "experimental"))

# continue to adjust variables until satisified with the fit
example13_sim = simulateCV(e.start = 0, e.switch = 0.8, e.form = 0.39,
                           scan.rate = 0.1, conc.bulk = 0.01,
                           ko = 0.009, alpha = 0.3)
plotCV(filename = list(example13_sim, example13_expt),
       legend_text = c("simulated", "experimental"))
```

Example 14: There are many reasons why it may not be possible to obtain a good fit between experimental data and simulated data. In this example we use the simulateCV function to generate a cyclic voltammogram and then use the sampleCV function to create an "experimental" data set that others can attempt to duplicate.

```
# create the experimental cyclic voltammogram with a small amount
# of noise added
example14_raw = simulateCV(e.start = 0, e.switch = 0.8, e.form = 0.4,
                           conc.bulk = 0.005, area = 0.05, d = 5e-5,
                           ko = 0.75, sd.noise = 0.5)
example14_expt = sampleCV(example14_raw, data.reduction = 2.5)
plotCV(filenames = list(example14_expt))

# use simulateCV to test a set of conditions and then overlay the
# simulated data and the experimental data; as first guess we use
# the initial potential of 0 V, the switching potential of 0.8 V,
# approximate the formal potential as 0.4 V, and leave all other
# values at their default levels
example14_sim = simulateCV(e.start = 0, e.switch = 0.8, e.form = 0.4)

# overlay the experimental data and the simulated data and
# evaluate the quality of the fit
plotCV(filename = list(example14_sim, example14_expt),
       legend_text = c("simulated", "experimental"))

# continue to adjust variables until satisified with the fit
example14_sim = simulateCV(e.start = 0, e.switch = 0.8, e.form = 0.4,
                     conc.bulk = 0.005, area = 0.05, d = 5e-5,
                     ko = 0.75)
plotCV(filename = list(example14_sim, example14_expt),
       legend_text = c("simulated", "experimental"))
```

Effect of Stirring in Linear Sweep Voltammetry

Stirring the solution during a linear sweep voltammetry experiment results in a diffusion layer at a fixed distance from the electrode's surface instead of a diffusion layer that grows with time. This is modeled in the `simulateLSV` function by using a scaling factor to limit the thickness of the diffusion layer and, therefore, which elements in the diffusion grid are calculated using the method outlined in the vignette on `Computation Details` and which are set to the boundary condition given by their respectivce bulk concentrations.

Example 15: Although there is no rigorous relationship here between the stir rate and the scaling factor, as is clear from this example, the qualitiative affect of stirring is modeled.

```
# simulate the linear sweep voltammograms
example15a = simulateLSV(stir.rate = "off")
example15b = simulateLSV(stir.rate = "slow")
example15c = simulateLSV(stir.rate = "medium")
example15d = simulateLSV(stir.rate = "fast")

# overlay the linear sweep voltammograms
plotLSV(filenames = list(example15a, example15b,
                         example15c, example15d),
        legend_text = c("stir: off", "stir: slow",
                        "stir: medium", "stir: fast"),
        legend_position = "topleft",
        main_title = "Effect of Stir Rate on Current in LSV")
```