This is a PDF printout of Code-O235.mw found in 9-Code.zip.
Permalink: http://www.nanopolitan.com/Files/9/9-Code.pdf

This Maple worksheet contains the code used for parameter estimation in the paper "V.A. Sethuraman, S. Khan, J.S. Jur, A.T. Haug, J.W. Weidner, "Measuring Oxygen, Carbon Monoxide and Hydrogen Sulfide Diffusion Coefficient and Solubility in Nafion Membranes", Electrochimica Acta, Volume 54, Issue 27, 30 November 2009, Pages 6850-6860. DOI: 10.1016/j.electacta.2009.06.068. Permalink: https://doi.org/10.1016/j.electacta.2009.06.068".

A PDF copy of the above paper is available for academic use at http://www.nanopolitan.com/Files/Sethuraman_EA_54_6850_2009.pdf.

The code estimates the diffusion coefficient and solubility of a gas through a membrane from data obtained in a chrono-amperometry experiment from a Devanathan–Stachurski type diffusion cell [1]. The construction of the cell and the electrochemical-monitoring experiment are explained in detail in the above paper. This code was originally written in Maple 8 in 2002 and later updated in Maple 2018.2 in 2020 (both at the University of South Carolina campus in Columbia, South Carolina). This is essentially a parameter estimation problem using the least-squares method. If you have any questions or comments, you are welcome to contact me at vj@berkeley.edu or at vijay.sethuraman@gmail.com.

```
> restart:
> with(linalg):
> with(plots):
  ### WARNING: persistent store makes one-argument readlib obsolete
  readlib(readdata):
```

N is the number of data points and confidence represents the necessary confidence interval.

Inputs/Constants:
You can change the area (across which the gas diffuses) and the membrane thickness.
Area = (Pi * radius^2) if circular or Length x Width, if rectangular, in cm^2, . If it is an irregular shape, you may enter the actual measured area of the membrane across which the gas diffuses.
Membrane thickness, L in cm. Note: Since the membrane used here is Nafion 112, and it is 2 mils (1 mil = 1/1000th of an inch) thick, the formula for L converts 2 mils to cm.
'ne' refers to the number of electrons transfered in the reaction. For oxygen reduction to water (half-reaction corresponding to equation #8 in the above paper), it is 2.
F refers to the Faraday's constant in C/mole equivalent.
'confidence' refers to the confidence interval of 95% for the parameter estimation problem.

```
> ne:=2:
  F:=96485:
  Area:=10.0:
  L:=2*25.4/1000000*100:
  confidence:=0.95:
```

Data Input:
Specify the text file from which the data is contained in the line below. Note that it must be a text file. The two float commands specify that each column of data is not an integer, but rather a decimal. If you want to use the exact dataset used in this simulation, it is part of the ZIP file (9-Code.zip). The data contained in this text file correspond to Time (second) and Current (Ampere) from the electrochemical-monitoring experiment for oxygen diffusing across a Nafion 112 membrane at 308K and ambient pressure (101253 Pa at 89 m above sea level in Columbia, South Carolina).

```
> simdata:=readdata("E:\\Web Pages\\Nanopolitan -
  Basic\\Files\\9\\O235C.txt",[float,float]):
  # print(simdata):
> N:=nops(simdata):
```

This section reduces the number of data points. The program would take too long to run (without any improvements in confidence or parameter-estimation efficiency) if it had to analyze 1000+ points. So, if you import more than 500, it reduces that number to 1/5. This is customizable - just change the denominator in 'trunc(N/5)'.

```
> if N > 500 then N:=trunc(N/5)+100-19:
  tempdata:=matrix(N,2):
  for j from 1 to 2 do
  for i from 1 to 100 do
  tempdata[i,j]:=simdata[i,j]:
  od;od;
  for j from 1 to 2 do
  for i from 1 to N-100 do
  tempdata[i+100,j]:=simdata[(100+(i-1)*5),j]:
  od:od:
  for i from 1 to N do
  tempdata2[i,1]:=tempdata[i,1]:
  tempdata2[i,2]:=tempdata[i,2]/1000:
  od:
  simdata:=tempdata2:fi:
> #print(tempdata, tempdata2):
```

These are the matrices being used:
C[1,1] is the diffusion coefficient.
C[1,2] is the solubility.

```
> C:=matrix(2,1):
  Z:=matrix(N,2):
  Dm:=matrix(N,1):
> tau:=C[1,1]*t/L^2:
  ii[inf]:=ne*F*Area*C[1,1]*C[2,1]/L:
```

This is equation 25 presented in Fan and White's paper [2] for finding the solubility in a membrane. Three terms provide sufficient accuracy, however more can be used. C[1,1] is the diffusion coefficient (cm^2/s) and C[1,2] is the solubility (mol/cm^3).

```
> eqn1:=ii[inf]*(1+2*sum((-1)^k*exp(-1*k^2*Pi^2*tau), k=1..12)):
```

The least squares method is used to solve for the values of C[1,1] and C[1,2]. The process used was taken from Chapra and Canale for a non-linear least squares regression p. 358-9 [3]. After an initial guess for the unknowns, matrix manipulation is performed on the data to determine new estimates for the unknowns. Successive iterations of this technique are performed until the changes to the values for the unknowns (solubility and diffusion coefficient) becomes small. Usually, less than 10 iterations are more than sufficient.

```
> deq[1]:=diff(eqn1,C[1,1]):
> deq[2]:=diff(eqn1,C[2,1]):
```

Initial Guesses:

These are the initial guesses for the diffusion coefficient, C[1,1], and solubility, C[2,1]. Units are cm^2/s, and mol/cm^3, respectively.

```
> C[1,1]:=.15e-6: C[2,1]:=80e-5:
```

This is the rest of the least squares procedure. The new values for C[1,1] and C[1,2] are shown below.

```
> for i from 1 to 2 do
  for j from 1 to N do
  t:=simdata[j,1]:
  Z[j,i]:=evalf(deq[i]):
  od;od:
  ZT:=evalm(transpose(Z)):
  Zprod:=evalm(inverse(ZT&*Z)):
  for i from 1 to N do
  t:=simdata[i,1]:
  Dm[i,1]:=evalf(simdata[i,2]-eqn1): od:
  dC:=evalm(Zprod &* ZT &* Dm):
  C:=evalm(C + dC):
```

Here we compare data values using the formula from Fan and White [2] with the new values for C and the simulated data values.

```
> comparem:=matrix(N,3):
  for j from 1 to N do
  t:=simdata[j,1]:
  comparem[j,1]:=t:
  comparem[j,2]:=simdata[j,2]:
  comparem[j,3]:=evalf(eqn1):
  od:
```

Here are three more iterations of the least squares method.

```
> m:=1:dC[1,1]:
  while m < 25 and abs(dC[1,1]) > 5e-11 do
  for i from 1 to 2 do
  for j from 1 to N do
  t:=simdata[j,1]:
  Z[j,i]:=evalf(deq[i]):
  od:od:
  ZT:=evalm(transpose(Z)):
  Zprod:=evalm(inverse(ZT&*Z)):
  for i from 1 to N do
```

```
  t:=simdata[i,1]:
  Dm[i,1]:=evalf(simdata[i,2]-eqn1): od:
  dC:=(evalm(Zprod &* ZT &* Dm)):
  C[1,1]:=abs(C[1,1] + dC[1,1]):
  C[2,1]:=Re(C[2,1] + dC[2,1]):
  #print(dC,C): m:=m+1:
  od:
```

The final values of the diffusion coefficient and solubility are shown. Then below is shown in 4 columns: time (s), current data, current using eq. 25 of White and Fan [4], and the difference between the two currents.

```
> #print(C): #C[1,1]:=.497e-6: C[2,1]:=31.1e-6:
  comparem:=matrix(N,4):
  for j from 1 to N do
  t:=simdata[j,1]:
  comparem[j,1]:=t:
  comparem[j,2]:=simdata[j,2]:
  comparem[j,3]:=evalf(eqn1):
  comparem[j,4]:=evalf(eqn1-simdata[j,2]):
  od:
```

This section of the program calculates the confidence interval for the data values. The formula for this is shown as equation 14 in the Kimble and White paper [5]. In order to determine the confidence interval, it is necessary to solve for a t-distribution at 95% confidence, the variance and an approximate Hessian matrix (see eq. 18).

```
> f:=matrix(N,1):
  for i from 1 to N do
  t:=simdata[i,1];
  H[1,i]:=2*evalf(deq[1])*evalf(deq[1]);
  H[2,i]:=2*evalf(deq[1])*evalf(deq[2]);
  H[3,i]:=2*evalf(deq[2])*evalf(deq[1]);
  H[4,i]:=2*evalf(deq[2])*evalf(deq[2]);
  f[i,1]:=evalf(simdata[i,2]-eqn1);
  od:
  #print(comparem,f);
> writedata("E:\\Web Pages\\Nanopolitan -
  Basic\\Files\\9\\O235Cout.txt",comparem);
  writedata("E:\\Web Pages\\Nanopolitan -
  Basic\\Files\\9\\O235CDCo.txt",C);
> Hessian:=matrix(2,2):
  Hessian[1,1]:=sum('H[1,j]', 'j'=1..N):
  Hessian[1,2]:=sum('H[2,j]', 'j'=1..N):
```

```
  Hessian[2,1]:=sum('H[3,j]', 'j'=1..N):
  Hessian[2,2]:=sum('H[4,j]', 'j'=1..N):
> inv_Hessian:=inverse(Hessian):
> Ckk[1]:=inv_Hessian[1,1]: Ckk[2]:=inv_Hessian[2,2]:
> s2:=sum('(f[j,1])^2', 'j'=1..N)/(N-2):
  s:=sqrt(s2):
```

This is a matrix showing t-distribution values for various confidences (taken from Ref. 6) and is used in determining the confidence intervals.

```
Tdistmat:=matrix(9,6,[0,40,50,100,200,400,0.5,0,0,0,0,0,0.6,0.26,
0.26,0.25,0.25,0.25,0.7,0.53,0.53,0.53,0.53,0.52,0.8,0.85,0.85,0.85,
0.84,0.84,0.9,1.3,1.3,1.29,1.29,1.28,0.95,1.68,1.68,1.66,1.65,1.65,
0.99,2.42,2.4,2.37,2.35,2.33,0.999,3.31,3.26,3.17,3.13,3.09]):
> Tdist:=1.65:
  # for i from 1 to 6 do
  # if N = Tdistmat[1,i] then temp1:=i; fi;od;
  # for i from 1 to 9 do
  # if confidence = Tdistmat[i,1] then temp2:=i;fi;od;
  # Tdist:=Tdistmat[temp2,temp1]:
```

Here are the resulting confidence intervals for the diffusion coefficient and solubility.

```
> Conf_interval[Diff]:=Tdist*s*sqrt(Ckk[1]);
  Conf_interval[Sol]:=Tdist*s*sqrt(Ckk[2]);
```

$$Conf\_interval_{Diff} := 3.997878426\ 10^{-9}$$

$$Conf\_interval_{Sol} := 1.870937945\ 10^{-6} \quad \textbf{(1)}$$

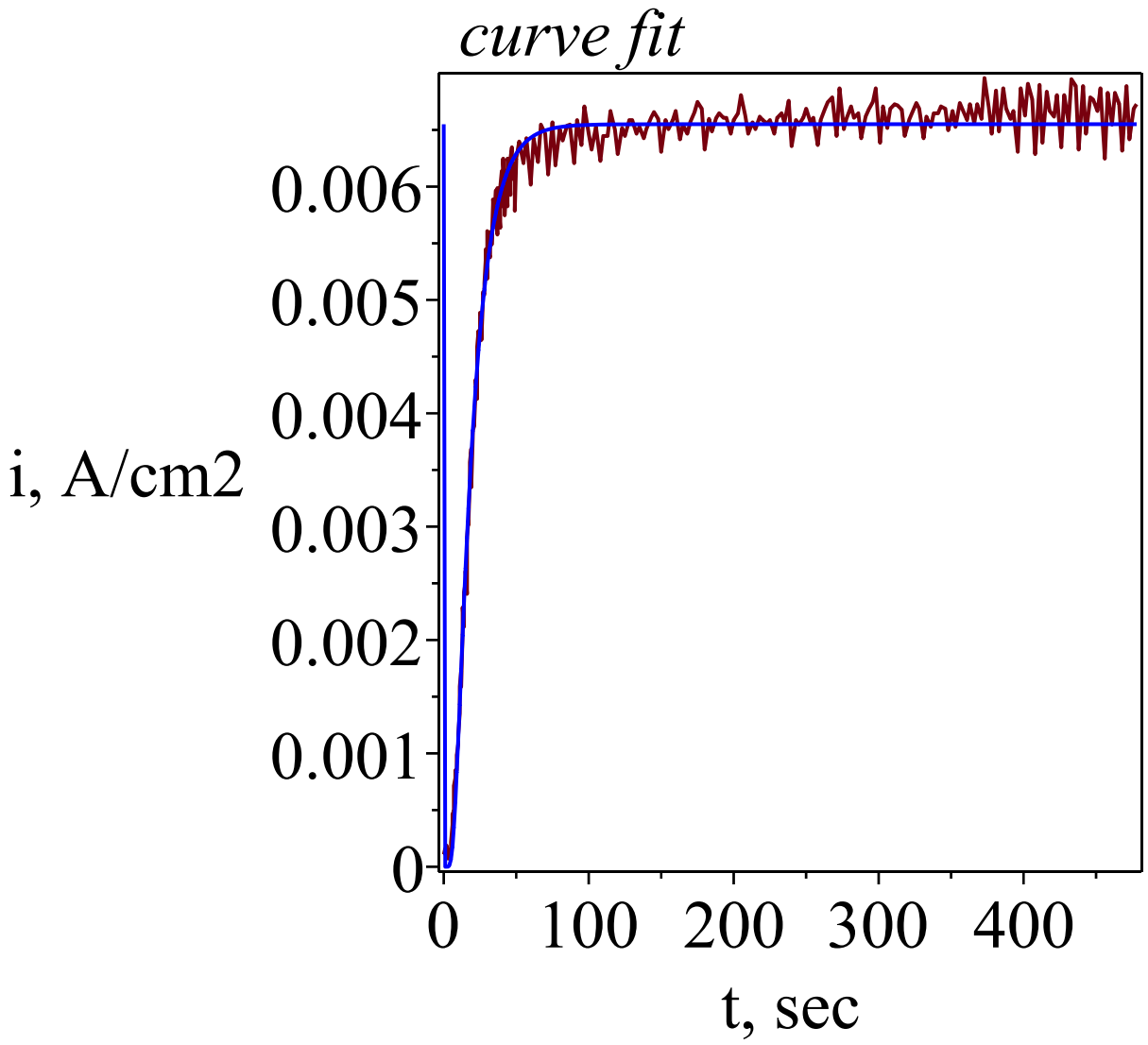
Here are the diffusion-coefficient and the solubility values, in cm^2/s and mol/cm^3, respectively.

```
> print(C);
```

$$\begin{bmatrix} 2.049658552\ 10^{-7} \\ 0.00008412738994 \end{bmatrix} \quad \textbf{(2)}$$

Note that the rise-time depends on the diffusion coefficient, and the steady-state current depends on both the diffusion-coefficient and the solubility.

```
> graph1:=plot([seq([simdata[j,1],simdata[j,2]],j=1..N)]):graph2:=
  plot([seq([simdata[j,1],comparem[j,3]],j=1..N)], color=blue):
> display(graph1, graph2,axes=boxed,labels=["t, sec","i, A/cm2"],
  title=`curve fit`);
```



Acknowledgements:

The following colleagues are gratefully acknowledged for their pedagogical role in developing this code:
Professor Ralph E. White (at The University of South Carolina in Columbia, SC),
Professor Venkat R. Subramanian (at The University of Texas in Austin, TX), and
Dr. Andrew T. Haug (at 3M in St. Paul, MN).

References:

1. M. Devanathan, Z. Stachurski, Proc. R. Soc. Edinb. Ser. A 270 (1962) 90.
2. D. Fan, R.E. White, N. Gruberger, J. Appl. Electrochem. 22 (1992) 770.
3. Steven C. Chapra, Raymond P. Canale. Numerical Methods for Engineers. Boston, MA: McGraw-Hill Higher Education (2006). ISBN-13: 978-0073397924. ISBN-10: 007339792X.
4. D. Fan, R.E. White, N. Gruberger, J. Appl. Electrochem. 22 (1992) 770.
5. M.C. Kimble, R.E. White, Y.M. Tsou, R.N. Beaver, J. Electrochem. Soc. 137 (1990) 2510.
6. E. Kreyszig, H. Kreyszig, E.J. Norminton, Advanced Engineering Mathematics. Hoboken, NJ: Wiley (2011). ISBN: 0470458364.

© Vijay Anand Sethuraman. All rights reserved. Last updated: August 26, 2020.
541 Main St., Lab 126, Columbia, South Carolina 29208, United States of America.