# Conductivity of electrolyte solutions essay

Experiment 4: Conductivity of electrolyte solutions (Dated: October 29, 2009) I. INTRODUCTION Pure water does not conduct electricity, but any solvated ionic species would contribute to conduction of electricity. An ionically conducting solution is called an electrolyte solution and the compound, which produces the ions as it dissolves, is called an electrolyte. A strong electrolyte is a compound that will completely dissociate into ions in water. Correspondingly, a weak electrolyte dissolves only partially. The conductivity of an electrolyte solution depends on concentration of the ionic species and behaves diffrently for strong and weak electrolytes. In this work the electric conductivity of water containing various electrolytes will be studied. The data will be extrapolated to infinitely dilute solutions and the acidity constant for a given weak electrolyte will also be determined. Additional theoretical background for electrolyte solutions can be found from Refs. [1–3]. II. THEORY Movement of ions in water can be studied by installing a pair of electrodes into the liquid and by introducing a potential difference between the electrodes.

Like metallic conducting materials, electrolyte solutions follow Ohm’s law: R= U I (1) where R is the resistance (? , “ ohms”), U is the potential difference (V, “ Volts”), and I is the current (A, “ Amperes”). Conductance G (S, Siemens or ?? 1 ) is then defined as reciprocal of the resistance: G= 1 R (2) Conductance of a given liquid sample decreases when the distance between the electrodes increases and increases when the effective area of the electrodes increases. This is shown in the following relation: G=? A l (3) where ? is the conductivity (S m? 1 ), A is the cross-sectional area of the lectrodes (m2 ; e. g. the effective area available for conducting electrons through the liquid), and l is the distance between the electrodes (m). Molar conductivity ? m (S m2 mol? 1 ) is de? ned as: ? m = ? c (4) where c is the molar concentration of the added electrolyte. A " typical" value for molar conductivity is 10 mS m2 mol? 1 . The molar conductivity of an electrolyte would be independent of concentration if ? were proportional to the concentration of the electrolyte. In practice, however, the molar conductivity is found to vary with the concentration (see Fig. ). One reason for this variation is that the number of ions in the solution might not be proportional to the concentration of the electrolyte. For example, the concentration of ions in a solution of a weak acid depends on the concentration of the acid in a complicated way, and doubling the concentration of the acid does not double the number of ions. Another issue is that ions interact with each other and tend to slow down each other leading reduced conductivity. In this limit, the molar conductivity depends on square root of electrolyte molar concentration.

In the 19th century Friedrich Kohlrausch discovered the following empirical relation between the molar concentration of a strong electrolyte and the molar conductivity (Kohlrausch's law) at low concentrations: v ? m = ? 0 ? K c m Typeset by REVTEX (5) FIG. 1: Variation of molar conductivity as a function of molar concentration. a) Strong electrolute and b) weak electrolyte. where K is a non-negative constant depending on the electrolyte and ? 0 is the limiting molar conductivity (e. g. the m molar conductivity in the limit of zero concentration of the electrolyte).

Furthermore, Kohlrausch was able to show that ? 0 can be expressed as a sum of contributions from its individual ions. If the limiting molar conductivity for m the cations is ? + and for the anions ?? , the “ law of the independent migration of ions” states: 0 ? m = v+ ? + + v? ? ? (6) where v+ is the number of cations per formula unit, v is the corresponding number of anions, and ? + and ?? are the limiting molar conductivities for cations and anions, respectively. For example, for HCl v+ = 1 and v? = 1 but for MgCl2 we have v+ = 1 and v = 2.

Because weak electrolytes are not fully ionized in solution, the number of ions is not proportional to the concentration of the electrolyte but depends on the degree of dissociation (? ). The e? ective molar conductivity can then be approximated in terms of ? and the hypothetical molar conductivity of the fully ionized case (? 0 ): m ? m = ?? 0 m When a weak acid dissociates in water solution, we have: HA(aq) + H2 O(l) ? H3 O+ (aq) + A? (aq) (8) (7) The e? ective concentrations in solution are then given by (subscript 0 refers to the initial concentration of the acid): H3 O+ = ? HA]0 , A? = ? [HA]0 , [HA] = (1 ? ?) [HA]0 The acidity constant (Ka ) can now be written in terms of the ion activities (a): a H3 O+ a A? a H3 O+ a A? = a (HA) a (H2 O) a (HA) = 1 (solvent) (9) Ka = (10) In order to proceed, we write activity in an alternative form for each species (here i = H3 O+ , A? , HA, H2 O): bi b? a(i) = ? i (11) 2 where ? i is the activity coe? cient (dimensionless) for species i, bi is the molality for i (mol kg? 1 ) and b? is the ideal solution molality (constant, 1 mol kg? 1 ). Inserting Eq. (11) into Eq. (10) we get: ? H3 O+ ? A? ? H O b ? ? 3 A = K? ? Kb ?

HA bHA b? Ka = (12) where notation of K? and Kb are used for convenience. Do not confuse Kb with the acidity constant! In dilute solutions v the mean activity coe? cient (? ave = ? H3 O+ ? A? ; geometric mean value) can be calculated using the Debye-Hckel limiting law: v log (? ave ) = ? | zH3 O+ zA? | A I (13) where z’s are the ionic charges, A is a constant (typically 0. 509 for an aqueous solution at 25 o C) and I is the dimensionless ionic strength of the solution given by: 1 2 Nions 2 zi i= 1 I= bi b? (14) where Nions is the number of di? erence ionic species in the solution.

Note that log here denotes a logarithm with base 10 (ln would denote the natural base logarithm). Since the activity coe? cient for the neutral species (? HA ) is 2 equal to one and ? ave = ? H3 O+ ? A? , the Eq. (12) gives: 2 Ka = ? ave ? Kb (15) or by using logarithms: log (Ka ) = 2 log (? ave ) + log (Kb ) In dilute solutions molalities are directly related to concentrations by: [i] = ci = bi ? (17) (16) where ci is the molar concentration of species i (mol L? 1 ; usually denoted by species in brackets) and ? is the density of the solution (kg L? 1 ). Inserting Eq. (17) into Eq. (16) we have: H3 O+ A? ? b? HA] log (Ka ) = 2 log (? ave ) + log (18) Note that the numerical value of ? b? is approximately one, so it is only required for getting correct units. In many cases equilibrium constants are written in terms of concentrations, which tends to lead confusion in units. For example, for Eq. (8) we would normally write in terms of concentration: H3 O+ A? =: Kc [HA] Ka ? (19) while this gives the correct magnitude, it gives wrong units as the equilibrium constant is dimensionless (see the de? nition in Eq. (10)). Variable Kc was introduced to refer to the equilibrium constant obtained directly from concentrations.

Thus we can simplify Eq. (18) as: Kc ? b? log (Ka ) = 2 log (? ave ) + log (20) 3 From Eqs. (9) and (19) Kc can be obtained as: ? 2 [HA]0 1?? Kc = (21) Next we apply the Debye-H? ckel limiting law (Eqs. (13) and (14)) and Eq. (21) to Eq. (20): u v log (Ka ) = ? 2 | zH3 O+ zA? | A I + log OR pKa = ? log (Ka ) =

2A ? ? [HA]0 ? log ? b? ? 2 [HA]0 ? b? (1 ? ?) ? ? [HA]0 + log ? b? ? 2 [HA]0 ? b? (1 ? ?) Kc ? b? = ? 2A (22) In practice ? can be obtained from Eq. (7). However, before Eq. (7) can be applied we must know the limiting molar conductivities for solution consisting of H3 O+ and A? In order to calculate the limiting conductivity mentioned above, we must use Eq. (6). Note that we cannot use Eq. (5) since it only applies to strong electrolytes. For the present case Eq. (6) reads (HA = CH3 COOH): ? 0 (CH3 COOH) = ? + (H3 O+ ) + ?? (CH3 COOH? ) m (23) Such data is not directly available, but can be calculated, for example, by measuring the limiting molar conductivities of the following strong electrolyte (water) solutions: HCl > H3 O+ + Cl? CH3 COONa > CH3 COO + ? (? 0 = ? + (H3 O+ ) + ?? (Cl? )) 1, m (? 0 2, m Na+ (? 0 3, m = ? + (Na ) + ?? (Cl )) ? + (24) NaCl > Na + Cl + = ? + (Na+ ) + ?? (CH3 COO? )) If the three limiting conductivities can be measured then the limiting conductivity for Eq. (23) is given by: 0 ? m (CH3 COOH) = ? 0 ? ? 0 + ? 0 1, m 2, m 3, m (25) Thus Eq. (7) now reads: “ to be measured” ?= ? m (CH3 COOH) ? m (CH3 COOH) = 0 0 (CH COOH) ? m ? 3, m ? ? 0 + ? 0 3 2, m 1, m “ to be determined separately” (26) III. EXPERIMENTAL Task overview: Measure conductivities of 0. 03 M, 0. 02 M, 0. 015 M, 0. 01M and 0. 005 M solutions of three strong electrolytes (NaCl, HCl, CH3 CO2 Na) and one weak electrolyte (acetic acid; CH3 CO2 H) using a conductivity meter.

Calibration solution: If calibration solution (0. 01 KCl in deionized water) is not available, it can be prepared by weighing 0. 3728 g of anhydrous KCl and dissolving it in 500. 0 mL of deionized water. Electrolyte solutions: Prepare 200 mL electrolyte solutions in volumetric ? asks. Use the electrolytes and concentrations given in the task overview above. CH3 COOH and HCl are given as 0. 1 M stock solutions whereas the rest of the compounds are given as solids. Be sure to mix the solutions well before measurement. 4 Measurements: Use a beaker, which allows full immersion of the electrodes in the solution.

Handle the electrode with care and be sure to connect the electrode cable to the meter correctly – otherwise you may damage the instrument! Always rinse the electrode with deionized water before use. Use a magnetic stirrer to ensure that the solution is homogenous during the measurement. Before taking readings, always shake the electrode brie? y to release possible air bubbles trapped in the electrode. First, perform calibration measurement using the 0. 01 M KClsolution. The literature value for conductivity of this solution is 1. 413 mS cm? 1 at 25 o C.

If the readout deviates from this, perform calibration of the conductivity meter (see the manual; adjust the two RVs for both measurement scales). Note that the instrument readout units are mS / cm. After calibration, proceed in measuring your electrolyte solutions. Carry out measurements always starting from the most dilute sample and working towards the most concentrated one. When proceeding to a new compound, change the beaker. IV. DATA ANALYSIS 1. Limiting conductivities for strong electrolytes: Use Eq. (5) to obtain the limiting molar conductivities (? ) for m each strong electrolyte by using the Maxima program given on the course web pages (“ experiment4a. mac”). Note that the values entered in the program are molar quantities (i. e. , use Eq. (4) to get them). Remember to press shift-return at the end of each line. Record both ? tted values as well as their errors from the least squares procedure. The obtained limiting conductivity values are required to calculate ? in Eq. (26) and ? nally pKa for CH3 COOH. The Maxima program for obtaining the limiting conductivities is given below: /* * Unconstrained nonlinear least squares fit module for Maxima. * nlsq(funcs, ydata, xdata, yvars, xvars, params, initvars, errtol, debug) * * funcs = list of functions for fitting [fun1, fun2, … ]. * ydata = list of y-data vectors [yvec1, yvec2, … ]. * xdata = list of x-data vectors [xvec1, xvec2, … ]. * yvars = list of y variables for the functions given in funcs [y1, y2, … ]. * xvars = list of x variables for the functions given in funcs [x1, x2, … ]. * params = list of parameters to be fitted [a, b, … ]. * initvars = list of initial values for params [ai, bi, … ]. * errtol = requested error tolerance in BFGS (1E-4). debug = optimization output vector for BFGS ([1, 2] = full output, [-1 0] = no output). * * Returns: [F, R2, V, C, E]. * * F = list of optimized functions. * R2 = list of r^2 values for each function. * V = list of optimized parameter values. * C = covariance matrix for the parameters. * E = list of standard errors for parameters. * */ load(“ lbfgs”); nlsq([ArgList]):= block([narg, funcs, ydata, xdata, yvars, xvars, params, initvars, errtol, debug, neqs, i, sq, tmp, lbfgs_nfeval_max: 1000, /* BFGS tends to converge somewhat slowly… / fun, fit, F, ss, sstot, sm, st, R2, C, lp, E, sigma2, Vm, k, np], narg: length(ArgList), if narg # 9 then ( print(“ nlsq: bad number of function arguments (requires 9 arguments). “), return(false) ), 5 funcs: ArgList[1], ydata: ArgList[2], xdata: ArgList[3], yvars: ArgList[4], xvars: ArgList[5], params: ArgList[6], initvars: ArgList[7], errtol: ArgList[8], debug: ArgList[9], neqs: length(funcs), lp: length(params), if neqs # length(ydata) or neqs # length(xdata) or neqs # length(yvars) or neqs # length(xvars) then ( print(“ nlsq: Inconsistent lengths for arguments 1 – 5. ), return(false) ), /* Construct the least squares sum and call lbfgs to optimize */ tmp: 0, for i: 1 thru neqs do ( if length(xdata[i]) #

length(ydata[i]) then ( print(“ nlsq: Inconsistent lengths for X and Y in set “, i), return(false) ), sq:(lhs(funcs[i]) – rhs(funcs[i]))^2, sq: subst(’xdata[i][j], xvars[i], sq), sq: subst(’ydata[i][j], yvars[i], sq), tmp: tmp + ’sum(sq, j, 1, length(xdata[i])) ), /* fun: ev(tmp, nouns),*/ fun: tmp, fit: lbfgs(fun, params, initvars, errtol, debug), if fit = [] then ( print(“ nlsq: BFGS convergence error. Fit failed. ), return(false) ), /* Substitute the optimized values back into the equations */ F: makelist(0, i, 1, neqs), for i: 1 thru neqs do ( F[i]: float(rhs(funcs[i])), for j: 1 thru lp do F[i]: subst(rhs(fit[j]), params[j], F[i]) ), /* Calculate r^2 for each data set */ sstot: 0, R2: makelist(0, i, 1, neqs), for i: 1 thru neqs do ( ss: sum((subst(xdata[i][j], xvars[i], F[i]) – ydata[i][j])^2, j, 1, length(xdata[i])), sstot: sstot + ss, sm: sum(ydata[i][j], j, 1, length(xdata[i])) / length(xdata[i]), st: sum((subst(xdata[i][j], xvars[i], F[i]) – sm)^2, j, 1, length(xdata[i])), R2[i]: float(ev(1 – ss / st, nouns)) ), /* Calculate the covariance matrix */ C: ematrix(lp, lp, 0, 1, 1), for i: 1 thru lp do for j: 1 thru lp do ( C[i, j]: diff(fun, params[i], 1, params[j], 1), for k: 1 thru lp do C[i, j]: subst(rhs(fit[k]), params[k], C[i, j]) ), C: float(ev(C, nouns)), /* watch out when lp = 1, invert returns a number not a matrix */ if lp = 1 then ( C[1, 1]: 1/C[1, 1] ) else ( C: invert(C) ), /* Calculate standard errors */ np: sum(length(xdata[i]), i, 1, neqs), sigma2: sstot / (np – lp), E: makelist(0, i, 1, lp), V: makelist(0, i, 1, lp), for i: 1 thru lp do ( E[i]: float(sqrt(sigma2 * C[i, i])), V[i]: float(rhs(fit[i])) ), return([F, R2, V, C, E]) ); /* * Experiment 4.

Conductivity of electrolyte solutions. * * Given strong electrolyte concentrations and molar conductivities, * this program extracts the limiting conductivity for the electrolyte solution. * Use SI units for your molar conductivities (e. g. S m^2 / mol). * * This program should be run separately for each electrolyte solution * (except CH3COOH). * */ remvalue(all); /* 5 different concentrations (approx. 0. 03 M, 0. 02 M, 0. 015 M, 0. 01 M, 0. 005 M) */ print(“ Enter concentration and molar conductivity (S m^2 / mol) pairs for the given electrolyte. ); print(“ The values must be separated by space and each pair must be terminated by pressing return. “); print(“ The input consists of five lines each terminated by return. “); print(“ Enter the lowest concentration first. “); print(“”); x: makelist(0, i, 1, 5); y: makelist(0, i, 1, 5); for i: 1 thru 5 do ( print(“ enter data: “), tmp: tokens(readline(? *standard-input*)), x[i]: parse_string(tmp[1]), y[i]: parse_string(tmp[2]) ); /* Functions to be fitted */ funs:[’y=-k*sqrt(c) + lc]; 7 /* Initial guesses for k and lc */ ki: 0; lci: 0; /* Least squares fit */ print(“ Optimizing – WAIT… ); fit: nlsq(funs,[y],[x],[’y],[’c],[’k,’lc],[ki, lci], 1E-4,[-1, 0]); /* Plot graphs (exp. vs. calculated) */ /* remove wx if used under command line maxima */ wxplot2d([fit[1][1], [discrete, x, y]], [c, x[1], x[5]], [style, [lines, 1, 1], [points, 1, 2, 6]], [legend,” Theory (conductivity)”,” Experiment (conductivity)”], [xlabel,” Concentration (M)”],[ylabel,” Molar conductivity (S m2 / mol)”]); print(“ r2 for the fit = “, fit[2][1]); print(“ k = “, fit[3][1],” +- “, fit[5][1], ” (“, 100 * fit[5][1] / abs(fit[3][1]),”%). ); print(“ lc = “, fit[3][2],” +- “, fit[5][2], ” (“, 100 * fit[5][2] / abs(fit[3][2]),”%). “); 2. pKa of acetic acid: After determining the limiting conductivities for the weak electrolytes, the conductivity data for acetic acid (weak electrolyte) will be used to determine the pKa . This is done with the following Maxima program (“ experiment4b. mac”): /* * Unconstrained nonlinear least squares fit module for Maxima. * * nlsq(funcs, ydata, xdata, yvars, xvars, params, initvars, errtol, debug) * * funcs = list of functions for fitting [fun1, fun2, … . * ydata = list of y-data vectors [yvec1, yvec2, … ]. * xdata = list of x-data vectors [xvec1, xvec2, … ]. * yvars = list of y variables for the functions given in funcs [y1, y2, … ]. * xvars = list of x variables for the functions given in funcs [x1, x2, … ]. * params = list of parameters to be fitted [a, b, … ]. * initvars = list of initial values for params [ai, bi, … ]. * errtol = requested error tolerance in BFGS (1E-4). * debug = optimization output vector for BFGS ([1, 2] = full output, [-1 0] = no output). * Returns: [F, R2, V, C, E]. * * F = list of optimized functions. * R2 = list of r^2 values for each function. * V = list of optimized parameter values. * C = covariance matrix for the parameters. * E = list of standard errors for parameters. * */ load(“ lbfgs”); nlsq([ArgList]):= block([narg, funcs, ydata, xdata, yvars, xvars, params, initvars, errtol, debug, neqs, i, sq, tmp, lbfgs_nfeval_max: 1000, /* BFGS tends to converge somewhat slowly… */ fun, fit, F, ss, sstot, sm, st, R2, C, lp, E, sigma2, Vm, k, np], narg: length(ArgList), 8 f narg # 9 then ( print(“ nlsq: bad number of function arguments (requires 9 arguments). “), return(false) ), funcs: ArgList[1], ydata: ArgList[2], xdata: ArgList[3], yvars: ArgList[4], xvars: ArgList[5], params: ArgList[6], initvars: ArgList[7], errtol: ArgList[8], debug: ArgList[9], neqs: length(funcs), lp: length(params), if neqs # length(ydata) or neqs # length(xdata) or neqs # length(yvars) or neqs # length(xvars) then ( print(“ nlsq: Inconsistent lengths for arguments 1 – 5. ), return(false) ), /* Construct the least squares sum and call lbfgs to optimize */ tmp: 0, for i: 1 thru neqs do ( if length(xdata[i]) # length(ydata[i]) then ( print(“ nlsq: Inconsistent lengths for X and Y in set “, i), return(false) ), sq:(lhs(funcs[i]) – rhs(funcs[i]))^2, sq: subst(’xdata[i][j], xvars[i], sq), sq: subst(’ydata[i][j], yvars[i], sq), tmp: tmp + ’sum(sq, j, 1, length(xdata[i])) ), /* fun: ev(tmp, nouns),*/ fun: tmp, fit: lbfgs(fun, params, initvars, errtol, debug), if fit = [] then ( print(“ nlsq: BFGS convergence error. Fit failed. ), return(false) ), /* Substitute the optimized values back into the equations */ F: makelist(0, i, 1, neqs), for i: 1 thru neqs do ( F[i]: float(rhs(funcs[i])), for j: 1 thru lp do F[i]: subst(rhs(fit[j]), params[j], F[i]) ), /* Calculate r^2 for each data set */ sstot: 0, R2: makelist(0, i, 1, neqs), for i: 1 thru neqs do ( ss: sum((subst(xdata[i][j], xvars[i], F[i]) – ydata[i][j])^2, j, 1, length(xdata[i])), sstot: sstot + ss, sm: sum(ydata[i][j], j, 1, length(xdata[i])) / length(xdata[i]), st: sum((subst(xdata[i][j], xvars[i], F[i]) – sm)^2, j, 1, length(xdata[i])), 9

R2[i]: float(ev(1 – ss / st, nouns)) ), /* Calculate the covariance matrix */ C: ematrix(lp, lp, 0, 1, 1), for i: 1 thru lp do for j: 1 thru lp do ( C[i, j]: diff(fun, params[i], 1, params[j], 1), for k: 1 thru lp do C[i, j]: subst(rhs(fit[k]), params[k], C[i, j]) ), C: float(ev(C, nouns)), /* watch out when lp = 1, invert returns a number not a matrix */ if lp = 1 then ( C[1, 1]: 1/C[1, 1] ) else ( C: invert(C) ), /* Calculate standard errors */ np: sum(length(xdata[i]), i, 1, neqs), sigma2: sstot / (np – lp), E: makelist(0, i, 1, lp), V: makelist(0, i, 1, lp), for i: 1 thru lp do ( E[i]: float(sqrt(sigma2 * C[i, i])), V[i]: float(rhs(fit[i])) ), return([F, R2, V, C, E]) ); /* * Experiment 4. Conductivity of electrolyte solutions. * * Given the limiting conductivities and concentration vs. molar conductivity for acetic acid, * this program extracts pKa for acetic acid. * * Note that you must download the nlsq. mac separately. * */ remvalue(all); /* In Maxima, log() is natural logarithm. Base 10 logarithm must be defined as: */ log10(x) := log(x) / log(10); /* Limiting conductivities for strong electrolytes */ print(“ Enter limiting conductivity for HCl and press enter. “); lc1: parse_string(readline(? *standard-input*)); print(“ Enter limiting conductivity for NaCl and press enter. “); lc2: parse_string(readline(? *standard-input*)); print(“ Enter limiting conductivity for CH3COONa and press enter. “); lc3: parse_string(readline(? *standard-input*)); /* 5 different acetic acid concentrations (approx. 0. 03 M, 0. 02 M, 0. 015 M, 0. 01 M, 0. 005 M) */ print(“ Enter concentration and molar conductivity (S m^2 / mol) pairs for acetic acid. “); print(“ The values must be separated by space and each pair must be terminated by pressing enter. “); print(“ Overall, the input consists of five lines each terminated by enter. “); x: makelist(0, i, 1, 5); 10 y: makelist(0, i, 1, 5); xa: makelist(0, i, 1, 5); ya: makelist(0, i, 1, 5); for i: 1 thru 5 do ( print(“ enter data: “), tmp: tokens(readline(? *standard-input*)), x[i]: parse_string(tmp[1]), y[i]: parse_string(tmp[2]), alpha: y[i] / (lc1 – lc2 + lc3), /* Transform to a linear problem */ xa[i]: sqrt(alpha * x[i]), ya[i]:-log10(alpha * alpha * x[i] / (1 – alpha)) ); /* Function to be fitted (note rhob^theta is approx. 1) */ funs:[’y= k*’x + b]; /* Initial guesses for k and b */ ki : 0; bi : 0; /* Least squares fit */ print(“ Optimizing – WAIT… “); fit: nlsq(funs,[ya],[xa],[’y],[’x],[’k,’b],[ki, bi], 1E-8,[-1, 0]); /* Plot graphs (exp. vs. calculated) */ /* remove wx if used under command line maxima */ wxplot2d([fit[1][1], [discrete, xa, ya]], [’x, xa[1], xa[5]], [style, [lines, 1, 1], [points, 1, 2, 6]], [legend,” Theory”,” Experiment”], [xlabel,” x”],[ylabel,” y”]); print(“ r2 for the fit = “, fit[2][1]); print(“ k = “, fit[3][1],” +- “, fit[5][1], ” (“, 100 * fit[5][1] / abs(fit[3][1]),”%). ); print(“ b = “, fit[3][2],” +- “, fit[5][2], ” (“, 100 * fit[5][2] / abs(fit[3][2]),”%). “); When the program is run, it will ask the user for the required data and output the results from the least squares ? t. The output will indicate the slope (k) and intercept (b) according to: ? 2 [HA]0 ? b? (1 ? ?) “ y” 1 ? ? b? ? log = ? 2A “ k” ? [HA]0 + pKa “ x” “ b” (27) where, for example, the value of b corresponds directly to pKa . V. WRITTEN LABORATORY REPORT Follow the general instructions for written laboratory reports. In addition, include the requested data in the following section: Results. This section must include the values and error estimates for the values of ? 0 (NaCl), ? (HCl), m m 0 ? m (CH3 COONa), ? 0 (CH3 COOH) and pKa (from least squares analysis). Find also the literature values for these m 11 variables (see Ref. [1], for example). Discuss the possible sources of error, if your data deviates signi? cantly from a straight line in any of the ? ts. VI. REFERENCES [1] P. W. Atkins and J. de Paula, Physical Chemistry (7th ed. ) (Oxford University Press, Oxford, UK, 2002). [2] R. J. Silbey, R. A. Alberty, and M. G. Bawendi, Physical Chemistry (4th ed. ) (Wiley, New York, 2004). [3] R. Chang, Physical chemistry for the chemical and biological sciences (University Science Books, Sausalito, California, 2000). 12