



Supporting information B. C-program for calculating the model results.

Template for parameter input file (name = graph.in)

```
0.1      | NaCl Concentration for Graph
0.02     | density increment for Graph
0.2      | minimum density (kg/dm3)
1.9      | maximum density (kg/dm3)
|crystallographic ssa
748.9|
|charge
-0.81 |
|Input for models 1 and 2
ssa | nc | nb_Debye      |
748.9| 3 | 2          |
|Input for models 3 and 4
ssa | ncmin | nb_Debye | dporemin (nDL) | hint_min |
748.9| 1.5 | 2          | 5.51          | 0.31e-9   |
```

C. file

```
#include <stdio.h>
#include <math.h>
#include <float.h>
#include <stdlib.h>

double rhomm = 2.841; /* kg/dm3 */

int main() {

/***** Graph characteristics - start *****/

static int get_line(char *s, int lim, FILE *f_in);
static int copy_until(int start_pos, char *in, char *out, char stop);

double Concentration = 0;
double delta_rho = 0;
double rho_min = 0;
double rho_max = 0;
char line[1000], l1[50], stop_char;
int start_pos;
double ssa_cryst;
double charge_model;
double ssa_model1, ssa_model2;
double nc_model1, ncmin_model2;
double nb_debye_model1, nb_debye_model2;
double dpore_min, hint_min;
static double Debye_length(double C0);
static double C_Donnan(double C0, double q);
```

```

static double h_int(double C0, double rho);
static double Volume_Donnan(double surf, double nb_Debye, double C0);
static double Calculate_nc(double rho, double C0, double surf, double ncm, double nDL);
static double model(double rho, double C0, double charge, double surf, double nb_debye,
double ncm, double nDL, double hintmin);
double rho;
double por_tot, por_inta, por_intb;
double hint;
double model0, model1, model2, model3, model4;
double nca, ncb;

FILE *graphout, *graphin = fopen("Graph.in", "r");
stop_char = "|";

/* NaCl Concentration */
get_line(line, 1000, graphin); copy_until(0, line, 11, stop_char);
Concentration = atof(11);
/* Density increment */
get_line(line, 1000, graphin); copy_until(0, line, 11, stop_char);
delta_rho = atof(11);
/* Min density */
get_line(line, 1000, graphin); copy_until(0, line, 11, stop_char);
rho_min = atof(11);
/* max density*/
get_line(line, 1000, graphin); copy_until(0, line, 11, stop_char);
rho_max = atof(11);

/***** Graph characteristics - end *****/

/***** Model inputs - start *****/

// double ssa_cryst;
get_line(line, 1000, graphin);
get_line(line, 1000, graphin); copy_until(0, line, 11, stop_char);
ssa_cryst = atof(11);

// double charge_model;
get_line(line, 1000, graphin);
get_line(line, 1000, graphin); copy_until(0, line, 11, stop_char);
charge_model = atof(11);

// double ssa_model1, ssa_model2;
// double nc_model1, ncm_model2;
// double nb_debye_model1, nb_debye_model2;
// double dpore_min, hint_min;

get_line(line, 1000, graphin);
get_line(line, 1000, graphin);
get_line(line, 1000, graphin);
start_pos = copy_until(0, line, 11, stop_char); ssa_model1 = atof(11);

```

```
start_pos = copy_until(start_pos, line, l1, stop_char); nc_model1 = atof(l1);
copy_until(start_pos, line, l1, stop_char); nb_debye_model1 = atof(l1);
```

```
get_line(line, 1000, graphin);
get_line(line, 1000, graphin);
get_line(line, 1000, graphin);
start_pos = copy_until(0, line, l1, stop_char); ssa_model2 = atof(l1);
start_pos = copy_until(start_pos, line, l1, stop_char); ncmin_model2 = atof(l1);
start_pos = copy_until(start_pos, line, l1, stop_char); nb_debye_model2 = atof(l1);
start_pos = copy_until(start_pos, line, l1, stop_char); dpore_min = atof(l1);
copy_until(start_pos, line, l1, stop_char); hint_min = atof(l1);
```

```
printf (" NaCl Concentration: \t%.5f mol/L\n density increment: \t%.5f kg/dm3\n min
density: \t%.5f kg/dm3\n max density: \t%.5f kg/dm3\n", Concentration, delta_rho,
rho_min, rho_max);
```

```
printf (" ssa_cryst: \t%.5f m2/g\n charge: \t%.5f molc/kg\n ssa model 1: \t%.5f m2/g\n
nc model 1: \t%.5f\n nb debye model 1: \t%.5f\n", ssa_cryst, charge_model, ssa_model1,
nc_model1, nb_debye_model1);
```

```
printf (" ssa model 2: \t%.5f m2/g\n nc min model 2: \t%.5f\n nb debye model 2: \t%.5f\n
dpore min: \t%.12f * DL\n hint min: \t%.12f m\n", ssa_model2, ncmin_model2,
nb_debye_model2, dpore_min, hint_min);
```

```
fclose(graphin);
```

```
/****** Model inputs - end ******/
```

```
/****** Output -start ******/
```

```
// static double Debye_length(double C0);
// static double C_Donnan(double C0, double q);
// static double h_int(double C0, double rho);
// static double Volume_Donnan(double surf, double nb_Debye, double C0);
// static double Calculate_nc(double rho, double C0, double surf, double ncmin, double
nDL);
// static double model(double rho, double C0, double charge, double surf, double nb_debye,
double ncmin, double nDL, double hintmin);
// double rho;
// double por_tot, por_inta, por_intb;
// double hint;
// double model0, model1, model2, model3;
// double nca, ncb;
```

```
nca = 3;
ncb = 25;
```

```
/*FILE */graphout = fopen("Graph.out", "w");
```

```
fprintf (graphout, "NaCl (mol/L)\tDensity (kg/dm3)\tTotal porosity\tth int (nm)\tInterlayer
porosity nc = 3 and ssa_cryst\tInterlayer porosity nc = 25 and ssa_cryst\tModel BK\tModel
1\tModel 2\tModel 3\tModel 4\n");
```

```

for (rho = rho_min; rho <= rho_max + delta_rho; rho += delta_rho) {

    por_tot = 1 - rho / rhomm;
    hint = h_int(Concentration, rho);
    por_inta = hint * ssa_cryst * 1000000 * rho * (1 - 1/nca) / 2;
    if (por_inta > por_tot)
        {
            por_inta = por_tot;
        }
    por_intb = hint * ssa_cryst * 1000000 * rho * (1 - 1/ncb) / 2;
    if (por_intb > por_tot)
        {
            por_intb = por_tot;
        }
    hint = h_int(Concentration, rho);
    /* Birgersson & Karland, all porespace is Donnan... */
    model0 = model(rho, Concentration, charge_model, ssa_cryst, 1500, 1, 0, 0);

    model1 = model(rho, Concentration, charge_model, ssa_model2, 1500, 8.4, 0, 0) ;
    model2 = model(rho, Concentration, charge_model, ssa_model2, nb_debye_model1, 2.48,
0, 0.0) ;
    model3 = model(rho, Concentration, charge_model, ssa_model2, nb_debye_model2,
ncmin_model2, dpore_min, 0.62e-9) ;
    model4 = model(rho, Concentration, charge_model, ssa_model2, nb_debye_model2,
ncmin_model2, dpore_min, hint_min) ;

    fprintf                                     (graphout, "
%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.5f\t%.12f\t%.12f\t%.12f\t%.12f\t%.12f\n  ", Concentration,
rho, por_tot, hint*1e9, por_inta, por_intb, model0, model1, model2, model3, model4 );
    }

    fclose(graphout);
    return 0;
}

/***** Output *****/
*****/

/***** Functions for calculation -start *****/

double Debye_length(double C0) {
    return pow( (2.0 * pow(96485, 2) * 1e3 * C0 / (8.314 * 298 * 0.00000000069328)), -0.5);
}

double C_Donnan(double C0, double q) {
    return (2*C0*C0 / (-q + pow( (q*q + 4* C0*C0), 0.5)));
}

```

```

}

double h_int(double C0, double rho) {
    double x2, x3, rho_lim;
    rho_lim = 1.3 - 3 * C0;
    x3 = 1.0; x2 = 0.0;
    if (rho > rho_lim)
    {
        x2 = (rho - rho_lim) / (1.6 - rho_lim);
        x3 = 1 - x2;
    }
    if (rho > 1.6)
    {
        x3 = 0.0; x2 = 1.0;
    }
    return (x2 * 0.62 + x3 * 0.94) * 1e-9;
}

```

```

double Volume_Donnan(double surf, double nb_Debye, double C0) {
    return surf * nb_Debye * Debye_length(C0) * 1e3;
}

```

```

double Calculate_nc(double rho, double C0, double surf, double ncmin, double nDL) {
    double nc2 = ncmin;
    double dpore;
    double vol_int, surf_int, vol_stern, vol_ext, surf_ext;

    surf_int = surf * (nc2 - 1)/nc2 * rho * 1000;      /* in m2 */
    vol_int = surf_int * h_int(C0, rho) * 1000 / 2;   /* in dm3 */
    surf_ext = surf / nc2 * rho * 1000;              /* in m2 */
    vol_stern = 0; /* surf_ext * 1.84e-10 * 1000; */
    vol_ext = 1 - vol_int - vol_stern - rho / rhomm;  /* in dm3 */
    if (vol_ext < 0)
    {
        vol_ext = 0;
    }

    dpore = 2 * vol_ext / 1000 / surf_ext;           /* in m */

    while ((nc2 < 30) && (dpore < nDL * Debye_length(C0)))
    {
        nc2 += 0.01;
        surf_ext = surf / nc2 * rho * 1000;
        surf_int = surf * (nc2 - 1)/nc2 * rho * 1000;
        vol_int = surf_int * h_int(C0, rho) * 1000 / 2;
        vol_stern = 0; /* surf_ext * 1.84e-10 * 1000; */
        vol_ext = 1 - vol_int - vol_stern - rho / rhomm;
        dpore = 2 * vol_ext / 1000 / surf_ext;
    }
    return nc2;
}

```

```
}
```

```
/****** Functions for calculation - end *****/
```

```
/******  
***** Models -start  
******/
```

```
/******  
***** Model  
******/
```

```
double model(double rho, double C0, double charge, double surf, double nb_debye, double  
ncmin, double nDL, double hintmin) /*** surface in m2/g, charge in molc/kg *****/
```

```
{
```

```
double vol_int, surf_int, vol_stern, vol_ext, surf_ext, vol_Donnan, vol_free, q;  
double nc2 = ncmin;  
double result;  
double hint;  
double dpore;
```

```
nc2 = Calculate_nc(rho, C0, surf, ncmin, nDL);  
hint = h_int(C0, rho);
```

```
surf_int = surf * (nc2 - 1)/nc2 * rho * 1000; /* in m2 */  
vol_int = surf_int * hint * 1000 / 2; /* in dm3 */  
surf_ext = surf / nc2 * rho * 1000; /* in m2 */  
vol_stern = 0; /*surf_ext * 1.84e-10 * 1000;*/  
vol_ext = 1 - vol_int - vol_stern - rho / rhomm; /* in dm3 */  
dpore = 2 * vol_ext / 1000 / surf_ext;
```

```
while ((hint > hintmin) && (dpore < nDL * Debye_length(C0)))
```

```
{  
hint -= 0.001e-9;  
vol_int = surf_int * hint * 1000 / 2;  
vol_ext = 1 - vol_int - vol_stern - rho / rhomm;  
dpore = 2 * vol_ext / 1000 / surf_ext;  
}
```

```
if (vol_ext < 0)
```

```
{  
vol_ext = 0;  
}
```

```
vol_Donnan = Volume_Donnan(surf_ext, nb_debye, C0); /* in dm3 */
```

```
if (vol_Donnan > vol_ext)
```

```
{  
vol_Donnan = vol_ext;  
}
```

```
vol_free = vol_ext - vol_Donnan;
```

```

if (vol_Donnan == 0)
{
    result = 0;
}
else
{
    q = charge / nc2 * rho / vol_Donnan;
    result = (vol_free + vol_Donnan * C_Donnan(C0, q) / C0);
}
return result;
}

```

```

/*****                               Models                               -end                               of                               models
*****/

```

```

/***** Procedures for files reading - start *****/

```

```

int get_line(char *s, int lim, FILE *f_in) {
    int i, c, space;
    space = c = 1;
    for (i = 0; i < lim && (c = fgetc(f_in)) != EOF && c != (int) '\n'; i++) {
        s[i] = (char) c;
        if (space && c != 32) space = 0;
    }
    if (space) i = 0;
    s[i] = '\0';
    if (i == 0 && c == EOF)
        return EOF;
    else
        return 0;
}

```

```

int copy_until(int i, char *in, char *out, char stop) {
    int j;
    j = 0;
    while ((in[i] != stop) && (out[j] = in[i]) != '\0') {
        i++; j++;
    }
    out[j] = '\0';
    return ++i;
}

```

```

/***** Procedures for files reading - end *****/

```