Electronic Supplementary Material (ESI) for Soft Matter.

Charged polymer with constrained equilibrium in *μ-NpT* ensemble

Setup

$\beta = 1$; (* 1 $k_B T$ *)

$\gamma = 0.0257$; (* 1 $k_B T$ = 0.0257 meV @ 298 K *)

$n_{mer} = 200$;

$\xi = 0.1$;

$b_{bond} = 0.38$;

$\sigma = 0.3385$;

$z_{mer} = 1$;

$\sigma_{co} = \sqrt{2}\,\sigma$;

$\epsilon_{co} = 0.1$;

$z_{co} = z_{mer}$;

$\sigma_{cx} = \sigma$;

$\epsilon_{cx} = 0.1$;

$z_{cx} = -z_{mer}$;

rid[n_: n_{mer}] := $b_{bond}\sqrt{\frac{n}{6}}$

potLJ[r_, σ_, ϵ_] := $4\,\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)$

evalB2LJ[σ_, ϵ_] := evalB2[σ, ϵ] = $-2\,\pi\int_0^\infty rr^2\,(\text{Exp}[-\beta\,\text{potLJ}[rr, \sigma, \epsilon]] - 1)\,d rr$

evalB3HS[σ_: σ] := $10.0\left(\frac{\pi}{6}\right)^2\sigma^6$

evalB4HS[σ_: σ] := $18.3647684\left(\frac{\pi}{6}\right)^3\sigma^9$

$b_{2\,cc}$ = evalB2LJ[σ, $0.3\,\beta^{-1}$];

$b_3 = 2\,\sigma^6$;

b_4 = evalB4HS[σ];

Automatic setup

In[21]:=
```
b3 mm = b3 ;
b4 mm = b4 ;
b2fn[eps_] := IntegerString[IntegerPart[10 eps], 10, 2] ;
b3fn[b3_: b3] := Switch[b3, evalB3HS[σ], "hs", 2 σ^6, "2sg6", 0, "0", _, "other"] ;
b4fn[b4_: b4] := Switch[b4, evalB4HS[σ], "hs", 0, "0", _, "other"] ;
b3label[b3_: b3] := Switch[b3, evalB3HS[σ], "B3 ~ HS",
   2 σ^6, "B3 = 2 σ^6", 0., "B3 = 0", _, StringForm["B3 = `` nm^6", b3]] ;
b4label[b4_: b4] := Switch[b4, evalB4HS[σ], "B4 ~ HS", 0.,
   "B4 = 0", _, StringForm["B4 = `` nm^9", b4]] ;

outputdir = NotebookDirectory[] <> "output" ;
If[! DirectoryQ@outputdir, CreateDirectory@outputdir] ;
SetDirectory@outputdir ;
```

Chain without cosolute

In[31]:=

$$v[r\_] := \frac{4}{3} \pi r^3$$

$$c[r\_, n\_] := \frac{n}{v[r]}$$

$$\alpha[r\_, n\_: n_{mer}] := \frac{r}{rid[n]}$$

$$f_{Fchain}[r\_, n\_] := \beta^{-1} \frac{4}{9} \left(\alpha[r, n]^2 + \alpha[r, n]^{-2}\right)$$

$$f_{B2chain}[r\_, n\_, \epsilon mm\_] := \beta^{-1} v[r]\, c[r, n]^2\, evalB2LJ\left[\sigma, \epsilon mm\, \beta^{-1}\right]$$

$$f_{BNchain}[r\_, n\_] := \beta^{-1} v[r] \left(\frac{b_{3\,mm}}{2} c[r, n]^3 + \frac{b_{4\,mm}}{3} c[r, n]^4\right)$$

$$f_{chain}[r\_, n\_, \epsilon mm\_] := f_{Fchain}[r, n] + f_{B2chain}[r, n, \epsilon mm] + f_{BNchain}[r, n]$$

Chain in the cosolute environment

Common setup

In[590]:=
```
asCsvList[l_] :=
 Map[NumberForm[#1, 4] &, Map[Flatten, Map[#1[[2]] &, l, {3}], {1}], {2}]
asPlotList[l_] := Transpose@
  Apply[Partition[Riffle[#1[[-1]] & /@ #2 , #1, {1, -2, 2} ] , 2 ] &, l, {1}]
csvfilename[name_, ξ_] := name <> ToString@IntegerPart[nmer ξ] <> "q.csv"
```

In[40]:=
```
ϵmm = 0.4 ;
lϵmc = {0.1, 0.3, 0.5, 0.7, 0.9} ;
```

Free energy minimization

Chemical potential and osmotic pressure of the reservoir

$$\beta F = \frac{B_2 N^2}{V} + \frac{B_3 N^3}{2 V^2} + \frac{B_4 N^4}{3 V^3} + N\left(\log\left(\Lambda^3 \frac{N}{V}\right) - 1\right)$$

$$\beta \mu = \beta \left.\frac{\partial F}{\partial N}\right|_{V,T} = 2 B_2 c + \frac{3}{2} B_3 c^2 + \frac{4}{3} B_4 c^3 + \log(c) + 3 \log(\Lambda)$$

$$\beta p = -\beta \left.\frac{\partial F}{\partial V}\right|_{N,T} = c + B_2 c^2 + B_3 c^3 + B_4 c^4$$

```
In[42]:= μout[cout_] := 2 b2 cc (2 cout) + 3/2 b3 (2 cout)^2 + 4/3 b4 (2 cout)^3 + Log[cout] (*+3Log[Λ]*)
pout[cout_] := β^-1 (2 cout + b2 cc (2 cout)^2 + b3 (2 cout)^3 + b4 (2 cout)^4)
```

Helmholtz free energy of the globule (*NVT*)

Using the number of cosolute particles inside the globule instead of cosolute concentration speeds up the minimization.

```
In[44]:= fin[r_, ncx_, nco_, ϵcx_, ϵco_] :=
  1/v[r] (b2 cc (ncx + nco)^2 + 2 nmer (evalB2LJ[σcx, ϵcx] ncx + evalB2LJ[σco, ϵco] nco)) +
   b3/(2 v[r]^2) ((ncx + nco + nmer)^3 - nmer^3) + b4/(3 v[r]^3) ((ncx + nco + nmer)^4 - nmer^4) +
   ncx Log[ncx/v[r]] + nco Log[nco/v[r]] + (ncx + nco) ((*3Log[Λ]*) - 1)
```

Gibbs free energy of the globule

```
ϵmm = 0.4;
lϵcx = {0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5};
lξ = {-0.1, -0.05, 0, 0.05, 0.1};
lξ = {0, 0.05, 0.1};
ϕ = 0 ;
```

```
In[50]:= charge[ncx_, nco_, ξ_] := ξ zmer nmer + zcx ncx + zco nco
g[r_, ncx_, nco_, cout_, ξ_, ϵmm_, ϵcx_, ϵco_] :=
  fchain[r, nmer, ϵmm] + fin[r, ncx, nco, ϵcx, ϵco] +
   pout[cout] v[r] - μout[cout] (ncx + nco) + ϕ charge[ξ, ncx, nco]
g[r_, ncx_, nco_, cout_, ϵmm_, ϵcx_, ϵco_] :=
  fchain[r, nmer, ϵmm] + fin[r, ncx, nco, ϵcx, ϵco] + pout[cout] v[r] - μout[cout] (ncx + nco)
```

In[53]:=
```
ClearAll[gMin]
gMin[cs_, ξ_, ϵmm_, ϵcx_, ϵco_] := gMin[cs, ξ, ϵmm, ϵcx, ϵco] = FindMinimum[
   {Simplify[g[r, ncx, nco, cs, ϵmm, ϵcx, ϵco]] /. ncx → - (ξ zmer nmer + zco nco) / zcx,
    nco ≥ 0, r > 0}, {{r, 1.1 rid[]}, {nco, cs v[rid[]]}}, AccuracyGoal → 4]
```

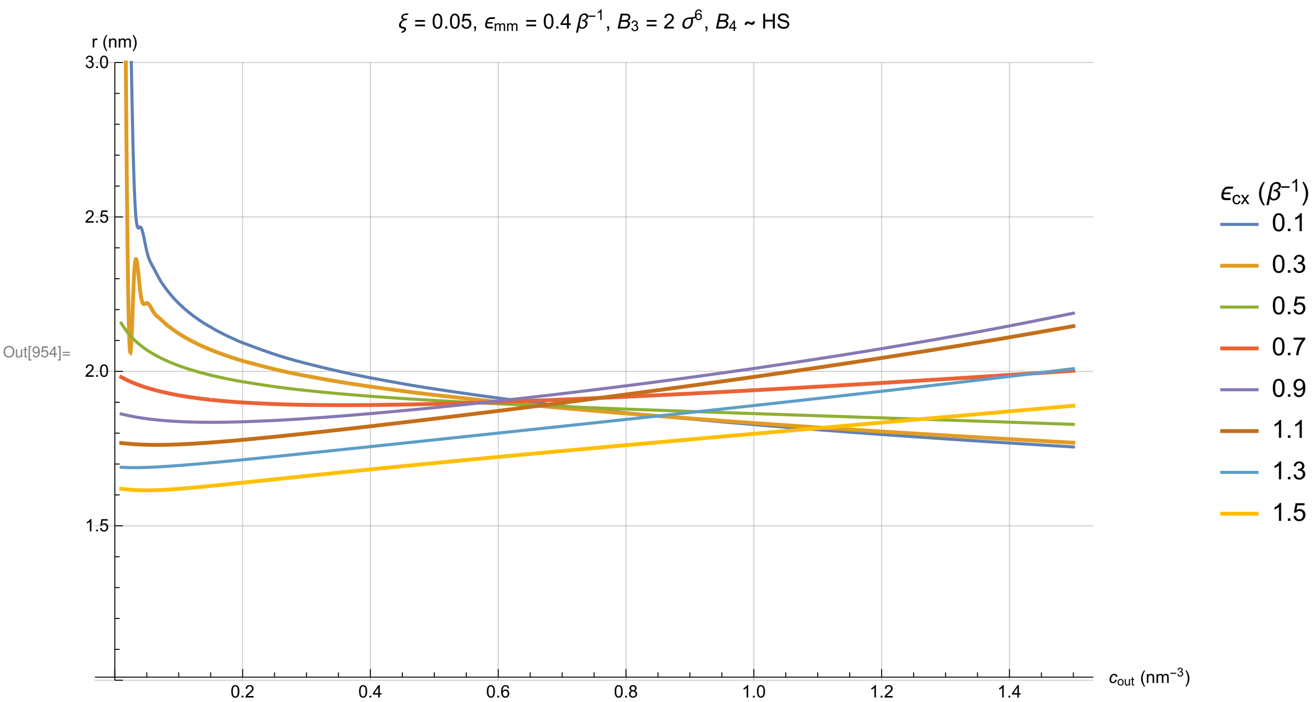
Minimal radius of gyration

In[948]:=
```
ξ = 0.05 ;
```

In[952]:=
```
trmin = Table[{vcs, Table[{vϵcx, r /. gMin[vcs, ξ, ϵmm, vϵcx, ϵco][[2]]}, {vϵcx, lϵcx}]},
   {vcs, Union[Table[vcs, {vcs, 0.01, 0.2, 0.01}],
     Table[vcs, {vcs, 0.15, 0.5, 0.05}], Table[vcs, {vcs, 0.5, 1.5, 0.1}]]}] ;
```

In[953]:=
```
plotOpts = {PlotLabel →
    StringForm["ξ = ``, ϵmm = ``β^-1, ``, ``", ξ, ϵmm, b3label[], b4label[]],
   ImageSize → Large, PlotStyle → {PointSize[Large], Thickness[Large]},
   PlotLegends → LineLegend[Table[NumberForm[ϵcx, {2, 1}], {ϵcx, lϵcx}],
     LegendLabel → "ϵcx (β^-1)"], GridLines → Automatic, InterpolationOrder → 3} ;
ListLinePlot[asPlotList[trmin], PlotRange → {1.0, 3.0},
 AxesLabel → {"cout (nm^-3)", "r (nm)"}, Evaluate@plotOpts]
```

Out[954]=



In[638]:=
```
trmin = Table[{vcs, Table[{vϵcx, r /. gMin[vcs, ξ, ϵmm, vϵcx, ϵco][[2]]}, {vϵcx, lϵcx}]},
   {vcs,
    Union[Table[vcs, {vcs, 0.01, 0.2, 0.01}], Table[vcs, {vcs, 0.2, 1.5, 0.02}]]}] ;
```

In[642]:=
```
Export[csvfilename["rmin", ξ], asCsvList[trmin]]
```

Mean free energy

Critical temperature

```
lc_out = {0.5, 1.0, 2.0} ;
lc_out = {0.4, 1.0, 2.0, 3.0} * 0.6022 ; (* in M units *)
ϵmco = 0.1 ;
ξ = 0.05 ;
```

```
ClearAll[rFluctuationByParams]
Options[rFluctuationByParams] =
  {cRange → cRangeEstimation, rRange → rRangeEstimation} ;
rFluctuationByParams[cs_, ξ_, ϵmm_, ϵmcx_, ϵmco_, opts : OptionsPattern[]] /;
   ! OrderedQ[{opts}] :=
  rFluctuationByParams[cs, ξ, ϵmm, ϵmcx, ϵmco, Sequence @@ Sort[{opts}]] ;
rFluctuationByParams[cs_, ξ_, ϵmm_, ϵmcx_, ϵmco_, opts : OptionsPattern[]] :=
 rFluctuationByParams[cs, ξ, ϵmm, ϵmcx, ϵmco, opts] =
  NIntegrate[Simplify[r^4 Exp[-g[r, ncx, nco, cs, ϵmm, ϵmcx, ϵmco]] /.
        {ncx → - (ξ z_mer n_mer + z_co v[r] cco ) / z_cx, nco → v[r] cco}],
      Flatten[{cco, OptionValue[cRange]}], Flatten[{r, OptionValue[rRange]}],
      PrecisionGoal → 4, MaxRecursion → 10,
      Method → {"GlobalAdaptive", Method → "GaussKronrodRule"}] /
     NIntegrate[Simplify[r^2 Exp[-g[r, ncx, nco, cs, ϵmm, ϵmcx, ϵmco]] /.
         {ncx → - (ξ z_mer n_mer + z_co v[r] cco ) / z_cx, nco → v[r] cco}],
       Flatten[{cco, OptionValue[cRange]}], Flatten[{r, OptionValue[rRange]}],
       PrecisionGoal → 4, MaxRecursion → 10,
       Method → {"GlobalAdaptive", Method → "GaussKronrodRule"}]^2 *
    NIntegrate[Simplify[Exp[-g[r, ncx, nco, cs, ϵmm, ϵmcx, ϵmco]] /.
       {ncx → - (ξ z_mer n_mer + z_co v[r] cco ) / z_cx, nco → v[r] cco}],
     Flatten[{cco, OptionValue[cRange]}], Flatten[{r, OptionValue[rRange]}],
     PrecisionGoal → 4, MaxRecursion → 10,
     Method → {"GlobalAdaptive", Method → "GaussKronrodRule"}] - 1
```

```
rRangeEstimationLocal = {1.2, 3 rid[]} ;
rFluctuationByParams[1.0, ξ, 0.15, 0.1, 0.1,
 cRange → {0.06, 3.0}, rRange → rRangeEstimationLocal]
```

```
concTable := Table[{cs, rFluctuationByParams[cs, ξ, ϵmm,
      ϵmcx, ϵmco, cRange → findcRangeSimplified[cs, ξ, ϵmm, ϵmcx, ϵmco],
      rRange → rRangeEstimationLocal]}, {cs, lc_out}] ;
aFluctuations = <|
    (ϵmcx = 0.1) → Table[{ϵmm, Evaluate@concTable}, {ϵmm, 0.05, 0.5, 0.005}] ,
    (ϵmcx = 0.5) → Table[{ϵmm, Evaluate@concTable}, {ϵmm, 0.05, 0.5, 0.005}] ,
    (*(ϵmcx=0.7)→ Table[{ϵmm,Evaluate@concTable},{ϵmm,0.1,0.6,0.05}] ,*)
    (ϵmcx = 0.9) → Table[{ϵmm, Evaluate@concTable}, {ϵmm, 0.05, 0.5, 0.005}]
    |> ;
```

```
csvfilename[name_, ξ_, ϵmc_] :=
 name <> ToString@IntegerPart[100 ξ] <> "-" <> ToString@IntegerPart[10 ϵmc] <> ".csv"
KeyValueMap[
 (ϵmc = #1; Export[csvfilename["fluctuations-", ξ, ϵmc], asCsvList@#2]) &,
 aFluctuations]
```

```
plotOpts := {PlotLabel → StringForm["ξ = ``, ϵ_mc = ``β^-1, ``", ξ, ϵmc, b3label[]],
   PlotLegends → LineLegend[lc_out, LegendLabel → "c_out (nm^-3)"],
   AxesLabel → {"ϵ_mm (β^-1)", "⟨r^4⟩/⟨r^2⟩^2-1"}} ;
KeyValueMap[(ϵmc = #1; Print@ListLinePlot[asPlotList@#2, Evaluate@plotOpts]) &,
 aFluctuations]
```

Isospheric point

```
In[61]:= gMinDiff[ϵcx1_, ϵcx2_, vξ_, vcs_] :=
  N[(r /. gMin[vcs, vξ, ϵmm, ϵcx1, ϵco][[2]]) - (r /. gMin[vcs, vξ, ϵmm, ϵcx2, ϵco][[2]])]
```

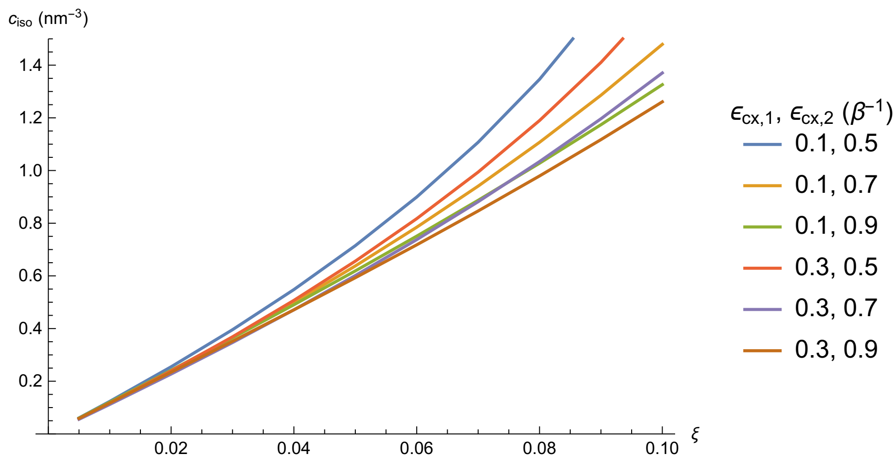
```
In[973]:= ϵmm = 0.4 β^-1 ;
ϵco = 0.1 β^-1;
ϵcxPairs = {{0.1, 0.5}, {0.1, 0.7}, {0.1, 0.9}, {0.3, 0.5},
   {0.3, 0.7}, {0.3, 0.9}, {0.3, 1.0}, {0.3, 1.1}, {0.3, 1.2}, {0.3, 1.3}};
csEstimations = <|0.10 → 1.37, 0.09 → 1.18 , 0.08 → 1.02, 0.07 → 0.88, 0.06 → 0.69,
   0.05 → 0.60, 0.04 → 0.47, 0.03 → 0.35, 0.02 → 0.23, 0.01 → 0.11, 0.005 → 0.06|>;
```

```
In[977]:= tiso =
  Table[{vξ, Table[{ϵcxPair, vcs /. FindRoot[gMinDiff[ϵcxPair[[1]], ϵcxPair[[2]],
        vξ, vcs], {vcs, csEstimations[vξ]}, AccuracyGoal → 3,
       PrecisionGoal → 3, MaxIterations → 5, Evaluated → False]},
     {ϵcxPair, ϵcxPairs}]}, {vξ, Keys@csEstimations}] ;
```

```
tiso0 = Append[t, {0.0, {#1[[1]], 0.0} & /@ tiso[[1]][[2]]}] ;
```

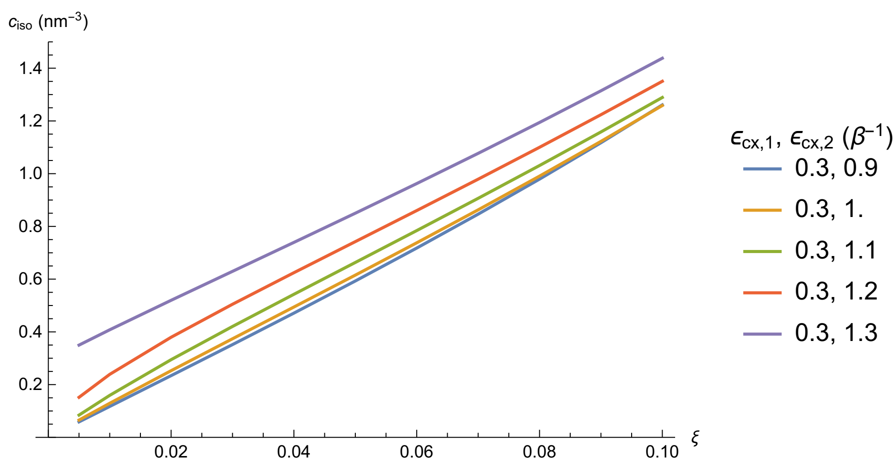
In[573]:=
```
ListLinePlot[asPlotList[tiso][[1 ;; 6]],
 AxesLabel → {"ξ", "c_iso (nm^-3)"}, PlotLegends →
  LineLegend[StringForm["``, ``", #1[[1]], #1[[2]]] & /@ ϵcxPairs[[1 ;; 6]],
   LegendLabel → "ϵ_cx,1, ϵ_cx,2 (β^-1)"], PlotRange → {0., 1.5}, ImageSize → Medium]
```

Out[573]=



In[978]:=
```
ListLinePlot[asPlotList[tiso][[6 ;; -1]],
 AxesLabel → {"ξ", "c_iso (nm^-3)"}, PlotLegends →
  LineLegend[StringForm["``, ``", #1[[1]], #1[[2]]] & /@ ϵcxPairs[[6 ;; -1]],
   LegendLabel → "ϵ_cx,1, ϵ_cx,2 (β^-1)"], PlotRange → {0., 1.5}, ImageSize → Medium]
```

Out[978]=



In[571]:=
```
Export["isospheric.csv", asCsvList[tiso0]];
```

Donnan potential

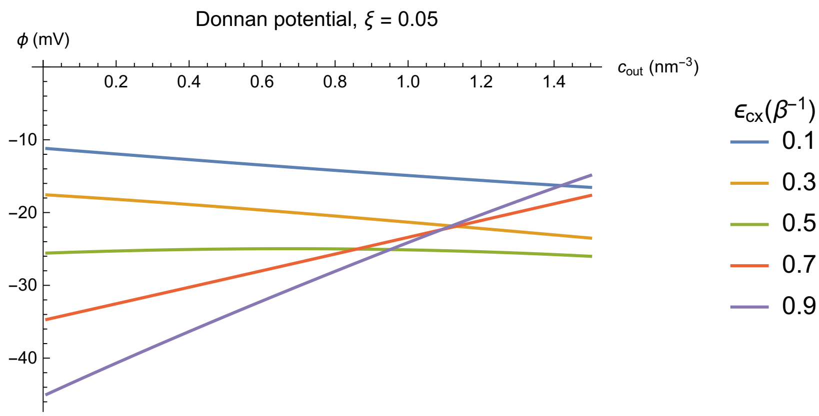
In[936]:=
```
ϵmm = 0.4;
lϵcx = {0.1, 0.3, 0.5, 0.7, 0.9};
ξ = 0.00;
```

In[894]:=
```
ClearAll[μco, μcx]
μco[r_, ncx_, nco_, ϵco_] :=
 μco[r, ncx, nco, ϵco] = 2 b2cc c[r, ncx + nco] + 2 evalB2LJ[σco, ϵco] c[r, nmer] +
   3/2 b3 c[r, nmer + ncx + nco]^2 + 4/3 b4 c[r, nmer + ncx + nco]^3 + Log[c[r, nco]]
μcx[r_, ncx_, nco_, ϵcx_] := μcx[r, ncx, nco, ϵcx] =
  2 b2cc c[r, ncx + nco] + 2 evalB2LJ[σcx, ϵcx] c[r, nmer] +
   3/2 b3 c[r, nmer + ncx + nco]^2 + 4/3 b4 c[r, nmer + ncx + nco]^3 + Log[c[r, ncx]]
donnan[cs_, ξ_, ϵmm_, ϵcx_, ϵco_] :=
 γ 1/2 (μcx[r, ncx, nco, ϵcx] - μco[r, ncx, nco, ϵco]) /. ncx → nco + nmer ξ /.
  gMin[cs, ξ, ϵmm, ϵcx, ϵco][[2]]
donnancx[cs_, ξ_, ϵmm_, ϵcx_, ϵco_] :=
 γ (μcx[r, ncx, nco, ϵcx] - μout[cs]) /. ncx → nco + nmer ξ /.
  gMin[cs, ξ, ϵmm, ϵcx, ϵco][[2]]
```

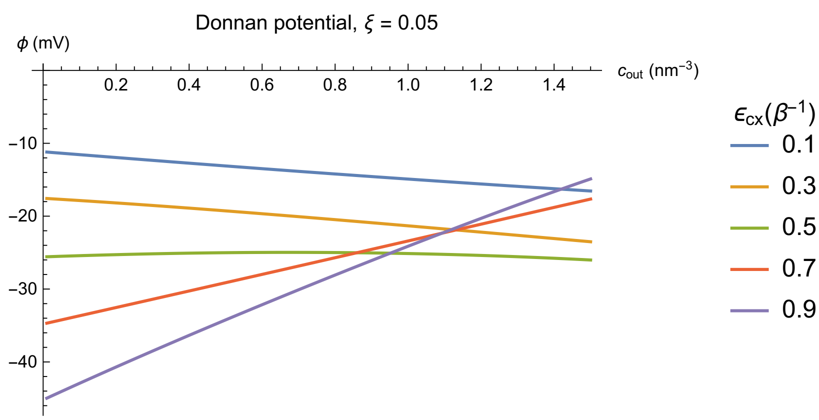
In[939]:=
```
tdonnan = Table[Table[{vcs, 1000 donnan[vcs, ξ, ϵmm, vϵcx, ϵco]},
     {vcs, Union[Table[vcs, {vcs, 0.01, 0.2, 0.01}],
       Table[vcs, {vcs, 0.2, 1.5, 0.02}]]}], {vϵcx, lϵcx}
  ];
tdonnancx =
  Table[Table[{vcs, 1000 donnancx[vcs, ξ, ϵmm, vϵcx, ϵco]}, {vcs, Union[Table[vcs,
        {vcs, 0.01, 0.2, 0.01}], Table[vcs, {vcs, 0.2, 1.5, 0.02}]]}], {vϵcx, lϵcx}
  ];
```

In[979]:=
```
ListLinePlot[tdonnan, PlotLabel → StringForm["Donnan potential, ξ = ``", ξ],
  AxesLabel → {"c_out (nm^-3)", "ϕ (mV)"} ,
  PlotLegends → LineLegend[lϵcx, LegendLabel → "ϵcx (β^-1)"], ImageSize → Medium]
ListLinePlot[tdonnancx, PlotLabel → StringForm["Donnan potential, ξ = ``", ξ],
  AxesLabel → {"c_out (nm^-3)", "ϕ (mV)"} ,
  PlotLegends → LineLegend[lϵcx, LegendLabel → "ϵcx (β^-1)"], ImageSize → Medium]
```

Out[979]=



Out[980]=



In[912]:=
```
asCsvList[l_] :=
  Prepend[NumberForm[#1[[2]], 4] & /@ #1, #1[[1]][[1]]] & /@ Transpose[l]
Export[csvfilename["donnan", ξ], asCsvList[tdonnan]]
```