```
(*code modified from Eduardo Sandoval/Jian Liu*)
```

Dump

Mains()

```
In[ ]:= Quiet@Remove["Global`*"]; Quiet@ClearAll["Global`*"];
```

```
In[ ]:= Clear@interpmembraneProfile;
interpmembraneProfile[ρ_, h_, η_, Δs_, nsteps_, interpolationnumber_] :=
  Block[{index1, index2, ρc, hc, ρnew, hnew, interpolationpts, interpolationFn,
    interpolation, ρinterp, hinterp, interpolationFn2, interpolation2, ρnewf, hnewf},

   If[True,
    (*to extrapolate we first take whatever our extrapolation number is defined as
      interpolation number because when we consider the total vesicle/cell,
     technically we are just interpolating as we are filling in between known points),
    and find the minimum index for either end of the
     contour that corresponds to this value *)
    index1 = First@Flatten@Position[#, Min@#] &@
      Abs[ρ[[IntegerPart[η * nsteps] ;; nsteps + 1]] - interpolationnumber];
    index2 = First@Flatten@Position[#, Min@#] &@
      Abs[ρ[[1 ;; IntegerPart[η * nsteps]]] - interpolationnumber];
    (*we use the index to construct a new
     vector and thus solution that reflects the rest of the solution
     about the y axis and generates a vector to fill in the rest*)
    ρc = ρ[[index2 ;; index1 + η * nsteps]];
    hc = h[[index2 ;; index1 + η * nsteps]];

    (*for first interpolation we use newradius and newheight*)
    ρnew = ConstantArray[0, 2 Length[ρc]];
    ρnew[[1 ;; Length@ρc]] = ρc;
    ρnew[[Length[ρc] + 1 ;;]] = -Reverse[ρc];

    hnew = ConstantArray[0, 2 Length[hc]];
    hnew[[1 ;; Length@hc]] = hc;
    hnew[[Length[hc] + 1 ;;]] = Reverse[hc];

    interpolationpts = Range[-interpolationnumber, interpolationnumber, Δs];

    interpolationFn =
     Interpolation[Thread[{ρnew[[Length[ρnew] / 2 - interpolationnumber ;;
            Length[ρnew] / 2 + interpolationnumber]],
          hnew[[Length[ρnew] / 2 - interpolationnumber ;; Length[ρnew] / 2 +
              interpolationnumber]]}], Method → "Spline", InterpolationOrder → 2];
```

```
    interpolation = Table[interpolationFn[x], {x, interpolationpts}];
    (*for second interpolation we use interpradius and interpheight*)
    ρinterp = ConstantArray[0, 2 Length@ρc];
    ρinterp[[1 ;; Length@ρc]] = Reverse@ρc;
    ρinterp[[Length[ρc] + 1 ;;]] = -ρc;
    hinterp = ConstantArray[0, 2 Length@ρc];
    hinterp[[1 ;; Length@hc]] = Reverse@hc;
    hinterp[[Length[hc] + 1 ;;]] = hc;

    interpolationFn2 =
     Interpolation[Thread[{ρinterp[[Length[ρnew] / 2 - interpolationnumber
          ;; Length[ρnew] / 2 + interpolationnumber]],
        hinterp[[Length[ρnew] / 2 - interpolationnumber ;; Length[ρnew] / 2 +
           interpolationnumber]]}], Method → "Spline", InterpolationOrder → 2];
    interpolation2 = Table[interpolationFn2[x], {x, interpolationpts}];

    (*final vectors with interpolations incorporated*)
    ρnewf = ConstantArray[0, 2 Length[ρc] + 2 Length[interpolationpts]];
    ρnewf[[1 ;; Length[interpolationpts]]] = interpolationpts;
    ρnewf[[Length[interpolationpts] + 1 ;; Length[ρc] + Length[interpolationpts]]] = ρc;
    ρnewf[[Length[ρc] + Length[interpolationpts] + 1
       ;; Length[ρc] + 2 Length[interpolationpts]]] = -interpolationpts;
    ρnewf[[Length[ρc] + 2 Length[interpolationpts] + 1 ;;
       2 Length[ρc] + 2 Length[interpolationpts]]] = -Reverse[ρc];

    hnewf = ConstantArray[0, 2 Length[ρc] + 2 Length[interpolationpts]];
    hnewf[[1 ;; Length[interpolationpts]]] = interpolation2;
    hnewf[[Length[interpolationpts] + 1 ;; Length[ρc] + Length@interpolationpts]] = hc;
    hnewf[[Length[ρc] + Length[interpolationpts] + 1
       ;; Length[ρc] + 2 Length[interpolationpts]]] = interpolation;
    hnewf[[Length[ρc] + 2 Length[interpolationpts] + 1 ;;
       2 Length[ρc] + 2 Length[interpolationpts]]] = Reverse[hc];

    {ρnewf, hnewf}
   ]
  ];
```

In[ ]:=

```
Clear@membraneShapeFn;
membraneShapeFn[ℂ_, nsteps_, η_, δ0_, ξ0_, ρ0_, σ_, κ_, P_, extrap_ : True | False,
   interpolationnumber_] := Block[{θ, δ, ξ, ϱ, ρ, h, Δs, ℂvector, ℋ, ρiter, hiter},
   Δs = ℂ / nsteps;
   ℋ = η (ℂ / Δs);
   ℂvector = Subdivide[ℂ, nsteps];
   θ = δ = ξ = ϱ = ρ = h = ConstantArray[0, Length@ℂvector];
```

```
ρ[[1]] = ρ0;
ρiter = ρ0;
h[[1]] = 0;
hiter = 0.;
θ[[1]] = π / 2;
δ[[1]] = δ0;
ξ[[1]] = ξ0;

ϱ[[1]] = N[-1/2 δ[[1]]^3 - 2 Cos[θ[[1]]]/ρ[[1]] ξ[[1]] + 3/2 Sin[θ[[1]]]/ρ[[1]] δ[[1]]^2 + (3 Cos[θ[[1]]]^2 - 1)/(2 ρ[[1]]^2) δ[[1]] -
    (Cos[θ[[1]]]^2 + 1)/(2 ρ[[1]]^3) Sin[θ[[1]]] + σ[[1]]/κ[[1]] (δ[[1]] + Sin[θ[[1]]]/ρ[[1]]) + P/κ[[1]]];

Do[
 Which[j == 2,
   (*we use a central finite difference method, which results in a modified step
     for the second timestep of the simulation. we assume θ[[i-2]]=θ[[i-1]]*)
   θ[[j]] = θ[[j - 1]] + 2 δ[[j - 1]] Δs;
   δ[[j]] = δ[[j - 1]] + 2 ξ[[j - 1]] Δs;
   ξ[[j]] = ξ[[j - 1]] + 2 ϱ[[j - 1]] Δs;
   (*we update the radius and height for each step,
   this is estimated by using trigonometry and assuming that our
    Δs is small enough that this error will not be far off*)
   ρiter += Δs * Cos[θ[[j]]] // N;
   ρ[[j]] = ρiter;
   hiter += Δs * Sin[θ[[j]]] // N;
   h[[j]] = hiter;
   (*again we have the shape equation*)
   ϱ[[j]] = N[-1/2 δ[[j]]^3 - 2 Cos[θ[[j]]]/ρ[[j]] ξ[[j]] + 3/2 Sin[θ[[j]]]/ρ[[j]] δ[[j]]^2 + (3 Cos[θ[[j]]]^2 - 1)/(2 ρ[[j]]^2) δ[[j]] -
       (Cos[θ[[j]]]^2 + 1)/(2 ρ[[j]]^3) Sin[θ[[j]]] + σ[[1]]/κ[[1]] (δ[[j]] + Sin[θ[[j]]]/ρ[[j]]) + P/κ[[1]]];
   ,

   j ≤ IntegerPart[nsteps / 2] + 1,

   Which[j ≤ IntegerPart[ℋ] + 1,
    θ[[j]] = θ[[j - 2]] + 2 * δ[[j - 1]] * Δs;

    (*this if loop is purely for debugging
     and seeing where a simulation breaks to adjust guesses*)
```

```
If[θ[[j]] > 2 π || θ[[j]] < -2 π,
  (*this part should not run*)
  Print@j;
  δ[[j]] = δ[[j - 2]] + 2 ξ[[j - 1]] Δs;
  ξ[[j]] = ξ[[j - 2]] + 2 ϱ[[j - 1]] Δs;
  ρiter += N@Δs * Cos[θ[[j]]];
  ρ[[j]] = ρiter;
  hiter += N@Δs * Sin[θ[[j]]];
  h[[j]] = hiter;
  ϱ[[j]] = N[-1/2 δ[[j]]^3 - 2 Cos[θ[[j]]]/ρ[[j]] ξ[[j]] + 3/2 Sin[θ[[j]]]/ρ[[j]] δ[[j]]^2 + (3 Cos[θ[[j]]]^2 - 1)/(2 ρ[[j]]^2)
      δ[[j]] - (Cos[θ[[j]]]^2 + 1)/(2 ρ[[j]]^3) Sin[θ[[j]]] + σ[[1]]/κ[[1]] (δ[[j]] + Sin[θ[[j]]]/ρ[[j]]) + P/κ[[1]]];
  Continue[],

  δ[[j]] = δ[[j - 2]] + 2 ξ[[j - 1]] Δs;
  ξ[[j]] = ξ[[j - 2]] + 2 ϱ[[j - 1]] Δs;
  ρiter += N@Δs * Cos[θ[[j]]];
  ρ[[j]] = ρiter;
  hiter += N@Δs * Sin[θ[[j]]];
  h[[j]] = hiter;
  ϱ[[j]] = N[-1/2 δ[[j]]^3 - 2 Cos[θ[[j]]]/ρ[[j]] ξ[[j]] + 3/2 Sin[θ[[j]]]/ρ[[j]] δ[[j]]^2 + (3 Cos[θ[[j]]]^2 - 1)/(2 ρ[[j]]^2)
      δ[[j]] - (Cos[θ[[j]]]^2 + 1)/(2 ρ[[j]]^3) Sin[θ[[j]]] + σ[[1]]/κ[[1]] (δ[[j]] + Sin[θ[[j]]]/ρ[[j]]) + P/κ[[1]]];
],
j == IntegerPart[ℋ] + 2,
(*this part should not run for our case*)
(*we have reached our phase boundary
 or transition. Relevant formulae can be found in Jian Liu's
 paper: endocytic vesicle scission by lipid phase boundary forces*)
θ[[j]] = θ[[j - 2]] + 2 δ[[j - 1]] * Δs;
ρiter += N@Δs * Cos[θ[[j]]];
ρ[[j]] = ρiter;
hiter += N@Δs * Sin[θ[[j]]];
h[[j]] = hiter;
phase1δ = δ[[j - 2]] - 2 ξ[[j - 1]] Δs;
phase1ξ = ξ[[j - 2]] - 2 ϱ[[j - 1]] Δs;
phase2δ = κ[[1]]/κ[[2]] phase1δ - Sin[θ[[j]]] (κ[[2]] - κ[[1]])/(κ[[2]] × ρ[[j]]);
δ[[j]] = phase2δ;
```

```
    phase2ξ = κ[[1]]/κ[[2]] (phase1ξ - (Sin[θ[[j]]] Cos[θ[[j]]])/ρ[[j]]^2 + Cos[θ[[j]]] phase1δ/ρ[[j]]) -
      (Cos[θ[[j]]] δ[[j]])/ρ[[j]] + (Sin[θ[[j]]] Cos[θ[[j]]])/ρ[[j]]^2;
    ξ[[j]] = phase2ξ;
    ϱ[[j]] = N[-1/2 δ[[j]]^3 - 2 Cos[θ[[j]]]/ρ[[j]] ξ[[j]] + 3/2 Sin[θ[[j]]]/ρ[[j]] δ[[j]]^2 + (3 Cos[θ[[j]]]^2 - 1)/(2 ρ[[j]]^2) δ[[j]] -
        (Cos[θ[[j]]]^2 + 1)/(2 ρ[[j]]^3) Sin[θ[[j]]] + σ[[2]]/κ[[2]] (δ[[j]] + Sin[θ[[j]]]/ρ[[j]]) + P/κ[[2]]];
    ,
    True
    ,
    θ[[j]] = θ[[j - 2]] + 2 δ[[j - 1]] Δs;
    δ[[j]] = δ[[j - 2]] + 2 ξ[[j - 1]] Δs;
    ξ[[j]] = ξ[[j - 2]] + 2 ϱ[[j - 1]] Δs;
    ρiter += N@Δs * Cos[θ[[j]]];
    ρ[[j]] = ρiter;
    hiter += N@Δs * Sin[θ[[j]]];
    h[[j]] = hiter;
    ϱ[[j]] = N[-1/2 δ[[j]]^3 - 2 Cos[θ[[j]]]/ρ[[j]] ξ[[j]] + 3/2 Sin[θ[[j]]]/ρ[[j]] δ[[j]]^2 + (3 Cos[θ[[j]]]^2 - 1)/(2 ρ[[j]]^2) δ[[j]] -
        (Cos[θ[[j]]]^2 + 1)/(2 ρ[[j]]^3) Sin[θ[[j]]] + σ[[2]]/κ[[2]] (δ[[j]] + Sin[θ[[j]]]/ρ[[j]]) + P/κ[[2]]];
   ],
   j == IntegerPart[nsteps/2] + 2,
   (*when we have reached halfway through the simulation,
   we have to back calculate the remaining half,
   since we began at half of the contour. We start by initializing a
    new index to make the computation easier, since now we are back
    calculating and will subtract the index by one each step *)
   i = IntegerPart[nsteps / 2];
   (*we then take simulation results so far and
    define them as the second half of the solution i.e. we have a
    1000 length vector and got to 500 and now must backcalculate,
   we redefined the last 500 of the vector to our current result *)
   θ[[i + 1 ;;]] = θ[[1 ;; j - 1]];
   ξ[[i + 1 ;;]] = ξ[[1 ;; j - 1]];
   δ[[i + 1 ;;]] = δ[[1 ;; j - 1]];
   ϱ[[i + 1 ;;]] = ϱ[[1 ;; j - 1]];
   (*the bottom four lines may not be needed and are only used to placate
    paranoia that previously defined values are not refound and maintain
    inaccuracies *)
```

```
θ[[1 ;; i]] = 0;
ϱ[[1 ;; i]] = 0;
ξ[[1 ;; i]] = 0;
δ[[1 ;; i]] = 0;

h[[i + 1 ;;]] = h[[1 ;; j - 1]];
ρ[[i + 1 ;;]] = ρ[[1 ;; j - 1]];
h[[1 ;; i]] = 0;
ρ[[1 ;; i]] = 0;
θ[[i]] = θ[[i + 2]] - 2 δ[[i + 1]] * Δs;
ρiter = ρ0;
ρiter -= N@Δs * Cos[θ[[i]]];
ρ[[i]] = ρiter;
hiter = 0;
hiter -= N@Δs * Sin[θ[[i]]];
h[[i]] = hiter;

If[i == IntegerPart[ℋ] + 2,
 (*this part should not run for our case*)
 phase1δ = δ[[i + 2]] - 2 ξ[[i + 1]] Δs;
 phase1ξ = ξ[[i + 2]] - 2 ϱ[[i + 1]] Δs;
 phase2δ = κ[[1]]/κ[[2]] phase1δ - Sin[θ[[i]]] (κ[[2]] - κ[[1]])/(κ[[2]] × ρ[[i]]);
 δ[[i]] = phase2δ;
 phase2ξ = κ[[1]]/κ[[2]] (phase1ξ - Sin[θ[[i]]] Cos[θ[[i]]]/ρ[[i]]^2 + Cos[θ[[i]]] phase1δ/ρ[[i]]) -
   Cos[θ[[i]]] δ[[i]]/ρ[[i]] + Sin[θ[[i]]] Cos[θ[[i]]]/ρ[[i]]^2;
 ξ[[i]] = phase2ξ;
 (*note that since we are in the second phase, the membrane shape is now using
  the second measurements of membrane tension and binding modulus*)
 ϱ[[i]] = N[-1/2 δ[[i]]^3 - 2 Cos[θ[[i]]]/ρ[[i]] ξ[[i]] + 3/2 Sin[θ[[i]]]/ρ[[i]] δ[[i]]^2 + (3 Cos[θ[[i]]]^2 - 1)/(2 ρ[[i]]^2) δ[[i]] -
    (Cos[θ[[i]]]^2 + 1)/(2 ρ[[i]]^3) Sin[θ[[i]]] + σ[[2]]/κ[[2]] (δ[[i]] + Sin[θ[[i]]]/ρ[[i]]) + P/κ[[2]]];
 ,

 δ[[i]] = δ[[i + 2]] - 2 ξ[[i + 1]] Δs;
 ξ[[i]] = ξ[[i + 2]] - 2 ϱ[[i + 1]] Δs;
 ϱ[[i]] = N[-1/2 δ[[i]]^3 - 2 Cos[θ[[i]]]/ρ[[i]] ξ[[i]] + 3/2 Sin[θ[[i]]]/ρ[[i]] δ[[i]]^2 + (3 Cos[θ[[i]]]^2 - 1)/(2 ρ[[i]]^2) δ[[i]] -
```

```
        (Cos[θ[[i]]]^2 + 1)/(2 ρ[[i]]^3) Sin[θ[[i]]] + σ[[2]]/κ[[2]] (δ[[i]] + Sin[θ[[i]]]/ρ[[i]]) + P/κ[[2]]];
      ],

      True,

      i = i - 1;
      θ[[i]] = θ[[i + 2]] - 2 δ[[i + 1]] Δs;
      If[θ[[i]] > 2 π || θ[[i]] < -2 π,
       Print@j;
       Break[],

       ρiter -= N@Δs * Cos[θ[[i]]];
       hiter -= N@Δs * Sin[θ[[i]]];
       δ[[i]] = δ[[i + 2]] - 2 ξ[[i + 1]] Δs;
       ξ[[i]] = ξ[[i + 2]] - 2 ϱ[[i + 1]] Δs;
       ρ[[i]] = ρiter;
       h[[i]] = hiter;
       ϱ[[i]] = N[-1/2 δ[[i]]^3 - 2 Cos[θ[[i]]]/ρ[[i]] ξ[[i]] + 3/2 Sin[θ[[i]]]/ρ[[i]] * δ[[i]]^2 + (3 Cos[θ[[i]]]^2 - 1)/(2 ρ[[i]]^2)
          δ[[i]] - (Cos[θ[[i]]]^2 + 1)/(2 ρ[[i]]^3) Sin[θ[[i]]] + σ[[2]]/κ[[2]] (δ[[i]] + Sin[θ[[i]]]/ρ[[i]]) + P/κ[[2]]];
      ];
     ];
    , {j, 2, Length[ℂvector]}
   ];
   {ρ, h};
   If[extrap, interpmembraneProfile[ρ, h, η, Δs, nsteps, interpolationnumber]]
  ];
```

In[ ]:=
```
data = Import["C:\\Users\\hashmial\\Downloads\\nuclei
      curvature and dP fit estimates\\dump\\shape.xlsx", "Data"][[1]];
```

In[ ]:=
```
{headings, data} = {First@data, Rest@data};
```

In[ ]:=
```
initialslopelist = data[[All, 1]];
initialcurvaturelist = data[[All, 2]];
sigmalist = data[[All, 3]];
Plist = data[[All, 4]];
baselinesigma = 0.1;
ℂ = 1000 π;
baselineP = -2 baselinesigma / (ℂ / π);
initialradiilist = data[[All, 5]];
extrapolationnumbers = data[[All, 6]];
```

In[ ]:=
```
Clear@membraneShapePhasePlane;
membraneShapePhasePlane[] := Block[{},
    maxradius = minradius = maxheight = minheight = 0;
    ℂ = 1000 π;
    nsteps = 2000;

    η = 1 / 2;
    κ = {16 000, 16 000};
    σ = {1 / 10, 1 / 10};
    P = -2 σ[[1]] / (ℂ / π);
    extrapolation = True;
    interpolationnumber = 10;

    ρ0 = ℂ / π;
    δ0 = 1 / (ℂ / π);
    ξ0 = 0;
    spacing = 0.2;
    magicnum = 3000;
    (* rest of the code comes here*)
    Table[

     σ = {baselinesigma, sigmalist[[l]]};
     δ0 = initialslopelist[[l]];
     ξ0 = initialcurvaturelist[[l]];
     ρ0 = initialradiilist[[l]];
     P = Plist[[l]];
     interpolationnumber = extrapolationnumbers[[l]];
     {radius, height} =
      membraneShapeFn[ℂ, nsteps, η, δ0, ξ0, ρ0, σ, κ, P, True, interpolationnumber];

     (*changeinrad=(sigmalist[[l]]-baselinesigma)/(baselinesigma spacing)*magicnum;

     changeinheight=(P-baselineP)/(baselineP spacing)*magicnum;
```

```
    height+=changeinheight;

    radius+=changeinrad;*)

    If[extrapolation, {sigmalist[[l]] / baselinesigma, P / baselineP,
      (*ListLinePlot[Thread@{radius,height},AspectRatio→{1,1},Axes→False,ImageSize→
        Tiny,MaxPlotPoints→200,InterpolationOrder→1,PlotStyle→RandomColor[]],*)
      Graphics[{FaceForm[Directive[Opacity[0.2], RandomColor[]]],
        EdgeForm[Directive[Black]], FilledCurve[BSplineCurve[
          Thread[{radius, height}], SplineClosed → True]]}, ImageSize → Tiny],
      Graphics[{FaceForm[Directive[RGBColor[0.255, 0.333, 0.902, 0.2]]],
        EdgeForm[Directive[Black]], FilledCurve[BSplineCurve[
          Thread[{radius, height}], SplineClosed → True]]}, ImageSize → Tiny]
     }]
    , {l, Length[Plist]}
   ]

  ];
```

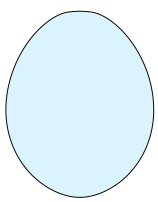
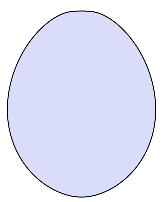
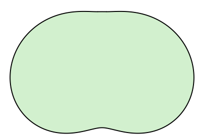
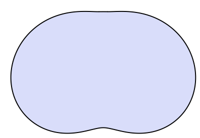
In[ ]:=

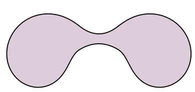
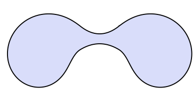
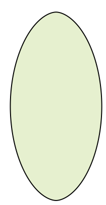
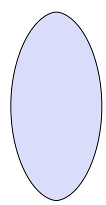
```
res = membraneShapePhasePlane[];
```

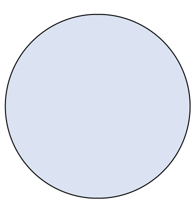
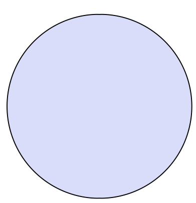
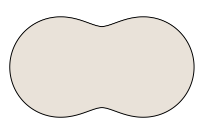
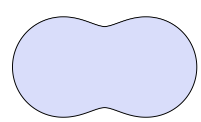
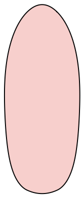
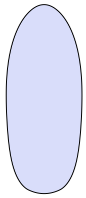
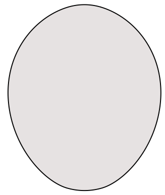
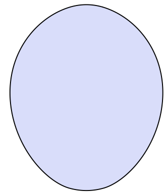
In[ ]:=

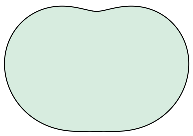
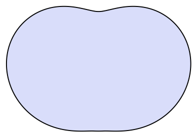
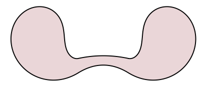
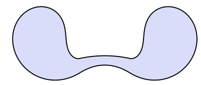
```
res1 = SortBy[res, First]
```

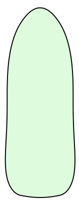
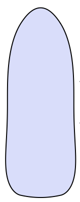
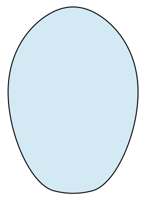
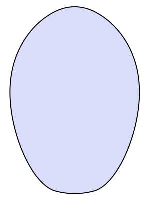
Out[ ]=

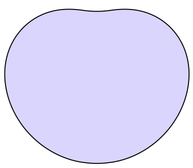
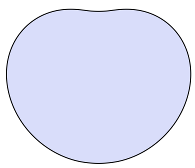
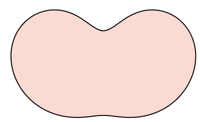
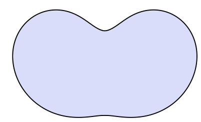
{{0.8, 0.8, , }, {0.8, 1., , },

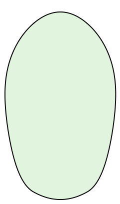
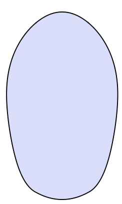
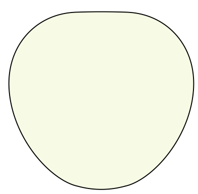
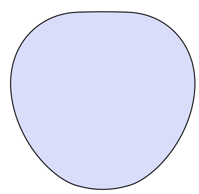
{0.8, 1.2, , }, {1., 0.8, , },

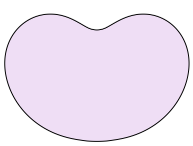
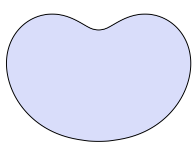
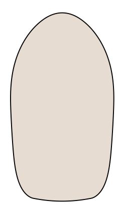
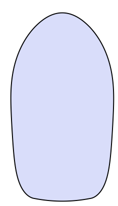
{1., 1., , }, {1., 1.2, , }, {{1.2, 0.8, , }, {1.2, 1., , },

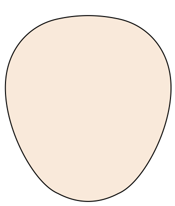
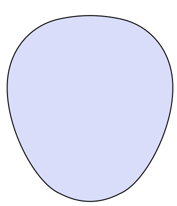
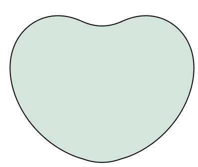
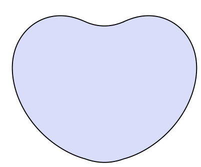
{1.2, 1.2, , }, {1.2, 1.4, , },

{1.4, 0.8, , }, {1.4, 1., , },

{1.4, 1.2, , }, {1.4, 1.4, , },

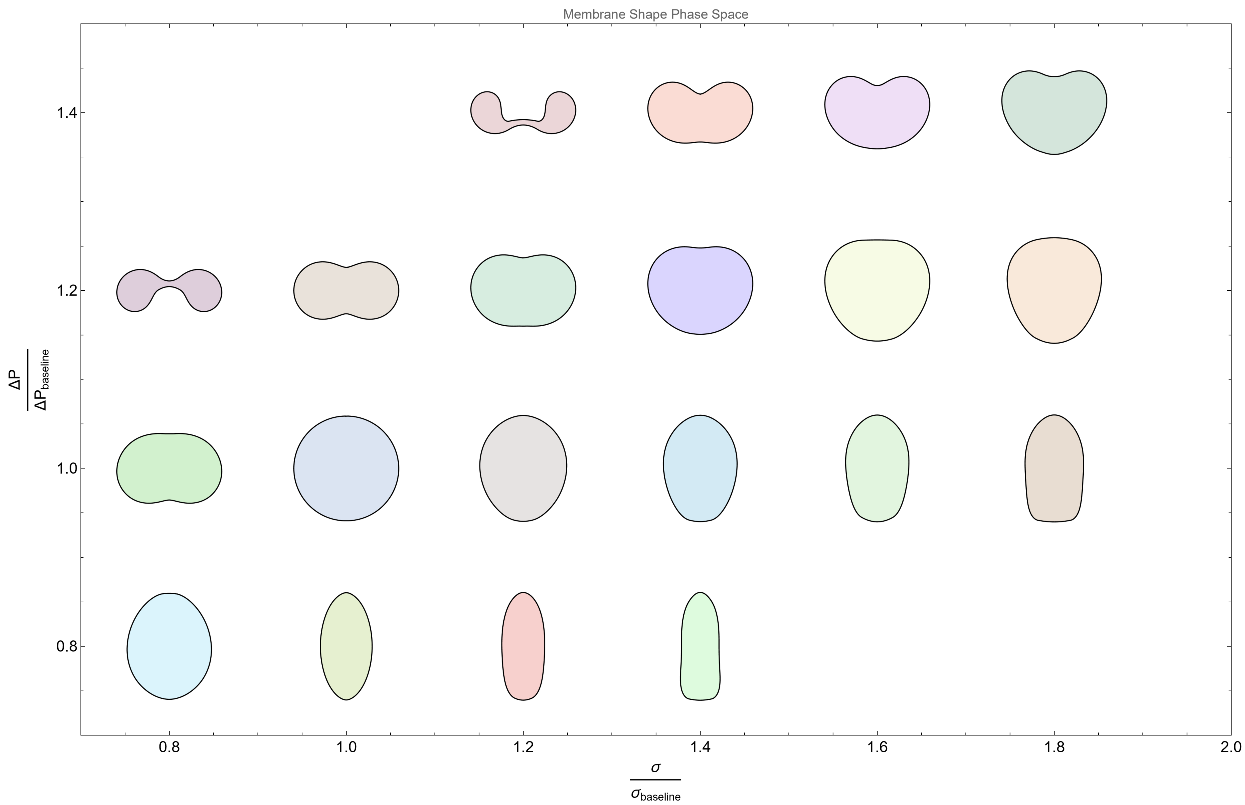
{1.6, 1., , }, {1.6, 1.2, , },

{1.6, 1.4, , }, {1.8, 1., , },

{1.8, 1.2, , }, {1.8, 1.4, , }}

In[ ]:=
```
Rasterize[ListPlot[Thread[{res[[All, 1]], res[[All, 2]]}], PlotStyle → None,
  Epilog → (Inset[#[[3]], {#[[1]], #[[2]]}] & /@ res), ImageSize → Full,
  PlotRange → {{0.7, 2}, {0.7, 1.5}}, PlotLabel → "Membrane Shape Phase Space",
  Frame → True, FrameStyle → Directive[Black, 14], FrameLabel → {"σ/σ_baseline", "ΔP/ΔP_baseline"}]]
```

Out[ ]=



In[ ]:=
```
Rasterize[ListPlot[Thread[{res[[All, 1]], res[[All, 2]]}], PlotStyle → None,
  Epilog → (Inset[#[[-1]], {#[[1]], #[[2]]}] & /@ res), ImageSize → Full,
  PlotRange → {{0.7, 2}, {0.7, 1.5}}, PlotLabel → "Membrane Shape Phase Space",
  Frame → True, FrameStyle → Directive[Black, 14],
  FrameLabel → {"σ/σbaseline", "ΔP/ΔPbaseline"}, FrameTicksStyle → Blue]]
```

Out[ ]=