Example 2 - two variables and two parameters

This is the less simple example: nonlinear model

```
In[•]:= x = 0; ClearAll["Global`*"]; DateList[Date[]] // Most
        SetDirectory[NotebookDirectory[]];

Out[•]= {2022, 3, 17, 16, 23}
```

Initial settings

```
In[•]:= SetOptions[ListPlot, AspectRatio -> Automatic];
        SetOptions[ListPlot,
          AxesLabel → {Style[a, Medium, Bold, Blue], Style[b, Medium, Bold, Blue]}];
        SetOptions[RegionPlot, AspectRatio -> Automatic];
        SetOptions[ListPlot3D, AspectRatio -> 1];
        SetOptions[ContourPlot, AspectRatio -> Automatic];
        SetOptions[ContourPlot,
          AxesLabel → {Style[a, Medium, Bold, Blue], Style[b, Medium, Bold, Blue]}];
        SetOptions[ContourPlot3D, AspectRatio -> 1];
        SeedRandom[Method → "MersenneTwister"];
        SetOptions[FindMaximum, AccuracyGoal -> 5, PrecisionGoal -> 5];
```

Define the confidence set levels for various degrees of freedom

Model specification and data

This example has two parameters but also two variables. The functional form is nonlinear.

Number of data points.

I choose a small number so that one can see each step

```
In[720]:= numdatapoints = 10;
```

Model:

```
In[•]:= params = {a, b};
        numparams = Length[params];
        vars = {x, y};
        model[x_, y_] = a + b x + b^2 y + b^3 x^2;
```

```
In[•]:= Clear[model]
        model[x_, y_] = a + b x +  4 b^2  y + b^3 x^2;
```

True parameters

We choose a degenerate true model

```
In[•]:= truparams = Thread[params -> 0];
        model[x, y] /. truparams

Out[•]= 0
```

Choose the distribution for the errors:

```
In[•]:= dist = NormalDistribution[0, 1];
        RV := RandomVariate[dist, WorkingPrecision → 32]
```

Define the likelihood function (likf) (which is the density) and its log (loglikf)

```
In[•]:= likf = PDF[dist, x]
```

Out[•]= $\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$

```
In[•]:= loglikf = Log[%] // PowerExpand
```

Out[•]= $-\frac{x^2}{2}+\frac{1}{2}(-\mathrm{Log}[2]-\mathrm{Log}[\pi])$

Log likelihood is quadratic when errors are Gaussian

We set the seed so that we can replicate experiments

```
In[•]:= SeedRandom[0];
```

```
In[•]:= SeedRandom[440];
```

Generate data

Choose a random collection of x values, uniform on [0, 1]

```
In[•]:= data = Table[{RandomReal[], RandomReal[]}, {numdatapoints}];
```

I assume that the true value of each parameter is zero.

```
In[736]:= truth = model @@@ data /. truparams;
```

Observations

obs is the vector of observations

```
In[737]:= noise = Table[RV, {numdatapoints}];
          obs = truth + noise;
```

Compute maximum likelihood

I now compute the maximum likelihood estimate assuming that the log likelihood function is quadratic in the errors (noise is Gaussian with known mean and variance). noisehat is the errors assuming that the parameters are (a, b):

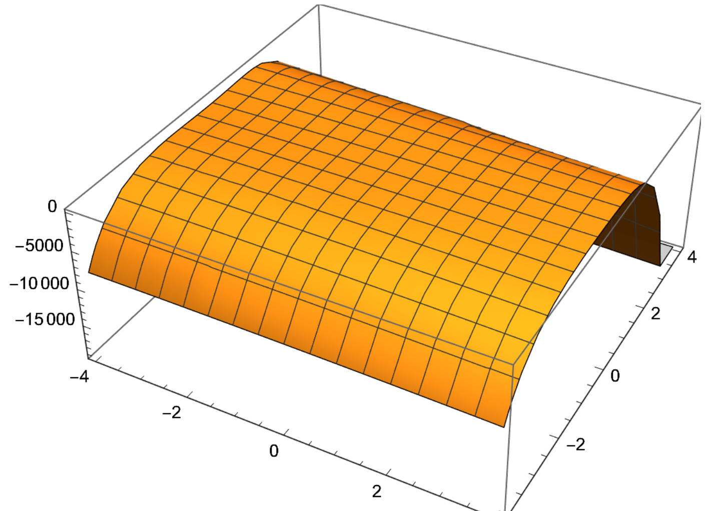
```
In[739]:= noisehat = obs - model @@@ data;

In[740]:= loglik = -noisehat.noisehat // Expand
```

Out[•]= $-10.965+4.92237\,a-10\,a^2+0.993284\,b-11.5484\,a\,b+8.69824\,b^2-38.9595\,a\,b^2-19.6318\,b^3-8.0648\,a\,b^3-54.7329\,b^4-13.8523\,b^5-2.36068\,b^6$

```
In[•]:= loglik = loglik /. zz_Real :> SetPrecision[zz, 32] // N
```

Out[•]= $-10.965+4.92237\,a-10.\,a^2+0.993284\,b-11.5484\,a\,b+8.69824\,b^2-38.9595\,a\,b^2-19.6318\,b^3-8.0648\,a\,b^3-54.7329\,b^4-13.8523\,b^5-2.36068\,b^6$

```
In[•]:= likfcn[a_, b_] = loglik;  Plot3D[loglik, {a, -4, 4}, {b, -4, 4}]
```

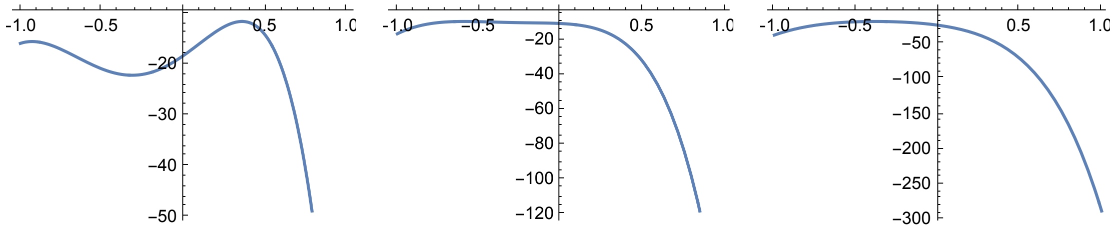
Out[•]=


Find and record the maximum likelihood estimate:

```
In[•]:= estimate = FindMaximum[loglik, {a, b}, WorkingPrecision → 32] // N; // Quiet
        maxlik = estimate[[1]]
        maxpt = {a, b} /. estimate[[2]]

Out[•]= -9.61746

Out[•]= {0.191426, -0.399729}
```

NOT quadratic!!

```
In[•]:= slices = Table[Plot[likfcn[maxpt[[1]] + a, maxpt[[1]] + b], {b, -1, 1}], {a, -1, 1}] //
          GraphicsRow

Out[•]=
```



Plot the estimate for future graphs

```
In[•]:= plotmaxpt =
          ListPlot[{{maxpt}, {{0, 0}}}, PlotStyle -> {Black}, PlotMarkers → {"*", "O"}];
```

Two-parameter confidence sets

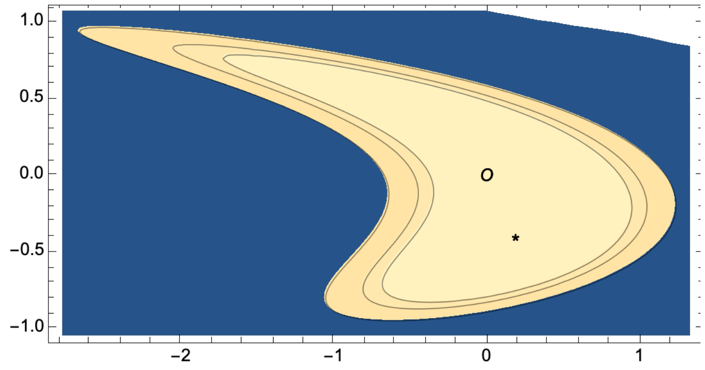
Compute ranges

Contour Plots

We plot contours of the likelihood corresponding to the 90%, 95%, and 99% confidence levels. They are centered on the estimate (maxpt).

In[∘]:=
```
contours = ContourPlot[loglik, {a, amin, amax},
    {b, bmin, bmax}, Contours -> levels, AxesLabel -> {a, b}];
contourplot = Show[contours, plotmaxpt]
```

Out[∘]=



Grid evaluation

I compute the likelihood function (normalized by the number of data points) on a grid of points.
I then choose those where the likelihood function exceeds some critical value (which here I set to be maxlik - 1/2).
I then plot that 2D set of points, which turns out to be an ellipse

In[∘]:=
```
SetOptions[ListPlot, PlotRange -> {{amin, amax}, {bmin, bmax}}];
SetOptions[ListPlot, PlotStyle -> Red];
SetOptions[ListPlot, AspectRatio -> Automatic];
SetOptions[RegionPlot, AspectRatio -> Automatic];
SetOptions[RegionPlot, PlotRange -> {{amin, amax}, {bmin, bmax}}];
```

Construct grid:

```
In[•]:= (* step is the distance between consecutive points *)
astep = (amax - amin) / 50;
bstep = (bmax - bmin) / 50;

(* define a grid of width 2 x range *)
grida = Range[amin, amax, astep];
gridb = Range[bmin, bmax, bstep];
(* Construct the tensor product of the grids *)
gridpts = Flatten[Outer[List, grida, gridb], 1];
(* Evaluate the likelihood function at each grid point*)
gridvals = Map[likfcn @@ # &, gridpts];
(* Save each grid point along with its likelihood value*)
griddata = {gridpts, gridvals} // Transpose;
(* Each entry in griddata is a grid point along with its likelihood value *)
griddata[[1]]
```

```
Out[•]= {{-2.77076, -1.04624}, -62.6283}
```

Choose the points in gridpts that have likelihood greater than level90 (level95, level99)

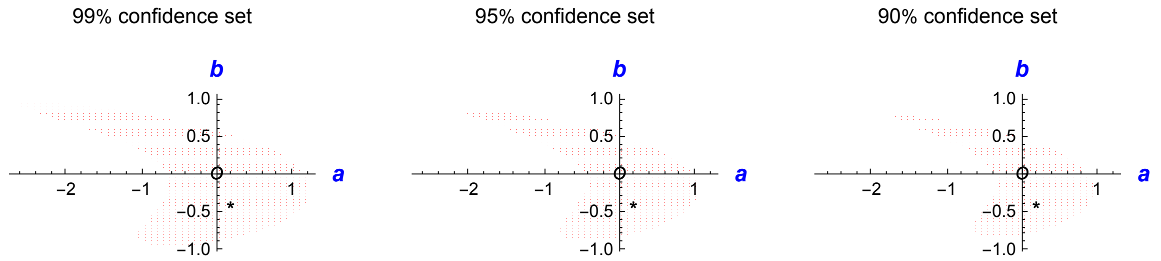
```
In[•]:= data99 = First /@ (Select[griddata, #[[2]] > level99 &]);
        mesh99 = Show[ListPlot[data99, PlotLabel → "99% confidence set"], plotmaxpt];

In[•]:= data95 = First /@ (Select[griddata, #[[2]] > level95 &]);
        mesh95 = Show[ListPlot[data95, PlotLabel → "95% confidence set"], plotmaxpt];

In[•]:= data90 = First /@ (Select[griddata, #[[2]] > level90 &]);
        mesh90 = Show[ListPlot[data90, PlotLabel → "90% confidence set"], plotmaxpt];
```

Display

I compute the likelihood function on a grid of points.
I choose those points where the likelihood function exceeds some critical value, and plot that 2D set of points.

In[•]:=
```
gridplot = GraphicsRow[{mesh99, mesh95, mesh90}, ImageSize -> Full]
```

Out[•]=



Dimension-by-dimension solutions for boundary points

Find values for (a, b) such that likfcn[a, b] = level.
We construct 1D equations: first fix b and solve for a, and second fix a and solve for b.
The command NSolve finds all solutions (possible here because the function is a polynomial), and then we keep only the real solutions. In these examples, we only got two solutions but approach will work even if there are multiple real solutions.
We then merge the two sets of boundary points.

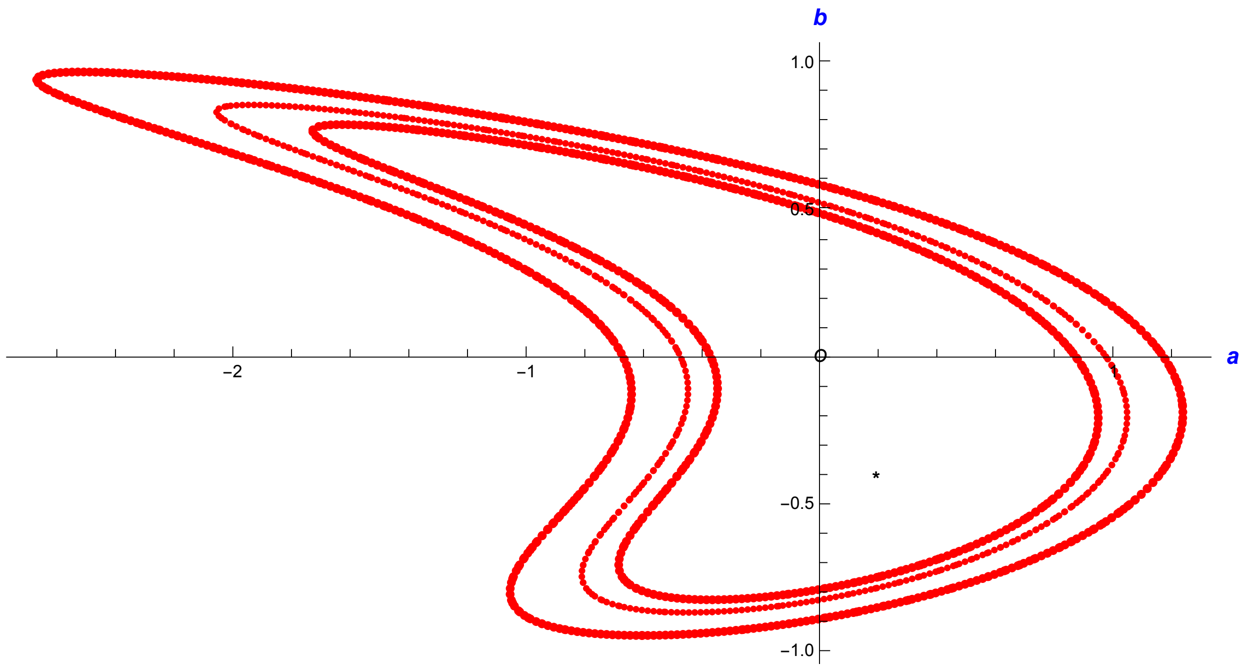
```
In[•]:= step = 2 / 100;

In[•]:= Bndry :=
         ((* Choose b from {-range, range, step} and solve
           for a such that the loglikelihood is level90 *)tab = Table[
            Outer[List, a /. NSolve[likfcn[a, b] == level, a], {b}], {b, bmin, bmax, step}];
        list1 = Flatten[tab, 2];
        list1 = Select[list1, Element[#[[1]], Reals] && Element[#[[2]], Reals] &];
        plot1 = ListPlot[list1];
        (* Choose a from {-range, range, step} and solve
           for b such that the loglikelihood is level90 *)tab = Table[
            Outer[List, {a}, b /. NSolve[likfcn[a, b] == level, b]], {a, amin, amax, step}];
        list2 = Flatten[tab, 2];
        list2 = Select[list2, Element[#[[1]], Reals] && Element[#[[2]], Reals] &];
        plot2 = ListPlot[list2];
        (* Combine the two lists of points and plot*)
          list = Union[list1, list2];
          ListPlot[list])

In[•]:= level = level90;  bndry90 = Bndry;

In[•]:= level = level95;  bndry95 = Bndry;
```

Display

Find values for (a, b) such that likfcn[a, b] = level.

```
In[∘]:= crossplot = Show[bndry90, bndry95, bndry99, plotmaxpt, ImageSize -> Full]
```

Out[∘]=



Rays centered at the estimate

In this case, we find boundary values along rays originating at the ML. In this case, we define the rays to be uniformly distributed on a circle.

Real computations

We now do this with more rays.

```
In[•]:= rays = Table[maxpt + {Sin[θ], Cos[θ]} / 2, {θ, -Pi , Pi, Pi / 64}] // Union;
        listrays = ListPlot[rays, PlotStyle → Blue];
```

```
In[•]:= ptλ := λ maxpt + (1 - λ) ray
```

```
In[•]:= tab = Table[
            λs = λ /. NSolve[likfcn @@ ptλ == level90, λ];
            pts = Table[λs[[j]] maxpt + (1 - λs[[j]]) ray, {j, λs // Length}],
            {ray, rays}];
        list = Flatten[tab, 1];
        list = Cases[list, {__Real}];
        ray90 = ListPlot[list];
```

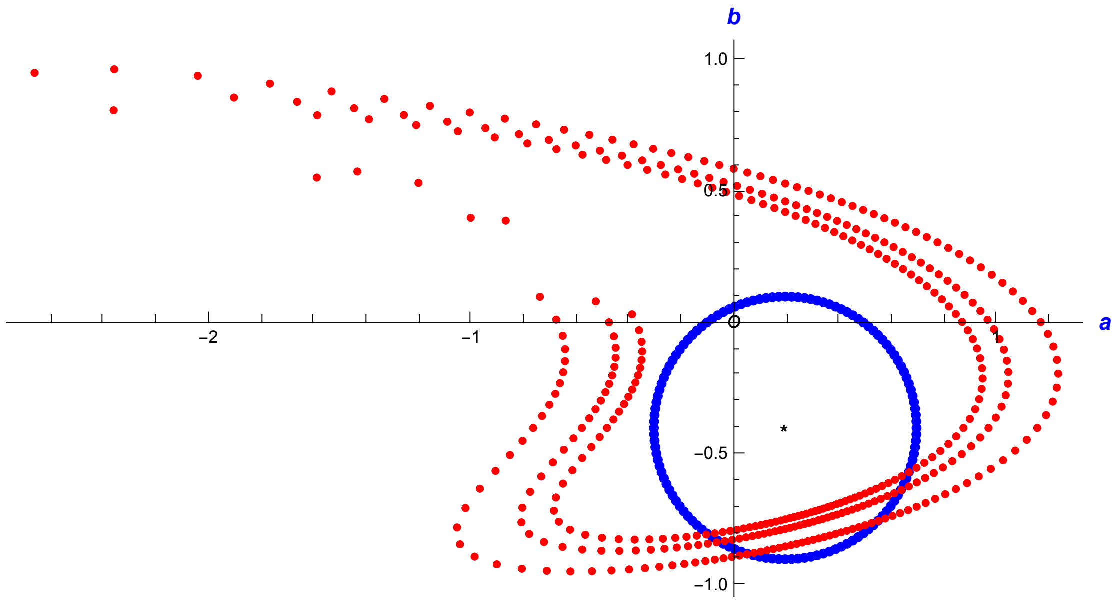
```
In[•]:= tab = Table[
            λs = λ /. NSolve[likfcn @@ ptλ == level95, λ];
            pts = Table[λs[[j]] maxpt + (1 - λs[[j]]) ray, {j, λs // Length}],
            {ray, rays}];
        list = Flatten[tab, 1];
        list = Cases[list, {__Real}];
        ray95 = ListPlot[list];
```

```
In[•]:= tab = Table[
            λs = λ /. NSolve[likfcn @@ ptλ == level99, λ];
```

Display

In[∘]:=
```
rayplots = Show[listrays, ray99, ray95, ray90, plotmaxpt, ImageSize → Large]
```

Out[∘]=



One-parameter confidence sets

Optimization approach

Let’s compute the confidence intervals for a and then b.

Choose levels

```
In[•]:= levels = {level90, level95, level99} = maxlik - chisquare[[1]]

Out[•]= {-12.3235, -13.4585, -16.2525}
```

We first take the projections of the 2D sets to get bounds on the intervals

```
In[•]:= listT = list // Transpose;
        {amin, amax} = Rationalize[{Min[listT[[1]]], Max[listT[[1]]]}, 10^-32];
        {bmin, bmax} = Rationalize[{Min[listT[[2]]], Max[listT[[2]]]}, 10^-32];
        {amin, amax} // N
        {bmin, bmax} // N

Out[•]= {-2.66665, 1.2309}

Out[•]= {-0.946179, 0.965794}
```

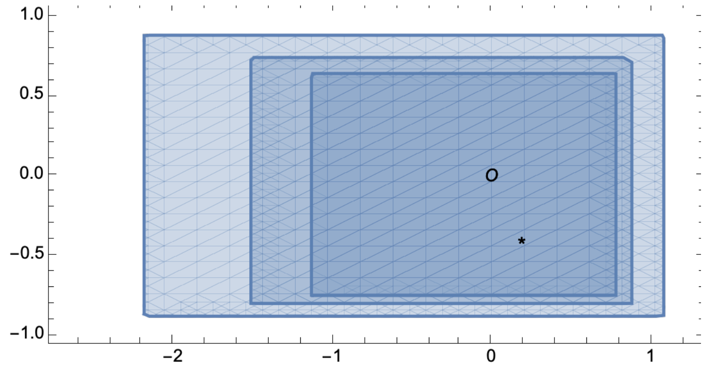
We choose a grid of a values in the interval [amin, amax]. For each a, we find the b which maximizes the likelihood function.
We keep those points which produce likelihoods in excess of the Chi-square level.
This produces the confidence level for a.

```
In[•]:= step = 1. / 100;

In[•]:= tab = Table[
            {a, FindMaximum[loglik, {b}, Method -> "Newton"][[1]]}, {a, amin, amax, step}];
        Select[tab, #[[2]] > level90 &] // Transpose // First;
```

```
In[•]:= tab = Table[
           {a, FindMaximum[loglik, {b}, Method -> "Newton"][[1]]}, {a, amin, amax, step}];
        Select[tab, #[[2]] > level95 &] // Transpose // First;
        {aleft, aright} = {Min[%], Max[%]}
        tab = Table[
           {b, FindMaximum[loglik, {a}, Method → "Newton"][[1]]}, {b, bmin, bmax, step}];
        Select[tab, #[[2]] > level95 &] // Transpose // First;
        {bleft, bright} = {Min[%], Max[%]}
        CS1D95 = RegionPlot[
           aleft <= a <= aright && bleft <= b <= bright, {a, amin, amax}, {b, bmin, bmax}];

Out[•]= {-1.51665, 0.873352}

Out[•]= {-0.796179, 0.743821}

In[•]:= tab = Table[
           {a, FindMaximum[loglik, {b}, Method -> "Newton"][[1]]}, {a, amin, amax, 0.01}];
        Select[tab, #[[2]] > level99 &] // Transpose // First;
        {aleft, aright} = {Min[%], Max[%]}
        tab = Table[
           {b, FindMaximum[loglik, {a}, Method → "Newton"][[1]]}, {b, bmin, bmax, 0.01}];
        Select[tab, #[[2]] > level99 &] // Transpose // First;
        {bleft, bright} = {Min[%], Max[%]}
        CS1D99 = RegionPlot[
           aleft <= a <= aright && bleft <= b <= bright, {a, amin, amax}, {b, bmin, bmax}];

Out[•]= {-2.18665, 1.07335}

Out[•]= {-0.876179, 0.883821}
```

```
In[•]:= Show[CS1D90, CS1D95, CS1D99, plotmaxpt]
```

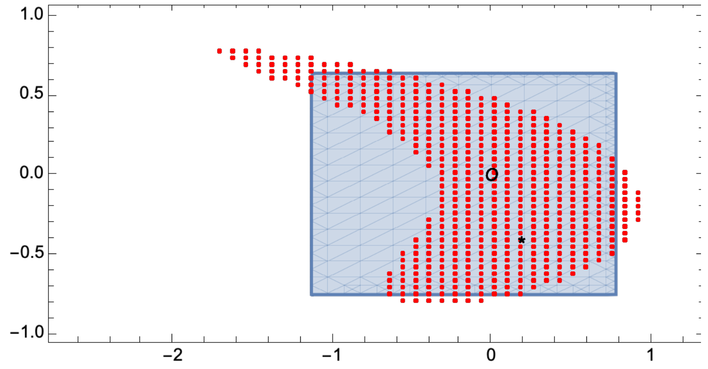
Out[•]=



Comparison

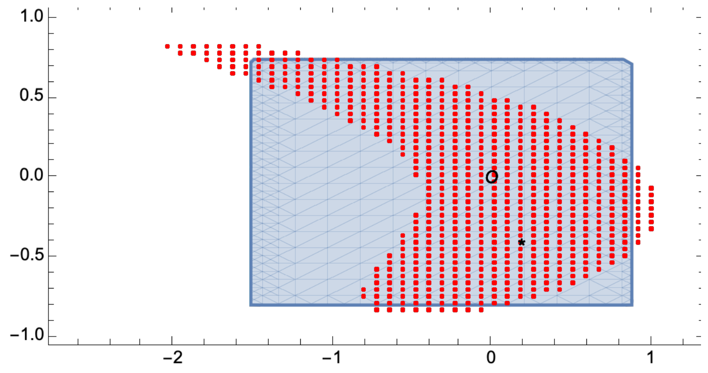
```
In[•]:= Show[CS1D90, mesh90, plotmaxpt]
```

Out[•]=



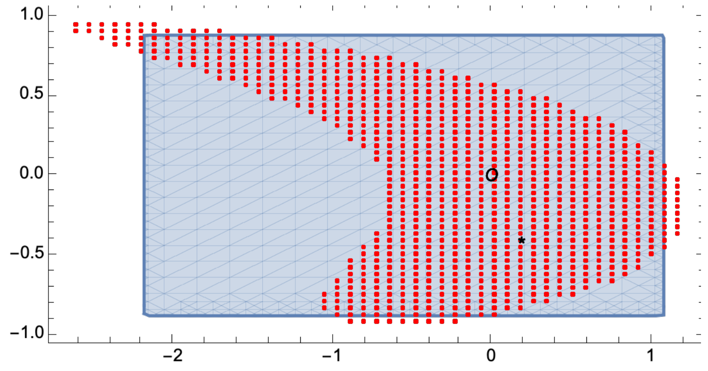
In[∘]:=
```
Show[CS1D95, mesh95, plotmaxpt]
```

Out[∘]=



In[∘]:=
```
Show[CS1D99, mesh99, plotmaxpt]
```

Out[∘]=


1.0
0.5
0.0
−0.5
−1.0
−2
−1
0
1