39. Show that (32) is still true even if the (positive) masses of the particles are not all equal.

(32) The centroid Z must lie in the interior of the convex hull H.

If the mass centroid were in the exterior of the convex hull, we may draw a line passing through the centroid so that all the particles lie to one side of the line and do not touch the line. Since all the vectors from a point on the line to the particles have a component normal to the line, and they all point in the same direction with respect to the line, the masses could obviously not cancel. If the mass centroid were a point on the convex hull (i.e. boundary), we could draw a line through the centroid so that all the particles lie on or to one side of the line. All the vectors of particles off the line have a component normal to the line and they all point in the same direction with respect to the line. It is not possible for the masses to cancel. Hence, the mass centroid must be interior to the hull.

The demo program scatters 10 particles as circles with areas scaled by randomly selected relative mass. It draws a line outside the point farthest from the calculated centroid (centered by red concentric circles) and orthogonal to the line that runs through the centroid and the farthest point.

The demo also draws a convex hull as a polygon passing through the center of each particle (black circle) that defines the hull. All the randomly located particles are given a location (z - c) relative to the mass centroid. Choose the most distant particle from the mass centroid as an initial pivot and hull point. Rotate the pivot point about the centroid to the angle Pi. Construct imaginary lines from the pivot point to every other particle and calculate their angles using the Arg[z] function of Mathematica, which returns real numbers in the range (-Pi, Pi]. Choose the particle of the line with the smallest angle, which must always be negative given the recentering on the centroid, unless all the particles happen to fall on the real line. This particle becomes a hull point. Then rotate this particle clockwise to -Pi and set the pivot index to point to it. Using the new pivot, repeat the search for the next hull particle. Stop when the smallest angle points to the original pivot particle.

```
In[239]:= Clear["Global`*"

(* GLOBALS *)

extensor = 1.25; (* factor applied to line to farthest point *)
pr = 1; (* plot range *)

(* FUNCTIONS *)

v[z_] := {Re[z], Im[z]}; (* complex to vector *)
SetAttributes[{v}, Listable];

```
(* maxDtoCentroid[c, zList] returns \{z, Max[|z-Z|]\}, which is the point at the maximum distance from the centroid, and its distance from the centroid. *)

maxDtoCentroid[c_, zList_] :=
(* zList = \{pos, z, mass\} *) (* function debugged *)

Module[{zDistList, distOnly, maxD, i},
    zDistList = \{#[[1]], #[[2]], Abs[#[[2]] - c]\} & /@ zList;
    (* list of \{pos, z, |z-cZ|\} *)
    distOnly = #[[3]] & /@ zDistList;
    maxD = Max[distOnly];
    i = Position[distOnly, maxD] // Flatten; (* \{i, 3\} *)
    i = i[[1]]
];

(* rsL returns random list of complex numbers, Re and Im [-.5 .. .4] by .1; the point locations of the particles *)

rsL := Module[{rsrl, rsim},
    rsrl = RandomSample @ Range[-.5, .4, .1];
    rsim = RandomSample @ Range[-.5, .4, .1];
    Table[rsrl[[i]] + rsim[[i]] I, {i, 1, Length @ rsrl}]
];

rsM := RandomSample @ Range[.1, 1, .1]; (* the masses of the particles *)

(* newSet is basically a spreadsheet for essential data *)

newSet := Module[{zList, massList, sample, centroidList, cz, maxIndex, p = "obsolete", recentrdList, recentrdMaxZ, recentrdP},
    zList = rsL; (* \{z_1, z_2, \ldots\} *)
    massList = rsM // Flatten; (* \{m_1, m_2, \ldots\} *)
    sample = Table[{i, zList[[i]], massList[[i]]}, {i, 1, Length[zList]}];
    (* index, z, mass *)
    centroidList = Partition[Riffle[zList, massList], 2];
    (* \{\{z_1, m_1\}, \{z_2, m_2\}, \ldots\} *)
cZM = Total[(Sqrt[#1]/Pi) & /@ rsM]; (* Radii for masses of centroid, where mass is represented as area, r = Sqrt(m/Pi). *)

cZ = centroidZ @ centroidList; (* centroid *)

maxIndex = maxDtoCentroid[cZ, sample]; (* returns list index of point farthest from centroid *)

recentrList = {#[[1]], #[[2]] - cZ, #[[3]]} &/@ sample; (* {{index, z-cZ, mass}...} *)

(* record of pt at max distance from centroid after recentering *)
recentrMaxZ = recentrList[[maxIndex]];
(* returns {index, z-cZ, max|z-Z|} *)

(* point that will be anchor for line external to hull *)
recentrP = extensor recentrMaxZ[[2]];

{zList, massList, sample, cZ, maxIndex, p, recentrList, recentrMaxZ, recentrP, cZM}]

(* hull[newSet] returns the graphics primitives for the arrows of the hull *)

hull[newSet_] :=

Module[

ns = newSet, (* copies data from external call to newSet *)

pivotZ = newSet[[7, 5, 2]], (* z-cZ from (pos, z-cZ, max|z|) *)

pivotIndex = newSet[[5]], (* newSet.maxIndex *)

hullData = {},

rotationArg = 0,

rotatedZList = newSet[[7]], (* {{index, z-cZ, mass}...} *)

oldCentrZList = newSet[[7]],

frmPivotArgs = {},

minArg = 0,

hullIndices = {},

hullIndex = 0,

hullRec = 0,

done = False),
While[! done, 

pivotZ = rotatedZList[[pivotIndex, 2]]; (* pivotIndex changes each cycle *)

(* find an argument that will rotate the pivot to pi or -pi *)
rotationArg = (Sign[Arg @ pivotZ] Pi) - Arg @ pivotZ;
(* difference from -Pi or Pi *)

(* rotate all z's in the list of centered points by same amount as pivot was rotated *)
(* this should rotate the original pivot as well *)
rotatedZList = #[[1]], # [[2]] Exp[I rotationArg], #[[3]] & /@ rotatedZList;
(* [pos, ze^θ, |z-Z|] *)

(* get the new value of the pivot; it should be a negative real value *)
pivotZ = rotatedZList[[pivotIndex, 2]]; 

(* draw lines from pivot to each point and find the argument for each line *)
(* on the last pivot, this should also find the argument of the original pivot. *)
frmPivotArgs = Arg[#[[2]] - pivotZ] & /@ rotatedZList;

(* make sure the pivot isn't selected as the minimum arg in next step *)
frmPivotArgs[[pivotIndex]] = Max @ frmPivotArgs;

(* the point with the minimum argument wrt the pivot will be the next point on the convex hull *)
minArg = Min @ frmPivotArgs;

(* find the index number of the minimum argument *)
hullIndex = (Position[frmPivotArgs, minArg] // Flatten)[[1]];

(* reset the pivotIndex *)
pivotIndex = hullIndex;

If[hullIndex == ns[[5]], done = True];

hullIndices = Append[hullIndices, hullIndex];
]
(* END WHILE *)

hullIndices = Prepend[hullIndices, pivotIndex];

hullData = oldCentrdZList[[#]] & /@ hullIndices;
(* recentrdList[[#]] = {pos, z, m } *)

(* r = Rasterize[Length[hullData]]; *)
Table[Arrow[{v @ hullData[[i, 2]], v @ hullData[[i + 1, 2]]}],
{i, 1, Length[hullData] - 1}]
(* Line[v[#[[2]]]&/@hullData], (* draws line around hull *)

(* centroid.hull.nb *)
BezierCurve[v[[2]] @ hullData]) (* draws Bezier curve with hull points as control pts *)
]

(* GRAPHICS PRIMITIVES *)
(* draw particles and centroid *)
massFactor = 10;
(* dot accepts a complex number z and draws a circle centered at z *)
dot[z_, r_] := Circle[v[z], r];

(* dotCZ[] returns the centroid circles *)
dotCZ[cZ_, cZM_, rsm_] := Module[{},
  {Red, Circle[v[cZ], #]} & /@ Range[.005, cZM/massFactor, cZM/(massFactor*5)]];

(* dots[list_] draws all the particles *)
dots[list_] := dot[#, 2 Sqrt[#3]/(Pi massFactor)] & /@ list;
(* recentrList newSet[[7]] = {pos, z, mass} *)

(* allDots[] passes data from newSet to dotCZ[] and dots[] *)
allDots[newSet_] := {dotCZ[ns[[4]], ns[[10]], ns[[2]]], dots @ newSet[[7]]};
(* recentrList newSet[[7]] = {pos, z, mass} *)

(* lines and arrows *)
externalLine[newSet_] := Line[{v @ 0, v @ newSet[[9]]}]; (* the line "L" *)
orthoLn[newSet_] :=
  Rotate[externalLine @ newSet, #, v @ newSet[[9]]] & /@ (Pi/2, -Pi/2);
(* the line orthogonal to L *)

(* pivot vectors uses data from newSet to draw the thin, gray arrows from the pivot to the particles *)
pivotVectors[newSet_] := {Thin, Gray, Arrowheads[.02],
  Arrow[{v @ newSet[[9]], v @ #}]} & /@ newSet[[7]];

(* allLns passes data from newSet to orthoLn[] and pivotVectors[] *)
allLns[newSet_] := {orthoLn[newSet], pivotVectors[newSet]};
(* TEXT *)

centroidTxt[newSet_] := Module[{ns},
   ns = newSet;
   Text[Style["Z", Bold],
   v[
   ns[[4]] - Sign @ Re[ns[[9]]] (0.03 + 0.03 I) (* cZ - Sign(extCZPt) *)
   ]]
];

lineTxt[newSet_] := Module[{ns},
   ns = newSet;
   Text[Style["L", Bold],
   v @ ns[[9]]
   + 0.1 v @ Sign[Re[ns[[9]]]] (* a small vector *)
   ]]
];

allTxt[newSet_] := {lineTxt @ newSet, centroidTxt @ newSet};

(* MANIPULATE *)

Manipulate[
   n = False;
   ns = newSet; (* call this only once per manipulate cycle *)
   data = {allDots @ ns, allLns @ ns, hull @ ns, allTxt @ ns};
   Graphics[data, Background -> White,
   PlotRange -> {{-pr, pr}, {-pr, pr}}, ImageSize -> Medium],
   Button["Get new particles", n = True],
   TrackedSymbols -> {n}, SaveDefinitions -> True
]