Trayectorias experimento

Gerardo Miguel Lucario

1.- Importar los txt de las trayectorias

MESRM

```
MESRMVertical = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
MESRMHorizontal = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
MESRMCirculo = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
MESRMInfinito = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
MESRMTodas = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

FJLH

```
FJLHCirculo = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
FJLHHorizontal = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
FJLHInfinito = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
FJLHVertical = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
FJLHTodas = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

AOGM

```
AOGMCirculo = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
AOGMVertical = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
AOGMHorizontal = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
AOGMInfinito = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

```
AOGMTodas = Import["PathCoordenadasTrayectoria", "Table"];
```
importa tabla

2.- Limpiar datos y asignarlos a pacientes

Función limpiar data y girarlo

```
In[•]:= filterData[dat_, angleRot_] := Module[
                                       módulo
        {usefulData = 0, matRotZ = 0, thisData = Table[dat[[i]], {i, 3, Length[dat]}]},
                                                  tabla                longitud
        (*Linea de código obtenida de: https://
         stackoverflow.com/questions/18391937/deleting-empty-values-in-a-mathematica-table*)
        thisData = Select[thisData, UnsameQ[#, {}] &];
                   selecciona       ¿diferentes?
        usefulData = Table[0, {i, Length[thisData]}];
                     tabla          longitud
        matRotZ = {{Cos[angleRot], -Sin[angleRot]}, {Sin[angleRot], Cos[angleRot]}};
                  coseno           seno             seno            coseno
        Do[
        repite
         usefulData[[i]] = matRotZ.thisData[[i, {2, 3}]];
         , {i, 1, Length[thisData]}];
                  longitud

        usefulData
       ]
```

Asignando data a los pacientes

- MESRM

```
In[•]:= MESRM = <|
          circular → filterData[MESRMCirculo, 180 °],
          vertical -> filterData[MESRMVertical, 180 °],
          horizontal -> filterData[MESRMHorizontal, 180 °],
          infinito -> filterData[MESRMInfinito, 180 °],
          todas -> filterData[MESRMTodas, 180 °],
          m1 → 65,
          m2 → 28,
          m3 → 16,
          brazo -> "izquierdo"
          |>;
```

- FJLH

```
In[•]:= FJLH = <|
          circular → filterData[FJLHCirculo, 180 °],
          vertical → filterData[FJLHVertical, 180 °],
          horizontal → filterData[FJLHHorizontal, 180 °],
          infinito → filterData[FJLHInfinito, 180 °],
          todas → filterData[FJLHTodas, 180 °],
          m1 → 66,
          m2 → 37,
          m3 → 16,
          brazo -> "izquierdo"
          |>;
```

- AOGM

```
In[•]:= AOGM = <|
          circular → filterData[AOGMCirculo, 180 °],
          vertical → filterData[AOGMVertical, 180 °],
          horizontal → filterData[AOGMHorizontal, 180 °],
          infinito → filterData[AOGMInfinito, 180 °],
          todas → filterData[AOGMTodas, 180 °],
          m1 → 67,
          m2 → 36,
          m3 → 15,
          brazo -> "derecho"
          |>;
```

3.- Funciones auxiliares

Identificar ciclos circulares

- Obtener los ángulos de los puntos correspondientes a la trayectoria circular y puntos sin repetirse

```
In[•]:= getAngulosTrayCircular[data_] := Module[
                                         módulo
  {meanX, meanY, center, angulos = Table[0, Length[data]], vectorI, thisData = {0}, thisAng = {0}},
                                   tabla      longitud
  meanX = Mean[{Max[data[[All, 1]]], Min[data[[All, 1]]]}];
          media  máximo      todo          mínimo     todo
  meanY = Mean[{Max[data[[All, 2]]], Min[data[[All, 2]]]}];
          media  máximo      todo          mínimo     todo
  center = {meanX, meanY};

  Do[
  repite
   vectorI = data[[n]] - center;
                  ArcSin[ vectorI[[2]] / Norm[vectorI] ]
   angulos[[n]] = ------------------------------------- ;
                                  °

   Which[
   cuál
    (Sign[vectorI[[1]] == -1 && Sign[vectorI[[2]] == -1) || (Sign[vectorI[[1]] == -1 && Sign[vectorI[[2]] == 1),
     función signo              función signo               función signo              función signo
        angulos[[n]] = 180 - angulos[[n]],
    (Sign[vectorI[[1]] == 1 && Sign[vectorI[[2]] == -1),
     función signo             función signo
        angulos[[n]] = 360 + angulos[[n]],
    (Sign[vectorI[[1]] == 1 && vectorI[[2]] == 0),
     función signo
        angulos[[n]] = 0,
    (vectorI[[1]] == 0 && Sign[vectorI[[2]] == 1),
                          función signo
        angulos[[n]] = 90,
    (Sign[vectorI[[1]] == -1 && vectorI[[2]] == 0),
     función signo
        angulos[[n]] = 180,
    (vectorI[[1]] == 0 && Sign[vectorI[[2]] == -1),
                          función signo
        angulos[[n]] = 270
   ];

   , {n, Length[data]}];
        longitud



  (*Borrando angulos iguales*)
  Do[
  repite
   If[angulos[[h]] ≠ angulos[[h + 1]],
   si
    thisAng = Append[thisAng, angulos[[h]]];
              añade
    thisData = Append[thisData, data[[h]]];
               añade
   ]
   , {h, Length[angulos] - 1}];
        longitud
```

```
        longitud
    thisAng = Drop[thisAng, 1];
              elimina
    thisData = Drop[thisData, 1];
               elimina

    {thisAng, thisData}]
```

- Obtener los ciclos de la trayectoria circular

```
In[•]:= getCyclesCircle[data_] := Module[
                                  módulo
    {sentido = 1, thisAngulos = 0, flag = 0,
     ptoInicial = data[[1]], ciclos = {{0}}, inicio = 1, fin = 0, graphCycles = 0, thsiDat = 0},
    {thisAngulos, thsiDat} = getAngulosTrayCircular[data];

    (*Bor*)
    (*Identificar si es sentido horario o antihorario
      Sentido horario→1
       Sentido anti horario→-1
    *)
    Which[
    cuál
     thisAngulos[[2]] > thisAngulos[[1]],
         sentido = -1,
     355 ≤ thisAngulos[[1]] < 360 && 0 < thisAngulos[[2]] ≤ 5,
         sentido = -1
    ];

    (*Reconociendo los ciclos, dependiendo de la dirección*)
    If[sentido == 1,
    si
      (*Sentido horario*)
         Do[
         repite
               Which[
               cuál
                 (*Primer cuadrante*)
                 (0 < thisAngulos[[1]] < 90 && thisAngulos[[n]] ≥ 90 && thisAngulos[[n + 1]] < 90),
                     flag = 1,
                 (*Segundo cuadrante*)
                 (90 < thisAngulos[[1]] < 180 && thisAngulos[[n]] ≥ 180 && thisAngulos[[n + 1]] < 180),
                     flag = 1,
                 (*Tercer cuadrante*)
                 (180 < thisAngulos[[1]] < 270 && thisAngulos[[n]] ≥ 270 && thisAngulos[[n + 1]] < 270),
                     flag = 1,
                 (*Cuarto Cuadrante*)
                 (270 < thisAngulos[[1]] < 360 && thisAngulos[[n]] > 0 && thisAngulos[[n + 1]] < 360),
                     flag = 1
                ];

               Which[
               cuál
                 (0 < thisAngulos[[n]] < 180 && flag == 1 && thsiDat[[n, 1]] ≥ ptoInicial[[1]]),
                     fin = n;
                     ciclos = Append[ciclos, Table[thsiDat[[m]], {m, inicio, fin}]];
                              añade          tabla
                     flag = 0;
                     inicio = n,
                 (180 < thisAngulos[[n]] < 360 && flag == 1 && data[[n, 1]] ≤ ptoInicial[[1]]),
                     fin = n;
```

```
                ciclos = Append[ciclos, Table[thsiDat[[m]], {m, inicio, fin}]];
                         añade          tabla
                flag = 0;
                inicio = n
            ];
        , {n, Length[thsiDat] - 1}],
             longitud
    (*Sentido anti horario*)
        Do[
        repite
            Which[
            cuál
              (*Primer cuadrante*)
              (0 < thsiAngulos[[1]] < 90 && thsiAngulos[[n]] > 270 && thsiAngulos[[n + 1]] < 90),
                flag = 1,
              (*Segundo cuadrante*)
              (90 < thsiAngulos[[1]] < 180 && thsiAngulos[[n]] ≤ 90 && thsiAngulos[[n + 1]] > 90),
                flag = 1,
              (*Tercer cuadrante*)
              (180 < thsiAngulos[[1]] < 270 && thsiAngulos[[n]] ≤ 180 && thsiAngulos[[n + 1]] > 180),
                flag = 1,
              (*Cuarto Cuadrante*)
              (270 < thsiAngulos[[1]] < 360 && thsiAngulos[[n]] ≤ 270 && thsiAngulos[[n + 1]] > 270),
                flag = 1
              ];

            Which[
            cuál

              (0 < thsiAngulos[[n]] < 180 && flag == 1 && thsiDat[[n, 1]] ≤ ptoInicial[[1]]),
                fin = n;
                ciclos = Append[ciclos, Table[thsiDat[[m]], {m, inicio, fin}]];
                         añade          tabla
                flag = 0;
                inicio = n,
              (180 < thsiAngulos[[n]] < 360 && flag == 1 && thsiDat[[n, 1]] ≥ ptoInicial[[1]]),
                fin = n;
                ciclos = Append[ciclos, Table[thsiDat[[m]], {m, inicio, fin}]];
                         añade          tabla
                flag = 0;
                inicio = n
            ];
        , {n, Length[thsiDat] - 1}]
             longitud
];

ciclos = Drop[ciclos, 1];
         elimina
graphCycles = Table[0, Length[ciclos]];
              tabla    longitud
Do[
repite
 graphCycles[[m]] = ListPlot[ciclos[[m]]];
                    representación de lista
 , {m, Length[graphCycles]}];
       longitud

{ciclos, graphCycles}]
```

- Obtener ciclos trayectoria vertical, dirección positiva

In[•]:=
```
getCyclesVerticalUpp[data_] := Module[
                               módulo
  {upp = 0, ciclos = {0}, thisP = {0}, thisY = {0}, graphCycles = 0},

  (*Limpiar la trayectoria de coordenadas repetidas en Y*)
  Do[
  repite
   If[data[[m, 2]] ≠ data[[m + 1, 2]] ,
   si
     thisP = Append[thisP, data[[m]]];
             añade
    ];

   , {m, Length[data] - 1}];
       longitud
  thisP = Drop[thisP, 1];
         elimina
  (*Identificar los ciclos*)
  Do[
  repite
    If[(thisP[[k - 1, 2]] < thisP[[k, 2]] > thisP[[k + 1, 2]]) &&
    si
   (thisP[[k - 2, 2]] < thisP[[k, 2]] > thisP[[k + 2, 2]]) && (thisP[[k - 3, 2]] < thisP[[k, 2]] > thisP[[k + 3, 2]]) &&
   (thisP[[k - 4, 2]] < thisP[[k, 2]] > thisP[[k + 4, 2]]) && (thisP[[k - 5, 2]] < thisP[[k, 2]] > thisP[[k + 5, 2]]),
          thisY = Append[thisY, k];
                  añade
          upp += 1;
      ];


   , {k, 6, Length[thisP] - 5}];
           longitud
  thisY = Drop[thisY, 1];
         elimina


  (*Guardando los ciclos*)
  Do[
  repite
   ciclos = Append[ciclos, Table[thisP[[i]], {i, thisY[[l]], thisY[[l + 1]]}]]
            añade               tabla
   , {l, Length[thisY] - 1}];
       longitud
  ciclos = Drop[ciclos, 1];
          elimina
  graphCycles = Table[0, Length[ciclos]];
                tabla      longitud


  (*Graficas de los ciclos*)
  Do[
  repite
    graphCycles[[c]] = ListPlot[ciclos[[c]]];
                       representación de lista
   , {c, Length[ciclos]}];
       longitud
  {ciclos, graphCycles}]
```

- Obtener ciclos trayectoria vertical, dirección negativa

```
In[•]:= getCyclesVerticalDown[data_] := Module[
                                        módulo
  {down = 0, ciclos = {0}, thisP = {0}, thisY = {0}, graphCycles = 0},

  (*Limpiar la trayectoria de coordenadas repetidas en Y del punto i y del punto i+1*)
  Do[
  repite
   If[data[[m, 2]] ≠ data[[m + 1, 2]] ,
   si
     thisP = Append[thisP, data[[m]]];
             añade
    ];

   , {m, Length[data] - 1}];
        longitud
  thisP = Drop[thisP, 1];
          elimina
  (*Identificar los ciclos*)
  Do[
  repite
       If[(thisP[[k, 2]] < thisP[[k - 1, 2]] < thisP[[k - 2, 2]] < thisP[[k - 3, 2]] < thisP[[k - 4, 2]] < thisP[[k - 5, 2]]) &&
       si
      (thisP[[k, 2]] < thisP[[k + 1, 2]] < thisP[[k + 2, 2]] < thisP[[k + 3, 2]] < thisP[[k + 4, 2]] < thisP[[k + 5, 2]]),
            thisY = Append[thisY, k];
                    añade
            down += 1;
        ];


   , {k, 6, Length[thisP] - 5}];
           longitud
  thisY = Drop[thisY, 1];
          elimina



  (*Guardando los ciclos*)
  Do[
  repite
       ciclos = Append[ciclos, Table[thisP[[i]], {i, thisY[[l]], thisY[[l + 1]]}]]
                añade             tabla
   , {l, Length[thisY] - 1}];
        longitud
  ciclos = Drop[ciclos, 1];
           elimina
  graphCycles = Table[0, Length[ciclos]];
                tabla    longitud


  (*Graficas de los ciclos*)
  Do[
  repite
       graphCycles[[c]] = ListPlot[ciclos[[c]]];
                          representación de lista
   , {c, Length[ciclos]}];
        longitud
  {ciclos, graphCycles}]
```

- Obtener ciclos trayectoria horizontal

```
In[•]:= getCyclesHorizontal[data_] := Module[
                                      módulo

  {Xmean = 0, Pbase = 0, vector = 0, angle = Table[0, Length[data]],
                                     tabla    longitud
```

```
                                                                    tabla      longitud
 ciclos = {0}, flag = 0, inicio = 0, fin = 0, graphCycles = {0}, thisData = {0}, cleanAngles = {0}},
Xmean = Mean[{Max[data[[All, 1]]], Min[data[[All, 1]]]}];
        media  máximo    todo       mínimo    todo
Pbase = {Xmean, 0};

(*Obteniendo los ángulos del vector que recorre la trayectoria*)
Do[
repite
     vector = data[[k]] - Pbase;
                vector[[1]]
                Norm[vector]
     angle[[k]] = ------------ ;
                      °
     If[(Sign[vector[[1]]] == -1 && Sign[vector[[2]]] == 1) || (Sign[vector[[1]]] == -1 && Sign[vector[[2]]] == -1),
     si  función signo              función signo              función signo              función signo
          angle[[k]] = 180 - angle[[k]]
      ];
 , {k, Length[data]}];
       longitud

(*Si hay ángulos es el mismo punto por lo que se eliminarán*)
Do[
repite
     If[angle[[b]] ≠ angle[[b + 1]],
     si
          thisData = Append[thisData, data[[b]]];
                     añade
          cleanAngles = Append[cleanAngles, angle[[b]]];
                        añade
       ];
 , {b, Length[angle] - 1}];
       longitud
thisData = Drop[thisData, 1];
           elimina
cleanAngles = Drop[cleanAngles, 1];
              elimina
(*
flag=0, Sentido horario;
flag=1, Sentido anti horario
*)
If[cleanAngles[[1]] < 90, flag = 1];
si
inicio = 6;

(*Comparando los ángulos para saber si ya dio la vuelta*)
If[flag == 0,
si
 Do[
 repite
  If[(cleanAngles[[l + 1]] < cleanAngles[[l]] > cleanAngles[[l - 1]]) &&
  si
    (cleanAngles[[l + 2]] < cleanAngles[[l]] > cleanAngles[[l - 2]]) && (cleanAngles[[l + 3]] < cleanAngles[[l]] >
      cleanAngles[[l - 3]]) && (cleanAngles[[l + 4]] < cleanAngles[[l]] > cleanAngles[[l - 4]]) &&
    (cleanAngles[[l + 5]] < cleanAngles[[l]] > cleanAngles[[l - 5]]) && cleanAngles[[l]] > 90,
       fin = l;
       ciclos = Append[ciclos, Table[thisData[[o]], {o, inicio, fin}]];
                añade          tabla
       inicio = l;
  ]
  , {l, 6, Length[cleanAngles] - 5}],
          longitud
```

```
                longitud

 Do[
  repite
   If[(cleanAngles[[l + 1]] > cleanAngles[[l]] < cleanAngles[[l - 1]]) &&
    si
     (cleanAngles[[l + 2]] > cleanAngles[[l]] < cleanAngles[[l - 2]]) && (cleanAngles[[l + 3]] > cleanAngles[[l]] <
       cleanAngles[[l - 3]]) && (cleanAngles[[l + 4]] > cleanAngles[[l]] < cleanAngles[[l - 4]]) &&
     (cleanAngles[[l + 5]] > cleanAngles[[l]] < cleanAngles[[l - 5]]) && cleanAngles[[l]] < 90,
       fin = l;
       ciclos = Append[ciclos, Table[thisData[[o]], {o, inicio, fin}]];
                añade          tabla
       inicio = l;
   ]
   , {l, 6, Length[cleanAngles] - 5}];
            longitud
 ];
 ciclos = Drop[ciclos, 1];
          elimina

 graphCycles = Table[0, Length[ciclos]];
               tabla    longitud

 Do[
 repite

  graphCycles[[p]] = ListPlot[ciclos[[p]]];
                     representación de lista
  , {p, Length[ciclos]}];
       longitud

 {ciclos, graphCycles}]
```

Recorrer la trayectoria punto por punto

```
In[•]:= stepByStepPlot[data_] := Module[
                                 módulo
 {},
 Manipulate[ListPlot[Table[data[[k]], {k, 1, g}]], {g, 1, Length[data], 1, Appearance → "Open"}]
 manip···   representa··  tabla                    longitud                apariencia     abre
]
```

Obtener promedio centro y radio círculo

```
In[•]:= getMean[datos_] := Module[
                           módulo
 {Xmean = 0, Ymean = 0, Rmean = 0},
 Xmean = Mean[datos[[All, 1, 1]]];
         media  todo
 Ymean = Mean[datos[[All, 1, 2]]];
         media  todo
 Rmean = Mean[datos[[All, 2]]];
         media  todo
 {{Xmean, Ymean}, Rmean} // N]
                            valor nu
```

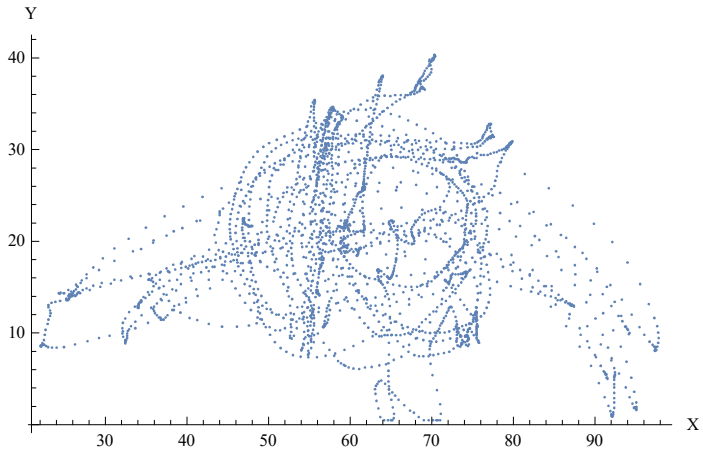
Ciclos de trabajo

3.- Gráficas y ciclos por paciente

MESRM

In[•]:=
```
ListPlot[MESRM[todas], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

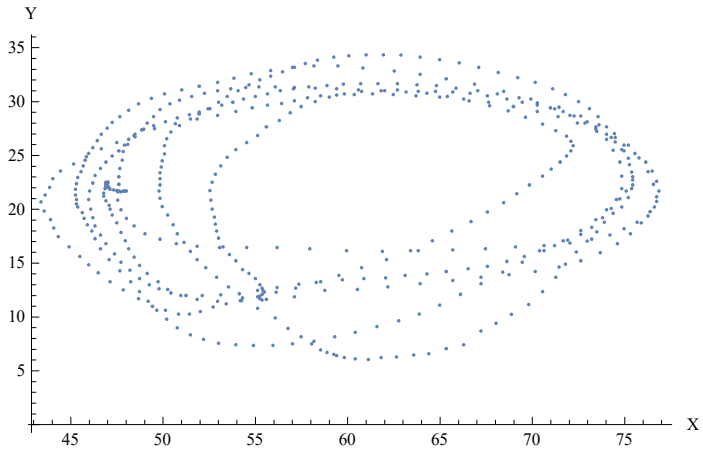
Out[•]=



Circular

In[•]:=
```
ListPlot[MESRM[circular], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

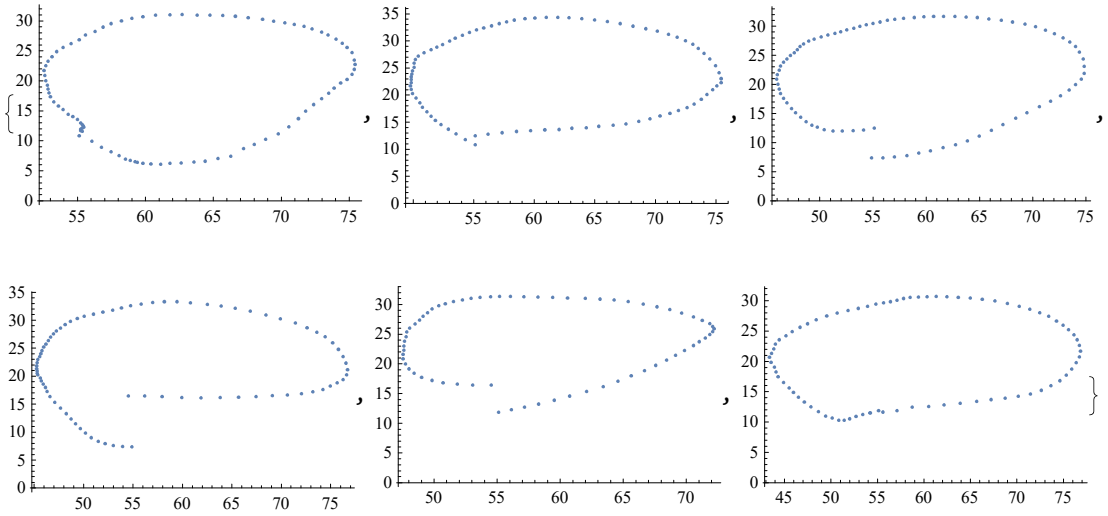
Out[•]=



Ciclos de trabajo

In[•]:=
```
getCyclesCircle[MESRM[circular]][[2]]
```

Out[•]=

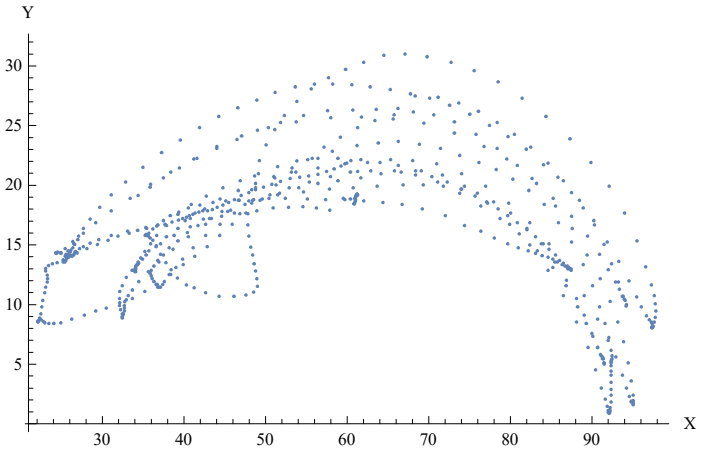


In[•]:=
```
MESRM = Append[MESRM, ciclosCircular -> getCyclesCircle[MESRM[circular]][[1]]];
```
añade

Horizontal

In[•]:=
```
ListPlot[MESRM[horizontal], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

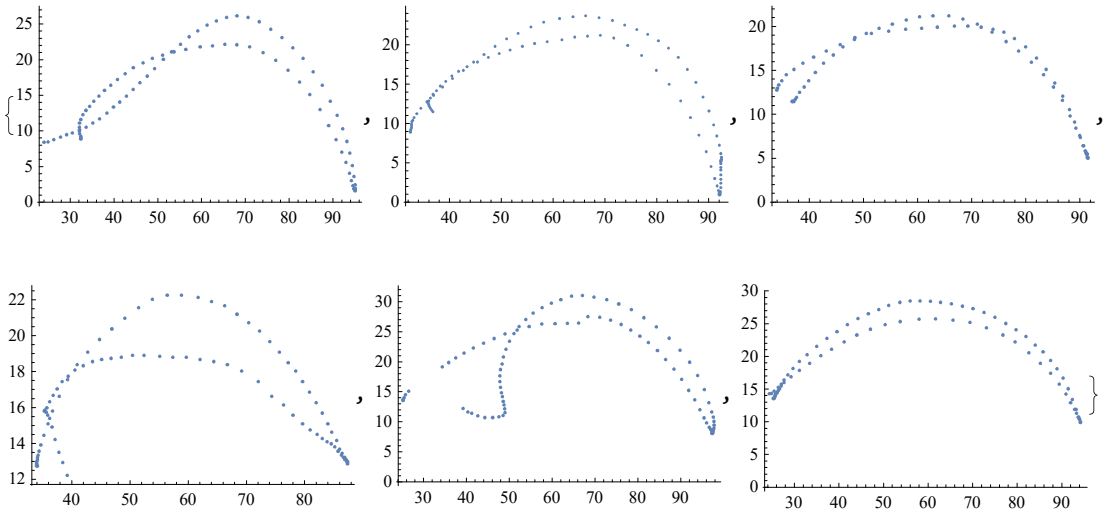
Out[•]=



Ciclos de trabajo

In[•]:=
```
getCyclesHorizontal[Table[MESRM[horizontal][[o]], {o, 63, Length[MESRM[horizontal]]}]][[2]]
```
tabla longitud

Out[•]=

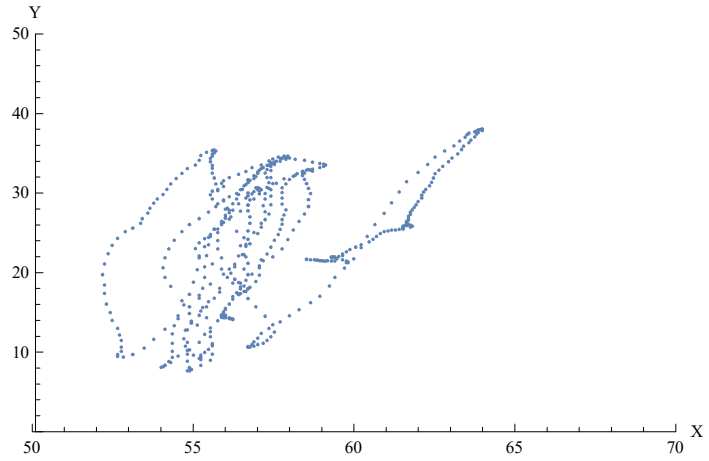


In[•]:=
```
MESRM = Append[MESRM,
  ciclosHorizontal -> getCyclesHorizontal[Table[MESRM[horizontal][[o]], {o, 63, Length[MESRM[horizontal]]}]][[1]]];
```
añade tabla longitud

Vertical

In[•]:=
```
ListPlot[MESRM[vertical], PlotRange → {{50, 70}, {0, 50}}, AxesLabel → {"X", "Y"}]
```
representación de lista | rango de representación | etiqueta de ejes

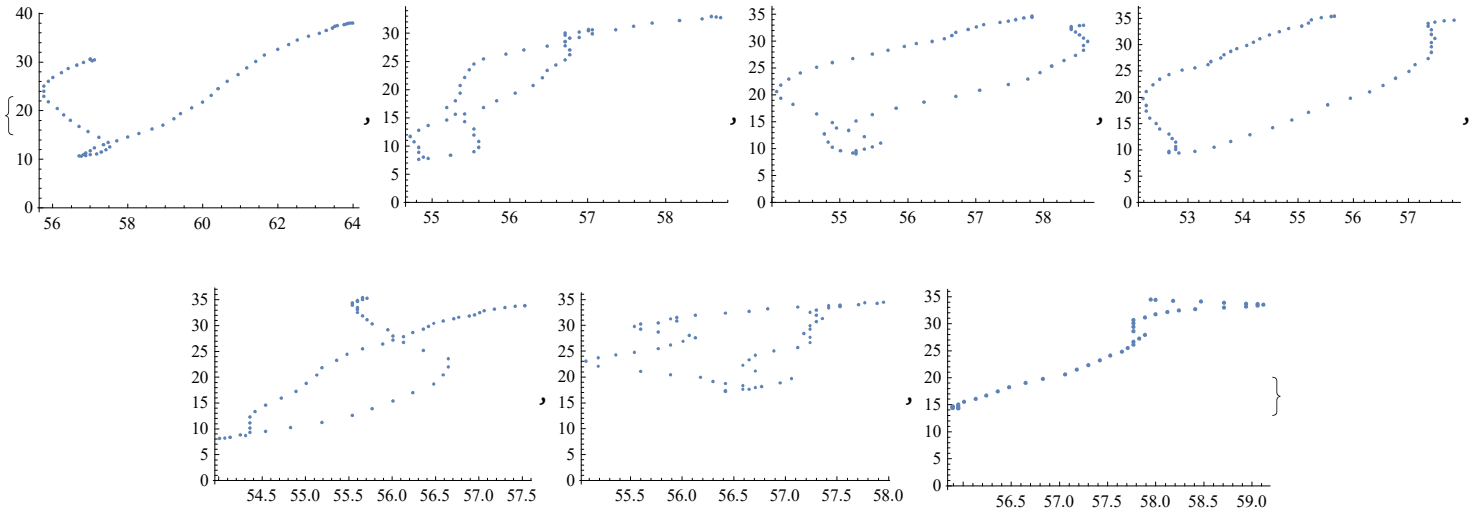
Out[•]=



Ciclos

In[•]:=
```
getCyclesVerticalUpp[MESRM[vertical]][[2]]
```

Out[•]=

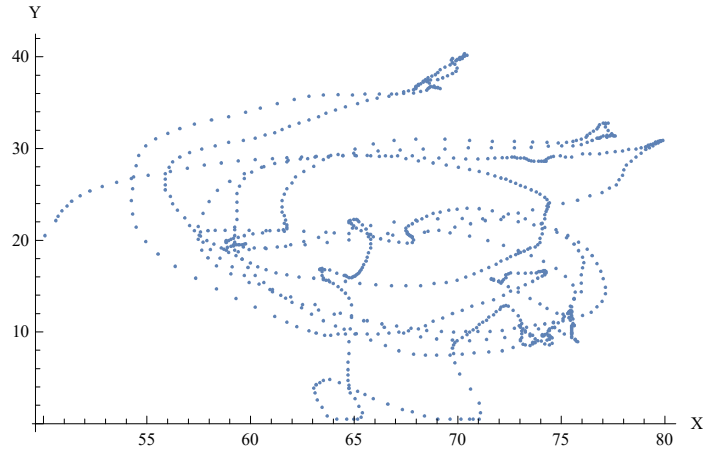


In[•]:=
```
MESRM = Append[MESRM, ciclosVertical → Table[getCyclesVerticalUpp[MESRM[vertical]][[1]][[i]], {i, 6}]];
```
añade | tabla

Infinito [El paciente no logró realizar un ciclo, estaba muy limitado]

In[•]:=
```
ListPlot[MESRM[infinito], AxesLabel → {"X", "Y"}]
```
representación de lista | etiqueta de ejes

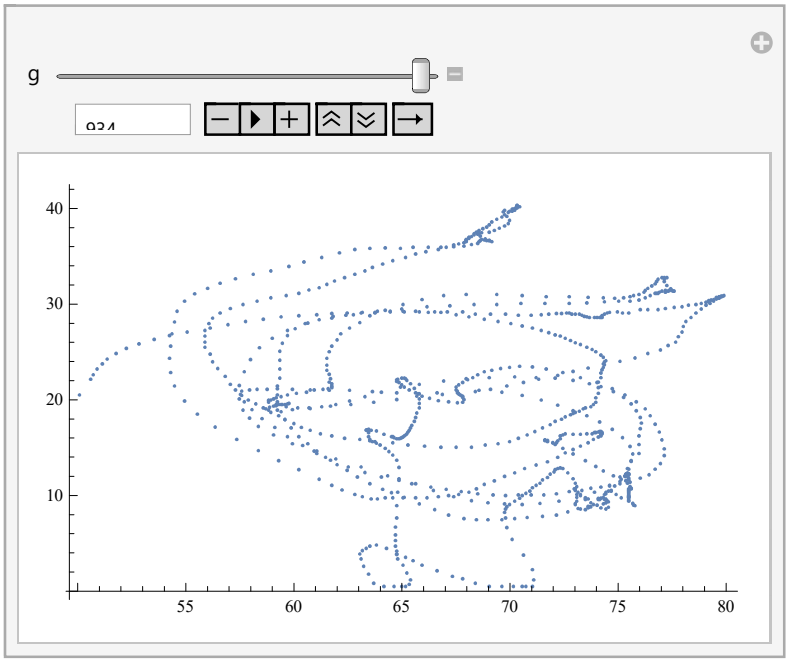
Out[•]=



Ciclos

In[•]:=
```
stepByStepPlot[MESRM[infinito]]
```

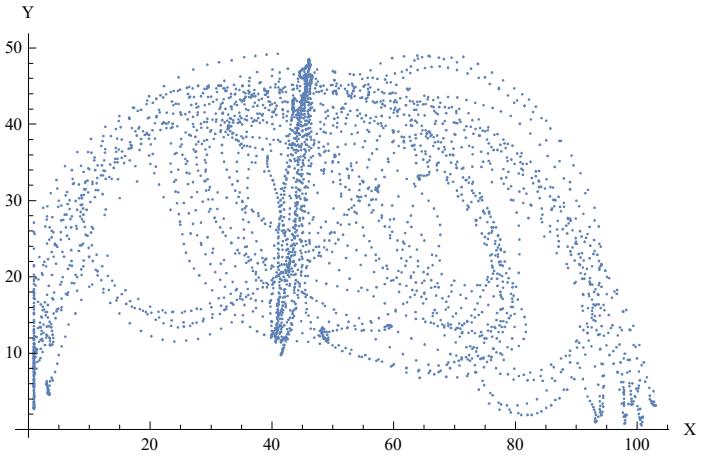
Out[•]=



FJLH

In[•]:=
```
ListPlot[FJLH[todas], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

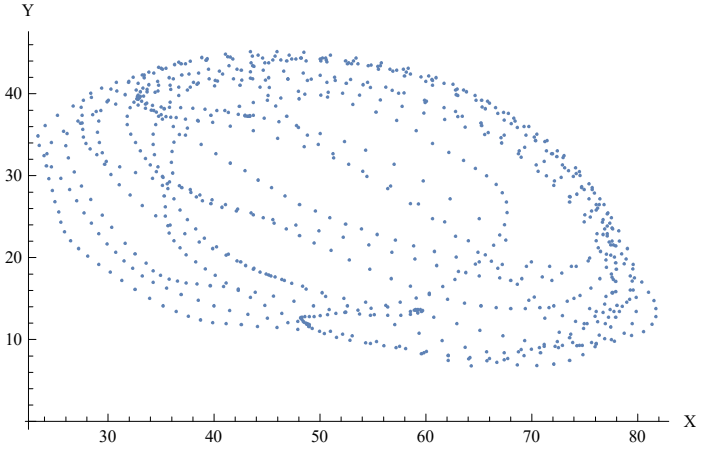
Out[•]=



Circular

In[•]:=
```
ListPlot[FJLH[circular], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

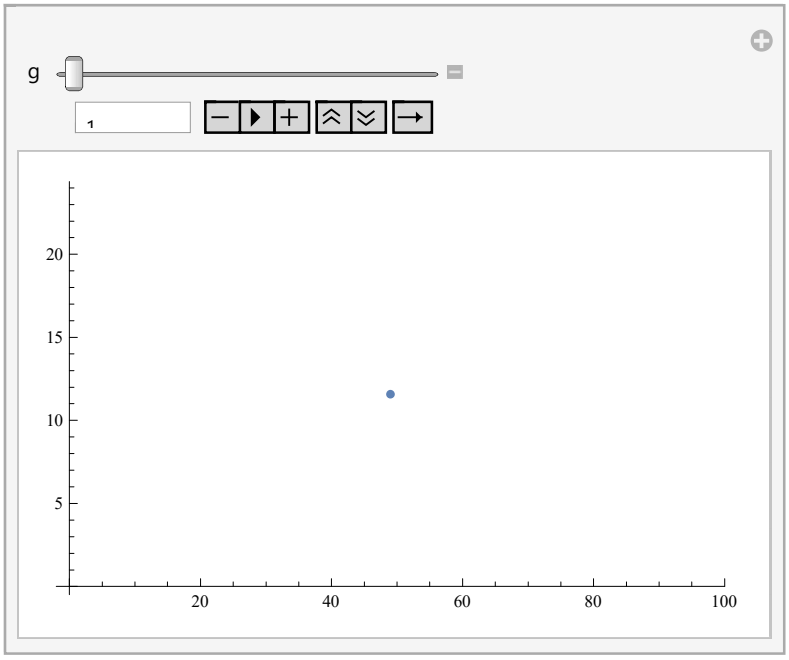
Out[•]=



Ciclos

In[•]:=
```
stepByStepPlot[FJLH[circular]]
```

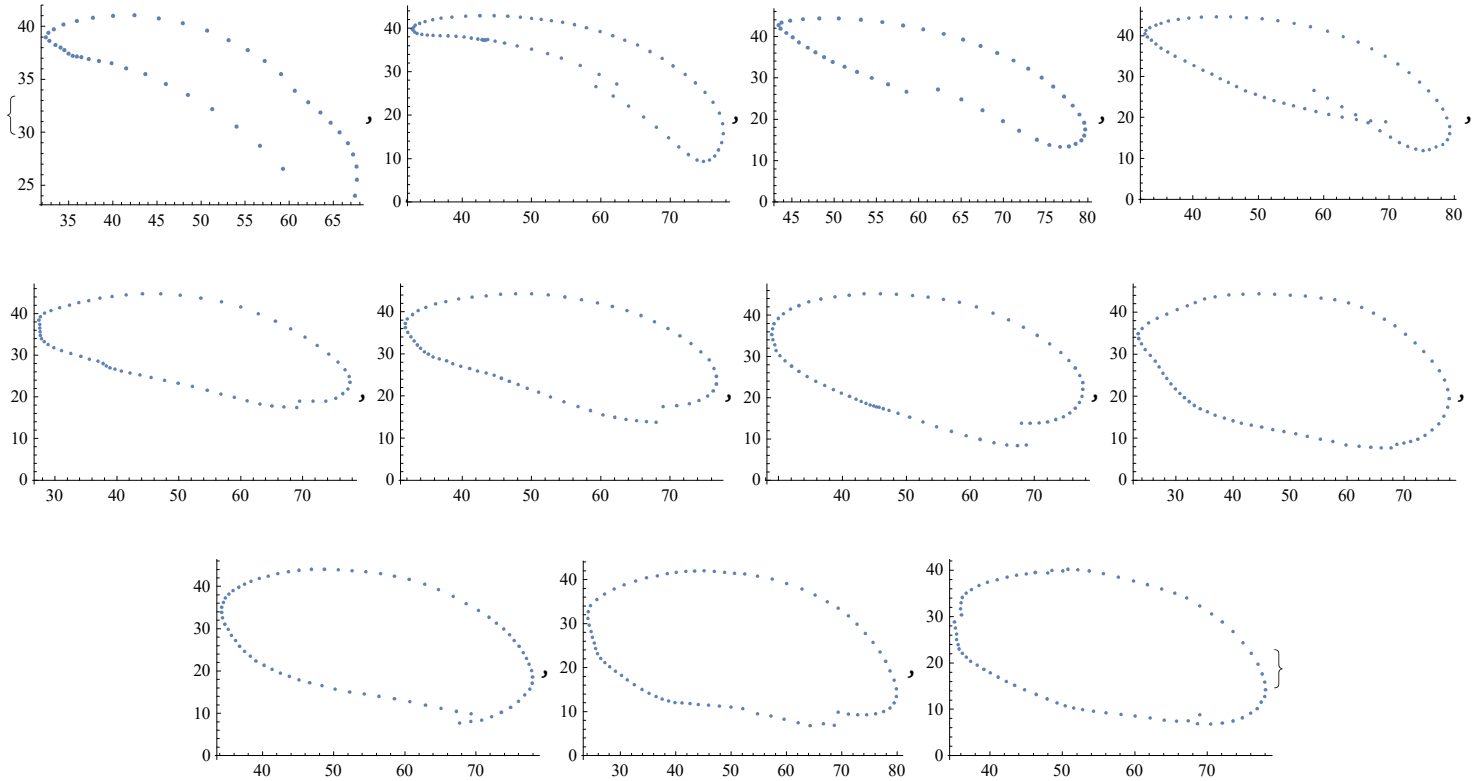
Out[•]=



Como se puede observar en la anterior animación, el paciente comenzó el ciclo hasta el punto 40, por lo que unicamente se tomará a partir de ese punto.

In[•]:=
```
getCyclesCircle[Table[FJLH[circular][[o]], {o, 40, Length[FJLH[circular]]}]][[2]]
```
tabla longitud

Out[•]=

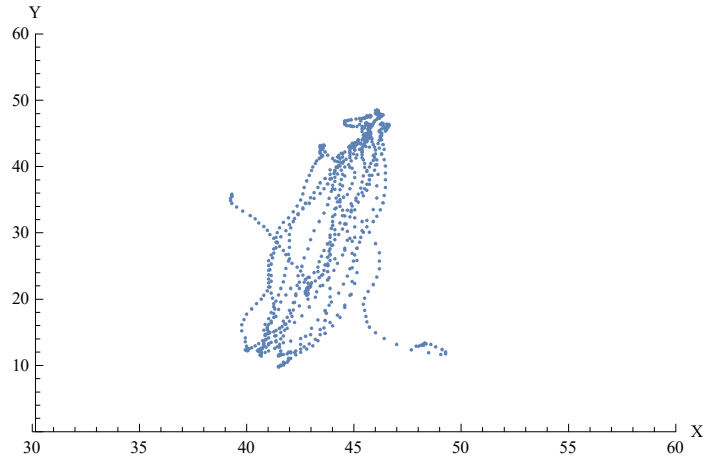


In[•]:=
```
FJLH = Append[FJLH, ciclosCircular -> getCyclesCircle[Table[FJLH[circular][[o]], {o, 40, Length[FJLH[circular]]}]][[1]]];
```
añade tabla longitud

Vertical

```
In[•]:= ListPlot[FJLH[vertical], PlotRange → {{30, 60}, {0, 60}}, AxesLabel → {"X", "Y"}]
        representación de lista   rango de representación   etiqueta de ejes
```

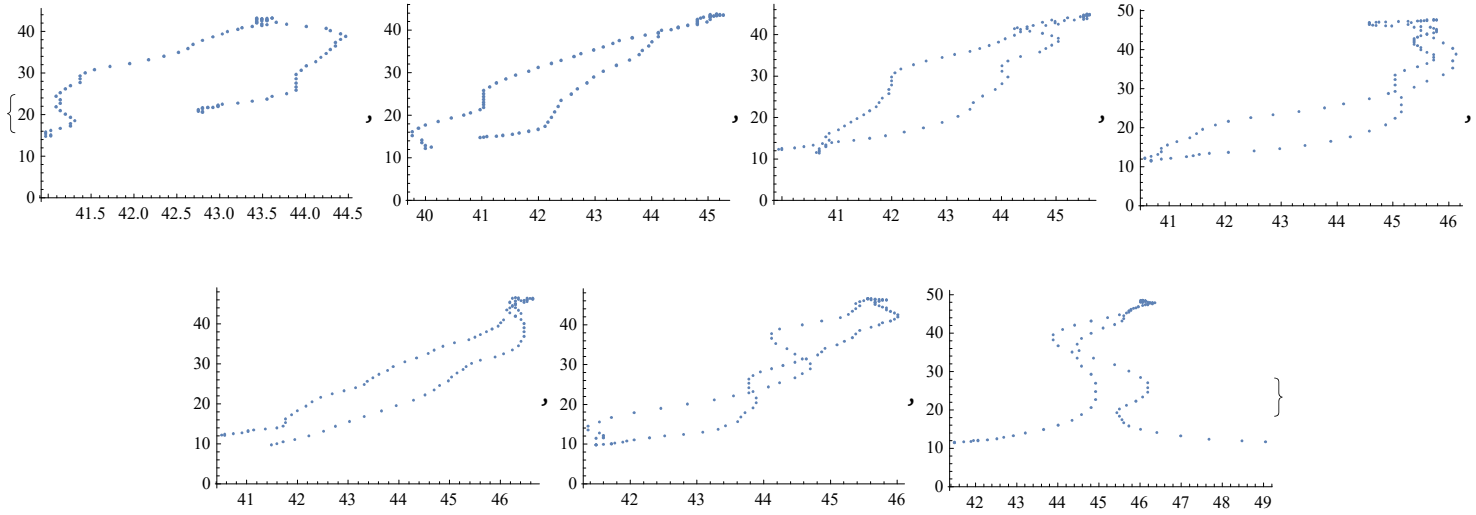
Out[•]=



Ciclos

```
In[•]:= getCyclesVerticalDown[FJLH[vertical]][[2]]
```

Out[•]=

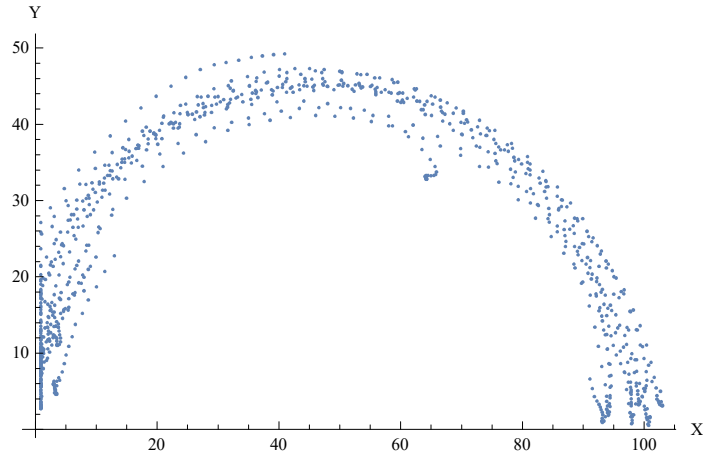



```
In[•]:= FJLH = Append[FJLH, ciclosVertical -> getCyclesVerticalDown[FJLH[vertical]][[1]]];
        añade
```

Horizontal

```
In[•]:= ListPlot[FJLH[horizontal], AxesLabel → {"X", "Y"}]
        representación de lista   etiqueta de ejes
```

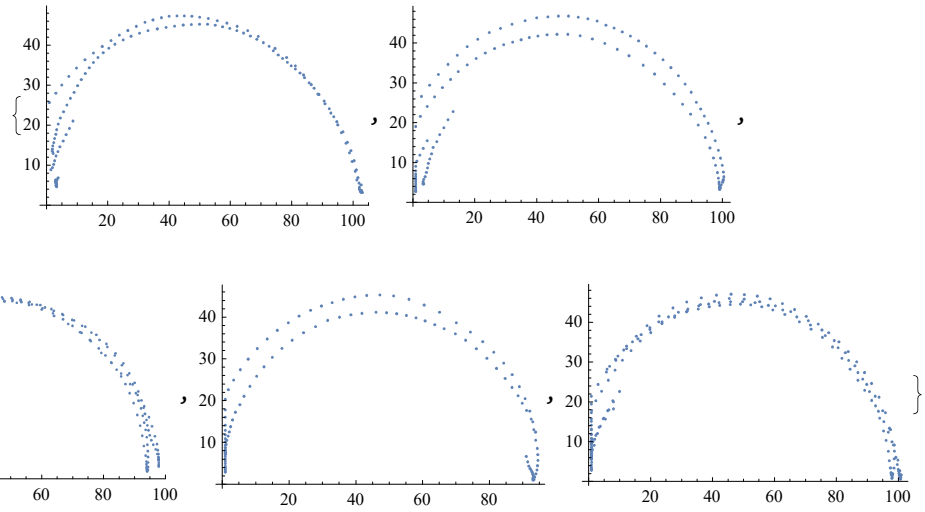
Out[•]=



Ciclos

In[•]:=
```
getCyclesHorizontal[Table[FJLH[horizontal][[o]], {o, 43, Length[FJLH[horizontal]]}]][[2]]
```
tabla longitud

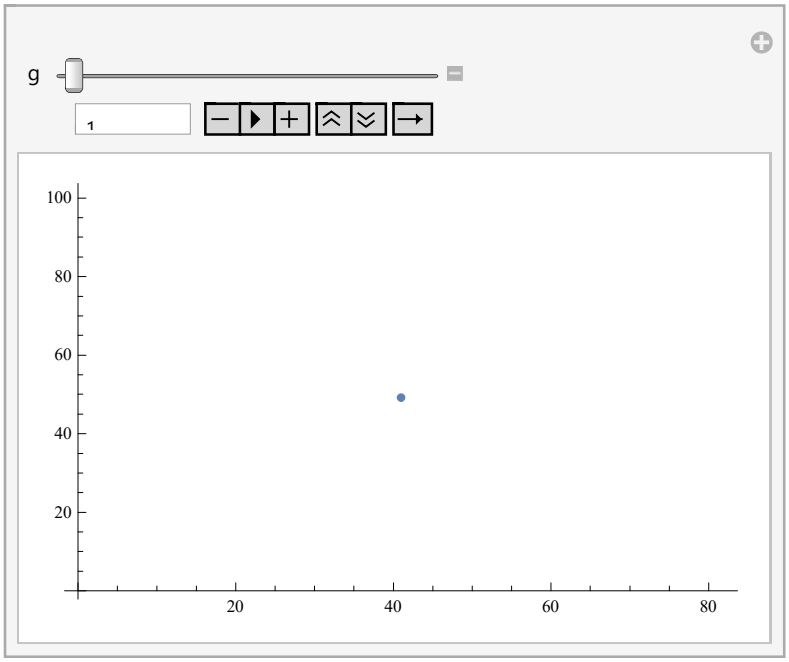
Out[•]=



En esta trayectoria, el paciente se salió del área de grabación por lo que la identificación de los ciclos se hará manualmente.

In[•]:=
```
stepByStepPlot[FJLH[horizontal]]
```

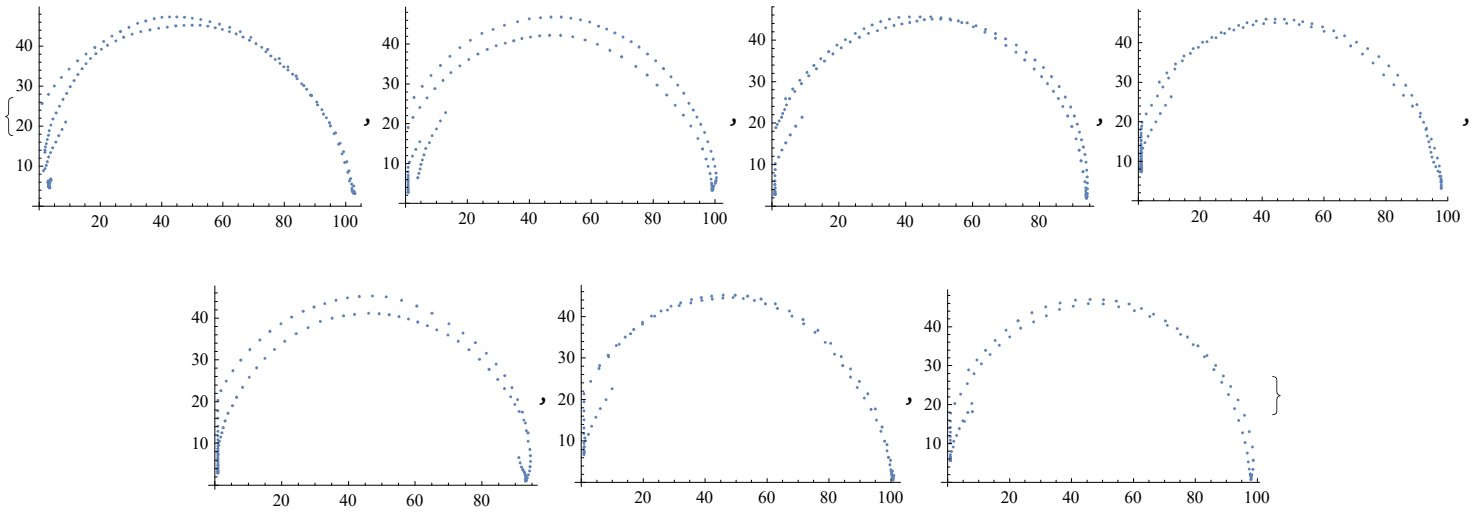
Out[•]=



```
In[•]:= {ListPlot[Table[FJLH[horizontal][[p]], {p, 49, 201}]],
          representa··  tabla
         ListPlot[Table[FJLH[horizontal][[p]], {p, 201, 321}]],
          representa··  tabla
         ListPlot[Table[FJLH[horizontal][[p]], {p, 321, 445}]],
          representa··  tabla
         ListPlot[Table[FJLH[horizontal][[p]], {p, 445, 601}]],
                    tabla
         ListPlot[Table[FJLH[horizontal][[p]], {p, 601, 732}]],
                    tabla
         ListPlot[Table[FJLH[horizontal][[p]], {p, 732, 832}]],
          representa··  tabla
         ListPlot[Table[FJLH[horizontal][[p]], {p, 832, 937}]]
          representa··  tabla

        }
```

Out[•]=



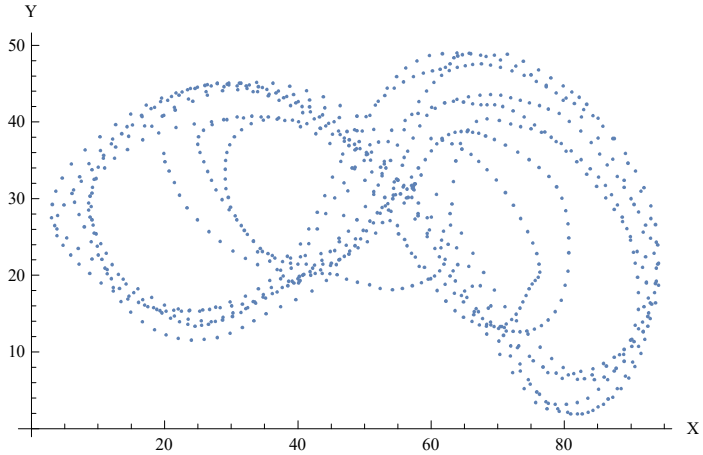
A pesar de ser ciclos “confusos”, el método distingue relativamente bien los ciclos

```
In[•]:= FJLH = Append[FJLH, ciclosHorizontal → {
           añade
          Table[FJLH[horizontal][[p]], {p, 445, 601}],
          tabla
          Table[FJLH[horizontal][[p]], {p, 601, 732}],
          tabla
          Table[FJLH[horizontal][[p]], {p, 832, 937}]
          tabla
         }];
```

- **Infinito**

```
In[•]:= ListPlot[FJLH[infinito], AxesLabel → {"X", "Y"}]
         representación de lista   etiqueta de ejes
```

Out[•]=

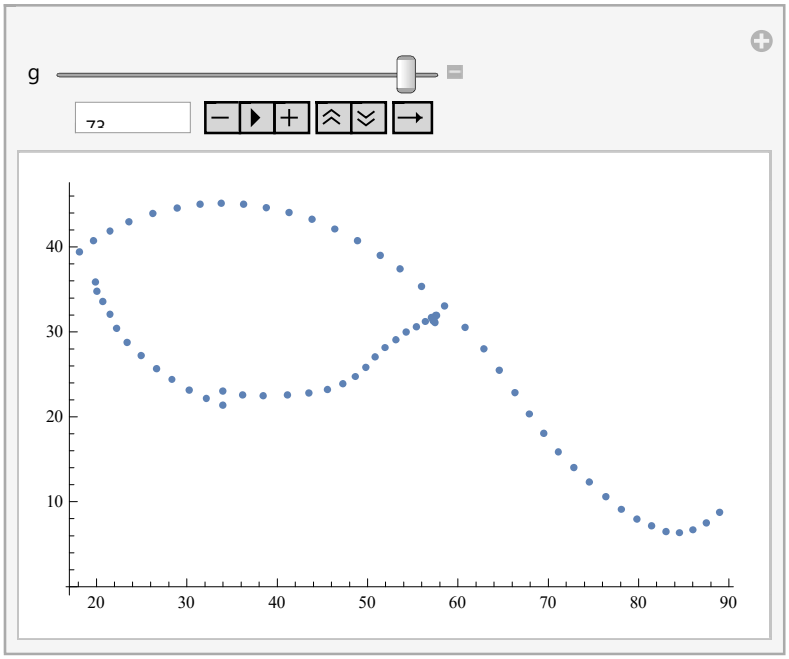


In[•]:=

Ciclos

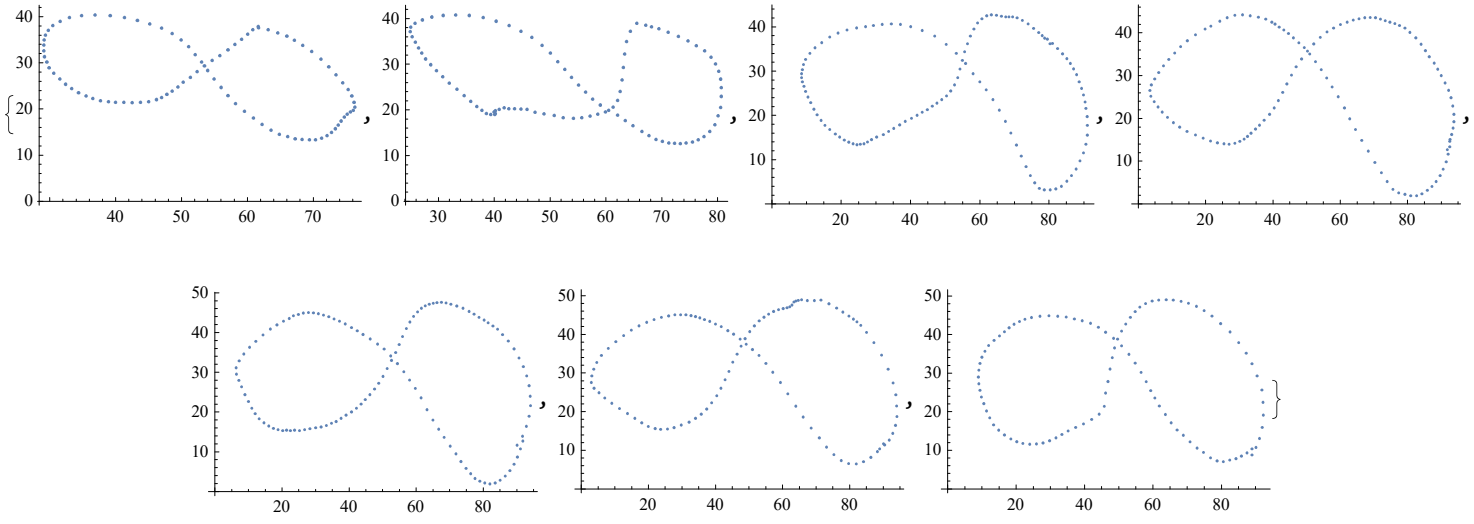
In[•]:=
```
stepByStepPlot[Table[FJLH[infinito][[y]], {y, 802, Length[FJLH[infinito]]}]]
```
tabla longitud

Out[•]=



In[•]:=
```
{ListPlot[Table[FJLH[infinito][[y]], {y, 2, 95}]],
 ListPlot[Table[FJLH[infinito][[y]], {y, 101, 199}]],
 ListPlot[Table[FJLH[infinito][[y]], {y, 207, 328}]],
 ListPlot[Table[FJLH[infinito][[y]], {y, 340, 461}]],
 ListPlot[Table[FJLH[infinito][[y]], {y, 461, 587}]],
 ListPlot[Table[FJLH[infinito][[y]], {y, 587, 698}]],
 ListPlot[Table[FJLH[infinito][[y]], {y, 698, 802}]]
}
```
representa·· tabla (×7)

Out[•]=

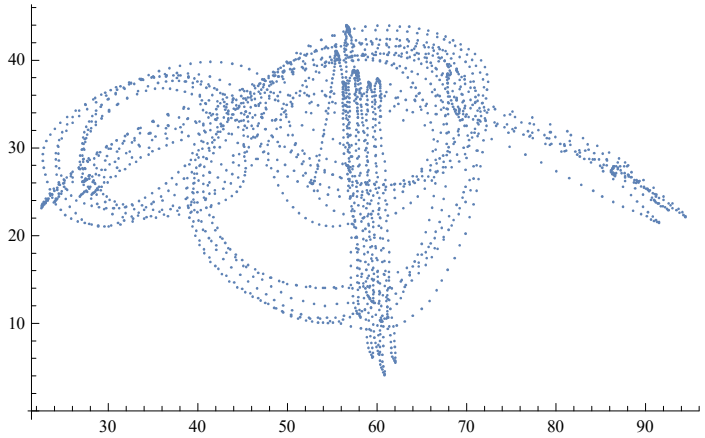


```
In[•]:= FJLH = Append[FJLH, ciclosInfinito → {Table[FJLH[infinito][[y]], {y, 2, 95}],
          añade                              tabla
    Table[FJLH[infinito][[y]], {y, 101, 199}],
    tabla
    Table[FJLH[infinito][[y]], {y, 207, 328}],
    tabla
    Table[FJLH[infinito][[y]], {y, 340, 461}],
    tabla
    Table[FJLH[infinito][[y]], {y, 461, 587}],
    tabla
    Table[FJLH[infinito][[y]], {y, 587, 698}],
    tabla
    Table[FJLH[infinito][[y]], {y, 698, 802}]}];
    tabla
```

AOGM

```
In[•]:= ListPlot[AOGM[todas]]
        representación de lista
```

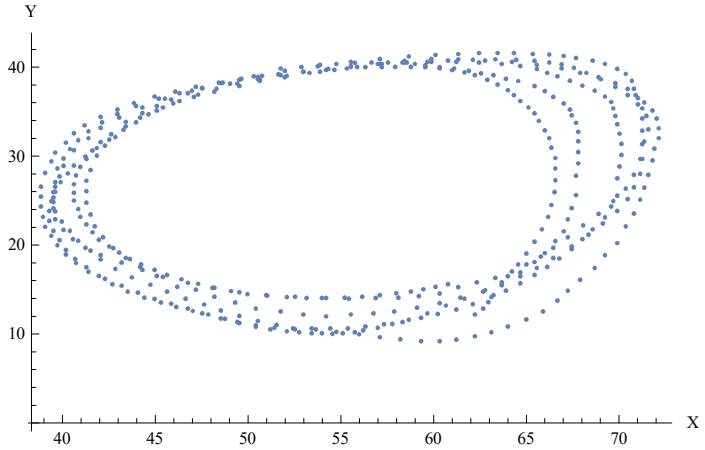
Out[•]=



Circular

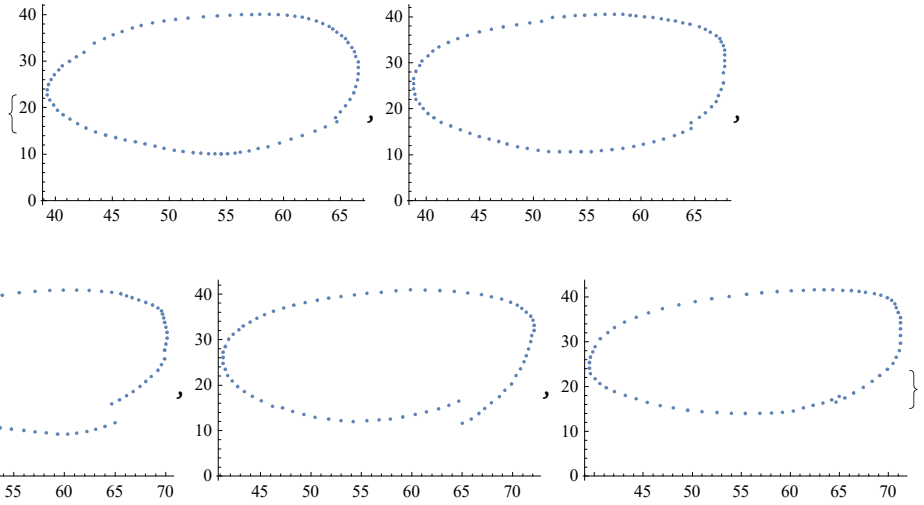
```
In[•]:= ListPlot[AOGM[circular], AxesLabel → {"X", "Y"}]
        representación de lista    etiqueta de ejes
```

Out[•]=



In[•]:=
```
getCyclesCircle[Table[AOGM[circular][[i]], {i, 7, Length[AOGM[circular]]}]][[2]]
```
tabla longitud

Out[•]=

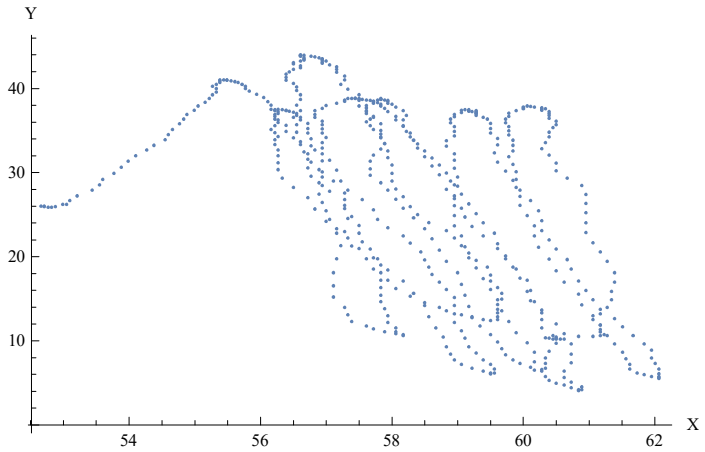


In[•]:=
```
AOGM = Append[AOGM, ciclosCircular → getCyclesCircle[Table[AOGM[circular][[i]], {i, 7, Length[AOGM[circular]]}]][[1]]];
```
añade tabla longitud

■ Vertical

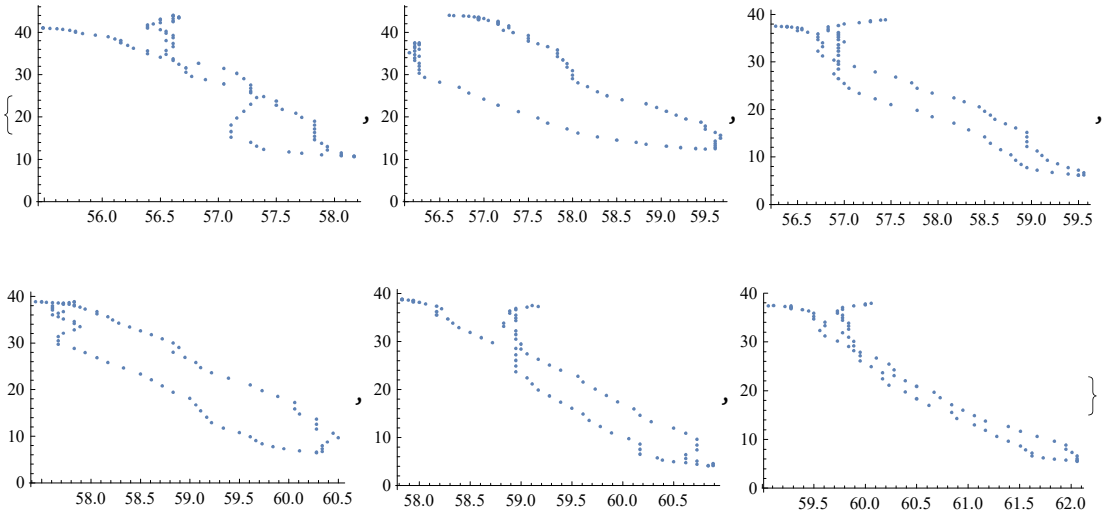
In[•]:=
```
ListPlot[AOGM[vertical], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

Out[•]=



In[•]:=
```
getCyclesVerticalUpp[AOGM[vertical]][[2]]
```

Out[•]=

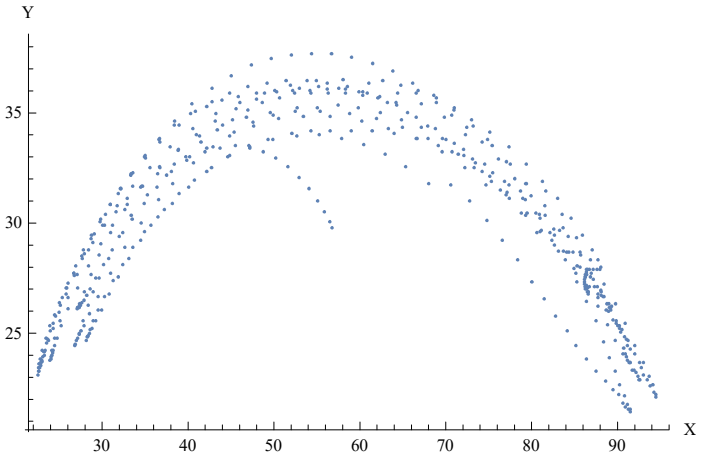


In[•]:=
```
AOGM = Append[AOGM, ciclosVertical → getCyclesVerticalUpp[AOGM[vertical]][[1]]];
```
añade

Horizontal

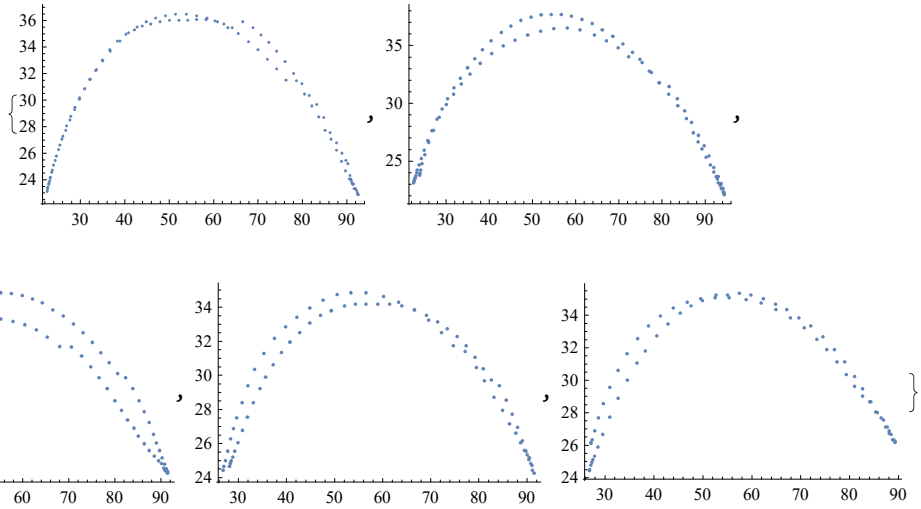
In[•]:=
```
ListPlot[AOGM[horizontal], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

Out[•]=



In[•]:=
```
getCyclesHorizontal[Table[AOGM[horizontal][[i]], {i, 28, Length[AOGM[horizontal]]}]][[2]]
```
tabla longitud

Out[•]=

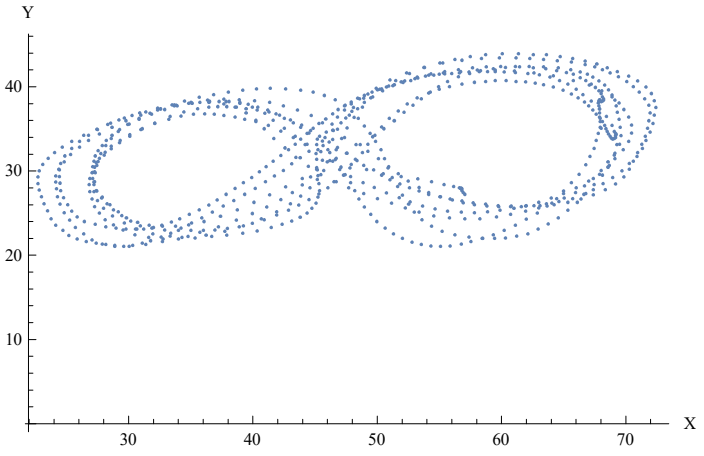


In[•]:=
```
AOGM = Append[AOGM,
```
añade
```
  ciclosHorizontal → getCyclesHorizontal[Table[AOGM[horizontal][[i]], {i, 28, Length[AOGM[horizontal]]}]][[1]]];
```
tabla longitud

Infinito

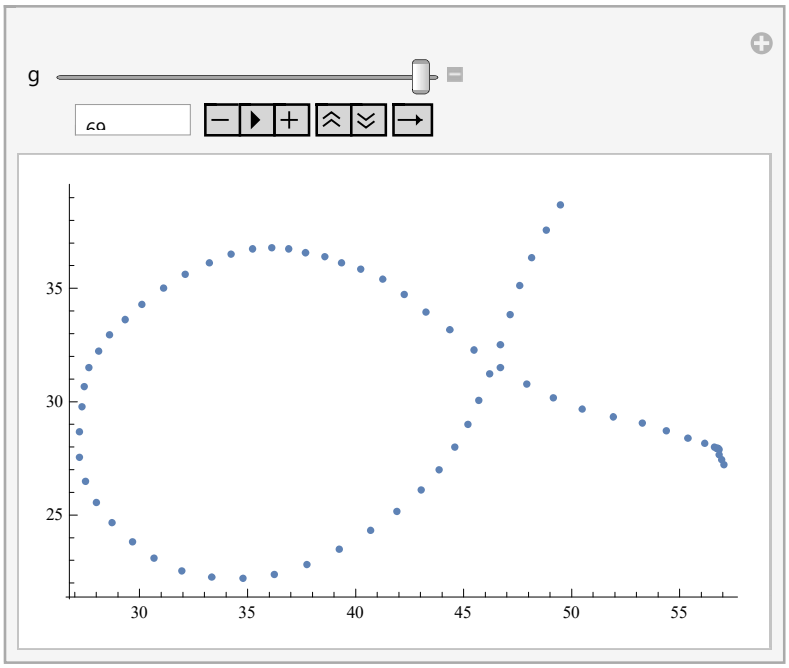
In[•]:=
```
ListPlot[AOGM[infinito], AxesLabel → {"X", "Y"}]
```
representación de lista etiqueta de ejes

Out[•]=



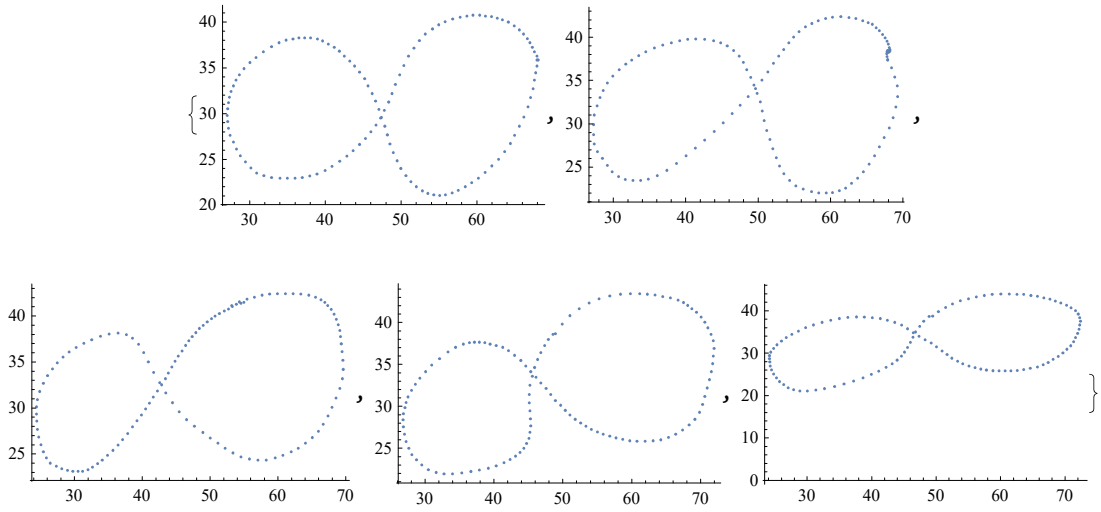
In[•]:=
```
stepByStepPlot[Table[AOGM[infinito][[i]], {i, 764, Length[AOGM[infinito]]}]]
               tabla                      longitud
```

Out[•]=




```
{ListPlot[Table[AOGM[infinito][[i]], {i, 14, 140}]],
 ListPlot[Table[AOGM[infinito][[i]], {i, 143, 266}]],
 ListPlot[Table[AOGM[infinito][[i]], {i, 284, 410}]],
 ListPlot[Table[AOGM[infinito][[i]], {i, 530, 652}]],
 ListPlot[Table[AOGM[infinito][[i]], {i, 652, 764}]]
}
```

Out[•]=



```
In[•]:= AOGM = Append[AOGM, ciclosInfinito → {
          añade
          Table[AOGM[infinito][[i]], {i, 14, 140}],
          tabla
          Table[AOGM[infinito][[i]], {i, 143, 266}],
          tabla
          Table[AOGM[infinito][[i]], {i, 284, 410}],
          tabla
          Table[AOGM[infinito][[i]], {i, 530, 652}],
          tabla
          Table[AOGM[infinito][[i]], {i, 652, 764}]
          tabla
         }];
```

Círculos promedio

4.- Círculo promedio

Las licenciadas, del hospital general, me comentaron que el ejercicio que abarcaba más grados de libertad era la trayectoria infinito. Por lo que, se buscará que el circulo abarque las siguientes trayectorias: vertical, circular e infinito.

De los ciclos obtenidos en el punto anterior, se elegirán los tres mejores definidos (punto de inicio es el mismo que el punto final)

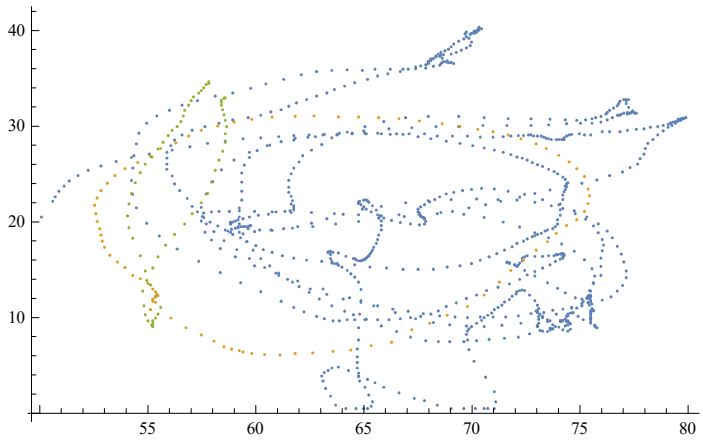
■ MESRM

En este paciente, los ciclos más definidos fueron los de la trayectoria circular

■ Ciclo 1

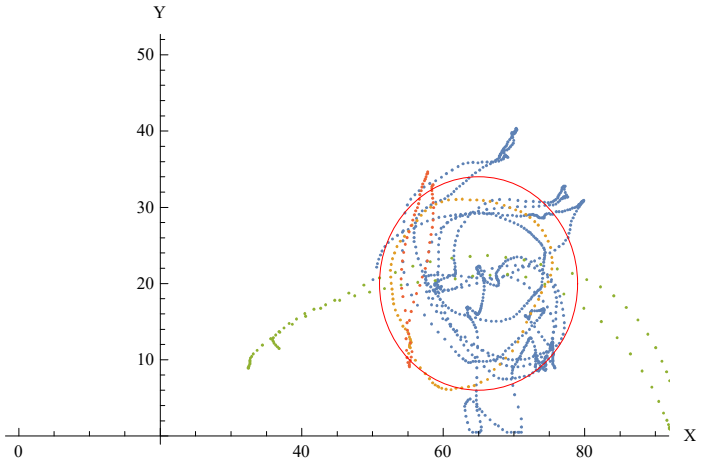
```
In[•]:= ListPlot[{MESRM[infinito], MESRM[ciclosCircular][[1]], MESRM[ciclosVertical][[3]]}]
        representación de lista
```

Out[•]=



In[•]:=
```
Show[ListPlot[{MESRM[infinito], MESRM[ciclosCircular][[1]], MESRM[ciclosHorizontal][[2]], MESRM[ciclosVertical][[3]]}],
Graphics[{Red, Circle[{65, 20}, 14]}], PlotRange → {{0, 90}, {0, 50}}, AxesLabel → {"X", "Y"}, AxesOrigin → {20, 0}]
```
mue·· representación de lista
gráfico rojo círculo rango de representación etiqueta de ejes origen de ejes

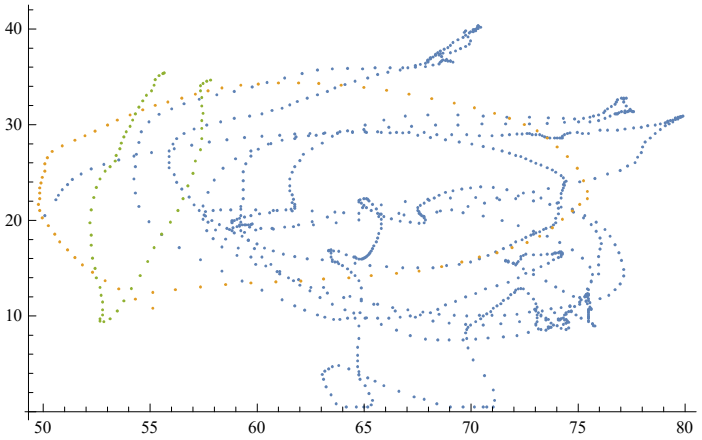
Out[•]=



- **Ciclo 2**

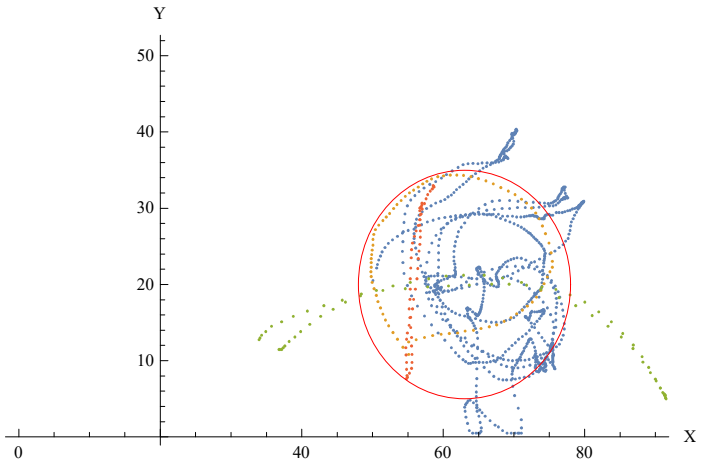
In[•]:=
```
ListPlot[{MESRM[infinito], MESRM[ciclosCircular][[2]], MESRM[ciclosVertical][[4]]}]
```
representación de lista

Out[•]=



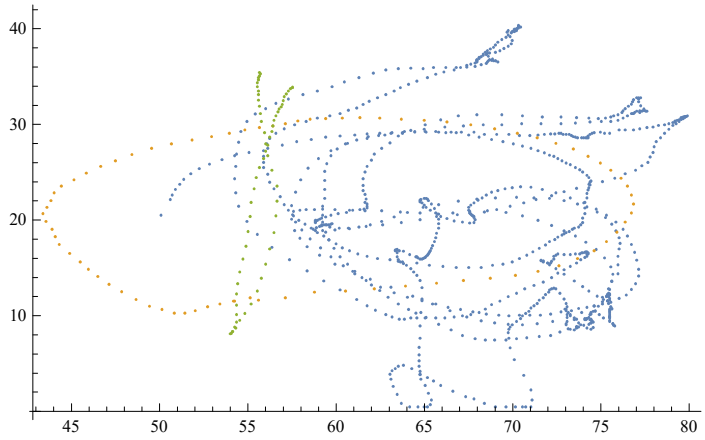
In[•]:=
```
Show[ListPlot[{MESRM[infinito], MESRM[ciclosCircular][[2]], MESRM[ciclosHorizontal][[3]], MESRM[ciclosVertical][[2]]}],
Graphics[{Red, Circle[{63, 20}, 15]}], PlotRange → {{0, 90}, {0, 50}}, AxesLabel → {"X", "Y"}, AxesOrigin → {20, 0}]
```
mue·· representación de lista
gráfico rojo círculo rango de representación etiqueta de ejes origen de ejes

Out[•]=

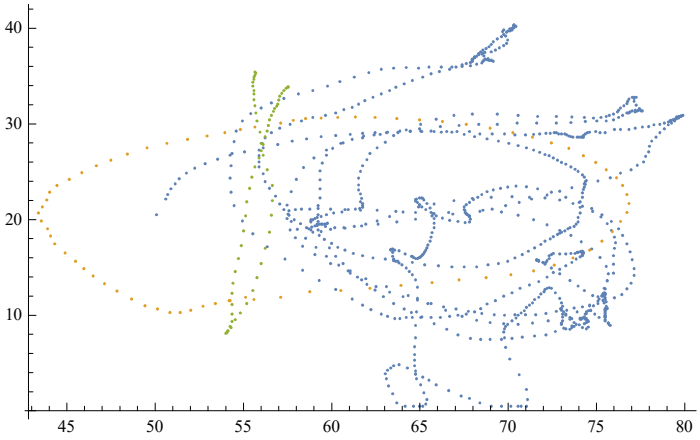


Ciclo 3

In[•]:=
```
ListPlot[{MESRM[infinito], MESRM[ciclosCircular][[6]], MESRM[ciclosVertical][[5]]}]
```
representación de lista

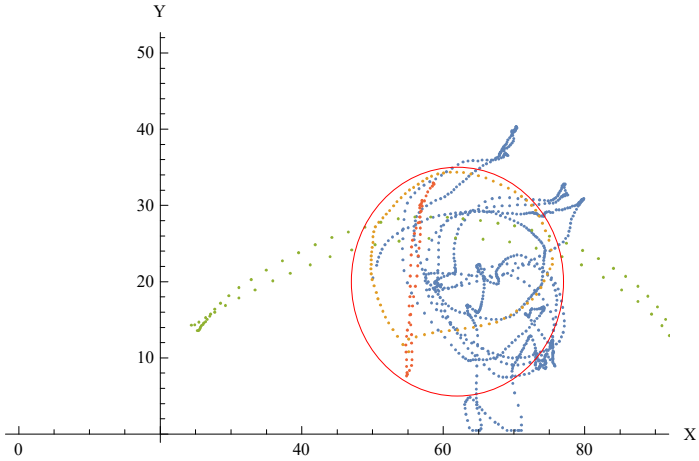


Out[•]=



In[•]:=
```
Show[ListPlot[{MESRM[infinito], MESRM[ciclosCircular][[2]], MESRM[ciclosHorizontal][[6]], MESRM[ciclosVertical][[2]]}],
```
mue·· representación de lista
```
 Graphics[{Red, Circle[{62, 20}, 15]}], PlotRange → {{0, 90}, {0, 50}}, AxesLabel → {"X", "Y"}, AxesOrigin → {20, 0}]
```
gráfico rojo círculo rango de representación etiqueta de ejes origen de ejes

Out[•]=



Promedio centro círculo y radio

In[•]:=
```
getMean[{{{65, 20}, 14}, {{63, 20}, 15}, {{62, 20}, 15}}]
```

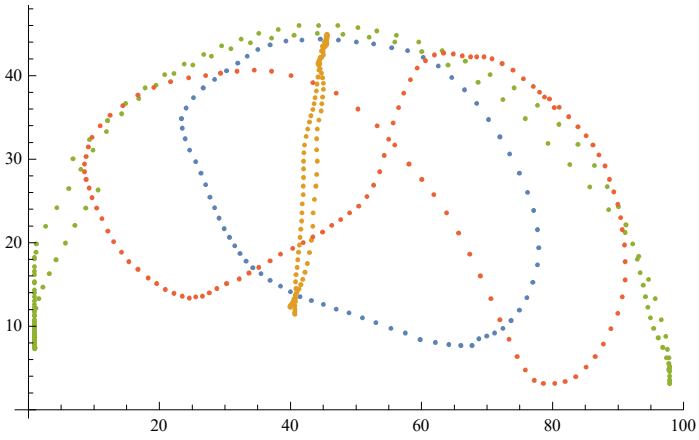
Out[•]=
```
{{63.3333, 20.}, 14.6667}
```

FJLH

Ciclo 1

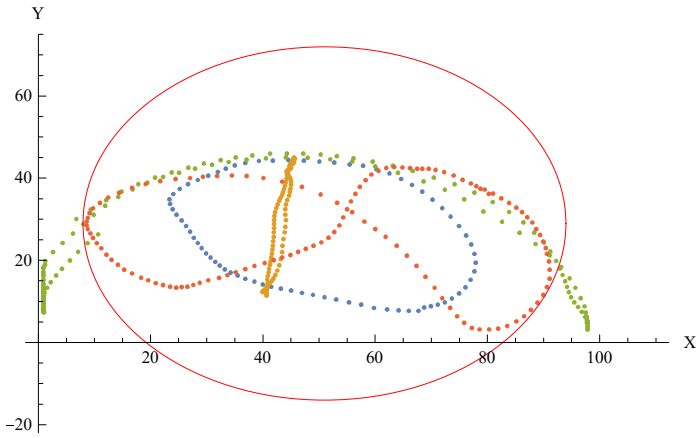
In[•]:=
```
ListPlot[{FJLH[ciclosCircular][[8]], FJLH[ciclosVertical][[3]], FJLH[ciclosHorizontal][[1]], FJLH[ciclosInfinito][[3]]}]
```
representación de lista

Out[•]=



In[•]:=
```
Show[
 ListPlot[{FJLH[ciclosCircular][[8]], FJLH[ciclosVertical][[3]], FJLH[ciclosHorizontal][[1]], FJLH[ciclosInfinito][[3]]}],
 Graphics[{Red, Circle[{51, 29}, 43]}], PlotRange → {{0, 110}, {-20, 70}}, AxesLabel → {"X", "Y"}, AxesOrigin → {0, 0}]
```
muestra · representación de lista · gráfico · rojo · círculo · rango de representación · etiqueta de ejes · origen de ejes

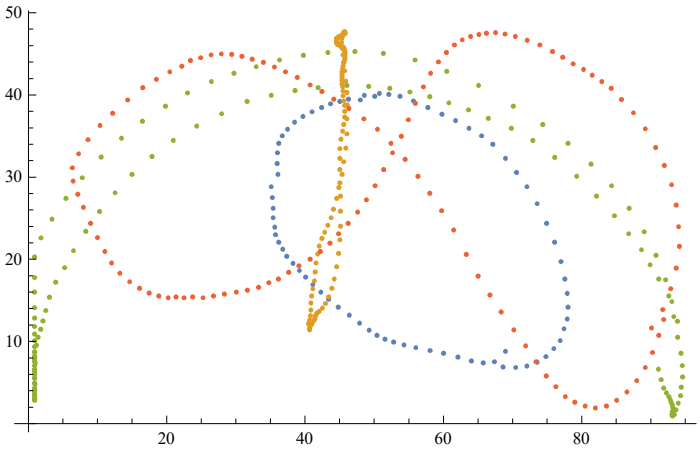
Out[•]=



Ciclo 2

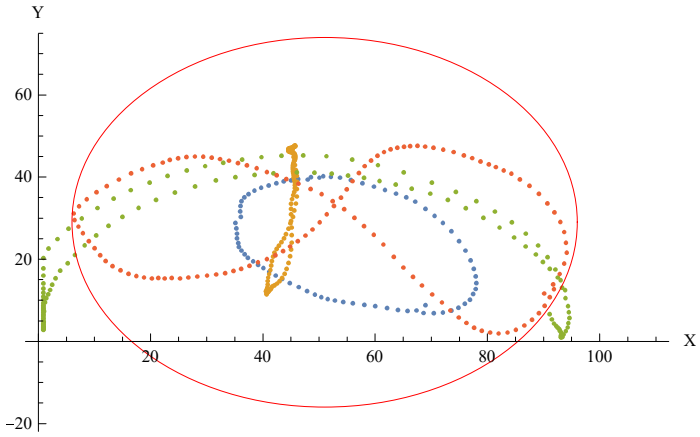
In[•]:=
```
ListPlot[{FJLH[ciclosCircular][[11]], FJLH[ciclosVertical][[4]], FJLH[ciclosHorizontal][[2]], FJLH[ciclosInfinito][[5]]}]
```
representación de lista

Out[•]=



In[•]:=
```
Show[ListPlot[
 {FJLH[ciclosCircular][[11]], FJLH[ciclosVertical][[4]], FJLH[ciclosHorizontal][[2]], FJLH[ciclosInfinito][[5]]}],
Graphics[{Red, Circle[{51, 29}, 45]}], PlotRange → {{0, 110}, {-20, 70}}, AxesLabel → {"X", "Y"}, AxesOrigin → {0, 0}]
```
mue·· representación de lista — rojo círculo — rango de representación — etiqueta de ejes — origen de ejes

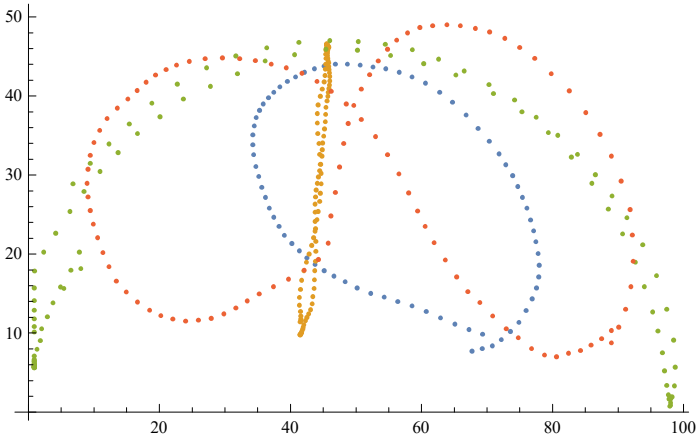
Out[•]=



- **Ciclo 3**

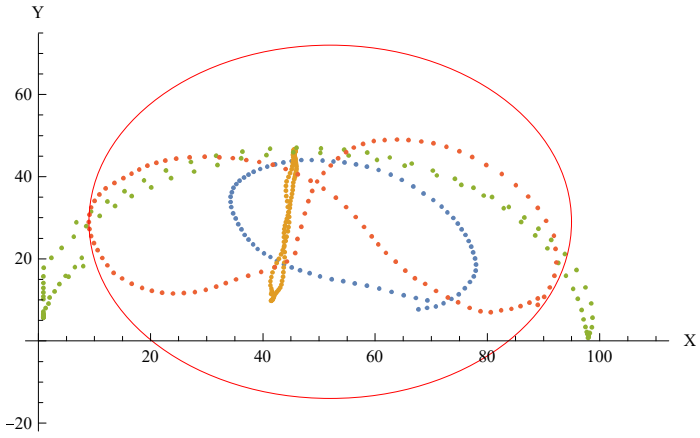
In[•]:=
```
ListPlot[{FJLH[ciclosCircular][[9]], FJLH[ciclosVertical][[6]], FJLH[ciclosHorizontal][[3]], FJLH[ciclosInfinito][[7]]}]
```
representación de lista

Out[•]=



In[•]:=
```
Show[
 ListPlot[{FJLH[ciclosCircular][[9]], FJLH[ciclosVertical][[6]], FJLH[ciclosHorizontal][[3]], FJLH[ciclosInfinito][[7]]}],
Graphics[{Red, Circle[{52, 29}, 43]}], PlotRange → {{0, 110}, {-20, 70}}, AxesLabel → {"X", "Y"}, AxesOrigin → {0, 0}]
```
muestra — representación de lista — gráfico — rojo — círculo — rango de representación — etiqueta de ejes — origen de ejes

Out[•]=



Promedio centro círculo y radio

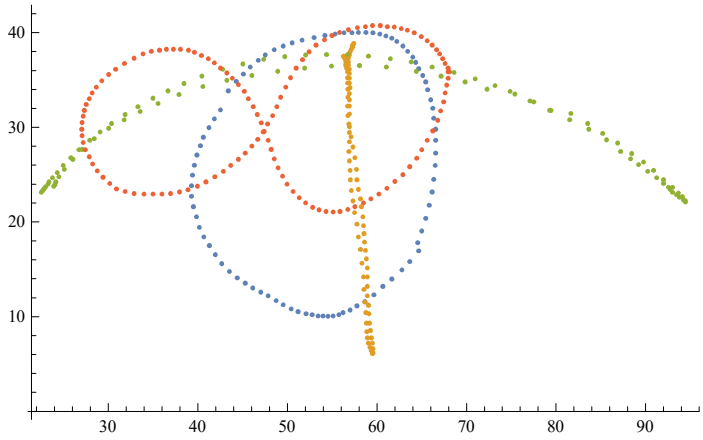
```
In[•]:= getMean[{{{51, 29}, 43}, {{51, 29}, 45}, {{52, 29}, 43}}]
Out[•]= {{51.3333, 29.}, 43.6667}
```

AOGM

Ciclo 1

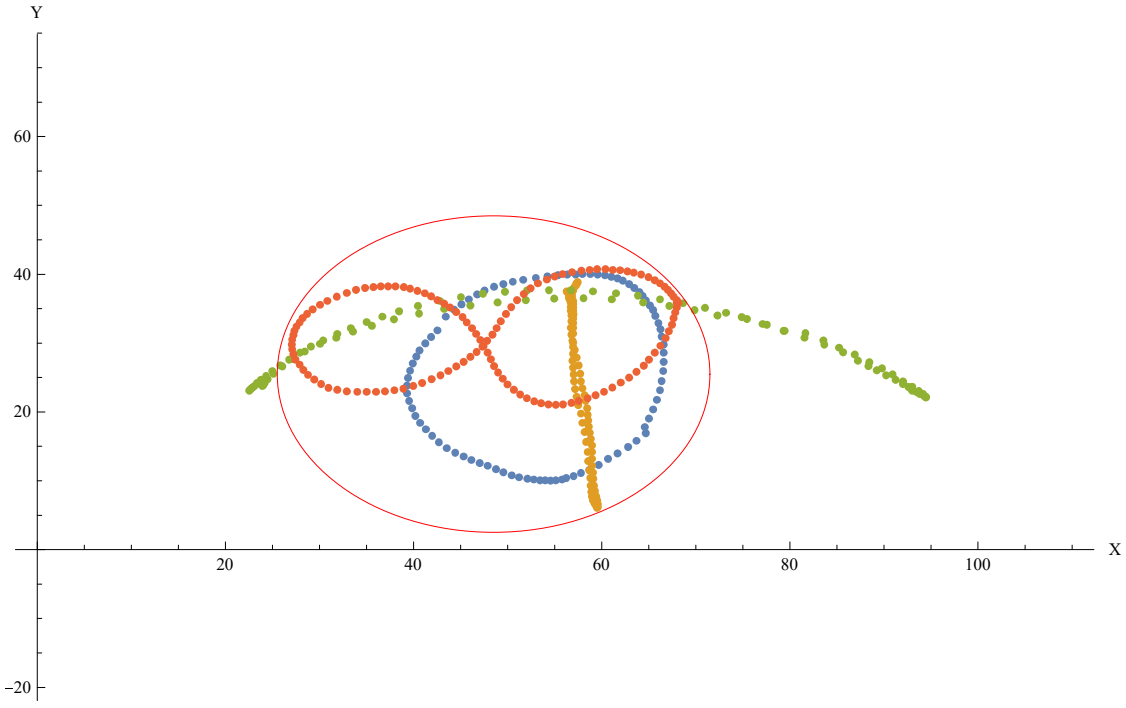
```
In[•]:= ListPlot[{AOGM[ciclosCircular][[1]], AOGM[ciclosVertical][[3]], AOGM[ciclosHorizontal][[2]], AOGM[ciclosInfinito][[1]]}]
        representación de lista
```

Out[•]=




```
In[•]:= Show[
        muestra
  ListPlot[{AOGM[ciclosCircular][[1]], AOGM[ciclosVertical][[3]], AOGM[ciclosHorizontal][[2]], AOGM[ciclosInfinito][[1]]}],
          representación de lista
  Graphics[{Red, Circle[{48.5, 25.5}, 23]}],
           gráfico   rojo  círculo
  PlotRange → {{0, 110}, {-20, 70}}, AxesLabel → {"X", "Y"}, AxesOrigin → {0, 0}]
                                      etiqueta de ejes        origen de ejes
```

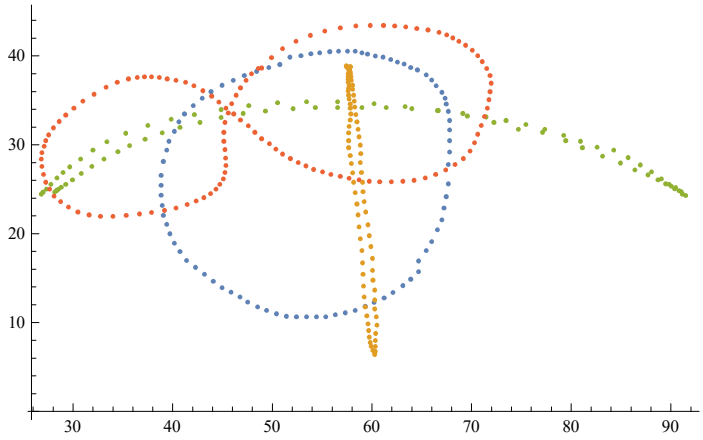
Out[•]=



Ciclo 2

```
In[•]:= ListPlot[{AOGM[ciclosCircular][[2]], AOGM[ciclosVertical][[4]], AOGM[ciclosHorizontal][[4]], AOGM[ciclosInfinito][[4]]}]
```
representación de lista

Out[•]=



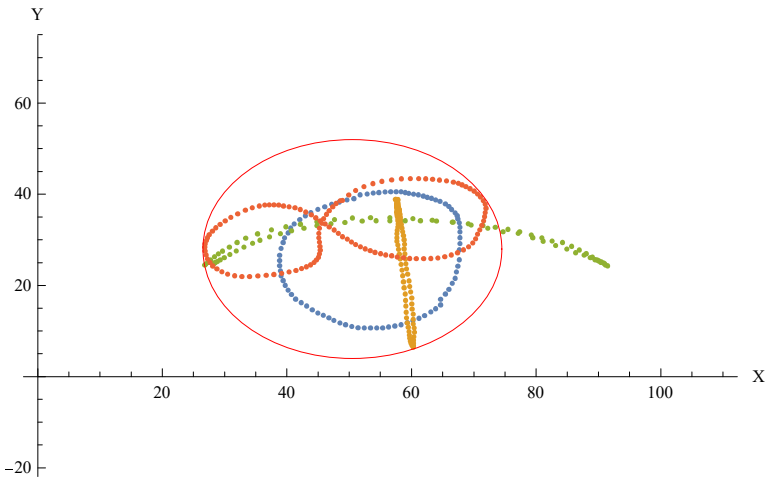

```
In[•]:= Show[
```
muestra

```
  ListPlot[{AOGM[ciclosCircular][[2]], AOGM[ciclosVertical][[4]], AOGM[ciclosHorizontal][[4]], AOGM[ciclosInfinito][[4]]}],
```
representación de lista

```
  Graphics[{Red, Circle[{50.5, 28}, 24]}], PlotRange → {{0, 110}, {-20, 70}},
```
gráfico rojo círculo rango de representación

```
  AxesLabel → {"X", "Y"}, AxesOrigin → {0, 0}]
```
origen de ejes

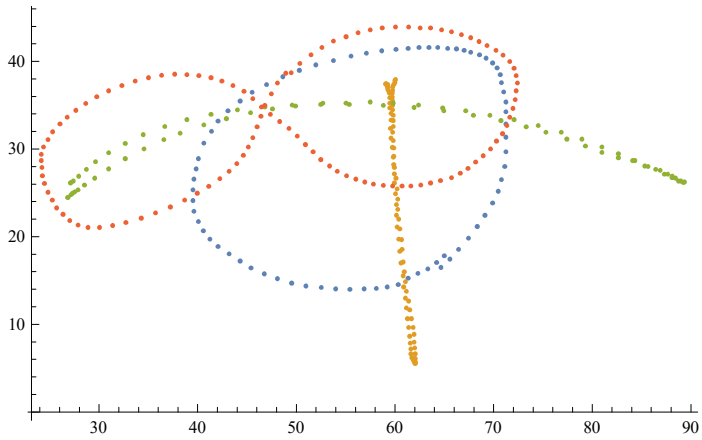
Out[•]=



Ciclo 3

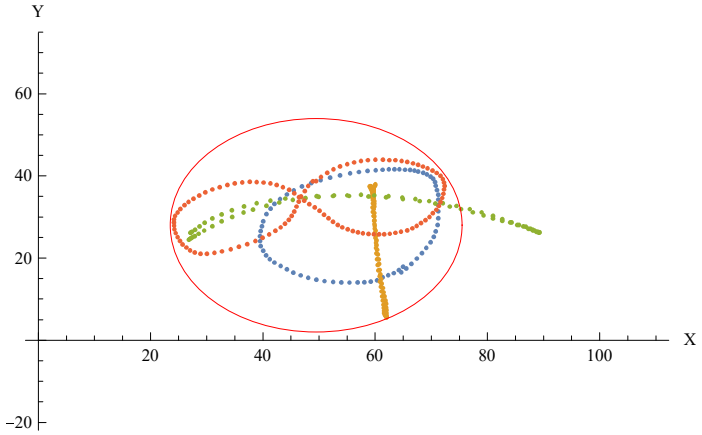
```
In[•]:= ListPlot[{AOGM[ciclosCircular][[5]], AOGM[ciclosVertical][[6]], AOGM[ciclosHorizontal][[5]], AOGM[ciclosInfinito][[5]]}]
```
representación de lista

Out[•]=



In[•]:=
```
Show[
 muestra
  ListPlot[{AOGM[ciclosCircular][[5]], AOGM[ciclosVertical][[6]], AOGM[ciclosHorizontal][[5]], AOGM[ciclosInfinito][[5]]},
  representación de lista
  Graphics[{Red, Circle[{49.5, 28}, 26]}], PlotRange → {{0, 110}, {-20, 70}},
  gráfico  rojo  círculo  rango de representación
  AxesLabel → {"X", "Y"}, AxesOrigin → {0, 0}]
  origen de ejes
```

Out[•]=



- **Promedio centro círculo y radio**

In[•]:=
```
getMean[{{{48.5, 25.5}, 23}, {{50.5, 28}, 24}, {{49.5, 28}, 26}}]
```

Out[•]=
```
{{49.5, 27.1667}, 24.3333}
```

Área ocupada por paciente en todas las trayectorias, arcos de movimiento y medidas antropométricas

Para obtener una gráfica comparativa

- Función rotar label

In[•]:=
```
rotateLabel[label_] := Style[Rotate[label, Pi / 4], 30, Bold, Opacity[0.2], FontFamily → "Helvetica"]
                       estilo  rota  número pi  negrita  opacidad  familia de tipo de letra

 (*Obtenida de la documentación de wolfram*)
```

- Función área rectangular

In[•]:=
```
getRectangleArea[data_] := Module[
                           módulo
 {b = Max[data[[All, 1]]] - Min[data[[All, 1]]], a = Max[data[[All, 2]]] - Min[data[[All, 2]]]},
   máximo  todo  mínimo  todo  máximo  todo  mínimo  todo


 b * a]
```

Agregando data

La información se registrará en el siguiente orden: Rangos de movimiento, Medidas antropométricas. Los primeros cinco datos son los rangos de movimiento de Hombro (flexión, abducción), Codo (Rotación interna y externa); las sexta esla suma de las medidas antropométricas y la última el área cuadrada que alcanzó el paciente [cm^2], entre diez para que todas los datos esten proporcionados en la gráfica

```
MESRM =
  Append[MESRM, dataGraf → {20, 30, 135, 20, 2, MESRM[m1] + MESRM[m2] + MESRM[m3], getRectangleArea[MESRM[todas]] / 10}];
  añade

FJLH = Append[FJLH, dataGraf → {180, 76, 112, 90, 80, FJLH[m1] + FJLH[m2] + FJLH[m3], getRectangleArea[FJLH[todas]] / 10}];
  añade

AOGM = Append[AOGM, dataGraf → {90, 90, 180, 58, 55, AOGM[m1] + AOGM[m2] + AOGM[m3], getRectangleArea[AOGM[todas]] / 10}];
  añade
```

Gráfica

```
In[•]:= BarChart[{MESRM[dataGraf], FJLH[dataGraf], AOGM[dataGraf]}, ChartStyle → 24,
          diagrama de barras                                        estilo de diagrama
  LabelingFunction → (Placed[#, Above] &), ChartLegends → {"(A)HombroFlex", "(B)HombroAbd",
    función de etiquetado  colocado  encima  leyendas de diagrama
    "(C)CodoFlex", "(D)CodoRotInter", "(E)CodoRotExtern", "(F)SumMedAntrop", "(G)AreaCuadra"},
      constante  deriva  número e
  ChartLabels → {Placed[{"MESRM", "FJLH", "AOGM"}, Center, rotateLabel], {"A", "B", "C", "D", "E", "F", "G"}}]
    etiquetas de diag···  colocado  centro  con··  der···  número e
```

Out[•]=