Funciones auxiliares

- Restricciones de diseño

In[86]:=
```
restrictionDesign =
  (2 + Sqrt[2]) l1 ≤ Sqrt[2] l2 && l1 + (Sqrt[2] (l2 - l1))/2 < l2 && l1 < Sqrt[2] (l2 - l1) < l2 && l1 < l2 && rmic ≥ rpro;
```

- Función a evaluar/optimizar

In[87]:=
```
LCI[b1_, b2_] := Quiet[NIntegrate[1/KtraDes /. {l1 → b1, l2 → b2}, {ωt, 0, 2 π}]]
```

- Obtener números Random

In[88]:=
```
getRandomPoints[{l1min_, l1max_}, {l2min_, l2max_}] := Module[
  {ptos, check, i},
  ptos = Table[{RandomReal[{l1min, l1max}], RandomReal[{l2min, l2max}]}, 3];
  check = Table[checkRestricciones[ptos[[i]]], {i, Length[ptos]}];

  Do[
   While[check[[i]] == 0,
    ptos[[i]] = {RandomReal[{l1min, l1max}], RandomReal[{l2min, l2max}]};
    check[[i]] = checkRestricciones[ptos[[i]]];
    ]
   , {i, Length[ptos]}];

  ptos
 ]
```

- Ordenar puntos

In[89]:=
```
ordenarPoints[ptos_] := Module[
  {ptosWithFuncValue, i},
  If[Length[ptos[[1]]] == 2,
   ptosWithFuncValue = Table[{ptos[[i, 1]], ptos[[i, 2]], LCI[ptos[[i, 1]], ptos[[i, 2]]]}, {i, 1, Length[ptos]}];
  ];
  If[Length[ptos[[1]]] == 3,
   ptosWithFuncValue = ptos;
  ];

  Sort[ptosWithFuncValue, #1[[3]] < #2[[3]] &]
 ]
```

- Nuevos puntos

In[90]:=
```
(*Generar nuevos puntos a partir del best*)
newPoints[points_, factor_] := Module[
   {res, aux, p, pn1, pn2, pn3},
   p = Table[points[[i, {1, 2}]], {i, Length[points]}];
   aux = 0;

   aux = Table[p[[3]] + factor (p[[i]] - p[[3]]), {i, 1, 2}];

   pn1 = Flatten[{p[[3]], LCI[p[[3, 1]], p[[3, 2]]]}];
   pn2 = Flatten[{aux[[1]], LCI[aux[[1, 1]], aux[[1, 2]]]}];
   pn3 = Flatten[{aux[[2]], LCI[aux[[2, 1]], aux[[2, 2]]]}];

   {pn1, pn2, pn3}
  ];
```

- Puntos para la nueva iteración

In[91]:=
```
addNewPoint[puntos_, new_] := Module[
   {P1 = puntos[[2]], P2 = puntos[[3]], P3 = new},

   ordenarPoints[{P1, P2, P3}]
  ];
```

- Checar si Xpunto esta dentro del área de diseño

In[92]:=
```
checkRestricciones[ptoProye_] := Module[
   {flag, thisl1 = ptoProye[[1]], thisl2 = ptoProye[[2]]},
   flag = 0;
   (*If[0<l1<7&&0<l2<17&&l2>l1&&l1≤Sqrt[2]/2 (l2-l1)&&Sqrt[2]/2 (l2-l1)+l1+l2<30,*)
   If[restrictionDesign /. {l1 -> thisl1, l2 → thisl2},
        flag = 1
   ];
   flag]
```

- GetXmeanXr

In[93]:=
```
getXmeanXr[p_, factor_] := Module[
   {Xmean, Xr, p1 = p[[1, {1, 2}]], p2 = p[[2, {1, 2}]], p3 = p[[3, {1, 2}]]},

   Xmean = Mean[{p2, p3}];
   Xr = Xmean + factor (Xmean - p1); (*expansion*)

   {Xmean, Flatten[{Xr, LCI[Xr[[1]], Xr[[2]]]}]}
  ]
```

- Estatus de la projección

In[94]:=
```
getProjectionStatus[pts_, Xr_, Xmean_, α_, ξ_] := Module[
  {check, thisPts = pts, thisXr = Xr, thisXmean = Xmean, inf, newPtos},

  check = checkRestricciones[Xr];
  (*Para saber el status de la proyección*)
  If[check == 0,
   inf = "\nLa proyección: " <> ToString[Xr // N] <> ", de los puntos " <>
     ToString[thisPts // N] <> ", esta fuera del area de diseño. Se cambia por nuevos ptos.",
   inf = "\nLa proyección: " <> ToString[Xr // N] <> ", de los puntos " <>
     ToString[thisPts // N] <> ", esta dentro del area de diseño"
  ];

  (*Si la proyección esta fuera del área de diseño se crean nuevos pts*)
  While[check == 0,
   thisPts = newPoints[thisPts, ξ];
   thisPts = ordenarPoints[thisPts];
   {thisXmean, thisXr} = getXmeanXr[thisPts, α];
   check = checkRestricciones[thisXr];
  ];

  {inf, thisPts, thisXmean, thisXr}]
```

- Realizar una proyección desde Xmean con un factor

In[95]:=
```
getProjection[Xmean_, factor_, Xr_] := Module[
  {thisXr = Xr[[{1, 2}]], thisXe},

  thisXe = Xmean + factor (thisXr - Xmean);

  Flatten[{thisXe, LCI[thisXe[[1]], thisXe[[2]]]}]
 ]
```

- Dibujar los tríangulos del método.

In[96]:=
```
getTriangle[points_, Xmean_, proyecti_, proyectf_] := Module[
  {thisPoints, best = points[[3, {1, 2}]],
   good = points[[2, {1, 2}]], worst = points[[1, {1, 2}]], thisProyi = proyecti[[{1, 2}]],
   thisProyf = proyectf[[{1, 2}]], vpf, color, graphPoints, vectors, labels, graphTriangle},
  thisPoints = {worst, good, best, thisProyi, thisProyf, Xmean};
  color = {{1, 0, 0}, {1, 1, 0}, {0, 1, 0}, {0, 0, 1}, {1, 0, 1}, {0.57, 0.57, 0.59}, {1, 1 / 2, 0}};
  (*red,yellow,green,blue,purple,gray,orange*)
  vpf = {points[[1, 3]], points[[2, 3]], points[[3, 3]], proyecti[[3]], proyectf[[3]]};

  graphTriangle = Graphics[
    {EdgeForm[Directive[Thick, Black, Dotted]], FaceForm[None], Triangle[thisPoints[[{1, 2, 3}]]]}];
  graphPoints = Table[Graphics[{PointSize[0.1], RGBColor[color[[i]]], Point[thisPoints[[i]]]}],
    {i, Length[thisPoints]}];
  vectors = {Graphics[{Thick, {Arrowheads[Large], Arrow[{Xmean, thisProyi}]}}],
    Graphics[{Thick, {Arrowheads[Large], Arrow[{Xmean, thisProyf}]}}]};
  labels = Table[Graphics[Text[Style[ToString[vpf[[k]]], Bold, color[[7]], 15], thisPoints[[k]]]],
    {k, 1, Length[vpf]}];

  Show[graphTriangle, graphPoints, vectors, labels]
 ]
```

Método

In[112]:=
```
NelderMeadMethod[ptosIniciales_, α_, β_, γ_, ξ_, it_] := Module[
  {xmean, xr, xe, xoc, xic, Xnew, ptosInicio,
   infoIteracion, i = 1, new, info = {0}, flag = 0, best, good, worst, graphSystem},

  ptosInicio = ordenarPoints[ptosIniciales];
  graphSystem = Table[0, it];

  While[i ≤ it,
   (*Get Xmean y Xr-Proyectar el peor punto-*)
   {xmean, xr} = getXmeanXr[ptosInicio, α];

   (*Checar si la proyección esta dentro del area de diseño*)
   {infoIteracion, ptosInicio, xmean, xr} = getProjectionStatus[ptosInicio, xr, xmean, α, ξ];
       infoIteracion = infoIteracion <> "\n==> Puntos iteracion " <> ToString[i] <>
     ": " <> ToString[ptosInicio // N] <> " \n\t%% pto reflejado: " <> ToString[xr // N];

   (*Puntos:*)
   best = ptosInicio[[3, 3]];
   good = ptosInicio[[2, 3]];
   worst = ptosInicio[[1, 3]];

   (*Primer caso*)
   If[xr[[3]] > best,
    infoIteracion = infoIteracion <> "\n\t->Primer Caso<-";
    xe = getProjection[xmean, β, xr];
        If[(xe[[3]] > xr[[3]]) && (checkRestricciones[xe] != 0),
             new = xe;
             infoIteracion = infoIteracion <> "\t: Con una expansión" <> ToString[new // N],

             new = xr
         ];
    flag = 1;
   ];

   (*Segundo caso*)
   If[good < xr[[3]] ≤ best,
    infoIteracion = infoIteracion <> "\n\t->Segundo Caso<-";
    new = xr;
    flag = 1;
   ];

   (*Tercer caso*)
   If[xr[[3]] ≤ good,
    infoIteracion = infoIteracion <> "\n\t->Tercer Caso: ";
    (*En dirección del punto reflejado xr*)
        If[xr[[3]] > worst,
             infoIteracion = infoIteracion <> "---Direccion pto reflejado---";
             xoc = getProjection[xmean, γ, xr];
                 If[xoc[[3]] > xr[[3]],
                      new = xoc;
                      infoIteracion = infoIteracion <> " El cual es: " <> ToString[new // N];
```

```
                    flag = 1,

                    ptosInicio = newPoints[ptosInicio, ξ];
                    infoIteracion = infoIteracion <> "\n\tPero con pts nuevos: " <>
                      ToString[ptosInicio // N];
                ];
        ];

        (*En dirección del worst point*)
        If[worst ≥ xr[[3]],
            infoIteracion = infoIteracion <> "---Direccion worst pto---";
            xic = getProjection[xmean, -γ, xr];
                If[xic[[3]] > worst,
                    new = xic;
                    infoIteracion = infoIteracion <> " El cual es: " <> ToString[new // N];
                    flag = 1,

                    ptosInicio = newPoints[ptosInicio, ξ];
                    infoIteracion = infoIteracion <> "\n\tPero con ptos nuevos: " <>
                      ToString[ptosInicio // N]
                ];
        ];
    ];

    (*Si se creo un nuevo punto agrgarlo a los otros pa la nueva iteración y dibujar el sistema*)
    If[flag == 1,
        graphSystem[[i]] = getTriangle[ptosInicio, xmean, xr, new];
        ptosInicio = addNewPoint[ptosInicio, new],

        ptosInicio = ordenarPoints[ptosInicio];
    ];

    (*Restableciendo variables*)
    flag = 0;
    info = Append[info, infoIteracion];
    i += 1;
  ];

  {Drop[info, 1], DeleteCases[graphSystem, 0]}]
```

Pruebas

- **Buenos puntos**

Converge a 2.58984 --> {14.4753, 35.072}

In[106]:=
```
opc1 = {{17.210883923780564`, 42.86004293357725`},
    {26.71104944896245`, 78.33647702220259`}, {15.401371836171872`, 49.014299168856766`}};
```

COnverge a 2.46496 --> {36.2072, 87.412}

```
Opc2 = {{32.75068205098804`, 87.05891014484499`},
    {21.543140565272402`, 64.93606301113412`}, {23.50801209252834`, 75.40658272207367`}};
```

In[113]:=
```
(*Para los puntos iniciales*)
ptosRandom = getRandomPoints[{15, 40}, {30, 90}]
```

Out[113]=
```
{{20.5861, 64.809}, {36.348, 89.966}, {27.9263, 80.6471}}
```

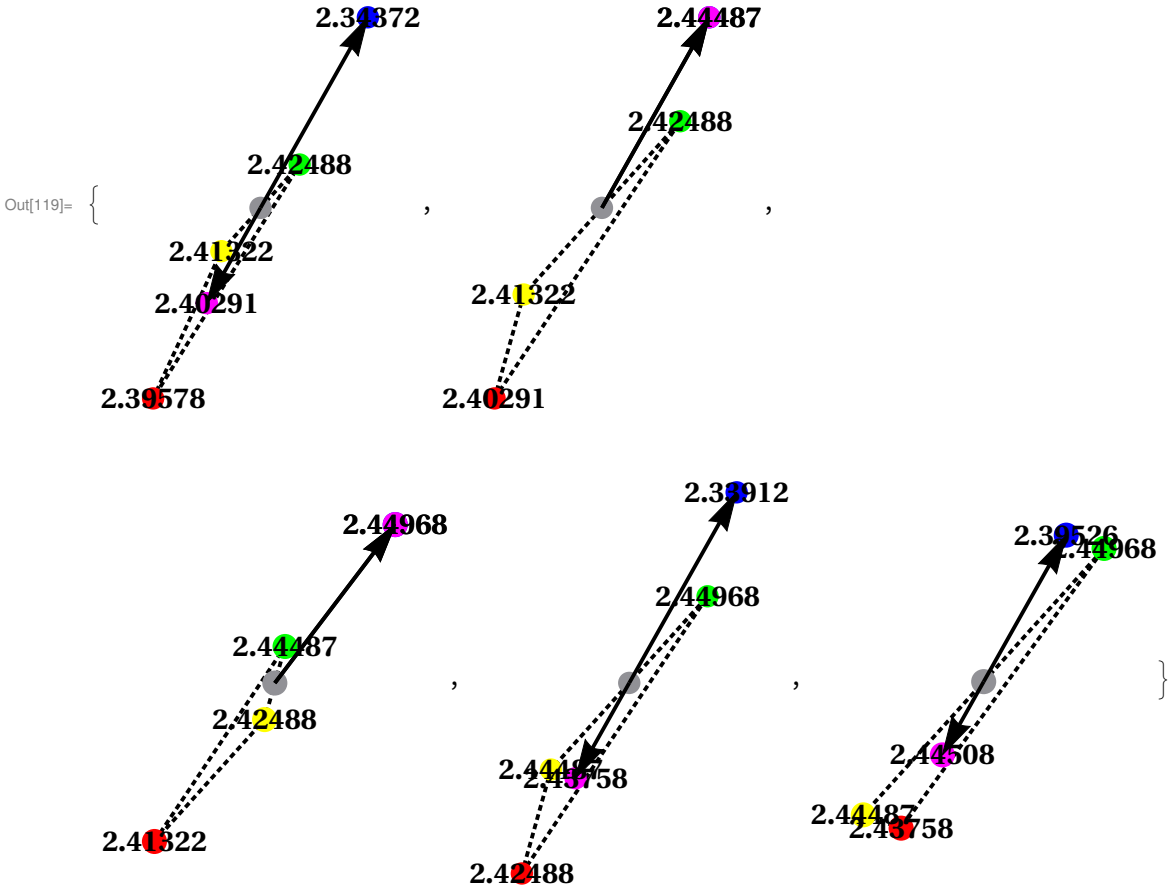
In[117]:=
```
{textInfo, graphicalInfo} = NelderMeadMethod[ptosRandom, 1, 2, 1 / 2, 1 / 2, 5];
```

In[118]:=
```
textInfo
```

Out[118]=
```
{
La proyección: {12.1644, 55.4902, 2.38899}, de los
   puntos {{36.348, 89.966, 2.33438}, {27.9263, 80.6471, 2.35173}, {20.5861,
   64.809, 2.39578}}, esta fuera del area de diseño. Se cambia por nuevos ptos.
==> Puntos iteracion 1: {{20.5861, 64.809, 2.39578}, {24.2562,
   72.7281, 2.41322}, {28.4671, 77.3875, 2.42488}}
    %% pto reflejado: {32.1372, 85.3066, 2.34372}
    ->Tercer Caso: ---Direccion worst pto--- El cual es: {23.4739, 69.9334, 2.40291},
La proyección: {29.2494, 80.1822, 2.44487}, de los puntos {{23.4739, 69.9334, 2.40291},
   {24.2562, 72.7281, 2.41322}, {28.4671, 77.3875, 2.42488}}, esta dentro del area de diseño
==> Puntos iteracion 2: {{23.4739, 69.9334, 2.40291}, {24.2562,
   72.7281, 2.41322}, {28.4671, 77.3875, 2.42488}}
    %% pto reflejado: {29.2494, 80.1822, 2.44487}
    ->Primer Caso<-,
La proyección: {33.4603, 84.8416, 2.44968}, de los puntos {{24.2562, 72.7281, 2.41322},
   {28.4671, 77.3875, 2.42488}, {29.2494, 80.1822, 2.44487}}, esta dentro del area de diseño
==> Puntos iteracion 3: {{24.2562, 72.7281, 2.41322}, {28.4671,
   77.3875, 2.42488}, {29.2494, 80.1822, 2.44487}}
    %% pto reflejado: {33.4603, 84.8416, 2.44968}
    ->Primer Caso<-,
La proyección: {34.2426, 87.6363, 2.33912}, de los puntos {{28.4671, 77.3875, 2.42488},
   {29.2494, 80.1822, 2.44487}, {33.4603, 84.8416, 2.44968}}, esta dentro del area de diseño
==> Puntos iteracion 4: {{28.4671, 77.3875, 2.42488}, {29.2494,
   80.1822, 2.44487}, {33.4603, 84.8416, 2.44968}}
    %% pto reflejado: {34.2426, 87.6363, 2.33912}
    ->Tercer Caso: ---Direccion worst pto--- El cual es: {29.911, 79.9497, 2.43758},
La proyección: {32.7987, 85.0741, 2.39526}, de los puntos {{29.911, 79.9497, 2.43758},
   {29.2494, 80.1822, 2.44487}, {33.4603, 84.8416, 2.44968}}, esta dentro del area de diseño
==> Puntos iteracion 5: {{29.911, 79.9497, 2.43758}, {29.2494,
   80.1822, 2.44487}, {33.4603, 84.8416, 2.44968}}
    %% pto reflejado: {32.7987, 85.0741, 2.39526}
    ->Tercer Caso: ---Direccion worst pto--- El cual es: {30.6329, 81.2308, 2.44508}}
```

In[119]:= graphicalInfo

Out[119]=



Referencias

1. Gao, F., & Han, L. (2012). Implementing the Nelder-Mead simplex algorithm with adaptive parameters. Computational Optimization and Applications, 51(1), 259-277. Doi: 10.1007/s10589-010-9329-3
2. Cantarella J. (2010). Nelder-Mead Method. Disponible en: http://www.jasoncantarella.com/downloads/NelderMeadProof.pdf