Exercice 4

1) Après avoir calculé la dérivée (en θ) de l'expression, il est possible de définir les deux fonctions du programme `f` (pour l'équation à résoudre) et `der` (sa dérivée). Le programme est similaire à celui développé dans le cours, en passant néanmoins β comme argument.

Python

```
# Définition de l'équation non linéaire
def f(x,Beta):
    global h,L,R,d
    return (d*cos(x)+R*cos(Beta))**2+\
           (d*sin(x)-h+R*sin(Beta))**2-L**2

# Définition de la dérivée
def der(x,Beta):
    global h,L,R,d
    return -2*d*(d*cos(x)+R*cos(Beta))*sin(x)+\
            2*d*(d*sin(x)-h+R*sin(Beta))*cos(x)

# Recherche de solution par la méthode de Newton
def resolution(f, der, x0, epsilon, Beta):
    x=x0  # Initialisation à la première estimation fournie
    i=0
    # Tant que le critère de convergence n'est pas satisfait
    while (abs(f(x,Beta))>epsilon):
        x=x-f(x,Beta)/der(x,Beta)
        i=i+1
    return x # On renvoie la dernière estimation
```

Scilab

```
// Définition de l'équation non linéaire
function y=f(x,Beta)
    global h,L,R,d
    y=(d*cos(x)+R*cos(Beta))^2+...
        (d*sin(x)-h+R*sin(Beta))^2-L^2
endfunction

// Définition de la dérivée
function y=der(x,Beta)
    y=-2*d*(d*cos(x)+R*cos(Beta))*sin(x)+...
        2*d*(d*sin(x)-h+R*sin(Beta))*cos(x)
endfunction

// Recherche de solution par la méthode de Newton
function xsol=resolution(f, der, x0, epsilon, Beta)
    x=x0  // Initialisation à la première estimation fournie
    i=0
    // Tant que le critère de convergence n'est pas satisfait
    while (abs(f(x,Beta))>epsilon)
        x=x-f(x,Beta)/der(x,Beta)
        i=i+1
    end
    xsol=x // On renvoie la dernière estimation
    disp(i)
endfunction
```

2) Le programme doit maintenant traiter le cas des 10 positions pour les 10 valeurs de β imposées. Une boucle sur 10 valeurs de β est engagée et pour chaque valeur de β l'angle θ est calculé et affiché.

Python

```
for Beta in linspace(-0.5,0.3,10):  # Boucle sur les angles Beta
    # Calcul de theta
    theta=resolution(f,der,0,1e-12,Beta)
    print(theta)
```

Scilab

```
for Beta=-0.5:0.1:0.3  // Boucle sur les angles Beta
    // Calcul de theta
    theta=resolution(f,der,0,1e-12,Beta)
    disp(theta)
end
```

3) En complétant le programme précédent, pour chaque valeur de β l'angle θ est calculé, ce qui permet de calculer les deux matrices de rotation et les polylignes après rotation, et enfin de tracer les polylignes.

Python

```
clf()  # Effacement de la figure
plot([-15,5,5,-15],[-10,-10,50,50],"m") # Tracé du bâti
for Beta in linspace(-0.5,0.3,10):  # Boucle sur les angles Beta
    # Calcul de theta
    theta=resolution(f,der,0,1e-12,Beta)
    # Calcul des matrices de rotation
    Rt=array([[cos(theta),-sin(theta)],[sin(theta),cos(theta)]])
    Rb=array([[cos(Beta),-sin(Beta)],[sin(Beta),cos(Beta)]])
    # Rotation des pinces
    pince1r=dot(Rb,pince1)
    pince2r=dot(Rt,pince2)
    # Tracés des pinces après rotation (en bleu)
    plot(pince1r[0,:],pince1r[1,:],"b")
    plot(pince2r[0,:],pince2r[1,:]+h,"b")
    # Tracé de la bielle (en rouge)
    plot([pince1r[0,1],pince2r[0,0]],\
         [pince1r[1,1],pince2r[1,0]+h],"-or")

plot(0,0,"o") # Tracés de deux cercles aux articulations.
plot(0,h,"o")
```

Scilab

```
clf  // Effacement de la figure
plot([-15,5,5,-15],[-10,-10,50,50],"m") // tracé du bâti
for Beta=-0.5:0.1:0.3  // Boucle sur les angles Beta
    // Calcul de theta
    theta=resolution(f,der,0,1e-12,Beta)
    // Calcul des matrices de rotation
    Rt=[cos(theta),-sin(theta);sin(theta),cos(theta)]
    Rb=[cos(Beta),-sin(Beta);sin(Beta),cos(Beta)]
    // Rotation des pinces
    pince1r=Rb*pince1
    pince2r=Rt*pince2
    // Tracés des pinces après rotation (en bleu)
    plot(pince1r(1,:),pince1r(2,:),"b")
    plot(pince2r(1,:),pince2r(2,:)+h,"b")
    // Tracé de la bielle (en rouge)
    plot([pince1r(1,2),pince2r(1,1)],...
         [pince1r(2,2),pince2r(2,1)+h],"-or")
end
plot(0,0,"o") // Tracés de deux cercles aux articulations.
plot(0,h,"o")
```

La figure obtenue montre le mouvement du mécanisme (voir figure 3.38 page suivante).