DIOU Guillaume – EDMOND Cyril – METTEY Thomas

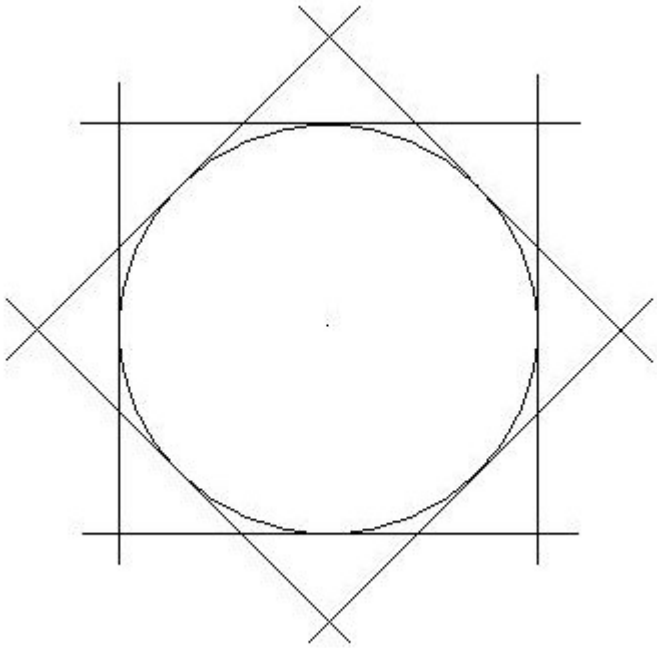
« UN MODÈLE DE DYNAMIQUE DES PIÉTONS »

1. **Analyse et compréhension**

1) On utilise la formule suivante pour calculer $¿v\vee¿$:

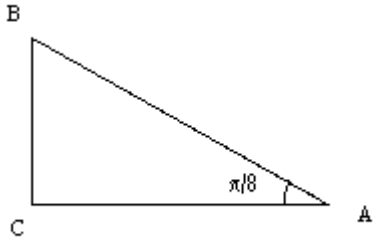
$$|v|=\sqrt{v_1^2+v_2^2}$$

On fait l'hypothèse que l'erreur que l'on peut avoir est l'erreur maximale possible. Grâce aux conditions sur les e_i on obtient la figure d'erreur suivante :



Cette figure est invariante par rotation de $\frac{\pi}{4}$

L'erreur maximale est donc atteinte dans le cas suivant. On obtient donc la figure ci-dessous :



On a donc $\cos\left(\frac{\pi}{8}\right) = \frac{rayon}{longueur\,maximale} = \frac{AC}{AB}$

D'où $AB = \frac{AC}{\cos\left(\frac{\pi}{8}\right)}$

L'erreur vaut donc $\frac{AB - AC}{AB} = \frac{\frac{1}{\cos\left(\frac{\pi}{8}\right)} - 1}{1} = 8.23$

Si Θ(y) = Ø, alors la vitesse du piéton en ce point ne peut vérifier aucune des contraintes. Soit sa vitesse est supérieure à V_{max}, soit il va percuter un obstacle, soit les deux.
Comme les contraintes sont imposées, Θ(y) = Ø est un résultat possible mais les piétons ne peuvent se trouver en ces points.

2) I(x,t) = min $\int_{t}^{td} ¿v(s)\vee ds$

Or, f(a+h) = f(a) + hf'(a) pour f suffisamment dérivable sur R et lorsque h->0.
On a donc dans ces conditions :

I(x,t+dt) = I(x,t) + dt*I'(x,t) pour dt->0

De plus :

I'(x,t) = $\frac{\partial I}{\partial t}(x,t) + \frac{\partial I}{\partial x} * ¿ \quad \frac{\partial x}{\partial t}(x,t)$

Dû à :

$\frac{\partial x}{\partial t}$ = v(t)

On en déduit

I(x,t+dt) – I(x,t) = min $\int_{t+dt}^{td} ¿v(s)\vee ds$ - min $\int_{t}^{td} ¿v(s)\vee ds$

I(x,t+dt) – I(x,t) = - min ($\int_t^{t+dt} ¿ v(s) \vee ds$))

Donc

$\frac{-1}{dt}$ * min ($\int_t^{t+dt} ¿ v(s) \vee ds$) = $\frac{\partial I}{\partial t}(x,t) + \frac{\partial I}{\partial x} * ¿$ v(t)

Or

$\int_t^{t+dt} |v(s)| ds = \int_t^{t+dt} |\dot{y}(s)| ds = \left[|x(s)| \right]_t^{t+dt} = |x(t+dt)| - |x(t)|$

$\frac{-1}{dt}$ * min ($\int_t^{t+dt} ¿ v(s) \vee ds$) = - min ($\frac{|x(t+dt)| - |x(t)|}{dt} ¿$ = - min ($¿ v(t) \vee ¿$

$\frac{\partial I}{\partial t}(x,t) = ¿$ - min ($¿ v(t) \vee ¿$ - $\frac{\partial I}{\partial x} * ¿$ v(t)

Où $\frac{\partial I}{\partial x} = p$

$\frac{\partial I}{\partial t}(x,t) = H\left(x, \frac{\partial I}{\partial t}(x,t)\right)$ avec $H(y,p) = min_{¿ \vee v(t) \vee ¿}^{¿}$ - $p * ¿$ v(t))

3) On a $H(y,p) = min_{¿ \vee v \vee ¿}^{¿}$ - $p * ¿$ v) sur v ϵ Θ(y)

On pose z = $¿ v \vee ¿$

$H(y,p) = min_{¿}^{¿ z}$ - $p * ¿$ v) sur z et v

Par définition on a : ei.v ≤ $¿ v \vee ¿$ ⇒ ei.v ≤ z

⇒ E.v ≤ z.**1**

On a également dans les contraintes : pour tout j ϵ 1,….,J(y)

$n_j(y).v \leq v_j(y)$

$\forall$ N(y).v $\leq$ $v_j(y)$

De plus, on a 0 $\leq \vee v \vee ¿$ $\leq$ V_{max}

Donc 0 $\leq z$ $\leq$ V_{max}

Notons V(y,p) la solution en v de (10)

4) $H = min_{z,v}(z + p.v) = max(-(z + p.v))$

$$\begin{vmatrix} E.v - z.1 \leq 0 \\ N(y).v - v(y) \leq 0 \\ z - V_{max} \leq 0 \\ z \geq 0 \end{vmatrix}$$

$\Leftrightarrow$ $H = min_{z,v}(z + p.v) = max(-(z + p.v))$

$$\begin{vmatrix} E.v - z.1 + \alpha = 0 \\ N(y).v - v(y) + \beta = 0 \\ z - V_{max} + \gamma = 0 \\ z - \delta + \varepsilon = 0 \\ \alpha, \beta, \gamma, \delta, \varepsilon \geq 0 \end{vmatrix}$$

$\Leftrightarrow$ $H = min_{z,v}(z + p.v) = max(-(z + p.v))$

$$\begin{vmatrix} E.v - z.1 + \alpha = 0 \\ N(y).v + \beta = v(y) \\ z + \gamma = V_{max} \\ z - \delta + \varepsilon = 0 \\ \alpha, \beta, \gamma, \delta, \varepsilon \geq 0 \end{vmatrix}$$

On a donc les variables v_1, v_2, v_3,v_4, v_5, v_6, v_7 et v_8 relatives aux vecteurs e_1, e_2, e_3, e_4, e_5, e_6, e_7 et e_8 ainsi que la variable z. On a ajouté les variables d'écart $\alpha, \beta, \gamma et \delta$ ainsi que la variable artificielle ε .

On peut ensuite le simplexe lorsque le problème est ainsi posé.

5) On émet l'hypothèse que :

$$\Theta(y) = \{v / ¿ v \vee \leq V_{max}\}$$

Etudions les cas suivants :

- Si $¿ p \vee \leq 1$, alors cela signifie physiquement que le piéton a effectué un trajet plus court que la ligne droite ($¿ p \vee ¿ 1 ¿$, ce qui est impossible ou alors qu'il a effectué une ligne droite parfaite ($|p| = 1 ¿$, ce qui est également impossible.

  La seule solution possible est donc :

$$\begin{cases} V(y,p) = 0 \\ H(y,p) = 0 \end{cases}$$

- Si $¿ p \vee ¿ 1$, étudions alors à nouveau 2 sous-cas :

  - $p > 1$

    On veut minimiser $|v| + p.v$ (1)

    On veut donc $p.v < 0$ pour cela il faut que le vecteur v soit opposé et colinéaire à p

$$\exists \mu > 0, v = -\mu . p$$

    Et $(|v| + p.v) \in [V_{max}(1-p); V_{max}(1+p)]$

    On retient ensuite le minimum donc $H(y,p) = V_{max}(1-p)$

    Qui est obtenu pour $|v| = V_{max}$ (2)

    Avec (1) et (2), il vient $V(y,p) = -V_{max}\frac{p}{|p|}$

  - $p < -1$

Ici on veut minimiser $|v|+p.v$ (3)

On veut donc $p.v<0$ pour cela il faut que le vecteur v soit opposé et colinéaire à p

$$\exists \mu>0, v=-\mu.p$$

Et $(|v|+p.v)\in[V_{max}(1+p);V_{max}(1-p)]$

On retient ensuite le minimum donc $H(y,p)=V_{max}(1+p)$

Qui est obtenu pour $|v|=V_{max}$ (4)

Avec (3) et (4), il vient $V(y,p)=-V_{max}\frac{p}{|p|}$

On en conclut que :

$$\begin{cases} H(y,p)=V_{max}(1+|p|) \\ V(y,p)=-V_{max}\frac{p}{|p|} \end{cases}$$

6) **Cas** $|p|>1$:

I(x,t) = λ(t).x

$\frac{\partial I}{\partial x}$ = λ(t) et

Si p>1

$\frac{\partial I}{\partial t}(x,t)$ = λ'(t).x = $H\left(x,\frac{\partial x}{\partial t}(x,t)\right)$ = min (¿v∨¿ - $\frac{\partial I}{\partial x}$*¿ v) = Vmax.(1-

¿λ(t)∨¿)

λ'(t) - $\frac{Vmax}{x}$. λ(t) = $\frac{Vmax}{x}$

D'où $\lambda(t) = A.\ e^{\frac{-Vmax}{x}.t} + 1$ avec $\lambda(0)=0=A+1$

Donc $\underline{\lambda(t) = -\ e^{\frac{-Vmax}{x}.t} + 1}$

Si p<-1

$\frac{\partial I}{\partial t}(x,t) = \lambda'(t).x = H\left(x, \frac{\partial x}{\partial t}(x,t)\right) = \min($ ¿v∨¿ - $\frac{\partial I}{\partial x}$*¿ v) = Vmax.(1- ¿λ(t)∨¿)

$\lambda'(t) + \frac{Vmax}{x}.\lambda(t) = \frac{Vmax}{x}$

D'où $\lambda(t) = B.\ e^{\frac{Vmax}{x}.t} - 1$ avec $\lambda(0)=0=B-1$

Donc $\underline{\lambda(t) = e^{\frac{-Vmax}{x}.t} - 1}$

Cas $|p|\leq 1$ **:**

$\frac{\partial I}{\partial t}(x,t) = \lambda'(t).x = H\left(x, \frac{\partial x}{\partial t}(x,t)\right) = 0$

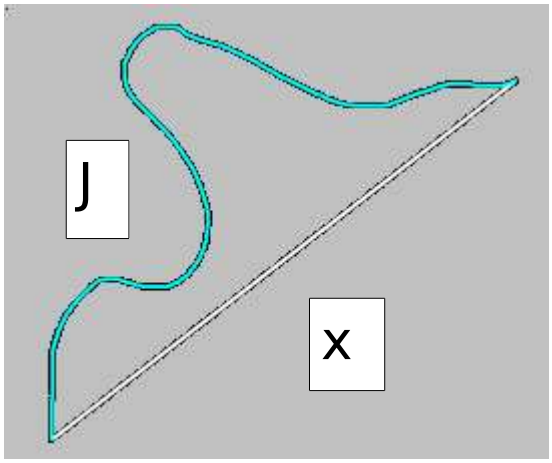
$\lambda'(t) = 0$

D'où $\lambda(t) = A$ avec $\lambda(0)=0=A$

Donc $\underline{\lambda(t) = 0}$

7) On a la condition $\left|\frac{\partial J}{\partial x}\right| > 1$

En effet J représente physiquement la distance parcourue entre 2 points typiquement.
X représente alors la distance la plus courte entre ces 2 mêmes points, c'est-à-dire la ligne droite.



Cette condition signifie donc qu'il est impossible de joindre un point à un autre en effectuant un parcours plus court que la ligne droite d'une part ($\left|\frac{\partial J}{\partial x}\right|<1$ est exclu) et d'autre part qu'un piéton ne marche jamais en effectuant une ligne droite parfaite ($\left|\frac{\partial J}{\partial x}\right|=1$ est exclu).

2. Programmation

Pour que la modélisation soit la plus réaliste possible, nous avons choisi de garder la position des piétons exacte, et de ne pas discrétiser l'espace en une matrice.

L'approximation temporelle apparait lors du déplacement des piétons qui bougent par pas de 1 seconde.
Le programme va alors chercher la vitesse maximale qui respecte les conditions de vitesses admissibles Nu portées sur les vecteurs de la matrice des contraintes N.

Des approximations spatiales seront effectuées lors de la recherche de la propagation optimale.

1) Fonction globale

```
function[A]=dynamique(t)
%t nombre total de secondes de simulation
%on crée 5 piétons aléatoirement dans la pièce
P=initialisation();
%on stocke les positions à tous les instants dans résultat
resultat=P;
for k=1:t
%on génère l'apparition d'un piéton toutes les 5 secondes
if isinteger(k/5)==1
   P=genererpieton(P);
end
resultat=[resultat;P];
%on actualise la position des piétons toutes les secondes
%grâce à l'algorithme du simplexe
P=actualiser(P);
% dans cette fonction,
%1) on vérifie s'il y a des contraintes grâce à obstacles (N)
%2) on propage le piéton vers la sortie grâce à direction
%-s'il n'y a pas de contrainte, il avance dans cette direction
%(size(N)==[0,0])
%-sinon on fait le simplexe(N)

end
%l'animation A est constituée du défilement des positions
%successives des piétons qui sont dans resultat
end
```

```
function[P]=actualiser(P)
% dans cette fonction,
%1) on vérifie s'il y a des contraintes grâce à obstacles (N)
%2) on propage le piéton vers la sortie grâce à direction
%-s'il n'y a pas de contrainte, il avance dans cette direction
%(size(N)==[0,0])
%-sinon on fait le simplexe(N)
[~,n]=size(P);
for k=1:n
   A=direction(P(:,k));
   [N,Nu]=obstacles(P,k);
   if N==[]
      P(:,k)=P(:,k)+Vmax*A/dictance(A);
   else
      V=simplexe(N,Nu,A);
      P(:,k)=P(:,k)+V;
   end
end


function[P]=genererpieton(P)
%on rajoute un piéton en (0,2) à la liste des piétons
P=[P,[0;2]];
end
```

2) Initialisation du problème

```
%place les piétons au départ
function[P]=initialisation()
%P liste des piétons
P=[];
%coordonnées des 2 coins extrêmes du pilier (de gauche à droite et de haut en bas )
x1=2.60;
x2=3;
y1=2;
y2=2.4;

k=0;
while k<5
   [a,b]=alea();
   c=chevauchement(P,[a;b]);
   if not( (a>x1)&&(a<x2)&&(b>y1)&&(b<y2)) && c==1
%on teste le fait que le piéton n'aille pas dans le pilier et qu'il ne soit
%pas trop proche d'un autre
      P=[P,[a;b]];
      k=k+1;
   end
end
end


function[c]=chevauchement(P,A)
% regarde si deux piétons à moins de 20 cm l'un de l'autre
%si c'est le cas on ressort c=0
k=1;

[~,n]=size(P);
c=1;
while k<n+1
   if distance(P(:,k)-A)<0.2
      k=n+1;
      c=0;
   else k=k+1;
   end
end
end


function[d]=distance(C)
%donne la norme du vecteur C

d=sqrt(C(1)*C(1)+C(2)*C(2));
end
```

3) Calcul de la matrice N

```
function[N,Nu]=obstacles(P,n)
%P est la liste des vecteurs [x;y] des coordonnées des piétons
%Le piéton n entre en interaction avec les autres éléments (murs et
%piétons)
N=[] % liste des normales [a;b]
Nu
[~,m]=size(P);
for i=1:m
    if i==n
    else
        [N,Nu]=interaction(N,Nu,P(:,i),P(:,n));
    end
end
[N,Nu]=murs(N,Nu,P(:,n));
[N,Nu]=pilier(N,Nu,P(:,n));
end

function[N,Nu]=interaction(N,Nu,A,B)
D=1.2;
% distance à partir de laquelle le piéton prend les autres en compte
C=A-B;
%si les piétons sont suffisamment proches, on rajoute une normale dans les
%conditions
if (distance(C)<D)
   N=[N,C/distance(C)];
Nu=[Nu,min((distance(C)-0.2)/0.7,2*sqrt(0.1946*(1-exp(-distance(C)/1.2))))]
end
end

function[N,Nu]=murs(N,Nu,P)
D=1.2;
% distance à partir de laquelle le piéton prend les murs en compte
% on a considère les quatre murs extérieurs de la pièce
if (4-P(1))<D
   N=[N,[1;0]];
Nu=[Nu,min(4-P(1)/0.7,2*sqrt(0.1946*(1-exp(-distance(C)/1.2))))];

end
if P(1)<D
   N=[N,[-1;0]];
Nu=[Nu,min(P(1)/0.7,2*sqrt(0.1946*(1-exp(-distance(C)/1.2))))];

end
if (4-P(2))<D
   N=[N,[0;1]];
Nu=[Nu,min(4-P(2)/0.7,2*sqrt(0.1946*(1-exp(-distance(C)/1.2))))];

end
```

```
if P(2)<D
   N=[N,[0;-1]];
Nu=[Nu,min(P(2)/0.7,2*sqrt(0.1946*(1-exp(-distance(C)/1.2))))];

end
end

function[N]=pilier(N,P)
D=1.2;
%coordonnées du pilier
x1=2.6;
x2=3;
y1=2;
y2=2.4;
%regarde si le piéton P est proche du pilier (on a choisi de délimiter
%cette zone par un carré)
%si c'est le cas, on détermine la normale
if ((P(1)<x2+D)&&(P(1)>x1-D)&&(P(2)<y2+D)&&(P(2)>y1-D))
   %on détermine l'équation de la normale A
   if (P(1)<x1 && P(2)<y1)|| (P(1)>x2 && P(2)>y2)||(P(1)<x1 && P(2)>y2)||(P(1)>x2 &&
P(2)<y(1))
      %le piéton arrive vers un coin du pilier
   minx=min(abs(P(1)-x1),abs(P(1)-x2));
   miny=min(abs(P(2)-y1),abs(P(2)-y2));
   A=[minx miny];
   elseif P(1)>x1 && P(1)<x2
   % le piéton arrive par le haut ou le bas du pilier
      miny=min(abs(P(2)-y1),abs(P(2)-y2));
      A=[0 miny];
   else
      %le piéton arrive par la gauche ou la droite du pilier
      minx=min(abs(P(1)-x1),abs(P(1)-x2));
      A=[minx 0];
   end

   N=[N,A/distance(A)];
   Nu=[Nu,min(distance(C)/0.7,2*sqrt(0.1946*(1-exp(-distance(C)/1.2))))];
end
end
```

4) Algorithme du simplexe

```
function[V]=simplexe(N,Nu,A)
T=[Nu;N];
Tab=[0,couts;T'];
%tab est la matrice sur laquelle on effectue le simplexe
%la première colonne est le second membre Nu
%la ligne supérieure représente les couts
%et on a au milieu la matrice des contraintes

%on dégage la variable artificielle
Tab=artificielle(Tab,L);
%tant qu'il y a des couts négatifs, on continue l'algorithme
while not(min(Tab(1,2:end))>0)
Tab=algo(Tab,[]);
end
V=variables(A,Tab);
end

function[Tab,ligne]=algo(Tab,ligne)
%ligne représente la ligne des coûts réels du tableau lorsque l'on utilise
%l'algorithme pour enlever les variables artificielles.
[m,n]=size(Tab);
%trouver le coût négatif le plus petit
[~,y]=minimum(Tab(1,2:n));

%trouver le pivot (on divise le second membre par la colonne entrante)
   P=Tab(:,1)/Tab(:,y);
[~,X]=minimum(P(2:end));
x=X+1;

%opération avec le pivot en Tab(x,y)

Tab(x,:)=Tab(x,:)/Tab(x,y);
ligne=ligne-Tab(x,:);
for k=1:m
   if k==x
   else
   Tab(k,:)=Tab(k,:)-Tab(k,y)*Tab(x,:);
   end
end

function[T]=artificielle(T,L)
%L partie de la matrice des variables artificielles
[~,n]=size(T);
[k,l]=size(L);
somme=zeros(1,n);

for j=1:l
   for i=1:k
```

```
    if L(i,j)==1
        somme=somme-T(i,:);
    end
  end
end
%sauvegarde de la 1ere ligne du tableau, on la remplace par les couts
%de valeur artificielle pour pouvoir utiliser algo dessus
couts=T(1,:);
T(1,:)=somme;
while not(T(1,:)==zeros(1,n))
[T,couts]=algo(T,couts);
end
T(1,:)=couts;
end


function[V]=variables(A,Tab);
%compare les valeurs possibles du vecteur V obtenues grâce à l'algorithme
du simplexe à la direction vers la sortie A et conserve la plus proche pour
effectuer l'actualisation de la trajectoire.
end
```