Mise en œuvre de la réalisation diffusive de l'opérateur \mathcal{Z}

La relation d'impédance s'exprime :

$$\widehat{\varphi|_{\partial\Omega}} = H(\partial_t).\widehat{\partial_\nu \varphi|_{\partial\Omega}}$$

Lors de la mise en œuvre numérique de l'opérateur \mathcal{Z} via la représentation d'état de $H(\partial_t)$, l'entrée $u \operatorname{est} \widehat{\partial_{\nu} \varphi|_{\partial \Omega}}^{y}$ (respectivement $\widehat{\partial_{\nu} \varphi|_{\partial \Omega}}^{\theta}$ dans le cas du cercle), la sortie étant alors $\widehat{\varphi|_{\partial \Omega}}^{y}$ (respectivement $\widehat{\varphi|_{\partial \Omega}}^{\theta}$). Ainsi, on a le schéma:

$$\begin{cases} \psi^{t+\Delta t} = F\psi^t + G\widehat{\partial_\nu \varphi|^t_{\partial\Omega}} , \ \psi^0 = 0\\ \widehat{\varphi|^{t+\Delta t}_{\partial\Omega}} = C\psi^{t+\Delta t}. \end{cases}$$

Il reste maintenant à discrétiser la frontière en N points. A t fixé, $\varphi|_{\partial\Omega}$ (fonction de y) devient alors un vecteur $(\varphi|_{\partial\Omega})_{n=1...N}$. Par transformation de Fourier discrète, $\widehat{\varphi|_{\partial\Omega}}$ (fonction de η) devient donc un vecteur $(\widehat{\varphi|_{\partial\Omega}}^y)_{n=1...N}$ des coefficients de Fourier. Avec n changent les pôles de $H_n(p)$, les coupures, donc γ également (c'est-à-dire les ξ_k), A, C, donc F et G. A chaque pas de temps, il faudra donc effectuer pour tout n la représentation d'état :

$$\begin{cases} \psi_n^{t+\Delta t} = F_n \psi_n^t + G_n (\widehat{\partial_\nu \varphi}|_{\partial\Omega}^t)_n , \ \psi^0 = 0\\ \widehat{(\varphi|_{\partial\Omega}^{t+\Delta t})_n} = C_n \psi_n^{t+\Delta t}. \end{cases}$$

Remarque 15 Le nombre N est en pratique fixé à 20, ce qui permet d'obtenir une erreur d'approximation relative dans tous les cas et à toute fréquence inférieure à 10^{-3} .

Remarque 16 Dans le cas d'une frontière circulaire, le fait de discrétiser la frontière a pour conséquence par Fourier de borner le spectre des fréquences balayées (n représente ici une fréquence car la longueur de la frontière est 2π), ce qui se traduit par le fait que n appartient non plus à \mathbb{Z} mais à $\left[-\frac{N}{2}, \frac{N}{2}\right]$.

 F_n , G_n et C_n sont indépendants du temps, on peut donc les calculer hors ligne et les stocker définitivement. En fait, on calculera au préalable les matrices A et les vecteurs C, qui ne dépendent que de la longueur de la frontière et du nombre de points, les matrices F_n et G_n s'établissant facilement à partir de A. Il faut pour cela faire un choix quant au contour γ_n considéré :

• Dans le cas d'une frontière droite, il est composé de γ_n^+ et γ_n^- respectivement $\mathbb{R}_- + \eta i$ et $\mathbb{R}_- - \eta i$. Du fait de la symétrie $H_\eta(\overline{p}) = \overline{H_\eta(p)}$, on ne considèrera que les calculs sur γ_η^+ .



• Dans le cas d'une frontière circulaire, il est composé de la coupure \mathbb{R}_{-} , et de deux contours γ_n^+ et γ_n^- conjugués entourant les pôles de H(p). En pratique, le contour entre les pôles et l'axe imaginaire est suffisant.



De même que dans le cas précédent, la symétrie $H_n(\overline{p}) = \overline{H_n(p)}$ permet de ne considérer que la coupure \mathbb{R}_- et le contour γ_n^+ .

Schéma récapitulatif :



7 Résultats de simulation

Les simulations ont été effectuées sur un maillage de 128 $(r) \times 1024$ (θ) points dans le disque unité privé d'un disque centré à l'origine, de rayon 0.1 avec conditions de Dirichlet nulles. La condition d'impédance est quand à elle discrétisée au moyen de 20 points en la variable ξ . La transformation de Fourier sur le bord transparent est la FFT, fonction disponible en standard sous Matlab.

Les résultats de simulation confirment dans tous les cas la parfaite absorption de la frontière, tant pour les incidences orthogonales aux bord qu'obliques ou rasantes. Il faut noter que des fluctuations trop rapides engendrent des erreurs numériques qui se propagent et dégradent la solution: il faut en pratique respecter un minimum de 10 points par période pour une onde sinusoïdale.

En Figure 1 est visible l'évolution d'un ébranlement provoqué par une source ponctuelle. La neutralité du bord aux ondes rasantes est clairement mise en évidence. Après passage de l'ébranlement, le milieu reste au repos.

Une onde rampante de la forme $\varphi = \operatorname{Re} \left(k(r) e^{i(\omega t + n\theta)} \right)$, provoquée par une source sur le cercle de rayon 0.375 animée d'un mouvement de rotation uniforme, est montrée en Figure 2. N.B.: la transition entre la zone de champ proche (c'est-à-dire lorsque la vitesse de l'onde est inférieure à la célérité du milieu, la décroissance étant alors exponentielle) et la zone de rayonnement par propagation (en spirale, à décroissance en $\frac{1}{\sqrt{r}}$) est clairement visible.

Enfin, on peut voir en Figure 3 l'évolution provoquée par une source ponctuelle constante. En simulant une évolution en espace libre, la condition d'impédance permet ainsi de résoudre au moyen de l'équation des ondes dans un domaine limité un problème d'équilibre statique du type: $\Delta \Phi = f$ dans \mathbb{R}^2 .

References

- M. Lavrentiev & B. Chabat, "Méthodes de la théorie des fonctions de la variable complexe", MIR 1977.
- [2] M.J. Grote & J.B. Keller, "Nonreflecting boundary conditions for time-dependent scattering", J. of Comput. Physics, 52, 1996
- [3] D. Levadoux & G. Montseny, "Diffusive formulation of the impedance operator on circular boundary for 2D wave equation", Sixth International Conference on Mathematical and Numerical Aspects of Wave Propagation, Jyvaskyla, Finland, June 30 - July 4, 2003
- [4] G. Montseny & J. Audounet, "Représentation diffusive: une introduction", Journées Doctorales d'Automatique JDA'2001, Toulouse, 25-27 septembre 2001



Figure 1: Propagation d'un ébranlement - t = 0.375, 0.75, 1.125, ..., 2.625 et 3 s.



Figure 2: Onde rampante.



Figure 3: Résolution d'un problème d'équilibre statique.

Listing des programmes

ondepol_imp_RE_stock.m

% résolution de l'équation d'ondes 2D pour un domaine circulaire absorbant. % Programme unifié qui permet d'utiliser indifféremment imp pour frontière droite sur un cercle ou imp adaptée pour le cercle en choisissant à l'appel 1 (pour adaptée) autre chose (pour frontière plane). % En commentaires : possibilité de mettre une source haute fréquence, une Gaussienne, une source temporaire Ŷ sur tout le bord interieur, ondes rampantes... % rm et rs sont les rayons min et max du domaine % Nr et Nth sont les nombres de points de discrétisation selon le rayon et l'angle % A0, A1 e A2 sont les matrices Nr*Nth-1 des valeurs de Phi en tn-1, tn et tn+1 % T est le nombre d'itérations en temps, donc à quelque chose près (multiplication par dt) le temps de simulation % X et Y maillage du domaine, M matrice des valeurs de phi. function [X,Y,M]=ondepol_imp_RE_stock(rm,rs,Nr,Nth,T,choix) dim=20; M=zeros(Nr,Nth,20); % vecteur de matrices Nr*Nth qui servira a stocker de temps en temps l'état du système V=zeros(20,1); % vecteur qui servira a stocker les valeurs en un point pour explosion dr=(rs-rm)/(Nr-1);dth=2*pi/(Nth-1); discretr=linspace(rm,rs,Nr); discrett=linspace(0,2*pi,Nth); X=discretr'*cos(discrett); % matrice des abscisses de discrétisation Y=discretr'*sin(discrett); % matrice des ordonnées de discrétisation % Impulsion de départ : Gaussienne %M(:,:,1)=gaussiennepol(X,Y,0,0.8,0.001); % CI pour la simulation de la source ponctuelle M(round(8*Nr/10), 1, 1)=1;A0=M(:,1:Nth-1,1); A1=A0; A2=zeros(Nr,Nth-1); dt=0.25*min(dr,rm*dth) temps_de_simu=dt*T c1=((1-dt^2/dr^2)*ones(1,Nr-2)-dt^2./(discretr(2:Nr-1)*dth).^2)'; c2=(dt^2./(2*discretr(2:Nr-1)*dr))'; c3=(dt^2./(discretr(2:Nr-1)*dth).^2)'; C1=c1*ones(1,Nth-3);C2=c2*ones(1,Nth-3);C3=c3*ones(1,Nth-3);mu=4; % fréquence de la source omega=2*pi*mu; % pulsation de la source psi=zeros(dim,Nth-2); % une colonne de psi est le vecteur à k composante psi_n, n balayant les fréquences. psi0=zeros(dim,1); fftphi=zeros(Nth-1,1); % vecteur sortie de la RE (vecteur Fourrier(phi)) F=zeros(dim,round((Nth-1)/2));

```
G=zeros(dim,round((Nth-1)/2));
k=15; % vitesse angulaire des ondes rampantes
MR=dim;
% évaluation du cas à traiter puis chargement de A,C et calcul de F0, G0 et
C0 en conséquent
if (choix==1) % imp adaptée
   load acpolimp.mat;
   F0=exp(diag(a0)*dt);
   G0=(1./diag(a0)).*(F0-1);
else % imp frontière droite
   load acpol.mat
   F0=eye(MR,1);
   G0=dt*eye(MR,1);
   CO=-eye(1,MR)/2; % pour palier le 2*real(...) de la RE
end
% calcul de F,G,C
8 *********
  * * * * * * * * * * * * * *
Ŷ
C=C(1:round((Nth-1)/2),:); % seuls les N de 1 à (Nth-1)/2 sont considérés,
même si C en contient bcp +
for N=1:round((Nth-1)/2)
  F(:,N) = \exp(A(:,N) * dt);
  G(:,N) = (1./A(:,N).*(F(:,N)-1));
end
% Début de la simulation
8 ***************
  8
for t=1:T % boucle de temps
   % Evolution du système
   % calcul de la première colonne à part (relatif raccord de theta en 0 et
2 pi)
  A1(1:Nr-2,1))) + c3.*(A1(2:Nr-1,2)+A1(2:Nr-1,Nth-1)) + 2*A1(2:Nr-1,1).*c1-
A0(2:Nr-1,1);
   % autres colonnes
  A2(2:Nr-1,2:Nth-2)=(dt^2/dr^2)*(A1(3:Nr,2:Nth-2)+A1(1:Nr-2,2:Nth-2)) +
(C2.*(A1(3:Nr,2:Nth-2)-A1(1:Nr-2,2:Nth-2))) + C3.*(A1(2:Nr-1,3:Nth-
1)+A1(2:Nr-1,1:Nth-3)) + 2*C1.*A1(2:Nr-1,2:Nth-2)-A0(2:Nr-1,2:Nth-2);
   % calcul de la dernière colonne à part(idem, relatif raccord de theta en
0 et 2 pi)
  A2(2:Nr-1,Nth-1)=(dt^2/dr^2)*(A1(3:Nr,Nth-1)+A1(1:Nr-2,Nth-1)) +
(c2.*(A1(3:Nr,Nth-1)-A1(1:Nr-2,Nth-1))) + c3.*(A1(2:Nr-1,1)+A1(2:Nr-1,Nth-
2)) + 2*A1(2:Nr-1,Nth-1).*c1-A0(2:Nr-1,Nth-1);
   % Réalisation de l'impédance adaptée par R.E
   & **********
   deriv_norm=zeros(Nth-1,1);
  deriv_norm=(A1(Nr,:)-A1(Nr-1,:))'/dr;
   fftderiv_norm=fft(deriv_norm); % passage dans le domaine des fréquences
```

```
% cas n=0
   psi0=F0.*psi0+G0*fftderiv_norm(1);
   fftphi(1)=C0*psi0;
   % n=1..round((Nth-1)/2)
   for n=1:round((Nth-1)/2)
      psi(:,n)=F(:,n).*psi(:,n)+G(:,n)*fftderiv_norm(n+1);
      fftphi(n+1)=C(n,:)*psi(:,n);
   end
   % n=1..round((Nth-1)/2)-1
   for n=1:round((Nth-1)/2)-1
      psi(:,Nth-n-1)=F(:,n).*psi(:,Nth-n-1)+G(:,n)*fftderiv_norm(Nth-n);
      fftphi(Nth-n)=C(n,:)*psi(:,Nth-n-1);
   end
   phi=ifft(fftphi);
   A2(Nr,:)=2*real(phi.');
   % Options
   8 ******
   % source
   %A2(round(9*Nr/10),1)=1-cos(omega*t*dt);
   % bord interieur source 1 periode
   %if (t*dt<=1/mu)</pre>
   8
       A2(1,:)=1-\cos(\operatorname{omega*t*dt});
   %end
   % source durant 2 periodes
   %if (t*dt<=2/mu)</pre>
   % A2(round(8*Nr/10),1)=1-cos(omega*t*dt);
   %end
   %A2(round(8*Nr/10),1)=1;
   %A2(round(Nr/3),round(5*Nth/7))=-1;
   % source ponctuelle
   A2(round(8*Nr/10),1)=1;
   %ondes rampantes 3 periodes sur le bord
   %A2(floor(3*Nr/8),1:Nth-1)=( 1-exp(-1000*(t*dt)^2/2) )*sin(
   10*2*pi*(1:Nth-1)/(Nth-1)+k*t*dt );
   %A2(1:floor(3*Nr/8)-1,1:Nth-1)=0;
   % Stockage
   8 *******
   % Sélection de "tranches spatiales" pour afficher l'évolution, tous les
10 pas de temps. Même si A a pour taille 20 à
   % l'initialisation, cette taille augmente au fur et à mesure que l'on
doit stocker.
   if (mod(t, 100) == 0)
      M(:,1:Nth-1,(t/100)+1)=A2;
      M(:,Nth,(t/100)+1)=A2(:,1); % rajout de la dernière colonne (2 pi)
égale à la première (0) pour le tracé
      V((t/100)+1) = A2(round(8*Nr/10),round(Nth/4));
   end
   if (mod(t, 100) == 0)
      t % indicateur de progression
   end
```

A0=A1; A1=A2; end

acpolimp.m

```
% fonction qui renvoie les matrices AO, CO, A et C de la RE pour une
frontière circulaire avec l'impédance adaptée
% L est la longueur du cercle
% Nth le nombre de points de discrétisation de [0,L]
% dt pas de temps
function [A0,C0,A,C]=acpolimp(L,Nth)
dim=20;
A=zeros(dim,round((Nth-1)/2)); % matrice des diagonales des An;
C=zeros(round((Nth-1)/2),dim);
% matrices A0 et C0
8 *******
۶ *********
MR=dim; %dimension de la realisation sur R-
omR=logspace(-1,3,MR); % definition des qsi sur R-
om=omR; % ensemble complet des points de discretisation
% Calcul de la matrice Q a "pseudo-inverser"
QN=1000; % nbre de points de discretisation en frequence de Kn
L=ON;
M=MR; % dimension de la realisation d'etat
w=logspace(-1,3,QN); % discretisation sur iR
OM=w;
OM = [OM - OM];
epsilon=1e-8; % pour penalisation
I=eye(M) ; % matrice identite
% matrice Q
Q = ones(2*L,1)*om+(i*OM'*ones(1,M));
Q=1./Q;
% Pseudo-inversion
&***********
QQ=Q'*Q+epsilon*I;
QI=inv(QQ); % pseudo-inverse de Q
% Calcul du transfert KO
8*********
k=conj((2*besselh(0,w)./(besselh(-1,w)-besselh(1,w)))./w);
H=k.';
H=[H;conj(H)];
mu=QI*Q'*H; % vecteur mu optimal
% Matrices de réalisation d'état
8****************************
A0=diag(-omR); % (par symétrie)
CO=mu.'/2; % pour palier le 2*re(..)
```

```
% matrices A et C
۶ **********
  * * * * * * * * * * * * *
ò
Mq=15; % dimension sur partie courbe de gamma
MR=dim-Mq; % dimension de la realisation sur Ni+R-
qsi=linspace(0,3,Mq);
               % nbre de points de discretisation en frequence de Hn
QN=1000;
L=QN;
M=MR+2*Mq; % dimension de la realisation d'etat
epsilon=1e-8; % pour penalisation de la pseudo inversion
I=eye(M) ; % matrice identite
for N=1:round((Nth-1)/2)
  Ν
   omR=N.*logspace(1,3,MR)./100; % definition des qsi sur R-
   gamma=N*i*exp(i*(1-tanh(qsi)).*pi./2);
   gamma=0.4*real(gamma)+i*imag(gamma);
   om=[omR -qamma -conj(qamma)]; % ensemble complet des points de
discretisation
   w=N.*logspace(1,5,QN)./100; % discretisation sur iR
   OM=w;
   OM = [OM - OM];
   % matrice O
   Q=ones(2*L,1)*om+(i*OM'*ones(1,M));
   0=1./0;
   % Pseudo-inversion
   8*************
   QQ=Q'*Q+epsilon*I;
   QI=inv(QQ); % pseudo-inverse de Q
   % Calcul des transferts Kn
   8*****************
   q0=besselh(1,w)./besselh(0,w);
   if (N==1)
     q=q0;
   else
      if (N==2)
        q=(2./w)-1./q0; % calcul de Q1
      else
         q=(2./w)-1./q0;
         for n=2:N-1
            q=(2*n./w)-1./q; % calcul de QN-1
         end
      end
   end
  k=(1-2./(1-(1./(-1+2*N*q./w))))/N; % calcul de KN
  H=k';
  H=[H;conj(H)];
  mu=QI*Q'*H; % vecteur mu optimal
   % Matrices de réalisation d'état
   A(:,N)=(-om(1:MR+Mq)).'; % diagonale de A (par symétrie)
   C(N,:) = transpose(mu(1:MR+Mq));
   C(N,1:MR)=C(N,1:MR)/2; % pour palier au 2*re(phi), on divise la partie
concernant R- par 2
```

Gaussiennepol.m

```
% renvoie une matrice contenant les valeurs d'une gaussienne dans un
domaine caractérisé par X et Y qui sont
                          les matrices de discrétisation en abscisse et
%
ordonnée du domaine.
% mx et my sont la moyenne de la gausienne et var sa variance
function [M]=gaussiennepol(X,Y,mx,my,var)
nx=size(X,1);
ny=size(X,2);
                 % Matrice des valeurs de la Gaussienne
M=zeros(nx,ny);
for j=1:nx
   for k=1:ny
     M(j,k) = \exp(-((X(j,k)-mx)^2 + (Y(j,k)-my)^2)/(2*var))/(2*pi*var);
   end
end
```

end