CentraleSupélec

Travaux pratiques EDP

Table des matières

1 Introduction

1.1 Sujets proposés

Les deux sujets proposés en travaux pratiques sont les suivants :

- Résolution numérique des EDP pour l'étude des lignes de transmission (voir paragraphe 2)
- Résolution d'un problème en électrostatique (voir paragraphe 3)

1.2 Travail demandé

Ces travaux pratiques seront réalisés au moyen d'un **travail en individuel**. Le compte-rendu décrira les méthodes employées pour la mise en équation des problèmes posés. Les codes Matlab devront être autonomes et bien documentés.

1.3 Livrables

Les livrables seront constitués des éléments suivants :

- Un compte-rendu écrit au format PDF pour les deux sujets
- Une archive des fichiers Matlab pour les deux sujets

La date limite de remise sera fixée sur l'équipe Teams.

2 Résolution numérique des EDP pour l'étude des lignes de transmission

2.1 Introduction

2.1.1 Objectifs

Il s'agit dans cette étude d'établir un schéma numérique permettant de simuler le fonctionnement d'une ligne de transmission, puis de réaliser un programme Matlab afin de tester différentes conditions aux limites.

2.1.2 Livrables

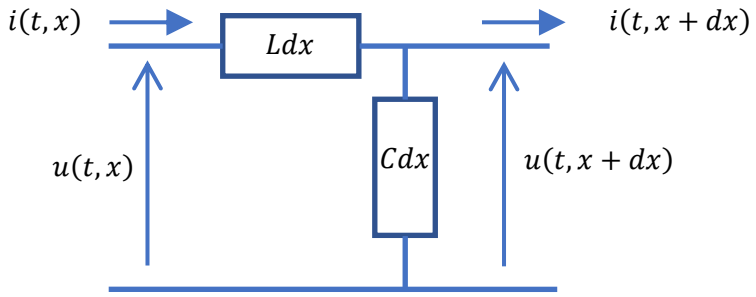
Dans le rapport devront figurer :

- La description du schéma numérique en question
- Les résultats de simulation selon les configurations proposées au paragraphe 2.7.

Dans l'archive devront être ajoutés les scripts Matlab permettant de réaliser les simulations demandées.

2.2 Mise en équation (rappel)

Une ligne de transmission sans pertes est modélisée sur un élément dx de sa longueur selon la figure ci-dessous.



Les grandeurs L et C désignent respectivement une inductance et une capacité par unité de longueur. Les tensions $u(t,x)$ et les courants $i(t,x)$ à l'instant t et à la position x vérifient alors les équations aux dérivées partielles ci-dessous.

$$\frac{\partial u}{\partial x} = -L\frac{\partial i}{\partial t}$$

$$\frac{\partial i}{\partial x} = -C\frac{\partial u}{\partial t}$$

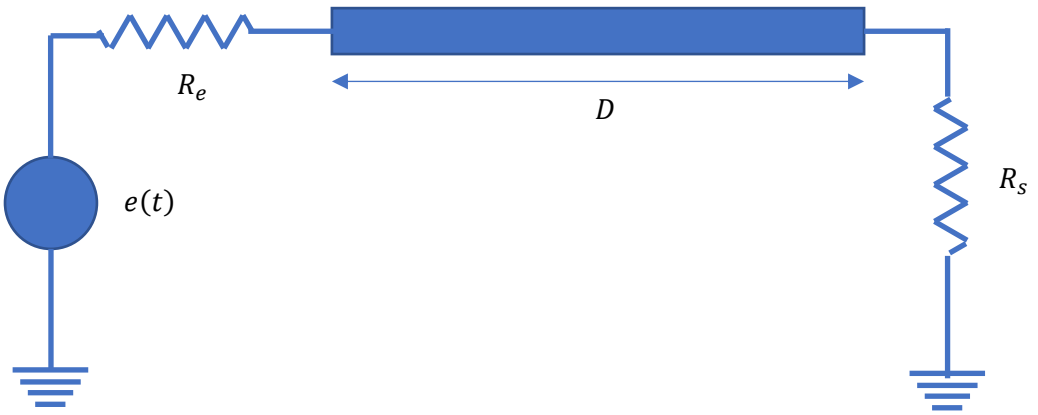
De ces équations initiales résultent les équations propres aux tensions et aux courants présents sur la ligne.

$$\frac{\partial^2 u}{\partial x^2} = LC\frac{\partial^2 u}{\partial t^2}$$

$$\frac{\partial^2 i}{\partial x^2} = LC\frac{\partial^2 i}{\partial t^2}$$

2.3 Conditions aux limites

L'entrée de la ligne est alimentée avec une source de tension $e(t)$ avec une impédance de sortie R_e. L'autre extrémité est connectée à une impédance R_s. La ligne a une longueur totale D.

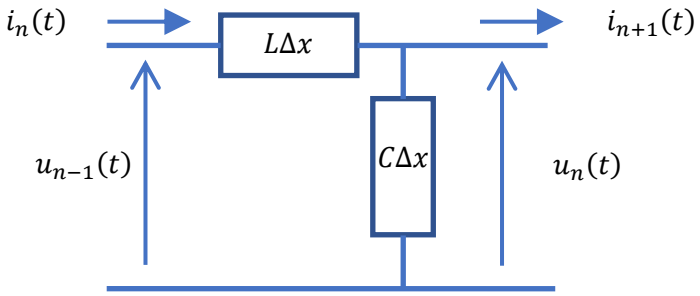


2.4 Discrétisation en espace

Pour simuler le fonctionnement de cette ligne, on se propose de la modéliser en considérant N circuits élémentaires mis en cascade. Chaque circuit approximera le comportement de la ligne sur une longueur $\Delta x = \frac{D}{N}$. Dans chaque circuit n pour $1 \leq n \leq N$, $i_n(t)$ et $u_n(t)$ sont des variables d'état.

Pour le premier circuit ($n = 1$), la tension $u_0(t)$ permet d'énoncer une première condition aux limites prenant en compte la source de tension $e(t)$ ainsi que l'impédance de sortie R_e.

Pour le dernier circuit ($n = N$), le courant $i_{N+1}(t)$ permet d'énoncer une seconde condition aux limites prenant en compte l'impédance R_s et la tension $u_N(t)$.



2.5 Equation d'état

On note $X(t) = \begin{pmatrix} U(t) \\ I(t) \end{pmatrix}$ le vecteur d'état du système avec $U(t)^T = (u_1(t) \quad u_2(t) \quad \cdots \quad u_N(t))$ et $I(t)^T = (i_1(t) \quad i_2(t) \quad \cdots \quad i_N(t))$.

Exprimer l'équation dynamique de ce système, avec les conditions aux limites précédemment décrites faisant notamment intervenir les impédances R_e et R_s ainsi que la source de tension $e(t)$.

$$\frac{dX(t)}{dt} = f\big(X(t), e(t)\big)$$

2.6 Discrétisation en temps

L'équation dynamique précédemment établie peut se résoudre numériquement par les méthodes classiques associées aux EDO, avec un pas en temps de Δt.

Remarquons que, pour l'équation initiale sous la forme $\frac{\partial^2 u}{\partial t^2} = \gamma \frac{\partial^2 u}{\partial x^2}$ avec $\gamma = \frac{1}{LC}$

- La limite de stabilité du schéma numérique est liée à la valeur de $\frac{\Delta t}{\Delta x}\sqrt{\gamma}$.
- La vitesse de propagation vaut $V = \sqrt{\gamma}$.

2.7 Configurations pour les tests à mener

2.7.1 Câble Ethernet RJ45

Pour illustrer la méthode de simulation, les paramètres associés aux câbles RJ45 (réseaux informatiques) peuvent être utilisés. La vitesse de propagation est $V = \frac{1}{\sqrt{LC}} = 2.3\ 10^8\ \mathrm{m/s}$ et l'impédance caractéristique est $Z_c = \sqrt{\frac{L}{C}} = 100\ \mathrm{Ohms}$, soit $L = \frac{Z_c}{V}$ et $C = \frac{1}{VZ_c}$.

On choisira une longueur $D = 500\ \mathrm{m}$, une largeur d'impulsion $T_e = 200\ \mathrm{ns}$ et un pas de temps Δt choisi de sorte que $\frac{\mathrm{V}\Delta t}{\Delta x} < 1$.

2.7.2 Méthodes d'intégration numérique

Les méthodes de Runge Kutta d'ordre 2 et 4 sont fournies sous la forme de deux programmes **myode23.m** et **myode45.m** . Pour chaque méthode, on pourra rechercher la valeur maximale du pas temporel Δt au-delà duquel le schéma numérique devient instable.

2.7.3 Exemple de source de tension

La source de tension $e(t)$ peut par exemple correspondre à une impulsion modélisée sous la forme d'une fonction de support $[0 \quad T_e\ ]$, avec $e(t) = \frac{1-\cos\left(2\pi t/T_e\right)}{2}$ pour $0 \le t \le T_e$.

2.7.4 Exemples pour les impédances

On envisagera les configurations suivantes :

- $R_e = 0, R_s = \infty$
- $R_e = \sqrt{\frac{L}{C}}, R_s = \infty$
- $R_e = \sqrt{\frac{L}{C}}, R_s = \sqrt{\frac{L}{C}}$

2.7.5 Durée

La durée de simulation T est choisir de sorte que l'onde fasse deux allers-retours sur la longueur D de la ligne étudiée, soit $T = 4\frac{D}{V}$ et $V = \frac{1}{\sqrt{LC}}$.

3 Résolution d'un problème en électrostatique

3.1 Introduction

3.1.1 Objectifs

On se propose de tester différents paramétrages pour la résolution numérique des EDP, sur un problème dont on connaît par ailleurs la solution analytique. Il sera alors possible d'évaluer l'erreur sur la solution obtenue, et de rechercher ainsi les moyens pour la réduire.

Les résolutions numériques s'effectueront à l'aide de la boîte à outil « *Partial Differential Equation Toolbox* » de Matlab.

3.1.2 Livrables

Dans le rapport devront figurer les calculs effectués pour la résolution analytique du problème et les résultats des résolutions numériques, associés aux commentaires dont ils pourront faire l'objet.

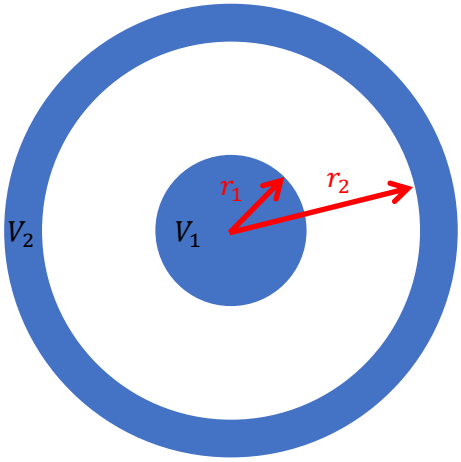
3.1.3 Conseil d'utilisation dans Matlab

Il est vivement conseillé d'utiliser un fichier script – un simple fichier texte avec une extension **.m** – rassemblant les différentes commandes décrites dans ce document. Ce fichier est aussi considéré comme une commande, et peut donc être exécuté comme telle.

3.2 Description et mise en équation du dispositif

3.2.1 Problème à résoudre

La section d'un dispositif de longueur infinie, constitué de deux éléments conducteurs à symétrie circulaire, est représenté ci-dessous.



On considère que l'espace séparant ces deux conducteurs est le vide. Le conducteur intérieur est au potentiel V_1 et celui à l'extérieur au potentiel V_2. Le potentiel V dans cet espace obéit donc à l'équation de Laplace $\Delta V = 0$ avec les conditions aux limites $V = V_k$ sur le contour du conducteur de rayon r_k.

3.2.2 Mise en équation et résolution analytique

Pour le potentiel $V(r,\theta)$ exprimé avec les coordonnées polaires, l'équation de Laplace devient :

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial V(r,\theta)}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 V(r,\theta)}{\partial \theta^2}=0$$

Compte-tenu de la symétrie circulaire du dispositif décrit, montrez que le potentiel s'exprime sous la forme :

$$V(r,\theta) = V(r) = V_0 \log\frac{r}{r_0}$$

Déterminez les valeurs de V_0 et de r_0 en fonction de r_1, r_2, V_1 et V_2.

3.3 Résolution numérique

3.3.1 Paramétrages pour la résolution numérique

3.3.1.1 La boîte à outil « pde »

Dans Matlab, la rubrique « Add-Ons » permet d'ajouter la boîte à outils « Partial Differential Equation Toolbox » à l'environnement de calcul. Il est alors possible de mettre en œuvre la résolution numérique de l'équation aux dérivées partielles associé au problème posé.

3.3.1.2 Description géométrique

L'aide de la fonction **geometryFromEdges** (tapez **doc geometryFromEdge** sur la ligne de commande pour y accéder) donne des indications pour créer un modèle géométrique. On peut s'en inspirer pour créer le modèle géométrique associé au problème posé.

Les opérations s'enchaînent selon le script ci-dessous, avec le tracé pour vérifier la validité des opérations effectuées.

```
model = createpde;
% les valeurs des rayons r1 et r2 peuvent être exprimées en unité réduite
r1=0.2;r2=1;
% créer les variables geom, sf et nf pour le problème posé
g = decsg(geom,sf,ns);
geometryFromEdges(model,g);
% tracé de la description géométrique obtenue
figure,pdegplot(model,'EdgeLabels','on'),drawnow
```

3.3.1.3 Spécification de l'équation à résoudre

L'aide de la fonction **specifyCoefficients** (tapez **doc specifyCoefficients** sur la ligne de commande pour y accéder) indique le moyen de spécifier l'équation aux dérivées partielles à résoudre.

```
% m,d,c,a et f sont les valeurs à déterminer,
% qui correspondent à l'équation de Laplace
specifyCoefficients(model,'m',m,'d',d,'c',c,'a',a,'f',f)
```

3.3.1.4 Définition des conditions aux limites

L'aide de la fonction **applyBoundaryCondition** (tapez **doc applyBoundaryCondition** sur la ligne de commande pour y accéder) décrit le moyen de fixer les conditions aux limites sur les différents segments désignés dans la description géométrique.

```
% les valeurs des potentiel V1 et V2 sont exprimés en volts
V1=10;V2=20;
% appel à la fonction applyBoundaryCondition, avec les potentiels imposés
```

3.3.1.5 Spécification du maillage utilisé

La fonction **generateMesh** réalise un maillage du domaine précédemment décrit. Les différents essais pourront porter sur le choix de différents paramètres de cette fonction.

3.3.2 Lancement de la résolution numérique et visualisation

3.3.2.1 Résolution numérique

Dans le script, la fonction **solvepde** réalise la résolution numérique, après que tous les paramètres du modèle soient correctement initialisés.

```
results = solvepde(model);
```

3.3.2.2 Visualisation des résultats

Les valeurs de la solution numérique sont collectées dans un tableau **u_n**, puis visualisées selon le script ci-dessous.

```
u_n = results.NodalSolution;
figure,pdeplot(model,'XYData',u_n,'ZData',u_n)
colormap(jet)
drawnow
```

Pour comparer avec la solution analytique stockée dans un tableau **u_a**, il faut récupérer aussi les coordonnées de chaque nœud du maillage.

```
x=results.Mesh.Nodes(1,:)';
y=results.Mesh.Nodes(2,:)';
r=sqrt(x.^2+y.^2);
% insérez ici le calcul préalable de V0 et r0 (voir 3.2.2)
u_a=V0*(log(r)-log(r0));
```

La comparaison entre la solution numérique et la solution analytique peut alors s'envisager, en visualisant par exemple l'erreur en valeur absolue.

```
erreur=abs(u_n-u_a);
figure,pdeplot(model,'XYData',erreur,'ZData',erreur)
colormap(jet)
drawnow
```

3.4 Tests à effectuer

3.4.1 Effets des paramètres pour le maillage

Pour la valeur de **Hmax** proposée, testez les 3 configurations ci-dessous et représentez l'erreur pour chacune d'elles.

```
Hmax=0.1;
generateMesh(model,'Hmax',Hmax,'GeometricOrder','linear');
generateMesh(model,'Hmax',Hmax,'GeometricOrder','quadratic');
% le tableau [1 2 3 4] contient les numéros de segment représentant
% le contour du disque de rayon r1
generateMesh(model,'Hmax',Hmax,'Hedge',{[1 2 3 4],Hmax/5});
```

Comparez les résultats et déduisez-en le choix le plus pertinent.

3.4.2 Evaluation de l'ordre de la méthode

Le code ci-dessous permet d'étudier l'effet de ce paramètre **Hmax** sur l'erreur commise par rapport à la solution littérale et le nombre de nœuds constituant le maillage.

Ce code doit être précédé des étapes décrites aux paragraphes 3.3.1.2, 3.3.1.3 et 3.3.1.4, qui sont communes aux opérations réalisées ensuite.

```
Hmax=logspace(-0.5,-1.5,100);
tab_erreur=zeros(length(Hmax),1);
tab_numel=zeros(length(Hmax),1);
for n=1:length(Hmax)
    generateMesh(model,'Hmax',Hmax(n),'Hedge',{[1 2 3 4],Hmax(n)/5});
    results = solvepde(model);
    u_n = results.NodalSolution;
    disp(length(u_n))
    tab_numel(n)=length(u_n);
    x=results.Mesh.Nodes(1,:)';
    y=results.Mesh.Nodes(2,:)';
    r=sqrt(x.^2+y.^2);
    % insérez ici le calcul préalable de V0 et r0 (voir 3.2.2)
    u_a=V0*(log(r)-log(r0));
    erreur=abs(u_n-u_a);
    tab_erreur(n)=max(erreur);
end
figure,semilogx(Hmax,20*log10(tab_erreur),Hmax,40*log10(Hmax)-20)
grid on
figure,semilogx(Hmax,20*log10(tab_numel),Hmax,-40*log10(Hmax)+26)
grid on
```

Testez et commentez.