

Déconvolution de trains d'impulsions. Application au contrôle non destructif par ultrasons

Yassine Jamoud, Samy Haffoudhi

29 décembre 2021

Introduction

Dans ce TP, l'objectif est d'analyser un objet à l'aide de signaux ultrasons. La séquence de réflectivité x sera un signal parcimonieux où chaque pic correspond à un changement d'impédances. L'objectif est de détecter ces pics efficacement. Pour se faire, on considère que notre signal reçu y , est la convolution du signal recherché x avec la réponse impulsionnelle h du transducteur avec une incertitude comportant entre autre le bruit de mesure et les erreurs de modélisation. On adopte donc le modèle :

$$y = x * h + \text{incertitudes}$$

1 Modèle Direct

Simulons le problème en utilisant un exemple de réponse impulsionnelle. On adopte ici le modèle matriciel

$$z = Hx + b$$

Pour simuler le problème efficacement on utilise la convolution de matlab, qui nous évite d'utiliser de grosse matrice dans l'algorithme. On peut alors générer un signal z avec un rapport signal sur bruit donné.

(a) Simulation d'un problème direct avec $RSB_{dB} = 10dB$

(b) Simulation d'un problème direct avec $RSB_{dB} = 40dB$

2 Déconvolution par pénalisation l_1

1. Commençons par utiliser naïvement la solution des moindres carrés $x_{MC} = (H^T H)^{-1} H^T z$.

FIGURE 2 – Simulation avec un rapport signal sur bruit infini

Pour un rapport signal sur bruit infini, le bruit est quasiment inexistant. La solution des moindres carrés fonctionne donc bien :

FIGURE 3 – Solution des moindres carrés avec un rapport signal sur bruit infini

On retrouve effectivement un signal z identique au signal x .

2. Utilisons maintenant un RSB_{dB} très bon.

FIGURE 4 – Simulation avec un rapport signal sur bruit de $40dB$

Cette fois-ci la solution des moindres carrés est catastrophique :

FIGURE 5 – Solution des moindres carrés avec un rapport signal sur bruit de $40dB$

En effet, $z = Hx + b$ et $x_{MC} = (H^T H)^{-1} H^T z$.

Si on injecte z on retrouve :

$$x_{MC} = (H^T H)^{-1} H^T (Hx + b) = x + (H^T H)^{-1} H^T b$$

Le bruit est donc multiplier par un facteur très grand devant lui. En effet, H étant mal conditionnée, les valeurs propres de $H^T H$ sont très petite, donc l'inverse fournit de très grande valeurs propres.

3. On pénalise donc la solution des moindres carrés par norme l_1 :

$$\hat{x} \text{ minimise } J(x) = \frac{1}{2} \|z - Hx\|^2 + \mu \sum_{m=1}^M |x_m|$$

On utilise l'algorithme *ISTA* car la non-differentiabilité en 0 ne permet pas d'avoir de solution analytique.

4. C est la plus grande valeur propre de $H^T H$. On utilise donc $\max(\text{eig}(H^T H))$ pour obtenir une valeur.
5. Évaluons les performances de cette algorithme. Empiriquement, on choisit $\mu = 0.01$. On retrouve alors les resultats suivant :

FIGURE 6 – Solution de l'algorithme ISTA avec un rapport signal sur bruit de $40dB$

FIGURE 7 – Solution de l'algorithme ISTA avec un rapport signal sur bruit de $20dB$

FIGURE 8 – Solution de l'algorithme ISTA avec un rapport signal sur bruit de $10dB$

Pour une presence de bruit presque nul ou très bonne, on arrive à obtenir un resultat très correct. Seul les pics vraiment proche posent un problèmes car ils sont associés comme un seul pics. Pour un rapport singal sur bruit moyen, on commence à voir apparaître des fausses valeurs due au bruit.

3 Mesures d'épaisseurs de plaques par ultrasons

Mettons en oeuvre nos solution sur des signaux réels

1. Pour le signal z_1 , les pics sont assez éloignés pour mesurer l'épaisseur de la plaque sans déconvoluer le signal. L'avantage de ce genre de signal est de pouvoir extraire h , la réponse impulsionnelle très facilement en récupérant le bon nombre de points.

FIGURE 9 – Déconvolution du signal z_1 avec $\mu = 1$

On retrouve un intervalle temporelle entre deux pics d'environ 1206 points. La fréquence d'échantillonnage étant de $100MHz$, cela correspond à un intervalle de $\delta t = 1.2 * 10^{-5}s$. La vitesse du son dans l'aluminium étant de $v = 6380m.s^{-1}$, la distance parcourue est donc $d = \delta t * v = 0.077m$. Ainsi l'épaisseur de la plaque est de $3.85cm$.

2. Pour une plaque plus fine, on retrouve la deconvolution suivante :

FIGURE 10 – Déconvolution du signal z_1 avec $\mu = 1$

Encore une fois, on peut identifier identiquement l'intervalle entre deux pics. On retrouve environ 123 points soit $1.23 * 10^{-6}s$. L'épaisseur de la plaque est donc de $0.4cm$.

Conclusion

Ce TP à ainsi permis de mettre en évidence l'exemple d'utilisation des algorithmes de deconvolution. On a pu mettre en lumière l'inefficacité des moindres carrés en présence de bruit. La régularisation est donc nécessaire. Pour la norme l_1 la non-differentiabilité nous force à utiliser des algorithmes pour approximer au mieux le min de J . Malgré sa sensibilité aux bruits importants, cet algorithme c'est révéler efficace dans la détection des piques par ultrasons. La limite pourra être atteinte pour des matériaux très fin et donc des pics trop proche.

A Simulation d'un problème direct

```
1 function z = probleme_direct(x, h, RSB)
2     N = length(x);
3     y = conv(x, h, "full");
4     E_signal = sum(y.^2);
5     var_bruit = E_signal*10^(-RSB/10)/N;
6     z = y + sqrt(var_bruit)*randn(size(y));
7 end
```

B Seuillage doux

```
1 function y = seuillage_doux(z, mu)
2     y = zeros(length(z),1);
3     for i = 1:length(z)
4         if z(i) >= mu
5             y(i) = z(i) - mu;
6         elseif z(i) <= -mu
7             y(i) = z(i) + mu;
8         end
9     end
10 end
```

C Algorithme ISTA

```
1 function x = ista(z, h, C, mu, max_iter, x0)
2     for q = 1:max_iter
3         x0 = seuillage_doux(x0 + 1/C*conv(z - conv(x0, h, 'full'), flip(h), 'valid'), mu/C);
4     end
5     x = x0;
6 end
```

D Code utilisation des méthodes

```
1 close all
2 clear all
3
4 load('dataCND.mat')
5
6 z = probleme_direct(x, h, inf);
7
8 figure(1)
9 plot(x);
10 hold on;
11 plot(z);
12 xlabel('n')
13 legend('x', 'z')
14 xlim([0 length(x)+1])
15
16 % solution moindres carres
17
18 z = probleme_direct(x, h, inf);
19
20 x_mc = linsolve(H'*H, H'*z);
21 figure(2)
22 plot(x_mc)
```

```

23 hold on
24 plot(x)
25 xlabel('n')
26 legend('x\_mc', 'x')
27 xlim([0 length(x)+1])
28 title('RSB infini')
29
30 z = probleme_direct(x, h, 40);
31
32 x_mc = linsolve(H'*H,H'*z);
33 figure(3)
34 plot(x_mc)
35 hold on
36 plot(x)
37 xlabel('n')
38 legend('x\_mc', 'x')
39 xlim([0 length(x)+1])
40 title('RSB infini')
41
42 % ista
43
44 max_iter = 100;
45 C = max(eig(H'*H));
46 mu = 0.01;
47 z = probleme_direct(x, h, 10);
48 x_ista = ista(z, h, C, mu, max_iter, zeros(length(z)-length(h)+1,1));
49 figure(4)
50 plot(x)
51 hold on
52 plot(x_ista)
53 legend('x', 'x_ista')
54
55
56 load('dataplaques.mat')
57 figure(5)
58 plot(z1)
59
60 h = z1(1:250);
61 C = max(abs(fft(h,8*2^nextpow2(length(h))))).^2);
62 mu = 1;
63 max_iter = 1000;
64 x_ista1 = ista(z1, h, C, mu, max_iter, zeros(length(z1)-length(h)+1,1));
65
66 figure(6)
67 plot(z1)
68 hold on
69 plot(x_ista1)
70 legend('z1', 'z1 ista')
71 xlabel('n')
72 ylabel('z1(n)')
73
74 mu = 1;
75 max_iter = 1000;
76 %x_ista2 = ista(z2, h, C, mu, max_iter, zeros(length(z2)-length(h)+1,1));
77
78 figure(7)
79 plot(z2)
80 hold on
81 %plot(x_ista2)

```

```
82 legend('z2', 'z2 ista')
83 xlabel('n')
84 ylabel('z2(n)')
```