UNIVERSITÉ DE PARIS-SUD
U.F.R. SCIENTIFIQUE D'ORSAY

Fichiers Matlab pour l'instrumentation associée au CharlyRobot et à l'impédancemètre HP4192

Cyril RAVAT

Sujet de thèse :
Conception de multicapteurs à courants de Foucault
et inversion des signaux associés
pour le contrôle non destructif

2005-2008

Table des matières

Script n°1

Diagramme de Bode

```
function bode_HP4192(fichier,mesure,ampl,freq_debut,freq_fin,nb_points,echelle,
    attente,titre)
% bode_HP4192
% bode_HP4192(fichier)
% bode_HP4192(fichier,mesure)
% bode_HP4192(fichier,mesure,amplitude)
% bode_HP4192(fichier,mesure,amplitude,freq_debut,freq_fin,nb_points)
% bode_HP4192(fichier,mesure,amplitude,freq_debut,freq_fin,nb_points,echelle)
% bode_HP4192(fichier,mesure,amplitude,freq_debut,freq_fin,nb_points,echelle,attente)
% bode_HP4192(fichier,mesure,amplitude,freq_debut,freq_fin,nb_points,echelle,attente,
    titre)
%
% Diagramme de Bode d'un dipole branché sur l'impédancemètre HP4192.
% Mesure d'impédance 4 voies ou 2 voies. Les valeurs de gain sont en échelle
% linéaire (et non en dB).
%
% fichier : chaîne désignant le chemin vers un fichier où seront stockés
%           les mesures effectuées. ampl est l'amplitude de l'excitation
%           (scalaire), freq est le vecteur contenant les valeurs de fréquences,
%           Module, Phase, Reel et Imag sont les valeurs mesurées, L est
%           l'inductance.
%           Pour ne pas sauvegarder, mettre une chaîne vide.
%
% mesure : '4fils' -> mesure 4 fils, mesure de R et X (en Omhs)
%                     (valeur par défaut)
%                => Les masses des 4 coax sont reliées.
%                => Les H sont reliés entre eux, les L sont reliés entre eux.
%                => Le composant est mis entre les H et les L pour une mesure
%                d'impédance.
%          '2fils' -> mesure 2 fils (voies A et B), mesure du gain (en
%                     linéaire) et de la phase (en degrés)
%
% amplitude : l'amplitude de l'excitation en V (entre 5mV et 1.1V)
%             (valeur par défaut : valeur courante de l'impédancemètre)
%
% [freq_debut,freq_fin] : plage de fréquence, en Hz
%                         (valeurs par défaut : [5,10e6])
%
% nb_points : nombre de fréquences. Le choix des valeurs de fréquences est
%             logarithmique (valeur par défaut : 100)
%
% echelle : 'log' -> axe des fréquences en échelle logarithmique
%                    (valeur par défaut)
%           'linear'  -> axe des fréquences en échelle linéaire
%
```

```
% attente : 'oui' -> attente de la stabilité du courant passant dans le
%                     composant. Cela est utile si on remarque manuellement,
%                     lorsque l'on modifie la fréquence, que la valeur de
%                     l'impédance mesurée n'est pas immédiatement la valeur
%                     finale (c'est-à-dire correcte).
%           'non' -> pas d'attente, les mesures sont prises instantannément.
%                     (valeur par défaut)
%
% titre : titre de la figure qui sera affichée
%         (valeur par défaut : 'Analyse en fréquence')
%
% Réglages par défaut :
%   mesure        =  '4fils'
%   freq_debut    =   5 Hz
%   freq_fin      =  10 MHz
%   nb_points     =  100
%   echelle       =  'log'
%   attente       =  'non'
%   titre         =  'Analyse en fréquence'
%
% Les réglages par défaut sont modifiables au début du fichier si besoin.
%
% Cyril Ravat, octobre 2005 - avril 2008

% Vérification des paramètres
if ~exist('fichier','var') || ~ischar(fichier)
    fichier = '';
end
if ~exist('mesure','var') || isempty(findstr('2fils4fils',mesure))
    mesure = '4fils';
end
if ~exist('freq_debut','var') || ~exist('freq_fin','var') || ~exist('nb_points','var'
    ) || ~isnumeric(freq_debut) || ~isnumeric(freq_fin) || ~isnumeric(nb_points)
    freq_debut=5;
    freq_fin=10e6;
    nb_points=100;
end
if ~exist('echelle','var') || isempty(findstr('loglinear',echelle))
    echelle='log';
end
if ~exist('attente','var') || isempty(findstr('ouinon',attente))
    attente='non';
end
if ~exist('titre','var') || ~ischar(titre)
    titre = 'Analyse en fréquence';
end

% Affichage
disp('Mesure des diagrammes de Bode par l''impédancemètre HP4192')

% Initialisation du HP4192
global HP4192_ID
% Au SATIE, HP4192_ID = gpib('ni',0,16);
% Au LGEP, HP4192_ID = gpib('ni',0,17);
HP4192_ID = gpib('ni',0,16);
fopen(HP4192_ID)
% Envoi de la première fréquence et d'autres paramètres
% T3 -> trigger en position manuelle
% F1 -> récupérer A/B/C
fprintf(HP4192_ID,['FR' num2str(freq_debut/1000) 'EN;T3;F1'])
if strcmp(mesure,'2fils')
    % A5B2 -> obtenir B-A en dB et la phase en degrés
    fprintf(HP4192_ID,'A5B2')
```

```
else
    % A2 -> obtenir R et X en Omhs
    fprintf(HP4192_ID,'A2')
end
if ~exist('ampl','var') || ~isnumeric(ampl)
    % Lecture de l'amplitude
    fprintf(HP4192_ID,'OLR;EX')
    reponse = fscanf(HP4192_ID);
    ampl = str2double(reponse(34:43));
else
    % Envoi de l'amplitude à l'analyseur
    fprintf(HP4192_ID,['OL' num2str(ampl,2) 'EN'])
end

% Initialisation des tableaux
A = zeros(1,nb_points);
B = A;
C = A;
freq=logspace(log10(freq_debut)-3,log10(freq_fin)-3,nb_points);

% Affichage : récapitulatif
if freq_debut < 1e3
    echo = [ num2str(freq_debut) ' Hz'];
elseif freq_debut < 1e6
    echo = [ num2str(freq_debut/1e3) ' kHz'];
else
    echo = [ num2str(freq_debut/1e6) ' MHz'];
end
if freq_fin < 1e3
    echo = [ echo ' à ' num2str(freq_fin) ' Hz'];
elseif freq_fin < 1e6
    echo = [ echo ' à ' num2str(freq_fin/1e3) ' kHz'];
else
    echo = [ echo ' à ' num2str(freq_fin/1e6) ' MHz'];
end
disp(['Les fréquences sont comprises dans une plage de ' echo '.'])
if ampl < 1
    echo = [ num2str(ampl*1000) ' mV'];
else
    echo = [ num2str(ampl) ' V'];
end
disp(['L''amplitude vaut ' echo '.'])

% Avancement : initialisation
% Paramètre arbitraire modifiable, nombre de caractères de la barre d'avancement
n_av = 25;
if nb_points > n_av
    pas_av = find(diff(floor((1:nb_points)*n_av/nb_points)));
else
    n_av = nb_points;
    pas_av = 1:nb_points;
end
% Chaîne contenant des caractères d'effacement
effacemax = blanks(n_av+12);
effacemax = strrep(effacemax,' ','\b');
% "Avancement : |--------------------          | XX% XXXs"
echo = '-----';
disp(['  Avancement : |' blanks(n_av) '| 0% 0s'])
debut = clock;
echo = blanks(n_av+9);

% Mesures
if strcmp(attente,'non')
```

```
169     for compteur = 1:nb_points
            % Changement de fréquence et déclenchement de la mesure
171         fprintf(HP4192_ID,['FR' num2str(freq(compteur)) 'EN;EX']);
            % Lecture du résultat
173         reponse = fscanf(HP4192_ID);
            % Récupération et stockage des valeurs
175         if reponse(1) == 'N'
                A(compteur) = str2double(reponse(5:15));
177         else
                A(compteur) = NaN;
179         end
            if reponse(17) == 'N'
181             B(compteur) = str2double(reponse(21:31));
            else
183             B(compteur) = NaN;
            end
185         % Avancement
            if ismember(compteur,pas_av)
187             efface = effacemax(1:2*(length(echo)-1));
                echo = [ '-' blanks(n_av-ceil(n_av*compteur/nb_points)) '| ' num2str(ceil
                    (100*compteur/nb_points)) '%% ' num2str(round(etime(clock,debut))) 's
                    '];
189             disp(sprintf([ efface echo ]))
            end
191     end

193 else
        for compteur = 1:nb_points
195         % Changement de fréquence et déclenchement de la mesure
            % On mesure aussi le courant qui passe dans le composant
197         fprintf(HP4192_ID,['FR' num2str(freq(compteur)) 'EN;TA;EX;']);
            % Lecture du résultat
199         reponse = fscanf(HP4192_ID);
            % Initialisation à une valeur impossible pour faire toujours au
201         % moins deux lectures
            courant = -1;
203         % Compteur du nombre de fois où on lit le courant : si on dépasse, c'est
            % qu'il ne se stabilise pas -> "timeout" mais on continue
205         % Tant que le courant n'est pas constant, relecture...
            n = 1;
207         while ( abs(courant - str2double(reponse(34:43))) > 1e-3 ) && ( n < 50 )
                n = n+1;
209             courant = str2double(reponse(34:43));
                fprintf(HP4192_ID,'EX;');
211             reponse = fscanf(HP4192_ID);
            end
213     % Affichage si erreur
            if n == 50
215             disp(['Mesure instable à ' num2str(freq(compteur)) 'kHz']);
                % Réinitialiser l'avancement
217             disp('  Avancement : |')
                for n = 1:floor(n_av*compteur/nb_points)
219                 disp(sprintf('\b-'))
                end
221             echo = [ '-' blanks(n_av-floor(n_av*compteur/nb_points)) '| ' num2str(
                    ceil(100*compteur/nb_points)) '%% ' num2str(round(etime(clock,debut))
                    ) 's'];
                disp(sprintf([ '\b' echo(2:end) ]))
223         end

225         % Récupération et stockage des valeurs
            if reponse(1) == 'N'
227             A(compteur) = str2double(reponse(5:15));
```

```
            else
229             A(compteur) = NaN;
            end
231         if reponse(17) == 'N'
                B(compteur) = str2double(reponse(21:31));
233         else
                B(compteur) = NaN;
235         end
        C(compteur) = courant;
237
            % Avancement
239         if ismember(compteur,pas_av)
                efface = effacemax(1:2*(length(echo)-1));
241             echo = [ '-' blanks(n_av-ceil(n_av*compteur/nb_points)) '| ' num2str(ceil
                    (100*compteur/nb_points)) '%% ' num2str(round(etime(clock,debut))) 's
                    '];
                disp(sprintf([ efface echo ]))
243         end
        end
245     % Remise des fréquences en affichage
        fprintf(HP4192_ID,'FRR;')
247 end

249 % Fermeture de la connexion au HP4192
    fprintf(HP4192_ID,'T1')
251 fclose(HP4192_ID)
    delete(HP4192_ID)
253 clear global HP4192_ID

255 % Avancement - fin
    disp(sprintf([ effacemax(1:2*(length(echo)-3)) '100%% ' num2str(round(etime(clock,
        debut))) 's' ]))
257
    % Récupération et formatage des résultats
259 if strcmp(mesure,'2fils')
        Module = 10.^(A/20); % Le module est enregistré en linéaire
261     Phase = B;
        Reel = Module.*cos(Phase*pi/180);
263     Imag = Module.*sin(Phase*pi/180);
    else
265     Reel = A;
        Imag = B;
267     Module = sqrt(Reel.*Reel+Imag.*Imag);
        Phase = atan2(Imag,Reel)*180/pi;
269 end
    % On remet freq en Hz
271 freq = freq*1000;
    % Calcul de L (freq en Hz, Imag en Omhs donc L en H)
273 L = Imag./(2*pi*freq);
    % Enregistrement
275 if ~isempty(fichier)
        save(fichier,'freq','ampl','Module','Phase','Reel','Imag','L')
277     if sum(C(:))
            Courant = C;
279         save(fichier,'-append','Courant')
        end
281     disp(['Les données ont été sauvées dans ' fichier '.'])
    end
283
    % Affichage
285 figure('Name',titre)
    subplot(221)
287 plot(freq,20*log10(Module))
```

```
   set(gca,'Xscale',echelle);
289 ylabel('Module (dB)')
   subplot(222)
291 plot(freq,unwrap(Phase))
   set(gca,'Xscale',echelle);
293 ylabel('Phase (degrés)')
   subplot(223)
295 plot(freq,Reel,'b',freq,Imag,'r')
   set(gca,'Xscale',echelle);
297 xlabel('Fréquence (Hz)')
   legend('Partie réelle (Omhs)','Partie imaginaire (Omhs)')
299 subplot(224)
   plot(freq,L)
301 set(gca,'Xscale',echelle);
   xlabel('Fréquence (Hz)')
303 ylabel('Inductance (H)')
   if sum(C(:))
305     plot(freq,C)
       ylabel('Courant (mA)')
307 end
```

Script n°2

Déplacement du bras robotisé

```
function robot(action,x,y,z,vx,vy,vz);
% robot(action,deltax,deltay,deltaz)
% robot(action,deltax,deltay,deltaz,vx,vy,vz)
%
% Commande du Charly Robot via une carte d'interface.
%
% action -> 'zero' : prend la position actuelle comme origine
%           'avance' : déplacement relatif
%             (avance d'abord sur x, puis sur y, puis sur z)
%           'moveto' : déplacement absolu
%             (avance d'abord dans le plan xy puis sur z)
%
% delta* -> déplacements ou coordonnées, en pas (80 pas = 1 mm)
% v* -> vitesse sur chaque axe, en pas par seconde
%       (défaut : 2000p/s , configurables au début du fichier)
%
% Cyril Ravat, mars 2006

if nargin==4
    vx=2000;
    vy=2000;
    vz=2000;
end

% Connexion de l'interface du robot
% Le port série est à configurer : /dev/ttySx ou COMx selon le système
% d'exploitation, x entier.
ROBOT_ID=serial('COM1','Terminator','CR');
fopen(ROBOT_ID)
% On spécifie au robot de prendre en compte les trois axes
fprintf(ROBOT_ID,'@07')
% Le robot renvoie '0' pour dire qu'il est ok, il faut supprimer ce caractère
% pour la suite.
fread(ROBOT_ID,1);
if strcmp(action,'avance')
    fprintf(ROBOT_ID,sprintf('@0A%g,%g,%g,%g,%g,%g,%g,%g',x,vx,0,vy,0,vz,0,32))
    % Très important : le robot renvoie '0' lorsque la course est finie. On le
    % lit et on le supprime.
    fread(ROBOT_ID,1);
    fprintf(ROBOT_ID,sprintf('@0A%g,%g,%g,%g,%g,%g,%g,%g',0,vx,y,vy,z,vz,0,32))
    fread(ROBOT_ID,1);
elseif strcmp(action,'moveto')
    fprintf(ROBOT_ID,sprintf('@0M%g,%g,%g,%g,%g,%g,%g,%g',x,vx,y,vy,z,vz,0,32))
    fread(ROBOT_ID,1);
elseif strcmp(action,'zero')
```

```
46      fprintf(ROBOT_ID,'@0n7');
        % Ne sert a priori à rien, la modification dans le robot doit être
48      % automatique (pas besoin d'attendre), mais on ne sait jamais.
        fread(ROBOT_ID,1);
50  end

52  fclose(ROBOT_ID)
    delete(ROBOT_ID)
```

Script n°3

Acquisition d'images

```
1  function acq_surface(fichier,dimensions,signal,parametres)
   % acq_surface(fichier)
3  % acq_surface(fichier,dimensions)
   % acq_surface(fichier,dimensions,signal)
5  % acq_surface(fichier,dimensions,paramètres)
   % acq_surface(fichier,dimensions,signal,paramètres)
7  %
   % Balayage d'une surface à l'aide du Charly Robot et de l'impédancemètre HP4192.
9  % Mesure d'impédance 4 voies ou 2 voies. Les valeurs de module sont en échelle
   % linéaire (et non en dB).
11 %
   % fichier : chaîne désignant le chemin vers un fichier où seront stockés
13 %           les mesures effectuées. X et Y sont des vecteurs lignes contenant
   %           les coordonnées en pas de robot (80 pas = 1 mm), ampl et freq les
15 %           réglages de l'excitation (scalaires), Module, Phase, Reel et Imag
   %           les valeurs d'impédance mesurées, cour et Courant les valeurs
17 %           d'intensité si la mesure avec suivi en courant est activée (cf
   %           variable parametres). L'affichage peut être effectué ultérieurement
19 %           grâce à la fonction affichage.
   %           Pour ne pas sauvegarder, mettre une chaîne vide.
21 %
   % dimensions : vecteur contenant les dimensions et les pas du balayage
23 %              Si vide, carré de 10mm sur 10mm, pas de 0.25mm sur x et y
   %              Si une seule composante, c'est la longueur du côté du carré en
25 %              mm, le pas sur x et y vaut côté/40
   %              Si deux composantes, ce sont les longueurs des côtés du rectangle
27 %              en mm sur x et y, le pas sur x vaut côté_x/40 (resp. sur y)
   %              Si quatre composantes : [côté_x côté_y pas_x pas_y]
29 %
   % signal : vecteur à deux composantes [amplitude fréquence]
31 %          amplitude en V (défaut : la valeur courante du HP4192)
   %          fréquence en Hz (défaut : la valeur courante du HP4192)
33 %
   % paramètres : liste à trois composantes {mesure mode affichage}
35 %   mesure : '4fils' -> mesure 4 fils, mesure de R et X (en Omhs)
   %                       (valeur par défaut)
37 %                   => Les masses des 4 coax sont reliées.
   %                   => Les H sont reliés entre eux, les L sont reliés entre eux.
39 %                   => Le composant est mis entre les H et les L pour une mesure
   %                   d'impédance. Le composant parcouru par le courant est entre
41 %                   les Xcur et celui aux bornes duquel on mesure la tension est
   %                   entre les Xpot pour une mesure de transimpédance.
43 %            '2fils' -> mesure 2 fils (voies A et B), mesure du gain (en
   %                       linéaire) et de la phase (en degrés)
45 %            un nombre -> mesure 4 fils avec suivi en courant
```

```
%                   => le nombre est la valeur de l'intensité à suivre en mA
%                   => au début de chaque image, le niveau en tension est réglé
%                   afin d'obtenir la valeur de courant la plus proche de celle
%                   souhaitée. Le réglage est effectué à 5mV près (précision du
%                   HP4192). Attention, aucun réglage ne sera effectué si
%                   l'intensité correspondant à la valeur initiale de tension
%                   (définie par le variable signal) est inférieure à la valeur
%                   souhaitée : il vaut mieux régler cette valeur à 1.1V.
%                   => les courants sont relevés et enregistrés : cour contient
%                   la valeur de courant au premier point (scalaire), Courant
%                   contient la cartographie des valeurs d'intensité.
%   mode : 'spirale' -> spirale, position initiale au milieu (valeur par défaut)
%          'discontinu' -> lignes horizontales, toujours de la gauche vers la
%                          droite, avec éloignement durant la durée de
%                          détection, position initiale "en haut à gauche"
%          'continu' -> lignes horizontales, en sens inverse une fois sur deux,
%                       sans éloignement durant la durée de détection, position
%                       initiale "en haut à gauche"
%          '[mode]+' -> avec un '+' à la fin d'une des trois valeurs précédentes
%                       demandera au robot, à chaque pas, de se surélever de 2mm
%                       et donc de déplacer le capteur sans frottement sur la
%                       pièce. L'acquisition peut prendre beaucoup plus de temps.
%   affichage : 'non' -> on n'affiche que l'avancement de l'acquisition, une
%                        ligne par acquisition, choix conseillé si beaucoup
%                        d'acquisitions sont lancées automatiquement à la suite
%               'oui' -> on affiche dans la console des données relatives à
%                        l'acquisition et on affiche à la fin les images
%                        obtenues (valeur par défaut)
%
% Les réglages par défaut sont modifiables au début du fichier si besoin.
%
% Exemples :
%  acq_surface('essai.mat')
%  acq_surface('essai.mat',[15 5 0.3 0.1])
%  acq_surface('essai.mat',[15 5 0.3 0.1],[1.1 4e6])
%  acq_surface('essai.mat',[15 5 0.1 0.1],{'4fils' 'continu' 'oui'})
%  acq_surface('essai.mat',[],[1.1 4e6],{'4fils' 'continu' 'oui'})
%  acq_surface('essai.mat',[15 5 0.3 0.1],[1.1 4e6],{'2fils' 'spirale' 'oui'})
%  acq_surface('essai.mat',[15 5 0.3 0.1],[1.1 4e6],{'2fils' 'spirale+' 'oui'})
%
% Cyril Ravat, octobre 2005 - avril 2008

% Réglages par défaut
 %%% Toute modification concernant le comportement par défaut ne doit être %%%
 %%% effectuée QUE sur ces quelques lignes, et surtout pas sur le reste du %%%
 %%% programme. Cela permet de ne pas avoir à se souvenir ce qui a été     %%%
 %%% modifié précédemment.                                                 %%%
defaut_excursion = 10;
defaut_nombre_pas = 40; % Doit obligatoirement être pair
defaut_mesure = '4fils';
defaut_mode = 'spirale';
defaut_affichage = 'oui';
% Paramètre de connexion à l'impédancemètre : 16 au SATIE, 17 au LGEP
% (réglable par les interrupteurs à l'arrière de l'appareil, cf. le paragraphe
% 3-117 de la documentation).
defaut_connexion = 17;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pour les communications avec les deux appareils :
% Ils sont vus par Matlab comme des fichiers à l'intérieur desquels on écrit
% grâce à la fonction fprintf. La connexion est ouverte en début de fonction, et
% n'est refermée qu'à la fin. Aucune autre connexion n'est possible
% simultanément.
```

```
% Ces "fichiers" ont leur nom stocké dans deux variables globales HP4192_ID et
% ROBOT_ID. Ce qui veut dire que même si la fonction est quittée précipitamment,
% il est possible de communiquer avec les appareils en tapant :
%  global HP4192_ID ROBOT_ID
% pour récupérer les variables dans l'espace de travail courant, puis en
% utilisant la fonction fprintf.
%
% Par exemple :
% fprintf(HP4192_ID,'Tx'); [x=1-3] -> mettre le trigger en position x
% fprintf(HP4192_ID,'AxBy'); -> mettre la voie sur la position x, B sur y
% fprintf(HP4192_ID,'EX'); -> prendre une mesure
% fscanf(HP4192_ID) -> lire la mesure prise
%
% fprintf(ROBOT_ID,sprintf('@0a%g,%g,%g,%g,%g,%g,%g,%g',x,vx,y,vy,z,vz,0,32));
% -> avancer de x,y,z pas aux vitesses vx,vy,vz en pas par seconde
% fprintf(ROBOT_ID,sprintf('@0m%g,%g,%g,%g,%g,%g,%g,%g',x,vx,y,vy,z,vz,0,32));
% -> aller au point x,y,z (en pas) aux vitesses vx,vy,vz en pas par seconde
% (le '0,32' à la fin ne veulent rien dire mais sont obligatoires selon la
% documentation du Charly Robot)
%
% SURTOUT NE PAS OUBLIER DE FERMER CES FICHIERS :
%  fclose(HP4192_ID);
%  delete(HP4192_ID);
%  clear global HP4192_ID
% et idem avec ROBOT_ID.
%
% Si le programme est volontairement ou non arrêté avant la fin, les connexions
% doivent être récupérées (grâce à 'global XXXX_ID') et fermées grâce aux lignes
% ci-dessus. Ceci peut être automatiquement fait en lançant simplement la
% fonction fermer_proprement.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Gestion du temps écoulé : départ du décompte
debut = clock;

% Vérification et récupération des paramètres
if exist('dimensions','var') && ~isempty(dimensions)
    excursion_x = dimensions(1);
    if length(dimensions) > 1
        excursion_y = dimensions(2);
        if length(dimensions) == 4
            pas_x = dimensions(3);
            pas_y = dimensions(4);
        end
    end
end
if exist('signal','var') && length(signal)==2
    ampl = signal(1);
    % Attention : dans le reste du programme, la fréquence est en kHz !
    freq = signal(2)/1000;
end
if exist('signal','var') && length(signal)>2
    parametres = signal;
end
if exist('parametres','var')
    % Si l'utilisateur a bien entré une liste
    if iscell(parametres)
        mesure = parametres{1};
        mode = parametres{2};
        affichage = parametres{3};
    % Sinon, on imagine qu'il a entré un tableau (et donc une chaîne unique, au
    % final). On se sert du fait que mesure fait toujours 5 caractères et que
    % affichage fait toujours 3 caractères.
```

```
    else
173         mesure = parametres(1:5);
            mode = parametres(6:end-3);
175         affichage = parametres(end-2:end);
    end
177 else
    mesure = defaut_mesure;
179 mode = defaut_mode;
    affichage = defaut_affichage;
181 end
remonte = mode(end);
183 if remonte == '+'
    mode = mode(1:end-1);
185 end

187 % Initialisation du HP4192
affiche('Initialisation de l''impédancemètre HP4192...',affichage)
189 global HP4192_ID
% Au SATIE, HP4192_ID = gpib('ni',0,16);
191 % Au LGEP, HP4192_ID = gpib('ni',0,17);
HP4192_ID = gpib('ni',0,defaut_connexion);
193 fopen(HP4192_ID)
% T3 -> trigger en position manuelle
195 % F1 -> récupérer A/B/C
fprintf(HP4192_ID,'T3;F1')
197 if ~exist('ampl','var')
    % Lecture de l'amplitude
199 fprintf(HP4192_ID,'OLR;EX')
    reponse = fscanf(HP4192_ID);
201 ampl = str2double(reponse(34:43));
    % Lecture de la fréquence
203 fprintf(HP4192_ID,'FRR;EX')
    reponse = fscanf(HP4192_ID);
205 freq = str2double(reponse(34:43));
else
207 % ampl doit être compris entre 0 et 1.1 et multiple de 5mV
    ampl = median([0 1.1 0.005*floor(ampl/0.005) ]);
209 % Envoi de l'amplitude et de la fréquence à l'impédancemètre
    fprintf(HP4192_ID,['FR' num2str(freq) 'EN'])
211 fprintf(HP4192_ID,['OL' num2str(ampl) 'EN'])
end
213 if ischar(mesure)
    if strcmp(mesure,'2fils')
215     % A5B2 -> obtenir B-A en dB et la phase en degrés
        fprintf(HP4192_ID,'A5B2')
217 else
        % A2 -> obtenir R et X en Omhs
219     fprintf(HP4192_ID,'A2')
    end
221 else
    % Réglage du niveau de tension pour satisfaire le niveau de courant
223 fprintf(HP4192_ID,'TA;EX');
    reponse = fscanf(HP4192_ID);
225 cour = str2double(reponse(34:43));
    if cour < mesure
227     affiche(['  L''intensité vaut actuellement ' reponse(34:43) ' mA.'],
            affichage)
        affiche('  Aucun réglage sur l''amplitude n''est effectué.',affichage)
229 else
        % L'amplitude est proportionnelle au courant.
231     % L'appareil est précis à
        ampl = median([0 1.1 0.005*floor((ampl*mesure/cour)/0.005) ]);
233     fprintf(HP4192_ID,['OL' num2str(ampl) 'EN;TA;'])
```

```
        % Pause pour que la valeur ait le temps de se stabiliser
235     pause(1)
        fprintf(HP4192_ID,'EX')
237     reponse = fscanf(HP4192_ID);
        cour = str2double(reponse(34:43));
239     % Si ce n'est pas bon, on va 5mV par 5mV vers le bon courant
        if abs( cour-mesure ) > 5e-2
241         compteur = 1;
            signe = sign(mesure-cour);
243         while ( sign(mesure-cour(compteur)) == signe ) && ( compteur < 50 )
                compteur = compteur+1;
245             ampl(compteur) = ampl(compteur-1)+0.005*signe;
                fprintf(HP4192_ID,['OL' num2str(ampl(compteur),2) 'EN;TA;'])
247             % Pause pour que la valeur ait le temps de se stabiliser
                pause(1)
249             fprintf(HP4192_ID,'EX')
                reponse = fscanf(HP4192_ID);
251             cour(compteur) = str2double(reponse(34:43));
            end
253         [cour compteur] = min(abs(cour-mesure));
            ampl = ampl(compteur);
255     end
        affiche([' L''amplitude a été réglée, l''intensité vaut ' num2str(cour) '
            mA.'], affichage)
257 end
end
259
% Déplacements : gestion des excursions et des pas
261 if ~exist('excursion_x','var')
    excursion_x = defaut_excursion;
263 end
if ~exist('excursion_y','var')
265     excursion_y = excursion_x;
end
267 % Si les pas ne sont pas renseignés : valeurs par défaut
if ~exist('pas_x','var')
269     pas_x = excursion_x/defaut_nombre_pas;
    pas_y = excursion_y/defaut_nombre_pas;
271     nx = defaut_nombre_pas+1;
    ny = nx;
273 % Sinon, vérification pour correspondre à des nombres de pas entiers et pour que
% il y ait autant de points sur x que sur y si on balaye en spirale.
275 else
    nx = excursion_x/pas_x+1;
277     ny = excursion_y/pas_y+1;
    if ~strcmp(mode,'spirale')
279         if mod(nx,1)
            nx = round(nx);
281             pas_x = excursion_x/(nx-1);
            affiche([' La variable pas_x a été modifiée, et vaut ' num2str(pas_x) '
                mm.'],affichage)
283         end
        if mod(ny,1)
285             ny = round(ny);
            pas_y = excursion_y/(ny-1);
287             affiche([' La variable pas_y a été modifiée, et vaut ' num2str(pas_y) '
                mm.'],affichage)
        end
289     else
        if nx~=ny || ~mod(nx,2) || ~mod(ny,2)
291             nx = max(2*round((nx-1)/2)+1,2*round((ny-1)/2)+1);
            ny = nx;
293             pas_x = excursion_x/(nx-1);
```

```
            pas_y = excursion_y/(ny-1);
            affiche('  La spirale doit avoir autant de lignes que de colonnes, et',
                affichage)
            affiche('  en nombre entier et impair. Les pas sont recalculés :',
                affichage)
            affiche(['    pas_x = ' num2str(pas_x) ' mm , pas_y = ' num2str(pas_y) '
                mm.'],affichage)
        end
    end
end
X = round(80*(0:pas_x:excursion_x));
Y = round(80*(0:pas_y:excursion_y));

% Génération des matrices de déplacement (contenant des indices)
affiche('Génération des matrices de déplacement...',affichage)
% Le retour à l'origine est effectué automatiquement à la fin des mesures, on ne
% le prend pas en compte ici.
% Ligne par ligne toujours de la gauche vers la droite (depuis le point
% en haut à gauche)
depl_x = [];
depl_y = [];
depl_z = [];
switch mode
    case 'discontinu'
        for y = 1:ny
            depl_x = [depl_x 1:nx nx nx 1];
            depl_y = [depl_y y*ones(1,nx) y y+1 y+1];
            depl_z = [depl_z zeros(1,nx) -2 -2 -2];
        end
        % On enlève les deux derniers points qui commenceraient une nouvelle
        % ligne
        depl_x = depl_x(1:end-3);
        depl_y = depl_y(1:end-3);
        depl_z = depl_z(1:end-3)*80;
    case 'continu'
        % Ligne par ligne en sens inverse une fois sur deux, sens horizontal
        % (depuis le point en haut à gauche)
        for y = 1:ny
            depl_x = [depl_x mod(y-1,2)*(nx+1)+(2*mod(y,2)-1)*(1:nx)];
            depl_y = [depl_y y*ones(1,nx)];
        end
        depl_z = zeros(size(depl_x));
    otherwise % Si faute de frappe, ça fonctionne aussi
        % En spirale (depuis le point central)
        x = (nx+1)/2;
        depl_x = x;
        depl_y = x;
        for y = 1:x-1
            depl_x = [depl_x (x+y)*ones(1,2*y)   x+y-1:-1:x-y  (x-y)*ones(1,2*y)    x
                -y+1:x+y    ];
            depl_y = [depl_y     x-y+1:x+y    (x+y)*ones(1,2*y)   x+y-1:-1:x-y  (x-y)
                *ones(1,2*y)];
        end
        % Translation du repère
        X = X-X(x);
        Y = Y-Y(x);
        depl_z = zeros(size(depl_x));
end
% Réorganiser la mémoire pour que les variables depl_* y soient continues
pack
n = length(depl_x);
A = zeros(nx,ny);
B = A;
```

```
if ~ischar(mesure)
    Courant = A;
end

% Affichage
if strcmp(affichage,'oui')
    if freq < 1
        echo = ['Fréquence : ' num2str(freq*1000) ' Hz,'];
    elseif freq > 1000
        echo = ['Fréquence : ' num2str(freq/1000) ' MHz,'];
    else
        echo = ['Fréquence : ' num2str(freq) ' kHz,'];
    end
    disp([ echo sprintf('\b')])
    if ampl < 1
        echo = [ num2str(ampl*1000) ' mV'];
    else
        echo = [ num2str(ampl) ' V'];
    end
    disp(['Amplitude : ' echo ])
    disp(['Balayage d''une surface de ' num2str(excursion_x) ' mm (par pas de '
       num2str(pas_x) ' mm) sur x '])
    disp(['                        de ' num2str(excursion_y) ' mm (par pas de '
       num2str(pas_y) ' mm) sur y '])
    switch mode
        case 'discontinu'
            disp('                        en lignes horizontales, toujours de la
                gauche vers la droite')
        case 'continu'
            disp('                        en lignes horizontales, en sens inverse une
                fois sur deux')
        otherwise
            disp('                        en spirale centrée')
    end
end

% Initialisation du Charly Robot
affiche('Initialisation du Charly Robot...',affichage)
global ROBOT_ID
ROBOT_ID = serial('COM1','Terminator','CR');
fopen(ROBOT_ID)
% Spécifie au robot de prendre en compte les trois axes
fprintf(ROBOT_ID,'@07')
% Règle le zéro absolu
fprintf(ROBOT_ID,'@0n7')
% Le robot renvoie un '0' à chaque commande pour dire qu'il est ok, il faut
% supprimer ces caractères pour la suite. (cf robot.m)
fread(ROBOT_ID,2);
% Vitesses par défaut, en pas par seconde
vx = 2000;
vy = 2000;
vz = 2000;

% Gestion de l'avancement - initialisation
% Combien d'espaces dans la barre / d'étapes de raffraichissement
n_av = 25; % Paramètre arbitraire modifiable
if n>n_av
    pas_av = find(diff(floor((1:n)*n_av/n)));
else
    n_av = n;
    pas_av = 1:n;
end
% "Avancement : |---------------     | XX% XXXs"
```

```
411 disp([' Avancement : |' blanks(n_av) '| 0% 0s'])
    echo = blanks(n_av+9);
413 % Chaîne contenant des caractères d'effacement
    effacemax = blanks(n_av+12);
415 effacemax = strrep(effacemax,' ','\b');

417 % Mesures
    for compteur = 1:n
419     % Gestion de la barre d'avancement
        if ismember(compteur,pas_av)
421         efface = effacemax(1:2*(length(echo)-1));
            echo = [ '-' blanks(n_av-ceil(n_av*compteur/n)) '| ' num2str(ceil(100*
                compteur/n)) '%% ' num2str(round(etime(clock,debut))) 's'];
423         disp(sprintf([ efface echo ]))
        end
425     x=depl_x(compteur);
        y=depl_y(compteur);
427     z=depl_z(compteur);
        try
429         % Ordre au Charly Robot d'aller au point X(x),Y(y),z
            fprintf(ROBOT_ID,sprintf('@0M%g,%g,%g,%g,%g,%g,%g,%g',X(x),vx,Y(y),vy,z,vz
                ,0,32))
431         % Le robot renvoie '0' lorsque la course est finie. On le lit et le supprime.
            fread(ROBOT_ID,1);
433         % Déclenchement de la mesure : pas de mesure si on est "en l'air"
            if ~z
435             fprintf(HP4192_ID,'EX')
                % Lecture du résultat
437             reponse = fscanf(HP4192_ID);
                % Exploitation
439             if reponse(1) == 'N'
                    A(x,y) = str2double(reponse(5:15));
441             else
                    A(x,y) = NaN;
443             end
                if reponse(17) == 'N'
445                 B(x,y) = str2double(reponse(21:31));
                else
447                 B(x,y) = NaN;
                end
449             if ~ischar(mesure)
                    Courant(x,y) = str2double(reponse(34:43));
451             end
                if remonte == '+'
453                 % Le robot remonte le capteur avant de se déplacer. À la prochaine
                    % itération, il se déplacera d'abord sur (x,y) et ensuite sur z
455                 % Ordre au Charly Robot d'aller au point X(x),Y(y),-2mm
                    fprintf(ROBOT_ID,sprintf('@0M%g,%g,%g,%g,%g,%g,%g,%g',X(x),vx,Y(y),vy
                        ,-2*80,vz,0,32))
457                 % Le robot renvoie '0' lorsque la course est finie. On le lit et le
                        supprime.
                    fread(ROBOT_ID,1);
459             end
            end
461     catch
            disp('ERREUR -- attente de 60 secondes avant de repartir')
463         if ~exist('compteur2','var')
                compteur2 = 1;
465         else
                compteur2 = compteur2+1;
467             if compteur2 >= 50
                    % Arrêt prématuré
469                 error('50 erreurs consécutives -> Problème important')
```

```
            end
        end
        compteur = compteur-1;
        pause(60)
    end
end

% Avancement - fin
disp(sprintf([ effacemax(1:2*(length(echo)-3)) '100%% ' num2str(round(etime(clock,
    debut))) 's' ]))

% Retour à l'origine
fprintf(ROBOT_ID,sprintf('@0M%g,%g,%g,%g,%g,%g,%g,%g',X(depl_x(n)),vx,Y(depl_y(n)),vy
    ,-2*80,vz,0,32))
fread(ROBOT_ID,1);
fprintf(ROBOT_ID,sprintf('@0M%g,%g,%g,%g,%g,%g,%g,%g',X(depl_x(1)),vx,Y(depl_y(1)),vy
    ,-2*80,vz,0,32))
fread(ROBOT_ID,1);
fprintf(ROBOT_ID,sprintf('@0M%g,%g,%g,%g,%g,%g,%g,%g',X(depl_x(1)),vx,Y(depl_y(1)),vy
    ,0,vz,0,32))

% Récupération et formatage des résultats
affiche('Enregistrement des résultats...',affichage)
if strcmp(mesure,'2fils')
    Module = 10.^(A/20); % Le module est enregistré en linéaire
    Phase = B;
    Reel = Module.*cos(Phase*pi/180);
    Imag = Module.*sin(Phase*pi/180);
else
    Reel = A;
    Imag = B;
    Module = sqrt(Reel.*Reel+Imag.*Imag);
    Phase = atan2(Imag,Reel)*180/pi;
end
% Repositionnement de X et Y à zéro
X = X-X(1);
Y = Y-Y(1);
% Enregistrement
if nargin && ischar(fichier) && ~isempty(fichier)
    save(fichier,'X','Y','ampl','freq','Module','Phase','Reel','Imag')
    if ~ischar(mesure)
        save(fichier,'-append','cour','Courant')
    end
    affiche(['Les données ont été sauvées dans ' fichier '.'],affichage)
end

% Fermeture des connexions avec les appareils
fprintf(HP4192_ID,'T1')
fclose(HP4192_ID)
delete(HP4192_ID)
clear global HP4192_ID
fclose(ROBOT_ID)
delete(ROBOT_ID)
clear global ROBOT_ID

% Affichage des images
if strcmp(affichage,'oui')
    if freq<1
        echo = [ num2str(freq*1000) ' Hz'];
    elseif freq > 1000
        echo = [ num2str(freq/1000) ' MHz'];
    else
        echo = [ num2str(freq) ' kHz'];
```

```
529     end
        figure('Name',['Acquisition à ' echo ])
531     subplot(221)
        pcolor(X/80,Y/80,Module'), shading interp, axis image
533     set(gca,'YDir','reverse')
        title('Module')
535     xlabel('axe X (mm)')
        ylabel('axe Y (mm)')
537     subplot(222)
        pcolor(X/80,Y/80,Phase'), shading interp, axis image
539     set(gca,'YDir','reverse')
        title('Argument')
541     xlabel('axe X (mm)')
        ylabel('axe Y (mm)')
543     subplot(223)
        pcolor(X/80,Y/80,Reel'), shading interp, axis image
545     set(gca,'YDir','reverse')
        title('Partie réelle')
547     xlabel('axe X (mm)')
        ylabel('axe Y (mm)')
549     subplot(224)
        pcolor(X/80,Y/80,Imag'), shading interp, axis image
551     set(gca,'YDir','reverse')
        title('Partie imaginaire')
553     xlabel('axe X (mm)')
        ylabel('axe Y (mm)')
555 else
        t=clock;
557     disp(sprintf('\b -- Fin d''exécution à %0.0fh%02.0f''%02.0f''''.',t([4 5 6])))
    end
559
    % Gestion du temps écoulé : calcul et affichage du temps écoulé total
561 affiche(['Durée totale en secondes : ' num2str(etime(clock,debut))],affichage)

563 function affiche(chaine,affichage)
    if strcmp(affichage,'oui')
565     disp(chaine)
    end
```

Script n°4

Résolution d'erreur

```
function fermer_proprement
%function fermer_proprement
%
% Fermer proprement les connexions au Charly Robot et à l'analyseur
% réseau HP4192 après un arrêt inatendu d'une fonction d'aquisition
% comme acq_surface.
%
% Cyril Ravat, mars 2006

disp('**Vérification de la connexion de l''analyseur HP4192')
global HP4192_ID
if strcmp(class(HP4192_ID),'gpib')
    disp('La connexion avec l''analyseur était déjà établie.')
    % On remet le trigger de l'analyseur en mode automatique
    fprintf(HP4192_ID,'T1')
    % On ferme la connexion
    fclose(HP4192_ID);
    delete(HP4192_ID);
    disp('La connexion avec l''analyseur est maintenant fermée.')
else
    disp('La connexion avec l''analyseur n''était pas déjà établie.')
end
clear global HP4192_ID

disp('**Vérification de la connexion du Charly Robot')
global ROBOT_ID
if strcmp(class(ROBOT_ID),'serial')
    disp('La connexion avec le robot était déjà établie.')
    ok = input(sprintf('Ramener le robot à la position d''origine définie récemment ?
        (o/n)\n'),'s');
    if strcmp(ok,'o') || strcmp(ok,'oui')
        fprintf(ROBOT_ID,sprintf('@0M0,1500,0,1500,0,1500,0,32'))
        fread(ROBOT_ID,1);
    end
    fclose(ROBOT_ID);
    delete(ROBOT_ID);
    disp('La connexion avec le robot est maintenant fermée.')
else
    disp('La connexion avec le robot n''était pas déjà établie.')
end
clear global ROBOT_ID
```