

Liste des programmes

Interpolation d'une courbe bidimensionnelle

```
function [Y] = interpolation_1D (v,r) % v est le vecteur à interpoler, Y le vecteur interpolé, r est le
rapport ancienne/nouvelle résolution et T la distance entre deux points
n=length(v); % calcul de la taille de v
Y=zeros(1,r*n); % Création du vecteur Y qui est rempli avec des zéros

for p=1:r*n, % pour créer chaque nouveau point un par un
    Z=sinc((p/r -[1:n]*1)/1); % création d'un vecteur contenant tous les sinus cardinaux centrés en p/r
    (p/r est l'abscisse du nouveau point
    Y(1,p)= sum(v(1,:).*Z); % produit scalaire du vecteur à interpoler avec le vecteur sinus cardinal
end

% affichage du résultat final
plot([1:n],v(1:n),'blac') % affichage de la courbe de départ
hold on % permet la superposition de deux courbes
plot(([1:r*n]/r),Y(1:r*n),'r') % affichage de la courbe interpolée en rouge (remise à l'échelle de
départ)
```

Interpolation d'une surface tridimensionnelle

```
function [X] = interpolation_2D (V,r) % V est la matrice à interpoler, X la matrice interpolée et r
est le rapport nouvelle/ancienne résolution
t=length(V); % calcul de la taille de V
Y=zeros(r*t,r*t); % Création de la matrice Y qui est rempli avec des zéros
Z=zeros(r*t,r*t); % % Création de la matrice Z qui est rempli avec des zéros

[N,M] = meshgrid(1:t,1:t) % Créé une mat 10x10 N aux lignes identiques et aux colonnes remplies
de 1 à 10 et une mat 10x10 M aux colonnes identiques et aux lignes remplies de 1 à 10
for q=1:r*t, % pour avancer dans la matrice interpolée
    q % affiche l'état d'avancement du programme
    for p=1:r*t, % pour avancer dans la matrice interpolée
        Y = sinc((p/r -N*1)/1);
        Z = sinc((q/r -M*1)/1);
        SIN = Y.*Z; % matrice contenant tous les sinus cardinaux ( effectue un pdt terme à terme de Y
et Z
        X(p,q) = sum(sum(V.*SIN)); % somme de tous les éléments de la mat V.*SIN (qui est un pdt
terme à terme de la mat à interpoler ac celle des sin
    end
end
```

Interpolation d'une surface tridimensionnelle complexe

```
function [X] = interpolation_2D_complex (V,r) % V est la matrice complexe à interpoler, X la
matrice interpolée
% r est le rapport nouvelle/ancienne résolution ( il faut une valeur entière)
t=length(V); % calcul de la taille de V
```

% interpolation de la partie réelle

```
Vr = real(V);
Y=zeros(r*t,r*t); % Création de la matrice Y qui est rempli avec des zéros
Z=zeros(r*t,r*t); % % Création de la matrice Z qui est rempli avec des zéros

[N,M] = meshgrid(1:t,1:t) % Créé une mat 10x10 N aux lignes identiques et aux colonnes remplies
de 1 à 10
% et une mat 10x10 M aux colonnes identiques et aux lignes remplies de 1 à 10
for q=1:r*t, % pour avancer dans la matrice interpolée
    q % affiche l'état d'avancement du programme
    for p=1:r*t, % pour avancer dans la matrice interpolée
        Y = sinc((p/r -N*1)/1);
        Z = sinc((q/r -M*1)/1);
        SIN = Y.*Z; % matrice contenant tous les sinus cardinaux ( effectue un pdt terme à terme de Y
et Z
        Xr(p,q) = sum(sum(Vr.*SIN)); % somme de tous les éléments de la mat V.*SIN (qui est un pdt
terme à terme de la mat à interpoler ac celle des sin
    end
end
```

% interpolation de la partie imaginaire

```
Vim = imag(V);
Y=zeros(r*t,r*t); % Création de la matrice Y qui est rempli avec des zéros
Z=zeros(r*t,r*t); % % Création de la matrice Z qui est rempli avec des zéros

[N,M] = meshgrid(1:t,1:t) % Créé une mat 10x10 N aux lignes identiques et aux colonnes remplies
de 1 à 10 et une mat 10x10 M aux colonnes identiques et aux lignes remplies de 1 à 10
for q=1:r*t, % pour avancer dans la matrice interpolée
    q % affiche l'état d'avancement du programme
    for p=1:r*t, % pour avancer dans la matrice interpolée
        Y = sinc((p/r -N*1)/1);
        Z = sinc((q/r -M*1)/1);
        SIN = Y.*Z; % matrice contenant tous les sinus cardinaux ( effectue un pdt terme à terme de Y
et Z
        Xim(p,q) = sum(sum(Vim.*SIN)); % somme de tous les éléments de la mat V.*SIN (qui est un
pdt terme à terme de la mat à interpoler ac celle des sin
    end
end
```

X = Xr + i*Xim

Interpolation en fréquences

function [Y] = interpolation_ligne_frequence (V,r) % ne fonctionne que pour des r PAIRS !!!

```

[a,b,c] = size(V)
Y = zeros(a,b,c*r);
for i=1:a
    for j=1:b
        i,j
        for f=(c*r/2.0 - (r/2)):(c*r/2.0 + (r/2))
            Z=sinc((f/r - [1:c]*1)/1);
            % Y(a,b,f)= sum(V(a,b,:).*Z);
            for k=1:c
                Y(i,j,f) = Y(i,j,f) + V(i,j,k)*Z(k);
            end
        end
    end
end
end

Y = Y(:,:(c*r/2.0 - (r/2)):(c*r/2.0 + (r/2)));

```

Affichage des lignes de zeros de partie réelle et imaginaire

```

function lignes_zeros(V) % V doit être une matrice (tableau de dim 2)

% Affichage du point de départ
contour(real(V(:,:)),[0 0],'color','b')
hold on
contour(imag(V(:,:)),[0 0],'color','r')
axis square

```

Détection de singularités

```

function [Y,Z] = singularitegood (V) % Y donne les coords des singularités >0 et Z celles des <0
[n,l,m] = size(V);
Y = zeros(n*n,2);
Z = zeros(n*n,2);
a = zeros(9,1);
i=1;
j=1;

for x=1:(n/2)-1, % pour avancer point par point
    for y=1:(n/2)-1,
        % repérage des huit points voisins
        p = x*2;
        q = y*2;
        a(1,1) = V(p-1,q-1);
        a(2,1) = V(p,q-1);
        a(3,1) = V(p+1,q-1);
        a(4,1) = V(p+1,q);
        a(5,1) = V(p+1,q+1);
        a(6,1) = V(p,q+1);

```

```

a(7,1) = V(p-1,q+1);
a(8,1) = V(p-1,q);
a(9,1) = V(p-1,q-1);
b = unwrap(a); % pour déplier la phase
c = b(9,1) - b(1,1);

if (c > 2*pi - 0.01) % test pour savoir si on a une singularité positive
    Y(i,1)=q;
    Y(i,2)=p;
    i=i+1;
elseif (c < -2*pi + 0.01)
    Z(j,1)=q;
    Z(j,2)=p;
    j=j+1;

end
end
end

if i==1
    Y=zeros(1,2);

else
    i = i-1;
    Y= Y(1:i,1:2);
end

if j==1
    Z=zeros(1,2);
else
    j = j-1;
    Z= Z(1:j,1:2);
end
end

```

Suivi d'une singularité positive en fonction de la fréquence

```

function [N,ff] = suivi_negative (A,D,fd)

[n1,n2,n3] = size(A);
N = zeros(1,3,n3); % Suivi des coordonnées de la singularité en fonction de la fréquence
for i=1:n3
    N(1,3,i) = i;
end
N(1,1,fd) = D(1);
N(1,2,fd) = D(2);
difference2 = 10000;
dc = 30; % différence carrée maximale autorisée pour considérer que l'on a toujours la même singularité

```

```

figure
lignes_zeros(A(:, :, fd))
line('XData', D(1), 'YData', D(2), 'Marker', '+', 'LineStyle', 'none', 'color', 'blac')

i = fd;
while i > 0
    i = i - 1; % on recule fréquence par fréquence
    [P2, N2] = singularitegood(angle(A(:, :, i))); % On cherche toutes les singularités
    % Il faut savoir laquelle des singularités est celle que l'on suit
    [a, b] = size(N2);
    P(1, 1, i) = 0; P(1, 2, i) = 0;
    for j = 1:a;
        difference = (N2(j, 1) - N(1, 1, i + 1)) * (N2(j, 1) - N(1, 1, i + 1)) + (N2(j, 2) - N(1, 2, i + 1)) * (N2(j, 2) - N(1, 2, i + 1));
        if (difference < difference2);
            if difference < dc;
                difference2 = difference;
                N(1, 1, i) = N2(j, 1); N(1, 2, i) = N2(j, 2);
            end
        end
    end
    end
    pause
    difference2 = 10000;
    figure
    lignes_zeros(A(:, :, i))
    line('XData', N(1, 1, i), 'YData', N(1, 2, i), 'Marker', '+', 'LineStyle', 'none', 'color', 'blac')
    if N(1, 1, i + 1) == 0
        break
    end
end

ff = i + 1;

N = N(:, :, ff:fd);

```

Suivi d'une singularité négative en fonction de la fréquence

```

function [P, ff] = suivi_positive (A, D, fd)

[n1, n2, n3] = size(A);
P = zeros(1, 3, n3); % Suivi des coordonnées de la singularité en fonction de la fréquence
% On remplit la colonne des fréquences avec des i
for i = 1:n3
    P(1, 3, i) = i;
end
P(1, 1, fd) = D(1);
P(1, 2, fd) = D(2);
difference2 = 10000;

```

dc = 30; % différence carrée maximale autorisée pour considérer que l'on a toujours la même singularité

```
figure
lignes_zeros(A(:,fd))
line('XData',D(1),'YData',D(2),'Marker','+', 'LineStyle','none','color','blac')

for i=(fd+1):n3; % on avance fréquence par fréquence
    [P2,N2] = singularitegood(angle(A(:,i))); % On cherche toutes les singularités
    % Il faut savoir laquelle des singularités est celle que l'on suit
    [a,b] = size(P2);
    P(1,1,i)=0; P(1,2,i)=0;
    for j=1:a;
        difference = (P2(j,1)-P(1,1,i-1))*(P2(j,1)-P(1,1,i-1)) + (P2(j,2)-P(1,2,i-1))*(P2(j,2)-P(1,2,i-1));
        if (difference<difference2);
            if difference<dc;
                difference2 = difference;
                P(1,1,i) = P2(j,1); P(1,2,i) = P2(j,2);
            end
        end
    end
end
pause
difference2 = 10000;
figure
lignes_zeros(A(:,i))
line('XData',P(1,1,i),'YData',P(1,2,i),'Marker','+', 'LineStyle','none','color','blac')
if P(1,1,i)==0
    break
end
end

ff = i-1;

P = P(:,fd:ff);
```

Traçage des singularités

```
function tracage(v)

[a,b,c] = size(v);

for z=1:c
    line('EraseMode','XData',v(1,1,z),'YData',v(1,2,z),'Marker','+', 'LineStyle','none','color','b')
    xlim([1 72])
    ylim([1 72])
    pause
end
```