

Rapport projet de filtration

PROF : BAKHER ZINEELABIDINE
Réaliser par : BAKRI ILYAS
BELFAIDA BARAE



- La filtration est un processus couramment utilisé dans de nombreux domaines tels que l'industrie alimentaire, l'industrie pharmaceutique ou encore l'industrie pétrochimique pour ne citer que ceux-là. C'est un processus qui permet de séparer les différentes particules d'un mélange en utilisant une méthode de tamisage ou de séparation par gravité. Ce processus est généralement réalisé manuellement, ce qui peut être coûteux, chronophage et sujet à des erreurs humaines. L'objectif de ce projet est de digitaliser le processus de filtration en utilisant l'outil Matlab. L'application Matlab développée dans le cadre de ce projet permet de simuler numériquement le processus de filtration. Elle permet également de visualiser les résultats de manière graphique pour mieux comprendre le processus. Cette application offre plusieurs avantages, notamment en termes de précision, d'efficacité et de fiabilité. En effet, la digitalisation permet de réduire les erreurs humaines et d'optimiser le processus de filtration pour obtenir des résultats plus précis. Le rapport présenté ici présente les différentes étapes de développement de l'application Matlab de digitalisation du processus de filtration. Il décrit les choix méthodologiques opérés, les différentes étapes de développement de l'application et les résultats obtenus. Il s'adresse aux professionnels de l'industrie et de la recherche qui souhaitent optimiser leur processus de filtration.



Interface du projet



filtrum

Projet N°1 de l'operation de filtration



Cette application vous permet de calculer les valeurs suivant :

1. la vitesse en fût vide
2. le pourcentage remplis dans le gâteau
3. le nombre REYNOLDS de pores correspondant (afiché devant la valeur "laminaire" ou bien "turbulent")
4. la perméabilité du gâteau et la résistance de l'unité de surface de support équivalente en fonction de temps.
5. figure de variation de Z en fonction de temps avec équation de tendance et la valeur de R2

l'opération de lavage est programmée puenb la perte de charge totale dans le filtre atteint 0,6 mbar:

6. Tracer la variation de perte de charge en fonction de temps.
7. Calculer le temps de fonctionnement d'un cycle avant de programmer l'opération de lavage.

Prof: BAKHER ZINEELABIDINE

Realiser par : BAKRI ILYAS et BELFAIDA BARAE




Diagram illustrating a filtration setup. A funnel containing a filter paper (Papier filtre) is placed over a flask (Verre de montre) containing a heterogeneous mixture (Mélange hétérogène). The filtrate (Filtrat) is collected in the flask. The substrate (Substrat) is shown below the flask.

Input fields for parameters:

- débit (m³/h)
- viscosité du liquide (Pl)
- Porosité du gateau
- diamètre du grain (m)
- Masse volumique (kg/m³)
- hauteur (mm)
- vitesse en fut vide (m/s)
- Permiabilité du gateau (m²)
- N Reynolds
- régime :
- Résistance (pa/m²)
- % dans le gateau

Buttons:

- calculer

Messages :

Code

```
% Callbacks that handle component events
methods (Access = private)

% Button pushed function: calculerButton
function calculerButtonPushed(app, event)

    MV = app.MassevolumiqueEditField.Value;
    viscosite = app.viscositduliquideEditField.Value;
    porosite = app.PorositdugateauEditField.Value;
    debit = app.dbitEditField.Value;
    diametreD = app.diamtredugrainEditField.Value;
    Z = app.hauteurEditField.Value;

    % calcule de vitesse en fut vide
    vitesse = (debit/3600)/1;

    app.vitesseenfutvideEditField.Value = vitesse;

    % calcule de poucentage dans le gateau

    app.danslegateauEditField.Value=(1-porosite)*100;

    % calcule du nombre de reynoldes

    app.NReynoldsEditField.Value=(MV*vitesse)/((6/diametreD)*(1-porosite)*viscosite);

    if app.NReynoldsEditField.Value < 2000.00 && app.NReynoldsEditField.Value >= 0.00
        app.Label.Text = 'Laminaire';

    elseif app.NReynoldsEditField.Value > 2000.00
        app.Label.Text = 'Turbulant';
    end
```


Les abréviations utilisés dans le code du bouton de calcul

```
% Button pushed function: calculerButton
```

```
function calculerButtonPushed(app, event)
```

```
    MV = app.MassevolumiqueEditField.Value;  
    viscosite = app.viscositduliquideEditField.Value;  
    porosite = app.PorositdugateauEditField.Value;  
    debit = app.dbitEditField.Value;  
    diametreD = app.diamtredugrainEditField.Value;  
    Z = app.hauteurEditField.Value;
```

Code vitesse fut vide

```
% calcule de vitesse en fut vide  
vitesse = (debit/3600)/1;  
  
app.vitesseenfutvideEditField.Value = vitesse;
```

Code pourcentage du gâteau

```
% calcule de poucentage dans le gateau  
app.danslegateauEditField.Value=(1-porosite)*100;
```


Nombre de Reynolds

```
% calcule du nombre de reynoldes

app.NReynoldsEditField.Value=(MV*vitesse)/((6/diametreD)*(1-porosite)*viscosite);

if app.NReynoldsEditField.Value < 2000.00 && app.NReynoldsEditField.Value >= 0.00
    app.Label.Text = 'Laminaire';

elseif app.NReynoldsEditField.Value > 2000.00
    app.Label.Text = 'Turbulant';

end
```

Code perméabilité

```
%calcule de la perméabilité du gâteau
```

```
app.PermiabilitdugateauEditField.Value = (porosite)^3/(4.16*((6/diametreD)^2)*((1-porosite)^2)*viscosite);
```

Code resistance du gâteau

```
%calcule de la résistance du gateau
```

```
app.RsistanceEditField.Value = (Z*10^-3)/app.PermiabilitdugateauEditField.Value;
```


Code des lampes

```
% checking if edit field is empty by using lamps
% check if editfield is empty

if MV == 0
    app.Lamp.Color = 'red';

else
    app.Lamp.Color = 'green';
end

% check if editfield is empty

if viscosite == 0
    app.Lamp_2.Color = 'red';

else
    app.Lamp_2.Color = 'green';
end
```

```
% check if editfield is empty

if porosite == 0
    app.Lamp_3.Color = 'red';

else
    app.Lamp_3.Color = 'green';
end

% check if editfield is empty

if diametreD == 0
    app.Lamp_4.Color = 'red';

else
    app.Lamp_4.Color = 'green';
end

% check if editfield is empty

if debit == 0
    app.Lamp_5.Color = 'red';

else
    app.Lamp_5.Color = 'green';
end
```

```
% check if editfield is empty

if debit == 0
    app.Lamp_5.Color = 'red';

else
    app.Lamp_5.Color = 'green';
end
```

```
% check if editfield is empty

if Z == 0
    app.Lamp_7.Color = 'red';

else
    app.Lamp_7.Color = 'green';
end
```

```

% Value changed function: MassevolumiqueEditField
function MassevolumiqueEditFieldValueChanged(app, event)
    value = app.MassevolumiqueEditField.Value;

    switch value
        case 0
            app.Lamp.Color = [0.90,0.90,0.90];
        otherwise
            app.Lamp.Color = 'green';
    end
end

% Value changed function: viscositduliquoteEditField
function viscositduliquoteEditFieldValueChanged(app, event)
    value = app.viscositduliquoteEditField.Value;
    switch value
        case 0
            app.Lamp_2.Color = [0.90,0.90,0.90];
        otherwise
            app.Lamp_2.Color = 'green';
    end
end

% Value changed function: PorositdugateauEditField
function PorositdugateauEditFieldValueChanged(app, event)
    value = app.PorositdugateauEditField.Value;
    switch value
        case 0
            app.Lamp_3.Color = [0.90,0.90,0.90];
        otherwise
            app.Lamp_3.Color = 'green';
    end
end

```

```

% Value changed function: diamtredugrainEditField
function diamtredugrainEditFieldValueChanged(app, event)
    value = app.diamtredugrainEditField.Value;
    switch value
        case 0
            app.Lamp_4.Color = [0.90,0.90,0.90];
        otherwise
            app.Lamp_4.Color = 'green';
    end
end

% Value changed function: dbitEditField
function dbitEditFieldValueChanged(app, event)
    value = app.dbitEditField.Value;
    switch value
        case 0
            app.Lamp_5.Color = [0.90,0.90,0.90];
        otherwise
            app.Lamp_5.Color = 'green';
    end
end

% Value changed function: hauteurEditField
function hauteurEditFieldValueChanged(app, event)
    value = app.hauteurEditField.Value;
    switch value
        case 0
            app.Lamp_7.Color = [0.90,0.90,0.90];
        otherwise
            app.Lamp_7.Color = 'green';
    end
end

```

Code message d'erreur

```
% Error message
if MV == 0 || viscosite == 0 || porosite == 0 || diametreD == 0 || debit == 0 || Z==0

    app.Label_2.Text = '!! SVP fait remplit les champs où la lamp est en rouge !!';
    app.Label_2.BackgroundColor = [0.91,0.57,0.57];
    app.Label_2.FontColor = 'red';

else
    app.Label_2.BackgroundColor = 'None';
    app.Label_2.Text = 'Votre calcule est pret';
    app.Label_2.FontColor = 'green';

end
```


L'interface des courbes

ajouter les données

Temps Z dP

Voir les données du tableau dans ce fichier

Tracer la courbe de $z=f(t)$

Tracer la courbe de $dP=f(t)$

Temps	Z	dP
-------	---	----

vider le tableau

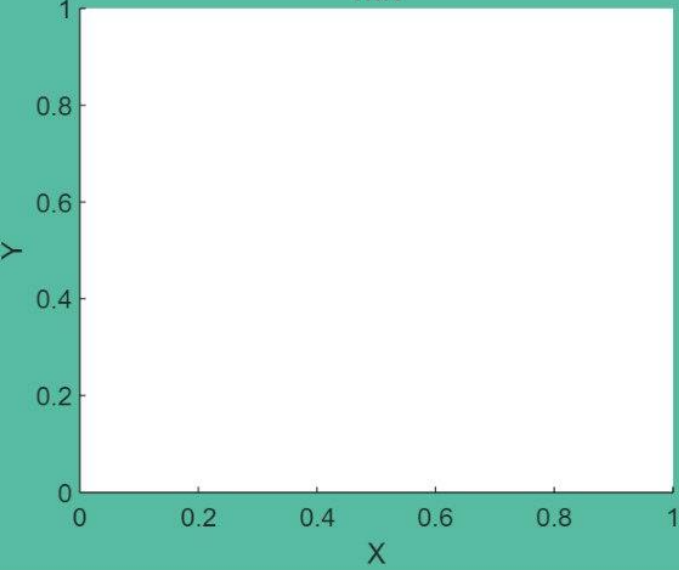
Voir l'équation de ghrap

Voir la valeur de R^2

Calculer le temps pour

Z ou dP =

Title



Le code utilisé pour ajouté les données au tableau

```
% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: ajouterlesdonneesButton
    function ajouterlesdonneesButtonPushed(app, event)

        Temps = app.TempsEditField.Value;
        Z = app.ZEditField.Value;
        dP = app.dPEditField.Value;

        newData = {Temps, Z, dP};
        currentData = get(app.UITable, 'Data');
        newTableData = [currentData; newData];
        set(app.UITable, 'Data', newTableData);

        app.TempsEditField.Value = 0;
        app.ZEditField.Value = 0;
        app.dPEditField.Value = 0;

    end
```

Le code de traçage des courbes à partir des données du tableau

```
% Button pushed function: TracerlacourbedezftButton  
function TracerlacourbedezftButtonPushed(app, event)
```

```
x = table2array(app.UITable.Data(:, "Temps"));  
y = table2array(app.UITable.Data(:, "Z"));  
plot(app.UIAxes, x, y);  
xlabel(app.UIAxes, 'Temps (min)');  
ylabel(app.UIAxes, 'Z (mm)');  
title(app.UIAxes, 'Z=f(t)');
```

```
app.Label.Text = ' ';  
app.Label_2.Text = ' ';
```

```
end
```

```
% Button pushed function: TracerlacourbededPftButton  
function TracerlacourbededPftButtonPushed(app, event)
```

```
x = table2array(app.UITable.Data(:, "Temps"));  
y = table2array(app.UITable.Data(:, "dP"));  
plot(app.UIAxes, x, y);  
xlabel(app.UIAxes, 'Temps (min)');  
ylabel(app.UIAxes, 'dP (mbar)');  
title(app.UIAxes, 'dP=f(t)');
```

```
app.Label.Text = ' ';  
app.Label_2.Text = ' ';
```

```
end
```


Le code de la lecture des données d'un tableau d'un fichier externe

```
% Button pushed function: VoirlesdonneesdutableaudanscefichierButton  
function VoirlesdonneesdutableaudanscefichierButtonPushed(app, event)  
    t = readtable(app.EditField_2.Value,"sheet",1, 'Range', 'A:C');  
    app.UITable.Data = t;  
end
```

Le code d'affichage de l'équation de la courbe

```
% Button pushed function: Voir l'équation de la courbe
function VoirEquationDeLaCourbePushed(app, event)
    % Retrieve the x and y data of the plot
    x = app.UIAxes.Children.XData;
    y = app.UIAxes.Children.YData;

    % Generate the equation of the plot
    p = polyfit(x, y, 1);
    eqn = sprintf('y = %.2fx + %.2f', p(1), p(2));

    % Update the text of the label with the equation
    app.Label.Text = eqn;
end
```

Le code d'affichage de la valeur de R^2 de la courbe

```
% Button pushed function: VoirLavaleurdeRButton
function VoirLavaleurdeRButtonPushed(app, event)

    % Retrieve the x and y data of the plot
    x = app.UIAxes.Children.XData;
    y = app.UIAxes.Children.YData;

    % Generate the R-squared value
    p = polyfit(x, y, 1);
    yfit = polyval(p,x);
    yresid = y - yfit;
    SSresid = sum(yresid.^2);
    SStotal = (length(y)-1) * var(y);
    rsq = 1 - SSresid/SStotal;
    eqn = sprintf('R² = %.2f', rsq);

    % Update the text of the label with the equation and R-squared value
    app.Label_2.Text = eqn;

end
```


Le code de vidage du tableau

```
% Button pushed function: viderletableauButton  
function viderletableauButtonPushed(app, event)  
    app.UITable.Data = {};  
    app.EditField_2.Value = ' ';  
end
```

Le code du calcul et d'affichage de temps de fonctionnement du cycle

```
% Button pushed function: CalculerletempspourButton
function CalculerletempspourButtonPushed(app, event)

% Get x and y data from UIAxes
x = app.UIAxes.Children.XData;
y = app.UIAxes.Children.YData;

% Fit a linear polynomial to the data
p = polyfit(x, y, 1);

% Equation to solve for x
eqn = sprintf('y = %.2fx + %.2f', p(1), p(2));

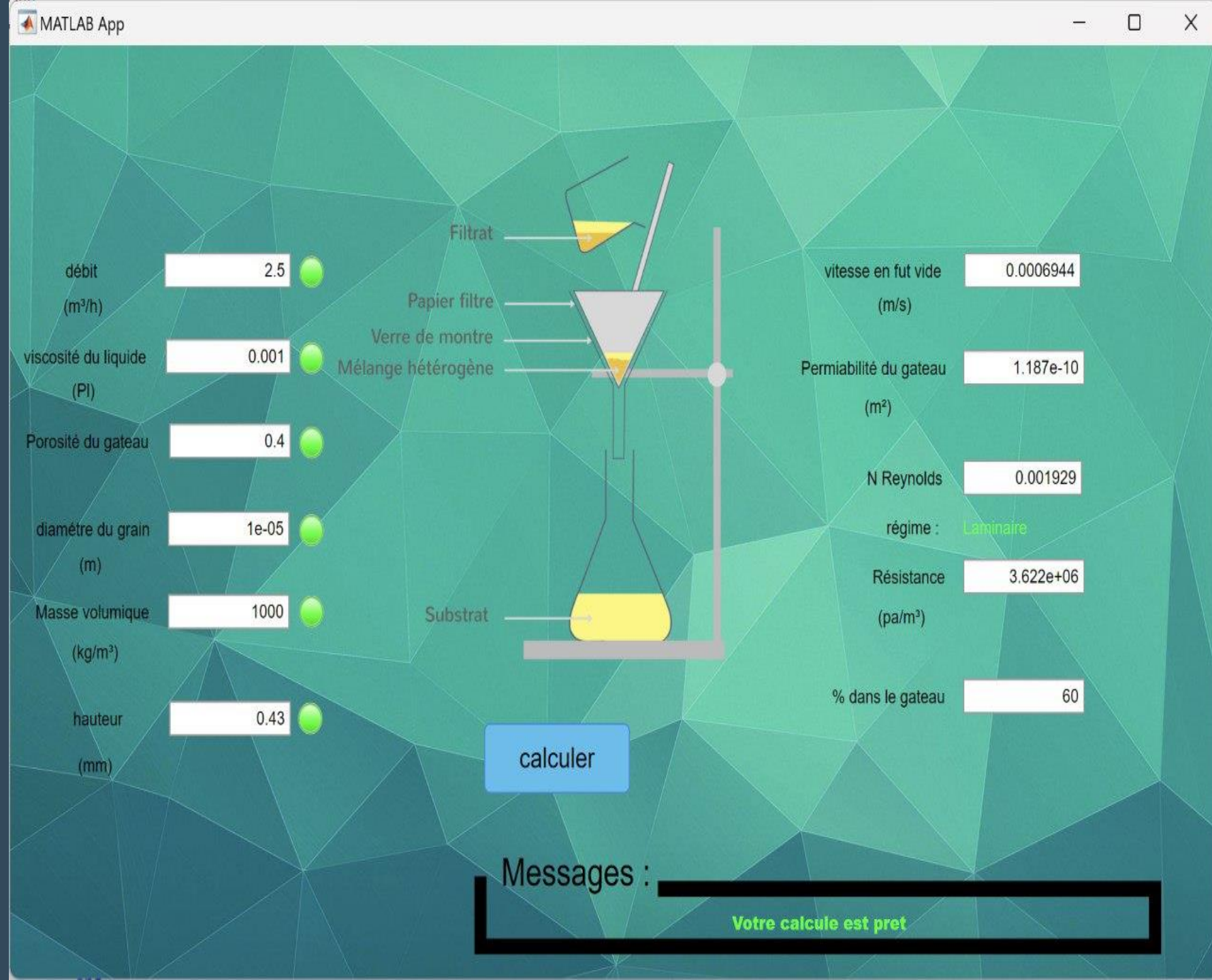
% Add an edit field and label to the app UI to input and display the value of y

% Add a button and modify its callback to execute the following code
y = app.ZoudPEditField.Value;
x = (y - p(2)) / p(1);
app.Label_4.Text = sprintf('t = %.2f', x);

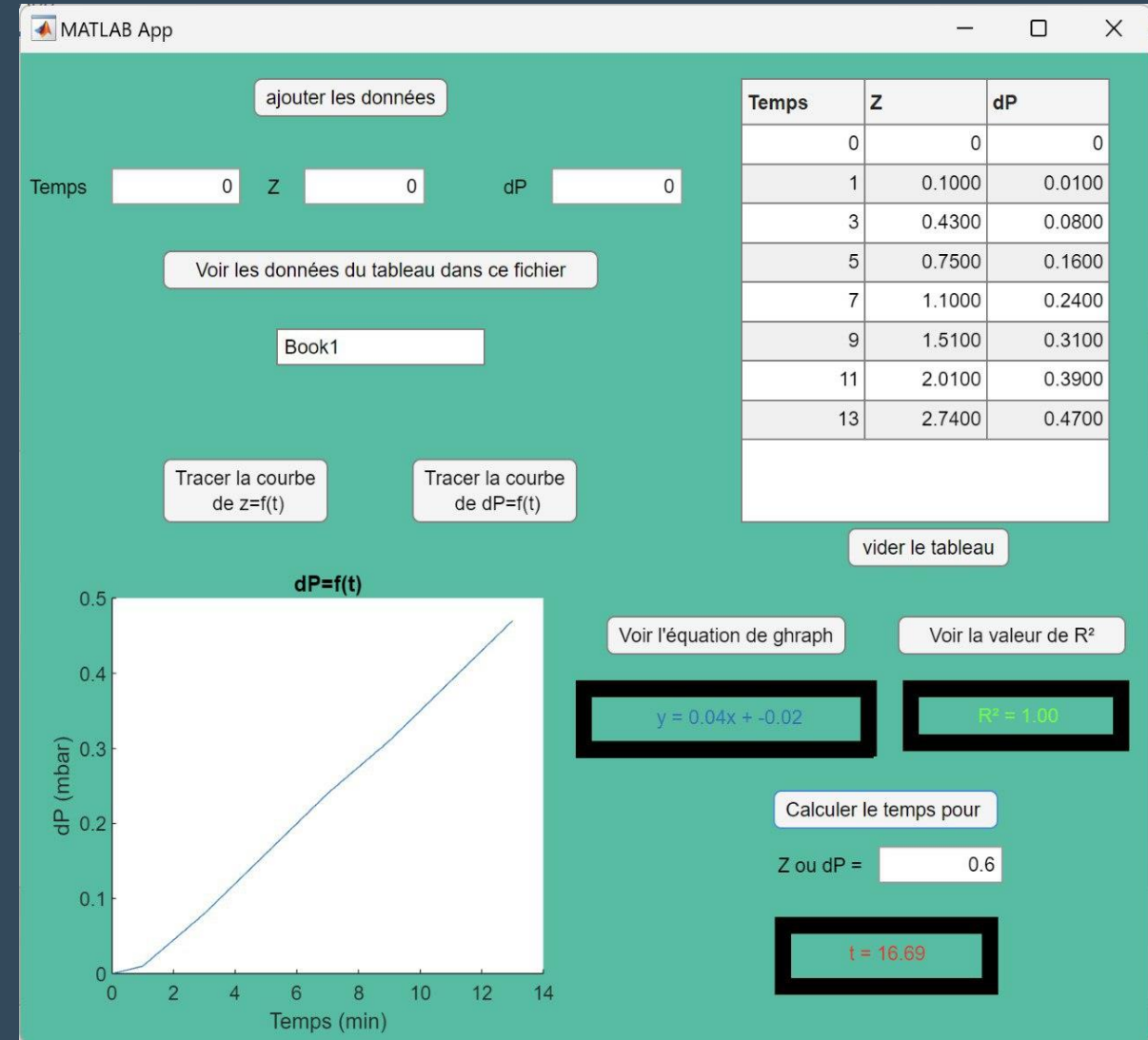
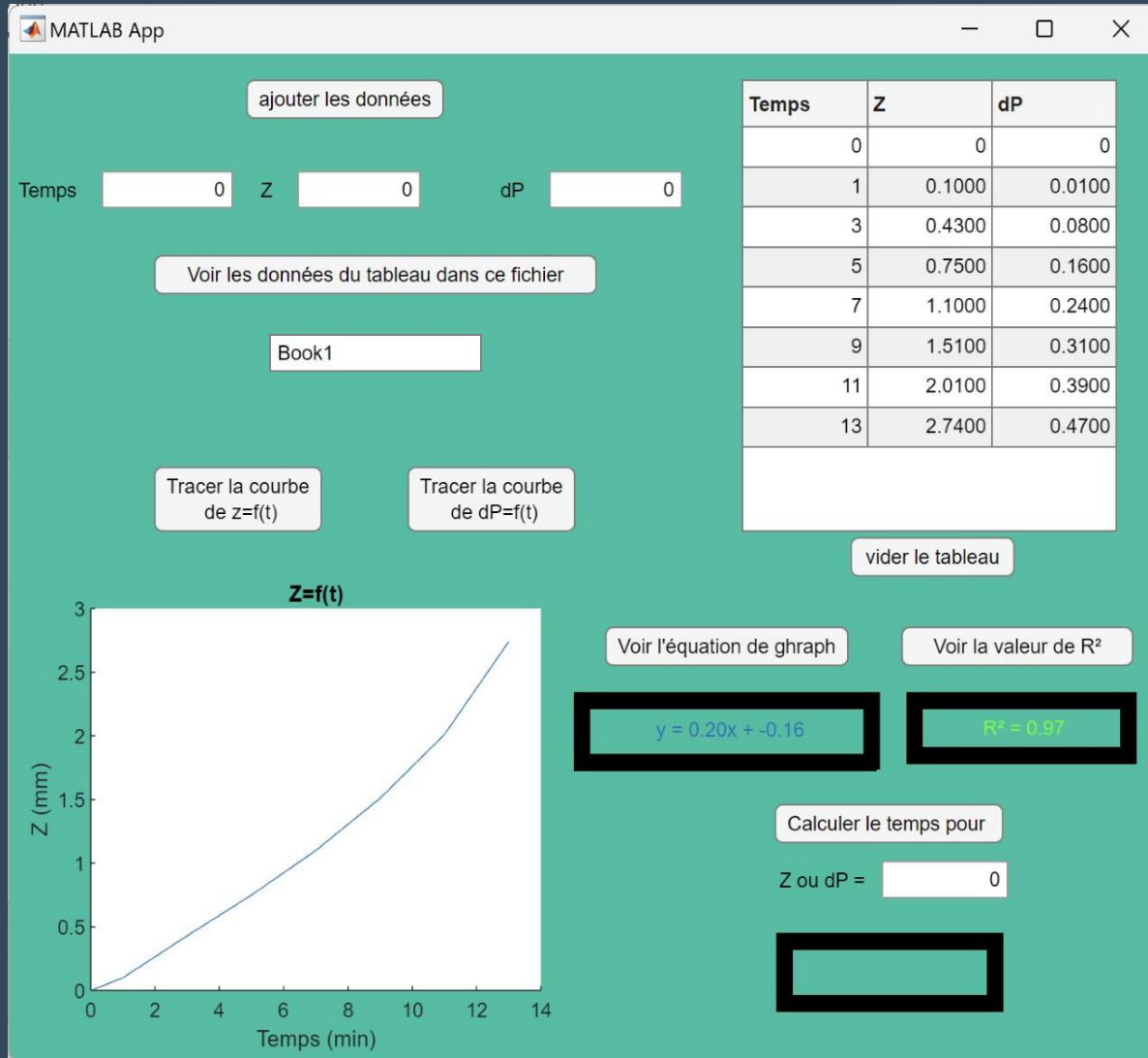
end

end
```

Résultats des question 1- 2- 3- 4- 5



Résultat des question 5-6-7



Voir le vidéo de fonctionnement pour plus
d'information