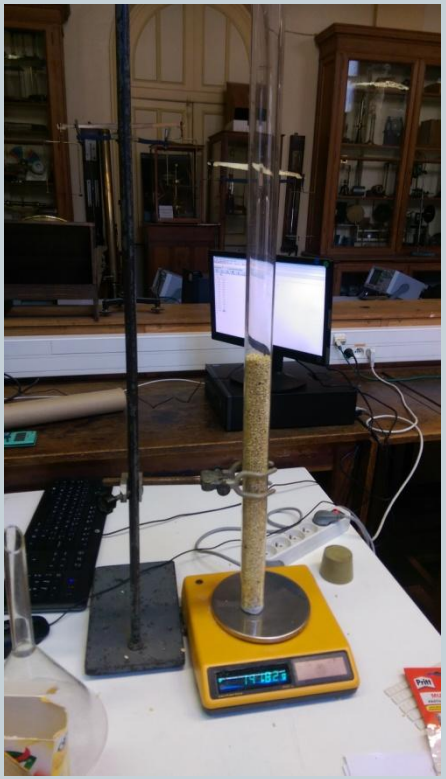
Etude des contraintes physiques dans un silo pour une construction optimale



RÉALISÉ PAR:

BLASSIAU ADRIEN

Introduction

Présentation :

Matériau granulaire

Stockage

Vidange

Silo-tour

Objectif :

Programme qui élabore l'architecture de silos selon :

- Cahier des charges imposé par un industriel
- Contraintes physiques liées au stockage et à la vidange

Le silo ne doit pas s'effondrer (nécessité)

Elle doit être constante (commodité)

Introduction

Problématique :

Comment caractériser un silo techniquement viable à construire et le modéliser informatiquement ?

Plan :

I/Etudes des contraintes physiques et élaboration du programme

II/Présentation et analyse des résultats expérimentaux

III/Application du programme sur un exemple et limites

I/1-Le stockage des grains

- Comment traduire physiquement la condition :

« le silo ne doit pas s'effondrer lors du stockage» ?

→Contrainte maximale exercée par les grains sur les parois < Pression de résistance des parois

= Confronter le modèle de Janssen à la Formule de Lamé

I/1-Le stockage des grains-Modèle de Janssen

- Distribution des contraintes horizontales donnée par le modèle de Janssen :

$$\sigma_x(z) = K\lambda\rho g(1-\exp(-\frac{H-z}{\lambda}))$$

avec:

$K = 0.45$: coefficien t de Janssen

$\lambda = \frac{D}{4\mu_s K}$: hauteur caractéristique (m)

ρ : masse volumique du milieu granulaire ($kg.m^{-3}$)

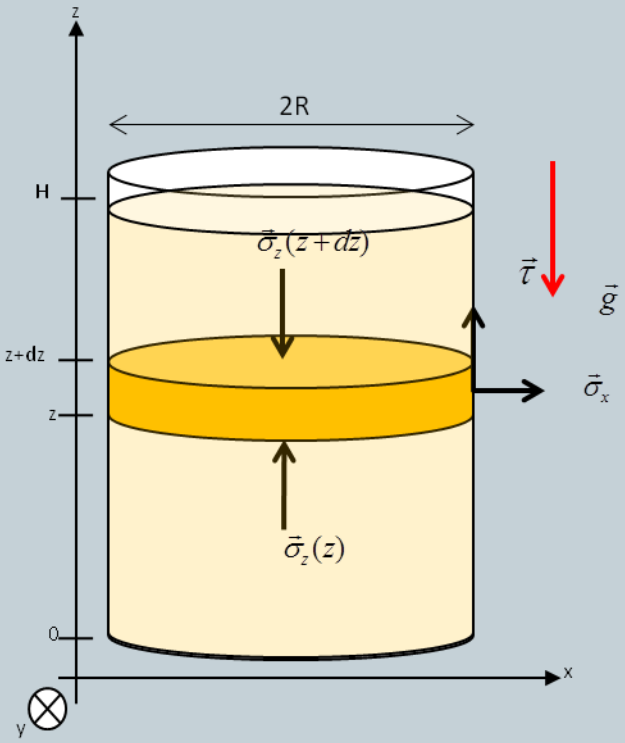
g : accélération de la pesanteur ($m.s^{-2}$)

H : hauteur de grains (m)

- Contrainte horizontale maximale de saturation exercée par les grains sur les parois : $\sigma_{x,sat} = K\lambda\rho g$

I/1-Le stockage des grains-Modèle de Janssen

- Modèle choisi :



Equilibre sur une tranche d'épaisseur infinitésimale dz, de surface $S = \pi R^2$ et de surface latérale $dS_l = \pi D dz$ prise dans un silo cylindrique de rayon R rempli entièrement à une hauteur H de grains de masse volumique ρ.

$$\sigma_x = K\sigma_z$$

(Effet de voûtes)

$$\tau S = \mu_s \sigma_x dS_l$$

(Limite équilibre statique)

σ_x : contrainte horizontale (Pa)

σ_z : contrainte verticale (Pa)

$K = 0.45$: coefficient de Janssen

σ_x : contrainte horizontale (Pa)

τ : contrainte tangentielle (Pa)

μ_s : coefficient de frottement

I/1-Le stockage des grains-Modèle de Janssen

- Démonstration :

Bilan des forces associées aux contraintes selon $\vec{e}_z$:

$$d\vec{P} = -\rho g S dz \vec{e}_z$$

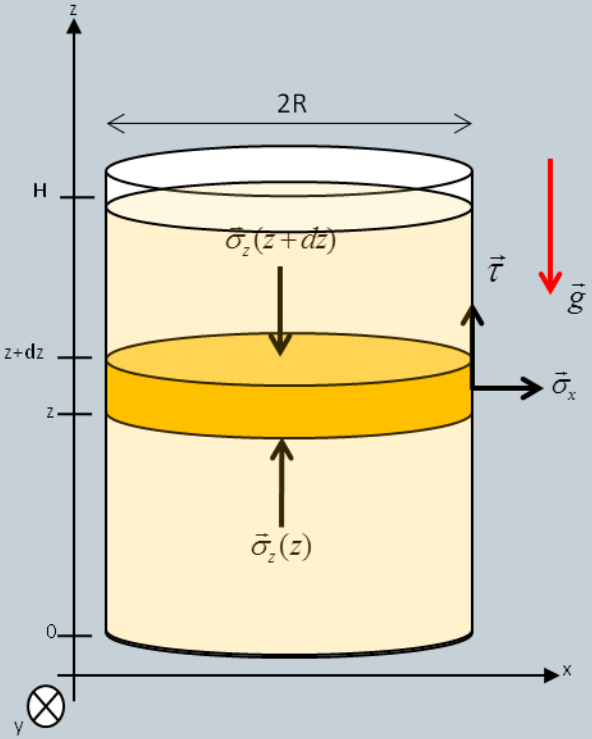
$$\vec{F}_z(z+dz) = S\sigma_z(z+dz)\vec{e}_z$$

$$\vec{F}_z(z) = -S\sigma_z(z)\vec{e}_z$$

$$\vec{T} = -\pi D \tau dz \vec{e}_z$$

Principe fondamental de la dynamique :

$$T + F(z) - F(z+dz) - dP = 0$$



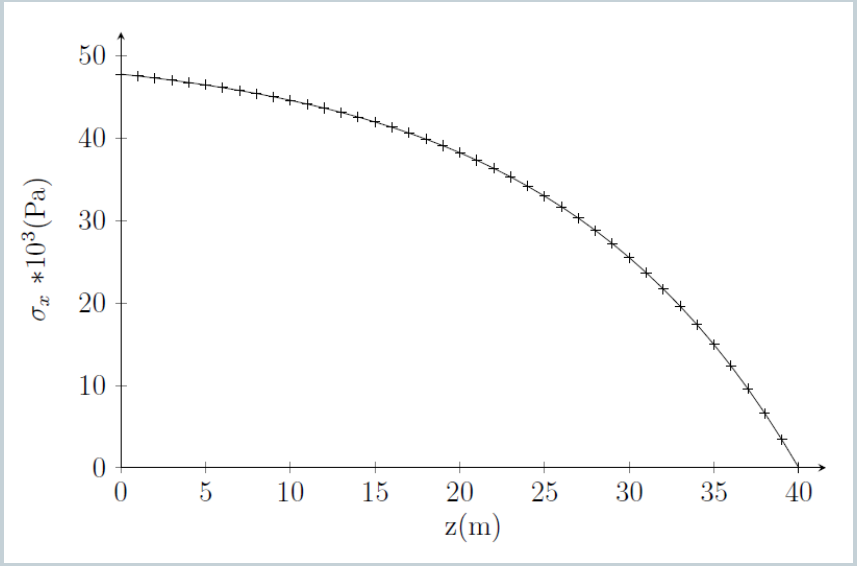
I/1-Le stockage des grains-Modèle de Janssen

Après simplifica tion :

$$-\frac{D}{4\mu_s K}\frac{d\sigma_z}{dz}+\sigma_z=-\frac{D\rho g}{4\mu_s K}$$

La solution de la forme :

$$\sigma_z(z)=\frac{-D\rho g}{4\mu_s K}+A\exp(\frac{z}{\lambda})$$



On considère la condition aux limites suivantes :

$$\sigma_z(H)=0$$

On obtient :

$$\sigma_x(z)=K\sigma_z(z)=K\lambda\rho g(1-\exp(-\frac{H-z}{\lambda})) \quad \text{et } \sigma_{x,sat}=K\lambda\rho g$$

I/1-Le stockage des grains-Formule de Lamé

- Pression de résistance des parois à une contrainte intérieure donnée par la Formule de Lamé:

$$P_{res} = \frac{\sigma_{rupt} e}{R_{moy} a} \quad \text{si} \quad \frac{R_{moy}}{10} > e \quad \text{(vérifié pour les silos)}$$

avec:

σ_{rupt} : contrainte d'élasticité en traction de la paroi (Pa)

e : épaisseur de la paroi (m)

R_{moy} : rayon moyen de la paroi (m)

a = 1.5 : coefficent de sécurité

- Condition de non effondrement :

$$P_{res} > \sigma_{x,sat}$$

I/1-Le stockage des grains-Expérience

- 2 objectifs :

-Vérifier l'existence de la contrainte de saturation

-Vérifier la forte influence de différents paramètres sur cette contrainte de saturation

→La masse soutenue par le fond du silo doit saturer

$$M_{sout} = M_{sat}(1 - \exp(\frac{M_{vers}}{M_{sat}}))$$

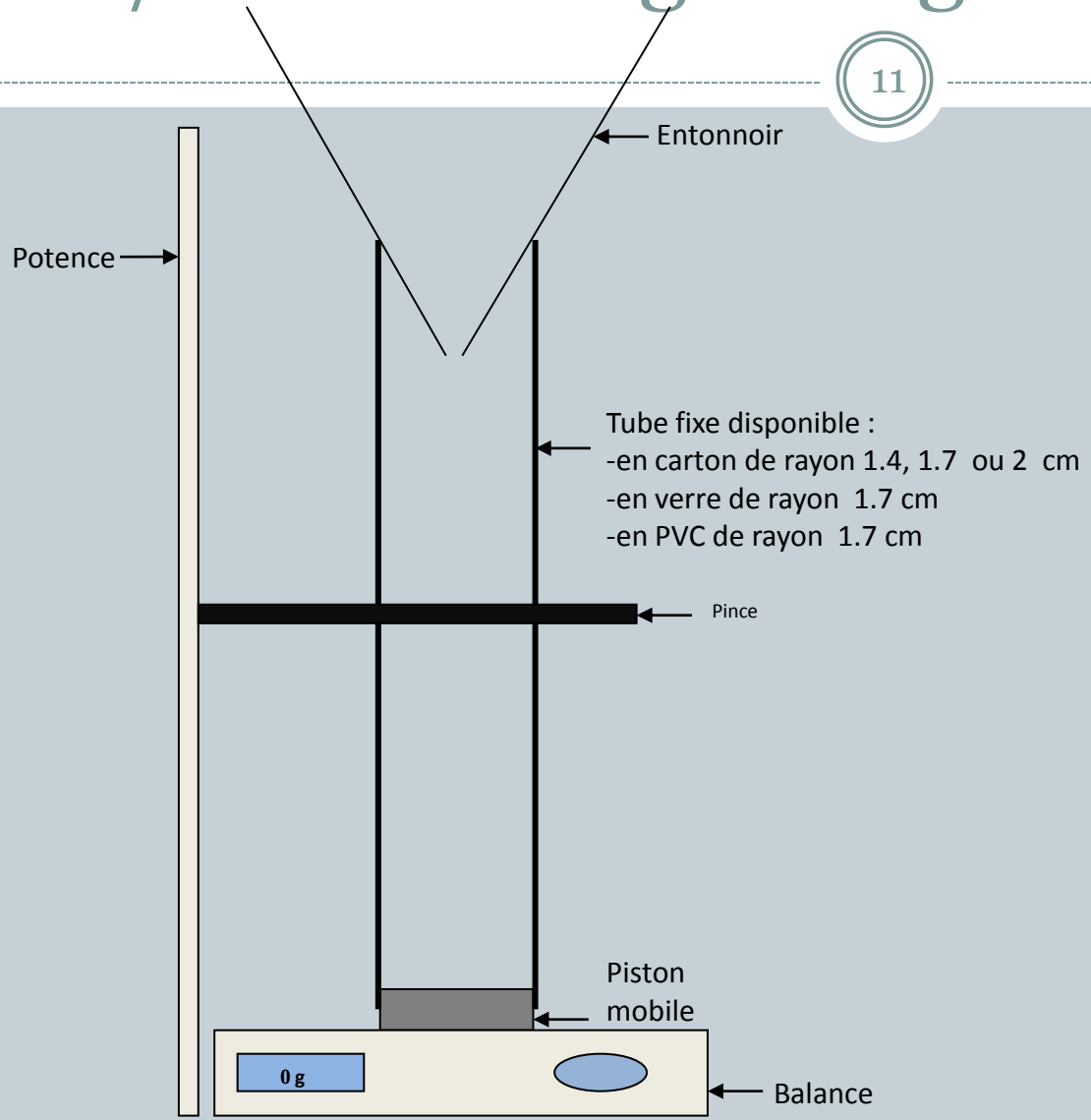
avec:

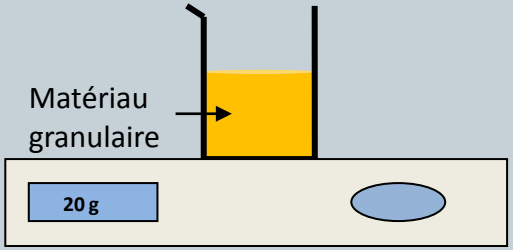
M_{sout} : Masse de grains mesurée au fond du silo

M_{sat} : Masse de grains maximale mesurée au fond du silo

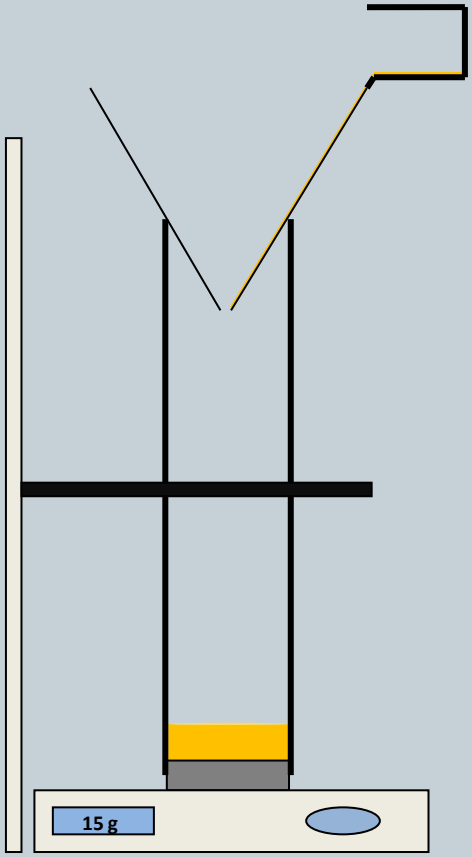
M_{vers} : Masse de grains versée dans le silo

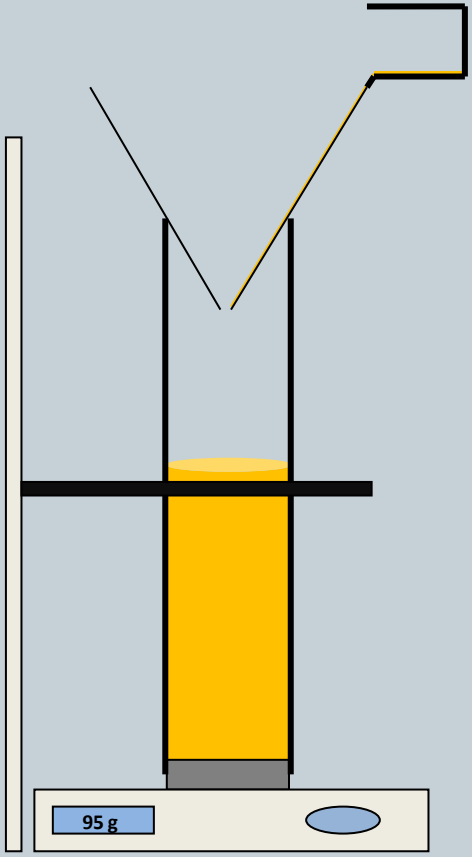
I/1-Le stockage des grains-Expérience

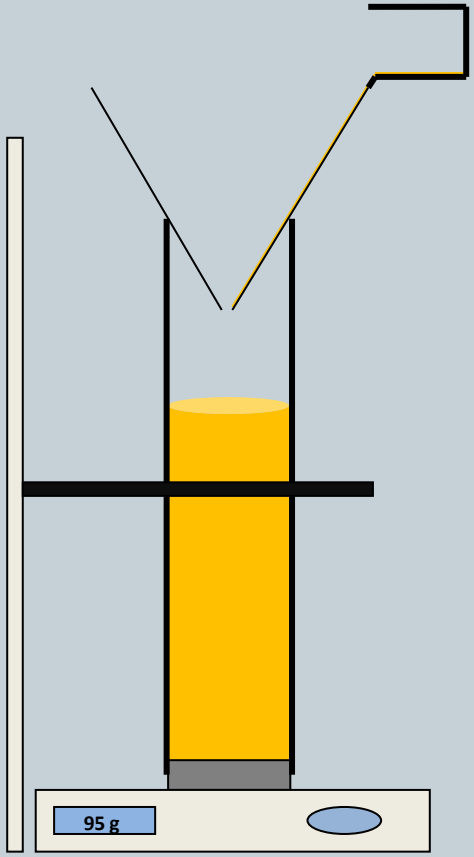




I/1-Le stockage des grains-Expérience







I/2-La vidange des grains

- Comment traduire physiquement la condition :

« le débit de vidange des grains doit être constant» ?

→Etude de la Loi de Beverloo et de ses limites

I/2-La vidange des grains-Loi de Beverloo

- Ecoulement constant des matériaux granulaires donné par la Loi de Beverloo :

$$Q_m = C\rho\sqrt{g}(d - kd_g)^{\frac{5}{2}} \quad \text{si } D_{int} > 2.5d \text{ et } D_{int} > d + 30d_g$$

avec:

C : compacité des grains

ρ : masse volulmique (kg.m^{-3})

g : accélération de la pesanteur (m.s^{-2})

d : diamètre de l'ouverture (m)

$k > 1.5$: coefficien t

d_g : diamètre du grain (m)

D_{int} : diamètre intérieur du silo (m)

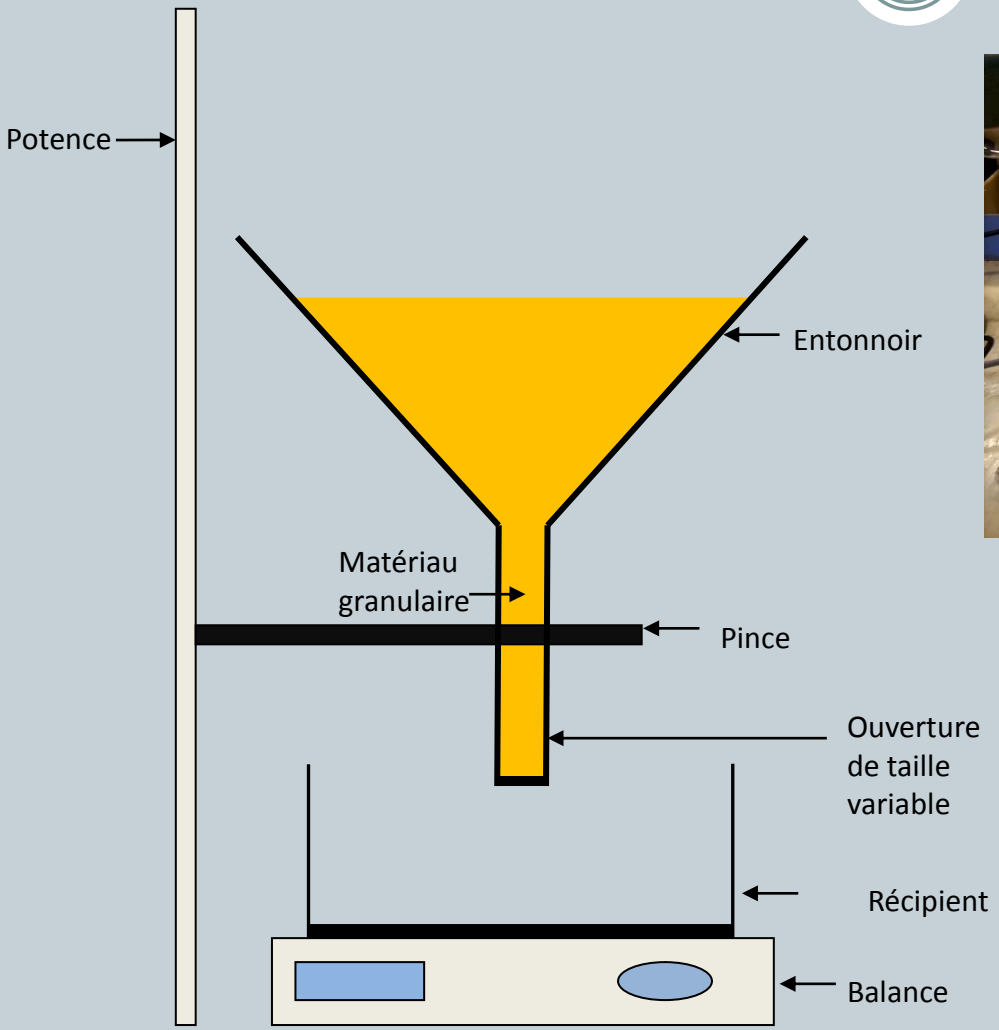
I/2-La vidange des grains-Expérience

- 2 objectifs :

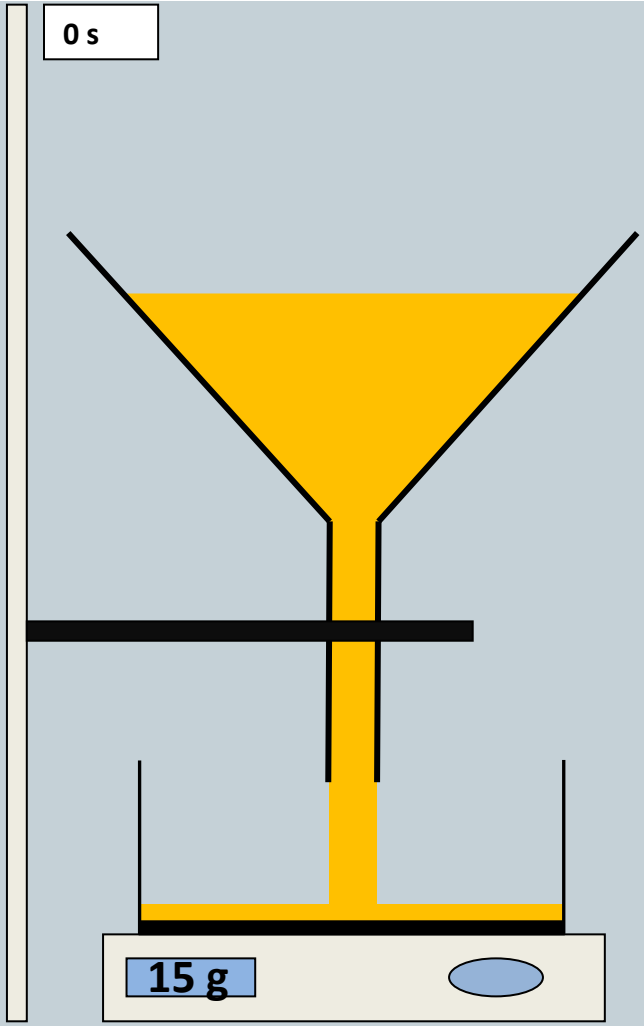
-Vérifier que le débit de vidange d'un matériau granulaire est bien constant dans le cas général

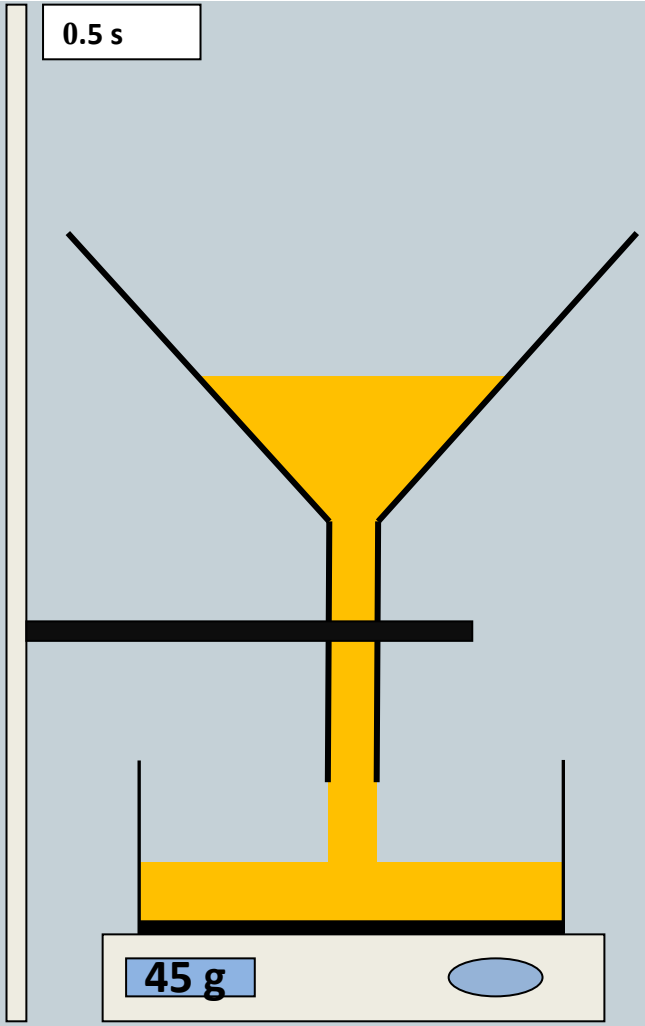
-Vérifier l'impact du rayon du grain sur le débit d'écoulement

I/2-La vidange des grains-Expérience



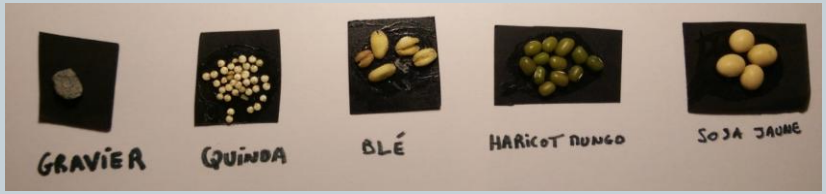
I/2-La vidange des grains-Expérience

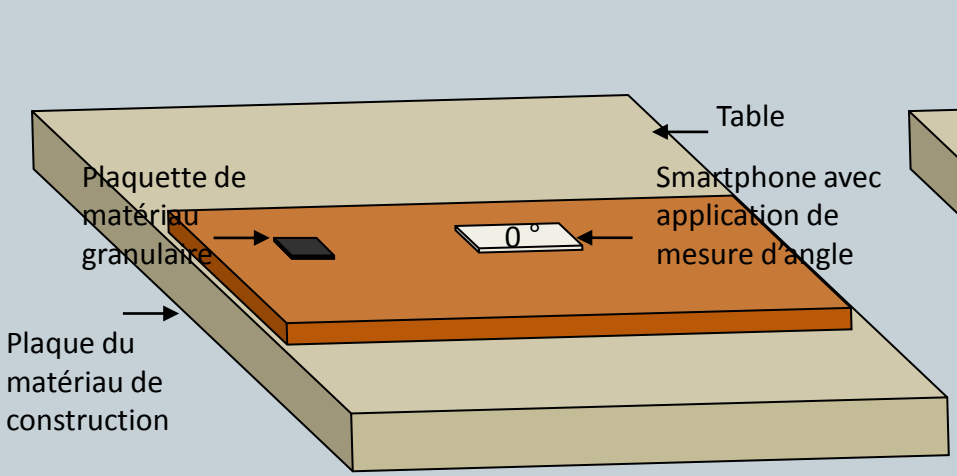


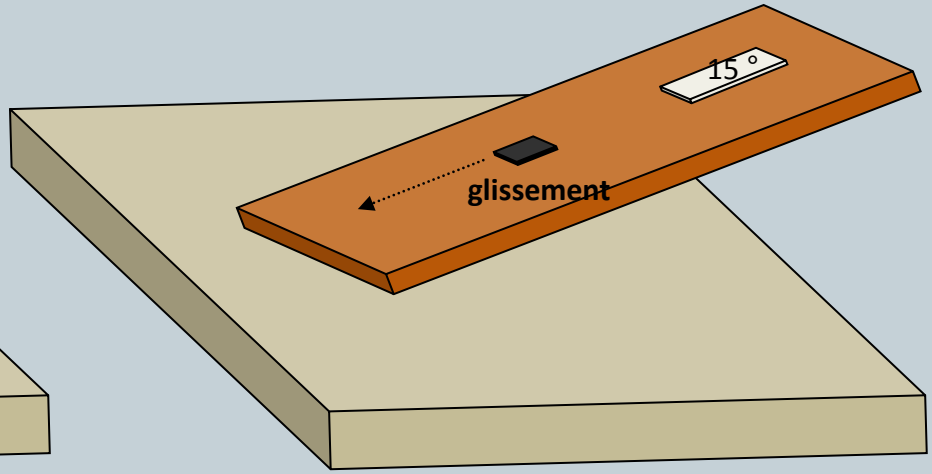


I/3-La mesure de paramètres supplémentaires

- Mesure de coefficients de frottement statique

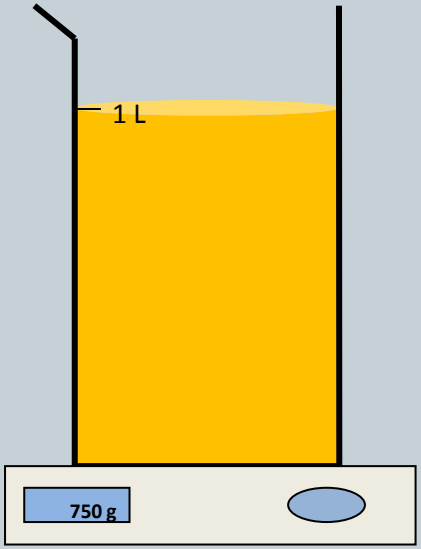


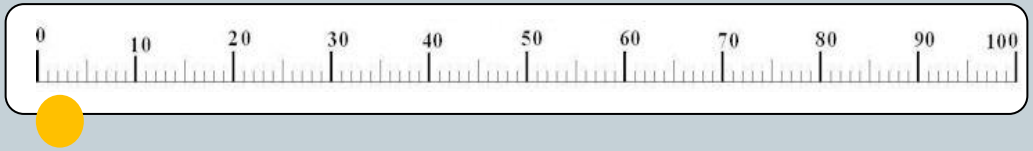




I/3-La mesure de paramètres supplémentaires

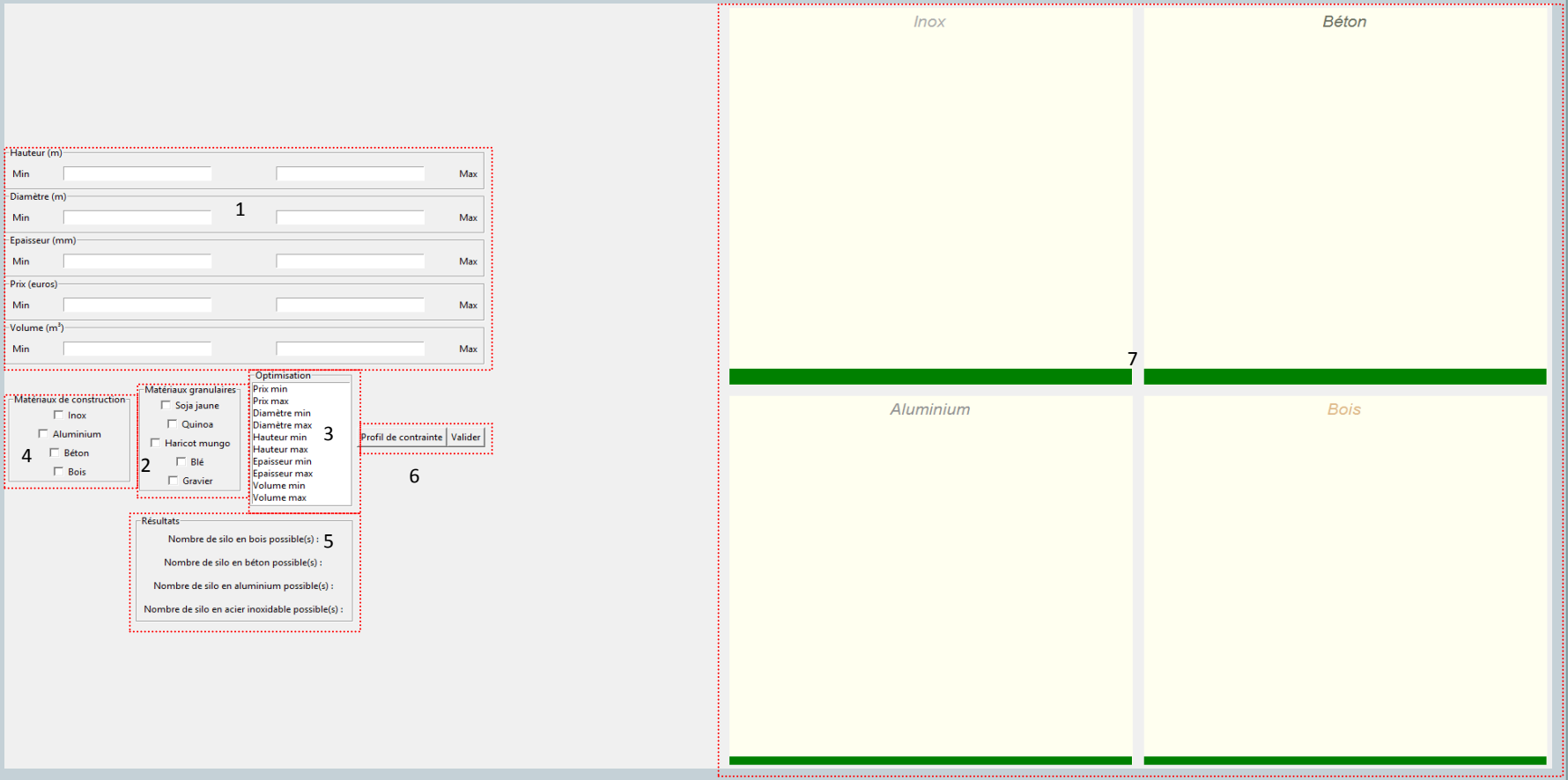
- Mesure de la masse volumique et du diamètre





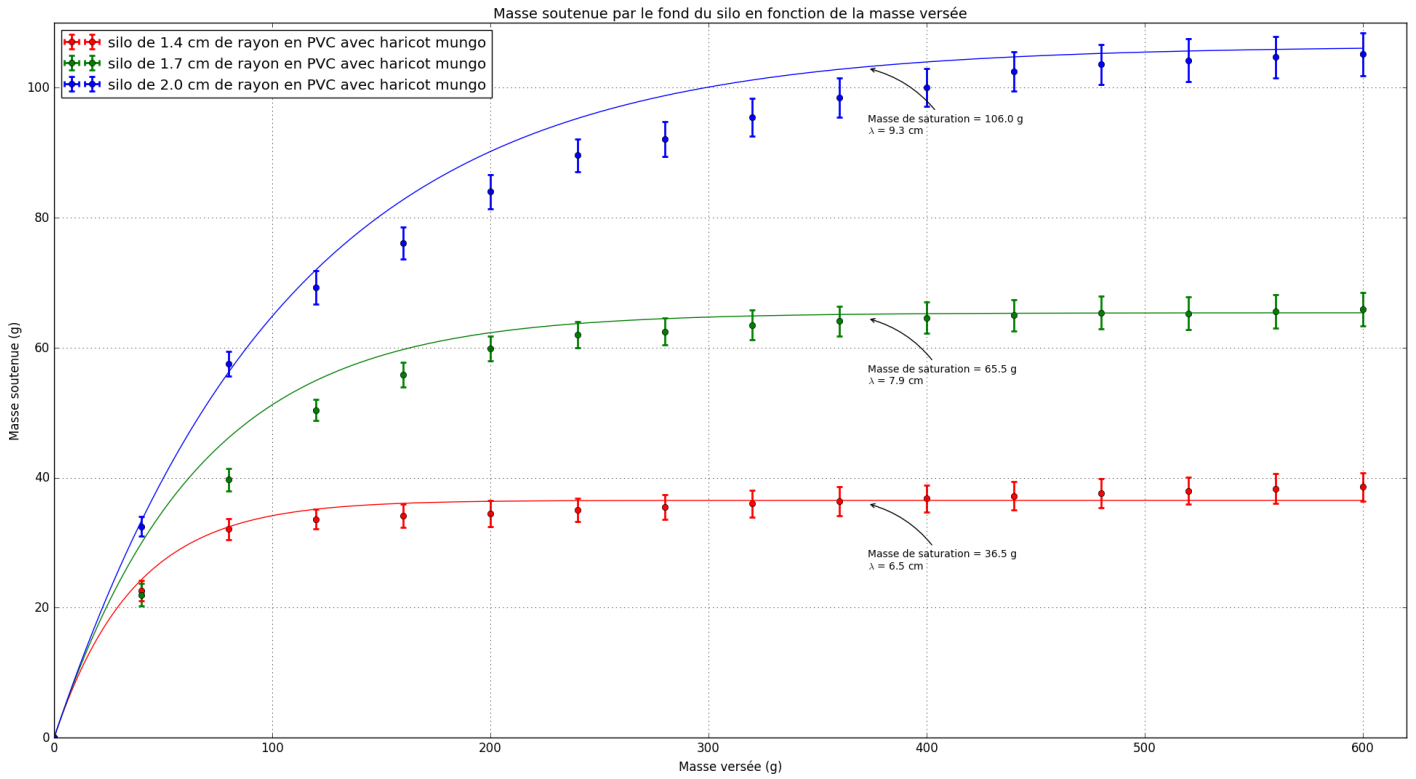
I/4-Présentation du programme

- Interface du programme



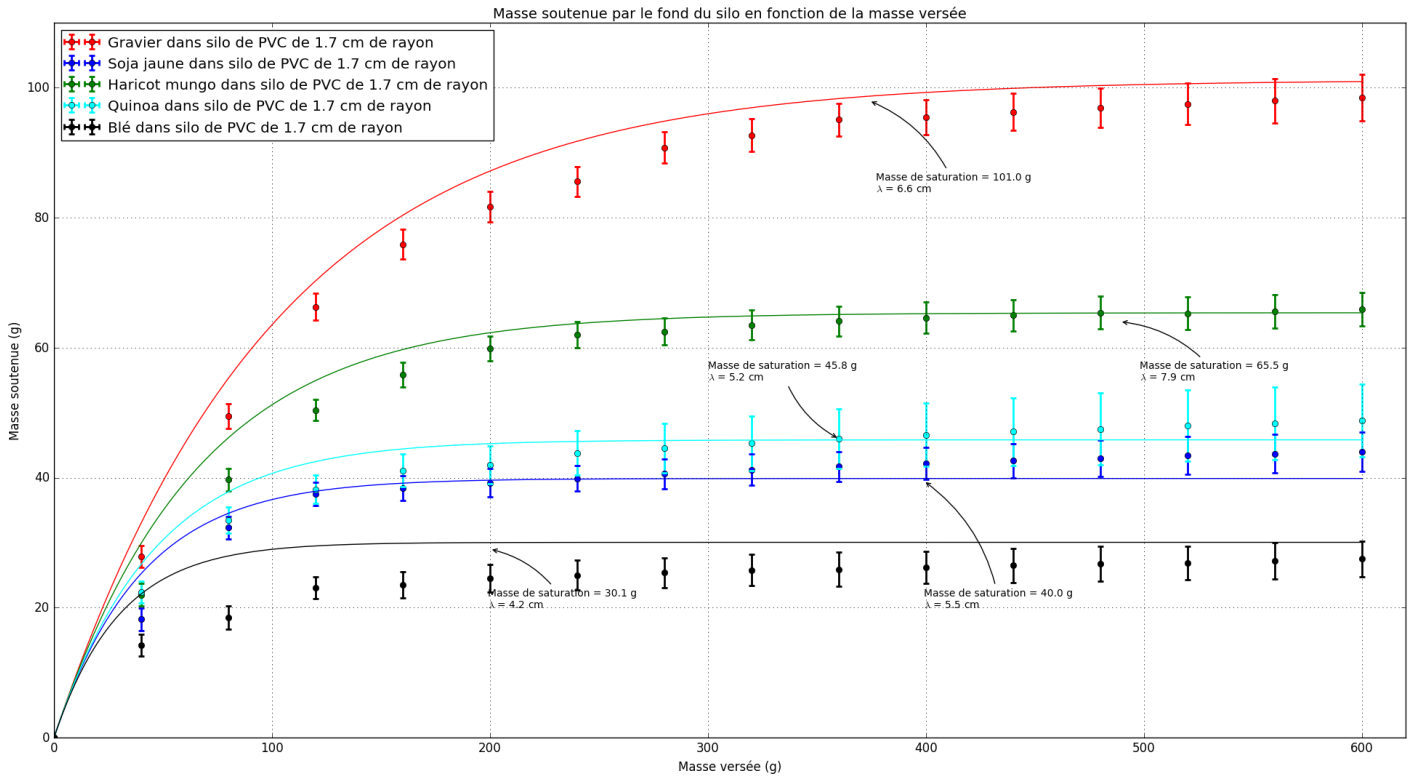
II/1-Le modèle de Janssen

- En fonction du rayon du tube



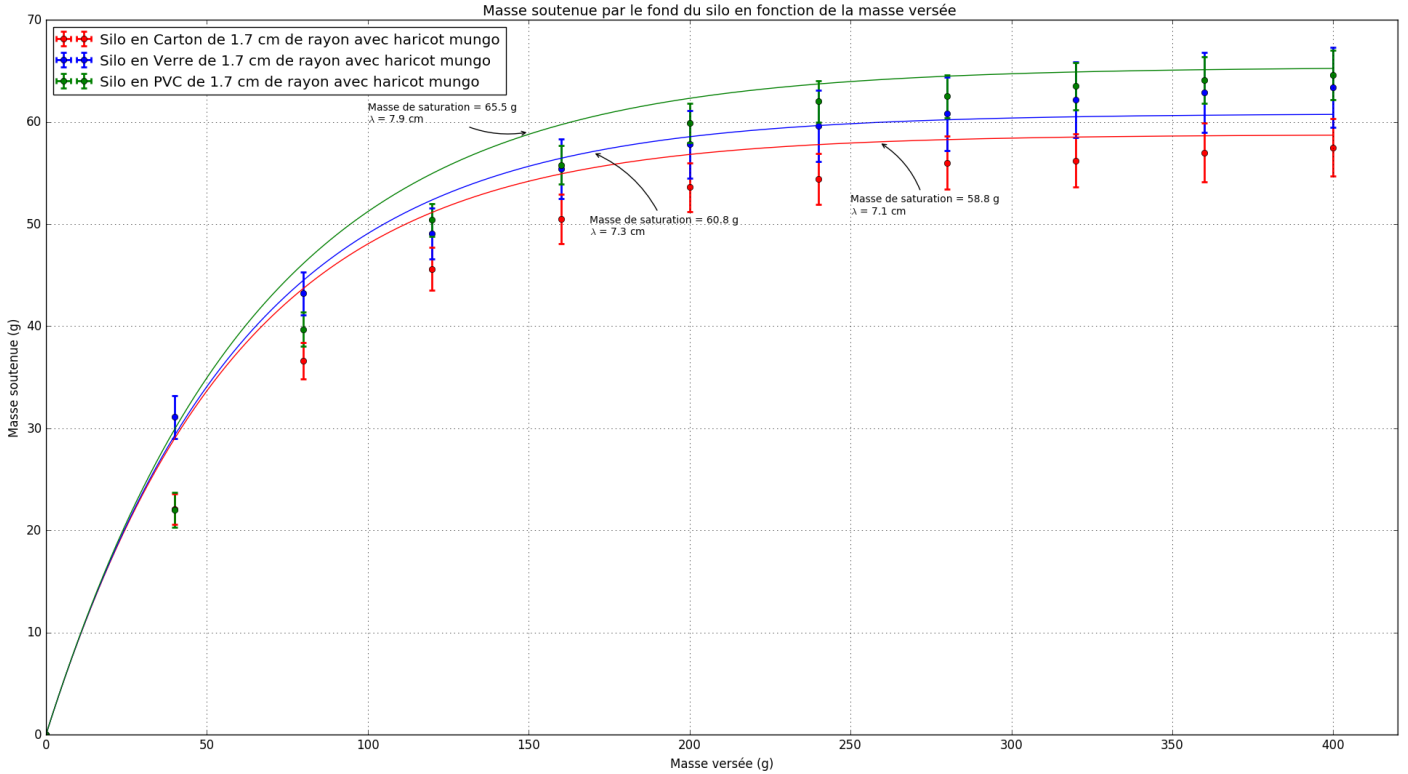
II/1-Le modèle de Janssen

- En fonction du matériau granulaire contenu

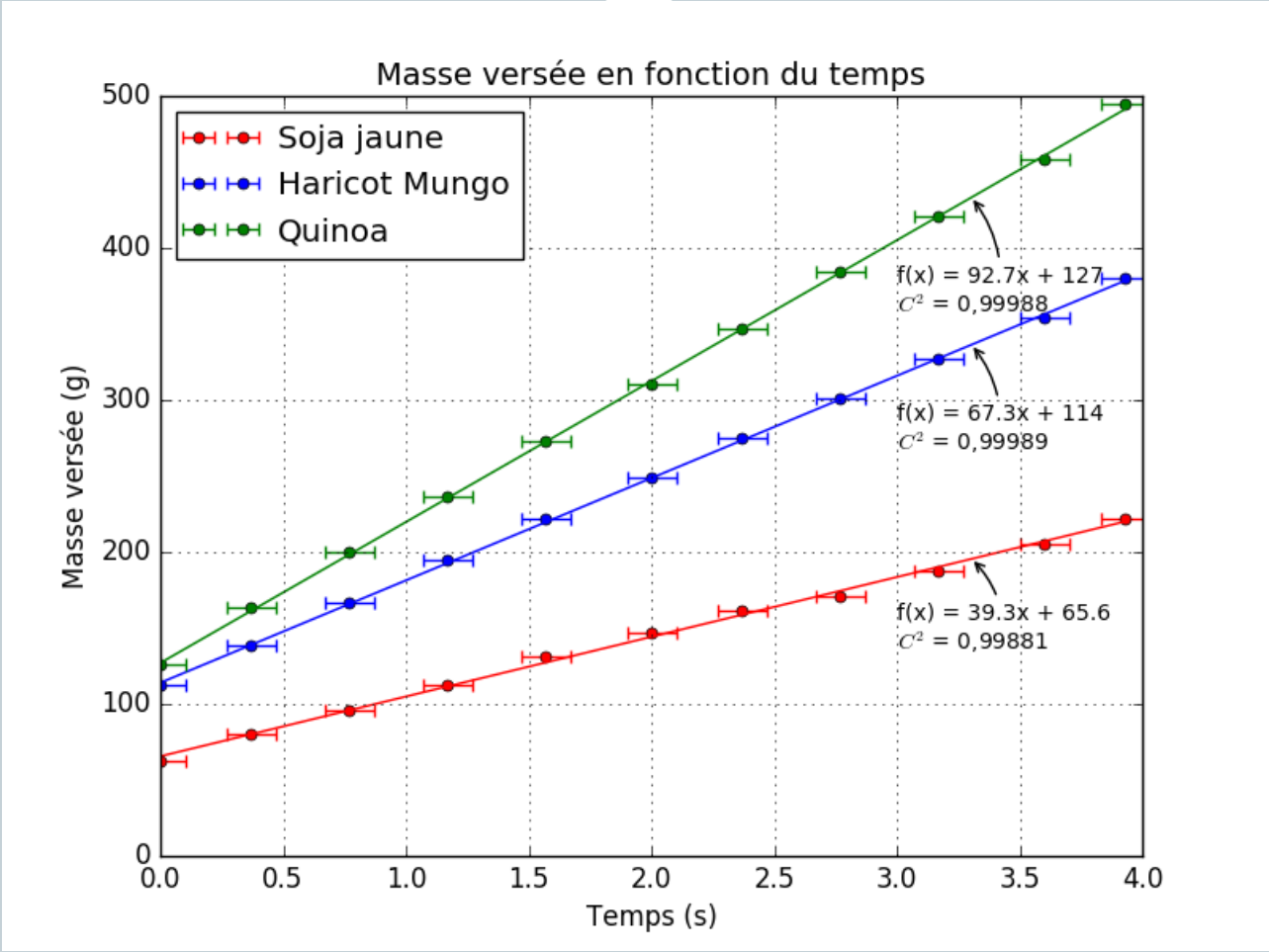


II/1-Le modèle de Janssen

- En fonction du matériau du tube



II/2-La Loi de Beverloo



II/3-Les différents paramètres

- Résultats :

Coefficient de frottement	Gravier	Soja jaune	Haricot mungo	Quinoa	Blé
Acier inoxydable	0.27	0.30	0.29	0.44	0.37
Aluminium	0.32	0.23	0.20	0.35	0.34
Bois	0.55	0.24	0.23	0.29	0.35
Béton	0.39	0.30	0.25	0.49	0.53
Carton	0.58	0.30	0.27	0.33	0.35
Verre	0.25	0.29	0.26	0.39	0.33
PVC	0.29	0.34	0.24	0.36	0.45
Incertitude élargie moyenne de type A	0.02	0.01	0.01	0.03	0.01
Diamètre (mm) *	8 ± 1	8 ± 1	5 ± 1	2 ± 1	8 ± 1
Masse volumique ($kg.m^{-3}$) *	1700 ± 170	800 ± 80	915 ± 92	970 ± 98	780 ± 78

II/3-Les différents paramètres

- Incertitudes de type B * :

$$u_{vol} = \frac{pas\ (\mathrm{mL})}{2} = \frac{100}{2} = 50\ \mathrm{mL}$$

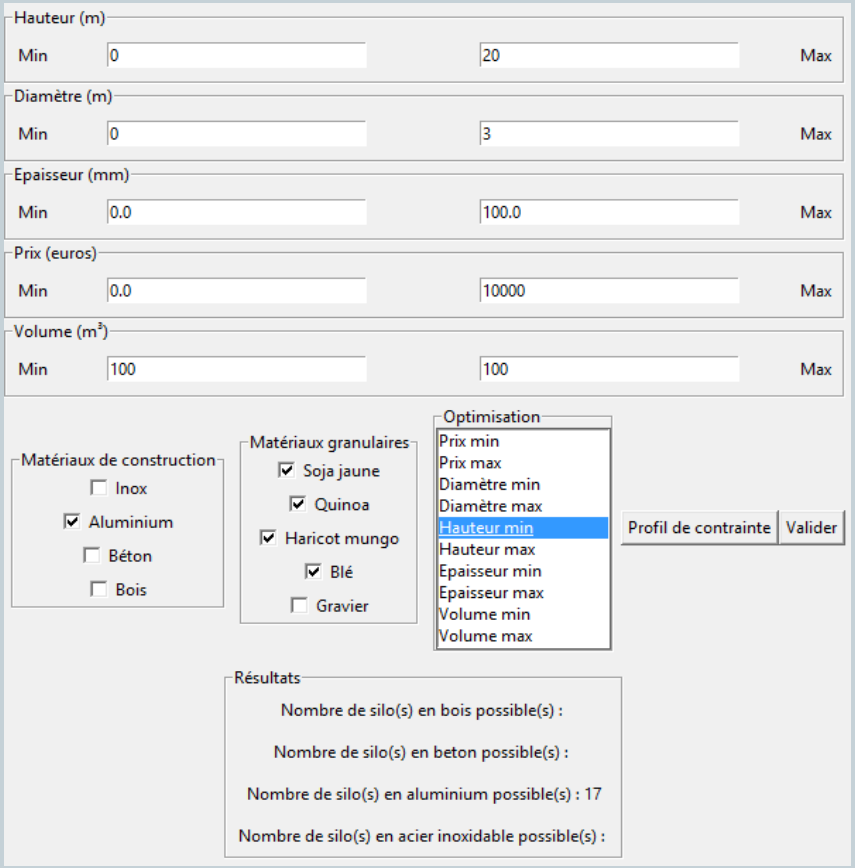
$$u_{masse} = \frac{résolution\ (\mathrm{g})}{2\sqrt{3}} = \frac{0.05}{2\sqrt{3}} = 0.014\ \mathrm{g}$$

$$u_{diam} = \frac{pas\ (\mathrm{mm})}{2} = \frac{1}{2} = 0.5\ \mathrm{mm}$$

$$u_\rho = \rho\sqrt{(\frac{u_m}{m})^2 + (\frac{u_{V_{occupé}}}{V_{occupé}})^2}$$

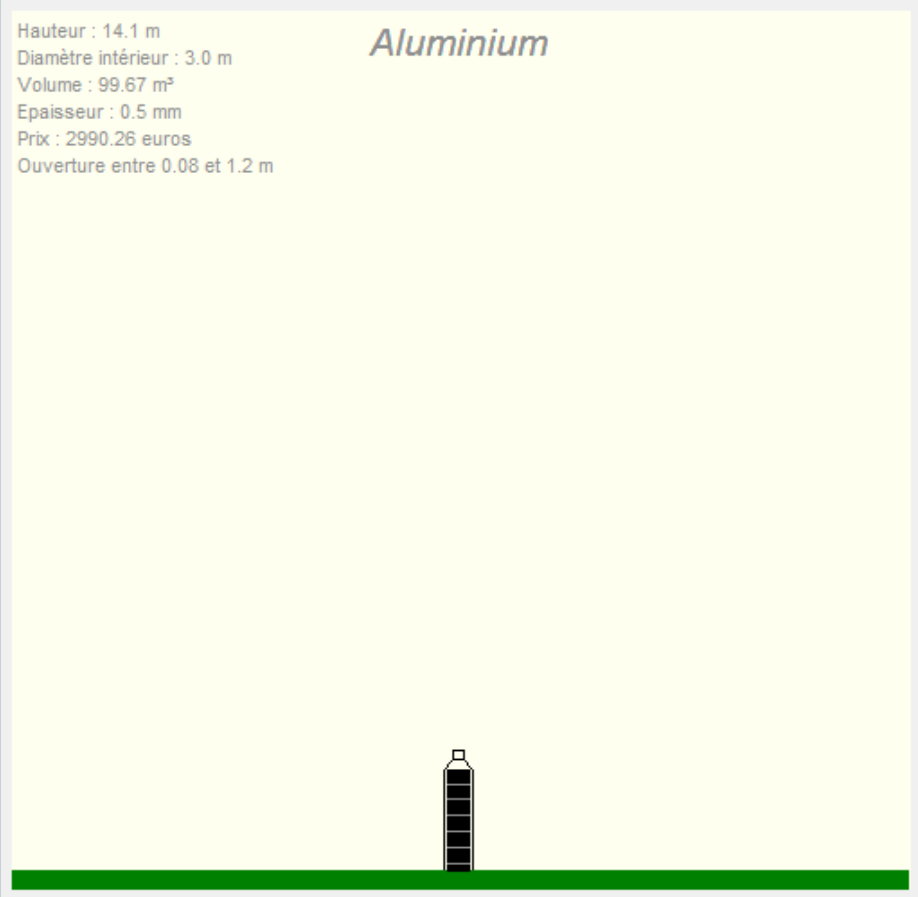
III/Application du programme sur un exemple

- Entrée



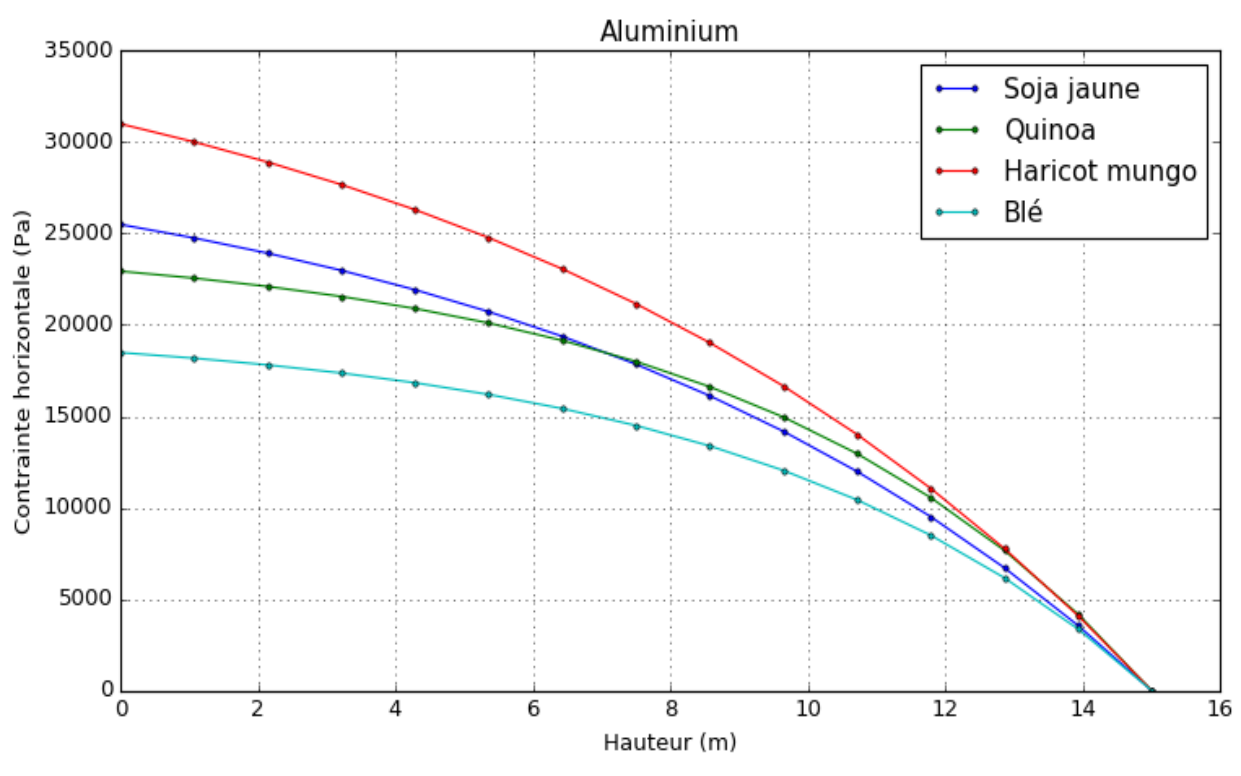
III/Application du programme sur un exemple

- Sortie « valider »



III/Application du programme sur un exemple

- Sortie « profil de contrainte »



III/Application par modélisation informatique

- Limites

→Remplissage non pris en compte

→D'autres contraintes comme la température, l'humidité, les explosions de poussières, etc

→D'autres formes de silo

Annexe : Programme


```
import numpy as np
import matplotlib.pyplot as plt
from math import *
from tkinter import *
from threading import Thread
from time import*
import os


## Partie construction des profils de pression

def profildepression(z,silopti,mat,gr):
    '''Fonction qui recupère les contraintes horizontales subies par les
parois du silo à différentes hauteurs'''
    y=[]
    for i in z:
        y.append(pp(i,silopti,mat,gr))
    return y

def pp(z,silopti,mat,gr):
    '''Fonction qui calcule la contrainte horizontale subie par les parois
du silo à une altitude donnée'''
    h=silopti[0]
    d=silopti[1]
    mu=tabfrot[mat][gr]
    k=0.45
    g=9.80
    rho=tabgrain[gr][2]

    l=d/(4*mu*k)

    y=l*rho*g*k*(1-exp(-(h-z)/l)) # Expression de la contrainte horizontale
subie par le silo à une altitude z pour un remplissage de hauteur h
    print (l)
    return y

def pression():
    '''Fonction qui réunit les profils de pression des différents silos'''
    global tabsilopti

    fig = plt.figure(figsize=(20,12))
    fig.patch.set_facecolor("white")

    if tabsilopti[0]!=[]:
        plt.subplot(2,2,1)  # Division en sous-fenêtres
        tracer("Inox",tabsilopti[0],0) # Appel de la fontion de traçage

    if tabsilopti[1]!=[]:
        plt.subplot(2,2,3)
        tracer("Aluminium",tabsilopti[1],1)

    if tabsilopti[2]!=[]:
        plt.subplot(2,2,2)
        tracer("Béton",tabsilopti[2],2)

    if tabsilopti[3]!=[]:
        plt.subplot(2,2,4)
        tracer("Bois",tabsilopti[3],3)

    plt.show()
```



```
def tracer(type,silopti,mat):
    '''Fonction qui trace le profil de pression de chaque silo'''
    global grain

    h=silopti[0]
    z = list(np.linspace(0, h, h))
    plt.title(type)
    for j in range(0,len(grain)):
        if grain[j]==1:
            y=profildepression(z,silopti,mat,j)
            plt.plot(z,y,marker='o',ms=3,label=tabgrain[j][0])
            plt.ylabel('Contrainte horizontale (Pa)')
            plt.xlabel("Hauteur (m)")
            plt.grid(True)
            plt.legend()


## Partie construction et optimisation de silo

tabgrain=[["Soja jaune",8.0,800.0],["Quinoa",2.0,970.0],["Haricot
mungo",5.0,915.0],["Blé",8.0,780.0]
,["Gravier",8.0,1700.0]]  # Tableaux des caractéristiques des grains[nom du
grain, diamètre, masse volumique]
tabmat=[["Inox",800.0,230.0,False,120.0,"#A9A9A9" ],
["Aluminium",300.0,50.0,False,45.0,"#8F8F8F"]
,["Béton",3.0,1.8,False,200.0,"#686C5E"],
["Bois",40.0,0.15,True,600.0,"#DEB887"]]
# Tableaux des caractéristiques des matériaux[nom du matériau, contrainte de
rupture, conductivité thermique, renouvelable, prix, couleur]
tabfrot=[[0.296,0.435,0.287,0.367,0.271],[0.226,0.345,0.204,0.344,0.321],
[0.300,0.488,0.249,0.532,0.394],[0.242,0.288,0.230,0.354,0.554]]
# Tableau des coefficients de frottement matériau-grain, chaque tableau
correspond à un matériau de construction.


def volume(Hmin,Hmax,Dimin,Dimax,Vmin,Vmax):
    '''Fonction qui débute la construction du silo optimal par la recherche
de ceux respectant les dimensions imposées'''
    V=0
    tabDi=[] # Prend le diamètre du silo crée
    tabH=[] # Prend la hauteur du silo crée
    tabV=[] # Prend le volume du silo crée
    for i in np.arange(Dimin,Dimax+0.1,0.1): # On itère sur la plage de
diamètre définit par l'utilisateur
        for j in np.arange(Hmin,Hmax+0.1,0.1): # On itère sur la plage de
hauteur définit par l'utilisateur
            V=ar((pi*j*i**2)/(4)) # On calcul le volume du silo crée avec le
couple (diamètre, hauteur) proposé
            if Vmin-1<=V<=Vmax+1: # Si le volume appartient à la plage de
volume désiré, on garde le silo
                tabDi.append(ar(i))
                tabH.append(ar(j))
                tabV.append(ar(V))
    return tabH,tabDi,tabV


def g(x,D,tabgrain,grain,tabfrot,resist,mat):
    '''Fonction qui calcule si les murs sont assez résistants à la
contrainte engendrée par les grains'''
```

Annexe : Programme


```
    grav=9.81
    diff=((resist*10**6*(x))/((D**2)*1.5))-
contraintemaxgrain(tabgrain,grain,tabfrot,mat)
    # On fait : pression limite de rupure - contrainte horizontale de
saturation
    return diff

def contraintemaxgrain(tabgrain,grain,tabfrot,mat):
    ''' Fonction qui cherche pour quel matériau granulaire la contrainte est
la plus forte'''
    grav=9.81
    max=0
    for i in range(0,len(tabgrain)):
        if grain[i]==1:    # On itère sur les diffrents grains
            janssen=(tabgrain[i][2]*grav)/(4.0*tabfrot[mat][i])  # On
cherche la contrainte horizontale la plus forte.
            if janssen>=max:
                max=janssen
    return max


def dicho(D,tabgrain,grain,tabfrot,resist,mat,Emin,Emax):
    ''' Fonction qui trouve l'épaisseur à respecter pour que le silo puisse
résister aux grains'''
    droite=Emax
    gauche=Emin

    milieu=(droite+gauche)/2.0
    fd=g(droite,D,tabgrain,grain,tabfrot,resist,mat)
    fg=g(gauche,D,tabgrain,grain,tabfrot,resist,mat)
    fm=g(milieu,D,tabgrain,grain,tabfrot,resist,mat)
    g(0.005,D,tabgrain,grain,tabfrot,resist,mat)
    #print(fd,fg,fm)

    if fm==0:
        return milieu
    if fg==0:
        return gauche
    if fd==0:
        return droite
    if fd<0 and fg<0:
        return -1
    if fd>0 and fg>0:
        return Emin
    while droite-milieu>10**(-5):
        if fm*fg<0:
            droite=milieu
            fd=fm
        else:
            gauche=milieu
            fg=fm
        milieu=(gauche+droite)/2
        fm=g(milieu,D,tabgrain,grain,tabfrot,resist,mat)
    if milieu<0.0005:
        milieu=0.0005
    return milieu  # Renvoit l'épaisseur que les murs du silo doivent
respecter

def contrainte(dimension,mat,tabgrain,grain,tabmat,tabfrot,Emin,Emax):
    '''Fonction qui continue la construction du silo en cherchant son
```



```
épaisseur de paroi'''
    Di=dimension[1]
    grav=9.81
    tabsilo=[]
    for i in range(0,len(Di)):
        e=dicho(Di[i],tabgrain,grain,tabfrot,tabmat[mat][1],mat,Emin,Emax) #
l'épaisseur se trouve par dichotomie
        if Emin<=e<=Emax :    #and ((2.0*e)/(2.0*Di[i]+e))<0.1 peut être
rajouté, traduction de (e/Rmoy)<0.1 (voir Beverloo)
            tabsilo.append([dimension[0][i],dimension[1][i],dimension[2]
[i],e]) # On complète le tableau regroupant l'ensemble des silos possibles
    return tabsilo

def couverture(silo):
    '''Fonction qui calcule l'aire à couvrir ou le volume à couvrir pour un
silo donné'''
    H=silo[0]
    Di=silo[1]
    De=silo[1]+silo[3]
    Vcouv=(pi*(De**2-Di**2)*H)/4
    Acouv=(pi*(De+Di)*H)/2

    return Vcouv,Acouv

def diametreouverture(silopti,tabgrain,grain):
    '''Fonction qui renvoie la plage de diamètres possibles pour l'ouverture
du silo'''
    maxtaillegrain=0
    for i in range(0,len(tabgrain)):
        if grain[i]==1:    # On itère sur les différents grains
            if tabgrain[i][1]>=maxtaillegrain:
                maxtaillegrain=tabgrain[i][1]

    limite1=silopti[1]/2.5          #Trois limites à respecter
    limite2=silopti[1]-30*(maxtaillegrain*10**(-3))
    limite3=10*(maxtaillegrain*10**(-3))

    if limite1>limite2:
        return limite2, limite3
    else:
        return limite1, limite3

def ar(x):
    '''Fonction qui arrondit au centième un float'''
    x=round(x,2)
    return x


def prix(silo,mat,tabmat,Prixmin,Prixmax):
    '''Fonction qui calcule le coût de fabrication d'un silo donnée'''
    tabsiloprix=[]
    for i in range(0,len(silo)): # On itère sur le tableau regroupant
l'ensemble des silos
        AV=couverture(silo[i])
        V=AV[0]  # Volume à couvrir du silo en question
        A=AV[1]  # Aire à couvrir du silo en question
        if mat==0 or mat==1 : # Si métal, le prix est au metre carré donc en
fonction de l'aire couverte
            prix=A*tabmat[mat][4]*(silo[i][3]*10**(3))
```

Annexe : Programme


```
        else: # Si bois ou béton, le priest au metre cube donc en fonction
du volume couvert
            prix=V*tabmat[mat][4]
        prix=ar(prix)
        if Prixmin<=prix<=Prixmax:  #Si le prix du silo entre dans la
fourchette de prix donnée par l'utilisateur, on garde le silo
            tabsiloprix.append([silo[i][0],silo[i][1],silo[i][2],silo[i]
[3],prix])
    return tabsiloprix


def optimisation(silo,choix):
    ''' Programme qui choisi le silo optimal selon un critère précis'''
    if choix=="Prix min":
        compt=4
        minmax="min"
    if choix=="Prix max":
        compt=4
        minmax="max"
    if choix=="Diamètre min":
        compt=1
        minmax="min"
    if choix=="Diamètre max":
        compt=1
        minmax="max"
    if choix=="Hauteur min":
        compt=0
        minmax="min"
    if choix=="Hauteur max":
        compt=0
        minmax="max"
    if choix=="Epaisseur min":
        compt=3
        minmax="min"
    if choix=="Epaisseur max":
        compt=3
        minmax="max"
    if choix=="Volume min":
        compt=2
        minmax="min"
    if choix=="Volume max":
        compt=2
        minmax="max"

    #   Algo min max selon la situation. On choisi le silo le plus "" ou le
moins "" de l'ensemble des silos possibles
    if minmax=="min":
        min=silo[0][compt]
        rgmin=0
        for i in range(0,len(silo)):
            if silo[i][compt]<min:
                min=silo[i][compt]
                rgmin=i
        return silo[rgmin]
    else:
        max=silo[0][compt]
        rgmax=0
        for i in range(0,len(silo)):
            if silo[i][compt]>max:
                max=silo[i][compt]
```



```
                rgmax=i

        return silo[rgmax]

def actulisertexte(mat,nbsilo):
    '''Fonction qui actualise le texte affiché sur la fenêtre graphique'''
    if mat==0: # Si Inox
        TexteInox.set("Nombre de silo(s) en acier inoxidable possible(s) :
"+nbsilo)
    if mat==1: # Si Alu
        TexteAlu.set("Nombre de silo(s) en aluminium possible(s) : "+nbsilo)
    if mat==2: # Si Beton
        TexteBeton.set("Nombre de silo(s) en beton possible(s) : "+nbsilo)
    if mat==3: # Si Bois
        TexteSapin.set("Nombre de silo(s) en bois possible(s) : "+nbsilo)
    if mat==-1: # Si rien n'est selectionné
        TexteInox.set("Nombre de silo(s) en acier inoxidable possible(s) :
"+nbsilo)
        TexteAlu.set("Nombre de silo(s) en aluminium possible(s) : "+nbsilo)
        TexteBeton.set("Nombre de silo(s) en beton possible(s) : "+nbsilo)
        TexteSapin.set("Nombre de silo(s) en bois possible(s) : "+nbsilo)


def Constructionsilo(r, h, mat, tabmat, silopti,limiteouv):
    '''Fonction qui construit un silo optimal dans la fenêtre graphique '''
    # On attribue à chaque partie de la fenêtre (canvas1, etc) un silo dans
un matériau particulier
    if mat==0:
        canvas=canvas1
    if mat==1:
        canvas=canvas2
    if mat==2:
        canvas=canvas3
    if mat==3:
        canvas=canvas4
     # On affiche les informations relative à chaque silo
    txt = canvas.create_text(250, 20, text=tabmat[mat][0], font="Arial 16
italic", fill=tabmat[mat][5])
    txt = canvas.create_text(5, 20, text="Hauteur : "+str(silopti[0])+" m",
font=("Arial 16 italic", "8"), fill=tabmat[mat][5],anchor=SW)
    txt = canvas.create_text(5, 35, text="Diamètre intérieur :
"+str(silopti[1])+" m", font=("Arial 16 italic", "8"), fill=tabmat[mat]
[5],anchor=SW)
    txt = canvas.create_text(5, 50, text="Volume : "+str(silopti[2])+"
m\u00b3", font=("Arial 16 italic", "8"),
    fill=tabmat[mat][5],anchor=SW)
    txt = canvas.create_text(5, 65, text="Epaisseur :
"+str(float(round(silopti[3],4)*10**3))+" mm", font=("Arial 16 italic",
"8"),
    fill=tabmat[mat][5],anchor=SW)
    txt = canvas.create_text(5, 80, text="Prix : "+str(silopti[4])+" euros",
font=("Arial 16 italic", "8"), fill=tabmat[mat][5],anchor=SW)
    txt = canvas.create_text(5, 95, text="Ouverture entre
"+str(limiteouv[1])+" et "+str(limiteouv[0])+" m", font=("Arial 16
italic","8"),
    fill=tabmat[mat][5],anchor=SW)

    # On dessine le mur de droite
    ligne1 = canvas.create_line((largeur/2)+r, hauteur-20-h, (largeur/2)+r,
hauteur-20)
```

Annexe : Programme


```
    ligne1 = canvas.create_line((largeur/2)+r+2, hauteur-20-h,
(largeur/2)+r+2, hauteur-20)
    f.after(10)

    # on dessine le mur de gauche
    ligne2 = canvas.create_line((largeur/2)-r, hauteur-20-h, (largeur/2)-r,
hauteur-20)
    ligne2 = canvas.create_line((largeur/2)-r-2, hauteur-20-h, (largeur/2)-
r-2, hauteur-20)
    f.after(10)

    # On dessine le toit
    ligne3 = canvas.create_line((largeur/2)-r-2, hauteur-20-h, (largeur/2)-
r/2, hauteur-20-0.1*h-h)
    ligne3 = canvas.create_line((largeur/2)+r+2, hauteur-20-h,
(largeur/2)+r/2, hauteur-20-0.1*h-h)
    f.after(10)
    rect1=canvas.create_rectangle((largeur/2)-r/2, hauteur-20-0.1*h-h,
(largeur/2)+r/2, hauteur-25-0.1*h-h)
    f.after(10)

    # On colorie le silo
    rect2 = canvas.create_rectangle((largeur/2)-r,hauteur-20, (largeur/2)+r,
hauteur-20-h,outline=tabmat[mat][5], fill=tabmat[mat][5])
    f.after(10)

    # On dessine les traits sur le silo
    for i in np.arange(0,floor(h)+floor(h)/50,h/50):
        canvas.create_line((largeur/2)-r, hauteur-20-i, (largeur/2)+r+1,
hauteur-20-i)
        f.after(10)
    canvas.pack()

def th():
    ''' On place la fonction de calcul des silos (silomaker())  en arrière
plan de manière à ne pas faire freezer la fenêtre'''
    mon_thread=Thread(target=silomaker)
    mon_thread.start()

def silomaker():
    '''Fonction principale qui se charge de construire un ou des silos
optimal(aux)'''
    global grain
    global tabsilopti
    valider.config(state=DISABLED) # Une fois que les choix de diamètre,
volume, ... sont fait, on desactive temporerement le bouton valider

    dimension=[]# tableau qui prend les silos possibles [[hauteur, largeur,
volume],...]
    silo1=[] # tableau qui prend les silos possibles [[hauteur, largeur,
volume, épaisseur],...]
    silo2=[]  # tableau qui prend les silos possibles [[hauteur, largeur,
volume, épaisseur, prix],...]

    # On récupère les plages et valeurs rentrées par l'utilisateur

    Prixmin=entree1.get()
    Prixmin=float(Prixmin)
    Prixmax=entree2.get()
    Prixmax=float(Prixmax)
```



```
    Vmin=entree3.get()
    Vmax=entree4.get()
    Vmin=float(Vmin)
    Vmax=float(Vmax)

    Hmin=entree5.get()
    Hmin=float(Hmin)
    Hmax=entree6.get()
    Hmax=float(Hmax)
    Dimin=entree7.get()
    Dimin=float(Dimin)
    Dimax=entree8.get()
    Dimax=float(Dimax)
    Emin=entree9.get()
    Emin=float(Emin)
    Emin*=10**(-3)
    Emax=entree10.get()
    Emax=float(Emax)
    Emax*=10**(-3)

    choixopti=liste.get(ACTIVE)
    # On récupère l'optimisation souhaitée par l'utilisateur (Le silo le
moins chère respectant les critères est choisi par défaut sinon)


    mat=[var1.get(),var2.get(),var3.get(),var4.get()]  # On récupère les
matériaux de construciton choisis
    grain=[var5.get(),var6.get(),var7.get(),var8.get(),var9.get()] # On
récupère les matériau granulaires choisis
    # si var=0, le matériau n'est pas choisi, si var=1 il est choisi

    tabsilopti=[[],[],[],[]]

    # On efface les précédents silos
    canvas1.delete(ALL)
    canvas2.delete(ALL)
    canvas3.delete(ALL)
    canvas4.delete(ALL)

    # On réaffiche titres et sols

    rect2 = canvas1.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')
    rect2 = canvas2.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')
    rect2 = canvas3.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')
    rect2 = canvas4.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')

    txt = canvas1.create_text(250, 20, text=tabmat[0][0], font="Arial 16
italic", fill=tabmat[0][5])
    txt = canvas2.create_text(250, 20, text=tabmat[1][0], font="Arial 16
italic", fill=tabmat[1][5])
    txt = canvas3.create_text(250, 20, text=tabmat[2][0], font="Arial 16
italic", fill=tabmat[2][5])
    txt = canvas4.create_text(250, 20, text=tabmat[3][0], font="Arial 16
italic", fill=tabmat[3][5])
```

Annexe : Programme


```
    actulisertexte(-1,"") # On efface le nombre de silo possible puisque
qu'une nouvelle recherche est lancée

    for j in range(0,len(mat)): # Débit de la boucle maitresse du programme,
on itère sur chaque matériau
        if mat[j]==1 and grain!=[0,0,0,0,0]: # si le matériau en question
est choisi on cherche les silos possibles dans ce matériau
            dimension=volume(Hmin,Hmax,Dimin,Dimax,Vmin,Vmax)
            silo1=contrainte(dimension,j,tabgrain,grain,tabmat,tabfrot,Emin,
Emax)
            silo2=prix(silo1,j,tabmat,Prixmin,Prixmax)

            actulisertexte(j,str(len(silo2)))   # On affiche le nombre de
silo possible
            #print("Nombre de solution : " + str(len(silo2)))

            if silo2!=[]:
                silopti=optimisation(silo2,choixopti) # On cherche le silo
optimal
                tabsilopti[j]=silopti
                limiteouv=diametreouverture(silopti,tabgrain,grain)
                Constructionsilo((silopti[1]/2)*4,
silopti[0]*4,j,tabmat,silopti,limiteouv) # On le construit virtuellement

    valider.config(state=NORMAL) # On rétablit l'état du bouton valider, une
nouvelle recherche peut être lancée

## Partie construction de la fenêtre graphique

f = Tk()  # On crée une fenêtre graphique
f.title("Construction d'un silo") # On la nomme
largeur=500 # Dimension de la fenêtre graphique (fg)
hauteur=500


Frame1 = Frame(f, borderwidth=0, relief=GROOVE) # Partie de gauche la fg
dans laquelle on place les options.
Frame1.pack(side=LEFT, padx=0, pady=0) # pack servira à chaque fois à placer
l'objet crée dans la fenêtre graphique

Frame3 = Frame(f, borderwidth=0, relief=GROOVE) # Partie centrale reservée à
la construction des silos alu et inox
Frame3.pack(side=RIGHT, padx=0, pady=0)

Frame2 = Frame(f, borderwidth=0, relief=GROOVE) # Partie de droite reservée
à la contrustion des silos bois et béton
Frame2.pack(side=RIGHT, padx=0, pady=0)

# On crée les carrés dans lesquels on dessine un silo
canvas1=Canvas(Frame2, width=largeur, height=hauteur , bg='ivory')
canvas2=Canvas(Frame2, width=largeur, height=hauteur , bg='ivory')
canvas3=Canvas(Frame3, width=largeur, height=hauteur , bg='ivory')
canvas4=Canvas(Frame3, width=largeur, height=hauteur , bg='ivory')

canvas1.pack(side=TOP, padx=5, pady=5)
canvas2.pack(side=BOTTOM, padx=5, pady=5)
canvas3.pack(side=TOP, padx=5, pady=5)
canvas4.pack(side=BOTTOM, padx=5, pady=5)
```



```
# On crée le sol vert
rect2 = canvas1.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')
rect2 = canvas2.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')
rect2 = canvas3.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')
rect2 = canvas4.create_rectangle(0,hauteur, largeur,
hauteur-20,outline='green',fill='green')

# On donne le matériau du silo construit dans chaque fenêtre
txt = canvas1.create_text(250, 20, text=tabmat[0][0], font="Arial 16
italic", fill=tabmat[0][5])
txt = canvas2.create_text(250, 20, text=tabmat[1][0], font="Arial 16
italic", fill=tabmat[1][5])
txt = canvas3.create_text(250, 20, text=tabmat[2][0], font="Arial 16
italic", fill=tabmat[2][5])
txt = canvas4.create_text(250, 20, text=tabmat[3][0], font="Arial 16
italic", fill=tabmat[3][5])

# Curseur Reservé à la hauteur du silo
h = LabelFrame(Frame1, text="Hauteur (m)", padx=5, pady=5) # On crée la
sous-fenêtre dans laquel on place le curseur
h.pack(fill="both", expand="yes")

Label(h, text="Min").pack(side=LEFT ) # On la légende
Label(h, text="Max").pack(side=RIGHT)


value5 = IntVar()
value6 = IntVar()
value5.set(0.0)
value6.set(100.0)

entree5 = Entry(h, textvariable=value5, width=30)
entree6 = Entry(h, textvariable=value6, width=30)

entree5.pack(expand=YES, fill=Y, side=LEFT, padx=5, pady=5)
entree6.pack(expand=YES, fill=Y, side=RIGHT, padx=5, pady=5)



# Curseur Reservé au diamètre du silo
d = LabelFrame(Frame1, text="Diamètre (m)", padx=5, pady=5)
d.pack(fill="both", expand="yes")

Label(d, text="Min").pack(side=LEFT )
Label(d, text="Max").pack(side=RIGHT)

value7 = IntVar()
value8 = IntVar()
value7.set(0.0)
value8.set(100.0)

entree7 = Entry(d, textvariable=value7, width=30)
entree8 = Entry(d, textvariable=value8, width=30)
```

Annexe : Programme


```
entree7.pack(expand=YES, fill=Y, side=LEFT, padx=5, pady=5)
entree8.pack(expand=YES, fill=Y, side=RIGHT, padx=5, pady=5)

# Curseur Reservé à l'épaisseur du silo
e = LabelFrame(Frame1, text="Epaisseur (mm)", padx=5, pady=5)
e.pack(fill="both", expand="yes")

Label(e, text="Min").pack(side=LEFT)
Label(e, text="Max").pack(side=RIGHT)


value9 = IntVar()
value10 = IntVar()
value9.set(0.0)
value10.set(100.0)

entree9 = Entry(e, textvariable=value9, width=30)
entree10 = Entry(e, textvariable=value10, width=30)

entree9.pack(expand=YES, fill=Y, side=LEFT, padx=5, pady=5)
entree10.pack(expand=YES, fill=Y, side=RIGHT, padx=5, pady=5)

# Champ réservé au prix du silo
p = LabelFrame(Frame1, text="Prix (euros)", padx=5, pady=5)
p.pack(fill="both", expand="yes")

Label(p, text="Min").pack(side=LEFT)
Label(p, text="Max").pack(side=RIGHT)

value1 = IntVar()
value2 = IntVar()
value1.set(0.0)
value2.set(100000000.0)

entree1 = Entry(p, textvariable=value1, width=30)
entree2 = Entry(p, textvariable=value2, width=30)

entree1.pack(expand=YES, fill=Y, side=LEFT, padx=5, pady=5)
entree2.pack(expand=YES, fill=Y, side=RIGHT, padx=5, pady=5)

# Champ réservé au volume du silo
v = LabelFrame(Frame1, text="Volume (m\u00b3)", padx=5, pady=5)
v.pack(fill="both", expand="yes")
Label(v, text="Min").pack(side=LEFT)
Label(v, text="Max").pack(side=RIGHT)

value3 = IntVar()
value4 = IntVar()
value3.set(900.0)
value4.set(1000.0)

entree3 = Entry(v, textvariable=value3, width=30)
entree4 = Entry(v, textvariable=value4, width=30)

entree3.pack(expand=YES, fill=Y, side=LEFT, padx=5, pady=5)
entree4.pack(expand=YES, fill=Y, side=RIGHT, padx=5, pady=5)

# Champ réservé à l'indication sur le nombre de silo possible
res = LabelFrame(Frame1, text="Résultats")
res.pack(side=BOTTOM,padx=5, pady=5)
```



```
TexteInox = StringVar()
TexteAlu = StringVar()
TexteBeton = StringVar()
TexteSapin = StringVar()

TexteInox.set("Nombre de silo(s) en acier inoxidable possible(s) : ")
TexteAlu.set("Nombre de silo(s) en aluminium possible(s) : ")
TexteBeton.set("Nombre de silo(s) en béton possible(s) : ")
TexteSapin.set("Nombre de silo(s) en bois possible(s) : ")

LabelResultatInox = Label(res, textvariable = TexteInox, fg ='black')
LabelResultatAlu = Label(res, textvariable = TexteAlu, fg ='black')
LabelResultatBeton = Label(res, textvariable = TexteBeton, fg ='black')
LabelResultatSapin = Label(res, textvariable = TexteSapin, fg ='black')

LabelResultatInox.pack(side = BOTTOM, padx = 5, pady = 5)
LabelResultatAlu.pack(side = BOTTOM, padx = 5, pady = 5)
LabelResultatBeton.pack(side = BOTTOM, padx = 5, pady = 5)
LabelResultatSapin.pack(side = BOTTOM, padx = 5, pady = 5)

# Case à cocher sur le choix du matériau de construction
mc = LabelFrame(Frame1, text="Matériaux de construction")
mc.pack(side=LEFT,padx=5, pady=5)

var1 = IntVar()
var2 = IntVar()
var3 = IntVar()
var4 = IntVar()

bouton1 = Checkbutton(mc, text="Inox",variable=var1)
bouton2 = Checkbutton(mc, text="Aluminium",variable=var2)
bouton3 = Checkbutton(mc, text="Béton",variable=var3)
bouton4 = Checkbutton(mc, text="Bois",variable=var4)

bouton1.pack()
bouton2.pack()
bouton3.pack()
bouton4.pack()

# Case à cocher sur le choix du matériau granulaire
mg = LabelFrame(Frame1, text="Matériaux granulaires")
mg.pack(side=LEFT,padx=5, pady=5)

var5 = IntVar()
var6 = IntVar()
var7 = IntVar()
var8 = IntVar()
var9 = IntVar()

bouton5 = Checkbutton(mg, text="Soja jaune",variable=var5)
bouton6 = Checkbutton(mg, text="Quinoa",variable=var6)
bouton7 = Checkbutton(mg, text="Haricot mungo",variable=var7)
bouton8 = Checkbutton(mg, text="Blé",variable=var8)
bouton9 = Checkbutton(mg, text="Gravier",variable=var9)

bouton5.pack()
bouton6.pack()
bouton7.pack()
bouton8.pack()
```

Annexe : Programme


```
bouton9.pack()

# Case à cocher sur le choix d'optimisation
op = LabelFrame(Frame1, text="Optimisation")
op.pack(side=LEFT,padx=5, pady=5)

liste = Listbox(op)
liste.insert(1, "Prix min")
liste.insert(2, "Prix max")
liste.insert(3, "Diamètre min")
liste.insert(4, "Diamètre max")
liste.insert(5, "Hauteur min")
liste.insert(6, "Hauteur max")
liste.insert(7, "Epaisseur min")
liste.insert(8, "Epaisseur max")
liste.insert(9, "Volume min")
liste.insert(10, "Volume max")

liste.pack()

tabsilopti=[[],[],[],[]]
grain=[0,0,0,0,0]

valider=Button(Frame1, text='Valider', command=th) # Si le bouton valider
est pressé, on lance la construction en appelant th()
valider2=Button(Frame1, text='Profil de contrainte', command=pression)
valider.pack(side=RIGHT)
valider2.pack(side=RIGHT)

f.mainloop()  # Le programme tourne tant que la fenêtre n'est pas fermée !
```