

**Utilisation du Pilote de CESAR pour le
calcul de l'écoulement entre deux tranchées parallèles
(parabole de Dupuit)**

Mise à jour : janvier 2023

Calcul de l'écoulement entre deux tranchées parallèles

1. Objectifs

Ce document s'adresse à un utilisateur du Pilote de CESAR qui dispose d'une part, du solveur CESAR, et, d'autre part, du Pilote de CESAR. Pour les installer, on pourra se référer à la section 2.

Ce document présente un exemple de modélisation d'un écoulement en milieu non saturé utilisant le Pilote de CESAR pour effectuer un calcul avec le module NSAT du solveur CESAR. Un script en « langage CESAR » permet de préparer l'ensemble des données pour un calcul d'écoulement en régime permanent, de lancer l'exécution et de relever la charge hydraulique à la base du maillage. La solution numérique est validée par comparaison avec une solution analytique approchée connue sous le nom de parabole de Dupuit ou de Dupuit-Forchheimer.

Le domaine étudié étant un rectangle maillé en éléments quadrangulaires, la liste des coordonnées des nœuds et la définition des éléments est constituée directement en quelques commandes Python. Le script utilise donc à la fois les fonctionnalités de la programmation Python et celles du langage CESAR. On pourrait faire appel à un mailleur externe (par exemple gmsh) : le lecteur intéressé pourra se reporter à l'exemple 2 « Calcul d'une courbe charge-déplacement » ("<https://cesar.univ-gustave-eiffel.fr/exemples/>").

2. Prérequis

Les étapes permettant d'installer les logiciels utiles, à partir d'une installation Ubuntu 20.04 « fraîche », sont listées ci-dessous. Il n'est pas nécessaire de les répéter si elles ont déjà été réalisées.

1/ Solveur CESAR

- Télécharger l'archive auto extractible (cesar.univ-gustave-eiffel.fr, page Téléchargements)
- Exécuter le script « .run »
- Contacter cesar-lcpc@ifsttar.fr pour obtenir votre fichier de licence SOLVCESV42.KEYC, à coller dans le répertoire fconfig/ créé par le script d'installation

2/ Pilote de CESAR

- Télécharger l'archive auto extractible (cesar.univ-gustave-eiffel.fr, page Téléchargements)
- Exécuter le script « .run »
- Suivre les instructions pour terminer l'installation (créer un lien symbolique vers le répertoire d'installation du solveur CESAR et installer python3-dev)

3/ Il faut également disposer de matplotlib (installer les paquets python3-numpy et python3-matplotlib).

3. Présentation du problème

On étudie l'écoulement plan qui se produit dans une couche de sol d'épaisseur H ($y \in [0, H]$) entre deux tranchées parallèles distantes d'une longueur L . Dans la tranchée de droite, la hauteur de nappe est égale à H ; la tranchée de droite impose une condition de drainage (pression nulle).

Le domaine étudié est le massif de sol entre les deux tranchées, représenté par le rectangle $[0, L] \times [0, H]$ avec $L=50$ m et $H=10$ m. L'eau s'écoule de droite à gauche. La solution proposée par Dupuit consiste à considérer que le flux à travers une section d'abscisse x est constant (conservation de la masse), et que la charge hydraulique h et la vitesse v dans une section ne dépendent que de x .

En assimilant la charge hydraulique $h(x)$ à la hauteur de la nappe, on a :

$$h(x) v(x) = \text{cte} = D$$

où D représente le débit à travers les sections $x=\text{cte}$.

Dans la partie saturée $y \leq h(x)$, la loi de Darcy donne la vitesse du fluide :

$$v = -k \, dh/dx$$

ce qui conduit, compte tenu de $h(L)=H$, à :

$$-k (h^2 - H^2) / 2 = D (x - L)$$

En imposant $h(x=0)=0$, on détermine la valeur du débit D :

$$D = -k H^2 / 2 L$$

D'autre part, on a une variation « parabolique » de la hauteur de nappe :

$$h = H (x/L)^{1/2}$$

La solution proposée est souvent appelée « parabole de Dupuit-Forchheimer » (on trouve aussi dans la littérature la solution du problème équivalent en axisymétrie).

Pour le gradient de charge, on a $dh/dx = -v / k = -D / kh$; en particulier en $x=L$, on trouve

$$dh/dx(x=L) = H/2L = 0,1.$$

Et pour la vitesse du fluide, on trouve

$$v = -k \, dh/dx = -\frac{k H}{2\sqrt{xL}}$$

4. Exécution du script

On lance l'exécution du script en tapant :

```
/home/user/cespil/bin/run_pilote dupuit.py
```

en adaptant le chemin (en particulier le nom de l'utilisateur `user`) au contexte dans lequel on travaille.

5. Préparation des données

5.1. Début du script en langage CESAR

Pour un script en langage « CESAR », on commence par importer les structures de données ad hoc :

```
from modele_cesar import *
```

On définit le répertoire de travail :

```
rep_trav = './'
```

NB : la ligne ci-dessus doit être adaptée par l'utilisateur pour donner le chemin du répertoire de travail.

On définit un objet de la classe EXEC comportant le mot-clé RSV4, afin que le Pilote recopie le fichier de résultats binaire dans le répertoire de travail à l'issue du calcul : ce fichier sera ensuite exploré pour afficher les résultats.

```
fic_data = rep_trav + 'dupuit_M1.data'
fic_rsv4 = rep_trav + 'dupuit_M1.rsv4'

exe = EXEC( RSV4 = fic_rsv4 )
```

5.2. Définition de la géométrie, du maillage et du modèle

On définit la largeur et la profondeur du domaine maillé et la taille des éléments (supposés carrés) :

```
# parametres geometriques

L = 50.          # largeur du domaine etudie
H = 10.          # epaisseur du domaine

# longueur des elements
d = 1.
```

La géométrie étant très simple, on constitue la liste des coordonnées des nœuds à l'aide de commandes Python :

Le maillage comporte $N_v = H/d$ couches horizontales comportant chacune $N_h = L/d$ éléments. Cela conduit à introduire $2 N_v + 1$ lignes horizontales de nœuds, les lignes impaires ayant $2 N_h + 1$ nœuds, les lignes paires ayant seulement $N_h + 1$ nœuds -parce que les éléments quadrangulaires Q8 n'ont pas de nœud au centre.

Les commandes permettant de définir les lignes de nœuds sont les suivantes :

```
# nombre d elements dans la largeur, dans la hauteur et total
nh = int(L/d)
nv = int(H/d)
nelt = nh*nv
# nombre de noeuds sur les lignes horizontales / verticales
nph = 2*nh+1
npv = 2*nv+1
# largeur et hauteur des éléments
disth = L/nh
distv = H/nv
```

```

# constitution des coordonnees des noeuds
# on fabrique des lignes horizontales qui comportent 2*nh+1 noeuds
# ou nh+1 noeuds (pour les noeuds aux milieux des cotes verticaux)
x = 0.
y = 0.
nnt = 0

# vcorg represente la liste des coordonnees
vcorg = []

for i in range(nph):
    vcorg.append(x)
    vcorg.append(y)
    nnt += 1
    x += disth/2.

for j in range(nv+1):
    x = 0.
    y += distv/2.
    for i in range(nh+1):
        vcorg.extend([x,y])
        nnt += 1
        x += disth
    y += distv/2.
    x = 0.
    for i in range(nph):
        vcorg.extend([x,y])
        nnt += 1
        x += disth/2.

```

On appelle `coor` l'objet de la classe `COOR` contenant la liste des coordonnées :

```

# definition de l objet coor de la classe COOR
coor = COOR( M = 2, M1 = 0, NNT = nnt, NDIM = 2, VCORG = vcorg )

```

On constitue ensuite la connectivité du maillage, c'est-à-dire la liste `KPNT` (ici tous les éléments ont 8 nœuds, donc c'est la suite des entiers $1+8k$, avec $k \in [0, N_h \cdot N_v]$) et la liste `KNTE`, qui permet de définir la numérotation de chaque élément :

```

# constitution de la connectivite

kpnte = [1]
knte = []

ielt = 1
for i in range(nelt):
    ielt += 8
    kpnte.append(ielt)

idecv = 0

i1 = 1 ; i2 = 3 ;      i3 = nph+nh+4 ; i4 = nph+nh+2
i5 = 2 ; i6 = nph+2 ; i7 = nph+nh+3 ; i8 = nph+1

```

```

for i in range(nv):
    idec1 = 0
    idec2 = 0
    for j in range (nh) :
        n1 = i1+idec1+idecv ; n2 = i2+idec1+idecv
        n3 = i3+idec1+idecv ; n4 = i4+idec1+idecv
        n5 = i5+idec1+idecv ; n6 = i6+idec2+idecv
        n7 = i7+idec1+idecv ; n8 = i8+idec2+idecv
        idec1 = idec1+2
        idec2 = idec2+1
        knte.extend( [ n1,n2,n3,n4,n5,n6,n7,n8 ] )
    idecv += nph+nh+1

```

5. 3. Caractéristiques de la couche de sol

On définit un objet `modele1` définissant le modèle de la teneur en eau et de perméabilité en fonction de la hauteur piézométrique. L'objet `modele1` est constitué d'un groupe d'éléments de classe DB (pour « Diffusion, Bidimensionnel »). On adopte ici le modèle reposant sur des courbes « préprogrammées » proposé par le module NSAT du solveur CESAR (voir l'annexe), pour lequel l'indicateur IMOD prend la valeur 4 :

```

modele1 = [DB(NOMG = 'G2',
              ACTI = 'A',
              IMOD = 4,
              INAT = 1,
              CARA = DB_CP( AKXS = 1., AKYS = 1., AKXYS = 0., CE = 1.,
                           A = 1., B = 2., C = 1., D = 2.,
                           CER = 0., CES = 0., P0 = 0.1)
            )]

```

Note : comme le solveur CESAR, le Pilote ne gère pas les unités des grandeurs physiques. C'est à l'utilisateur de choisir le système d'unités qu'il utilise et de veiller à fournir des données cohérentes avec ce choix.

D'autre part, le paramètre P0 est ici assimilable à un intervalle de valeurs de la hauteur piézométrique correspondant à la transition entre les régimes saturé et non-saturé : il ne s'agit pas d'une pression, mais d'une hauteur d'eau (ce qui explique pourquoi le poids volumique du fluide ne fait pas partie des données du modèle). Il est intéressant de signaler que, si l'on donne une valeur trop faible, la transition est abrupte, et on risque d'avoir de mauvais résultats, à cause de la difficulté d'estimer correctement

le terme $\frac{d\theta}{d\psi}$ de l'équation de Richards.

On constitue ensuite un objet de la classe ELEM pour définir le maillage (nœuds, connectivité, type des éléments pour un calcul en diffusion 2D, et le groupe auquel chaque élément appartient) et les propriétés de chaque groupe d'élément (ici, il y en a un seul) :

```

elem = ELEM( M = 0, M1 = 0, NELT = nelt , NGRPE = 1,
            PNUMEL = kpnte, NUMEL = knte,
            TYPE = ['DBQ8'] * nelt,
            GROUPE = [1] * nelt,
            GRPES = modele1
            )

```

5. 4. Conditions aux limites

Les conditions aux limites à fournir dans NSAT sont les suivantes :

- la charge hydraulique h est constante et égale à 10 m sur le côté droit du maillage ($x=50$ m)
- à gauche, on impose une charge nulle $h=0$; comme h est définie comme la somme de la hauteur hydraulique ψ et de la cote z , cela revient à imposer des valeurs de ψ négatives, donc à imposer que la limite gauche est non saturée.

On est donc amené à constituer la liste des (numéros des) nœuds situés sur le côté gauche, où la charge est nulle, et la liste des nœuds sur le côté droit, où la charge est imposée à une valeur non nulle.

```
# definition des conditions aux limites :
# on constitue la liste des noeuds situés en x=0 et en x=L
# (de bas en haut)

nd_charge_nulle = []
nd_charge_impos = []
uimp = []
nlg = 1
nld = 2*nh+1
nd_charge_nulle.append(nlg)
nd_charge_impos.append(nld)
uimp.append(10.)
for i in range (nv) :
    nlg += 2*nh+1
    nld += nh+1
    nd_charge_nulle.append(nlg)
    nd_charge_impos.append(nld)
    uimp.append(10.)
    nlg += nh+1
    nld += 2*nh+1
    nd_charge_nulle.append(nlg)
    nd_charge_impos.append(nld)
    uimp.append(10.)
```

On définit ensuite les conditions correspondantes, pour imposer des degrés de liberté nuls (GEN_NUL) ou non nuls (GEN_IMP) :

```
## definition des conditions aux limites

cond = COND( M = 2,
             OPT = [COND_NUL(GEN = [GEN_NUL(IGEN = 2,
                                             NP = len(nd_charge_nulle),
                                             NUM = nd_charge_nulle,
                                             IDL = [1],
                                             NDLN = 1) ] ),
                   COND_IMP(GEN = [GEN_IMP(IGEN = 2,
                                             NP = len(nd_charge_impos),
                                             NUM = nd_charge_impos,
                                             IL = 1,
                                             UIMP = uimp )])
             ] )
```

5. 5. *Chargements*

Il n’y a pas de terme de source, ni de flux imposé sur le contour, et donc pas de chargements à définir.

5. 6. *Définition des paramètres du calcul*

L’analyse que l’on veut effectuer est un calcul d’écoulement en milieu non saturé. On précise l’orientation de la direction verticale (par rapport au maillage), la méthode utilisée, le nombre de pas de temps, le nombre maximal d’itérations pour chaque pas de temps, la tolérance demandée, etc. :

```
# definition des parametres du calcul

kvcond = KVCOND( KCOND = [1], VCOND = [1.] )

ini = INI( MOD = 'NSAT', IPERM = 1, M2 = 1, VU0 = 0. )

lim = LIM('NSAT', TAB = kvcond )

sre = SRE( MOD = 'NSAT', KSRE= [1], ISRC =2)

nsat = NSAT( M = 1, V = [0., 1., 0.],
             MET1 = 'N', MET2 = 'H',
             NITER = 50, TOL = 1.e-3, NPAS1 = 1,
             T0 = 0. , DT = [ ] ,
             OPT = [ ini, lim, sre ] )
```

5. 7. *Constitution du jeu de données et lancement du calcul*

On constitue un objet de la classe JDD qui réunit les données du calcul. La classe JDD possède une méthode `lancer` qui déclenche l’exécution du calcul, en faisant appel au solveur CESAR :

```
jeu = JDD( EXEC = exe,
           COOR = coor,
           ELEM = elem,
           COND = cond,
           NSAT = nsat)

jeu.lancer()
```

L’exécution du calcul produit :

- un fichier binaire contenant le maillage : `dupuit_mail.resu`,
- un fichier binaire contenant les résultats `dupuit_M1.rsv4`,
- un fichier texte contenant différentes informations relatives au déroulement du calcul `dupuit_M1.list`. Son contenu peut être paramétré à l’aide de la classe IMPR, qui permet d’imprimer dans ce fichier les déplacements de tous les nœuds du maillage ou seulement certains, les contraintes aux nœuds de tout ou partie des éléments.

6. Exploitation des résultats

6.1. Utilisation de la classe *RSV4* et extraction des résultats

On utilise ici la classe *RSV4* pour explorer le fichier de résultats binaire constitué par le solveur CESAR.

```
rsv4 = RSV4 (jeu, fic_rsv4)
```

Le fichier de résultats binaire contient :

- des « entités » qui correspondent aux pas de temps (TimeStep) dans le calcul. Ici, le calcul est fait en régime permanent ; il y a donc un seul « TimeStep »
- des tables de type « NodesByGroup » pour les inconnues principales (ici, la charge hydraulique h),
- des tableaux de type « ElementsByGroup » pour les inconnues secondaires (ici, les vitesses).

Les tables comportent elles-mêmes des « datasets ».

Les datasets correspondant à la charge hydraulique sont associés au code « Hydraulic Head ». Ils comportent une seule composante par nœud. Les datasets correspondant à la vitesse horizontale sont associés au code « 2D Hydraulic Head velocities ». Ils comportent deux composantes par nœud (une pour la vitesse horizontale, une pour la vitesse verticale).

La base du maillage ($y=0$) est constituée par les $N_{ph} = 2N_h + 1$ premiers nœuds (numérotés de 1 à $2N_h + 1$ de gauche à droite entre $x=0$ et $x=L$). Le principe est de définir une liste H_num pour recueillir les valeurs de la charge pour les nœuds de la base du maillage.

On constitue une autre liste Vx_num pour les vitesses horizontales, qu'on relève uniquement pour le nœud de chaque élément situé sur le milieu du côté $y=0$.

On constitue en parallèle la liste des abscisses des nœuds pour chacune des listes, et la liste des valeurs données par l'approximation de Dupuit.

Le code ci-dessous illustre l'exploration du fichier *.rsv4*. On relève la première composante (d'indice 0) pour chacune des tables concernées (la charge hydraulique et la vitesse horizontale).

```
index_compH = 0
H_num = []
H_Dupuit = []
AbscisseH = []
xH = 0.

# pour la vitesse horizontale, Vx est 1re composante du dataset
index_compV = 0
Vx_num = []
Vx_Dupuit = []
AbscisseVx = []
xV = disth/2.
for entity in rsv4.entities:
    if isinstance(entity, TimeStep):
        for table in entity.tables:
            if isinstance(table, NodesByGroup) and table.type_dataset == "Hydraulic
Head":
                node=0
                for ind in range(nph):
```

```

        node += 1
        valeur_num = table.val[node][index_compH]
        H_num.append(valeur_num)
        valeur_analytique = H*(xH/L)**0.5
        H_Dupuit.append(valeur_analytique)
        AbscisseH.append(xH)
        xH += disth/2.

    if isinstance(table,ElementsByGroup) and table.type_dataset == "2D
Hydraulic Head velocities":
        elt=0
        for ind in range(nh):
            elt += 1
            node = 2
            valeur_num=table.val[elt][node-1][index_compV]
            Vx_num.append(valeur_num)
            valeur_analytique=-H/(2*(xV*L)**0.5)
            Vx_Dupuit.append(valeur_analytique)
            AbscisseVx.append(xV)
            xV += disth

```

6.2. Utilisation de matplotlib

Le code ci-dessous affiche deux figures permettant de comparer la solution numérique et la solution de Dupuit.

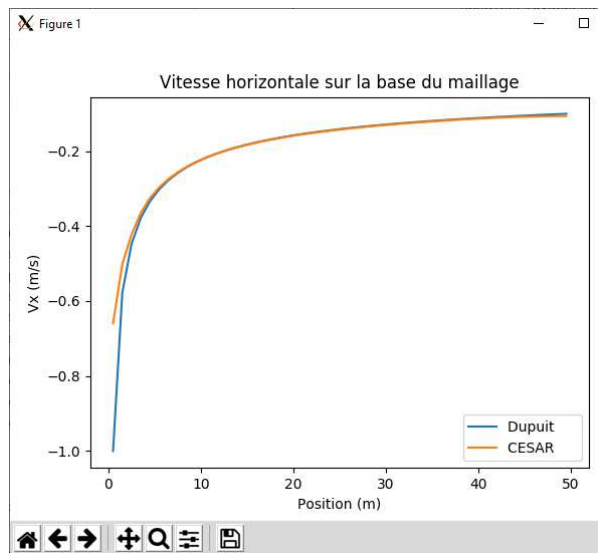
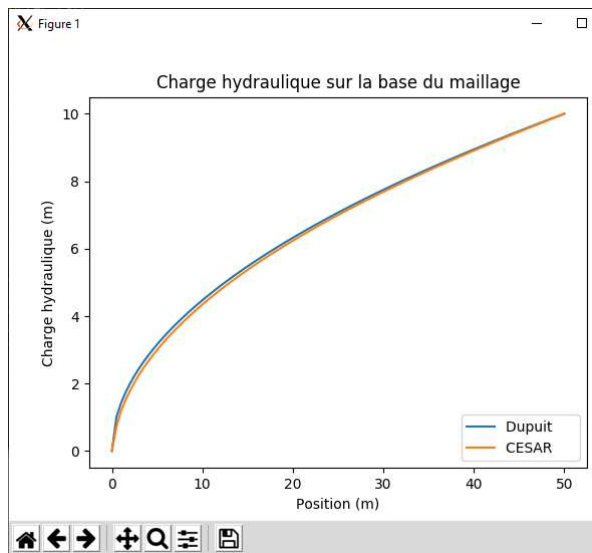
```

import matplotlib.pyplot as plt

plt.title('Charge hydraulique sur la base du maillage')
plt.plot(AbscisseH, H_Dupuit)
plt.plot(AbscisseH, H_num)
plt.xlabel("Position (m)")
plt.ylabel("Charge hydraulique (m)")
plt.legend(['Dupuit', 'CESAR'], loc='lower right')
plt.show()

plt.title('Vitesse horizontale sur la base du maillage')
plt.plot(AbscisseVx, Vx_Dupuit)
plt.plot(AbscisseVx, Vx_num)
plt.xlabel("Position (m)")
plt.ylabel("Vx (m/s)")
plt.legend(['Dupuit', 'CESAR'], loc='lower right')
plt.show()

```



7. Conclusion

Ce document présente un exemple d'utilisation du Pilote de CESAR, qui met en œuvre le « langage CESAR » pour un calcul d'écoulement dans un milieu non saturé. Le script utilise le langage Python pour créer le maillage d'un domaine rectangulaire avec des éléments quadrangulaires à 8 nœuds. On peut paramétrer la taille des éléments : ils sont ici carrés, mais on pourrait facilement définir des éléments rectangulaires, et/ou mailler plus finement en bas et à gauche du maillage.

La mise en données est très simple. On montre comment extraire la charge calculée sur la base du maillage et la comparer à la solution de référence.

8. Annexe

Rappels sur le module NSAT

Le module NSAT modélise les écoulements dans des milieux poreux saturés ou non en résolvant l'équation de Richards :

$$\frac{d\theta}{d\psi}(\psi) \cdot \frac{\partial h}{\partial t} - \text{div} [k_r(\psi) \underline{K}_s \cdot \underline{\text{grad}} h] = q$$

où

- ψ désigne la hauteur piézométrique, définie comme l'écart entre la pression et la pression atmosphérique p_{atm} rapporté au poids volumique du fluide $\psi = \frac{p-p_{\text{atm}}}{\rho_w g}$;
- $\theta(\psi)$ désigne la teneur en eau ;
- h désigne la charge hydraulique, $h = \psi + z$ (z étant la cote du point considéré) ;
- \underline{K}_s est le tenseur des coefficients de perméabilité de Darcy à saturation ;
- q est un terme de source (ou débit) volumique éventuel, pouvant modéliser par exemple l'absorption de l'eau par une végétation diffuse.

Seule la phase liquide est représentée, la pression de l'air étant supposée connue et égale à p_{atm} .

En pratique, le module NSAT utilise une seule variable pour chaque nœud, qui est la hauteur piézométrique ψ . L'équation permet de décrire les aspects saturé ou non-saturé de l'écoulement au travers des fonctions $\theta(\psi)$ et $k_r(\psi)$: elles représentent les variations de la teneur en eau (volumique) θ et la perméabilité relative k_r en fonction de ψ .

Les fonctions $\theta(\psi)$ et $k_r(\psi)$ présentent en général les allures suivantes :

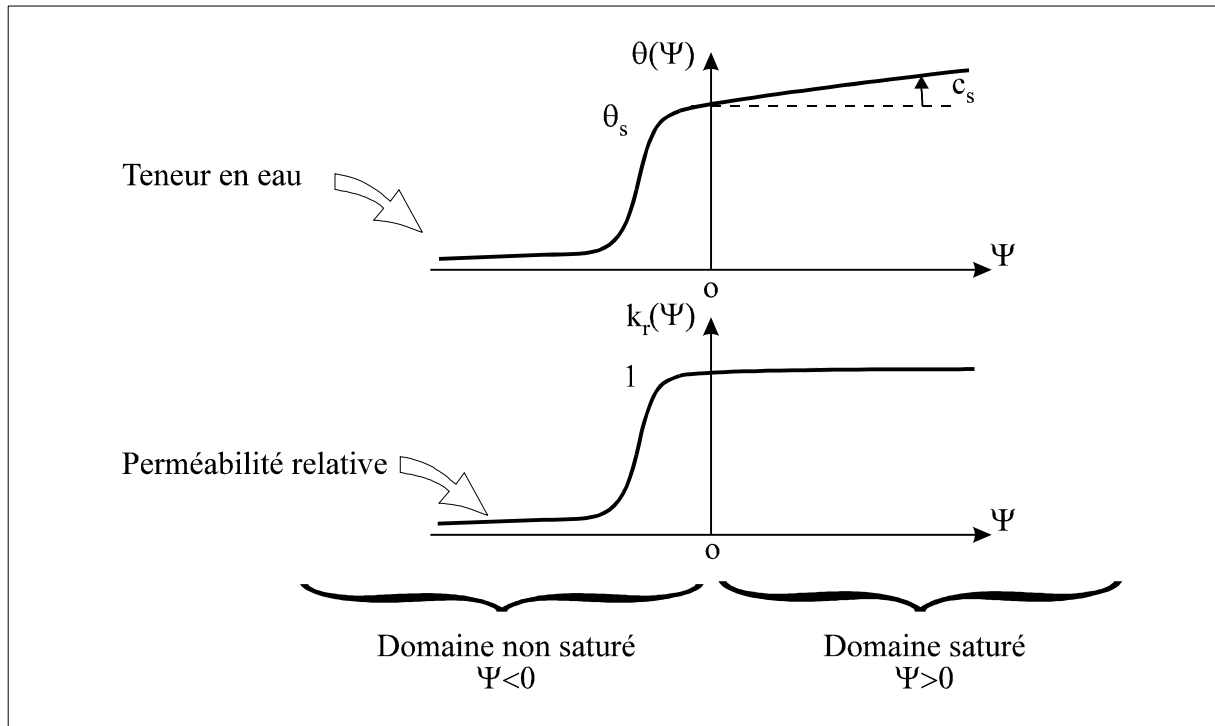


Fig. 1 - Courbes de teneur en eau et de perméabilité relative en fonction de la pression du fluide.

Dans le domaine saturé ($\psi > 0$), la teneur en eau présente l'allure d'une droite horizontale, ou de pente positive C_s très faible, si l'on tient compte des compressibilités des grains solides, du milieu poreux et du fluide (due principalement à la présence de bulles d'air au sein de celui-ci).

Dans le domaine non saturé ($\psi < 0$), la teneur en eau décroît rapidement vers 0 lorsque la succion augmente (c'est-à-dire lorsque ψ diminue). Il en résulte que la partie prépondérante du terme d'emménagement $\frac{d\theta}{d\psi}$ est due à la présence des zones non-saturées. Ce terme peut même être nul dans la zone saturée, si l'on néglige C_s .

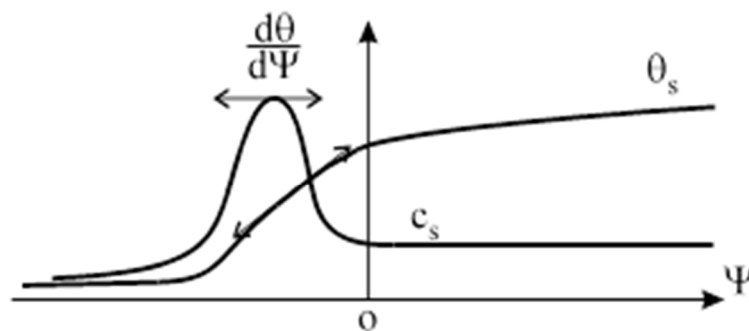


Fig. 2 - Représentation du coefficient d'emménagement en fonction de ψ .

De façon similaire, la perméabilité relative k_r est constante et égale à 1 dans le domaine saturé, et rapidement décroissante en fonction de la succion dans le domaine non-saturé.

Les courbes $k_r(\psi)$ et $\theta(\psi)$ doivent être déterminées expérimentalement. Dans la modélisation avec CESAR, l'utilisateur a le choix entre plusieurs options, qui correspondent à des valeurs différentes de l'indicateur IMOD qui définit le modèle que l'on associe à un groupe d'éléments du maillage :

- IMOD=4 correspond à l'utilisation de courbes préprogrammées

- IMOD=40 la donnée des valeurs prises par les fonctions $\theta(\psi)$ et $k_r(\psi)$ en un certain nombre de points. On a introduit récemment de nouveaux modèles de perméabilité relative et de teneur en eau (modèle de van Genuchten et Gardner).

Le module NSAT présente deux originalités :

- la prise en compte de non-linéarités généralement plus « raides » que ceux relevant du domaine d'application de DTNL (hormis peut-être les problèmes de séchage), liées à la forme des courbes de teneur en eau et de perméabilité dans la zone de transition entre les domaines saturé et non saturé ;
- l'introduction d'une non linéarité supplémentaire, liée à la recherche des surfaces de suintement, c'est-à-dire de l'intersection entre la limite entre zones saturée et non saturée et les contours du domaine étudié. Cette position est recherchée de manière itérative.

Compte tenu de la nature des non linéarités traitées par le module NSAT, la convergence peut être difficile à obtenir. Le fonctionnement du module prévoit une sous-incrémentation automatique du pas de temps fourni par l'utilisateur en cas de non convergence. Toutefois, ce procédé ne dispense pas d'une calibration appropriée des coefficients des lois de perméabilité et de teneur en eau.

Le modèle à courbes préprogrammées IMOD=4

Pour prendre en compte la zone non saturée, il faut utiliser un modèle non linéaire. Dans l'exemple présenté, on utilise le modèle IMOD=4 dans lequel les lois de perméabilité et de teneur en eau dans le domaine non saturé sont les suivantes :

$$K_r = A / (A + (-\psi/\psi_o)^{*}B)$$

$$\theta_r = C / (C + (-\psi/\psi_o)^{*}D)$$

On prend ici $A=C=1$; $B=D=2$.

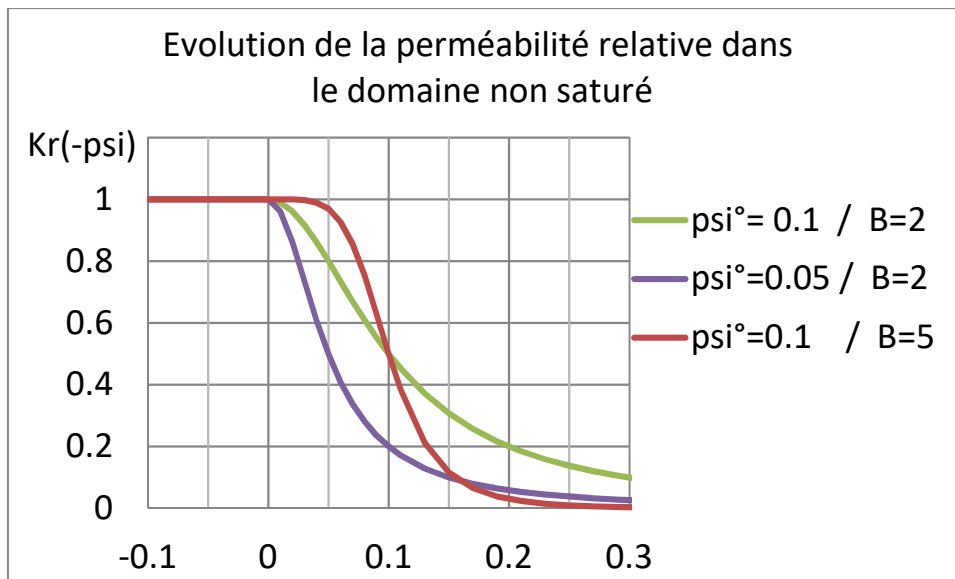
On prendra garde au fait que le poids volumique du fluide ne figure pas dans les données du module NSAT ni dans les données du groupe d'éléments. Le module NSAT fonctionne donc en charge hydraulique et en hauteur hydraulique, si bien que ψ_o caractérise en fait la hauteur sur laquelle s'étend la transition entre la zone saturée et la zone non saturée. On aura donc intérêt à ne pas lui donner une valeur trop faible devant la taille des éléments du maillage : cela rend la convergence de l'algorithme plus difficile.

Pour illustrer ce point, on représente la fonction $K_r(-\psi)$ pour $\psi \in [0 ; 1]$.

Avec $A=1$, K_r vaut 0,5 pour $\psi=\psi_o$.

Si on diminue la valeur de ψ_o avec $A=1$ et $B=2$, K_r diminue donc de manière plus rapide.

Si on conserve pour ψ_0 la valeur 0,1, si on prend $B=4$ au lieu de $B=2$, la transition se fait pour des valeurs voisines de 0,1 mais sur un intervalle plus court : la transition est à nouveau plus abrupte qu'avec les valeurs de référence.



Par ailleurs, l'utilisateur doit fournir : la porosité efficace CE , le coefficient d'emmagasinement dans le domaine saturé CES et le coefficient d'emmagasinement résiduel dans le domaine non saturé CER . On prend ici $CE=0.3$ et $CES=CER=0.0$.