scripts python
pour la construction de diagrammes de phase avec arxim

contents

introduction

Arxim comprend des procédures pour le calcul de spéciation ou d'équilibre suivant une suite univariante de conditions prédéfinies (calculs de type SPCPATH ou EQUPATH), mais ne comprend pas de procédures pour étudier un système suivant plus d'une "dimension".

Arxim permet par exemple de calculer un diagramme de Sillen (activités des espèces en fonction du pH), mais ne permet pas de construire directement un diagramme sur un espace bivariant, tel que pH - redox. De même, on peut calculer l'évolution de la paragénèse métamorphique d'une composition globale donnée le long d'un chemin T-P, mais on ne peut pas faire varier indépendamment T et P en vue de construire un diagramme de phase.

Plutôt que d'étendre les fonctionnalités d'arxim par des procédures internes au code fortran, il a paru plus pratique de développer des programmes externes dans un langage de script - en l'occurrence python - plus facile à mettre en œuvre qu'un langage à compilation séparée comme fortran.

Certaines des méthodes développées ici pourront être ensuite ré-implémentées en fortran pour résoudre plus efficacement certains problèmes (diagrammes T-P par exemple).

Les scripts python proposés ici ont été développés avec **python 2.7**. La seule librairie non standard utilisée est **matplotlib**.

développements en cours

Les nouveaux développements ont donc porté surtout sur ces scripts python, mais Arxim a été aussi amendé pour faciliter les échanges avec un script externe: on a ajouté des procédures d'écriture "compacte" des résultats du calcul, pour simplifier l'accès du script externe aux résultats du calcul arxim. On a aussi remis en fonction la possibilité de renseigner le mode de calcul (SPC, EQU, ou GEM) en argument de la ligne de commande. Le lancement d'arxim par la fonction python *os.system* s'effectue ainsi avec trois arguments: le chemin du script arxim, le niveau de débogage, et le mode de calcul (SPC, GEM). Le niveau de débogage doit être 1, pour que les dialogues écran-clavier soient désactivés lors du run. Dans les scripts réalisés jusqu'ici, arxim est appelé seulement en mode de calcul ponctuel; il pourra être intéressant à terme d'intégrer aussi des calculs de type PATH, qui, pour diverses raisons, seraient a priori plus rapides.

Les scripts actuellement disponibles permettent de réaliser:

- des diagrammes binaires de type pH - Eh (ou pH - log(fO2)) en mode **SPC** (spéciation), représentant l'espèce la plus abondante, à T,P donnés et pour des quantités totales données d'un ou plusieurs éléments. Il s'agit donc ici de diagrammes "de prédominance", et non de diagrammes de phase au sens strict.

- des diagrammes de phase, calculés en mode **GEM**, qui "cartographient", pour des quantités données de constituants inertes, les champs de stabilité des assemblages de phase en fonction des activités de deux constituants mobiles ou en fonction de la température et la pression (mais pas, pour l'instant, en espace température - activité).

Dans le premier cas, Arxim est appelé pour calculer la spéciation de la solution aqueuse et écrire la composition de la solution dans un fichier (*tmp_spc_mole.dat*), que le script python lit ensuite pour trouver l'espèce solutée la plus abondante. Dans les autres cas, Arxim est appelé pour calculer la composition du système à l'équilibre et l'écrire dans un fichier (*tmp_gem.tab*) qui sera relu par le script.

Les fichiers temporaires (*tmp_**) peuvent également être relus par Arxim (niveau 1, à fixer par *sDebug*) pour servir de données initiales au point de calcul suivant, ce qui doit (ou devrait) accélérer le calcul de spéciation. (cet aspect n'est apparemment pas au point, en cas de problème faire le calcul avec *sDebug*="*2*")

méthode de localisation des limites de domaines

Pour ces différents types de diagrammes, le schéma du script est similaire. On commence par un calcul point par point sur une grille (x,y) de résolution donnée, relativement large, qui donne une première image de la répartition des différents domaines (nature de l'espèce la plus abondante, en mode SPC, ou assemblage de phase, en GEM), et qui permet de localiser les couples de points (p1,p2) relevant de deux domaines distincts. Les limites des domaines sont ensuite précisées par dichotomies successives (suivant x pour les couples de même y, ou y pour les couples de même x). On peut faire ce raffinement séparément, après le calcul initial de tous les points de la grille, ou au fur et à mesure qu'on trouve un point dont un voisin direct déjà traité n'est pas du même domaine. La deuxième approche est a priori plus rapide quand arxim peut travailler en "warm restart", et c'est le cas en mode SPC (et en GEM en l'absence de mélanges susceptibles de démixion), mais les premiers tests ne montrent pas de différences significatives de temps de calcul. Le raffinement est effectué de la manière suivante: si le milieu du segment p1-p2 appartient au même domaine que p1, le calcul suivant se fera avec ce point milieu comme nouveau point p1; inversement, si le milieu de p1-p2 tombe dans le domaine de p2, il sera le nouveau point p2 à l'itération suivante. L'itération s'arrête quand la distance p1-p2 passe en dessous d'un seuil de tolérance donné par l'utilisateur.

```
def refine(F0,F1,x0,x1,tolerance):
while abs(x0-x1)>tolerance:
  x= (x0+x1) /2.    #----------------interval halving
  include_modify(x) #----------------update the include file
  OK= arxim_ok()    #----------------arxim run
  if OK:
    paragen= arxim_result()      #------phase assemblage at x
    if paragen in lis_paragen:
      F= lis_paragen.index(paragen)
      # if the paragen at x is the same as the paragen at x0,
      # then the paragen change occurs between x and x1,
      # then x becomes the new x0
      if   F==F0: x0= x
      elif F==F1: x1= x
      else:
        OK= False
```

```
        break
    else:
      lis_paragen.append(paragen)
      F= lis_paragen.index(paragen)
      OK= False
      break
if OK: x= (x0+x1)/2.
return x,F,OK
```

Il peut arriver, quand la maille de la grille de calcul est plus large qu'un domaine, qu'entre les domaines 1 et 2 des points p1 et p2, un domaine 3 distinct à la fois de 1 et 2 soit en fait présent. On repart alors sur deux procédures de raffinement qui fournissent en général, au lieu d'un seul point de la frontière 1=2, deux points, l'un sur la frontière 1=3, l'autre sur 2=3. On ne peut exclure qu'on ne trouve encore un nouveau domaine lors de ces raffinements de deuxième degré, mais ce point n'est pour l'instant pas traité.

Les listes, gérées de manière dynamique en python, et les tuples (type de variable python pour gérer des collections d'objets hétérogènes) permettent de gérer assez simplement les limites entre les différents domaines. On commence par une liste globale à laquelle, chaque fois qu'on rencontre un nouveau point de changement de domaine, on ajoute un "tuple" qui contient les coordonnées du point et un code décrivant le changement de domaine. Il suffit ensuite de relire cette liste pour en extraire les listes correspondant aux différents changements de domaine répertoriés au fur et à mesure du balayage de la grille de départ.

mise en œuvre

Concernant le script arxim (fichier *.inn* décrivant le système, la base de données, etc), le principal aménagement par rapport au script utilisé pour un run arxim classique consiste à écrire dans un fichier à part ("fichier include") le bloc sur lequel le script python doit intervenir.

Pour la construction de diagrammes d'espèces dominantes en solution aqueuse (mode SPC), ainsi que pour les diagrammes de phase dans l'espace T,P, c'est le bloc SYSTEM que le script python modifie à chaque pas de calcul, car c'est lui qui permet de définir les contraintes de pH ou de redox, la pression, la température.

Pour la construction de diagrammes de phases en fonction des activités de constituants mobiles (équivalents à des "diagrammes en potentiels"), c'est un bloc SPECIES que l'on devra modifier.

Par exemple, pour construire un diagramme pH – redox, le script de calcul de spéciation standard comprend au minimum les blocs suivants:

```
SYSTEM
  TDGC 25.
  FE MOLE FE+3  1E-3
  H      PK        H+ 2
  OX     PK        O2(G) 3
END
INCLUDE dtb/elements.dtb
INCLUDE dtb/logk_eq36_hkf.dtb
../..
```

Il est réécrit comme suit:

```
INCLUDE inn/a5a3_fe_ox.include
INCLUDE dtb/elements.dtb
INCLUDE dtb/logk_eq36_hkf.dtb
../..
```

le bloc SYSTEM étant écrit sur un fichier séparé:

```
!! file path: inn/a5a3_fe_ox.include
SYSTEM
  TDGC 25.
  FE MOLE FE+3  1E-3
  H  PK   H+    2
  OX PK   O2(G) 3
END
```

Par défaut, le nom du fichier (i.e. le chemin relatif au répertoire de travail du script python) est celui du script arxim maître dans lequel le suffixe *.inn* est remplacé par *.include*.

diagrammes de prédominance en solution aqueuse

```
script python: test/script/arxim_map_dominant.py
```

Ce script est dédié à la construction d'un diagramme classique de prédominance des différentes espèces du fer en fonction du pH et de l'activité de l'oxygène.

Il fait appel au script arxim suivant:

fichier inn/map1a_fe_ox.inn

```
INCLUDE inn/map1a_fe_ox.include

INCLUDE dtb/elements.dtb
INCLUDE dtb/logk_eq36_hkf.dtb
SOLVENT
  MODEL DAV_1 !DH1 !
END
```

Le bloc include définit le système:

```
SYSTEM
  TDGC 25.
  FE  MOLE FE+3  1E-3
  H   PK   H+    13
  OX  PK   O2(G) 4
END
```

et les paramètres variables sont définis dans le script python:

```
lisX= ["H"]
lisY= ["OX"]
iKeyword= 0
iValue=   3
```

Ce qui signifie qu'on fera varier suivant x la ligne contenant le mot "H" en position 0 (en python, l'indexation des listes et autres tableaux commence à zéro), et en y la ligne contenant le mot "OX" en position 0 ... La ligne "FE" n'est pas modifiée: la quantité totale de Fer dans le système est de 1 millimole.

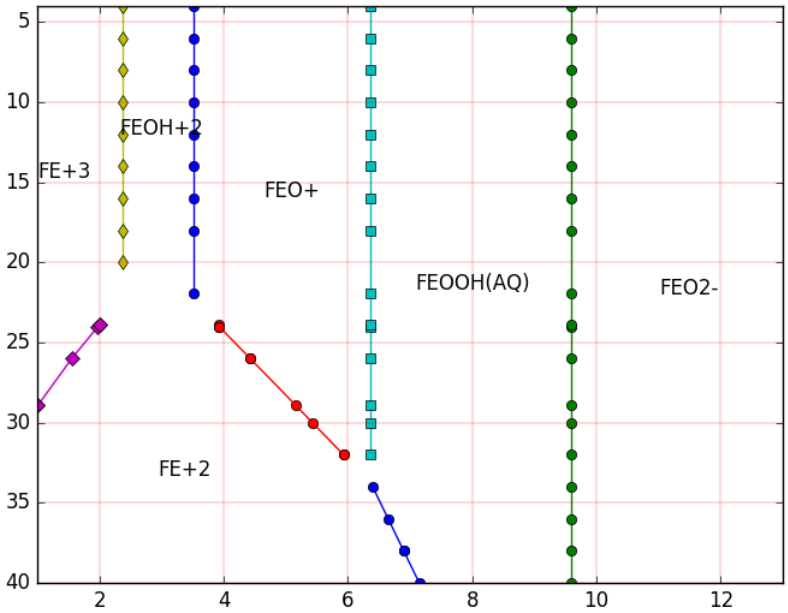
Les paramètres que l'utilisateur.e est susceptible de modifier sont pour l'essentiel dans le bloc dénoté "input file":

```
#-------------------------------------------------------------input files
fInn= "inn/map1a_fe_ox.inn"

lisX= ["H"]
lisY= ["OX"]

Xmin,Xmax,Xdelta= 1., 13., 1.
tol_x= 0.02

Ymin,Ymax,Ydelta= 40., 4., 2.
tol_y= 0.02
```

```
#-----------------------------------------------------------//input files
```

Le diagramme obtenu est disponible à l'adresse suivante:

https://github.com/moutte/arxxim/blob/master/test/png/map1a_fe_ox.png



On obtient bien la topologie générale de ce type de diagramme, avec des limites entre différents domaines de prédominance localisées avec une précision suffisante par la fonction de raffinage, mais il faudrait un traitement supplémentaire des extrémités de lignes pour avoir un diagramme complet.

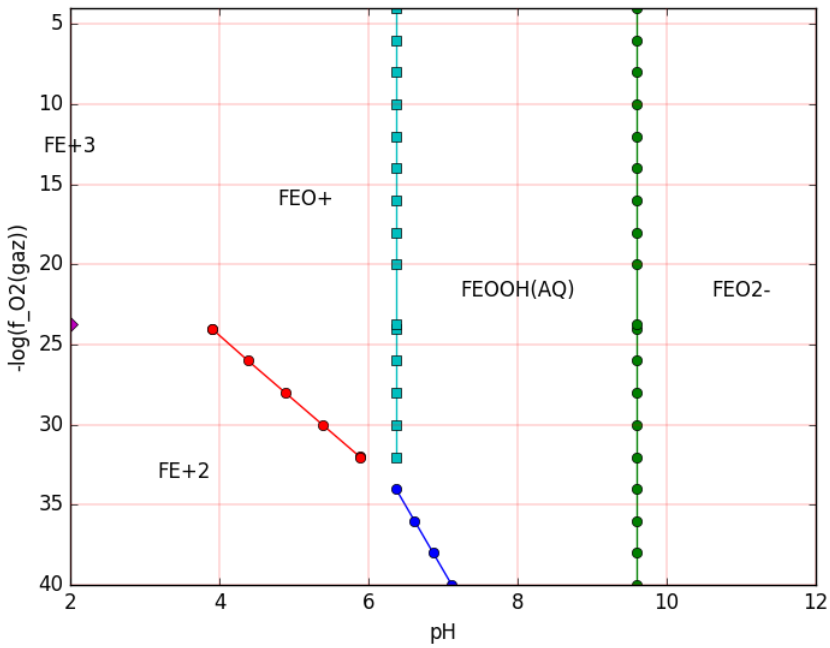
Un test avec une grille de départ plus large montre l'importance du choix de la maille de départ. Si, au lieu de

```
Xmin,Xmax,Xdelta= 1., 13., 1.
```

on choisit

```
Xmin,Xmax,Xdelta= 2., 12., 2.
```

on va manquer certains champs, et certaines limites ne sont pas repérées:



Les calculs arxim sont fait en mode SPC, donc en spéciation de la phase aqueuse, sans s'intéresser aux sursaturations éventuelles vis-à-vis de phases solides. Il est prévu, dans un deuxième temps, de faire apparaître les domaines de sursaturations de phases non-aqueuses, soit en récupérant les taux de saturation des minéraux et gaz par rapport à la solution aqueuse, soit en travaillant en mode EQU.

Les diagrammes présentés ici montrent l'espèce la plus abondante dans les différents domaines. On peut aussi carter les domaines suivant la nature de l'espèce dont l'activité est la plus élevée. Dans le cas présent, les diagrammes résultants seront pratiquement identiques, qu'il s'agisse d'activité ou d'abondance. Le switch entre abondance et activité s'effectue au niveau du choix du fichier de résultats arxim qui est lu par python:

```
fResult= "tmp_species_mole.dat"
```

pour représenter les espèces dominantes en abondance,

```
fResult= "tmp_species_lact.dat"
```

pour représenter les espèces dont l'activité est la plus élevée.

Pour paramétrer le caractère redox du système par le potentiel de l'électron, pe, plutôt que par la fugacité d'oxygène, on part de l'équilibre de dissociation de H2O:

$2\ H_2O \leftrightarrow O_2 + 4\ H^+ + 4\ e^-$

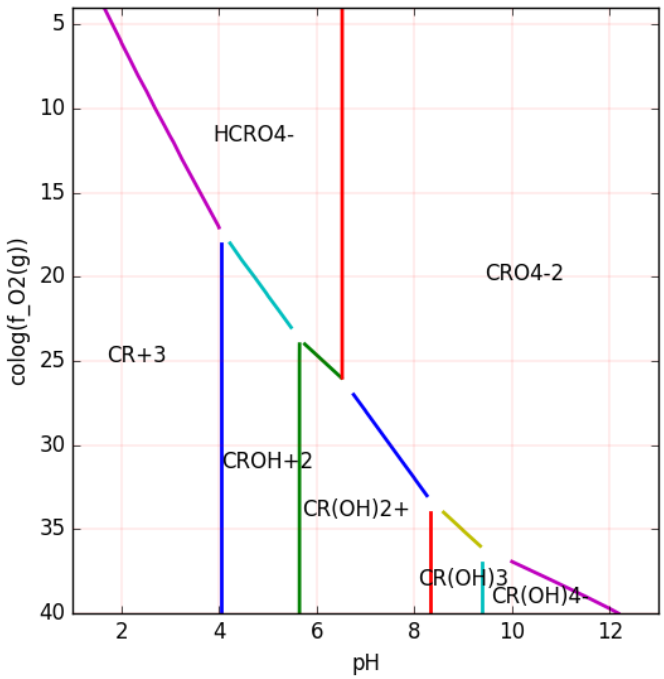
$2\ \mu(H_2O) = \mu(O_2) + 4\ \mu(H^+) + 4\ \mu(e^-)$

$2\ \mu°(H_2O)/RT/\ln 10 + 2\ \mathrm{Log}[H_2O] = \mu°(O_2)/RT/\ln 10 + \mathrm{Log}[O_2] - 4\ (pH + pe)$

../..

Autre exemple: le Chrome

script: map1b_cr_ox.inn (à vérifier)



A noter que, si on part d'une base de données telle qu'associée à PHREEQC, il faut, pour une utilisation par Arxim, commencer par construire une base de logK de toutes les espèces à partir d'un jeu unique d'espèces primaires, alors que les bases phreeqc, llnl, etc donnent, pour prendre l'exemple du chrome, les logK des réactions de formation des espèces secondaires Cr(VI) en fonction de CrO_4^{-2}, et ceux des espèces Cr(III) en fonction de Cr^{+3}. Il faut, pour arxim, choisir par exemple CrO_4^{-2} en primaire et, tenant compte du logK de la réaction $CrO_4^{-2} \rightarrow Cr^{+3}$, recalculer les logK des espèces Cr(III) en fonction de CrO_4^{-2} ...

diagrammes de phase en température – pression

> **script python**: test/script/**arxim_map_tp.py**

Les principales différences par rapport au cas précédents concernent le mode de calcul arxim, qui est GEM et non SPC, et les indices utilisés pour ré-écrire le bloc SYSTEM en fonction des valeurs des paramètres TDGC et PBAR.

Le script python **arxim_map_tp.py** fait appel au script arxim suivant

inn/map3a_tp.inn:

```
INCLUDE dtb\elements.dtb
INCLUDE dtb\hsvthr_thcalc_db55_min.tab
INCLUDE inn/map3a_tp.include
```

qui fait appel au fichier include pour définir le système:

```
SYSTEM.GEM
  TDGC  900
  PBAR  6500
  SIO2   SiO2  1.0
  AL2O3  Al2O3 1.0
END
```

remarque: dans ce fichier SIO2 et AL2O3 sont les noms des constituants (on aurait pu écrire 'silice' et 'alumine') alors que SiO2 et Al2O3 décrivent les stœchiométries

(et sont donc "case-sensitive").

Du fait des différences de format entre les blocs SYSTEM pour système aqueux et les blocs SYSTEM.GEM pour GEM, les valeurs des index sont différentes:

```
iKeyword= 0
iValue=   1
```

mais n'ont pas à être modifiées par l'utilisateur. De même, les instructions

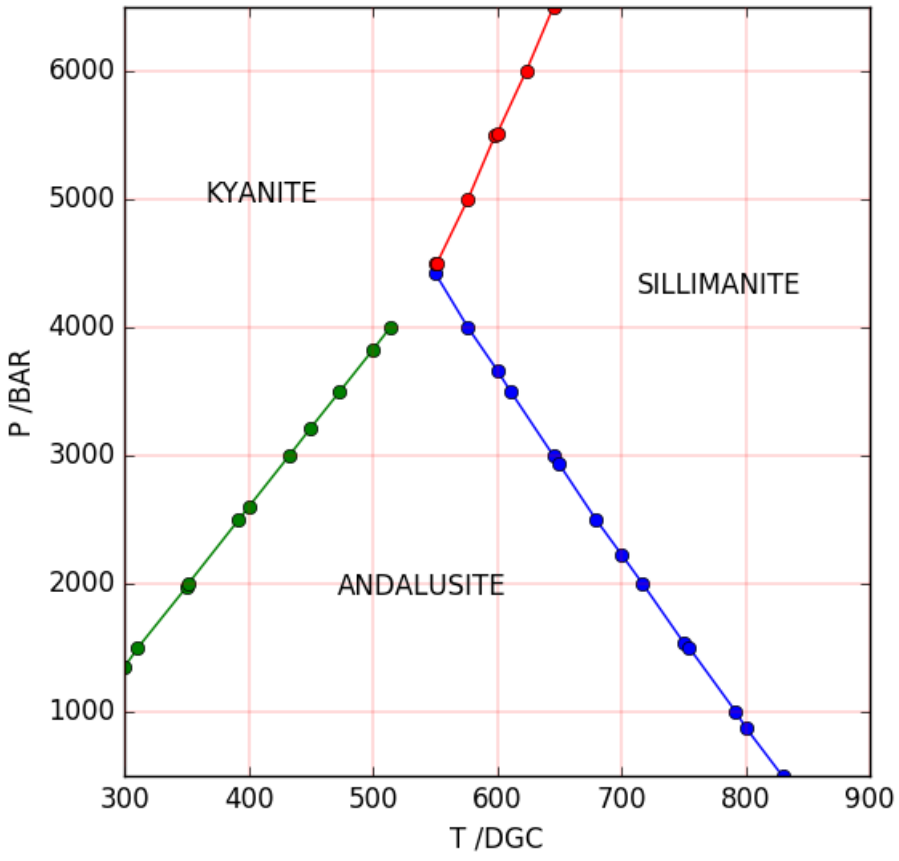
```
lisX= ["TDGC"]
lisY= ["PBAR"]
```

ne vont pas changer.

L'utilisateur devra seulement définir le domaine T,P à explorer, la dimension de la maille (Xdelta, Ydelta) et la précision de l'affinage (Xtol,Ytol):

```
Xmin,Xmax,Xdelta,Xtol= 300., 900.,   50.,  5.
Ymin,Ymax,Ydelta,Ytol= 500., 6500., 500., 10.
```

Le résultat (figure ci-dessous) montre qu'ici aussi on peut avoir besoin d'une procédure additionnelle de "prolongement des lignes", comme par exemple celle utilisée par C. de Capitani dans DOMINO.



diagrammes de phase en potentiels d'espèces mobiles

Pour la construction de diagrammes de phases en fonction des activités de deux constituants mobiles (équivalents à des "diagrammes en potentiels"), on ne modifie pas le système, mais les données thermodynamiques, c'est à dire un bloc SPECIES. A la différence du bloc SYSTEM, qui n'est lu qu'une fois, dans un run arxim, pour un calcul SPC, EQU ou GEM, les données thermodynamiques peuvent être réparties sur plusieurs blocs SPECIES. On peut donc mettre dans un bloc SPECIES spécifique les espèces dont les paramètres doivent être modifiés, les données constantes étant dans d'autres blocs SPECIES.

On a d'abord essayé de calculer ce type de diagramme avec une approche LMA, en mode EQU, avec des contraintes de type MOBILE sur les axes x et y, mais il est apparu que la minimisation globale (mode GEM) permettait d'explorer un domaine plus large d'activités des constituants mobiles.

On fait donc appel à la fonction arxim GEM, alors même que ce mode, à la différence de SPC ou EQU, ne prévoit pas la définition de constituants mobiles au niveau du bloc SYSTEM. Aussi, pour fixer l'activité d'un constituant à un niveau donné, on définit une phase fictive qui en a la stœchiométrie et l'on met un excès de ce constituant dans le système. D'autre part, au lieu de modifier le bloc SYSTEM, ce sont les données thermodynamiques de ces phases mobiles (fictives) que l'on modifie.

Dans le cas test (fichiers *map2a_activ.inn* et *map2b_activ.inn*), qui concerne le système classique KSAOH (K_2O-SiO_2-Al_2O_3-H_2O), en présence de H_2O en excès, le constituant Al_2O_3 étant inerte et SiO_2 et KOH mobiles, on a d'une part le système suivant, défini dans le script maître, et qui n'est donc pas modifié:

```
SYSTEM.GEM
  H2O    H2O   1000.
  KOH    KOH   1000.
  SIO2   SiO2  1000.
  AL     Al2O3 0.1
END
```

et d'autre part le bloc SPECIES suivant (fichier *include* appelé par le script arxim maître), qui va être modifié à chaque itération sur le potentiel de SiO2 ou sur celui de KOH:

```
SPECIES
  MIN  FICTIVE  SIO2_BUFF  SI(1)O(2)     2700.  0.
  MIN  FICTIVE  KOH_BUFF   K(1)O(1)H(1)  2700.  0.
END
```

Ce fichier *include* est modifié par le script python au niveau de la dernière valeur de la ligne SIO2_BUFF ou KOH_BUFF, qui donne le logK (décimal) de la réaction de formation de l'espèce (en l'occurrence une phase pure) en question.

Contrairement à ce qu'affirment certains inconditionnels de l'approche GEM (par exemple Kulik, qui gère le programme GEM-PSI de Karpov), celle ci ne demande pas nécessairement que l'on dispose d'une base classique de type "HSV" des énergies libres de formation des espèces à partir des éléments, elle peut aussi bien se pratiquer avec les bases en logK d'usage courant dans les approches LMA (law of mass action). Une base de logK de réactions entre espèces peut en effet toujours se transposer en une base d'énergies de formation (au facteur RT près) à partir d'un jeu d'espèces choisies comme primaires.

L'utilisation d'une base en logK a l'avantage, pour les diagrammes qui nous intéressent, de "graduer" directement les axes des potentiels des constituants mobiles en logs d'activité. Par contre, si on part d'une base "HSV", on doit commencer par construire une base de logK adéquate avant de lancer le script de

construction du diagramme.

Dans le cas test sur le système KSAOH, la base de données a été construite à partir de la base logK llnl.dat de Phreeqc 3.37 avec H_2O, H^+, K^+, Al^{+3}, SiO_2(aq) comme espèces primaires. En conséquence de quoi, si le logK de l'espèce fictive KOH_BUFFER est à une valeur X, l'équilibre du système avec la phase pure KOH impose la contrainte suivante sur les activités des espèces (les activités sont notées par des []):

```
log[KOH] - { log[K+] + log[H2O] – log[H+] } = X
```

c'est à dire, les activités [KOH] et [H_2O] valant 1,

```
colog([K+]/[H+])= X
```

De même, le logK de l'espèce fictive SiO2_BUFFER permet de contrôler l'activité de SiO_2(aq) dans le système.

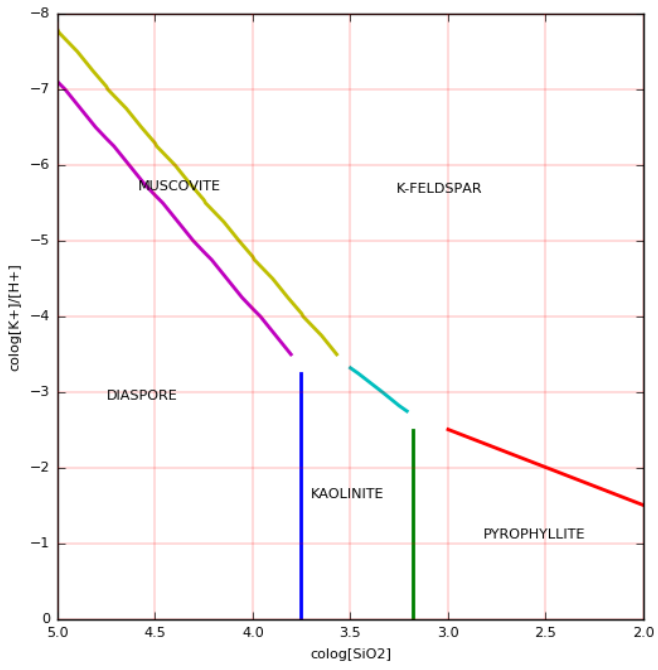
```
script python: test/script/arxim_map_activity.py
```

Ce script fait appel au script arxim *inn/map2b_activ.inn*, qui contient la description du système, avec H2O, KOH et SiO2 en large excès par rapport à Al2O3.

```
SYSTEM.GEM
  TDGC   25.
  H2O    H2O   1000.
  KOH    KOH   1000.
  SIO2   SiO2  1000.
  AL     Al2O3 0.1
END

INCLUDE  inn\map2b_activ.include
```

Il comprend aussi un bloc SPECIES de base de données fixes (non reproduite ici) et un appel au bloc de base de données variable *inn/map2b_activ.include*.

Le résultat du calcul est en accord avec les diagrammes trouvé dans la littérature pour ce système très souvent représenté.



25°C, base llnl tirée de phreeqc-337

On a noté sur le diagramme uniquement les phases qui distinguent les différents domaines, sachant que trois autres phases sont présentes partout: H_2O, et les deux phases tampons SiO_2_Buff et KOH_Buff.

Sur ce cas très simple, les solutions analytiques se calculent facilement et permettent de valider l'approche numérique programmée ici. Par exemple, pour localiser la limite diaspore – kaolinite, on sait qu'elle correspond à un équilibre entre ces deux phases, donc à ΔG=0 pour la réaction

$Si_2O_5Al_2(OH)_4$ (kaol) + H_2O (l) ↔ 2 $Al(OH)_3$ (diasp) + 2 SiO_2 (aq)

ΔG=0 :: μ°(kaol) + μ°(H_2O) = 2 μ°(diasp) + 2 { μ°(SiO_2) + RT ln[SiO_2(aq)] }

soit, en divisant les μ par - RT .ln10, et notant logK= - μ°/ RT / ln10

logK(kaol) + logK(H2O) = 2 logK(diasp) + 2 { logK(SiO_2) – log[SiO_2(aq)] }

Dans notre base de données, à 25°C

logK(H_2O) = 0

logK(SiO_2) = 0

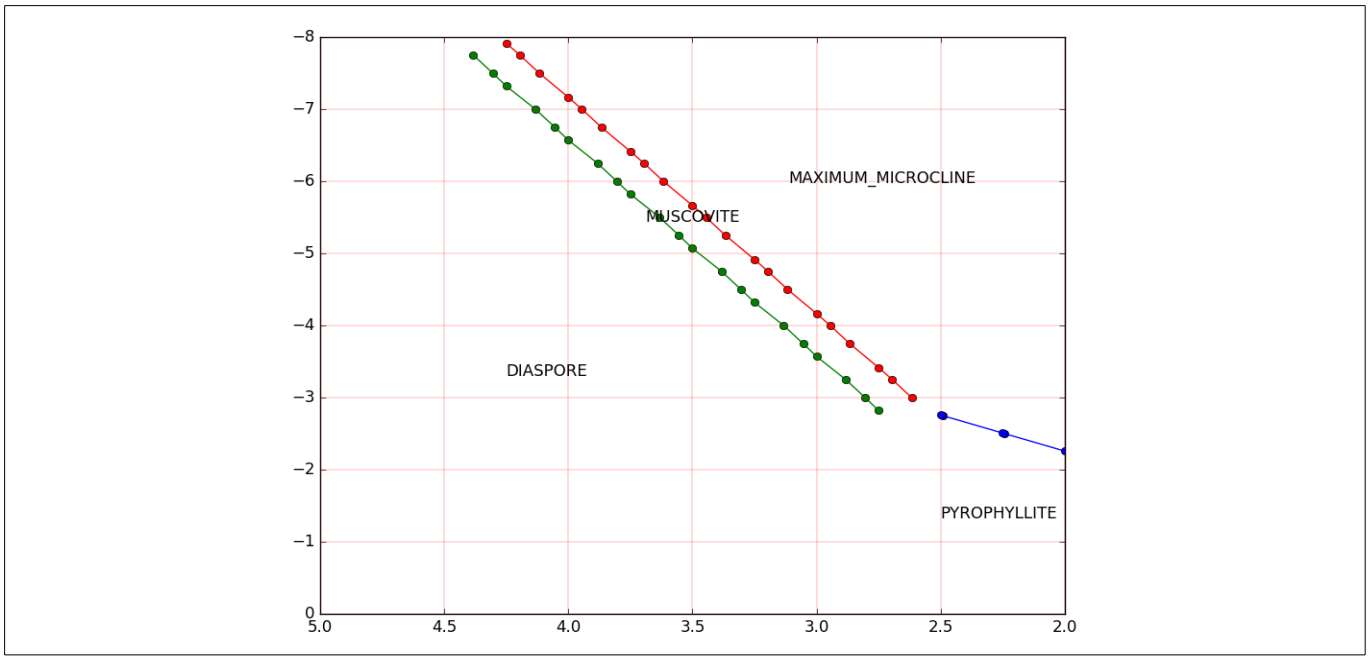
logK(kaol) = -6.81

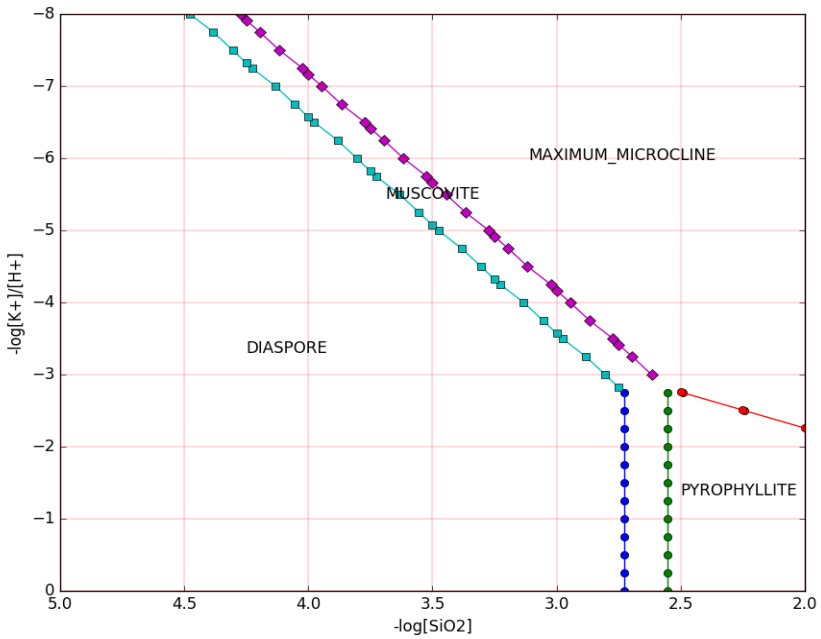
logK(diasp) = -7.16 (valeur la + élevée des différents Al-hydroxydes)

l'équilibre diaspore – kaolinite est donc réalisé pour

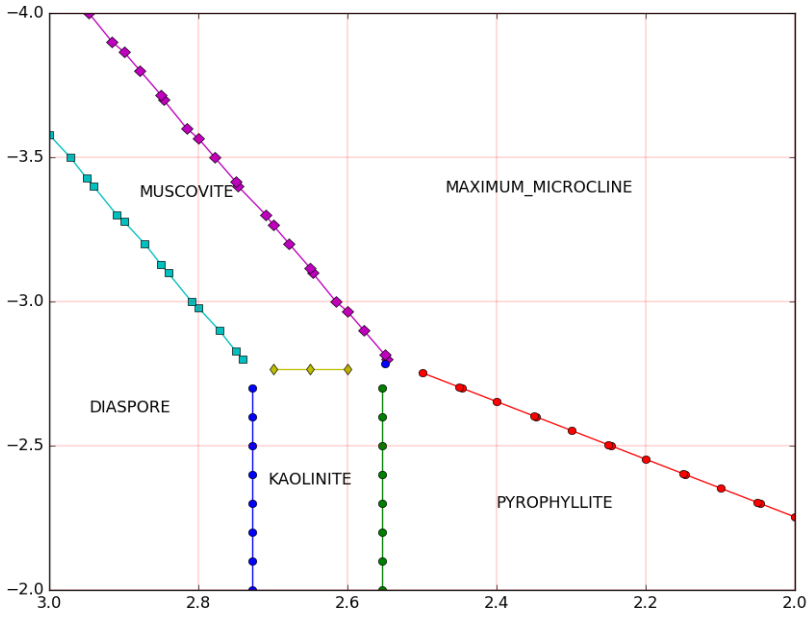
– log[SiO_2(aq)] = -6.81 /2 - (-7.16) = 3.75

A 100°C (à modifier dans *inn/map2b_activ.inn*), les champs de stabilité se déplacent, et celui de la kaolinite se réduit, le même script donne alors un diagramme incomplet quand on se contente d'un raffinement du premier ordre (figure du haut ci-dessous). Un raffinement du second ordre permet de retrouver le champ de la kaolinite (figure du bas), mais le nom du champ n'apparaît pas car ceux ci sont positionnés grâce au maillage initial (on calcule le centre de masse des points du maillage qui appartiennent à un domaine donné).



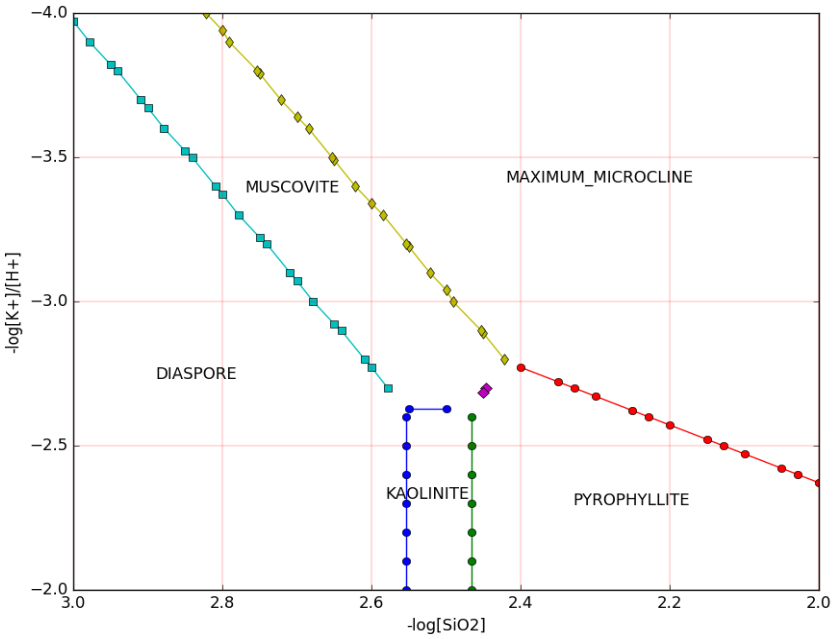


On peut "récupérer" le champ kaolinite dès le maillage initial en ajustant les valeurs de départ de Xmin, Xmax, etc:



à 100°C

A 120°C, le champ de la kaolinite se réduit encore et ne sépare plus complètement les champs de Muscovite et Pyrophyllite:



120°C

exemple de script pour générer, à partir de bases HSV, une base logK en vue de construire un diagramme en activités de SiO2 et KOH:

```
TEST
  COMPUTE SPC
  COMPUTE Q
END
CONDITIONS
  TITLE   build logK dtb
  OUTPUT  out\map2a !--> donnera un fichier out/map2a_logk_prim.dtb
  DEBUG   3
END
SYSTEM
  TDGC   25.
  SI  MOLE   SiO2,aq
  AL  MOLE   Al+3
  NA  MOLE   NA+
  K   MOLE   K+
  CA  MOLE   CA+2
  MG  MOLE   MG+2
  FE  MOLE   FE+2
  CL  MOLE   CL-
END

TP.TABLE  ! classical EQ3/6 series
TdgC  0.01 25.0 60.0 100.0 150.0 200.0 250.0 300.000
Pbar  1.01 1.01 1.01 1.01  4.757 15.53 39.73 85.8380
END TP.TABLE

!-------------------------------------------------------------database
INCLUDE  dtb\elements.dtb
INCLUDE  dtb\hsvhkf_aqu.dtb
INCLUDE  dtb\hsvthr_min.dtb
```

Réaliser une spéciation sur ce système produira dans *out/* le fichier

map2a_logk_prim.dtb qui donne les logK de formation des différents minéraux à partir des espèces aqueuses choisies dans SYSTEM (H_2O, K^+, Na^+, etc).

diagrammes en composition vs température

> exemples de scripts: **arxim_map_xt_fm.py**, **arxim_map_xt_fsp.py**

Il peut être intéressant de faire varier suivant l'axe x ou y plus d'un paramètre, qu'il s'agisse de composition ou de température / pression. C'est le cas des diagrammes X-T qui permettent d'analyser l'effet d'un paramètre de composition, par exemple le rapport Fe/(Fe+Mg) (ou Na/(Na+K), CO_2/(CO_2+H_2O), etc) sur la stabilité des assemblages de phases.

On est alors amené à faire varier de manière concomitante deux paramètres sur le même axe. Aussi, plutôt que de définir deux listes de valeurs x (un 'linspace' pour Fe et un autre pour Mg), on définit une liste pour Fe et une fonction pour Mg:

```
Xlis= ["FEO","MGO"]
Xmin,Xmax,Xdelta,Xtol=  0.05, 0.95, 0.05, 0.01
def Xfunc(x): return 1. - x
```

On fera ainsi varier simultanément Fe de 0.05 à 0.95 et Mg de 0.95 à 0.5, et la somme Fe+Mg restera égale à 1, non seulement dans le calcul de la grille initiale, mais aussi lors du raffinement, la fonction Xfunc étant également prise en compte par la fonction de raffinement.

De même, plutôt qu'un diagramme X-T, il peut être intéressant d'étudier l'évolution des paragénèses le long d'un chemin T-P:

```
Ylis= ["TDGC","PBAR"]
Ymin,Ymax,Ydelta,Ytol=  450.,  600.,  20.,   5.
def Yfunc(y): return 3000. + (y-400.)*10.
```

(pour implémenter un diagramme isobare, il suffira de définir *Yfunc* comme fonction constante)

--

```
#-------------------------------------------------------y-loop
for y in Yser:
  include_modify(lisY,y) #---------modify the include file for y
  #-------------------------------------------------------x-loop
  for x in Xser:
    include_modify(lisX,x) #-------modify the include file for x
    OK= arxim_ok(sArximCommand) #------------------execute arxim
    if OK:
      phase= arxim_result(fResult) #-----------read arxim result
      if not phase in lis_phase:
        lis_phase.append(phase)
      tab_phase[iX,iY]= j
      tab_x[iX,iY]= x
      tab_y[iX,iY]= y
```