Épreuve orale de « *Mathématiques et algorithmique* » de la Banque PT – Rapport 2017

Les futurs candidats trouveront dans ce rapport des remarques et des conseils qui pourraient leur être utiles pour leur futur passage. Ce rapport n'est pas exhaustif et ne met l'accent que sur quelques points jugés importants par l'équipe d'interrogateurs de cet oral. Nous suggérons aux futurs candidats de consulter le **site de la Banque PT**, où ils pourront trouver les mémentos, disponibles lors de l'oral, les exercices types d'informatique, ainsi que les rapports des années antérieures comportant en annexe certains exercices d'algorithmique et de simulation numérique posés en 2015 et 2016.

1 – Objectifs

Le but d'une telle épreuve est d'abord de contrôler l'assimilation des connaissances des programmes de mathématiques et d'informatique (items 2, 3 et 5) de toute la filière (première et deuxième années), sans oublier celle des connaissances de base du programme des classes du lycée (seconde, première, terminale).

Cette épreuve permet aussi d'examiner :

- l'aptitude du candidat à lire attentivement un sujet et à répondre à la question posée ;
- sa capacité d'initiative ;
- son aisance à exposer clairement ses idées avec un vocabulaire précis ;
- sa réactivité et son aptitude à communiquer dans un dialogue avec l'interrogateur ;
- son aptitude à mettre en œuvre ses connaissances et son savoir-faire pour résoudre un problème (par la réflexion et non par la mémorisation de solutions toutes faites) ;
- sa maîtrise des algorithmes et manipulations de base, des calculs sur des nombres entiers, décimaux ou complexes, et du langage de programmation pour mettre en œuvre une solution informatique ;
- sa faculté à critiquer, éventuellement, les résultats obtenus et à changer de méthode en cas de besoin.

2 – Modalités de cette épreuve

La durée de cet oral de « *Mathématiques et algorithmique* » est de 1 heure, préparation incluse.

Il comporte deux exercices de durées comparables :

- l'un porte sur le programme de mathématiques des deux années de la filière PTSI/PT (algèbre, analyse, géométrie et probabilités) et se déroule au tableau ;
- l'autre exercice porte sur les items 2, 3 et 5 du programme d'informatique et se déroule sur ordinateur. Pour ce deuxième exercice, les candidats disposent d'un ordinateur dans lequel sont installés *Python* 3.5 et ses principales bibliothèques (dont **`numpy`**, **`scipy`**, **`matplotlib`**, **`random`**, aides incluses) [1], des mémentos plastifiés et en couleurs au format A3, rendus publics bien avant l'oral, et de feuilles de brouillon, qu'il ne faut pas hésiter à utiliser. L'environnement de développement est *IDLE*, comme annoncé depuis 2014, muni de l'extension *IDLEX* qui permet notamment d'afficher plus clairement les numéros de ligne, de faire exécuter une partie d'un programme seulement (F9 au lieu de F5), ou de rappeler dans la console une commande déjà saisie (flèches montante et descendante). Quelques candidats ont avoué avoir préparé l'oral avec *Spyder* ou *Pyzo*, ce qui est un peu surprenant. Nous ne pouvons que conseiller de se placer dans les conditions de passage de l'oral tout au long des deux années de préparation.

1. *Scilab* 5.5 est également installé mais, depuis 2015, aucun candidat n'a demandé à programmer en *Scilab*.

3 – Organisation

Cette dernière session s'est déroulée dans des conditions identiques aux sessions précédentes. Comme les autres années, elle a eu lieu dans les locaux de « *Arts et Métiers ParisTech* », 155 boulevard de l'Hôpital à Paris (13^{e}). En raison de l'état d'urgence, il n'a pas été possible cette année d'accueillir de futurs candidats ou des enseignants de classes préparatoires, hormis deux représentants de l'Union des Professeurs de classes préparatoires Scientifiques.

4 – Conseils généraux

Lors d'une épreuve orale, le candidat doit être particulièrement attentif :

- bien lire le sujet et bien écouter une question permet de répondre à la question effectivement posée ;
- écouter les consignes de l'interrogateur est en général utile ;
- lorsqu'une indication est donnée pour aider le candidat, il faut savoir l'écouter et réagir à celle-ci.

Ces capacités d'attention, d'écoute et de réaction sont des éléments d'évaluation. De manière générale, la passivité, l'attentisme ou l'obstination dans une voie infructueuse sont déconseillés lors de l'oral.

Les exercices peuvent être de longueurs variables. L'objectif poursuivi est l'évaluation par l'interrogateur des capacités de chaque candidat grâce à l'exercice proposé, et non pas que le candidat termine nécessairement l'exercice.

L'oral, contrairement à une « *colle* », ne sert qu'à évaluer les capacités du candidat et non plus à participer à sa formation ; des indications seront en général données par l'interrogateur si le candidat reste bloqué trop longtemps, ou si celui-ci demande de l'aide par des questions dont il reconnaît implicitement ignorer la réponse (exemples : « *Est-ce que je peux utiliser tel théorème ?* », ou « *Pourquoi la figure ne s'affiche-t-elle pas ?* »).

Quelques détails utiles en mathématiques comme en informatique :

- la correspondance entre un point du plan et son affixe, ainsi que les interprétations géométriques du module, de l'argument, des parties réelles et imaginaires, du conjugué d'un nombre complexe sont supposées maîtrisées ;
- de même pour des manipulations géométriques de base comme le calcul des coordonnées du milieu d'un segment, de la longueur de ce segment (en repère orthonormé), des coordonnées des sommets d'un polygone usuel – en vue par exemple de faire tracer les côtés de ce polygone à l'écran –, de l'aire de polygones usuels (triangle, trapèze, carré, rectangle), ou de la hauteur d'un triangle équilatéral en fonction de la longueur de son côté ;
- il peut également être bon de connaître l'expression du coefficient binomial $\binom{n}{k}$ sous la forme $\dfrac{n \times (n-1) \times \cdots \times (n-k+1)}{k \times (k-1) \times \cdots \times 1}$, et pas seulement sa définition avec des factorielles.

5 – Conseils pour l'exercice de mathématiques

5.1 – Généralités

- L'oral n'est pas un écrit sur tableau ; les justifications et commentaires doivent être donnés au moment où l'on est interrogé ; le temps étant limité, il est inutile d'écrire de longues phrases, notamment pour justifier une linéarité ou une continuité triviales.
- Le candidat doit être précis dans ses propos, et, en particulier lorsqu'il énonce une définition, une propriété ou un théorème au programme de mathématiques, il doit énoncer l'ensemble des hypothèses sans en oublier ; le jury attend d'un candidat qu'il connaisse les résultats de cours.

- On attend également d'un candidat qu'il maîtrise les techniques de calcul en connaissant les concepts sous-jacents ; par exemple, maîtriser le procédé de calcul puis de recherche des racines du polynôme caractéristique ne dispense pas de connaître les définitions de valeur propre et de sous-espace propre ; lorsque plusieurs procédés de calcul sont possibles, par exemple pour la résolution d'un système linéaire ou la détermination du rang d'une matrice (méthode du pivot, substitution, combinaisons linéaires, etc.), le candidat peut utiliser celui qu'il préfère à condition d'être efficace.
- Les candidats doivent s'attendre à être interrogés sur la nature des objets qu'ils manipulent ; ils doivent pouvoir dire s'ils manipulent un nombre, une fonction, un vecteur ; par exemple, il n'est pas acceptable à ce niveau de confondre aire et primitive.
- Il est bon de connaître et de savoir justifier succinctement certaines inégalités usuelles comme : $\ln(x) \leqslant x-1$ sur $\mathbb{R}_+^*$, $1+x \leqslant \exp(x)$ sur $\mathbb{R}$, $|\sin(x)| \leqslant |x|$ sur $\mathbb{R}$, $x \leqslant \tan(x)$ sur $[0, \pi/2[$, etc.

5.2 – Polynômes et nombres complexes

- La manipulation des nombres complexes pose trop souvent des difficultés ; la conjugaison et ses propriétés ne sont pas toujours maîtrisées.
- La résolution dans $\mathbb{C}$ d'équations polynomiales n'est pas toujours maîtrisée.
- Les racines n-ièmes de l'unité et leurs propriétés, notamment leur somme, doivent être connues.
- De même pour les identités et factorisations remarquables comme : $x^2+2\,x+1$, x^n-1, $x^{2\,n+1}+1$, etc.
- La confusion entre « *polynôme scindé* » et « *polynôme scindé à racines simples* » est hélas trop répandue, révélant un manque de précision du langage ; cette confusion est bien évidemment problématique lorsqu'on étudie le polynôme caractéristique d'une matrice carrée.

5.3 – Algèbre linéaire

- En algèbre comme ailleurs, on doit veiller à utiliser un vocabulaire précis ; il a trop souvent été constaté une confusion entre « *matrice symétrique* » et « *matrice de symétrie* », entre « *endomorphisme symétrique* » et « *isométrie* », entre le sous-espace vectoriel $\{\mathbf{0}\}$ et l'ensemble vide $\emptyset$; d'autre part, caractériser une matrice *orthogonale* ne se limite pas au simple calcul de son déterminant, contrairement au cas d'une matrice *inversible*.
- Les liens entre les notions de valeur propre, de rang, de noyau, gagneraient en général à être mieux connus ; par exemple, les équivalences entre $\det(A) \neq 0$ et $\ker(A) = \{\mathbf{0}\}$, entre $\dim(\ker(A)) \geqslant 1$ et « 0 *est valeur propre de* A », entre « le vecteur non nul $\mathbf{u}$ est invariant par l'endomorphisme f » et « $\mathbf{u}$ *est vecteur propre de* f *pour la valeur propre* 1 ».
- On peut déplorer que l'interprétation géométrique des symétries et projections, et de leurs sous-espaces propres, soit négligée au profit d'une simple propriété opérationnelle sur $f \circ f$.
- Pour montrer qu'une famille est libre, penser aux lignes (ou colonnes) échelonnées pour les matrices, ou aux degrés échelonnés pour les polynômes.

5.4 – Intégration

- Lorsqu'on étudie l'intégrabilité d'une fonction, penser à regarder d'abord si celle-ci est continue.
- La recherche d'équivalents pour des fonctions (ou des suites) à valeurs positives, en vue d'étudier la convergence d'une intégrale ou d'une série, mériterait d'être davantage travaillée pendant l'année pour gagner du temps lors de l'oral.
- Dans l'étude de la convergence d'intégrales généralisées, il faut bien penser à énoncer les vérifications de base (continuité de la fonction, convergence de chaque intégrale dans le cas d'une décomposition en somme d'intégrales, etc.).

- La confusion entre primitive et intégrale a encore été trop souvent observée.

5.5 – Suites et séries

- Les séries géométriques doivent être parfaitement maîtrisées, ce qui est heureusement très souvent le cas.
- Des lacunes ont pu être observées, ce qui peut révéler un travail préalable à améliorer, sur la recherche de limites ou d'équivalents lorsque n tend vers l'infini de suites de terme général comme $\binom{n}{k} n^{-k}$, $(1-a/n)^{bn}$, ou $2\left(\sqrt{n}-\sqrt{n-1}\right)-1/\sqrt{n}$.

5.6 – Géométrie

- De nombreux sujets de géométrie sont posés, y compris parmi les exercices d'informatique. C'est une particularité de la filière PT. Il est plus que conseillé de faire un dessin lisible ; cela permet de mieux comprendre le sujet, et est très apprécié par les examinateurs.
- Les sujets de géométrie utilisent fréquemment la trigonométrie ; il convient donc de pouvoir donner rapidement les formules utiles à l'exercice, et aussi d'être capable d'étudier des fonctions trigonométriques simples, qui paramètrent souvent les courbes.
- Il faut surtout que les candidats, au lieu de se précipiter sur les calculs, mettent en place une démarche de résolution et annoncent à l'examinateur la liste des tâches pour arriver à la solution du problème posé.
- Trop peu de candidats ont réussi à mener à bien l'étude d'une courbe paramétrée.

5.7 – Fonctions de plusieurs variables et géométrie des courbes et surfaces

Liées aux notions de champs, de courbes et de surfaces, les fonctions de plusieurs variables, indispensables notamment en ingénierie mécanique, mériteraient davantage d'attention. En particulier, il est nécessaire de :

- savoir étudier leur continuité (ou plus généralement leur régularité $\mathcal{C}^1$) ;
- connaître la définition de ses dérivées partielles et savoir les calculer ;
- savoir utiliser la *règle de la chaîne* (dans le programme PT : « *Calcul des dérivées partielles d'ordres 1 et 2 de* $(u,v) \mapsto f(x(u,v),y(u,v))$ ») ;
- savoir déterminer la tangente et la normale à une courbe ainsi que le plan tangent à une surface, à partir d'équations cartésienne ou paramétrique. Sur ce dernier point, une amélioration a pu être observée en 2017 ; une confirmation de cette amélioration est espérée dans les années qui viennent.

5.8 – Équations différentielles linéaires et suites à récurrence linéaire forte

- Puisque dans ces deux situations, les candidats ont appris à rechercher les racines λ_i de l'équation caractéristique, les interrogateurs ont bien trop souvent observé, comme en 2016, une confusion finale troublante entre $\exp(\lambda_i\, t)$ et λ_i^n.

5.9 – Probabilités

- Les candidats sont en général bien préparés, avec une amélioration confirmée par rapport à celle constatée en 2016.
- On apprécie qu'un candidat justifie naturellement un résultat obtenu (probabilités totales, conditionnelles, etc) et donne des définitions correctes, notamment celle de l'indépendance de deux évènements, ou de deux variables aléatoires. Il est bien de prononcer le terme « *système complet d'évènements* » et encore mieux d'être en mesure de détailler de quoi il s'agit.

- On a cependant pu relever parfois des confusions entre : un évènement et sa probabilité ; évènements incompatibles et évènements indépendants ; probabilité de « A et B » et probabilité de « A sachant B » ; évènement et variable aléatoire, révélée par des notations fautives comme $\mathbb{P}(X)$ ou $\mathbb{E}(X > 0)$.
- Pour les variables aléatoires à valeurs dans $\mathbb{N}$, il peut être utile de connaître la formule de l'espérance du programme : $\mathbb{E}(X) = \sum_{n=1}^{\infty} \mathbb{P}(X \geqslant n) = \sum_{k=0}^{\infty} \mathbb{P}(X > k)$.

6 – Exercice d'algorithmique/simulation numérique

Les candidats ont en général été bien préparés pour cet oral.

L'ensemble des interrogateurs a pu constater une meilleure préparation moyenne des candidats sur les deux points faibles principaux soulevés dans le rapport 2016, à savoir la résolution numérique d'équations différentielles et l'extraction de données numériques d'un fichier ASCII. L'effort doit cependant être poursuivi, ainsi que sur d'autres points qui sont détaillés ci-dessous.

6.1 – Conseils généraux

- Si quelques lignes de code sont proposées à la compréhension, il est conseillé au candidat de taper ce code et de le comprendre en modifiant certains paramètres.
- Ne pas négliger les premières questions : elles contiennent le plus souvent des éléments de réponse pour la suite, voire des rappels.
- Ne pas hésiter à utiliser l'interpréteur pour effectuer des vérifications élémentaires et savoir utiliser les instructions `help` et `numpy.info` : il est normal de ne pas connaître toutes les fonctions apparaissant dans les exercices.
- Ne pas hésiter non plus à utiliser le mémento, surtout si le conseil en est donné par l'interrogateur.
- Il faut savoir mettre en œuvre une démarche en cas d'erreur : insérer des `print` pour contrôler pas à pas une exécution, etc. Il s'agit d'une compétence valorisée par le jury.
- Préférer une boucle `for` à un `while` quand le nombre d'itérations est connu à l'avance.
- Bien faire la distinction entre les entiers (type `int`) et les nombres à virgule flottante (type `float`), et maîtriser les conséquences induites. Une amélioration globale a été constatée en 2017 sur la connaissance des opérateurs `//` et `%` et la manipulation des complexes (notation `1j` et des écritures `z.real`, `z.imag` et `z.conjugate()`).
- La manipulation des chaînes de caractères fait aussi partie des capacités exigibles, et en particulier la connaissance des méthodes `split`, `strip`, `replace` qui peuvent être utiles pour la lecture de données structurées dans un fichier ASCII.
- Il n'est pas nécessaire de définir systématiquement une fonction pour chaque tâche demandée, et, plus généralement, il n'y a aucun style de programmation imposé ; le candidat est évalué sur la maîtrise des outils mis à sa disposition et non sur le respect dogmatique de telle ou telle règle ou interdiction le plus souvent arbitraire.
- En revanche, **une fonction doit toujours être testée**, soit dans l'éditeur (F5 ou F9), soit dans la console, comme cela est spécifié dans l'en-tête de chaque énoncé.
- L'utilisation de variables globales n'est pas conseillée, et encore moins exigée.

6.2 – Gestion du temps

Quelques candidats perdent un temps considérable avec des pratiques peu adaptées pour une épreuve de 30 minutes :

- **On ne mettra en œuvre « *à la main* » un algorithme élémentaire que s'il est explicitement demandé**. Sinon, il est bon de connaître et de savoir utiliser par exemple les fonctions intrinsèques `min`, `max`, `sum`, `sorted`, les méthodes `append`, `extend`, `sort`, `index` pour les listes, les méthodes `min`, `max`, `argmin`, `argmax`, `sum`, `mean`, `std`, `transpose`, `conjugate`, ... pour les tableaux `numpy.ndarray` (`T.real` et `T.imag` aussi pour un tableau de complexes), ainsi que les techniques de *slicing* (`U[debut:fin:pas]` pour une liste ou un vecteur, `M[Ldeb:Lfin:dL,Cdeb:Cfin:dC]` pour une matrice, etc.).
- De trop nombreux « `for i in range(len(iterable)) :` » au lieu de simplement « `for e in iterable :` » font perdre du temps et de la lisibilité.
- L'opérateur booléen `not` est souvent méconnu alors qu'il peut s'avérer très utile, comme par exemple dans « `if e not in L : L.append(e)` ».
- Il a été observé cette année un abus de la méthode `append` dans une boucle pour créer des listes simples. Par exemple :

`L = []`
`for i in range(15) :`
`  L.append(i+2)`
	au lieu de	`L = list(range(2,17))`

`L = []`
`for i in range(4) :`
`  L.append(2*i)`
	au lieu de	`L = [0,2,4,6]`

`S = []`
`i = len(L)-3`
`while i >= 0 :`
`  S.append(L[i])`
`  i -= 2`
	au lieu de	`S = L[-3::-2]`

- Même si les listes en compréhension ne sont pas exigibles, leur utilisation maîtrisée permet de gagner en efficacité et en lisibilité.
- Ne pas hésiter non plus à réutiliser les fonctions créées dans les questions précédentes, ou même à créer si cela peut être utile de petites fonctions intermédiaires ; les exercices sont très souvent structurés dans cet esprit.
- L'écriture systématique de commentaires et d'en-têtes (`"docstring"`) pour les fonctions est déconseillée pour l'oral ; même si elle est légitimement préconisée en génie logiciel, elle fait perdre un temps précieux ; les explications peuvent être données oralement par le candidat.

6.3 – Algorithmique

- Les algorithmes du cours et leurs coûts de calcul doivent être connus (algorithmes de tri, méthodes par dichotomie, de Newton, d'Euler, des trapèzes, pivot de Gauss, algorithme d'orthonormalisation de Gram-Schmidt, algorithme d'Euclide, etc.). Leur connaissance est fréquemment évaluée.
- La distinction claire entre *algorithme récursif* et *algorithme itératif* doit être acquise ; dans l'écriture d'une fonction récursive, un soin particulier doit être porté à la condition d'arrêt.

6.4 – À propos des fonctions

Trop de candidats n'ont pas parfaitement assimilé le concept de fonction.

Voici quelques symptômes que l'on retrouve hélas trop souvent également parmi les étudiants de première année en école d'ingénieurs :

- Le nom de la fonction est réutilisé comme nom d'objet dans la définition de la fonction.
- En dépit du bon sens, on effectue de nombreux appels de la même fonction avec les mêmes arguments, sans se rendre compte du caractère catastrophique d'une telle démarche. En voici un exemple caricatural, où `f` désigne une fonction d'arguments a, b, c, qui renvoie une liste de couples :

```
X = []
Y = []
for i in range(len(f(2.3,-1.2,0.4))) :
    X.append(f(2.3,-1.2,0.4)[i][0])
    Y.append(f(2.3,-1.2,0.4)[i][1])
```

au lieu de, par exemple,

```
P = f(2.3,-1.2,0.4)
T = numpy.array(P)
X = T[:,0].tolist()
Y = T[:,1].tolist()
```

  Si, en plus, la fonction `f` effectue à chaque fois un tirage pseudo-aléatoire de points, le résultat obtenu, en plus de prendre beaucoup trop de temps, devient faux.

- Beaucoup de candidats éprouvent des difficultés lorsque l'argument de la fonction est un objet de structure un peu complexe, comme par exemple une liste de couples, chaque couple contenant lui-même une chaîne de caractères suivie d'un couple de coordonnées ; dans ce cas, de nombreux candidats veulent créer d'abord l'objet, dans ou avant la définition de la fonction, au lieu d'extraire de l'objet les données à utiliser comme suit :

```
def f(L) :
    labels,X,Y = [],[],[]
    for element in L :
        label,coordonnees = element
        labels.append(label)
        x,y = coordonnees
        X.append(x)
        Y.append(y)
    ...
```

  ou de façon plus compacte :

```
def f(L) :
    labels,X,Y = [],[],[]
    for (label, (x,y)) in L :
        labels.append(label)
        X.append(x)
        Y.append(y)
    ...
```

  ou encore, en utilisant des listes en compréhension (non exigibles) :

```
def f(L) :
    labels = [ e[0] for e in L ]
    X = [ e[1][0] for e in L ]
    Y = [ e[1][1] for e in L ]
    ...
```

  Notons que cette dernière difficulté est du même ordre que celle rencontrée lors de la lecture de données structurées dans un fichier ASCII.

6.5 – Tableaux, matrices et vecteurs

Il est conseillé de bien connaître les propriétés des listes d'une part, et des tableaux **ndarray** du module **numpy** d'autre part, en particulier ce qui les différencie, de façon à pouvoir choisir le type le mieux adapté au problème à résoudre.

Trop souvent, un candidat utilise systématiquement des listes alors que des tableaux numériques seraient plus adaptés au cas à traiter. Voici deux exemples aboutissant au tracé d'un graphique grâce au module **matplotlib.pyplot** nommé ici **plt** :

⋆ Exemple du tracé d'un nuage de points caractérisé par une liste **P** de couples de coordonnées :

```
X,Y = [],[]
for x,y in P :
    X.append(x)
    Y.append(y)
plt.plot( X, Y, "or" )
plt.show()
```

Utilisation de listes

```
T = numpy.array(P)
plt.plot( T[:,0], T[:,1], "or" )
plt.show()
```

Utilisation de tableaux ***numpy.ndarray***

⋆ Exemple du tracé de la courbe représentative de $t \mapsto \sin(t^2)\exp(-t/4)$ sur l'intervalle $[-2, 8]$, en utilisant la vectorisation des fonctions du module **numpy** :

```
import math as m
nb_points = 501
tmin, tmax = -2, 8
dt = (tmax-tmin)/(nb_points-1)
T = [ tmin + i*dt for i in range(nb_points) ]
V = [ m.sin(t**2)*m.exp(-0.25*t) for t in T ]
plt.plot( T, V, "m-")
plt.show()
```

Utilisation de listes

```
import numpy as np
T = numpy.linspace(-2,8,501)
V = np.sin(T**2)*np.exp(-0.25*T)
plt.plot( T, V, "m-")
plt.show()
```

Utilisation de tableaux ***numpy.ndarray***

Pour le calcul vectoriel et matriciel, il faut bien faire la distinction entre :

⋆ vecteur (tableau **numpy.ndarray** à un seul indice) et liste de nombres, et savoir passer de l'un à l'autre par la méthode **tolist** de **numpy.ndarray** et la fonction **numpy.array** ;

⋆ vecteur (tableau **numpy.ndarray** à un seul indice) et matrice-ligne/matrice-colonne (tableaux **numpy.ndarray** à deux indices dont l'un ne prend que la valeur nulle) ;

⋆ produit matriciel/scalaire **A.dot(U)** (ou **numpy.dot(A,U)**) et produit ***** terme-à-terme.

7 – Analyse des résultats

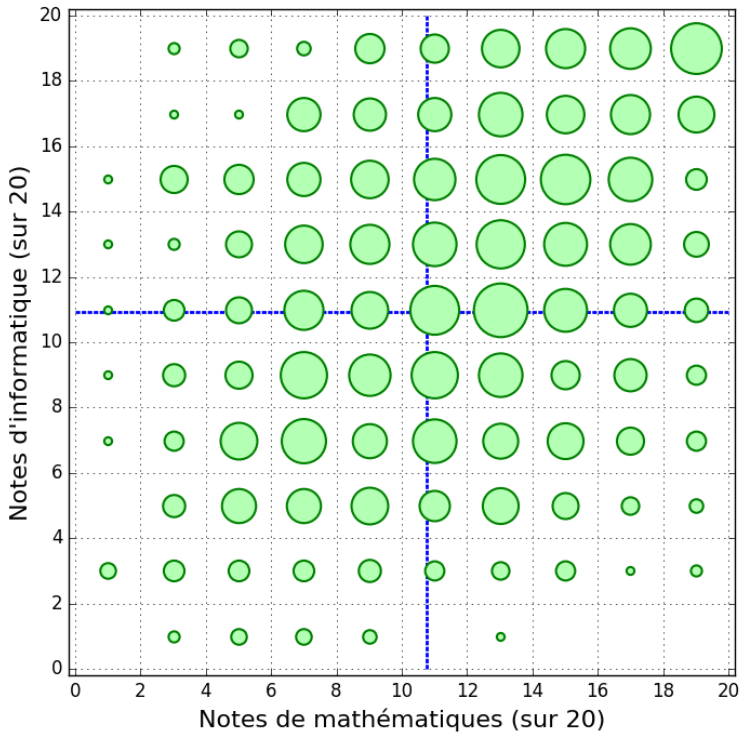
En 2017, 1475 candidats ont passé l'oral de « *Mathématiques et algorithmique* ». Chacun des 11 jours de l'oral, les 8 ou 9 jurys se sont efforcés de poser des exercices balayant l'ensemble du programme, tant en mathématiques qu'en algorithmique et simulation numérique.

Ainsi, 221 exercices différents d'analyse et de probabilités ont été proposés à 771 candidats contre 188 exercices différents de géométrie et d'algèbre proposés à 704 candidats.

157 exercices différents d'informatique à dominante algorithmique ont été posés à 803 candidats, contre 152 exercices à dominante « *simulation numérique* » pour 672 candidats.

Les statistiques sur les notes sont les suivantes [2] :

Oral 2017	**Note (sur 20)**	Math. (sur 10)	Algo. (sur 10)
Moyenne	**10,86**	5,39	5,47
Écart-type	**3,77**	2,22	2,38
Minimum	1	0	0
Maximum	20	10	10



Distribution des notes 2017

Éric Ducasse, Coordonnateur de l'épreuve orale de « *Mathématiques et algorithmique* » de la Banque PT, Le 19 juillet 2017.
eric.ducasse@ensam.eu

2. Rappelons que seule la note globale est communiquée au candidat.