Épreuve de mathématiques pratiques et informatique
Rapport du jury – Session 2015

21 septembre 2015

1 Modalités de l'épreuve

Un sujet d'oral est composé d'un unique exercice imposé au candidat, portant sur le programme de mathématiques et d'informatique de BCPST. Dans la mesure du possible, les sujets ont été choisis pour couvrir l'ensemble du programme des deux années (BCPST1 et BCPST2).

Les candidats disposent de 30 minutes de préparation dans une salle dédiée, où ils ont accès à un ordinateur sur lequel sont installés les logiciels Python 3 (Pyzo), Excel et Geogebra. (Conformément aux directives du concours, à titre exceptionnel, les candidats de la session 2015 avaient également à disposition les logiciels Scilab et Matlab).

Les candidats disposent d'une clé USB fournie par le concours au début de leur préparation, sur laquelle ils peuvent enregistrer leur travail.

A l'issue de la préparation, ils sont accompagnés dans une salle d'interrogation (5 minutes au maximum sont prévues pour la transition), où ils disposent d'environ 18-20 minutes pour exposer le résultat de leur travail de préparation et dialoguer avec l'examinateur. Dans cette salle de passage, ils disposent également d'un ordinateur connecté à un vidéoprojecteur, sur lequel ils peuvent exécuter les programmes sauvegardés sur la clé USB fournie.

A l'issue de cette première partie d'épreuve, les candidats sont accompagnés vers une deuxième salle, dans la mesure du possible voisine de la précédente, où ils présentent le résultat de leur projet informatique durant une durée de 18-20 minutes devant un second examinateur. Dans cette deuxième salle, les candidats disposent d'un ordinateur avec vidéoprojecteur sur lequel est affiché le dossier qu'ils ont au préalable envoyé au concours au début du mois de juin.

À la fin de chaque demi-journée, l'examinateur de la partie mathématiques et de la partie informatique de chaque jury confrontent leurs impressions sur les candidats qu'ils ont interrogés, en particulier sur leurs compétences en informatique. L'ensemble du jury a trouvé satisfaisante cette organisation avec des examinateurs dédiés à chaque partie, qui permet aux candidats d'être évalués par un spécialiste tant en mathématiques pratiques que pour son projet informatique.

2 Éléments statistiques

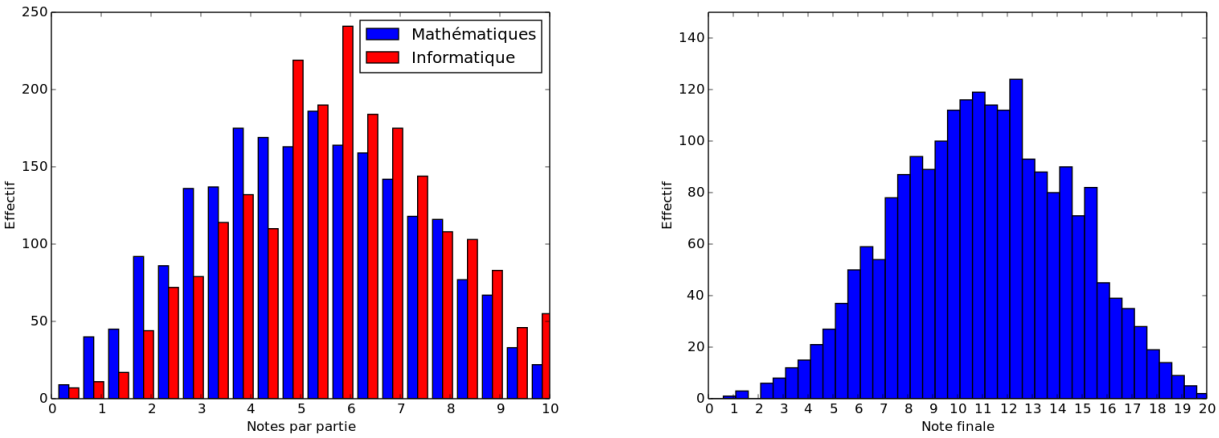
Un résumé statistique est fourni ci-dessous, pour chacune des parties de l'épreuve ainsi que pour la note finale (chaque partie était notée sur 10). On notera accessoirement que la corrélation entre la note d'un candidat sur la partie Mathématiques pratiques et sa note sur la partie Projet informatique est modeste ($r = 0{,}37$, $r^2 = 0{,}14$), ce qui n'est pas surprenant puisque les deux parties évaluent volontairement des compétences différentes.

Partie	Moyenne	Médiane	Écart-type	$0 \leq x < 2$	$2 \leq x < 4$	$4 \leq x < 6$	$6 \leq x < 8$	$8 \leq x \leq 10$
Mathématiques	5,29	5,5	2,14	4%	21%	32%	27%	15%
Informatique	5,86	6	2,03	2%	14%	30%	35%	18%

Résumé statistique par partie de l'épreuve

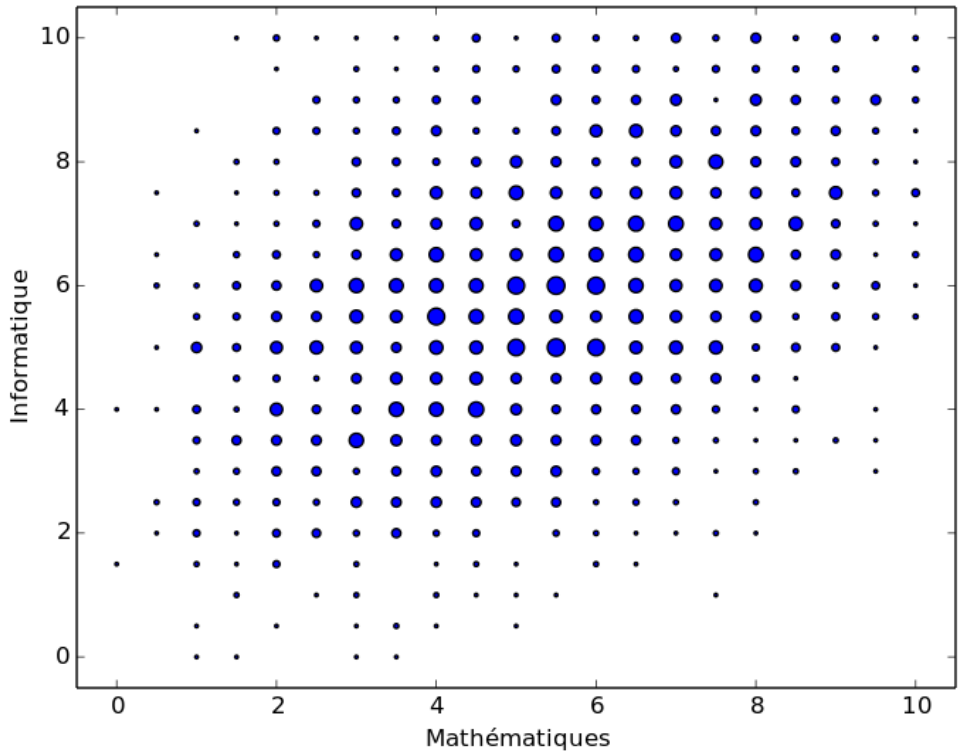
	Moyenne	Médiane	Écart-type	$0 \leq x < 4$	$4 \leq x < 8$	$8 \leq x < 12$	$12 \leq x < 16$	$16 \leq x \leq 20$
Note finale	11,15	11	3,47	1%	16%	39%	35%	9%

Résumé statistique général



Histogrammes des notes

Dans le graphique suivant, la taille du disque de centre (x, y) est proportionnelle au nombre de candidats ayant obtenu la note x à la partie Mathématiques pratiques et y à la partie Informatique. Comme dit plus haut, les notes sont assez faiblement corrélées ; on pourra quand même noter que les candidats ayant excellé en mathématiques ont presque tous obtenu des notes satisfaisantes pour leur projet (l'inverse est moins vrai).



3 Partie mathématiques pratiques

3.1 Ce qui est attendu des candidats en mathématiques pratiques

Dans un but d'équité, tous les candidats convoqués à une même heure, sont examinés sur le même sujet, quel que soit leur jury d'interrogation. Les examinateurs connaissent les énoncés sur lesquels porte l'interrogation, il est donc inutile de les leur rappeler.

Pendant la période d'interrogation, un dialogue entre le candidat et l'examinateur s'instaure. L'oral est un échange, n'est pas un écrit, et en aucun cas n'est une conférence du candidat face à un jury muet. Les questions du jury visent à évaluer les compétences du candidat, et cherchent avec bienveillance à l'aider pour compléter son argumentation ou se rendre compte d'une erreur. Elles ne constituent en aucun cas des pièges tendus. Le jury attend de la bonne volonté et une attitude ouverte pour répondre aux questions.

Il n'est pas indispensable d'avoir traité la totalité de l'exercice pour obtenir une excellente note. Il est préférable d'avoir mené un raisonnement rigoureux et argumenté, reposant sur des connaissances solides, plutôt que d'avoir donné tous les résultats (même justes), trop vite et sans explication réelle.

Au début de l'interrogation, le candidat est invité à préciser la liste des questions qu'il a eu le temps d'aborder pendant la préparation. Le jury tient à préciser que cette question n'a pas pour but d'évaluer a priori le candidat, mais permet de gérer au mieux le déroulement de l'oral, et éviter que certaines questions préparées par le candidat n'aient pas le temps d'être exposées.

Chaque sujet comporte au moins une question d'informatique et les examinateurs interrogent systématiquement sur au moins une de ces questions, qu'elle ait été préparée ou non par le candidat en amont. Le cas échéant, le jury peut poser une question élémentaire pour vérifier les compétences du candidat en informatique.

La durée de l'interrogation orale doit impérativement être respectée. Lorsque l'examinateur annonce la fin du temps réglementaire, il est attendu que le candidat efface efficacement le tableau et range rapidement ses affaires, afin de ne pas pénaliser le candidat suivant.

3.2 Remarques générales

- Les candidats semblent bien formés et se sont bien adaptés au nouveau format de l'épreuve. Nous tenons à féliciter les préparateurs qui ont su s'adapter aux nouvelles directives de l'oral.
- Le niveau général des candidats est plutôt bon. Peu de candidats sont en grande difficulté face aux notions du programme. La connaissance du cours est globalement satisfaisante. En revanche, certains candidats montrent de grosses lacunes en calcul et les bases du lycée ne semblent pas toujours acquises, surtout sur des notions élémentaires d'analyse (étude de fonctions, calculs de dérivées et primitives, connaissance des fonctions usuelles...).
- Plusieurs exercices invitent le candidat à conjecturer au préalable des résultats (par exemple à l'aide de l'informatique) puis à les démontrer. Nous regrettons que de nombreux candidats confondent alors les notions de conjecture et d'hypothèse pour répondre aux questions qui suivent.
- L'énoncé des définitions et des théorèmes au programme doit être précis, notamment sur le rappel des bonnes hypothèses. Le jury se réserve le droit de poser une question de cours à tout moment de l'interrogation s'il le juge nécessaire, pour s'assurer de la bonne compréhension du candidat.
- Le jury tient à encourager les candidats à émettre un avis critique sur leurs résultats lorsqu'ils sont clairement incohérents (tableau de variations en désaccord avec les limites, obtention d'une probabilité supérieure à 1...).
- Faire l'impasse sur une ou plusieurs parties du programme de mathématiques ou d'informatique peut avoir de lourdes conséquences sur la prestation du candidat et peut être très pénalisant pour son évaluation.

3.3 Remarques en algèbre

- La plupart des exercices proposés en algèbre étaient élémentaires. Cependant, le jury a regretté la mauvaise connaissance des candidats sur les notions de bases mises en jeu.
- L'étude du signe d'une expression ou la manipulation des inégalités restent un problème pour trop de candidats.
- La recherche des racines d'un trinôme du second degré peut encore poser de nombreux problèmes à certains candidats, surtout pour la recherche du signe du trinôme.

- Les candidats confondent régulièrement les propriétés des matrices (une matrice triangulaire n'est pas forcément inversible, une matrice inversible n'est pas forcément diagonalisable...).
- Nous avons constaté que les nouvelles notions du programme 2015 (produit scalaire, projection orthogonale,...) étaient parfois mieux maîtrisées que les résultats fondamentaux d'algèbre linéaire.
- Nous invitons les futurs candidats à bien maîtriser les exercices classiques d'algèbre linéaire, en particulier sur les fonctions polynomiales (factorisations, utilisation des degrés, dérivées successives) et sur la réduction des endomorphismes (bon emploi des éléments propres d'une matrice, confusions entre rang de A et rang de $A - \lambda I_n$).
- Il est rappelé que la notion de polynôme annulateur d'une matrice ou d'un endomorphisme n'est pas au programme en BCPST. Il est attendu des candidats qu'ils refassent le raisonnement qui conduit au résultat bien connu liant polynôme annulateur et valeurs propres.

3.4 Remarques en analyse

- Dériver ou « primitiver » une fonction n'est pas un acquis de tous les candidats. L'inversion entre ces deux notions est trop souvent rencontrée.
- La résolution des équations différentielles linéaires du premier ordre avec second membre n'est pas maîtrisée. Nombreux sont les candidats qui ne savent pas utiliser à bon escient la méthode de variation de la constante.
- Il est regrettable que peu de candidats soient à l'aise avec l'utilisation correcte des équivalents. Par exemple, le calcul de $\lim\limits_{n\to+\infty} \left(1+\frac{1}{n}\right)^n$ devrait être un acquis de tout candidat admissible, ce qui est loin d'être le cas et donne lieu à de nombreux résultats fantaisistes.
- Les fonctions usuelles, (fonction ln, exp, en particulier Arctan,...) sont particulièrement mal maîtrisées par certains candidats. Savoir dessiner schématiquement l'allure correcte de la courbe de l'une de ces fonctions est un attendu du jury et fait trop souvent défaut pour l'instant auprès des candidats.
- Les étudiants interrogés confondent parfois les notions relatives aux fonctions et aux suites, par exemple étude du signe de $f_{n+1}(x) - f_n(x)$ pour trouver les variations de la fonction f_n.
- Les principaux théorèmes d'analyse sont mal formulés (théorème de la bijection, théorème des accroissements finis).
- Les sommes de Riemann sont peu reconnues par les candidats, même lorsque l'examinateur guide énormément le candidat vers ce résultat.
- Il est rappelé que l'Inégalité des accroissements finis n'est pas au programme en BCPST, de même pour le théorème du point fixe pour l'étude des suites récurrentes. Il est attendu des candidats qu'ils refassent le raisonnement qui conduit à l'un de ces résultats.
- Nous rappelons que le jury peut interroger sur l'intégralité du programme. Il est regrettable que peu de candidats soient à l'aise avec les notions élémentaires concernant les fonctions de deux variables réelles.

3.5 Remarques en probabilités

- La manipulation d'événements et la traduction mathématique d'une situation sous la forme d'opérations sur des événements élémentaires restent un point délicat pour la plupart des candidats.
- Les formules principales de probabilités (formules des probabilités totales, formule des probabilités composées) sont souvent mal énoncées, voire par énoncées du tout. Par exemple, il est courant de voir des candidats faire une hypothèse d'indépendance pour justifier une multiplication de probabilités alors qu'il faudrait utiliser pour cela la formule des probabilités composées.
- De manière générale, la notion d'indépendance est confondue avec la notion d'incompatibilité, et est utilisée de manière abusive.
- Certains candidats montrent une méconnaissance des principales lois usuelles de probabilités, alors que ce sont des questions de cours élémentaires, ce qui les empêche d'avancer correctement dans les exercices proposés.
- En particulier, les élèves connaissent mal l'écriture correcte des densités et fonctions de répartition usuelles. Il est courant de voir comme fonction de répartition d'une loi exponentielle « $F(x) = 1 - e^{-\lambda x}$ » sans autre indication sur la variable x.
- Nous invitons les candidats étudiant une variable aléatoire (à densité ou non) à s'intéresser aux valeurs prises par celle-ci avant de démarrer toute étude. Cela peut permettre de simplifier les raisonnements, tant pour la loi, que pour la recherche des moments.

- Il n'est pas rare de voir un candidat confondre les notions d'événements et de variable aléatoire.
- L'utilisation de la formule du produit de convolution (systématiquement rappelée dans les énoncés) n'est pas un acquis de tout candidat, même dans des cas simples.
- L'utilisation des théorèmes limites en probabilités au programme (loi faible des grands nombres, théorème central limite) est mal maîtrisée par les candidats. Même s'ils en comprennent souvent l'idée générale, peu parviennent à formuler mathématiquement ces résultats.

3.6 Informatique

- Nous constatons avec plaisir que les candidats ont été bien préparés à l'utilisation des logiciels au programme pour accompagner l'étude de problèmes mathématiques. Nous tenons une nouvelle fois à féliciter leurs professeurs qui ne sont certainement pas étrangers au niveau de maîtrise constatée de ces nouvelles compétences.
- Nous rappelons que, lors du temps de préparation, le candidat ne doit pas forcément passer trop de temps à essayer de faire fonctionner ses scripts, mais plutôt se concentrer sur la structure générale des algorithmes mis en œuvre. Le fait d'avoir un programme qui ne fonctionne pas lors du passage n'est pas forcément pénalisé par le jury.
- Nous invitons les candidats à réfléchir sur le bon usage de la fonction `range` et notamment, en cas de doute, à faire des tests simples dans la console (en écrivant par exemple `list(range(0,2))`).
- Un attendu minimal et exigible des candidats est de savoir simuler des variables suivant les lois usuelles suivantes : uniforme, Bernoulli, binomiale, géométrique, exponentielle.
- Trop de candidats confondent le fait de simuler une variable aléatoire suivant une certaine loi, et le calcul des probabilités (ou de la densité) de la dite variable.
- Les candidats ne disposent pas de calculatrice lors de l'épreuve (préparation et passage), mais nous invitons les candidats à se servir des logiciels mis à leur disposition en cas de besoin de calcul numérique.
- Les candidats doivent savoir tracer la courbe d'une fonction, et pour cela utiliser le logiciel qui leur semble le plus adapté. Certains élèves se présentent en ayant utilisé plusieurs outils comme par exemple Python+Geogebra : ces efforts sont appréciés.
- Nous invitons les préparateurs à former leurs candidats à l'utilisation de la fonction « aide » des logiciels lorsqu'ils ne maîtrisent pas bien la syntaxe d'une fonction ou d'une bibliothèque.
- Les candidats maîtrisent cependant la plupart des questions classiques (calcul d'une somme ou d'un produit, raisonnement par dichotomie, suites récurrentes, recherche du min/max d'une liste, etc), ce qui est fort encourageant. Nous souhaiterions que les candidats soient également capables de mettre en œuvre une méthode numérique de résolution approchée d'une équation différentielle à l'aide de l'outil informatique.

3.7 Conclusion

Le niveau des candidats est très hétérogène, avec tous les intermédiaires entre le candidat brillant et le candidat n'ayant quasiment aucune connaissance ni compétence. L'examinateur s'attache à poser des questions adaptées pour évaluer le niveau du candidat de manière exacte. Le ressenti de certains candidats n'est pas toujours en adéquation avec la note obtenue. Tous les jurys s'efforcent d'être aimables et bienveillants, mais n'en ont pas moins pour mission d'évaluer et de classer les candidats.
L'ensemble des examinateurs a apprécié l'attitude respectueuse et courtoise de tous les candidats, ce qui a facilité le bon déroulement des oraux de cette session 2015.

Examinateurs

Pierre-Jean Aubry, Sébastien Besnard, Philippe Delbecque, Saïdia Fakhi-Souchu, Yoann Gelineau, Hervé Gras, Loïc Grenard, Marc Jalard, Kamal Marc Haidar, Francis Raccaglia, Robert Thai, Jean-Paul Truc, Raphaël Vinsu.

Experts et rapporteurs

Yoann Gelineau et Francis Raccaglia.

4 Partie informatique

4.1 Modalités de l'épreuve

La partie de l'épreuve consacrée au projet informatique durait vingt minutes : dix minutes (au maximum) de présentation, suivies d'un entretien. Le but de l'entretien est avant tout de juger de la maîtrise du projet par le candidat, mais également d'évaluer de manière plus générale ses compétences en informatique. Pour le premier point, le candidat doit être capable de répondre précisément aux questions qu'on peut lui poser sur le code : quels sont les arguments de cette fonction, que renvoie-t-elle si on l'appelle sur telle valeur… Pour évaluer le deuxième point, les examinateurs ont presque systématiquement demandé aux candidats d'écrire au tableau quelques lignes de code. Le plus souvent, ce code était basé sur leur projet : il pouvait par exemple s'agir d'écrire une version plus générale d'une fonction présente dans le projet ou d'écrire une boucle pour remplacer une série d'instructions. Dans certains cas, typiquement si le projet était très peu ambitieux et n'appelait pas de remarques particulières, les examinateurs ont pu poser un petit exercice indépendant, afin de voir si le candidat savait faire plus de choses que ne laissait supposer son projet.

4.1.1 Remarques générales

Comme souvent, la suite de ce rapport dresse surtout une liste des points sur lesquels les candidats doivent encore progresser. Cependant, l'impression globale du jury est tout à fait positive.

- La très grande majorité des candidats semble s'être investis dans leur projet, et avoir ce faisant acquis des compétences qui leur seront utiles tant dans la suite de leurs études que dans leur vie professionnelle.
- Les candidats extrêmement faibles (qui n'ont pas vraiment compris ce qu'était une variable ou une fonction) sont très rares.
- Plus fréquents sont les candidats incapables d'écrire une fonction élémentaire au tableau, mais comprenant visiblement comment marche leur programme (capables d'*expliquer* mais pas de *produire*). Ce n'est bien sûr pas satisfaisant, et témoigne sans doute d'une organisation défaillante du travail dans le groupe, mais savoir lire du code constitue déjà une compétence intéressante.
- Les présentations orales, malgré quelques défauts récurrents évoqués plus bas, étaient globalement de bonne qualité.
- L'organisation de l'épreuve a été jugée satisfaisante pour une première année, même si certains points de détail seront sans doute modifiés. En particulier, les candidats doivent s'attendre à voir la durée maximale de la présentation être réduite pour laisser plus de place à l'entretien (les modalités exactes seront précisées dans la notice du concours).
- Par rapport à l'ancienne épreuve d'option, le jury salue unanimement le choix d'un langage plus en adéquation avec l'utilisation qui en est faite. Les défauts et qualités des candidats et de leurs projets n'ont pas été radicalement modifiés, ce qui est en soi un progrès puisque les candidats les plus rétifs à l'informatique ne passaient auparavant pas l'épreuve.
- Il est cependant regrettable de constater que certains candidats arrivent avec un projet dont le code est extrêmement maladroit (et le plus souvent faux) alors que les dix minutes de l'entretien suffisent à l'améliorer considérablement. Ce temps aurait dû être pris avant la soumission du projet !
- Il est parfois rapidement apparent que certaines parties du code n'ont pas été écrites par le candidat (par exemple, un tri rapide tout droit sorti d'une correction de séance de Travaux Dirigés). Ce ne serait pas nécessairement problématique si les candidats en question maîtrisaient ce code, mais l'expérience semble montrer que c'est rarement le cas. Il vaut mieux alors que le candidat l'admette plutôt que de tenter désespérément de faire illusion.

4.2 Remarques sur la présentation

Lors de la phase de présentation du projet, le candidat dispose du diaporama qu'il a déposé préalablement par voie électronique et qui lui sert de support à sa présentation. Cette année, les quelques candidats n'ayant pas préparé un diaporama mais un rapport destiné à être imprimé (et donc illisible une fois projeté) n'ont pas été pénalisés, les instructions ayant été publiées très tardivement ; il n'en sera pas de même pour les prochaines sessions.

L'usage d'un chronomètre est autorisé durant cette phase, mais l'objectif n'est pas d'essayer de parler le plus vite possible en remplissant les dix minutes au détriment de l'intelligibilité. L'usage de notes, en revanche, est proscrit.

Une bonne présentation doit être synthétique. Le jury apprécie les candidats qui présentent l'enjeu de leur projet, les hypothèses qui ont été faites dans la modélisation choisie (le cas échéant), l'architecture générale du projet et une analyse des résultats et des perspectives ultérieures.

Certains écueils sont à éviter : une présentation de concepts biologiques durant cinq minutes, ou un cours de physique sont à proscrire. *A contrario*, certains candidats n'explicitent pas leur modèle, implémentent des jeux dont ils n'expliquent pas les règles, ou plus généralement ne fixent pas les contours de leur projet. Nous rappelons aux candidats que l'examinateur n'est pas censé interrompre le candidat durant cette phase, et que si les examinateurs ont pris connaissance du projet du candidat, ils n'ont pas passé une année à travailler sur le projet ; il faut donc éviter de se retrouver dans une situation où l'examinateur ne voit pas où le candidat veut en venir. Pour un jeu, demander à l'examinateur s'il connaît déjà les règles ou s'il souhaite qu'on les lui explique rapidement semble une idée raisonnable.

Il est fortement recommandé de commenter deux ou trois fonctions centrales durant cette phase de présentation. De préférence, le candidat choisira celles présentant des difficultés algorithmiques. Il est par contre inutile de lister l'ensemble des fonctions présentes dans le programme.

Une conclusion est évidemment la bienvenue, dans des proportions raisonnables : présenter les résultats de sa modélisation durant la moitié de l'oral est donc à éviter. Il peut être intéressant de proposer des améliorations possibles, mais seulement si le candidat y a effectivement réfléchi.

4.3 Thèmes et qualité des projets

Le thème du projet était complètement libre, ce qui permet en règle générale aux candidats de choisir un sujet qui les motive. Il est important de choisir un projet dont la difficulté soit en adéquation avec le niveau du candidat : un sujet qui pourrait être traité en une séance de deux heures avec un peu d'aide du professeur permettra tout au plus d'obtenir une note moyenne (si le candidat est clair et réactif) ; inversement, présenter un projet difficile mais mal maîtrisé conduit souvent à une note très décevante. Le jury comprend tout à fait que certains candidats puissent avoir envie de doter leur programme d'une interface et d'un graphisme recherchés : ils doivent cependant être bien conscients que leurs efforts sur ces points seront très peu valorisés dans la notation, et donc ne s'y attaquer que s'ils ont déjà terminé le cœur de leur projet.

Quelques exemples de projets parmi les plus fréquents :

- systèmes proies-prédateurs et dynamique des populations au sens large ;
- propagation d'un virus, d'un feu de forêt… ;
- jeux très simples (puissance 4, othello…) ;
- génétique des populations autour du modèle d'Hardy-Weinberg ;
- mutations de l'ADN…

Tous ces thèmes peuvent fournir matière à des projets de bonne qualité, à condition cependant de ne pas se limiter à leur version la plus basique : on ne peut pas obtenir une excellente note en présentant un puissance 4 sans même une intelligence artificielle.

Parmi les projets plus originaux ou plus ambitieux, on peut citer des algorithmes de compression d'images, de reconnaissance de caractères, des simulations physiques relativement poussées, des problèmes non triviaux d'optimisation dans des graphes, des jeux aux règles un peu plus complexes ou dotés d'une intelligence artificielle relativement sophistiquée… Les deux problèmes actuels de bio-informatique présentant le plus d'intérêt algorithmique (alignement de séquences et reconstruction d'arbres phylogénétiques) ont été abordés par certains candidats mais n'ont sans doute pas la popularité qu'ils méritent.

Certains projets se prêtent bien à une analyse statistique des résultats (par exemple en faisant varier certains paramètres dans la simulation de la propagation d'un virus) : il serait souhaitable que cette analyse soit alors systématiquement faite, et que les graphes soient générés à l'aide de Python.

La taille des projets (mesurée en nombre de lignes de codes) est très variable (d'une vingtaine de lignes à un gros millier), et finalement assez peu corrélée à leur qualité (même s'il est difficile de faire tenir un projet intéressant en moins de 60 lignes, et qu'écrire plus de 500 lignes de bonne qualité demande un temps dont peu d'étudiants de BCPST disposent). En revanche, les candidats sont invités à se demander systématiquement s'ils ne pourraient pas faire la même chose de manière plus concise et plus lisible (en remplaçant une série d'instructions quasiment identiques par une boucle, ou quatre fonctions très similaires par une seule plus générale par exemple).

4.3.1 Modélisation

De nombreux candidats présentent des projets contenant une part de modélisation. La qualité et la pertinence de cette modélisation ne sont pas les principaux éléments évalués (et en particulier, l'intérêt éventuel de la modélisation ne compense pas la pauvreté algorithmique de certains projets se limitant à la résolution numérique d'une équation différentielle). En revanche, on attend des candidats un minimum de lucidité sur la portée et les limites de leur modèle : affirmer qu'un modèle simpliste dans lequel quelques proies et prédateurs se déplacent dans une matrice pourrait servir de base de recherche pour la gestion des populations animales sauvages témoigne d'un manque de recul inquiétant. Un candidat ayant eu la curiosité de se renseigner sur le type de modèles réellement utilisés par les scientifiques du domaine sera bien sûr récompensé.

D'autre part, certains des projets les plus populaires cette année sont basés sur des algorithmes inspirés de systèmes biologiques. Il faut bien comprendre que l'objectif de ces algorithmes n'est nullement la modélisation biologique, mais bien la résolution d'un problème algorithmiquement difficile (d'optimisation, le plus souvent). Ainsi, « parce que cela se passe comme ça chez les fourmis » peut difficilement constituer une réponse acceptable (quelle que soit la question...). Un autre problème posé par ces projets est que les algorithmes sont souvent utilisés par les candidats dans des cas où ils ne présentent aucun intérêt : une colonie de fourmis pour résoudre de manière approchée le problème du voyageur de commerce est tout à fait légitime ; pour chercher le plus court chemin entre deux sommets d'un graphe, en revanche, on rappelle que l'algorithme de Dijkstra est au programme...

4.3.2 Style et lisibilité du code

On n'attend bien sûr pas des candidats qu'ils écrivent du code de qualité professionnelle. Nous nous contentons de donner ici quelques conseils pour améliorer la qualité stylistique du code, qui a un impact important tant sur sa lisibilité (par les différents membres du groupe et par l'examinateur) que sur la possibilité d'y détecter facilement les erreurs et d'y apporter des améliorations ultérieures.

- Le code doit être commenté, et nous remarquons que les candidats ont fait des efforts louables sur ce point. Il n'est certainement pas nécessaire de commenter chaque ligne, mais un commentaire d'une ou deux lignes au début de chaque fonction expliquant ce qu'elle prend en argument, ce qu'elle fait et ce qu'elle renvoie est très appréciable.
- Dans les noms de variables, de fonctions *et également de fichiers*, il faut éviter tous les caractères dits *spéciaux*, c'est-à-dire se restreindre aux lettres non accentuées (minuscules et majuscules), aux chiffres et au « tiret bas » `_`.
- Éviter d'utiliser des chemins absolus pour charger des fichiers : si l'examinateur souhaite tester le programme sur sa machine, il y a peu de chance qu'un appel `f = open("D:/MonProjetInfo/Donnees/especes.txt")` fonctionne (écrire à la place `f = open("Donnees/especes.txt")`).
- La longueur des lignes doit être raisonnable (idéalement, pas plus de 80 colonnes).
- Si l'on s'intéresse, par exemple, à l'évolution d'une population évoluant dans un environnement en deux dimensions représenté par une matrice, la taille de cette matrice gagnera à être stockée dans une variable (passée en paramètres aux différentes fonctions ou globale et définie au début du programme). On ne devrait jamais trouver une ligne telle que `for i in range(50)` dans un programme.
- De même, plutôt que d'écrire `if M[i][j] == 2` avec éventuellement un commentaire précisant « on regarde si la cellule est vivante », on préférera définir `VIVANTE = 2` au début du programme et écrire ensuite `if M[i][j] == VIVANTE`.

4.3.3 Notions « hors-programme »

Le jury ne pénalise pas *a priori* les candidats utilisant des notions hors-programme (ou à la limite du programme) comme les fonctions récursives, les dictionnaires ou la programmation orientée objet. Dans certains projets, leur emploi peut être très judicieux voire presque indispensable dans le cas des fonctions récursives. Cependant, un candidat utilisant ces notions doit savoir qu'il s'expose à des questions (élémentaires) portant dessus et être capable d'expliquer l'intérêt de leur utilisation dans son projet.

4.4 Remarques spécifiques de programmation et d'algorithmique

- Le code doit absolument être découpé en fonctions, et il faut réfléchir soigneusement au découpage le plus judicieux. Les quelques candidats n'utilisant aucune fonction (ou mettant la totalité de leur projet dans

une seule fonction, ce qui revient au même) ont été fortement pénalisés.

- On sépare en règle générale les fonctions effectuant des entrées-sorties (affichage à l'écran, lecture dans un fichier...) des autres. Ainsi, on n'utilisera normalement pas une même fonction pour calculer et afficher une valeur.
- De nombreux projets nécessitent de parcourir les voisins d'une case d'un tableau bidimensionnel. Il est très regrettable que la plupart des candidats utilisent pour cela plusieurs dizaines de lignes de code (et pas de boucle !), surtout que beaucoup sont capables d'écrire en quelques minutes quelque chose de satisfaisant avec l'aide de l'examinateur.
- Il est fortement déconseillé d'utiliser un algorithme que l'on n'est pas capable d'expliquer correctement lorsque l'examinateur le demande, surtout s'il s'agit de plus d'un algorithme au programme (Dijkstra, tri rapide...).
- Il est souvent possible en Python d'utiliser une boucle `for` là où une boucle `while` serait nécessaire dans certains langages (à l'aide typiquement d'un `return` dans le corps de la boucle). Les candidats sont encouragés à tirer partie de cette possibilité, mais *cela ne les dispense pas de savoir écrire une boucle* `while` *en gérant correctement les conditions de sortie.*
- Très peu de candidats utilisent correctement les booléens. Idéalement, on aimerait lire plus souvent des `return x > 0` (par exemple), mais le jury est conscient que ce n'est pas évident conceptuellement. En revanche, éviter les `if f(x) == True` (ou, pire, `if f(x) == 'Ca marche !'`) semble un objectif raisonnable.
- Une structure `if ... then ... else` n'est pas une boucle ! Il n'est pas vraiment gênant d'ignorer le terme de *branchement conditionnel*, mais parler de boucle à ce sujet suggère une confusion majeure.
- Très peu de candidats maîtrisent la différence entre une fonction renvoyant une valeur et une fonction modifiant son argument. C'est finalement assez compréhensible, mais peut malheureusement résulter en un très grand nombre de copies inutiles (copier toute la matrice à chaque fois que l'on modifie une case par exemple) qui sont parfois la principale raison pour laquelle le programme est trop lent pour traiter des données de taille réaliste.
- Certains candidats codent des chaînes de caractères très longues, ou des matrices de « grande » taille directement dans leur programme. Dans nombre de cas, utiliser un fichier externe aurait été plus adapté, et aurait permis aux candidats d'illustrer leur maîtrise de compétences acquises durant la préparation.

Examinateurs

Jean-Baptiste Bianquis, Jean-Loup Carré, Guillaume Connan, Christophe François, Nicolas Guillaud, Céline Hudelot, Laurent Jospin, Stéphane Legros, Antoine Manzanera, Valdimir Paun, Vojislav Petrov, Landry Salle

Expert et rapporteur

Jean-Baptiste Bianquis