

Corrigé du partiel de M.A.O. Calcul Formel

Mai 2020
Master 1 M.F., Orsay

Partie 1 : Une équation polynomiale

1. La condition (1) signifie qu'il existe $S \in \mathbb{K}[X]$ tel que $P(X)A(X)-Q(X) = X^{2d}S(X)$; autrement dit, cela signifie que pour tout $i < 2d$ le coefficient de X^i dans le polynôme $P(X)A(X) - Q(X)$ est nul. Lorsque $i \geq d$ cela équivaut à $\lambda_i = 0$ puisque $\deg Q < d$. Pour $i \leq d$ cela signifie simplement que le coefficient de X^i dans le polynôme Q est égal à λ_i. L'équivalence en découle, ainsi que le fait que $Q(X) = \sum_{i=0}^{d-1} \lambda_i X^i$.

2. Etant donné $P \in \mathbb{K}_d[X]$ on calcule $P(X)A(X)$ par l'algorithme de multiplication rapide de Karatsuba. Comme $\max(\deg A, \deg P) < 2d$ cela coûte $O(d^{\log(3)/\log(2)})$ opérations arithmétiques dans $\mathbb{K}$. Vérifier si les λ_i sont nuls ne coûte aucune opération arithmétique. *Certains étudiants ont répondu* $O((2d)^{\log(3)/\log(2)})$ *ou* $O((2d-1)^{\log(3)/\log(2)})$ *opérations : c'est correct, mais si vous écrivez cela sans constater ensuite que c'est la même chose que* $O(d^{\log(3)/\log(2)})$ *je me demande si vous avez compris ce qu'est un symbole* O.

3. ($\star$)

```
R.<x>=PolynomialRing(QQ)
def question3(A,P,d):
    AP=A*P #On peut remplacer cela par une version de Karatsuba
    Bool=True
    r=d
    while ((Bool)and(r<2*d)):
        Bool=(AP[r]==0)
        r=r+1
    if Bool:
        return (Bool,AP[:d])
    else:
        return (Bool,'Pas de tel polynôme!')
```

4. ($\star$) Avec $d = 3$, $P(X) = X^3 + 2X^2 - X - 1$ et $A(X) = 11X^5 - 3X^4 + 4X^3 + X + 2$, on trouve $P(X)A(X) = 11X^8 + 19X^7 - 13X^6 + 3X^2 - 3X - 2$ donc (1) est vérifiée avec $Q(X) = 3X^2 - 3X - 2$. En revanche lorsque $P(X) = X^3$ on a $P(X)A(X) = 11X^8 - 3X^7 + 4X^6 + X^4 + 2X^3$ donc la fonction renvoie `False`.

```
d=3
P=x^3+2*x^2-x-1
A=11*x^5-3*x^4+4*x^3+x+2
question3(A,P,d)

P=x^3
question3(A,P,d)
```

5. Les $d+1$ coefficients de P sont les inconnues. Les d équations sont les $\lambda_i = 0$ pour $i \in \{d, d+1, \ldots, 2d-1\}$. Chaque λ_i est une combinaison linéaire des inconnues, avec pour coefficients des coefficients de A (voir la question 9). Il s'agit donc d'un système linéaire homogène à coefficients dans le corps $\mathbb{K}$. Calculer la matrice de ce système ne nécessite aucune opération : les coefficients de la matrice sont des coefficients de A. L'algorithme du pivot de Gauss fournit les solutions en $O(d^3)$ opérations arithmétiques dans $\mathbb{K}$. *On pourrait vouloir raffiner en disant que $O(d^2 r)$ opérations suffisent, où r est le rang du système; mais comme on n'a aucune information sur r, ce raffinement n'a aucun intérêt et il vaut mieux en rester à $O(d^3)$ opérations.*

6. Le système linéaire homogène de la question précédente a strictement moins d'équations que d'inconnues, donc il possède une solution non triviale.

Partie 2 : Récurrences linéaires

7. Avec $d = 1$ et $P = 1$, on a $a_0 = 1$ et $a_1 = 0$, donc $\sum_{i=0}^{d} a_i u_{n+d-i} = u_{n+1}$ qui est nul pour tout $n \geq 0$: la suite (u_n) est annulée par P à l'ordre 1. Si elle était annulée à l'ordre 0 par un polynôme $P = a_0$ constant non nul, on aurait $a_0 u_n = 0$ pour tout $n \geq 0$, ce qui n'est pas le cas pour $n = 0$.

8. Une suite (u_n) est annulée à l'ordre 0 par un polynôme $P = a_0$ constant non nul si, et seulement si, on a $a_0 u_n = 0$ pour tout $n \geq 0$, ce qui signifie que (u_n) est la suite nulle. Considérons maintenant $P = a_1 X + a_0 \neq 0$ avec $d = 1$. Alors (u_n) est annulée par P à l'ordre 1 si, et seulement si, on a $a_0 u_{n+1} + a_1 u_n = 0$ pour tout $n \in \mathbb{N}$. Si $a_0 = 0$ alors $a_1 \neq 0$ et il s'agit de la suite nulle. Si $a_0 \neq 0$ cela signifie que (u_n) est géométrique de raison $-a_1/a_0$, donnée par $u_n = (-a_1/a_0)^n u_0$. Il faut prendre garde au cas où $a_1 = 0$: il s'agit alors d'une suite nulle à partir de u_1, mais pour laquelle u_0 peut être quelconque comme dans la question précédente. *J'ai apprécié les quelques copies qui ont repéré ce lien avec la question précédente.*

9. Comme $P = a_d X^d + \ldots + a_1 X + a_0 \in \mathbb{K}_d[X]$, le coefficient de X^j dans le polynôme $P(X)U_t(X)$ est
$$\sum_{i=\max(0,j-t+1)}^{\min(d,j)} a_i u_{j-i}.$$
Lorsque $t > j$ on a $\max(0, j-t+1) = 0$ donc ce coefficient ne dépend pas de t.

10. La question précédente montre que $\sum_{i=0}^{d} a_i u_{n+d-i}$ est le coefficient de X^{n+d} dans le polynôme $P(X)U_t(X)$, pour tous $n \in \mathbb{N}$ et $t > n+d$; l'équivalence entre (i) et (ii) en découle immédiatement. Par ailleurs, remarquons que deux polynômes sont congrus modulo X^t si, et seulement si, pour tout $j < t$ ils ont le même coefficient devant X^j. On en déduit l'équivalence entre (ii) et (iii); en effet, si (ii) est vérifiée, on construit Q tel que pour tout $j \in \{0, \ldots, d-1\}$ le coefficient de X^j dans Q soit égal à celui dans $P(X)U_t(X)$ pour $t > j$.

11. Pour tout t on a $P_1Q_2 \equiv P_1P_2U_t \equiv P_2Q_1 \bmod X^t$. Le polynôme $P_1Q_2 - P_2Q_1$ est divisible par X^t pour tout t, donc il est nul.

12. Soit $(P, Q) \in \mathcal{E}$ avec $P \neq 0$. Notons D le pgcd de P et Q, et posons $P_1 = P/D$ et $Q_1 = Q/D$. On a bien $\deg P_1 \leq \deg P \leq d$ et $\deg Q_1 \leq \deg Q < d$. Notons $D(X) = X^\delta D_1(X)$ avec $D_1(0) \neq 0$, où δ est la multiplicité de 0 comme racine de D; on a alors $P = X^\delta D_1 P_1$ et $Q = X^\delta D_1 Q_1$. Soit $t \in \mathbb{N}^*$. Alors $X^{t+\delta}$ divise $PU_{t+\delta} - Q = X^\delta D_1(P_1U_{t+\delta} - Q_1)$ donc X^t divise $D_1(P_1U_{t+\delta} - Q_1)$. Comme $D_1(0) \neq 0$, X^t est premier avec D_1 donc finalement X^t divise $P_1U_{t+\delta} - Q_1$. Comme X^t divise aussi $U_{t+\delta} - U_t$, on en déduit que $P_1U_t \equiv Q_1 \bmod X^t$.

13. Il est clair que si $(P, Q) \in \mathcal{E}$ et $\lambda \in \mathbb{K}^*$ alors $(\lambda P, \lambda Q) \in \mathcal{E}$. Il existe donc $(P, Q) \in \mathcal{E}$ avec P unitaire ; on choisit un tel couple (P_0, Q_0) pour lequel P_0 est de degré minimal. D'après la question 12, la minimalité de $\deg P_0$ montre que $\operatorname{pgcd}(P_0, Q_0) = 1$. Soit $(P, Q) \in \mathcal{E}$. D'après la question 11 on a $PQ_0 = P_0Q$. Comme P_0 est premier avec Q_0, il divise P : il existe $S \in \mathbb{K}[X]$ tel que $P = P_0S$. La relation $PQ_0 = P_0Q$ donne alors $Q = Q_0S$. Réciproquement, si un couple $(P, Q) \in \mathbb{K}_d[X] \times \mathbb{K}_{d-1}[X]$ est de la forme $P = P_0S$ et $Q = Q_0S$ avec $S \in \mathbb{K}[X]$, alors pour tout $t \in \mathbb{N}^*$ le polynôme $PU_t - Q = (P_0U_t - Q_0)S$ est multiple de X^t, donc $(P, Q) \in \mathcal{E}$. Enfin, en écrivant un couple $(P, Q) \in \mathcal{E}$ sous la forme $P = P_0S$ et $Q = Q_0S$ on voit que S est le pgcd de P et Q; si P et Q sont premiers entre eux alors ils sont respectivement associés à P et Q, ce qui entraîne l'unicité du couple (P_0, Q_0) du (i).

14. On reprend d'abord la preuve de la question 11 : on a $PQ_0 \equiv PP_0U_{2d} \equiv P_0Q \bmod X^{2d}$. Le polynôme $PQ_0 - P_0Q$ est de degré $< 2d$ et il est divisible par X^{2d} donc c'est le polynôme nul : on a $PQ_0 = P_0Q$. On suit alors une partie de la preuve de la question 13 : comme P_0 est premier avec Q_0, il existe $S \in \mathbb{K}[X]$ tel que $P = P_0S$. La relation $PQ_0 = P_0Q$ donne alors $Q = Q_0S$. Comme P_0 et Q_0 sont premiers entre eux, on a $\operatorname{pgcd}(P, Q) = \operatorname{pgcd}(P_0S, Q_0S) = S$ ce qui termine la preuve.

15. On peut supposer $d \geq 1$. On a vu à la question 5 comment trouver, en $O(d^3)$ opérations arithmétiques dans $\mathbb{K}$, un couple (P, Q) satisfaisant aux hypothèses de la question précédente. Le calcul de $D = \operatorname{pgcd}(P, Q)$ par l'algorithme d'Euclide coûte $O(d^2)$ opérations arithmétiques dans $\mathbb{K}$. Enfin le calcul de P/D et Q/D coûte à nouveau $O(d^2)$ opérations arithmétiques dans $\mathbb{K}$. Finalement, on détermine P_0 et Q_0 en $O(d^3)$ opérations arithmétiques dans $\mathbb{K}$.

Partie 3 : Application matricielle

16. Soient $M_1, M_2 \in M_d(\mathbb{K})$. Il y a d^2 coefficients du produit $M_1 M_2$ à calculer. Chacun d'entre eux s'écrit sous la forme $\sum_{k=1}^{d} a_{i,k} b_{k,j}$ donc le calculer nécessite d produits et $d-1$ sommes, donc $O(d)$ opérations arithmétiques dans $\mathbb{K}$. Finalement, le coût du calcul est bien $O(d^3)$ opérations arithmétiques dans $\mathbb{K}$.

17. On calcule M^n par exponentiation rapide. Cela nécessite d'effectuer $O(\log n)$ produits de matrices. D'après la question précédente, le coût est $O(d^3 \log n)$ opérations arithmétiques dans $\mathbb{K}$. Ensuite on calcule $M^n Y$: ce produit d'une matrice carrée par un vecteur colonne nécessite $O(d^2)$ opérations : il y a d coefficients à calculer, et chacun s'obtient comme somme de d produits. Le calcul de $u_k = {}^tX(M^k Y)$ ne nécessite alors que d produits et $d-1$ sommes. Donc le coût du passage de M^n à u_n est $O(d^2)$: il est négligeable devant le coût du calcul de M^n. Finalement le coût du calcul de u_n est $O(d^3 \log n)$ opérations arithmétiques dans $\mathbb{K}$. *L'indication donnée par l'énoncé selon laquelle on considère n extrêmement grand par rapport à d a induit en erreur plusieurs étudiants, qui ont donné une complexité en $O(\log n)$ opérations. Ce n'est pas faux, mais cela veut dire que la constante implicite dans le symbole O dépend de d. Ce n'est pas ce que j'attendais, mais je n'ai pas pénalisé ceux qui ont répondu ainsi.*

18. Une première méthode consiste à calculer d'abord les puissances de M. Chaque matrice M^k s'obtient à partir de la précédente M^{k-1} en $O(d^3)$ opérations d'après la question 16. Au total on utilise donc $O(nd^3)$ opérations arithmétiques dans $\mathbb{K}$ pour calculer M^0, M^1, ..., M^n. Ensuite pour chaque k on calcule $M^k Y$ puis $u_k = {}^tX(M^k Y)$; comme à la question précédente, ce calcul ne coûte au total que $O(nd^2)$ opérations, ce qui est négligeable devant le coût du calcul des M^k. Finalement, le coût du calcul des n premiers termes de la suite (u_n) par cette méthode est $O(nd^3)$ opérations arithmétiques dans $\mathbb{K}$.

    On peut cependant faire mieux : pour tout k compris entre 0 et n on calcule le vecteur colonne $M^k Y$. Chacun de ces vecteurs s'obtient en multipliant M par le vecteur précédent, ce qui coûte $O(d^2)$ opérations. On obtient donc Y, MY, ..., $M^n Y$ en $O(nd^2)$ opérations arithmétiques dans $\mathbb{K}$, ce qui permet de calculer les n premiers termes de la suite (u_n) en $O(nd^2)$ opérations.

19. ($\star$)Une version naïve :

```
k = GF(37)
Mat= MatrixSpace(k,3,3)
Vec= VectorSpace(k,3)
M=Mat([2,3,14,5,26,11,4,13,8])
X=Vec([1,3,7])
Y=Vec([28,19,10])
```

```
def question19(n):
    return [X*M^i*Y for i in range(n)]
#On peut aussi utiliser une exponentiation rapide pour le calcul des M^i
L=question19(100)
L[len(L)-1]
```

et une version utilisant l'algorithme plus efficace de la question 18 :

```
k = GF(37)
Mat= MatrixSpace(k,3,3)
Vec= VectorSpace(k,3)
M=Mat([2,3,14,5,26,11,4,13,8])
X=Vec([1,3,7])
Y=Vec([28,19,10])
def question19(n):
    L=[Y]
    for i in range(1,n+1):
        L.append(M*L[i-1])
    L=[X*i for i in L]
    return L
L=question19(100)
L[len(L)-1]
```

20. Notons $\chi_M(X) = a_0X^d + \ldots + a_{d-1}X + a_d$ le polynôme caractéristique de la matrice M, et posons $P(X) = X^d\chi_M(1/X) = a_dX^d + \ldots + a_1X + a_0$ (ce qui est cohérent avec les notations de la partie 2). D'après le théorème de Cayley-Hamilton, on a $\chi_M(M) = 0$, c'est-à-dire $a_0M^d + \ldots + a_{d-1}M + a_dI_d = 0$. En multipliant cette relation par M^nY à droite, et par ${}^tX$ à gauche, on voit que $a_0u_{n+d} + \ldots + a_{d-1}u_{n+1} + a_du_n = 0$ pour tout $n \in \mathbb{N}$. Donc (u_n) est annulée par P à l'ordre d, au sens défini au début de la partie 2. En particulier (u_n) satisfait à une récurrence linéaire d'ordre d.

21. Reprenons la preuve de la question 20, avec μ_M à la place de χ_M. On note $\delta = \deg \mu_M \leq d$ et $\mu_M = b_{d-\delta}X^\delta + \ldots + b_{d-1}X + b_d$; on peut avoir $\delta < d$, et dans ce cas on pose $b_i = 0$ pour tout entier i tel que $0 \leq i < d - \delta$. Alors $P(X) = X^d\mu_M(1/X) = b_dX^d + \ldots + b_1X + b_0$ annule (u_n) (de la même façon qu'à la question 20). D'après la question 10 il existe $Q \in \mathbb{K}[X]$ tel que $(P, Q) \in \mathcal{E}$. La question 13 (ii) fournit alors un polynôme S tel que $P(X) = S(X)P_0(X)$. On en déduit que $\deg S = \deg P - \deg P_0 \leq d - \deg P_0$ d'où $X^{d-\deg P_0}S(1/X) \in \mathbb{K}_d[X]$; donc $\mu_M(X) = X^dP(1/X) = X^{d-\deg P_0}S(1/X)X^{\deg P_0}P_0(1/X)$ est multiple de $X^{\deg P_0}P_0(1/X)$.

22. Si on ne suppose pas que M est inversible, on peut avoir $\mu_M(0) = 0$. Dans ce cas, avec les notations de la question précédente on a $b_d = 0$ et $\deg P < d$; la même preuve montre que $\mu_M(X)$ est multiple de $X^m \cdot X^{\deg P_0}P_0(1/X)$ où m est la

multiplicité de 0 comme racine de μ_M. Pour éviter ce problème l'énoncé suppose que M est inversible, donc 0 n'est pas racine de μ_M. Supposons alors que $\mu_M(X)$ n'est pas associé à $X^{\deg P_0}P_0(1/X)$ et notons $d_1 = \deg(X^{\deg P_0}P_0(1/X))$. Comme $T(X) = X^{\deg P_0}P_0(1/X)$ est un diviseur de $\mu_M(X)$, il n'y a qu'un nombre fini de possibilités pour ce polynôme T (à association près). D'après la question précédente on a alors $d_1 < \deg \mu_M$, et on constate que la suite (u_n) satisfait à une relation de récurrence linéaire d'ordre d_1. Cela implique notamment ${}^tXT(M)Y = 0$. Comme T est un diviseur strict de μ_M, on a $T(M) \neq 0$ donc $T(M)Y \neq 0$ (à moins que Y n'appartienne au noyau de $T(M)$, qui n'est pas l'espace tout entier ; comme $\mathbb{K}$ est infini et Y choisi au hasard cela a une probabilité nulle d'arriver). La relation ${}^tXT(M)Y = 0$ montre alors que X appartient à un certain hyperplan. Finalement, pour chaque T on obtient que ${}^tXT(M)Y \neq 0$ avec probabilité 1. Comme il n'y a qu'un nombre fini de T à considérer, cela termine la preuve.

23. D'après la question 20, la suite (u_n) satisfait à une récurrence linéaire d'ordre d. La question 15 qui conclut la partie 2 fournit un algorithme permettant de calculer le couple (P_0, Q_0) déterminé à la question 13. En admettant que $\mu_M(X) = cX^{\deg P_0}P_0(1/X)$, où c est l'inverse du coefficient dominant de $X^{\deg P_0}P_0(1/X)$, on obtient donc le polynôme minimal de M. Pour mettre en œuvre cet algorithme on doit d'abord calculer u_0, u_1, …, u_{2d-1}. D'après la question 18 cela coûte $O(d^3)$ opérations arithmétiques dans $\mathbb{K}$. Ensuite l'algorithme de la question 15 coûte également $O(d^3)$ opérations arithmétiques. Enfin passer de P_0 à $X^{\deg P_0}P_0(1/X)$ ne coûte *aucune* opération arithmétique : on renverse simplement la liste formée par les coefficients. *Attention, il faut bien dire ici ce qu'on compte. Renverser une liste prend a priori un temps linéaire en le nombre d'éléments de cette liste, mais dans ce partiel (et plus généralement dans ce cours) on ne s'intéresse qu'au nombre d'opérations arithmétiques effectuées, et ici il n'y en a aucune. Cela illustre le fait que calculer une complexité nécessite de faire des choix, qui peuvent être plus ou moins pertinents selon la situation concrète.* Il ne reste qu'à multiplier chaque coefficient par c, soit $O(d)$ opérations : le coût global est donc de $O(d^3)$ opérations arithmétiques dans $\mathbb{K}$. Comparons ce coût à celui de l'algorithme naïf. Calculer les matrices M^k, $0 \leq k \leq d$, coûte $O(d^4)$ opérations arithmétiques dans $\mathbb{K}$. Rechercher les relations linéaires entre ces matrices se fait par un pivot de Gauss : on a $d+1$ vecteurs dans l'espace vectoriel $M_d(\mathbb{K})$ qui est de dimension d^2. On travaille donc avec une matrice de taille $(d+1) \times d^2$. Le coût du pivot de Gauss est de $O((d+1)^2d^2) = O(d^4)$ opérations arithmétiques dans $\mathbb{K}$, puisque $d+1 \leq d^2$. Finalement, l'algorithme naïf a un coût de $O(d^4)$ opérations, et celui que propose le sujet est plus efficace.

24. ($\star$) Une première solution "à la main".

```
k=GF(37)
Mat= MatrixSpace(k,3,3)
Vec= VectorSpace(k,3)
```

```
M=Mat([2,3,14,5,26,11,4,13,8])
X=Vec([1,3,7])
Y=Vec([28,19,10])
R.<x>=PolynomialRing(k)
Matr=MatrixSpace(k,3,4)
S=Matr(lambda i,j: U[3+i-j])
K=S.right_kernel()
b=K.basis()[0]
P=sum(b[i]*x^i for i in range(len(b)))
Ud=sum(k(U[i])*x^i for i in range(len(U)))
d=3
Bool,Q=question3(Ud,P,d)

gcd(P,Q)

x^3*P.substitute(x=1/x)
```

et l'on peut vérifier le résultat grâce à la commande

```
M.minpoly()
```

et une solution complètement générale qui prend en entrée un corps k et une matrice à coefficients dans k et ressort le polynôme minimal de M sur k en utilisant l'algorithme décrit (le code est commenté dans le Notebook joint à ce pdf).

```
def question19(n,M,X,Y):
    L=[Y]
    for i in range(1,n+1):
        L.append(M*L[i-1])
    L=[X*i for i in L]
    return L

def totale(M,k):
    R.<x>=PolynomialRing(k)
    d=len(M.rows()) #
    if (d!=len(M.columns())):
        return 'Matrice non carrée!'
    else:
        V=VectorSpace(k,d)
        Bool=True
        compt=1
        while Bool:
```

```
                compt=compt+1
                X=V([k.random_element() for i in range(d)])
                Y=V([k.random_element() for i in range(d)])
                U=un(2*d-1,M,X,Y)
                Matr=MatrixSpace(k,d,d+1)
                S=Matr(lambda i,j: U[d+i-j])
                K=S.right_kernel()
                b=K.basis()[0]
                P=sum(b[i]*x^i for i in range(len(b)))
                Ud=sum(U[i]*x^i for i in range(len(U)))
                aux,Q=question3(Ud,P,d)
                if aux:
                    D=gcd(P,Q)
                    P0=P//D
                    p=P0[P0.degree()]
                    P0=P0//p
                    mu=x^(P0.degree())*P0.substitute(x=1/x)
                    mu=mu.numerator()
                    m=1/mu[mu.degree()]
                    mu=mu*m
                    if mu(M)==0:
                        Bool=False
                        return mu
                    else:
                        for i in range(d):
                            mu=x*mu
                            if mu(M)==0:
                                Bool=False
                                return mu
```

que vous pouvez tester avec

```
k=GF(7)
R.<x>=PolynomialRing(k)
Mat=MatrixSpace(k,4)
M=Mat([1,0,0,0,0,1,9,0,2,1,1,13,1,1,0,0])
M.minpoly()

totale(M,k)
```

ou

```
k=QQ
R.<x>=PolynomialRing(k)
Mat=MatrixSpace(k,4)
M=Mat([1,0,0,0,2,1,9,0,2,1,1,0,1,1,0,0])
M.minpoly()

totale(M,k)
```

25. Remarquons d'abord que pour tout $C \in M_{d,1}(\mathbb{K})$, le calcul du produit MC coûte $O(r)$ opérations arithmétiques dans $\mathbb{K}$. En effet chaque coefficient non nul de M apparaît dans exactement un produit avec un élément de C, et dans un calcul de somme. On commence par calculer MY, M^2Y, …, $M^{2d-1}Y$: chacun de ces vecteurs s'obtient en $O(r)$ opérations à partir du précédent, donc on les obtient tous en $O(dr)$ opérations. Ensuite on en déduit chaque u_k, $0 \leq k \leq 2d-1$, en $O(d)$ opérations. Finalement on a utilisé jusqu'ici $O(dr)$ opérations, puisque $r \geq d$. On trouve alors un couple (P, Q) vérifiant la relation (1) de la partie 1, avec $A = U_{2d}$: on admet qu'on peut faire ceci en $O(d^2)$ opérations (et le partiel d'avril 2017 montre comment faire), donc cela n'ajoute rien à la complexité puisque $r \geq d$. Comme à la question 15 on calcule alors $D = \text{pgcd}(P, Q)$ par l'algorithme d'Euclide en $O(d^2)$ opérations arithmétiques, puis P/D et Q/D ce qui coûte à nouveau $O(d^2)$ opérations. Finalement, on détermine P_0 et Q_0 en $O(dr)$ opérations arithmétiques dans $\mathbb{K}$. En admettant que $\mu_M(X) = cX^{\deg P_0} P_0(1/X)$, on obtient donc le polynôme minimal de M en $O(dr)$ opérations. Si $r = d^2$ on retrouve la complexité calculée à la question 23, mais si r est nettement plus petit on trouve un algorithme bien plus efficace. De son côté l'algorithme naïf a toujours la même complexité car le pivot de Gauss utilise toujours $O(d^4)$ opérations.

26. L'algorithme de la question 23 n'est pas plus efficace que celui dont on admet l'existence. En revanche, lorsque r est nettement plus petit que d^2 il n'utilise que $O(dr)$ opérations au lieu de $O(d^3)$ donc il est meilleur.