DS07

Problèmes stationnaires et algèbre linéaire

Sources :

Proposition de corrigé

Exercice 1 : Quelques exemples d'analyse

Importation des modules nécessaires :

```
import scipy.integrate as si
from math import sqrt, sin, cos, atan
from numpy import array
import numpy as np
```

Q 1 : Donner une approximation de $\int_{\alpha}^{\alpha+1}\cos(\sqrt{t})\,dt$ avec la méthode des trapèzes et 1000 subdivisions.

```
def trapeze(f,a,b,n):
    h=(b-a)/n
    S=0.5*(f(a)+f(b))
    for k in range(1,n):
        S+=f(a+k*h)
    return S*h

print("Qu. 1 : ", trapeze(lambda x : cos(sqrt(x)),alpha,alpha+1,1000))
```

Q 2 : Donnez une approximation (à 10^{-}5 près) de l'unique réel positif solution de l'équation $x^2+\sqrt{x}-10=\alpha$ avec la méthode de votre choix.

```
def newton(f, fp, x0, epsilon):
    """Zéro de f par la méthode de Newton
       départ : x0, f' = fp, critère d'arrêt epsilon"""
    u = x0
    v = u - f(u)/fp(u)
    k=0
    while abs(v-u) > epsilon:
        u, v = v, v - f(v)/fp(v)
        k+=1
    return u,k

def f2(t):
    return t**2+t**0.5-10-alpha
```

```
def f2p(t):
    return 2*t+0.5*t**(-0.5)

print("Qu. 2 : méthode de Newton ", newton(f2,f2p,1,1e-5)[0])
```

Q 3 : Donnez le nombre d'itérations nécessaires pour obtenir ce résultats avec la méthode de Newton en prenant pour valeur initiale α.

```
print("Qu. 3 : méthode de Newton ", newton(f2,f2p,alpha,1e-5)[1])
```

Q 4 : Donnez le nombre d'itérations nécessaires pour obtenir ce résultats avec la méthode de Dichotomie sur l'intervalle $[0,12+\alpha]$.

```
def dichotomie(f, a, b, epsilon):
    """Zéro de f sur [a,b] àepsilon près, par dichotomie
       Préconditions : f(a) * f(b) <= 0
                     f continue sur [a,b]
                     epsilon > 0"""
    c, d = a, b
    fc, fd = f(c), f(d)
    k=0
    while d - c > 2 * epsilon:
        m = (c + d) / 2.
        fm = f(m)
        if fc * fm <= 0:
            d, fd = m, fm
        else:
            c, fc = m, fm
        k+=1
    return (c + d) / 2.,k

print("Qu. 4 : méthode de dichotomie ", dichotomie(f2, 0, 12+alpha, 1e-5)[1])
```

Q 5 : Donnez à l'aide une approximation (à 10^-5 près) de l'unique réel positif t tel que

$$\int_{\alpha}^{\alpha+t} (2+\sqrt{x}+\cos x)\,dx = 10$$

```
print("Qu. 5 : ", dichotomie(lambda t : trapeze(lambda x : 2+sqrt(x)+cos(x),alpha,↙
    alpha+t,1000)-10,0,50,1e-5))
```

Q 6 : Donner une valeur approchée de $x(1)$ avec x l'unique fonction vérifiant $x(0)=\alpha$ et pour tout $t\in\mathbb{R}$, $x'(t)=3\cos(x(t))+t$.

```
def F (x,t) :
    return 3*cos(x) + t

les_t = [i/10000 for i in range(10001)]

print("Qu. 6 : ", si.odeint(F,alpha,les_t)[-1,0])
```

Q 7 : Donner une valeur approchée de $x(1+\frac{\alpha}{10})$ avec x l'unique fonction vérifiant $x(0)=0$, $x'(0)=0$ et pour tout $t\in\mathbb{R}$, $x''(t)=1+\sin(t+x(t))$.

```
def G (X,t) :
    a,b = X[0],X[1]
    return array([b,1+sin(t+a)])

les_t = [i*(1+alpha/10)/10000 for i in range(10001)]

print("Qu. 7 : ", si.odeint(G,array([0,0]),les_t)[-1,0])
```

Q 8 : Donner une approximation de $\beta \in \mathbb{R}$, pour que l'unique solution de l'équation différentielle non linéaire $x''(t) = 1 + \arctan(t + x(t))$ avec les conditions initiales $x(0) = 0$ et $x'(0) = \beta$ vérifie

$$x(1+\frac{\alpha}{10}) = 1 + \frac{2}{3}\alpha$$

```
def H (X,t) :
    a,b = X[0],X[1]
    return array([b,1+atan(t+a)])

les_t = [i*(1+alpha/10)/10000 for i in range(10001)]

f = lambda beta : si.odeint(H,array([0,beta]),les_t)[-1,0]-1-(2/3)*alpha

print("Qu. 8 : ",so.brentq(f,-2,2))
```

Exercice 2 : Algèbre linéaire

Importation des modules nécessaires :

```
from numpy import array
import numpy as np
```

Q 9 : Résoudre

$$A\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \alpha \end{pmatrix}$$

Donner la valeur de x_1.

```
A=array([[0,1,32,243],[1,32,243,1024],[32,243,1024,3125],[243,1024,3125,7776]])
B=array([[1],[2],[3],[alpha]])

X=np.linalg.solve(A,B)

print("Qu. 9 : ",X[0][0])
```

Q 10 : Calculer $B = A^3$ et donner le reste du coefficient de B situé sur la première ligne et la première colonne de B (donc d'indices 0 et 0 en numpy) dans la division par $10\,000 + \alpha$.

```
  B=np.dot(A,np.dot(A,A))
print("Qu. 10 : ",B[0][0]%(10000+alpha))
```

Cycle 01

alpha	1	2	3	4	5	6	7	8	9	10
1	0.34216	3.04233	5	20	3.26712	1.76984	0.86206	0.66243	-0.07723	5921
2	-0.00766	3.19568	4	20	3.06588	1.82162	1.04873	0.96118	-0.06054	5004
3	-0.29315	3.34242	3	20	2.63912	1.89042	1.25312	1.19673	-0.04386	4097
4	-0.52099	3.48334	3	20	2.19411	2.10183	1.47300	1.38252	-0.02718	3200
5	-0.69739	3.61906	3	20	1.90607	7.54315	1.70550	1.52843	-0.01049	2313
6	-0.82810	3.75013	4	20	1.88252	7.95400	1.94736	1.64188	0.00619	1436
7	-0.91842	3.87698	4	20	2.14522	8.03708	2.19531	1.72860	0.02288	569
8	-0.97326	4.00000	4	20	2.34453	8.08993	2.44649	1.79303	0.03956	9720
9	-0.99712	4.11951	4	21	2.20055	8.14967	2.69882	1.83872	0.05624	8874
10	-0.99417	4.23579	4	21	1.89821	8.28311	2.95132	1.86850	0.07293	8038
11	-0.96821	4.34909	5	21	1.64141	12.58572	3.20420	1.88467	0.08961	7212
12	-0.92272	4.45962	5	21	1.54868	14.19340	3.45887	1.88912	0.10629	6396
13	-0.86091	4.56758	5	21	1.67152	14.30237	3.71776	1.88343	0.12298	5590
14	-0.78567	4.67314	5	21	1.90725	14.35871	3.98406	1.86890	0.13966	4794
15	-0.69964	4.77645	5	21	1.93341	14.41326	4.26137	1.84664	0.15634	4008
16	-0.60522	4.87765	5	21	1.73724	14.50955	4.55323	1.81760	0.17303	3232
17	-0.50457	4.97686	5	21	1.50855	15.29450	4.86264	1.78259	0.18971	2466
18	-0.39963	5.07419	5	21	1.38240	20.40593	5.19149	1.74229	0.20639	1710
19	-0.29214	5.16975	5	21	1.42092	20.56440	5.54010	1.69731	0.22308	964
20	-0.18367	5.26363	5	21	1.60612	20.62737	5.90687	1.64818	0.23976	228
21	-0.07558	5.35591	5	21	1.72355	20.67949	6.28850	1.59537	0.25644	9523
22	0.03091	5.44667	5	21	1.62185	20.75540	6.68048	1.53928	0.27313	8808
23	0.13473	5.53599	5	21	1.42695	21.05014	7.07814	1.48028	0.28981	8103
24	0.23495	5.62393	5	21	1.28426	26.54846	7.47781	1.41870	0.30649	7408
25	0.33079	5.71055	5	21	1.26870	26.82107	7.87775	1.35483	0.32318	6723
26	0.42156	5.79591	5	21	1.39417	26.89529	8.27861	1.28893	0.33986	6048
27	0.50671	5.88006	5	21	1.54373	26.94729	8.68326	1.22126	0.35654	5383
28	0.58576	5.96306	5	21	1.52382	27.01126	9.09613	1.15203	0.37323	4728
29	0.65835	6.04494	6	21	1.36971	27.17806	9.52218	1.08144	0.38991	4083
30	0.72420	6.12576	6	22	1.22197	32.39318	9.96561	1.00970	0.40659	3448
31	0.78309	6.20555	6	22	1.16952	33.06839	10.42864	0.93696	0.42328	2823
32	0.83490	6.28436	6	22	1.24283	33.16174	10.91067	0.86341	0.43996	2208
33	0.87954	6.36221	6	22	1.38778	33.21590	11.40815	0.78919	0.45664	1603
34	0.91702	6.43913	6	22	1.43133	33.27281	11.91561	0.71445	0.47333	1008
35	0.94736	6.51518	6	22	1.32379	33.38584	12.42754	0.63934	0.49001	423
36	0.97065	6.59036	6	22	1.18074	35.43159	12.94037	0.56399	0.50669	9884
37	0.98703	6.66471	6	22	1.10282	39.29782	13.45384	0.48853	0.52338	9320
38	0.99666	6.73826	6	22	1.13344	39.42570	13.97110	0.41309	0.54006	8766
39	0.99973	6.81104	6	22	1.25511	39.48469	14.49761	0.33778	0.55674	8222
40	0.99649	6.88305	6	22	1.33979	39.53780	15.03917	0.26274	0.57343	7688
41	0.98717	6.95434	6	22	1.28157	39.62313	15.59969	0.18807	0.59011	7164
42	0.97205	7.02492	6	22	1.15209	40.07966	16.17947	0.11389	0.60679	6650
43	0.95142	7.09481	6	22	1.05762	45.48675	16.77476	0.04030	0.62348	6146
44	0.92560	7.16404	6	22	1.05384	45.68556	17.37924	-0.03257	0.64016	5652
45	0.89489	7.23261	6	22	1.14518	45.75309	17.98691	-0.10463	0.65684	5168
46	0.85963	7.30055	6	22	1.24902	45.80488	18.59503	-0.17578	0.67353	4694
47	0.82015	7.36788	6	22	1.23826	45.87442	19.20541	-0.24590	0.69021	4230
48	0.77679	7.43461	6	22	1.13040	46.09338	19.82364	-0.31491	0.70690	3776
49	0.72989	7.50075	6	22	1.02720	51.54964	20.45639	-0.38271	0.72358	3332
50	0.67980	7.56633	6	22	0.99606	51.93852	21.10813	-0.44921	0.74026	2898
51	0.62684	7.63135	6	22	1.05622	52.02042	21.77862	-0.51432	0.75695	2474
52	0.57138	7.69583	6	22	1.16128	52.07314	22.46276	-0.57797	0.77363	2060
53	0.51373	7.75979	6	22	1.19110	52.13332	23.15330	-0.64007	0.79031	1656
54	0.45422	7.82323	6	22	1.11145	52.27002	23.84508	-0.70057	0.80700	1262
55	0.39319	7.88618	6	22	1.00697	56.76139	24.53811	-0.75941	0.82368	878
56	0.33093	7.94863	6	22	0.95465	58.17901	25.23765	-0.81652	0.84036	504
57	0.26777	8.01060	6	22	0.98568	58.28586	25.95108	-0.87187	0.85705	140
58	0.20400	8.07210	6	22	1.07961	58.34190	26.68343	-0.92543	0.87373	9844
59	0.13989	8.13315	6	22	1.13918	58.39670	27.43406	-0.97716	0.89041	9501
60	0.07574	8.19375	6	22	1.09200	58.49468	28.19676	-1.02705	0.90710	9168
61	0.01181	8.25391	6	22	0.99345	59.35344	28.96386	-1.07510	0.92378	8845
62	-0.05166	8.31364	6	22	0.92567	64.39396	29.73159	-1.12130	0.94046	8532
63	-0.11442	8.37296	6	22	0.93078	64.54815	30.50281	-1.16566	0.95715	8229
64	-0.17623	8.43186	6	22	1.00653	64.61059	31.28500	-1.20821	0.97383	7936
65	-0.23689	8.49036	6	22	1.08327	64.66280	32.08501	-1.24896	0.99051	7653

66	-0.29619	8.54846	6	22	1.06950	64.73968	32.90387	-1.28795	1.00720	7380
67	-0.35394	8.60618	6	22	0.98376	65.04797	33.73581	-1.32523	1.02388	7117
68	-0.40995	8.66352	6	22	0.90615	70.54363	34.57253	-1.36083	1.04056	6864
69	-0.46407	8.72049	6	22	0.88900	70.80524	35.40997	-1.39480	1.05725	6621
70	-0.51614	8.77709	6	22	0.94345	70.87858	36.25162	-1.42719	1.07393	6388
71	-0.56602	8.83334	6	22	1.02537	70.93052	37.10576	-1.45806	1.09061	6165
72	-0.61358	8.88924	6	23	1.04234	70.99507	37.97860	-1.48747	1.10730	5952
73	-0.65870	8.94479	6	23	0.97539	71.16681	38.86890	-1.51547	1.12398	5749
74	-0.70129	9.00000	6	23	0.89368	76.42810	39.76879	-1.54213	1.14066	5556
75	-0.74125	9.05488	6	23	0.85806	77.05332	40.67072	-1.56750	1.15735	5373
76	-0.77849	9.10944	6	23	0.89071	77.14515	41.57449	-1.59163	1.17403	5200
77	-0.81296	9.16367	6	23	0.96815	77.19910	42.48743	-1.61460	1.19071	5037
78	-0.84460	9.21759	6	23	1.00982	77.25635	43.41767	-1.63644	1.20740	4884
79	-0.87335	9.27120	6	23	0.96611	77.37173	44.36639	-1.65723	1.22408	4741
80	-0.89919	9.32451	6	23	0.88615	79.71290	45.32637	-1.67701	1.24076	4608
81	-0.92209	9.37751	6	23	0.83607	83.28430	46.28912	-1.69583	1.25745	4485
82	-0.94202	9.43023	6	23	0.84792	83.40930	47.25354	-1.71374	1.27413	4372
83	-0.95899	9.48265	6	23	0.91416	83.46789	48.22710	-1.73078	1.29081	4269
84	-0.97300	9.53479	6	23	0.97234	83.52117	49.21818	-1.74702	1.30750	4176
85	-0.98405	9.58665	6	23	0.95406	83.60779	50.22719	-1.76248	1.32418	4093
86	-0.99217	9.63823	6	23	0.88162	84.09334	51.24587	-1.77720	1.34086	4020
87	-0.99739	9.68954	7	23	0.82129	89.47706	52.26630	-1.79123	1.35755	3957
88	-0.99973	9.74059	7	23	0.81424	89.66948	53.28989	-1.80460	1.37423	3904
89	-0.99925	9.79137	7	23	0.86533	89.73633	54.32578	-1.81735	1.39091	3861
90	-0.99598	9.84189	7	23	0.93126	89.78814	55.38049	-1.82951	1.40760	3828
91	-0.99000	9.89216	7	23	0.93784	89.85844	56.45017	-1.84111	1.42428	3805
92	-0.98134	9.94218	7	23	0.87824	90.08545	57.52500	-1.85218	1.44097	3792
93	-0.97009	9.99195	7	23	0.81217	95.55414	58.60107	-1.86275	1.45765	3789
94	-0.95631	10.04148	7	23	0.78867	95.92299	59.68525	-1.87283	1.47433	3796
95	-0.94008	10.09076	7	23	0.82286	96.00376	60.78676	-1.88247	1.49102	3813
96	-0.92149	10.13981	7	23	0.88857	96.05635	61.90556	-1.89168	1.50770	3840
97	-0.90060	10.18863	7	23	0.91667	96.11698	63.03217	-1.90048	1.52438	3877
98	-0.87752	10.23721	7	23	0.87430	96.25710	64.16002	-1.90889	1.54107	3924