Project 3
Scientific Computing

Rebecca Selvaggini

June 2022

1 Multigrid 1d

In the first part of the project I have implemented a multigrid method for the solution of the following elliptic boundary value problem: let $\Omega = (-2, 2) \subset \mathbb{R}$ and let $f, g \in \mathcal{C}^0(\bar{\Omega})$ be the functions $f(x) = (x-1)^3$ and $g(x) = \frac{|x|^2}{10}$, find a function $u \in \mathcal{C}^2(\bar{\Omega})$ such that

$$u''(x) - u(x) = f(x) \quad \text{in} \quad \Omega \tag{1}$$

$$u(-2) = g(-2) \quad u(2) = g(2). \tag{2}$$

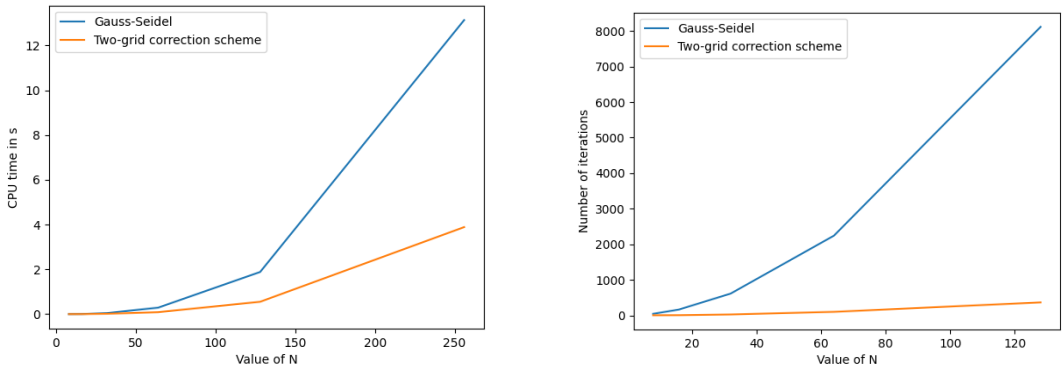
In order to find the solution u let

$$\hat{A_h}\hat{u_h} = \hat{f_h} \tag{3}$$

be the standard finite difference approximation to the given boundary value problem, where $\hat{A_h} \in \mathbb{R}^{N+1} \times \mathbb{R}^{N+1}$ and where we consider a uniform partitioning of the interval $[-2, 2]$ given by $x_0 = -2 < x_1 < ... < x_{N-1} < 2 = x_N$. From now on we usually assume that $N = 2^\ell$ for some integer $\ell \geq 1$.
The first function implemented is `gs_step_1d` which perform one step of the Gauss-Seidel method to solve the linear system (3), given an initial guess `uh` $= u_h^{(k)}$ and the right hand side of the equation `fh` $= \hat{f_h}$. The function will update `uh` to be the new solution $u_h^{(k+1)}$ and return as output the pseudo-residual $|u_h^{(k+1)} - u_h^{(k)}|_\infty$. The function `twogrid_step(uh,fh)` take as input the same numpy.array of the first one, and implement the two-grid correction scheme where instead of solving $\hat{A_{2h}}\hat{e_{2h}} = \hat{r_{2h}}$, we perform five Gauss–Seidel steps on the coarse grid. For both the functions we investigate the number of iterations and the time needed to find a solution with pseudo-residual less then $tol = 10^{-8}$, starting form the zero vector as initial guess. In Table 1 you can find the number of iteration (IT), the CPU time needed to perform it (in seconds) and the value of the pseudo-residual reached (residual), for different values of the mesh size $h = 4/N$ (in the table you find the value of ℓ, where $N = 2^\ell$). For both methods we can see that as the value of h decrease (i.e. the value of N increase) the number of iteration needed to reach a pseudo-residual less then 10^{-8} drastically increase, and consequently the time needed increase. However the two-gird correction scheme give an important improvement on both compared to the standard Gauss-Seidel method, as you can see in Figure 1.

	Gauss-Seidel			Two-grid		
ℓ	IT	CPU time	residual	IT	CPU time	residual
3	47	0.001995086669921875	7.848788285969022e-09	7	0.0019998550415039062	5.524753987629083e-10
4	168	0.01299905776977539	9.90554926971754e-09	9	0.003008604049682617	6.437632871225674e-09
5	616	0.05300259590148926	9.801360167926987e-09	29	0.016999483108520508	7.012209035650585e-09
6	2245	0.31158447265625	9.981448556573014e-09	103	0.09000515937805176	8.894776470924626e-09
7	8114	2.314413547515869	9.997855876520134e-09	369	0.5892281532287598	9.832912262197624e-09
8	28976	13.860627174377441	9.998154526513758e-09	1316	4.3507702350616455	9.997758620983177e-09

Table 1: Gauss-Seidel method and Two-grid correction scheme



(a) *CPU time* (b) *Number of iterations*

Figure 1: Gauss-Seidel method and two-grid correction scheme for different values of $N = 2^l$

The next function implemented is `v_cycle_step_1d` which take in input `uh` and `fh` as always, and two numbers `alpha1` and `alpha2` that give the number of, respectively, pre-smoothing and post-smoothing steps. The function depends on several routines as `I_2h(v2h)` that implement the prolongation operator, `I_h(vh)` that implement the full weighting restriction (as seen during the lectures) and the function `Auf(uh, fh)` that compute the vector `fh - Auh`, where `A` is the matrix of our linear system in (3).
The results of the investigation on the number of iterations and the CPU time needed for different values of ℓ and α_1, α_2 are given in Table 2 where you can find then number of iteration (IT) and the CPU time (in seconds) to have a pseudo-residual less then 10^{-8}, together with the pseudo-residual $|u_h^{(k+1)} - u_h^{(k)}|_\infty$ obtained in the last iteration.
As you can see in the table, increasing the value of ℓ (i.e. decreasing h) does not affect too much the number of iterations, but the CPU time increase as long as we increase ℓ.

α_1	α_2	ℓ	IT	CPU time	residual
1	1	3	7	0.0019986629486083984	1.016823070187911e-09
		4	8	0.0050144195556640625	4.142993947908735e-09
		5	8	0.005010366439819336	7.930703205261125e-09
		6	8	0.009000301361083984	3.725461994452672e-09
		7	8	0.01800060272216797	8.296853093447965e-09
		8	8	0.03400015830993652	3.5770645612220875e-09
		9	8	0.07300591468811035	1.4489407451634406e-09
		10	7	0.11900687217712402	3.448957786034157e-09
		11	7	0.2550196647644043	1.8311241967161607e-09
		12	7	0.5119023323059082	2.077941874123468e-09
		13	7	0.7969393730163574	2.0385089727348316e-09
		14	7	1.6390643119812012	2.0098861464035167e-09
		15	7	3.131812810897827	2.0034589542916592e-09
2	2	3	4	0.002002716064453125	7.299621795908706e-09
		4	5	0.004010677337646484	4.195958691610713e-09
		5	6	0.005014181137084961	2.110396302157369e-10
		6	6	0.011998891830444336	5.929862156861532e-10
		7	6	0.017999887466430664	4.339050785340248e-10
		8	5	0.033000946044921875	4.40734273382537e-09
		9	5	0.05700397491455078	4.307008250048483e-09
		10	5	0.10805726051330566	4.313898294139307e-09
		11	5	0.2281806468963623	4.3157031281992886e-09
		12	5	0.429995059967041	4.316153934258438e-09
		13	5	0.8520302772521973	4.316265955761622e-09
		14	5	1.6968178749084473	4.316294044404145e-09
		15	5	3.468492031097412	4.3163010388092005e-09

Table 2: V-cycle 1d

Next we have the function `full_mg_1d` which implement a full multigrid step which use the same prolongation operator as `v_cycle_step_1d` but the restriction operator `Injection`.
In the tables 3, 4, 5, 6 you can find the values of the CPU time (in seconds), of the pseudo-residual and of the residual $|f_h - A_h u_h|_\infty$ after one step of the full multigrid method for different values of ℓ and of the parameters α_1, α_2 and ν as specified in the caption of the tables.

ℓ	CPU time	pseudo-residual	residual
3	0.0	0.05428121079015735	0.1263162217298497
4	0.0019996166229248047	0.04955466114862506	0.24400927755758062
5	0.00299835205078125	0.027150271926458913	1.3118733991555303
6	0.00299835205078125	0.009656704628096868	2.4721163847927983
7	0.005000114440917969	0.0032834114343441856	3.362213308768446
8	0.008012533187866211	0.0009612285361641493	3.9371920841283554
9	0.021013975143432617	0.0002609143779873091	4.274821168943845
10	0.03200030326843262	6.808463680024746e-05	4.461994757341017
11	0.06599283218383789	1.7402783682729517e-05	4.5620353257108945
12	0.13425540924072266	4.400487928546415e-06	4.614246030221693
13	0.2526843547821045	1.1065181201863616e-06	4.641073377570137
14	0.46872401237487793	2.774429048169047e-07	4.6547195428283885
15	0.9753847122192383	6.946349551117947e-08	4.661616269513615

Table 3: Full multigrid with $(\alpha_1, \alpha_2, \nu) = (1,1,1)$

ℓ	CPU time	pseudo-residual	residual
3	0.0010020732879638672	0.005277309556834808	0.0024924665632575227
4	0.0010116100311279297	0.0023980750483256763	0.014194388146876236
5	0.0030088424682617188	0.0031856773633578417	0.09610353337686206
6	0.007012605667114258	0.001384058828477197	0.2146943051052972
7	0.011001110076904297	0.00044109679154542836	0.2988958050495967
8	0.017011404037475586	0.00012363247540467248	0.34802691448192036
9	0.035013675689697266	3.2671648293736144e-05	0.3744430195715722
10	0.06299328804016113	8.394210785134248e-06	0.3881399201200111
11	0.1262040138244629	2.1272040504061174e-06	0.395119510780205
12	0.2382955551147461	5.354048153716207e-07	0.3986447294591926
13	0.4719705581665039	1.3430302930883542e-07	0.40041688922792673
14	0.9398622512817383	3.363231476649631e-08	0.4013055303366855
15	1.8162224292755127	8.415152796814596e-09	0.40175053106213454

Table 4: Full multigrid with $(\alpha_1, \alpha_2, \nu)$ =(1,1,2)

ℓ	CPU time	pseudo-residual	residual
3	0.0009999275207519531	0.002008222417011929	0.008032889668049492
4	0.0020036697387695312	0.003255854322769647	0.05209366916431435
5	0.003999471664428711	0.002635850115619176	0.16869440739962727
6	0.005994558334350586	0.0010860964717104293	0.2780406967578699
7	0.00901031494140625	0.0003411054217082321	0.3492919518292297
8	0.015999555587768555	9.50101703552017e-05	0.3891616577749204
9	0.027998924255371094	2.5029236168400137e-05	0.41007900538306785
10	0.05201220512390137	6.420294222875267e-06	0.4207604021903535
11	0.09401655197143555	1.6256424553162674e-06	0.4261524158064276
12	0.18401288986206055	4.08993278910863e-07	0.42886053648544475
13	0.40028953552246094	1.0257184851480972e-07	0.43021751439664513
14	0.7770569324493408	2.568344570352820 7e-08	0.4308967161778128
15	1.4335987567901611	6.4259244036968255e-09	0.43123648688015237

Table 5: Full multigrid with $(\alpha_1, \alpha_2, \nu) = (2,2,1)$

ℓ	CPU time	pseudo-residual	residual
3	0.0010013580322265625	1.1309861073449667e-05	4.523944429379867e-05
4	0.0019989013671875	2.902896070944294e-05	0.00046446337135108706
5	0.005998849868774414	4.473443739749783e-05	0.002863003993439861
6	0.0069980621337890625	2.349144883173615e-05	0.006013810900952876
7	0.018001317977905273	8.100028431656803e-06	0.008294429114016566
8	0.02900099754333496	2.3471695165741546e-06	0.009614006339830894
9	0.05300092697143555	6.294473357260078e-07	0.010312865148989658
10	0.10200333595275879	1.628164317257763e-07	0.010670337669580476
11	0.2060239315032959	4.13922867092964e-08	0.010850739607121795
12	0.3640894889831543	1.0434437425210774e-08	0.010941300657577813
13	0.735865592956543	2.6194246771638063e-09	0.010986663517542183
14	1.4597198963165283	6.562093091133647e-10	0.011009364374331199
15	2.785592794418335	1.6422152526729406e-10	0.011020720003216411

Table 6: Full multigrid with $(\alpha_1, \alpha_2, \nu) = (2,2,2)$

The last request for the one-dimensional case is the plotting of an approximated solution of the problem discussed above. In Figure 2 you can find the solution `uh` obtained by iterating the V-Cycle method up to a solution with pseudo-residual less then 10^{-12} with fixed value $\ell = 8$, $\alpha_1 = \alpha_2 = 2$. For the solution obtained we can estimate that $\min u_h(x) = -3.694360963138051$.

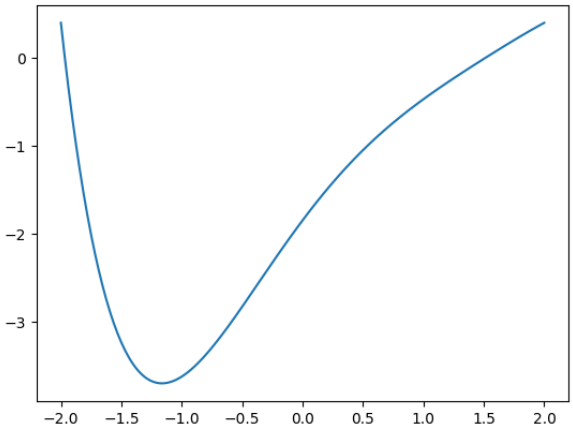


Figure 2: Approximation of solution u with V-Cycle method

2 Multigrid 2d

In this section we consider the multigrid method in the two-dimensional case. We have $\Omega = (-2,2)^2 \subset \mathbb{R}^2$, we fix $f(x) = f(x_1,x_2) = x_1^2 - x_2^2$ and $g(x) = g(x_1,x_2) = \frac{x_1^2+x_2^2}{10}$. With the same assumption on the first case, we repeat the simulations on the functions implemented.
In Table 7 you can find the results on the Gauss-Seidel method: the number of iterations (IT) and the CPU time to obtain a solution with pseudo-residual less then 10^{-8} together with the pseudo-residual at the last iteration, for different values of $\ell = 2,\dots,6$.
In Table 8 you can find the same results for the function `v_cycle_step_2d` for two different sets of value of $(\alpha_1,\alpha_2) = (1,1),(2,2)$.
In Tables 9, 10, 11, 12 you can find the CPU time to perform one full-multigrid step, with the pseudo-residual and residual obtained, for different values of ℓ and different values of (α_1,α_2,ν) as stated in the caption of each table.
In the end in Figure 3 you can find a plot of the solution obtained by applying the V-Cycle method (starting from the zero vector) since we find a pseudo-residual smaller then 10^{-12}, where we fixed $\ell = 8$. An estimate of the minimum of the function computed is $\min u(x_1,x_2) = -0.04150939155234839$.

ℓ	IT	pseudo-residual	CPU time
2	17	7.636066501337524e-09	0.0009999275207519531
3	60	9.289224500719229e-09	0.009001493453979492
4	216	9.394708955223763e-09	0.16592645645141602
5	777	9.886668095315798e-09	2.1902685165405273
6	2783	9.986929200289651e-09	27.886791706085205

Table 7: Gauss-Seidel method 2d

α_1	α_2	ℓ	IT	pseudo-residual	CPU time
1	1	2	6	4.642810358390648e-09	0.00199127197265625
		3	9	2.666458998046295e-09	0.008000373840332031
		4	10	2.417379324493041e-09	0.04200267791748047
		5	10	2.803542786811164e-09	0.16102051734924316
		6	10	3.0225845160103404e-09	0.6040449142456055
		7	10	4.422206600906975e-09	2.571983575820923
		8	10	5.3505865293956845e-09	9.610964059829712
2	2	2	4	3.416228411268207e-11	0.0009953975677490234
		3	6	2.6820405896188504e-09	0.008997917175292969
		4	6	7.926279761011346e-09	0.038002729415893555
		5	7	8.316240474037784e-10	0.14826130867004395
		6	7	1.2686598438449437e-09	0.6049520969390869
		7	7	1.6600080199324907e-09	2.4352219104766846
		8	7	1.838143304233597e-09	9.938615560531616

Table 8: V-Cycle method, two-dimensional case

ℓ	CPU time	pseudo-residual	residual
2	0.0009958744049072266	0.04193744799999999	0.03196053760000017
3	0.0010030269622802734	0.05062909824074574	0.20487454839655594
4	0.006999492645263672	0.016904812446795175	0.24475979309504847
5	0.027011394500732422	0.004598384782798559	0.29429662609910245
6	0.09200191497802734	0.001235232641809636	0.3162195563032526
7	0.4222724437713623	0.0003189257724263705 5	0.3265799909645466
8	1.392103910446167	8.087032322828813e-05	0.33124484394329556

Table 9: Full 2d-multigrid with $(\alpha_1, \alpha_2, \nu) = (1,1,1)$

ℓ	CPU time	pseudo-residual	residual
2	0.0010008811950683594	0.0018444765647399608	0.0028057531294799487
3	0.00299835205078125	0.00398873862389288	0.016957936569347076
4	0.013015031814575195	0.0015494472410443466	0.025096981748455427
5	0.04300236701965332	0.000454718566571477	0.029101988260576306
6	0.16403722763061523	0.00012428086037025166	0.031815900254756
7	0.6820902824401855	3.217653806386611e-05	0.03294877497728521
8	2.6287174224853516	8.16484447951904e-06	0.03344320298833736

Table 10: Full 2d-multigrid with $(\alpha_1, \alpha_2, \nu) = (1,1,2)$

ℓ	CPU time	pseudo-residual	residual
2	0.001001596450805664	0.0025501386880000254	0.004129134399999956
3	0.004000186920166016	0.0054187161913315934	0.03365030173170469
4	0.006997823715209961	0.0018408745942332938	0.041818278618330984
5	0.037011146545410156	0.0005225605560839897	0.04356167552870627
6	0.12779450416564941	0.00013785533655080728	0.044760341163360806
7	0.4753401279449463	3.5328335156237234e-05	0.04551460215310499
8	1.8876299858093262	8.93659314499251e-06	0.045925385089503834

Table 11: Full 2d-multigrid with $(\alpha_1, \alpha_2, \nu) = (2,2,1)$

ℓ	CPU time	pseudo-residual	residual
2	0.0010006427764892578	2.971798662837477e-05	2.715805202657684e-05
3	0.003998517990112305	0.00019384360196836314	0.001230101896276814
4	0.020003557205200195	5.8251308926310363e-05	0.00146373424122892
5	0.06900715827941895	1.2610385717926054e-05	0.0011176673498365375
6	0.2356739044189453	3.3957771836332085e-06	0.0010424296221884788
7	0.9359564781188965	8.893123585596019e-07	0.0010169582794219423
8	3.699510097503662	2.2638169272504172e-07	0.0010285213631959778

Table 12: Full 2d-multigrid with $(\alpha_1, \alpha_2, \nu) = (2,2,2)$

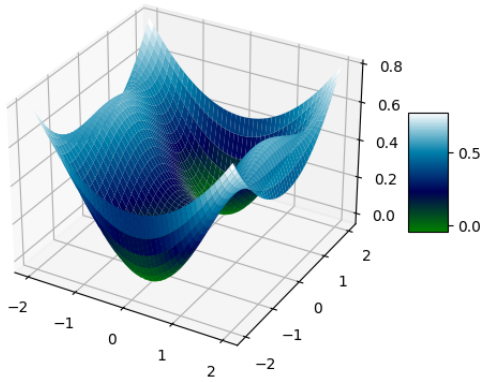


Figure 3: Approximation of the solution u with 2d-V-Cycle method