A Multi-Grid Continuation Algorithm for Nonlinear Parameter Dependent Systems of Equations

Werner Vogt Sebastian Schmidt

14th November 2005

Abstract

A nonlinear Multi-Grid Predictor-Corrector algorithm is developed using a modified Full Approximation Scheme. For this modification an extra predictor is introduced utilizing data obtained from the coarse grid correction. Convergence of this algorithm is proven for each continuation step and the performance of the algorithm is tested for different applications.

MSC: 65M55, 65N55, 65H20

Key Words: Multi-grid method, continuation method, nonlinear systems

Contents

1 Introduction

Consider the nonlinear, parameter dependent system of equations

$$N_n(u_n, \lambda) = 0,\ N_n : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n, \tag{1}$$

with $n \in \mathbb{N}$ assumably large, arising from the operator problem

$$N(u, \lambda) = 0,\ N : \mathcal{X} \times \mathbb{R} \longrightarrow \mathcal{Y} \text{ with } \mathcal{X}, \mathcal{Y} \text{ Banach Spaces.} \tag{2}$$

We seek approximate solutions $u_n(\lambda)$ for $\lambda \in [a, b]$. For this, two types of algorithms are currently beeing developed:

- Jacobian-free Newton-Krylov methods (JFNK, see [1]) and
- Nonlinear Multi-Grid (MG) methods such as the Full Approximation Scheme (FAS) in [4] or algorithms of nested type as in [5].,

as well as combinations of the above using FAS as a preconditioner for JFNK, studied in [1]. The subject of this paper is the modification of nonlinear MG methods to establish an efficient algorithm for continuation of parameter dependent problems.

Efficient Multi-Grid (MG) methods to solve the parameter independent version of (1) have already been developed (cf. [4]). As an obvious approach to solving (1) on a parameter interval $[a, b]$ one could, for every parameter λ_i, apply a MG method as a corrector to some $u(\lambda_{i-1})$. However, this treats the MG methods as a "black box", hence denying their ability to use coarse grid corrections from former parameters. Methods for utilizing that data are presented by introducing a concept we shall call Coarse Grid Prediction (CGP).

We define the sets

$$\mathbb{N}_{\geq 0} := \{m \in \mathbb{Z} \mid m \geq 0\},\ \mathbb{R}_{\geq 0} := \{r \in \mathbb{R} \mid r \geq 0\},\ \mathbb{N}_{>0} := \{m \in \mathbb{Z} \mid m > 0\}$$

Because Multi-Grid (MG) algorithms involve multiple grids, a distinguishing notation is important.

Let $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}_{>0}$ the space on which a solution to (2) is sought. We define a family of grid sizes $\{h_l\}_{l \in \mathbb{N}_{\geq 0}}$ with $h_l > h_{l+1} > 0$ for all $l \in \mathbb{N}_{\geq 0}$, a family of infinite grids

$$Q_l := \{x \in \mathbb{R}^d \,|\, x = (\alpha_1 h_l, \ldots, \alpha_d h_l),\ \alpha = (\alpha_1, \ldots, \alpha_d)^T \in \mathbb{Z}^d\},\ l \in \mathbb{N}_{\geq 0}, \tag{3}$$

and a family of subsets

$$\Omega_l := Q_l \cap \Omega,\ l \in \mathbb{N}_{\geq 0},$$

with grid points (depending on the dimension d of the domain)

$$\omega_{l,i_1,\ldots,i_d} \in \Omega_l, \tag{4}$$

as well as

$$n_l := |\Omega_l|,\ l \in \mathbb{N}_{\geq 0}. \tag{5}$$

By N_l we denote a family of nonlinear, parameter dependent systems of equations

$$N_l(u_l, \lambda) = f_l\ , N_l : \mathbb{R}^{n_l} \times \mathbb{R} \longrightarrow \mathbb{R}^{n_l},\ f_l \in \mathbb{R}^{n_l},\ l \in \mathbb{N}_{\geq 0}, \tag{6}$$

where, for $l \in \mathbb{N}_{\geq 0}$, N_l denotes some discretization of N from (1) on the grid Ω_l with the *grid function* u_l.

The parameter dependent problem. In the further scope of this paper, the problem

$$N_L(u_L, \lambda) = 0, \tag{7}$$

for $L \in \mathbb{N}_{>0}$, $n_L = n$ (from (1)) is to be solved, i.e. a solution for the discretized version of (2) is sought on Ω_L (the *finest grid*) for some finite subset of parameter values $\{\lambda_j,\ j \in I \subset \mathbb{N}\} \subset [a,b] =: \Lambda \subset \mathbb{R}$.

We furthermore call

$$N_L(u_L) = 0 \tag{8}$$

the *parameter independent problem.*

Definitions 1.1 (Restriction and Prolongation Functions).

We define functions that transfer grid functions from fine to coarse grids and vice versa.

(i) By $R_l, \hat{R}_l$ we define two families of linear functions (*restriction functions*). For $l \in \mathbb{N}_{>0}$, these map as

$$R_l, \hat{R}_l : \mathbb{R}^{n_l} \longrightarrow \mathbb{R}^{n_{l-1}}.$$

(ii) By P_l we define a family of linear functions (*prolongation functions*). For $l \in \mathbb{N}_{>0}$, these map as

$$P_l : \mathbb{R}^{n_{l-1}} \longrightarrow \mathbb{R}^{n_l}.$$

Notation and Parameter Dependency. Dependency of any notation on a parameter λ is written as an argument. For example, if we mean to denote the i^{th} component of a solution u_l for a specific parameter λ_j after m iterations we write $u^m_{l,i}(\lambda_j)$.

If an algorithm ALG with Parameters p_1 and p_2 is called with any of the parameters being zero, then we add a subscript to the 0, e.g. $ALG(p_1, 0_{p_2})$, to clarify the origin of the zero. In all other cases, notation is chosen to clarify the origin of the parameter.

2 The Multi-Grid Predictor-Corrector Algorithm

2.1 The Nonlinear Multi-Grid Concept

Temporarily and for ease of notation assume (7) to be independent of the parameter λ. Furthermore, assume $0 < l \leq L$ and the existence of an *initial solution* u^0_l on Ω_l of

$$N_l(u_l) = f_l. \tag{9}$$

Assuming (9) has at least one solution, we may state that there exists an *exact correction* $v_l \in \mathbb{R}^{n_l}$ to u^0_l so that

$$N_l(u^0_l + v_l) = f_l, \tag{10}$$

holds, i.e. v_l improves the *initial approximation* u^0_l. Calling $d_l = f_l - N_l(u^0_l)$ the defect, we may rewrite (10) as

$$N_l(u^0_l + v_l) - N_l(u^0_l) = d_l, \tag{11}$$

which is the so called exact defect equation on Ω_l (as in [10, (5.3.11-12), p. 155]). Trying to approximate (11) on a coarser grid Ω_{l-1} (and essentially solving for the correction v_{l-1} on Ω_{l-1}) might lead to the following problem:

Assume that v_l, as the difference between the exact solution to (9) and the initial approximation u^0_l, is non smooth (for a more accurate description of how to measure the smoothness of the error of an approximation see [4, Section 2.6.3]).

Then approximating a non smooth v_l on Ω_l would possibly loose much information (for this, see [10, p. 15-18]). Many iterative methods (Gauss-Seidel etc.), if appropriately applied, have a strong smoothing effect on the error of any approximation. Note that this does not mean the iteration makes the error smaller, it just smoothes the error.

As a consequence, we do not try to approximate the correction to u_l^0 on the coarser grid Ω_{l-1}, but rather do so for the correction to $\bar{u}_l^0$ resulting from applying some iterative smoothing method to u_l^0.

Using the restriction functions from definition 1.1 we approximate (11) on the coarser grid Ω_{l-1} by

$$N_{l-1}(\hat{R}_l\bar{u}_l^0 + v_{l-1}) - N_{l-1}(\hat{R}_l\bar{u}_l^0) = R_l d_l = R_l(f_l - N_l(\bar{u}_l^0)),$$

which is the determining equation for v_{l-1}, the *coarse grid correction*. Hence, we have reduced (9) to

$$N_{l-1}(w_{l-1}) = f_{l-1}, \tag{12}$$

with $w_{l-1} := \hat{R}_l\bar{u}_l^0 + v_{l-1}$ and

$$f_{l-1} := R_l(f_l - N_l(\bar{u}_l^0)) + N_{l-1}(\hat{R}_l\bar{u}_l^0), \tag{13}$$

which is to be solved for w_{l-1} on the coarser grid Ω_{l-1}. Assuming we have obtained a solution w_{l-1} of (12), we may compute $v_{l-1} = w_{l-1} - \hat{R}_l\bar{u}_l^0$ and prolongate it to $v_l = P_l v_{l-1}$. Simply adding v_l to $\bar{u}_l^0$ gives a new and corrected solution.

Remark 2.1. The use of two different restriction functions R and $\hat{R}$ is of a rather technical nature. In some applications it may happen that the coarse grid function has to be restricted from the finer grid with a different method than the defect. See [4, Note 9.3.3, p. 186] for further explanation. The functions R and $\hat{R}$ are identical throughout all applications in this paper.

Concluding this discussion, a nonlinear MG algorithm is divided into the three parts *smoothing, restriction* and *prolongation* as well as *coarse grid solving*.

Smoothing Iterations

In general, the nonlinear counterparts of any of the well known linear iterative algorithms (Jacobi, Gauss-Seidel, Richardson, et cetera) may be used as smoothing iterations as long as they have the desired effect of smoothing as previously mentioned.

Certainly, there are precise conditions that determine whether a smoothing algorithm is suited for a specific problem. That criteria is called *smoothing property*, defined in (42).

Definition 2.2. Fix $l \in \mathbb{N}_{>0}$. By

$$\mathcal{S}_l : \mathbb{R}^{n_l} \times \mathbb{R}^{n_l} \times \mathbb{R} \longrightarrow \mathbb{R}^{n_l},\ u_l \mapsto \mathcal{S}_l(u_l, f_l, \lambda) \tag{14}$$

we denote one smoothing iteration on Ω_l with starting value u_l, right hand side f_l and the parameter value λ. The process of smoothing with ν iterations is then denoted by $u_l \mapsto \mathcal{S}_l^{(\nu)}(u_l, f_l, \lambda)$.

Fix $\lambda \in [a, b]$ and disregard its notation where applicable. We give the nonlinear version of the Gauss-Seidel iteration for given u_l^0 in algorithm 2.1.

Algorithm 2.1 Gauss-Seidel smoothing iteration

1: $u_l = u_l^0$
2: **for** $i := 1(1)n_l$ **do**
3: $\quad u_{l,i} = u'_{l,i}$ with
4: $\quad N_{l,i}(u_{l,1}, \dots, u_{l,i-1}, u'_{l,i}, u^0_{l,i+1}, \dots) = f_{l,i}$
5: **end for** ▷ $N_{l,i}$, $f_{l,i}$ the i^{th} components of N_l, f_l.

Because line 4 in algorithm 2.1 may be nonlinear in $u'_{l,i}$ it has to be solved approximately, for instance by applying one step of Newtons Method. That results in a modified version of algorithm 2.1.

Algorithm 2.2 Gauss-Seidel-Newton smoothing iteration

1: **function** $u_l = \mathcal{S}_l^{GS}(u_l^0, f_l)$
2: $\quad u_l = u_l^0$
3: $\quad$ **for** $i := 1(1)n_l$ **do**
4: $\quad\quad u_{l,i} = u_{l,i} - (N_{l,i}(u_l) - f_{l,i}) \cdot \left[\frac{\partial N_{l,i}}{\partial u_{l,i}}(u_l)\right]^{-1}$, $\quad\triangleright$ assuming $\frac{\partial N_{l,i}}{\partial u_{l,i}}(u_l) \neq 0$.
5: $\quad$ **end for**
6: **end function**

Remark 2.3 (**Parameters for algorithm 2.2**).

$\mathbf{u_l^0}$	Starting value
$\mathbf{f_l}$	Right hand side of N_l, see (6)

The solution of nonlinear equations is completely avoided when we use smoothing iterations that only rely on evaluations of N_l. Such a smoothing method is proposed in [4, (9.3.5), p. 185] and given as

Algorithm 2.3 Richardson smoothing iteration

1: **function** $u_l = \mathcal{S}_l^{RIC}(u_l^0, f_l)$
2: $\quad u_l = u_l^0 - \omega h_l^\alpha \left[N_l(u_l^0) - f_l\right]$ $\quad\triangleright$ α the consistency order of N_l, $\omega \in (0, \frac{1}{2}]$
3: **end function**

Coarse Grid Solution

No actual solution of $N_L(u_L) = 0$ is found up to this point. Once the grid Ω_0 is reached, one has to actually solve

$$N_0(w_0) = f_0 \tag{15}$$

on Ω_0. This is achieved by the so called *coarse grid solver*, which can be any algorithm suitable for solving (15). Because the grid Ω_0 is in most scenarios much coarser than the grid Ω_L on which the actual solution is sought, one may choose the coarse grid solver mainly by criteria such as stability and robustness as opposed to computational cost. The Newton Method is a good choice in this respect. Even the algorithm used for smoothing, if convergent, is an applicable choice which, together with algorithm 2.3 for smoothing, would result in an easy to implement MG algorithm.

For the problem $N_0(u_0, \lambda) = f_0$ we define a solver

$$\Phi : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \times \mathbb{R} \longrightarrow \mathbb{R}^{n_0},\ u_0 = \Phi(u_0^0, f_0, \lambda), \tag{16}$$

with starting value u_0^0 and parameter value λ on the coarsest grid Ω_0. The implementation of Φ for the test problems in this paper is Newtons Method with approximated Jacobian.

Full Approximation Scheme The previous discussion shall be concluded in the recursive algorithm 2.5. It describes (for parameter $l = L$) one iteration of the FAS and is based on [10, p. 157]. In this paper algorithm 2.5 will be referred to as the *standard FAS algorithm*.

Note that algorithm 2.5 cannot readily be applied as a MG solver. Remark 2.7 states an algorithm for that purpose.

Remark 2.4. Before any modification to existing MG algorithms can be made, one has to decide which algorithm to modify. The setting in this paper are parameter dependent problems, for which a solution is to be found on a parameter interval through continuation. We assume the existence of (or at least an efficient way to find a solution for) the initial parameter value λ_0. Under these conditions, we favored the FAS above the nested iteration from [4, Section 9.3.4, p. 188f]. In [4, Chapter 13] a continuation algorithm based on the nested iteration is given by algorithm [4, (13.2.5)].

Algorithm 2.5. $u_l^{m+1} = FASCYCLE(l, \gamma, u_l^m, f_l, \nu_1, \nu_2)$

(1) **Presmoothing**

(1a) Perform ν_1 smoothing iterations (14) (using e.g. Algorithm 2.2).

$$\bar{u}_l^m = \mathcal{S}_l^{(\nu_1)}(u_l^m, f_l) \tag{17}$$

(2) **Coarse grid correction**

(2a) Compute the restricted defect $d_{l-1} = R_l(f_l - N_l(\bar{u}_l^m))$.
(2b) Restrict $\bar{u}_l^m$ $u_{l-1} = \hat{R}_l \bar{u}_l^m$.
(2c) Compute the right hand side $f_{l-1} = d_{l-1} + N_{l-1}(u_{l-1})$.
(2d) Set the initial value $w_{l-1}^0 = u_{l-1}$ for

$$N_{l-1}(w_{l-1}) = f_{l-1}. \tag{18}$$

(2e) Solve (18):

$$w_{l-1} = \begin{cases} \gamma \text{ iterations of } FASCYCLE(l-1, \gamma, w_{l-1}^0, f_{l-1}, \nu_1, \nu_2) & , l > 1 \\ \Phi(w_0^0, f_0) \text{ by (16)} & , l = 1. \end{cases} \tag{19}$$

(2f) Compute the correction $v_{l-1} = w_{l-1} - u_{l-1}$.
(2g) Prolongate the correction $v_l = P_l v_{l-1}$.
(2h) Compute corrected approximation on Ω_l $u_l^{CGC} = \bar{u}_l^m + v_l$.

(3) **Postsmoothing**

(3a) Perform ν_2 smoothing iterations (14)

$$u_l^{m+1} = \mathcal{S}_l^{(\nu_2)}(u_l^{CGC}, f_l). \tag{20}$$

Remark 2.6 (**Parameters for algorithm** (2.5))**.**

$\mathbf{l}$	Level which determines the space Ω_l for the current computation. This parameter is necessary because $FASCYCLE$ is called recursively in (19)
γ	Number of Iterations for the recursive call of $FASCYCLE$ (see (19))
$\mathbf{u_l^m}$	Initial value for the given iteration of $FASCYCLE$
$\mathbf{f_l}$	Right hand side of N_l, see (6)
$\boldsymbol{\nu_1}$	Number of presmoothing iterations, see (17)
$\boldsymbol{\nu_2}$	Number of postsmoothing iterations, see (20)

Algorithm 2.4 Iteration of algorithm 2.5

1: $u^{new} = u^0$
2: **repeat**
3: $u^{old} = u^{new}$
4: $u^{new} = FASCYCLE(L, \gamma, u^{old}, N_L, 0_{f_L}, \nu_1, \nu_2)$ ▷ use algorithm 2.5
5: **until** $||u^{new} - u^{old}||_2 \leq \varepsilon(||u^{new}|| + 1.0)$

Remark 2.7. For actually solving (8), one needs to define some initial solution u^0 on Ω_L, some tolerance ε as well as parameters γ, ν_1 and ν_2 and iteratively compute a solution as in algorithm 2.4 so that the FAS can be seen as a standard nonlinear iterative algorithm. As such it can be used as a "black box" corrector for Predictor-Corrector (PC) continuation algorithms. However, a closer look at $FASCYCLE$ (algorithm 2.5) makes clear that each iteration divides into many different parts where information for continuation could be used. That will be done in section 2.2.

2.2 The Adapted Predictor-Corrector Algorithm

The Full Approximation Scheme (FAS) (algorithm 2.5) in form of algorithm 2.4 provides a MG method for a corrector. Accompanied with a predictor that gives an initial approximation for a parameter value λ_j based on the solutions for previous parameters, we have a basic Predictor-Corrector (PC) algorithm at hand.

A Modification of the Full Approximation Scheme

Starting with that in mind, we will take a more detailed look at the FAS and introduce an additional predictor to algorithm 2.5 for the correction v_L from step (2f) in algorithm 2.5 to regard the existence of a parameter λ.

Definition 2.8 (Predictor functions). Fix $k, \hat{k} \in \mathbb{N}_{>0}$. Those shall be the numbers of previous values used for the prediction. Define, for L from (7) (hence $n = n_L$),

$$\Psi^k : \underbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}_{k} \longrightarrow \mathbb{R}^n \quad \text{and} \quad \hat{\Psi}^{\hat{k}} : \underbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}_{\hat{k}} \longrightarrow \mathbb{R}^n$$

where Ψ^k shall be the predictor for the starting value $u_l^0(\lambda_j)$ of the MG iterations and $\hat{\Psi}^{\hat{k}}$ the predictor for the first coarse grid correction $v_l(\lambda_j)$ within the MG iterations.

The modified computation for the coarse grid correction using prediction

As a starting value for solving (12), the FAS uses $\hat{R}_l \bar{u}_l^m$. While this seems to be the logical choice for the parameter independent problem (8), it is an occasion where, given the parameter dependent problem (7), data from the computation for the previous parameter(s) may be reused.

The following lemma 2.9 gives an idea of how this may be achieved.

Lemma 2.9. In algorithm 2.5 (FASCYCLE), let $\nu_1 = \nu_2 = 0$ (i.e., disregard smoothing). Then computing u_L^{m+1} (the $(m+1)$st iteration of the solution to (8)) by applying $FASCYCLE$ to u_L^m is equivalent to

$$u_L^{m+1} = u_L^m + P_L(P_{L-1}(\dots P_1(v_0^m)\dots)) \tag{21}$$

Put plainly, if we disregard smoothing, then $FASCYCLE$ just adds the L-fold prolonged coarsest grid correction to the initial solution u_L^m to obtain u_L^{m+1}.

Lemma 2.9 suggests to use the first correction that has been applied on the finest grid Ω_L in the MG solver for previous values of λ as a first correction in the computation for the current λ and we call this *Coarse Grid Prediction (CGP)*. Therefore, we add some value v_L^P predicted by $\hat{\Psi}$ to the initial solution u_L^0 predicted by Ψ.

To be precise, assume we have solved (7) for parameters $\lambda_{j-k} \dots \lambda_{j-1}$ and some $k \in \mathbb{N}_{>0}$. Then, in the first iteration ($m = 0$) of solving (7) for λ_j we reformulate the coarse grid correction (12) on Ω_{L-1} (using the predictor $\hat{\Psi}$ from Definition 2.8) as

$$N_{L-1}(w_{L-1}(\lambda_j), \lambda_j) = f_{L-1}(\lambda_j), \tag{22}$$

with the initial approximation $w_{L-1}^0(\lambda_j) := \hat{R}_L \bar{u}_L^{CGP}$.

The CGP value is used in $\bar{u}_L^{CGP}$ as

$$\bar{u}_L^{CGP} := \mathcal{S}_L^{(\nu_1)} \left(\bar{u}_L + \hat{\Psi}^k(v_L(\lambda_{j-k}), \dots, v_L(\lambda_{j-1})) \right) \tag{23}$$

$$\bar{u}_L := \mathcal{S}_L^{(\nu_1)}(u_L^m(\lambda_j), f_L(\lambda_j)), \tag{24}$$

where the smoothing in (23) has to be applied to guarantee a smooth initial approximation as discussed in section 2.1. The right hand side on the grid Ω_{L-1} is modified from (13) to

$$f_{L-1}(\lambda_j) := R_L(f_L - N_L(\bar{u}_L^{CGP}, \lambda_j)) + N_{L-1}(\hat{R_L} \bar{u}_L^{CGP}, \lambda_j).$$

As a possible predictor $\hat{\Psi}$ in (23) one could use for example

$$\hat{\Psi}^1(v_L(\lambda_{j-1})) := v_L(\lambda_{j-1}), \tag{25}$$

which appears to be a reasonable choice as suggested by Lemma 2.9. However, also predictors of order 2 or higher may be used here. Those are usually given through interpolation formulas as in [2, Section 6.3.5, p. 56ff].

In subsequent iterations ($m > 0$), $\hat{R}_l \bar{u}_L$, with $\bar{u}_L$ as in (20) shall be used as the starting value $w^0_{L-1}(\lambda_j)$ for the coarse grid correction, conforming to the standard FAS iteration.

Developing a Predictor-Corrector Algorithm

Based on the previous thoughts of this section, we develop a Predictor-Corrector (PC) algorithm using CGP (23).

We introduce an algorithm that has strong connections to the FAS (algorithm 2.5). The modifications will be solely useful for continuation, in case of $\lambda_0 = a = b$, the two algorithms coincide. Therefore the modified FAS will be embedded into a PC continuation algorithm 2.12.

As mentioned above, for computations on the parameter λ_j we use solutions of (7) (*fine grid solutions*)

$$u_L(\lambda_{j-k}), \ldots, u_L(\lambda_{j-1}) \ , k \in \mathbb{N}_{>0} \tag{26}$$

and *prolongated coarse grid corrections*

$$v_L(\lambda_{j-\hat{k}}), \ldots, v_L(\lambda_{j-1}). \tag{27}$$

from computations on former parameters for prediction.

Assume that, given some $k, \hat{k}, \nu_1, \nu_2, \gamma \in \mathbb{N}$, $k, \hat{k}, \gamma > 0$, $0 < l \leq L$, the values from (26),(27) have been computed for (7).

Then, as a core for a PC continuation algorithm for solving (7) we present Algorithm 2.10, where $PCFASCYCLE$ stands for Predictor-Corrector (PC) Full Approximation Scheme (FAS) Cycle.

Algorithm 2.10. $u_l^{m+1} = PCFASCYCLE(l, \gamma, u_l^m, f_l, \nu_1, \nu_2, k, \lambda, m)$

(1) **Presmoothing and Coarse Grid Prediction (CGP)**

(1a) Apply ν_1 smoothing iterations (14): $\bar{u}_l^m = \mathcal{S}_l^{(\nu_1)}(u_l^m, f_l, \lambda)$.

(1b) Predict a coarse grid correction (use, for example, (25)).

$$\bar{u}_l^{CGP} = \begin{cases} \mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^m + \hat{\Psi}^k(v_l(\lambda_{j-k}), \ldots, v_l(\lambda_{j-1})), f_l\right) & , l = L \text{ and } m = 0 \\ \bar{u}_l^m & , l < L \text{ or } m > 0. \end{cases} \tag{28}$$

(2) **Coarse grid correction**

(2a) Compute the restricted defect $d_{l-1} = R_l(f_l - N_l(\bar{u}_l^{CGP}, \lambda))$.

(2b) Restrict $\bar{u}_l^{CGP}$ $u_{l-1} = \hat{R}_l \bar{u}_l^{CGP}$.

(2c) Compute the right hand side $f_{l-1} = d_{l-1} + N_{l-1}(u_{l-1}, \lambda)$.

(2d) Set the initial value $w_{l-1}^0 = u_{l-1}$ for

$$N_{l-1}(w_{l-1}, \lambda) = f_{l-1}. \tag{29}$$

(2e) Solve (29) (Φ from (16)):

$$w_{l-1} = \begin{cases} \gamma \text{ iterations of } PCFASCYCLE(l-1, \gamma, w_{l-1}^0, f_{l-1}, \nu_1, \nu_2, \lambda) & , l > 1 \\ \Phi(w_0^0, f_0, \lambda) \quad \text{by (16)} & , l = 1. \end{cases} \tag{30}$$

(2f) Compute the correction $v_{l-1} = w_{l-1} - u_{l-1}$.

(2g) Interpolate the correction $v_l = P_l v_{l-1}$.

(2h) Compute corrected approximation on Ω_l $u_l^{CGC} = \bar{u}_l^{CGP} + v_l$.

(3) **Postsmoothing**

(3a) Perform ν_2 smoothing iterations (14): $u_l^{m+1} = \mathcal{S}_l^{(\nu_2)}(u_l^{CGC}, f_l, \lambda)$.

(4) **Storage of Coarse Grid Prediction (CGP) value**

(4a) If $l = L$ and $m = 0$ then store the accumulated correction for future prediction:

$$v_L(\lambda) = u_L^{m+1} - \bar{u}_L^m. \tag{31}$$

Remark 2.11 (Parameters for Algorithm 2.10).

k	Degree of coarse grid predictor $\hat{\Psi}$
λ	Current parameter value
m	Index to current iteration (m$\geq$ 0)

For Parameters $l, \gamma, u_l^m, f_l, \nu_1, \nu_2$ see Remark 2.6.

We use Algorithm 2.10 to construct a simple PC continuation algorithm (algorithm 2.12) for solving (7) for $\lambda \in [a, b]$. A constant stepsize is used and no specific predictor is given.

Algorithm 2.12. $PCFASCONT(a, b, h_\lambda, u_L(a), \gamma, \nu_1, \nu_2, k, \hat{k}, \varepsilon)$

(0a) Set $\lambda_1 = a + h_\lambda$
(0b) Set the counter for continuation steps $i = 1$;

(1) **Predictor step**

(1a) $p = \min(i, k)$
(1b) $u_L^0(\lambda_j) = \Psi^p(u_L(\lambda_{j-p}), \ldots, u_L(\lambda_{j-1}))$

(2) **Corrector step**

(2a) $m = 0$
(2b) **do**
$u^{old} = u_L^m(\lambda_j)$
$u_L^{m+1}(\lambda_j) = PCFASCYCLE(l, \gamma, u^{old}, N_l, 0_{f_L}, \nu_1, \nu_2, \hat{k}, \lambda_j, m)$
$m = m + 1$
while $||u_L^m(\lambda_j) - u^{old}||_2 > \varepsilon(||u^{new}||_2 + 1.0)$

(3) **Continuing** λ

(3a) $\lambda_{j+1} = \lambda_j + h_\lambda$
(3b) if $\lambda_{j+1} > b$ then STOP
(3c) $i = i + 1$

Remark 2.13 (**Parameters for algorithm 2.12**).

$\mathbf{a}$	Starting value for λ-continuation
$\mathbf{b}$	Stopping value for λ-continuation
$\mathbf{h_\lambda}$	Stepsize for λ-continuation
$\mathbf{u_L(a)}$	Solution of (7) for a (initial solution for continuation)
γ	Number of Iterations for the recursive call of $PCFASCYCLE$ (see (30))
$\nu_\mathbf{1}$	Number of presmoothing iterations, see (17)
$\nu_\mathbf{2}$	Number of postsmoothing iterations, see (20)
$\mathbf{k}$	Degree of predictor Ψ
$\hat{\mathbf{k}}$	Degree of coarse grid predictor $\hat{\Psi}$
ε	Tolerance for the corrector step

Remarks 2.14. Despite the fact that Algorithm 2.12 may readily be applied for continuation, the following might need to be considered for implementation.

Assume that we are using a predictor Ψ of order k for prediction on Ω_L. Then, depending on how "good" the initial approximation $u_L^0(a)$ is, the values in (27) change very much for $i = 0 \ldots k$. Therefore it has to be considered to start the CGP (28) later at $i = k + 1$. This modification is used in the application of algorithm 2.12 and its effect can be observed in test results in section 4. Figures 4 and 6 show that for the first k continuation steps no difference between using CGP and standard FAS is observed.

Because it may not be immediately clear how Algorithm 2.12 works using Algorithm 2.10, figure 1 is an attempt to sketch this. In Figure 1, both Ψ and $\hat{\Psi}$ (see definition 2.8) are linear predictors using the values from (26), (27) of the previous two λ-values.

Remark 2.15 (Computation of the solution for the initial parameter)**.**

As with any continuation, there is the problem of finding a solution for the initial parameter value for which no prediction is at hand. Put plainly, any method suitable for the given problem could be used to gain a solution. However, from an implementation point of view, it is reasonable to use some sort of MG method. For this we give two of many possible choices.

- A standard FAS Method (cf. [4], [10]) may be implemented for that purpose. One could also think of some kind of nested iteration of which one is described in [4, Section 9.3.4, p. 188ff].

- Because we have a solver Φ (16) on Ω_0 at hand, one may also use

$$u_0^0(a) = P_L(\mathcal{S}_{L-1}^{\nu} \circ P_{L-1}(\dots \mathcal{S}_1^{\nu} \circ P_1(\Phi(u_0^0, 0, a)) \dots)) \tag{32}$$

  for some $\nu > 0$ and some u_0^0, $a = \lambda_0$, i.e. use the inexpensive solver on the coarsest grid Ω_0 and prolong that solution onto the finest grid Ω_L, which is also not expensive. The resulting initial approximation $u_0^0(a)$ can then be used as a starting value for the standard FAS iteration (algorithm 2.5) to be improved further or can be used as $u_L^0(a)$ directly.

Instead of implementing a non-MG method or a nested MG iteration just for computation on the first parameter value λ_0, we favored (32) as an initial approximation for FAS for our implementation.

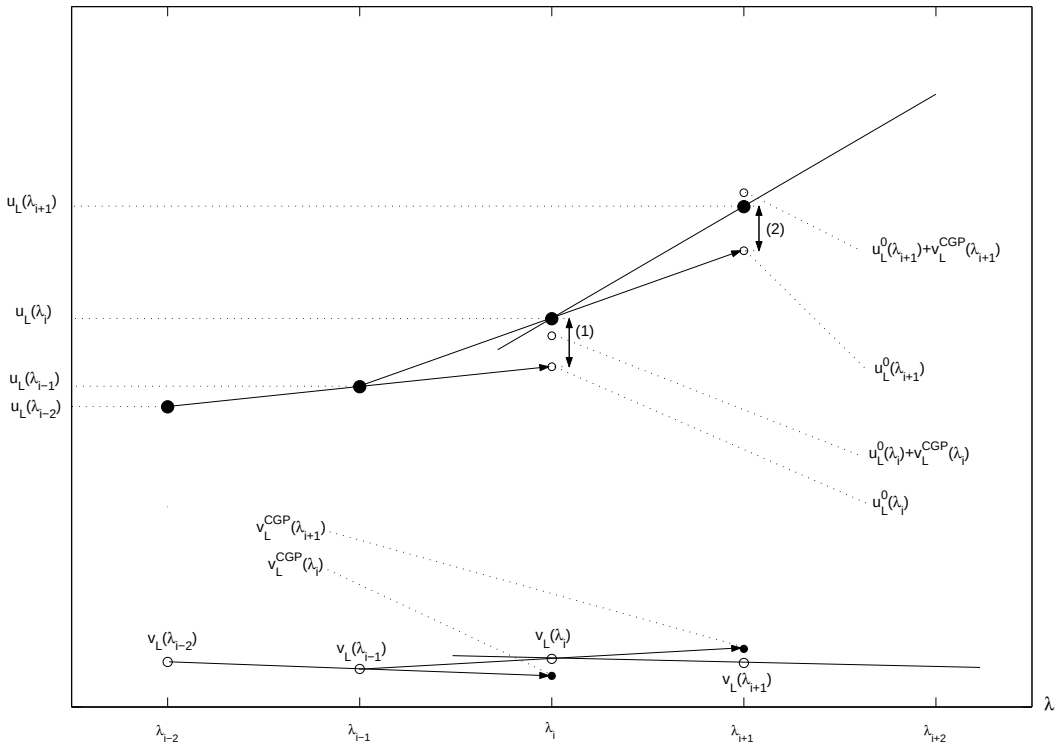

Figure 1: Simplified illustration (plot of norms and no smoothing) of Algorithm 2.12. In (1) and (2), the difference indicated by the arrows is basically the stored cumulative coarse grid correction as in step (4a) of Algorithm 2.10 (where actually the correction after the first iteration of 2.10 is stored).

3 Convergence Analysis

We prove convergence for algorithm 2.10, i.e. convergence of the iterations for one continuation step in algorithm 2.12. We do not consider topics such as existence of solutions on certain areas or areas in which the starting values u_l^0 and the coarse grid functions must lie for the algorithms to converge. Complete discussion of these topics can be found in [4, Chapter 9.5, p. 192ff]. Because Coarse Grid Prediction (CGP) only modifies the starting value w_{l-1}^0 for the correction equation, one would have to modify the results in [4] in such a way that by lowering the stepsize s in the continuation it would be assured that the coarse grid functions u_{l-1}^m and solutions u_l^m lie in the respective areas for all iterations m.

Because no emphasis is put on neighborhoods around the solution u_l^* for which the iteration converges, the convergence of the continuation in algorithm 2.12 is also not a topic in this paper. One can always reduce the stepsize s for the continuation to assure solveability.

We will give a brief overview for the ideas the upcoming estimates will be based on. Disregard the parameter λ for ease of notation where it is applicable.

Let u_l^* so that $N_l(u_l^*) = 0$. The basis of the convergence theory is the introduction of the iteration operator $M_l(\nu)$ of the nonlinear two grid iteration (either algorithm 2.10 ($PCFASCYCLE$) with two grids or the later introduced algorithm 3.1) for which, schematically speaking, we will have

$$u_l^{m+1} - u_l^* = M_l(\nu)(u_l^m - u_l^*),$$

if we disregard postsmoothing ($\nu_2 = 0$) and assume $\nu_1 = \nu$. This operator $M_l(\nu)$ is split in two parts

$$[I - P_l \mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l]\mathcal{D}\mathcal{S}_l^{(\nu)}, \tag{33}$$

using the linearizations $\mathcal{D}N_l, \mathcal{D}N_{l-1}^{-1}, \mathcal{D}\mathcal{S}_l$ of N_l, N_{l-1}^{-1} and $\mathcal{S}_l$ which will be introduced in definition 3.4. This leads to an estimate in the shape of

$$||M_l(\nu)|| \leq ||\mathcal{D}N_l^{-1} - P_l \mathcal{D}N_{l-1}^{-1} R_l|| \cdot ||\mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu)}||. \tag{34}$$

Let $\alpha > 0$. Because we are interested in boundedness of $||M_l(\nu)||$ for some large enough number of smoothing iterations ν, the estimate (34) makes the later introduced *smoothing property* (42)

$$||\mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu)}|| \leq \eta(\nu) h_l^{-\alpha} \text{ with } \eta(\nu) \to 0 \text{ as } \nu \to \infty, \tag{35}$$

and *approximation property* (43),

$$||\mathcal{D}N_l^{-1} - P_l \mathcal{D}N_{l-1}^{-1} R_l|| \leq C h_l^{\alpha}, \tag{36}$$

for some constant $C > 0$, fundamental.

The smoothing property states, in principle, that the smoothing iteration reduces the "high frequency"components of the error without amplifying the "low frequency" components. The approximation property requires the coarse grid discretization, together with prolongation and restriction, to relate to the fine grid discretization in a reasonable manner.

Hence we conclude that for estimating boundedness of the iteration operator $M_l(\nu)$ we have to suppose at least the conditions above to hold. All conditions needed to prove convergence are summarized in suppositions 3.7

On a further note, because α is the consistency order of the operator N_l it only depends on the problem (7) (for a second order discretization we have, for example, $\alpha = 2$). Hence, estimate (34) together with (35), (36) will show the h_l-independent boundedness of M_l ($||M_l(\nu)|| \leq \text{const} < 1$).

3.1 The Nonlinear Two Grid Method

The strategy in proving convergence for Algorithm 2.10 is to first assume using only two grids Ω_L and Ω_{L-1}. Furthermore, we pretend to be able to to solve the defect equation (12) on Ω_{L-1} exactly and correct the solution on the fine grid Ω_L with the prolonged exact correction. This is called a *nonlinear two grid method* (algorithm 3.1 and is the basis of the nonlinear MG methods.

Algorithm 3.1. $u_l^{m+1} = PCNTGCYCLE(u_l^m, f_l, \nu_1, \nu_2, k, \bar{\lambda}, m)$

Presmoothing and Coarse Grid Prediction (CGP)

(1) $\bar{u}_l^m = \mathcal{S}_l^{(\nu_1)}(u_l^m, f_l)$

(2) $\bar{u}_l^{CGP} = \begin{cases} \mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^m + \hat{\Psi}^k(v_l(\lambda_{j-k}), \ldots, v_l(\lambda_{j-1})), f_l\right) & , \ m = 0 \text{ and } l = L \\ \bar{u}_l^m & , \ l < L \text{ or } m > 0. \end{cases}$

Coarse grid solving

(3) $d = R_l(f_l - N_l(\bar{u}_l^{CGP}, \bar{\lambda}))$

(4) $u_{l-1} = \hat{R}_l \bar{u}_l^{CGP} = w_{l-1}^0$

(5) $f_{l-1} = d + N_{l-1}(u_{l-1}, \bar{\lambda})$

(6) $w_{l-1} = N_{l-1|_{\lambda=\bar{\lambda}}}^{-1}(f_{l-1})$

Coarse grid correction and postsmoothing

(7) $u_l^{CGC} = \bar{u}_l^{CGP} + P_l(w_{l-1} - u_{l-1})$

(8) $u_l^{m+1} = S_l^{(\nu_2)}(u_l^{CGC}, f_l)$

Storage of coarse grid correction (only applicable for $l = L$)

(9) $v_l(\lambda) = u_l^{m+1} - \bar{u}_l^m$

One is able to derive estimates sufficient for convergence on two grids. The exact solution on Ω_{l-1} is then recursively replaced by a solution obtained through a two grid method until level $l = 1$ (second to coarsest grid) is reached. Once that is the case, the solution on the coarsest grid Ω_0 is obtained through some iterative method Φ from (16). Estimating the error which is caused by this modification yields estimates for Algorithm 2.10.

Hence, the main work needs to be done in proving convergence for the nonlinear two grid method. Then, algorithm 3.1 will be modified to a MG method equivalent to algorithm 2.10.

3.2 Conditions on Operators

It is clear that the existence of at least one solution of the problem on each grid Ω_l, $0 \leq l \leq L$ is required.

Supposition 3.2. Throughout this section we assume that for $l \geq 0$ there exists a solution $u_l^* \in \mathbb{R}^{n_l}$ so that $N_l(u_l^*) = 0$.

We also need to find sets for which the computations with N_l and $\mathcal{S}_l$ are allowed. This certainly requires the u_l^* from supposition 3.2 to be *regular solutions*, for which a condition shall be given in the following remark.

Supposition 3.3. Let $l \in \mathbb{N}_{\geq 0}$ and assume that there exists a neighborhood $\mathcal{U}_l^*$ of u_l^* with $N_l \in \mathcal{C}^1(\mathcal{U}_l^*, \mathbb{R}^{n_l})$. Furthermore let

$$\det\left[\frac{\partial}{\partial u_l} N_l(u_l^*)\right] \neq 0. \tag{37}$$

Then by the implicit function theorem (see e.g. [3]) there exist neighborhoods $\mathcal{U}_l$ of u_l^* and $\mathcal{F}_l$ of 0 so that

$$N_l\Big|\mathcal{U}_l \to \mathcal{F}_l$$

is homeomorphic.

With the above argumentation it is reasonable to call the problem well posed if (37) is satisfied. We may then write $N_l^{-1}(f_l)$ for $f_l \in \mathcal{F}_l$.

To analyze algorithm 2.10 we shall use linearizations of the operators N_l and $\mathcal{S}_l$, $l \in \mathbb{N}_{\geq 0}$.

Definition 3.4. For u_l, $u_l' \in \mathcal{U}_l$ and f_l, $f_l' \in \mathcal{F}_l$ define the linearized operators $\mathcal{D}N_l$, $\mathcal{D}\mathcal{S}_l^{(\nu)}$ by

$$N_l(u_l) - N_l(u_l') = \mathcal{D}N_l(u_l, u_l')(u_l - u_l') \tag{38}$$

and

$$\mathcal{S}_l^{(\nu)}(u_l, f_l) - \mathcal{S}_l^{(\nu)}(u_l', f_l) = \mathcal{D}\mathcal{S}_l^{(\nu)}(u_l, u_l', f_l)(u_l - u_l') \tag{39}$$

as well as the inverse $(\mathcal{D}N_l)^{-1}$ by

$$N_l^{-1}(f_l) - N_l^{-1}(f_l') = (\mathcal{D}N_l)^{-1}\left(N_l^{-1}(f_l), N_l^{-1}(f_l')\right)(f_l - f_l'). \tag{40}$$

Remark 3.5. With the determining equation for $\mathcal{D}N_l^{-1}$

$$N_l^{-1}(f_l) - N_l^{-1}(f_l') = \mathcal{D}N_l^{-1}\left(f_l, f_l'\right)(f_l - f_l'),$$

it is clear that for $(\mathcal{D}N_l)^{-1}$ from (40) the relationship

$$(\mathcal{D}N_l)^{-1}\left(N_l^{-1}(f_l), N_l^{-1}(f_l')\right) = \mathcal{D}N_l^{-1}\left(f_l, f_l'\right)$$

holds. Hence it is permitted to write $\mathcal{D}N_l^{-1}$ instead of $(\mathcal{D}N_l)^{-1}$.

Remark 3.6. Even though in the introduction to this section we stated that we disregard topics such as areas of existence of the solutions and such, we briefly note that several conditions on N_l and $\mathcal{S}_l$, $l \geq 0$ would be required for the operations in algorithm 3.1 to be well defined. Put plainly, the values u_l^m and u_{l-1} need to lie in $\mathcal{U}_l$, $\mathcal{U}_{l-1}$ and f_l, f_{l-1} in $\mathcal{F}_l$, $\mathcal{F}_{l-1}$ from supposition 3.3 respectively. This, amongst other conditions, is achieved by supposing the following.

For $l > 0$ and $w_{l-1}^0 \in \mathcal{U}_{l-1}(\frac{\varepsilon_{l-1}}{2}) \subset \mathcal{U}_{l-1}$ let, for some constants C, C_{DN}

$$||\mathcal{D}N_{l-1}^{-1}(u_{l-1}, u_{l-1}')||_{\mathcal{U}\to\mathcal{F}} \leq C_{DN} \quad \text{for all } u_{l-1}, u_{l-1}' \in \mathcal{U}_{l-1}(\varepsilon_{l-1}),$$

as well as

$$||\mathcal{D}N_{l-1}^{-1}(u_{l-1}, u_{l-1}^*)[N_{l-1}(w_{l-1}^0 - R_l N_l(u_l))]||_{\mathcal{U}} \leq C h_{l-1}^{\alpha} \quad \text{for all } u_{l-1} \in \mathcal{U}_{l-1}(\varepsilon_{l-1}),$$

$$\frac{||R_l f_l||_{\mathcal{F}}}{C_{DN}} + C h_{l-1}^{\alpha} \leq \varepsilon_{l-1},$$

where α is the consistency order of N_l (i.e. $\alpha = 2$ for second order discretization) and h_l from (3). In [4, Remarks 9.5.2, 9.5.3, p. 193] similar conditions are given. However, the algorithm in [4] uses an extra parameter as stated in remark 2.14 (i), which is not necessary if the above conditions are satisfied.

Suppositions 3.7. Let the following conditions, which are essential for proofs of convergence, hold for all $\lambda \in [a, b]$, $0 \leq l \leq L$ and disregard λ for ease of notation:

1. For $u_l \in \mathcal{U}_l$ let $u_l = N_l^{-1}(f_l)$ a fixed point of $\mathcal{S}_l^{(\nu)}$:

$$u_l = \mathcal{S}_l^{(\nu)}(u_l, N_l(u_l)) \quad \text{for all } u_l \in \mathcal{U}_l, \quad \nu \geq 1. \tag{41}$$

2. The affine operator $\mathcal{D}\mathcal{S}_l$ possesses the *smoothing property on the linear operator* $\mathcal{D}N_l$, i.e. there exist functions $\eta(\nu)$ and $\bar{\nu}(h)$ and a number $\alpha \geq 0$ so that

$$\left.\begin{array}{ll} ||\mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu)}||_{\mathcal{U}\to\mathcal{F}} \leq \eta(\nu) h_l^{-\alpha} \quad \text{for all } 1 \leq \nu \leq \bar{\nu}(h_l),\ l \geq 1 & \text{with} \\ \lim_{\nu\to\infty} \eta(\nu) = 0 & \text{and} \\ \bar{\nu}(h) \equiv \infty \quad \text{or} \quad \lim_{h\to 0} \nu(\bar{h}) = 0, & \end{array}\right\} \tag{42}$$

   where α is the consistency order of N_l.

3. There exists a constant C_A so that for all $u_l, u_l' \in \mathcal{U}_l$ and $u_{l-1}, u_{l-1}' \in \mathcal{U}_{l-1}$, $l \geq 0$

$$||\mathcal{D}N_l^{-1}(u_l, u_l') - P_l \mathcal{D}N_{l-1}^{-1}(u_{l-1}, u_{l-1}')R_l||_{\mathcal{U}\to\mathcal{F}} \leq C_A h_l^{\alpha} \tag{43}$$

4. For the affine operator $\mathcal{DS}_l$ there exists a constant C_S so that

$$||\mathcal{D}\mathcal{S}_l^{(\nu)}||_{\mathcal{U}\to\mathcal{U}} \leq C_S \quad \text{for all } l > 0,\ 0 < \nu < \bar{\nu}(h_l) \tag{44}$$

5. There exist constants $C_P, C_P^{'}$ so that for all $l \geq 1$

$$C_p^{-1}||u_{l-1}||_{\mathcal{U}} \leq ||P_l u_{l-1}||_{\mathcal{U}} \leq C_p^{'}||u_{l-1}||_{\mathcal{U}} \quad \text{for all } u_{l-1} \in \mathcal{U}_{l-1}. \tag{45}$$

3.3 Convergence of the Nonlinear Multi-Grid Method

In the following theorem 3.8 we show convergence of the nonlinear two grid method modified with CGP (algorithm 3.1). This is the first step in showing convergence of the Multi-Grid (MG) method (algorithm 2.10) because the nonlinear two grid method will later be recursively modified to a MG method. We are concentrating only on the grids Ω_L and Ω_{L-1} because on all coarser grids no modifications from the standard two grid algorithm has been done by CGP as can be seen in algorithm 3.1(2) and (28).

Theorem 3.8 (Convergence of algorithm 3.1 ($PCNTGCYCLE$))**.**

Let $k > 0$, $j \geq k$ and $\lambda_{j-k}, \ldots, \lambda_{j-1}$ be given. For any stepsize $s > 0$ let $\lambda_j := \lambda_{j-1} + s$ and

$$u_L^0(\lambda_j) := \Psi^k(u_L^*(\lambda_{j-k}), \ldots, u_L^*(\lambda_{j-1})).$$

Assume that there exists a stepsize $s_{max} > 0$ so that for all $s \leq s_{max}$, when starting algorithm 3.1 with $u_L^0(\lambda_j) = u_L^0(\lambda_{j-1} + s)$, we have $f_{L-1} \in \mathcal{F}_{L-1}$ for all iterations. Then there exists $\bar{\nu}_1$ so that the iteration

$$u_L^{m+1}(\lambda_j) = PCNTGCYCLE(u_L^m(\lambda_j), 0_{f_L}, \bar{\nu}_1, 0_{\nu_2}, k, \lambda, m), \quad m \geq 0 \tag{46}$$

converges for $m \to \infty$ to the solution $u_L^*(\lambda_j)$, i.e. for every $\varepsilon > 0$ and every $s \leq s_{max}$ there exist $m(\varepsilon, s)$ so that

$$||u_L^m(\lambda_j) - u_L^*(\lambda_j)|| < \varepsilon \quad \text{for all } m > m(\varepsilon, s).$$

Proof. For ease of notation we set $l = L$.

The proof will happen in several steps. After some definitions and a consideration on the coarse grid predictor we will show a first estimate for the special case $m = 0$, in which CGP is used. Then we will treat the case $m > 0$ where we will fall back to the standard FAS Algorithm 2.5 with two grids Ω_l, Ω_{l-1} and will use results from [4]. We will disregard notation of the parameter λ where it is applicable.

Let

$$v_l^P(\lambda_j) := \hat{\Psi}^k(v_l(\lambda_{j-k}), \ldots, v_l(\lambda_{j-1})). \tag{47}$$

Furthermore, in accordance to the notation in algorithm 3.1 for $f_l = 0$ with $\bar{u}_l^{CGP} = \mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^0 + v_l^P, 0\right)$ let

$$\left.\begin{aligned} \bar{u}_l^0 &= \mathcal{S}_l^{(\nu_1)}(u_l^0, 0), \quad d = -R_l(N_l(\bar{u}_l^{CGP})) \\ u_{l-1} &= R_l \bar{u}_l^{CGP}, \quad f_{l-1} = d + N_{l-1}(u_{l-1}). \end{aligned}\right\} \tag{48}$$

(i) In this step we write L instead of l regardless of the fact that they are equal for the proof, to emphasize that CGP is only done on level L. In later steps, for ease of notation we will write l. Without loss of generality, assume we are using the simple coarse grid predictor (25), i.e.

the predicted coarse grid correction v_L^P for λ_j is the correction stored in the computation for λ_{j-1}, and by (31) given as

$$v_L^P(\lambda_j) = u_L^1(\lambda_{j-1}) - \bar{u}_L^0(\lambda_{j-1}). \tag{49}$$

Hence v_L^P is the prediction for the correction to $\bar{u}_L^0$ (as opposed to $\bar{u}_L^{CGP}$). This means we are predicting v_l from (2f) in algorithm 2.5 (used with two grids), which must not be confused with any exact correction. We are not predicting $u_L^*(\lambda_j) - \bar{u}_L^0(\lambda_j)$.

Algorithm 2.5 ($FASCYCLE$) used with the two grids Ω_L, Ω_{L-1} is a nonlinear two grid algorithm equivalent to [4, Algorithm 9.3.10, p. 187]. Also, as we mentioned before, we are predicting the correction in the first iteration that we would get if using algorithm 2.5 with two grids. This correction shall hence be called $v_L^{FAS}(\lambda_j)$ and therefore we have

$$v_L^P(\lambda_j) = v_L^{FAS}(\lambda_{j-1}). \tag{50}$$

The purpose of this is simply that at some later point in the proof we use estimates from [4] for the standard FAS.

Assuming $\lambda \mapsto v_L^{FAS}(\lambda)$ is differentiable on $[a, b]$, the mean value theorem of integral type [3, Theorem 3.10, p.174] gives

$$\begin{aligned} v_L^P(\lambda_j) - v_L^{FAS}(\lambda_j) &\overset{(50)}{=} v_L^{FAS}(\lambda_{j-1}) - v_L^{FAS}(\lambda_j) \\ &= (\lambda_{j-1} - \lambda_j) \int_0^1 \frac{\partial}{\partial \lambda} v_L^{FAS}\left(\lambda_j + t(\lambda_{j-1} - \lambda_j)\right) dt \end{aligned} \tag{51}$$

Define

$$\mathcal{L}(\lambda_{j-1}, \lambda_j) := \int_0^1 \frac{\partial}{\partial \lambda} v_L^{FAS}\left(\lambda_j + t(\lambda_{j-1} - \lambda_j)\right) dt.$$

With $s = \lambda_j - \lambda_{j-1}$ we rewrite (51) as

$$v_L^P(\lambda_j) - v_L^{FAS}(\lambda_j) = -s\mathcal{L}(\lambda_{j-1}, \lambda_j).$$

For ease of notation we disregard the parameter λ_j in v_L and conclude

$$v_L^P = v_L^{FAS} - s\mathcal{L}(\lambda_{j-1}, \lambda_j). \tag{52}$$

(ii) Case $m = 0$:
Since we disregard postsmoothing ($\nu_2 = 0$ in (46)), the error after the first iteration is given by

$$\begin{aligned} \delta u_l^1 &:= u_l^1 - u_l^* \\ &= u_l^{CGC} - u_l^* \\ &= \bar{u}_l^{CGP} + P_l(w_{l-1} - u_{l-1}) - u_l^* \\ &= \mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^0 + v_l^P, 0\right) + P_l(w_{l-1} - u_{l-1}) - u_l^*. \end{aligned} \tag{53}$$

We need to investigate the value $w_{l-1} - u_{l-1}$ further. The definition of the linearized operators (see definition 3.4) gives us

$$\begin{aligned} w_{l-1} - u_{l-1} &= N_{l-1}^{-1}(f_{l-1}) - N_{l-1}^{-1}(N_{l-1}(u_{l-1})) \\ &\overset{(48)}{=} N_{l-1}^{-1}\left(N_{l-1}(u_{l-1}) - R_l(N_l(\bar{u}_l^{CGP}))\right) - N_{l-1}^{-1}(N_{l-1}(u_{l-1})) \\ &\overset{N_l(u_l^*)=0}{=} N_{l-1}^{-1}\left(N_{l-1}(u_{l-1}) - R_l(N_l(\bar{u}_l^{CGP}) - N_l(u_l^*))\right) \\ &\qquad -N_{l-1}^{-1}(N_{l-1}(u_{l-1})) \\ &\overset{(38),(40)}{=} -\left[\mathcal{D}N_{l-1}^{-1}(w_{l-1}, u_{l-1}) R_l \mathcal{D}N_l(\bar{u}_l^{CGP}, u_l^*)\right](\bar{u}_l^{CGP} - u_l^*). \end{aligned}$$

With $\bar{u}_l^{CGP} = \mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^0 + v_l^P, 0\right)$ and (41) we get

$$\begin{aligned} w_{l-1} - u_{l-1} &= -\left[\mathcal{D}N_{l-1}^{-1}(w_{l-1}, u_{l-1}) R_l \mathcal{D}N_l(\bar{u}_l^{CGP}, u_l^*)\right] \\ &\qquad \cdot \left(\mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^0 + v_l^P, 0\right) - \mathcal{S}_l^{(\nu_1)}(u_l^*, 0)\right) \\ &\overset{(39)}{=} -\left[\mathcal{D}N_{l-1}^{-1}(w_{l-1}, u_{l-1}) R_l \mathcal{D}N_l(\bar{u}_l^{CGP}, u_l^*) \mathcal{D}\mathcal{S}_l^{(\nu_1)}(\bar{u}_l^{CGP}, u_l^*, 0)\right] \\ &\qquad \cdot(\bar{u}_l^0 + v_l^P - u_l^*). \end{aligned}$$

Setting

$$\begin{aligned} \mathcal{D}N_l &:= \mathcal{D}N_l(\bar{u}_l^{CGP}, u_l^*), \\ \mathcal{D}\mathcal{S}_l^{(\nu_1)} &:= \mathcal{D}\mathcal{S}_l^{(\nu_1)}(\bar{u}_l^{CGP}, u_l^*, 0), \\ \mathcal{D}N_{l-1}^{-1} &:= \mathcal{D}N_{l-1}^{-1}(w_{l-1}, u_{l-1}) \end{aligned}$$

leads to

$$w_{l-1} - u_{l-1} = -\left[\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu_1)}\right](\bar{u}_l^0 + v_l^P - u_l^*). \tag{54}$$

Inserting (54) into (53) yields

$$\begin{aligned} \delta u_l^1 &= \mathcal{S}_l^{(\nu_1)}\left(\bar{u}_l^0 + v_l^P, 0\right) - u_l^* - \left[P_l \mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu_1)}\right](\bar{u}_l^0 + v_l^P - u_l^*) \\ &\overset{(39),(41)}{=} \mathcal{D}\mathcal{S}_l^{(\nu_1)}(\bar{u}_l^0 + v_l^P - u_l^*) - \left[P_l \mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu_1)}\right](\bar{u}_l^0 + v_l^P - u_l^*) \\ &= \left[I - P_l \mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l\right] \mathcal{D}\mathcal{S}_l^{(\nu_1)}(\bar{u}_l^0 + v_l^P - u_l^*) \\ &= \left[\mathcal{D}N_l^{-1} - P_l \mathcal{D}N_{l-1}^{-1} R_l\right]\left[\mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu_1)}\right](\bar{u}_l^0 + v_l^P - u_l^*) \end{aligned} \tag{55}$$

Furthermore, by (52) and $\mathcal{L} := \mathcal{L}(\lambda_{j-1}, \lambda_j)$ we have

$$\bar{u}_l^0 + v_l^P - u_l^* = \bar{u}_l^0 + v_L^{FAS} - s\mathcal{L} - u_l^*. \tag{56}$$

As we mentioned before, algorithm 2.5 used with two grids is a nonlinear two grid algorithm equivalent to [4, Algorithm 9.3.10, p. 187]. Because $\bar{u}_l^0$ plus the correction that would be computed by the two grid version of algorithm 2.5 is exactly its first iterate, namely $u_l^{1,FAS}$, we can rewrite (56) as

$$\bar{u}_l^0 + v_l^P - u_l^* = u_l^{1,FAS} - u_l^* - s\mathcal{L}. \tag{57}$$

For a normwise estimate, we know from [4, Proposition 9.5.6, p. 194f] that there exists ν_1^0 so that (57) together with $C_P := |\mathcal{L}|$ gives

$$\begin{aligned} ||\bar{u}_l^0 + v_l^P - u_l^*|| &\leq ||u_l^{0,FAS} - u_l^*|| + s\, C_P \\ &= \xi' ||u_l^0 - u_l^*|| + s\, C_P, \end{aligned} \tag{58}$$

with $\xi' < 1$. The latter is achieved in [4] by a sufficient number of smoothing iterations.

Note that in (i) we have assumed using the simple coarse grid predictor (25). In case we would use a predictor of order $p > 1$ in (47),(49), the last term in (58) would change to $s^p\, C_P^{'}$ for some constant $C_P^{'}$.

Let

$$M_l(\nu) := \left[\mathcal{D}N_l^{-1} - P_l \mathcal{D}N_{l-1}^{-1} R_l\right]\left[\mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu)}\right]. \tag{59}$$

Then by (43) and (42) there exists ν_1^1 so that

$$C_0 := ||M_l(\nu_1^1)|| \leq \underbrace{||\mathcal{D}N_l^{-1} - P_l \mathcal{D}N_{l-1}^{-1} R_l||}_{\text{constant by (43)}} \cdot \underbrace{||\mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\nu_1^1)}||}_{\text{small by (42)}} < 1; \tag{60}$$

With ν_1^0, ν_1^1 from (58),(60) respectively, define $\nu_1^2 := \max\{\nu_1^0, \nu_1^1\}$. Going back to (55), we use (58), (59) and (60) to rewrite (55) as

$$\begin{aligned} ||\delta u_l^1|| &\leq ||M_l(\nu_1^2)||_{\mathcal{U}\to\mathcal{U}} \cdot \left(\xi' ||u_l^0 - u_l^*|| + s\, C_P\right) && (61)\\ &\leq C_0\, \xi' ||u_l^0 - u_l^*|| + s\, C_0\, C_P, && (62) \end{aligned}$$

with $C_0,\ \xi' < 1$.

(iii) Case $m > 0$:
This is to be discussed rather quickly, since we fall back to the standard FAS algorithm 2.5 with 2 grids equivalent to [4, Algorithm 9.3.10, p. 187] and use, for the first iteration, the case (ii). By [4, Proposition 9.5.6, p. 194f], there exists ν_1^3 so that for some $\xi < 1$ we have

$$||\delta u_l^m|| \leq \xi ||\delta u_l^{m-1}|| \leq \cdots \leq \xi^{m-1} ||\delta u_l^1|| \overset{(62)}{\leq} \xi^{m-1} \cdot \left(C_0\, \xi' ||u_l^0 - u_l^*|| + s\, C_0\, C_P\right).$$

We seek

$$\xi^{m-1}\, C_0\, \xi' ||u_l^0 - u_l^*|| + s\, C_0\, C_P\, \xi^{m-1} < \varepsilon, \tag{63}$$

which, by setting $\bar{\nu}_1 := \max\{\nu_1^3, \nu_1^2\}$ and using, for given ε and s the relationship

$$s < \frac{\varepsilon - \xi^{m-1}\, C_0\, \xi' ||u_l^0 - u_l^*||}{2\, C_0\, C_P\, \xi^{m-1}} \tag{64}$$

given by (63) yields $m(\varepsilon, s)$ and therefore proves the assertion.

□

Remark 3.9 (Interpretation of (64))**.** Based on the relation (64) we take another look at how the stepsize s may be chosen depending on the parameters ξ, ξ', the constants C_0, C_P and the distance of the initial approximation u_L^0 to the solution u_L. We rewrite (64) as

$$s = \frac{\varepsilon}{2C_0\, C_P\, \xi^{m-1}} - \frac{\xi' ||u_L^0 - u_L^*||}{2C_P}. \tag{65}$$

A few points may be concluded from (65).

- The smaller ξ and ξ' are, the larger s may be chosen. The values of ξ and ξ' are contraction numbers of the nonlinear two grid algorithm [4, Algorithm 9.3.10, p. 187] and are influenced by the number of smoothing iterations.
- The constant C_0, defined in (60) behaves like ξ, ξ'. The smaller it is, the larger s can be chosen. It is specific to the CGP, and provides a sort of "weak" contraction number in (62) for the first iteration $m = 0$ on the finest grid $l = L$.
- The effect of the constant C_P, defined near (58) can not be clearly interpreted. This is expected, because it is a constant coming from the mean value theorem and hence has no direct correlation to the algorithm.
- Interpretation of $||u_l^0 - u_l^*||$ is a little tricky. At first sight, one might conclude that the closer u_L^0 is to u_L^*, the larger s can be chosen. However, s is the stepsize for both predictors Ψ and $\hat{\Psi}$. The s in (64) is introduced in (52), from the predictor $\hat{\Psi}$. Therefore, the closer u_L^0 is to u_L^*, the larger s can be chosen. A larger s however, means a worse prediction u_L^0 by Ψ and possibly an u_L^0 which in turn is further from u_L^*.

Theorem 3.8 verifies convergence for two grids. However, the subject of this paper is MG algorithms. Hence we are recursively extending the two grid iteration to a MG iteration. Again we will only derive estimates for the level L, because only there we have modified the standard FAS algorithm with CGP. Estimates for all coarser grids , in which we conform to the standard FAS, exist in [4] and will be used in the convergence theorem 3.11 for algorithm 2.10.

Multi-Grid (MG) algorithm as a modified two grid algorithm

The transition from the two grid method to a MG method is achieved by recursively replacing step (6) in algorithm 3.1 by the γ-fold application of a converging iteration Φ_{l-1}:

$$w_{l-1} = \Phi_{l-1}^{\gamma}(w_{l-1}^0, f_{l-1}) \quad \text{for } l > 1 \tag{66}$$

and

$$w_0 = \Phi_0(w_0^0, f_0), \tag{67}$$

where we disregard λ for ease of notation. The coarse grid solver Φ_0 is assumed to solve $N_0(w_0) = f_0$ exactly, hence we do not write the γ-fold application but denote by Φ_0 a complete solver on Ω_0. That is contrary to Φ_l for $l > 0$, with which only one iteration of the modified two grid algorithm is denoted. This modified iteration shall be called algorithm 3.1_{MGM}. By setting

$$\Phi_{l-1}(w_{l-1}^0, f_{l-1}) = \text{output of algorithm } 3.1_{MGM} \text{ with parameters } w_{l-1}^0,\ f_{l-1}, l-1 \tag{68}$$

for $l > 0$ and for $l = 0$ using the coarse grid solver defined in (16)

$$\Phi_0(w_0^0, f_0) = \text{output of coarse grid solver } \Phi \text{ with parameters } w_0^0,\ f_0, \tag{69}$$

the new algorithm 3.1_{MGM} is equivalent to algorithm 2.10.

Lemma 3.10. Let $u_{L-1} \mapsto \Phi_{L-1}(u_{L-1}, f_{L-1})$ be an iteration converging to $N_{L-1}^{-1}(f_{L-1})$ with contraction number $\phi_{L-1} < 1$:

$$||\Phi_{L-1}(u_{L-1}, f_{L-1}) - N_{L-1}^{-1}(f_{L-1})||_{\mathcal{U}} \le \phi_{L-1}||u_{L-1} - N_{L-1}^{-1}(f_{L-1})||_{\mathcal{U}} \tag{70}$$

for all $u_{L-1} \in \mathcal{U}_{L-1}$, $f_{L-1} \in \mathcal{F}_{L-1}$. If step (6) in algorithm 3.1 is replaced by the γ-fold application of Φ_{L-1}:

$$w_{L-1} = \Phi_{L-1}^{\gamma}(w_{L-1}^0, f_{L-1}), \tag{71}$$

then there exists a constant C_L^* so that for $l = L$ the relation (62) from the proof of theorem 3.8 changes to

$$||\delta u_L^1|| \le \left(C_0 + \phi_{L-1}^{\gamma} C_L^*\right) \cdot \left(\xi' ||u_L^0 - u_L^*|| + s\, C_P\right). \tag{72}$$

Proof. Let $l = L$.

With introducing (71) to algorithm 3.1 we get an error δw_{l-1} which is given by

$$\delta w_{l-1} := N_{l-1}^{-1}(f_{l-1}) - w_{l-1}. \tag{73}$$

Note that $N_{l-1}^{-1}(f_{l-1})$ is the former w_{l-1} from algorithm 3.1 which has been changed by (71).

Call $\tilde{u}_l^1$ the output of the first iteration of algorithm 3.1 with modification (71). Then as in (53)

$$\begin{aligned} \delta\tilde{u}_l^1 &= \delta\bar{u}_l^{CGP} + P_l(w_{l-1} - u_{l-1}) \\ &\overset{(73)}{=} \delta\bar{u}_l^{CGP} + P_l(N_{l-1}^{-1}(f_{l-1}) - \delta w_{l-1} - u_{l-1}) \\ &= \delta\bar{u}_l^{CGP} + P_l(N_{l-1}^{-1}(f_{l-1}) - u_{l-1}) - P_l\delta w_{l-1}. \end{aligned} \tag{74}$$

By Algorithm 3.1 (7), (74) can be written as

$$\delta\tilde{u}_l^1 = \delta u_l^1 - P_l\delta w_{l-1}. \tag{75}$$

Furthermore we can use (73) to obtain

$$\begin{aligned} ||\delta w_{l-1}|| &= ||N_{l-1}^{-1}(f_{l-1}) - w_{l-1}|| \\ &\overset{(71)}{=} ||N_{l-1}^{-1}(f_{l-1}) - \Phi_{l-1}^{\gamma}(w_{l-1}^0, f_{l-1})|| \\ &\overset{(70)}{\le} \phi_{l-1}^{\gamma}||N_{l-1}^{-1}(f_{l-1}) - w_{l-1}^0||. \end{aligned}$$

Because in algorithm 3.1 we set $w_{l-1}^0 = u_{l-1}$, we can write

$$||\delta w_{l-1}|| \leq \phi_{l-1}^{\gamma} ||N_{l-1}^{-1}(f_{l-1}) - u_{l-1}|| \ ,$$

and therefore, (75) together with (62) gives

$$||\delta \tilde{u}_l^1|| \leq C_0 \cdot \left(\xi' ||u_l^0 - u_l^*|| + s\, C_P\right) + \phi_{l-1}^{\gamma} ||P_l||_{\mathcal{U}\to\mathcal{U}} ||N_{l-1}^{-1}(f_{l-1}) - u_{l-1}|| \ , \tag{76}$$

with C_0 defined in (60).

The value $N_{l-1}^{-1}(f_{l-1}) - u_{l-1}$ has already been discussed in the proof of theorem 3.8 where in (54) together with (58) we showed that

$$||N_{l-1}^{-1}(f_{l-1}) - u_{l-1}|| \leq ||\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}} \cdot \left(\xi' ||u_l^0 - u_l^*|| + s\, C_P\right), \tag{77}$$

with the number of smoothing iterations $\bar{\nu}_1$ from theorem 3.8.

Applying (77) to (76) yields

$$||\delta \tilde{u}_l^1|| \leq \left(C_0 + \phi_{l-1}^{\gamma} ||P_l||_{\mathcal{U}\to\mathcal{U}} ||\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}}\right) \cdot \left(\xi' ||u_l^0 - u_l^*|| + s\, C_P\right). \tag{78}$$

We have yet to estimate $||P_l||_{\mathcal{U}\to\mathcal{U}} ||\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}}$. Supposition (45) gives

$$\begin{aligned} ||P_l||_{\mathcal{U}\to\mathcal{U}} ||\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}} &\leq C_p' ||\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}} \\ &\overset{(45)}{\leq} C_p' C_p ||P_l \mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}}. \end{aligned} \tag{79}$$

Furthermore, by cheating a bit, we obtain

$$\begin{aligned} P_l \mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)} &= \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)} - \left[\mathcal{D}N_l^{-1} - P_l \mathcal{D}N_{l-1}^{-1} R_l\right] \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)} \\ &\overset{(59)}{=} \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)} - M_l(\bar{\nu}_1). \end{aligned} \tag{80}$$

Inserting (80) into (79) together with supposition (44) and the definition of C_0 in (60) we conclude

$$||P_l||_{\mathcal{U}\to\mathcal{U}} ||\mathcal{D}N_{l-1}^{-1} R_l \mathcal{D}N_l \mathcal{D}\mathcal{S}_l^{(\bar{\nu}_1)}||_{\mathcal{U}\to\mathcal{U}} \leq C_p C_p' (C_S + C_0) =: C_L^*. \tag{81}$$

The latter equation together with (78) completes the proof. □

Lemma 3.10 shows that the transition from algorithm 3.1 to the recursively modified two grid algorithm 3.1_{MGM} as in (68),(69) causes the extra term $\phi_{L-1}^{\gamma} C_L^*$ to appear in (72). That means we have to assure

$$C_0 + \phi_{L-1}^{\gamma} C_L^* < 1 \tag{82}$$

on level L with C_L^* from lemma 3.10. On all lower levels algorithm 2.10 behaves like the standard FAS which is treated in [4, Algorithm NMGM, p. 188] and hence we can use the estimates and C^* from [4, Section 9.5.2, p. 196ff] on these levels.

We shall conclude these statements by adapting [4, Remark 9.5.11, p. 197] to our modification for CGP on level L. The remark in [4] uses chapters 6 and 7 from [4], therefore we are using citations rather than including those chapters into this paper.

Let the contraction number ϕ_0 of the coarse grid solver, given by

$$||\Phi_0(u_0, f_0) - N_0^{-1}(f_0)||_{\mathcal{U}} \leq \phi_0 ||u_0 - N_0^{-1}(f_0)||_{\mathcal{U}}$$

be sufficiently small. Furthermore, let $C := \max\{C^*, C_L^*\}$ with C^* from [4, (9.5.8c), p. 197] (if no CGP is used, C^* and C_L^* are identical). With (66), (67) and (82) modify [4, (9.5.10), p. 197] to the system of inequalities

$$\left.\begin{array}{lcll} \phi_1 &=& \xi_1 + C \cdot \phi_0 < 1, & \\ \phi_l &=& \xi_l + C \cdot \phi_{l-1}^{\gamma} < 1 & \text{for } 1 < l < L, \\ \phi_L &=& C_0 + C \cdot \phi_{L-1}^{\gamma} < 1, & \end{array}\right\} \tag{83}$$

with ξ_l, $0 < l < L$ being the given contraction numbers from [4, Proposition 9.5.6, p. 194] for the standard two grid iterations on levels l already mentioned in (58) and C_0 from (72).

Then, if $\gamma \geq 2$, (83) has solutions

$$\phi_0, \dots \phi_L < 1 \tag{84}$$

because by assumption ϕ_0 is sufficiently small and by [4, Proposition 9.5.6, p. 194] and (60), for a large enough number of smoothing iterations ν, the quantities $\xi_1, \dots \xi_{L-1}$ and $\xi_L = C_0$ can be made sufficiently small as well.

With (84) we can use [4, Theorem 9.5.12, p. 197] and lemma 3.10 for proof of convergence of algorithm 2.10. Because we are only interested in convergence, no attention is given to the parts of [4, Theorem 9.5.12] in which areas of solveability and such are treated.

Theorem 3.11 (Convergence of algorithm 2.10 ($PCFASCYCLE$))**.**

Let $k > 0$, $j \geq k$ and $\lambda_{j-k}, \dots, \lambda_{j-1}$ be given and $\gamma \geq 2$. For any stepsize $s > 0$ let $\lambda_j := \lambda_{j-1} + s$ and

$$u_L^0(\lambda_j) := \Psi^k(u_L^*(\lambda_{j-k}), \dots, u_L^*(\lambda_{j-1})).$$

Assume that there exists a stepsize $s_{max} > 0$ so that for all $s \leq s_{max}$, when starting algorithm 2.10 with $u_L^0(\lambda_j) = u_L^0(\lambda_{j-1} + s)$, we have $f_0 \in \mathcal{F}_0$ for all iterations (i.e. the coarse grid solver Φ (16) on Ω_0 is well defined for the solution of $N_0(w_0) = f_0$). Furthermore assume that Φ satisfies

$$||\Phi_0(u_0, f_0) - N_0^{-1}(f_0)||_{\mathcal{U}} \leq \phi_0 ||u_0 - N_0^{-1}(f_0)||_{\mathcal{U}} \quad \text{for all } f_0 \in \mathcal{F}_0 \tag{85}$$

with ϕ_0 sufficiently small. Then there exists $\bar{\nu}_1$ so that the iteration

$$u_L^{m+1}(\lambda_j) = PCFASCYCLE(L, \gamma, u_L^m(\lambda_j), 0_{f_L}, \bar{\nu}_1, 0_{\nu_2}, k, \lambda, m), \quad m \geq 0 \tag{86}$$

converges for $m \to \infty$ to the solution $u_L^*(\lambda_j)$, i.e. for every $\varepsilon > 0$ and every $s \leq s_{max}$ there exist $m(\varepsilon, s)$ so that

$$||u_L^m(\lambda_j) - u_L^*(\lambda_j)|| < \varepsilon \quad \text{for all } m > m(\varepsilon, s).$$

Proof. For ease of notation disregard the parameter λ. Because the coarse grid solver Φ is assumed to solve $N_0(w_0) = f_0$ exactly, we can assume ϕ_0 from (85) to be sufficiently small. Then with analogue argumentation as in (58) and the same notation, (84), lemma 3.10 and [4, Theorem 9.5.12] yield that there exists a number of smoothing iterations ν_1^1 so that

$$||\delta u_L^1|| \leq C_1 \cdot \left(\xi' ||u_L^0 - u_L^*|| + s\, C_P\right), \tag{87}$$

with $\delta u_L^1 := u_L^1 - u_L^*$, $C_1 < 1$ by (84) and $\xi' < 1$ by [4, Theorem 9.5.12]. For iterations $m < 0$, argumentation is analogue to the proof of theorem 3.8(iii). By [4, Theorem 9.5.12], there exists ν_1^2 so that for some $\xi < 1$ we have

$$||\delta u_l^m|| \quad \leq \quad \xi ||\delta u_l^{m-1}|| \leq \dots \leq \xi^{m-1} ||\delta u_l^1|| \overset{(87)}{\leq} \xi^{m-1} \cdot \left(C_1\, \xi' ||u_l^0 - u_l^*|| + s\, C_1\, C_P\right).$$

Computations similar to (63),(64) complete the proof. □

The relationship of the constants $\xi, \xi', C_1, C_P, ||u_l^0 - u_l^*||$ to the stepsize s can be interpreted in analog manner to remark 3.9.

4 Test Problems

For testing algorithm 2.12, we will apply it to three different test problems of different character each.

The *Chandrasekhar-H equation* is a nonlinear integral equation which gives a full (as opposed to sparse) nonlinear system of equations if discretized. The *modified Bratu problem* on the other hand

is an elliptic PDE with boundary values, which, if discretized, gives a band-shaped nonlinear system of equations. The problem of *continuation of invariant tori* is different from both aforementioned types. Because of the periodic boundary values, discretization yields a nonlinear system of equations with band-shape and additional entries for the periodic boundary values.

We show results of applying algorithm 2.12 to these three test problems in the following sections. We used constant stepsizes as well as a simple heuristic stepsize control. Throughout all computations we use a set of options for algorithm 2.12. Some have common values for all test problems. These are γ from (30) and ν_1, ν_2 for smoothing from algorithm 2.10 with

$$\gamma = 2 \quad \text{and} \quad \nu_1 = \nu_2 = 2.$$

The value $\gamma = 2$ results in a W-Cycle (see for example [4, Section 2.5]) and ν_1, ν_2 determine the amount of smoothing iterations. For the coarse grid solver we use the Newton algorithm with approximated Jacobian in all test problems. For those options that do not have common values, we will give a description once in table 1, and announce their specific values in each section for the three problems algorithm 2.12 has been applied to.

Option	**Description**
Λ	Parameter interval, see (7)
$u_L^0(\lambda_0 = a)$	Starting solution vector for the initial parameter value in $\mathbb{R}^{n_L}$
s	Difference between two parameter values $\lambda_j - \lambda_{j-1}$ if no stepsize control is used
$k, \hat{k}$	Orders of predictors Ψ, $\hat{\Psi}$ from definition 2.8 respectively
ε	Relative tolerance for $\|\|u^m - u^{m-1}\|\|$

Table 1: Options for computations with algorithm 2.12

The computations that are done in this work are compared with results from [9] and [8]. We have tried to match the conditions those results were obtained in as closely as possible. Because we used more modern computational equipment for our computations than possibly has been used in [9],[8], factors may be applied to the original computation times. Those will be mentioned where necessary.

Furthermore, the computation times of algorithm 2.12 are compared with the computation times needed when algorithm 2.5 instead of 2.10 is used in step (2b) of algorithm 2.12. The latter scenario shall be called algorithm 2.12_{FAS}.

The computer used was equipped with an AMD Athlon64 3000+ CPU, running at 1.8 GHz with 512 MB RAM.

4.1 Matlab Implementation

All computations are done in MATLAB. The source code package as a collection of so called `M-Files` is freely available from the author [1].

The central part is the implementation of the $PCFASCYCLE$ algorithm 2.10 which itself is a recursive algorithm. However, for the implementation we transformed it into an equivalent non recursive algorithm and consequently have to save the current level in the variable `k` and the iterations done on each level in the vector `it` with L elements. Figure 2 displays a flow chart for the non recursive version of algorithm 2.10.

[1] eMail `schmidt@itwm.fhg.de`

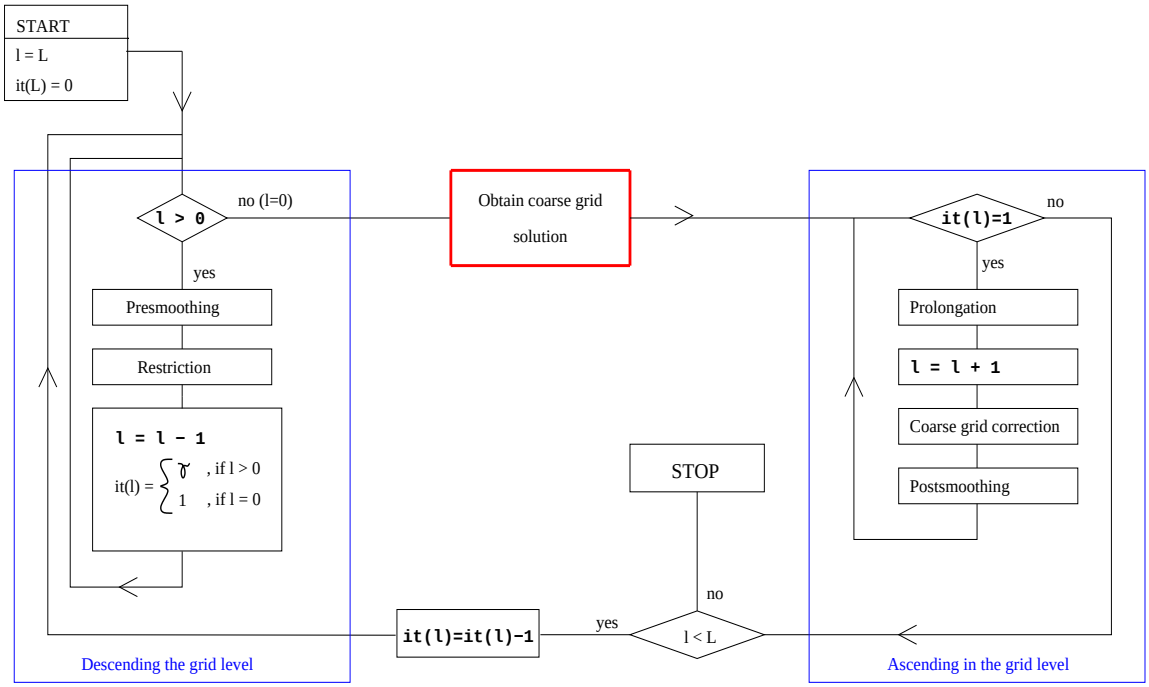

Figure 2: Flow chart for the non recursive version of algorithm 2.10. The current grid level is denoted by l and the amount of iterations on each level is stored in the vector it.

4.2 Chandrasekhar-H Equation

The Chandrasekhar-H equation

$$F(H,\lambda) = 0, \quad H : [0,1] \to \mathbb{R},$$

with parameter λ and the operator F as

$$F(H,\lambda)(\mu) = H(\mu) - \left(1 - \frac{\lambda}{2}\int_0^1 \frac{\mu H(\nu)d\nu}{\mu+\nu}\right)^{-1} \tag{88}$$

is discussed in [6, p. 87ff]. To gain a system of nonlinear equations on levels $l \in \mathbb{N}_{\geq 0}$, (88) is discretized by using an n_l point grid and

$$\int_0^1 f(\mu)d\mu \cong \frac{1}{n_l}\sum_{j=1}^{n_l} f(\mu_j), \quad \mu_i = \frac{i-0.5}{n_l} \quad \text{for} \quad 1 \leq i \leq n_l.$$

The resulting system of nonlinear equations (for levels $l \in \mathbb{N}_{\geq 0}$) is

$$N_{l,i}(u_l,\lambda) = u_{l,i} - \left(1 - \frac{\lambda}{2n_l}\sum_{j=1}^{n_l}\frac{\mu_i u_{l,j}}{\mu_i+\mu_j}\right)^{-1} = 0, \quad 1 \leq i \leq n_l. \tag{89}$$

When starting with $\mathbf{1}_{\mathbb{R}^{n_l}}$, system (89) has a solution for all $\lambda \in (0,1)$. Using Algorithm 2.2 for smoothing, we need

$$\frac{\partial N_{l,i}(u_l,\lambda)}{\partial u_{l,i}} = 1 + \frac{\lambda}{4n_l}\left(1 - \frac{\lambda}{2n_l}\sum_{j=1}^{n_l}\frac{\mu_i u_{l,j}}{\mu_j+\mu_i}\right)^{-2}. \tag{90}$$

Hence, line 4 from the Gauss-Seidel smoothing algorithm 2.2 can, using (89),(90), be written as

$$u_{l,i} = \left(\lambda \cdot u_{l,i} + 4n_l - 2\lambda \sum_{j=1}^{n_l} \frac{\mu_i u_{l,j}}{\mu_j + \mu_i} \right) \cdot \left[\lambda + 4n_l \left(1 - \frac{\lambda}{2n_l} \sum_{j=1}^{n_l} \frac{\mu_i u_{l,j}}{\mu_j + \mu_i} \right)^2 \right]^{-1}.$$

Matching the conditions from [8, Section 6.2.4, p. 105], the options from table 1 are set as

$$\begin{gathered} \Lambda = [0.001, 0.999], \quad u_L^0(a) = \mathbf{1} = (1, 1, \ldots, 1)^T \in \mathbb{R}^{n_L}, \quad s = 0.1, \\ k = 4, \quad \hat{k} = 2 \quad \varepsilon = 10e - 5. \end{gathered} \tag{91}$$

The coarsest grid Ω_0 is a 32 point grid. For smoothing we use algorithm 2.2. Computations are done on the grids $\Omega_1, \ldots, \Omega_L$ with the largest L being 6. The corresponding grid sizes are

$$n_1 = 64,\ n_2 = 128,\ n_3 = 256,\ n_4 = 512,\ n_5 = 1024,\ n_6 = 2048.$$

Applying algorithm 2.12 under the stated conditions with constant stepsize yields the results in table 2, where the times are measured for the whole continuation (11 continuation steps).

Grid points (L)	Comp. time in seconds	Comp. time $\mathcal{S}^{(\nu_i)}$ in %	Comp. time P_l, R_l in %	Comp. time CG It. in %	CG It. of Φ	MG iterations
64 (1)	**1.2030**	16.52	3.23	56.26	**53**	**46**
2.12_{FAS}	1.3120	17.80	2.92	58.37	59	51
128(2)	**2.2350**	36.84	1.75	46.63	**82**	**45**
2.12_{FAS}	2.4380	30.61	1.61	46.76	91	50
256 (3)	**4.1090**	51.64	0	26.95	**99**	**44**
2.12_{FAS}	4.9540	48.70	0	33.03	142	49
512 (4)	**9.6090**	64.02	0	12.02	**106**	**45**
2.12_{FAS}	10.9840	63.26	0	15.58	158	50
1024 (5)	**25.7310**	76.06	0	4.99	**115**	**44**
2.12_{FAS}	30.4530	73.53	0	6.36	168	50
2048 (6)	**86.5940**	79.71	0	1.90	**124**	**45**
2.12_{FAS}	95.3910	79.37	0	2.15	169	50

Table 2: Chandrasekhar-H Equation: Computational time of algorithm 2.12 (and algorithm 2.12_{FAS} in the respective rows). Measurements were taken as an average over 5 runs. The percentwise computation times in columns 3-5 were given through the `Matlab Profile Viewer` tool. Percentages less than 1% are noted as 0.

From table 2 it can be concluded that Coarse Grid Prediction (CGP) in (28) has an advantage over using the standard FAS algorithm 2.5.

The results in table 2 can be directly compared to the results from [8] because similar computational equipment was used in [8] (Pentium IV, 2.4GHZ, 512MB RAM) on a grid of 1000 points. The fastest of many methods tested in [8] yielded a computation time of 931.2030 seconds, meaning 2.12 is more than 30 times faster in that application.

Aside from the computation times it might be interesting to see how the number of MG iterations (iterations of algorithm 2.10) changes with the parameter value. Figure 3 compares algorithms 2.12 and 2.12_{FAS} on the grid Ω_5 (1024 points) in this respect. All further plots and comparisons in this section are also done on this grid.

From figure 3, one concludes that the CGP results in some cases in fewer MG iterations. This gives rise to the idea of applying a simple heuristic stepsize adaption to the continuation, where we define an optimal number of iterations m_{opt} and adapt the stepsize s_{j+1} according to the amount of iterations needed for the current parameter value λ_j as

$$s_{j+1} = s_j \cdot \max \left\{ \min \{ \frac{m_{opt}}{m_\lambda}, 2 \}, 0.5 \right\}. \tag{92}$$

The results are quite impressing, as they show a clear advantage if CGP is used. Figure 4 starts at the parameter value 0.4 because no stepsize adaption is done before the predictor has reached order 4. After that, a square/circle is plotted for each continuation step. The figure uses $m_{opt} = 3$. For values $m_{opt} = 2, 3, 4$, table 3 compares certain measures of performance.

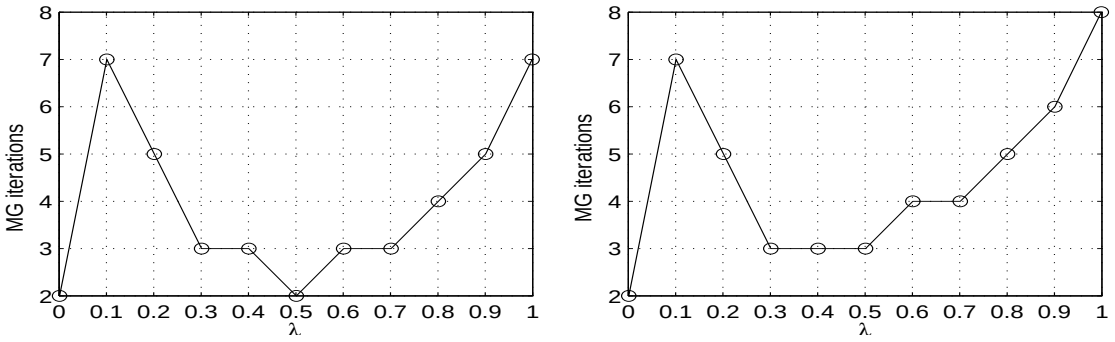

Figure 3: Chandrasekhar-H Equation: MG iterations plotted against the parameter value $\lambda \in [0.001, 0.999]$ with constant stepsize $s = 0.1$. Left: Algorithm 2.12 (i.e. use of CGP). Right: Algorithm 2.12_{FAS}

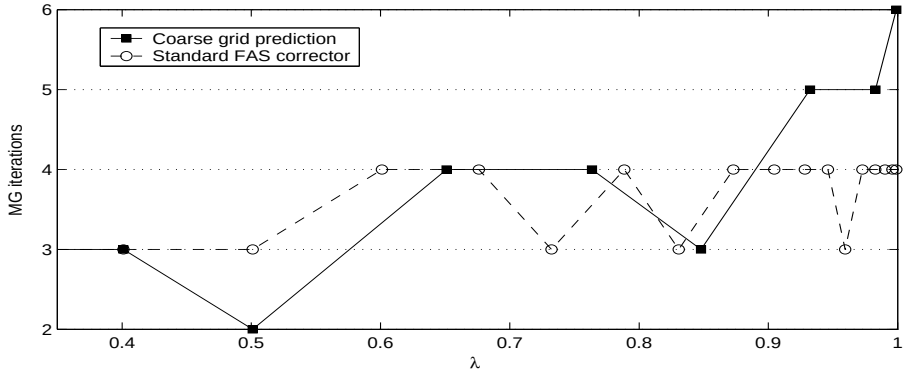

Figure 4: Chandrasekhar-H Equation: MG iterations plotted against the parameter value $\lambda \in [0.001, 0.999]$. Each square/circle represents one continuation step with adaptive stepsize and $m_{opt} = 3$.

4.3 Modified Two Dimensional Bratu Problem

The modified Bratu problem

$$\triangle v(x,y) + \kappa \frac{\partial v(x,y)}{\partial x} + \lambda e^{v(x,y)} = 0 \quad \text{in} \quad \Omega = [0,1] \times [0,1], \quad v = 0 \quad \text{on} \quad \partial\Omega, \tag{93}$$

where $v : \Omega \to \mathbb{R}$, κ a given parameter and λ a parameter is discussed in [7, p. 310f]. We seek a solution $v(\lambda)$.

Let Ω_l be uniform grids on $[0,1] \times [0,1]$ and fix λ for ease of notation. We discretize (93) on levels $l \in \mathbb{N}_{\geq 0}$ on the grids Ω_l using the grid function u with $n_l := |\Omega_l|$ as in (5), $M_l := \sqrt{n_l}$ and

$$u_{l,i,j} := v(ih_l, jh_l), \quad 0 \leq i,j \leq M_l - 1.$$

Furthermore, assume

$$u_{l,i,j} = 0 \quad \text{for} \quad \{i,j\} \cap \{0, M_l - 1\} \neq \emptyset \tag{94}$$

to satisfy the boundary values in (93). The differential operators $\triangle$ and $\frac{\partial}{\partial x}$ are discretized using *first and second order centered difference approximations* which are defined by

$$D(u_{l,i,j}) := \frac{1}{2h_l}(u_{l,i+1,j} - u_{l,i-1,j}) \tag{95}$$

$$D_2(u_{l,i,j}) := \frac{1}{h_l^2}(-4u_{l,i,j} + u_{l,i-1,j} + u_{l,i+1,j} + u_{l,i,j-1} + u_{l,i,j+1}), \tag{96}$$

$$\tag{97}$$

Defined opt. # of iterations m_{opt}	Comp. time in seconds	Continuation Steps	CG Iterations from Φ	MG iterations
2	**46.4690**	**29**	**224**	**79**
2.12_{FAS}	56.4220	38	149	98
3	**28.1250**	**12**	**121**	**49**
2.12_{FAS}	46.5930	21	173	80
4	**26.7650**	**10**	**123**	**46**
2.12_{FAS}	35.7500	12	159	59

Table 3: Chandrasekhar-H Equation: Certain performance measures for the continuation with algorithm 2.12 (and algorithm 2.12_{FAS} in the respective rows) and adaptive stepsize.

respectively, with $0 < i, j < M_l - 1$. Hence, the resulting system of nonlinear equations is

$$N_{l,i,j}(u_l, \lambda) = D_2(u_{l,i,j}) + \kappa \cdot D(u_{l,i,j}) + \lambda e^{u_{l,i,j}} = 0,$$

with $0 < i, j < M_l - 1$. For implementation we use lexicographic ordering of the grid functions u_l with the new index $k := (i-1)+(j-1)(M_l-2)$ and regard only indexes i, j with $0 < i, j < M_l - 1$, i.e the inner points of the grids Ω_l, hence automatically satisfying the boundary value condition in (93). Regarding (94), this results in a system of $(M_l - 2)^2$ equations

$$N_{l,k}(u_l, \lambda) = D_2(u_{l,k}) + \kappa \cdot D(u_{l,k}) + \lambda e^{u_{l,k}} = 0, \tag{98}$$

which is to be solved. For smoothing with Algorithm 2.2 we also need

$$\frac{\partial N_{l,k}(u_l, \lambda)}{\partial u_{l,k}} = -\frac{4}{h_l^2} + \lambda e^{u_{l,k}}, \quad 0 \le k < (M_l - 2)^2$$

and hence for $0 \le k < (M_l - 2)^2$, line 4 from algorithm 2.2 can be written as

$$u_{l,k} = u_{l,k} - [D_2(u_{l,k}) + \kappa \cdot D(u_{l,k}) + \lambda e^{u_{l,k}}] \cdot \left(-\frac{4}{h_l^2} + \lambda e^{u_{l,k}}\right)^{-1}.$$

The treatment of this problem is very similar to the treatment of the Chandrasekhar-H equation in section. Therefore we will present the results in the same manner, with less explanation.

Matching the conditions from [8, Section 6.2.3, p. 98] in most aspects, the options from table 1 are set as

$$\begin{gathered} \Lambda = [0.1, 6.8], \quad u_L^0(a) = \mathbf{1} = (1, 1, \ldots, 1)^T \in \mathbb{R}^{n_L}, \quad s = 0.3, \\ k = 2, \quad \hat{k} = 2 \quad \varepsilon = 10e - 8. \end{gathered} \tag{99}$$

During the computations we found that using predictors of orders higher than 2 (linear predictor) yields no further improvement for this specific problem, hence we set $k = 2$ in (99) as opposed to $k = 4$ in (91). The coarsest grid Ω_0 is a 64 point grid. For smoothing we use algorithm 2.2. Computations are done on $2^{k_l} \times 2^{k_l}$ point grids $\Omega_1, \ldots, \Omega_L$ with $k_0 = 3$ and the largest L being 7. The corresponding grid sizes are

$$n_1 = 256,\ n_2 = 1024,\ n_3 = 4096,\ n_4 = 16384,\ n_5 = 65536,\ n_6 = 262144,\ n_7 = 1048576.$$

We set $\kappa = 10$ in (93), and measure again the computation times for the complete continuation (24 continuation steps) with constant stepsize. Table 4 displays the results.

For comparison, the results for a 1024 point grid with $\kappa = 0$ and $\Lambda = [0.1, 3.5]$ from [8] are used. The comparison should be fair, because all modifications in our paper make the problem harder to solve and most importantly, the continuation in [8] only covers about half the interval. Here the fastest of many methods tested in [8] yielded a computation time of 1020.4840 seconds, which means algorithm 2.12 is more than 200 times faster in that application.

Grid points (L)	Comp. time in seconds	Comp. time $\mathcal{S}^{(\nu_i)}$ in %	Comp. time P_l, R_l in %	Comp. time CG It. in %	CG It. of Φ	MG iterations
1024 (2)	**3.3440**	6.45	3.85	70.70	**197**	**93**
2.12_{FAS}	4.8620	9.18	2.63	73.36	291	130
4096(3)	**6.1870**	15.27	4.21	61.02	**329**	**90**
2.12_{FAS}	9.2180	13.35	5.77	67.92	527	124
16384 (4)	**11.7500**	26.60	9.07	43.51	**470**	**81**
2.12_{FAS}	17.4680	24.05	6.92	48.81	775	109
65536 (5)	**30.1870**	36.91	13.65	22.35	**627**	**73**
2.12_{FAS}	44.9380	35.42	12.91	25.02	1074	107
262144 (6)	**93.1090**	42.39	14.88	9.03	**825**	**68**
2.12_{FAS}	128.4070	42.56	15.35	11.23	1389	93
1048576 (7)	**342.6090**	45.60	15.57	3.67	**1189**	**64**
2.12_{FAS}	449.3430	44.72	15.84	4.45	1849	84

Table 4: Modified Bratu Problem: Computational time of algorithm 2.12 (and algorithm 2.12_{FAS} in the respective rows). Measurements were taken as an average over 5 runs. The percentwise computation times in columns 3-5 were given through the `Matlab Profile Viewer` tool. Percentages less than 1% are noted as 0.

If one compares tables 4 and 2 it becomes clear that the shares in computation times of prolongation P and restriction R have opposite tendencies between the Chandrasekhar-H equation in table 2 and the modified Bratu problem in table 4. This is caused by the more complicated nature of restriction and prolongation on a two dimensional grid (2D) compared to a one dimensional grid (1D). In the former case, `if`-statements have to be evaluated in every call to the operator N_l to accommodate for the boundary values. Moreover, on a 1D grid, 2 neighboring points are considered for each grid point, whereas on a 2D grid 8 points are used.

We plot the number of MG iterations against the parameter value λ comparing algorithms 2.12 and 2.12_{FAS} in figure 5 on the grid Ω_4 with 16384 grid points. Because of the larger amount of continuation steps (24), we sketch both graphs in one plot, similar to figure 4, with constant stepsize s.

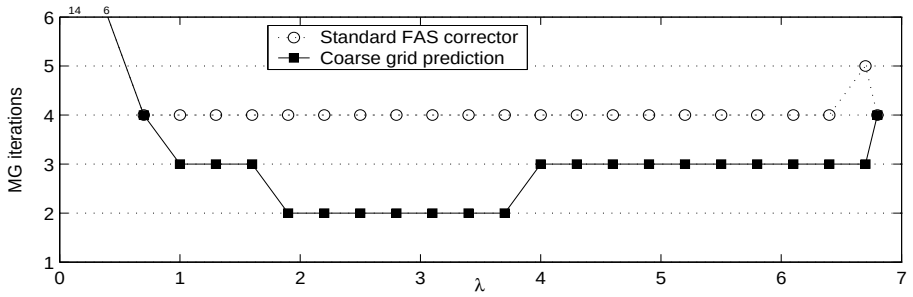

Figure 5: Modified Bratu Problem. MG iterations plotted against the parameter value $\lambda \in [0.1, 6.8]$. Each square/circle represents one continuation step with constant stepsize $s = 0.3$.

The fact that in figure 5, the number of iterations with CGP is nearly everywhere less than without it, again suggests to apply the stepsize control (92). This is plotted in figure 6. Again only parameter values starting at 0.4 are plotted because no coarse stepsize adaption happens before that (see section 4.2). The value $m_{opt} = 3$ is used in figure 6.

For performance measures for other values of m_{opt} see table 5. These computations were again done on Ω_4.

Figure 6 deserves a further remark. At a first look it may seem that for some parameter values a larger amount of MG iterations is needed with CGP than without it. However, in these occasions, the stepsize in the continuation plot with CGP is already much larger than without it. Hence, the corrector certainly needs a few more iterations for a possibly less accurate prediction. This is especially obvious between parameter values $\lambda \in [2, 3]$ and $\lambda \in [5, 6]$.

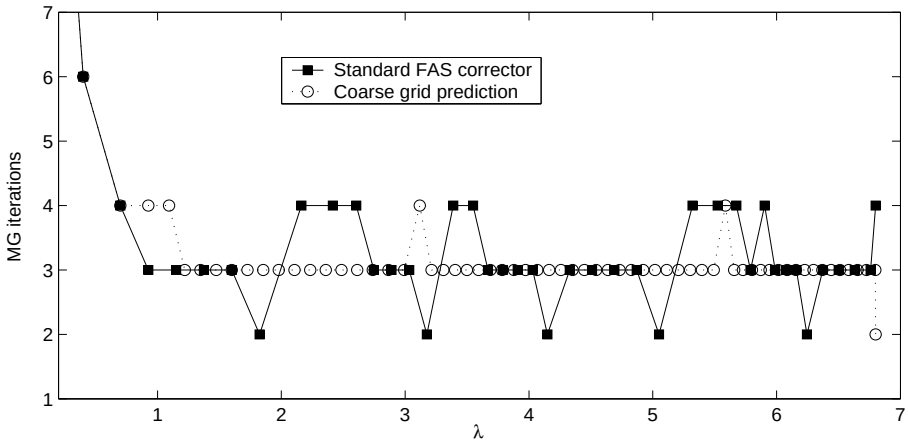

Figure 6: Modified Bratu Problem: MG iterations plotted against the parameter value $\lambda \in [0.1, 6.8]$. Each square/circle represents one continuation step with adaptive stepsize and $m_{opt} = 3$.

Defined opt. # of iterations m_{opt}	Comp. time in seconds	Continuation Steps	CG Iterations from Φ	MG iterations
3	**21.718**	**41**	**896**	**143**
2.12_{FAS}	33.7030	65	1440	213
4	**11.516**	**15**	**510**	**70**
2.12_{FAS}	17.5940	24	775	109
5	**9.5940**	**11**	**428**	**58**
2.12_{FAS}	12.5320	14	561	77

Table 5: Modified Bratu Problem: Certain performance measures for the continuation with algorithm 2.12 (and algorithm 2.12_{FAS} in the respective rows) and adaptive stepsize.

4.4 Continuation of Quasiperiodic Invariant Tori

The following application is motivated by [9], where complete coverage of the theoretical background can be found. In this context, we shall just briefly note the parts interesting for applying the MG continuation algorithm 2.12 to the problem of continuation of quasi periodic invariant tori for a specific ODE.

Let

$$\dot{x} = f(x, \omega_1 t, \omega_2 t), \quad f : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^d \tag{100}$$

an ordinary differential equation, $\omega := (\omega_1, \omega_2)$ and assume that f is periodic with period 2π in its second and third argument, i.e.

$$f(x, \theta_1, \theta_2) = f(x, \theta_1 + 2\pi, \theta_2) = f(x, \theta_1, \theta_2 + 2\pi)$$

for all $(x, \theta_1, \theta_2) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}$. We require the *frequencies* ω_1 and ω_2 to be rationally independent, i.e. $k_1\omega_1 + k_2\omega_2 \neq 0$ for all $k_1,\ k_2 \in \mathbb{Q}$.

Let $\mathbb{T}^m := \mathbb{R}^m/(2\pi\mathbb{Z})^m$ for $m > 0$. Transforming (100) into an autonomous differential equation yields

$$\dot{x} = f(x, \vartheta), \quad \dot{\vartheta} = \omega, \tag{101}$$

with $(x, \vartheta) \in \mathbb{R}^d \times \mathbb{T}^2$. Let $f \in \mathcal{C}^r(\mathbb{R}^d \times \mathbb{T}^2, \mathbb{R}^d)$ for $r \in \mathbb{N}$ and let φ_t be the flow that is induced by (101). In [9, Section 2.3.2, p. 29ff] it is shown that

$$\omega_1 \frac{\partial v}{\partial \theta_1} + \omega_2 \frac{\partial v}{\partial \theta_2} = f(v, \theta_1, \theta_2), \quad \theta := (\theta_1, \theta_2) \in \mathbb{T}^2 \tag{102}$$

is the determining equation for a torus function $v : \mathbb{T}^2 \to \mathbb{R}^d$ of an, with respect to φ_t, invariant quasiperiodic torus. The question of existence of such a torus is also discussed in [9].

For testing algorithm 2.12 we look at a specific, parameter dependent function f. Let $d = 4$ and the following system $\dot{x} = f(x, \theta_1, \theta_2, \lambda)$ with

$$\begin{aligned} f_1(x, \theta_1, \theta_2, \lambda_1) &= -\lambda_1 x_1 - (\omega_1 + 1)\, x_2 + x_1(x_1^2 + x_2^2) + \sqrt{\lambda_1} \sin\theta_1 \\ f_2(x, \theta_1, \theta_2, \lambda_1) &= -\lambda_1 x_2 + (\omega_1 + 1)\, x_1 + x_2(x_1^2 + x_2^2) - \sqrt{\lambda_1} \cos\theta_1 \\ f_3(x, \theta_1, \theta_2, \lambda_1) &= -\lambda_2 x_3 - (\omega_2 + 1)\, x_4 + x_3(x_3^2 + x_4^2) + \sqrt{\lambda_2} \sin\theta_2 \\ f_4(x, \theta_1, \theta_2, \lambda_1) &= -\lambda_2 x_4 + (\omega_2 + 1)\, x_3 + x_4(x_3^2 + x_4^2) + \sqrt{\lambda_2} \cos\theta_2 \end{aligned} \tag{103}$$

as well as $\omega_1 = 1$, $\omega_2 = \sqrt{5}$, $\lambda_2 = 10$ and $\lambda_1 = \lambda$ the parameter given similar to [9, Section 3.1.3, p. 70ff]. The invariance equation (102) with the special f from (103) yields

$$\omega_1 \frac{\partial v_i}{\partial \theta_1} + \omega_2 \frac{\partial v_i}{\partial \theta_2} = f_i(v, \theta_1, \theta_2, \lambda), \quad i = 1 \dots 4, \tag{104}$$

which has a solution $v = (v_1, v_2, v_3, v_4)^T$ with

$$v_1(\theta) = \sqrt{\lambda_1} \cos\theta_1,\ v_2(\theta) = \sqrt{\lambda_1} \sin\theta_1,\ v_3(\theta) = \sqrt{\lambda_2} \cos\theta_2,\ v_4(\theta) = \sqrt{\lambda_2} \sin\theta_2 \tag{105}$$

To solve (104) we discretize the Torus $\mathbb{T}^2$ through an $M_l \times M_l$-point grid with the respective grid sizes and grid functions

$$h_l = \frac{2\pi}{M_l - 1}, \quad u_{l,i,j} = v(ih_l, jh_l), \quad i, j = (0 \dots M_l - 1) \mod M_l. \tag{106}$$

The latter modulo notation means that by the periodic nature of $\mathbb{T}^2$, indexes of the grid functions have to be taken modulo M_l, e.g. index $i = M_l$ is equivalent to index $i = 0$. This also means that in our discretization we disregard the "upper" and the "right" boundary and store the periodic boundary values in the other two. This choice is arbitrary.

With $\alpha := \frac{\omega_1}{2h_l}$, $\beta := \frac{\omega_2}{2h_l}$ and finite difference discretization of (104) we gain the the M_l^2 equations

$$\begin{aligned} \alpha\, u_{l,i+1,j} - \alpha\, u_{l,i-1,j} + \beta u_{l,i,j+1} - \beta u_{l,i,j-1} &= f(u_{l,i,k}, ih_l, jh_l, \lambda) \\ i, j = (0 \dots M_l - 1) \mod M_l. \end{aligned} \tag{107}$$

Note that in (107), each $u_{l,i,j}$ is a vector in $\mathbb{R}^4$. Hence, for implementation we use lexicographical ordering with the index k and have a system of $4M_l^2$ scalar, nonlinear, parameter dependent equations which we can directly apply algorithm 2.12 to.

For reasons described in section 4.4, we use the Richardson smoothing algorithm 2.3 for this particular problem. Hence, for smoothing, line 2 from algorithm 2.3 is given as

$$u_l = u_l - \omega h_l N_l(u_l, f_l),$$

for some $\omega \in (0, \frac{1}{2}]$. In (107) we have written out N_l, and hence the smoothing calculation is given as

$$\begin{aligned} u_{l,i,j} = u_{l,i,j} - \omega h_l \,(u_\alpha\, u_{l,i+1,j} - \alpha\, u_{l,i-1,j} + \beta u_{l,i,j+1} - \beta u_{l,i,j-1} - f(u_{l,i,k}, ih_l, jh_l, \lambda)) \\ i, j = (0 \dots M_l - 1) \mod M_l. \end{aligned}$$

The choice of ω depends on the given problem and can be used to fine-tune the behavior of the MG algorithm. In [4, Section 3.3.2, p. 52f] the choice of ω is discussed further. Matching the conditions from [9, Section 3.1.3, p. 70ff] the options from table 1 are set as

$$\begin{aligned} \Lambda = [9, 25], \quad u_L^0(a) = 0.01 \cdot (1, 1, \dots, 1)^T \in \mathbb{R}^{n_L}, \quad s = 2, \\ k = 1, \quad \hat{k} = 1 \quad \varepsilon = 10e - 5. \end{aligned} \tag{108}$$

We found that for this problem the simple predictor which uses only the previous value is best with and without CGP. This may be caused by the weak dependency of the solution (105) on the parameter λ. Even a linear predictor may give an initial solution too far away from the solution.

Compared to the previous two sections a few aspects change for this problem. At first, the periodic boundary values should be mentioned. The experiments showed that the smoothing algorithm 2.2 is not well suited. It seems that the periodic boundary values cause the smoother to diverge. However, algorithm 2.3 with $\omega = 1/8$ has the desired smoothing effect and hence we use it for smoothing in this application.

The coarsest grid Ω_0 is a $8 \times 8 = 64$ point grid, i.e. $M_0 = 8$ (M_l from (106)). The coarse grid solver (16) always operates on this grid, i.e. solves systems of $64 \cdot 4 = 256$ equations. Computations are done on $2^{k_l} \times 2^{k_l}$ point grids with each grid point in $\mathbb{R}^4$, $k_0 = 3$ and the largest L being 4. The grids are $\Omega_1, \ldots, \Omega_L$ with corresponding grid sizes

$$\begin{gathered} n_1 = 1024\ (16 \times 16),\ n_2 = 4096\ (32 \times 32),\ n_3 = 16384\ (64 \times 64), \\ n_4 = 65536\ (128 \times 128). \end{gathered} \tag{109}$$

Again we measure the computation times for the complete continuation (11 continuation steps) with constant stepsize on the grids (109).

Grid points (L)	Comp. time in seconds	Comp. time $\mathcal{S}^{(\nu_i)}$ in %	Comp. time P_l, R_l in %	Comp. time CG It. in %	CG It. of Φ	MG it.
1024 ($16 \times 16 \times 4$)	**4.8390**	2.27	6.88	86.87	**26**	**37**
2.12_{FAS}	5.2820	2.38	6.21	85.47	29	40
4096($32 \times 32 \times 4$)	**8.2500**	9.51	26.84	63.64	**29**	**50**
2.12_{FAS}	9.3590	8.82	25.52	64.13	32	56
16384 ($64 \times 64 \times 4$)	**21.2660**	17.19	62.98	26.45	**33**	**65**
2.12_{FAS}	25.2970	19.63	58.60	27.18	39	76
65536 ($128 \times 128 \times 4$)	**94.8910**	24.47	76.67	6.52	**35**	**90**
2.12_{FAS}	106.0780	23.70	75.69	7.69	40	99

Table 6: Continuation of quasiperiodic invariant tori: Computational time of algorithm 2.12 (and algorithm 2.12_{FAS} in the respective rows). Measurements were taken as an average over 5 runs. The percentwise computation times in columns 3-5 were given through the `Matlab Profile Viewer` tool.

For comparison we use the computations in [9, Section 3.1.3, p. 70f]. The reference does not give computation times for continuation, only for the parameter values $\lambda_1 = \lambda_2 = 9$. Computations in [9] were done on a Pentium III running at 800Mhz, therefore we apply the factor 3 to our computation times.

In this respect, on a 128×128 point grid, algorithm 2.12 needed 34 MG iterations of algorithm 2.10 and $3 \cdot 35.2030sec = 105.609sec$ for finding a solution. The computations in [9] took 4 iterations and $836.7sec$ on a 100×100 point grid. This means our algorithm performs about 8 times faster.

Because in table 6, use of CGP always results in fewer MG iterations, in figure 7 we plot the number of MG iterations on the 128×128 point grid Ω_4 for each continuation step as in figures 3 and 5 to see where CGP saves MG iterations.

In figure 7 we find that in the continuation, the number of MG iteration reduces faster with CGP than without it. That is a promising result in its own respect. Figure 8 however, shows the real advantage of CGP if the heuristic stepsize control (92) is used on the 128×128 point grid Ω_4.

Certainly, (92) yields different results for different values m_{opt}. We chose $m_{opt} = 4$ for figure 8 because it was best suited for display. Additionally, performance measures for the values $m_{opt} = 3, 5$ are displayed in table 7 to give an idea of how the choice of m_{opt} affects the continuation.

The results from table 7 suggest that the smaller we chose m_{opt} (i.e. the less MG iterations we allow in each continuation step), the more of an advantage CGP has over the standard FAS corrector.

Remark 4.1 (Use of different smoothing algorithms)**.** Note that we have used different smoothing algorithms for the three problems discussed above. Aside from performance reasons, in the special case of

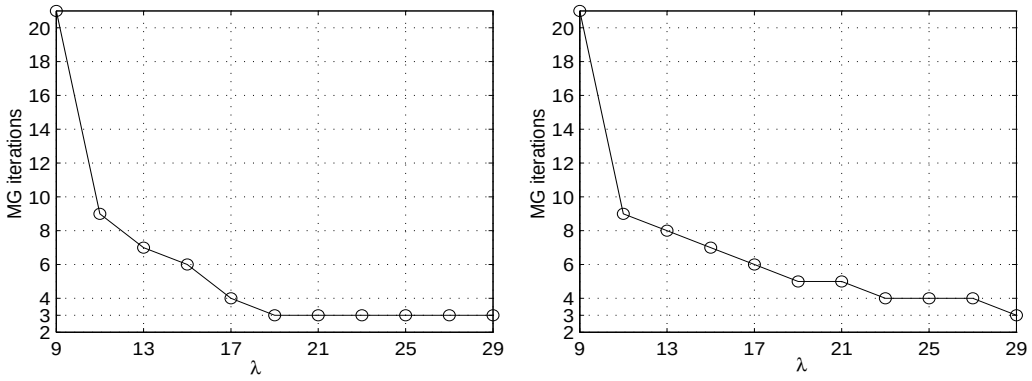

Figure 7: Continuation of quasiperodic invariant tori: Continuation with constant stepsize $s = 2$ on Ω_4. MG iterations plotted against the parameter value $\lambda \in [9, 29]$. Left: Algorithm 2.12 (i.e. use of CGP). Right: Algorithm 2.12_{FAS}

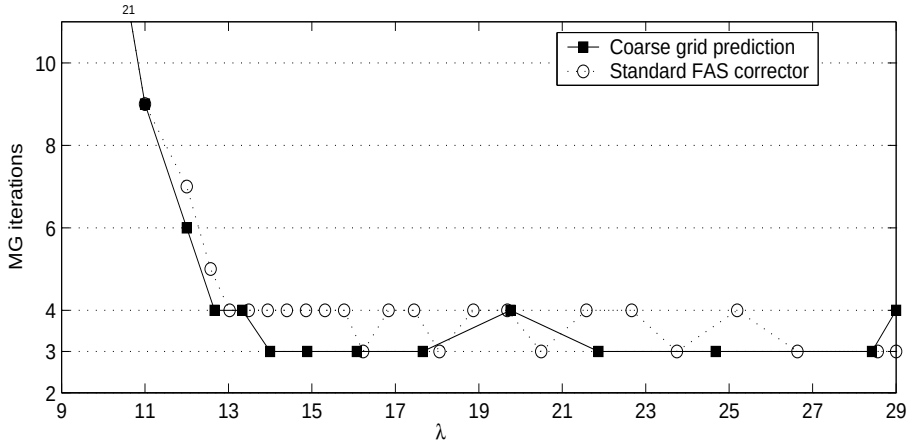

Figure 8: Continuation of quasiperiodic invariant tori: MG iterations plotted against the parameter value $\lambda \in [9, 29]$ on Ω_4. Each square/circle represents one continuation step with adaptive stepsize and $m_{opt} = 4$.

the continuation of quasiperiodic tori this is due to the fact that the Gauss-Seidel smoothing algorithm 2.2 yielded strong divergence as has been mentioned in section 4.4. Therefore we used algorithm 2.3 for that particular problem.

In our tests we found that one may always apply the Richardson iteration (algorithm 2.3) for smoothing. This may not always yield the best performance, but it seemed to have a smoothing effect in all applications and is easy to implement since only evaluations of N_l are required. Hence as a start, we recommend trying algorithm 2.3 for smoothing first, and from there try other smoothing algorithms to possibly improve performance.

Remark 4.2 (Dependency of computation time on the amount of grid points in Ω_L)**.** We are looking for a quantity which, the larger it gets, higher the computational time for finding a solution is expected to be. It seems reasonable to choose the amount of grid points in the fine grid. With this in mind, we have plotted the dependence of computation times on the amount of grid points in figure 9 for all three test problems with appropriate continuous fitting curves.

Defined opt. # of iterations m_{opt}	Comp. time in seconds	Continuation Steps	CG Iterations from Φ	MG iterations
3	**34.15**	**26**	**54**	**103**
2.12_{FAS}	84.8910	75	133	255
4	**23.829**	**14**	**37**	**73**
2.12_{FAS}	42.2500	25	80	119
5	**21.547**	**11**	**34**	**66**
2.12_{FAS}	32.9690	16	60	95

Table 7: Continuation of quasiperiodic invariant tori: Certain performance measures for the continuation with algorithm 2.12 (and algorithm 2.12_{FAS} in the respective rows) and adaptive stepsize.

5 Conclusion

Based on the Multi-Grid (MG) algorithm Full Approximation Scheme (FAS) we developed a modification in section 2, which we called Coarse Grid Prediction (CGP). The corresponding procedure 2.10 was embedded into a simple Predictor-Corrector (PC) continuation algorithm 2.12. In this continuation algorithm we use two predictors, one for the solution, and one for the first coarse grid correction as illustrated in figure 1.

In section 3 we could prove convergence of the corrector 2.10 in every continuation step of algorithm 2.12 under reasonable conditions. We did not focus on specifics of the continuation such as bifurcation detection and verification of the solveability of the continuation These topics are briefly covered in [4, p. 270-276] for the nested type of MG iterations.

The concept of our CGP predicts the first correction on the fine grid Ω_L coming from the coarse grid solution on the grid Ω_0. In this aspect, CGP is specific to MG algorithms. However, one could also look at it as a prediction for the first correction in an iterative algorithm, regardless of where that correction originates from. Let g be an iterative algorithm. Furthermore, assume that an initial approximation $u^0(\lambda_j)$ for the parameter value λ_j is given through some prediction method and that g produces iterates

$$u^m(\lambda_j) = g(u^{m-1}, \lambda_j), \quad m > 0\,. \tag{110}$$

One could also store, for some $k > 0$,

$$v(\lambda_{j-1}) := u^1(\lambda_{j-1}) - u^0(\lambda_{j-1}), \ldots, v(\lambda_{j-k}) := u^1(\lambda_{j-k}) - u^0(\lambda_{j-k}),$$

and in (110) use $u^0(\lambda_j) + v(\lambda_j)$ as the initial approximation, where $v(\lambda_j)$ is predicted from $v(\lambda_{j-k}), \ldots, v(\lambda_{j-1})$.

In Section 4 we applied our continuation algorithm to three different numerical problems. The results show that for constant stepsize, the use of CGP has an advantage of 10% to 30% in computation time over the use of the standard FAS as a corrector as can be observed from tables 2, 4, 6. However, the more important conclusion that can be drawn is that the use of CGP results in less MG iterations, as it can also be observed from the aforementioned tables.

Following this observation consequently, and because the amount of MG steps needed is a direct measure for the computational time, we applied a heuristic stepsize control which adapts the stepsize based on the amount of MG iterations needed compared to a given reference number m_{opt} of iterations. Using this, CGP showed potential, by achieving peaks of less than half the computation times (e.g. table 7) than FAS. Even on average, CGP needed about $2/3^{rd}$ of the computation times of the standard FAS corrector.

Aside from comparisons between CGP and the FAS as a corrector, the continuation algorithm 2.12 was also briefly compared to Newton-Krylov (NK) algorithms from [8] and computations from [9]. In both cases algorithms 2.12 and 2.10 are significantly faster than the NK algorithms used in the references. However, we must refrain this judgment to the measure of computation times and the three problems tested. No other aspects were compared.

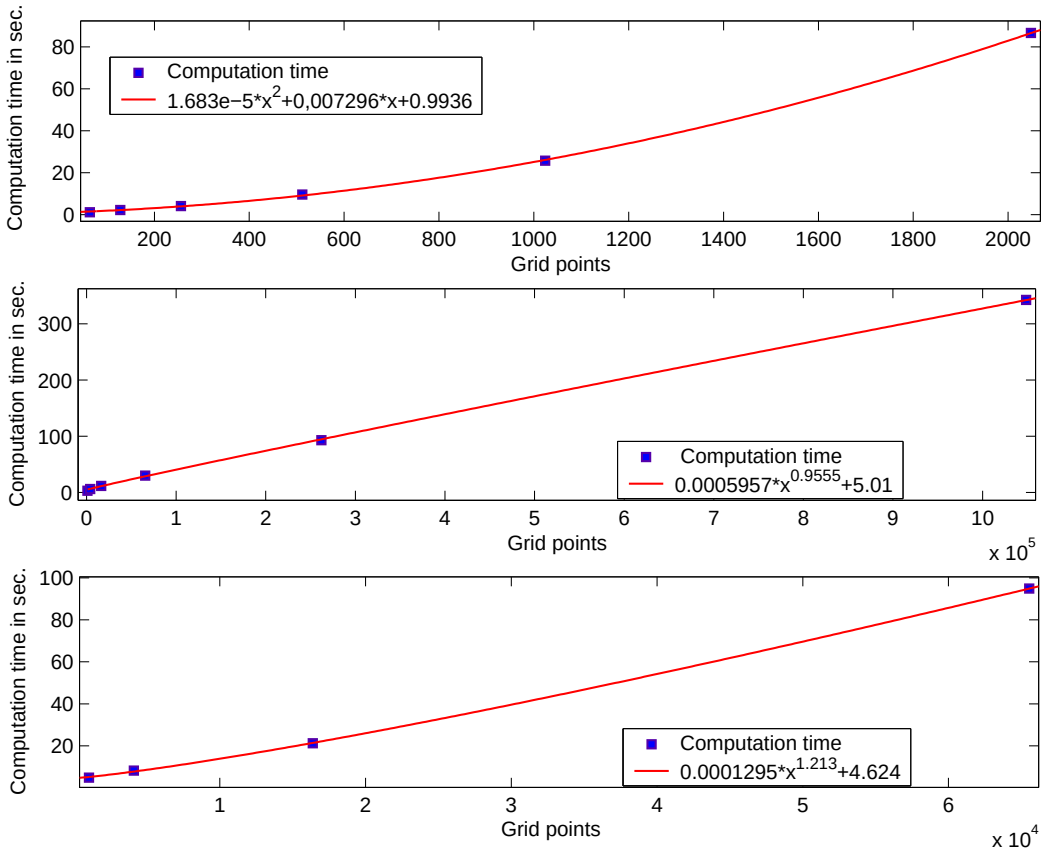

Figure 9: Plots and respective curve fittings of the computation times from tables 2,4 and 6 in that order from top to bottom. To obtain a fit for the middle plot, the first value has been excluded. From top to bottom, the plots show the Chandresekhar-H equation, the modified Bratu problem and the continuation of quasiperiodic invariant tori. All plots have been created using the MATLAB CURVEFITTING TOOLBOX.

As a final note, we recommend trying the FAS algorithm for nonlinear systems of equations of any type, as long as there exist reasonable versions of the system on different grids. For continuation, the use of CGP seems to have an advantage over using the "black boxed" FAS as a corrector.

6 Contact

Sebastian Schmidt, Fraunhofer Institut ITWM,
Strömungen und komplexe Strukturen
Gottlieb-Daimler-Strasse, Geb. 49
67663 Kaiserslautern, Germany
(schmidt@itwm.fraunhofer.de)

Werner Vogt, Technische Universität Ilmenau,
Institut für Mathematik,
Postfach 100565,
98684 Ilmenau, Germany
(werner.vogt@tu-ilmenau.de)

References

[1] D. A. Knoll, D. E. Keyes: *Jacobian-free newton-krylov methods: A survey of approaches and applications.* Journal of Computational Physics, 193:357–397, 2002.

[2] E. L. Allgower, K. Georg: *Numerical Continuation Methods.* Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, 1980.

[3] H. Ammann, J. Escher: *Analysis II.* Birkhäuser, Basel, Boston, Berlin, erste Auflage, 1999.

[4] Hackbusch, W.: *Multi-Grid Methods and Applications.* Springer-Verlag, Berlin, Heidelberg, 1st edition, 1985.

[5] Hackbusch, W.: *Comparison of different multi-grid variants for nonlinear equations.* ZAMM, 72:148–151, 1992.

[6] Kelley, C. T.: *Iterative Methods for Linear and Nonlinear Equations.* SIAM, Philadelphia, 1995.

[7] M. Pernice, H. F. Walker: *Nitsol: A newton iterative solver for nonlinear systems.* SIAM, 19:302–318, 1998.

[8] Meyer, S.: *Numerische Lösungsfortsetzung parameterabhängiger nichtlinearer Gleichungssysteme mit Newton-Krylov-Verfahren.* Diplomarbeit, Technische Universität Ilmenau, 2005.

[9] Schilder, F.: *Numerische Approximation quasiperiodischer invarianter Tori unter Anwendung erweiterter Systeme.* Dissertation, Technische Universität Ilmenau, 2004.

[10] U. Trottenberg, C. W. Oosterlee, A. Schüller: *Multigrid.* Academic Press, San Diego, San Francisco, New York, Boston, London, Sydnay, Tokyo, 2000.