4
Numerical Analysis

4.1
An Inversion Algorithm

In this section, we begin to describe an algorithm to solve

$$-\Delta u - f(u) = g, \quad u|_{\partial\Omega} = 0.$$

This equation is interpreted as the computation of the preimages of g under $F : H_0^1(\Omega) \to H^{-1}(\Omega)$.

The discretizations will be performed by choosing appropriate finite elements in $H_0^1(\Omega)$.

Our main hypothesis is on the nonlinearity: $f : \mathbb{R} \to \mathbb{R}$ is a C^1 function interacting with a complete set J. From the previous section, the function $F : H_0^1(\Omega) \to H^{-1}(\Omega)$ is essentially flat with respect to the orthogonal decompositions $W_1 \oplus V_1$ and $W_{-1} \oplus V_{-1}$.

In a nutshell, we follow the advantages of having a global Lyapunov-Schmidt decomposition: split $g = g_W + g_V = P_{-1}g + Q_{-1}g$. The vertical affine space $g_v + V_{-1}$, when inverted under F, gives rise to a fiber α_g which contains all the solutions of the original equation. The algorithm first identifies a point in α_g — the search of solutions then boil down to a finite dimensional problem, which correspond essentially to the bifurcation equations associated to the decomposition.

4.1.1
Moving in The Space of Fibers

Starting from a point $u_0 \in X$, our first goal is to reach a point u_N in the fiber α_g, or more realistically in a fiber close to it.

From Proposition 3, identifying α_g is equivalent to solving

$$F_v(w) = P_{-1}g, \tag{4-1}$$

for an arbitrary $v \in V_1$. Since F is essentially flat, it would seem natural to use Newton's method to advance horizontally until we reach the α_g. This

PUC-Rio - Certificação Digital Nº 0812261/CA

requires that we invert the operator $F'_v(w)$, defined in W, so that a suitable finite element discretization of this space would need to be designed. What we propose instead is to work in the full spaces $X,\ Y$, where a standard finite-element space can be used. Let us point out that working with F' in its full domain comes at a high cost near critical points.

We need an invertible extension of the derivative of F_v to the whole space. This not only saves work but also ensures that the matrices arising in the process remain sparse.

Definition 5. *For $u_0 \in H^1_0(\Omega)$ we define the linear operator $L(u_0)$ from $H^1_0(\Omega)$ to $H^{-1}(\Omega)$ by*

$$L(u_0)\, z = -\Delta z - P_{-1} f'(u_0) P_1 z.$$

Proposition 8. *The operator $L(u_0)$ extends $F'_{v_0}(w_0)$ to the whole space isomorphically. In fact, its restriction to W_1 (resp. V_1) is an isomorphism to W_{-1} (resp. V_{-1}).*

Proof. For $z_w = P_1 z,\ z_v = Q_1 z$, we have

$$L(u_0)\, z = L(u_0)\,(z_w + z_v) = L(u_0)\,|_{W_1} z_w + L(u_0)\,|_{V_1} z_v.$$

Next, for a given $u_0 = w_0 + v_0 \in W_1 \oplus V_1$, we note that $L(u_0)\,|_{W_1} = F'_{v_0}(w_0)$, which is an isomorphism from W_1 to W_{-1}. To conclude the argument, we claim that $L(u_0)\,|_{V_1}$ is also an isomorphism between V_1 and V_{-1}, since it is given by $v \mapsto \lambda_k v$. □

In our search for α_g, we update a given $u_n \in H^1_0(\Omega)$ by setting $u_{n+1} = u_n + h$, where h solves

$$L(u_n)\, h = P_{-1}(g - F(u_n)). \tag{4-2}$$

According to Proposition 8, the function h above is the step obtained from Newton's method, since the right-hand side is in W_{-1}.

At this point we would like to add a remark concerning the actual numerical work. We already took care of the difficulties of solving a system in the horizontal spaces by extending F'_v, but still face two small hurdles: the need to project the right-hand side in (4-2) and the possibility of deviating from the subspace W_1 because of numerical errors introduced by solving the linear system. One idea that proved useful is to replace h by its projection $\tilde{h} = P_1 h$. The gain is that, since we are projecting the solution h, it can be chosen to solve

$$L(u_n)\, h = g - F(u_n) \tag{4-2*}$$

PUC-Rio - Certificação Digital Nº 0812261/CA

instead, eliminating the need for the projection in the dual space W_{-1}. This is a direct consequence of Propostion 8.

As previously noted, moving within a horizontal plane amounts to moving from one fiber to another, so we iterate u_0 horizontally until u_N is close enough to α_g. We can check this by computing the norm of the error $e_n = P_{-1}(g - F(u_n))$.

There is still one technicality to consider. The algorithm indeed proceeds like this, but not with respect to the Lyapunov-Schmidt decomposition $F : V_1 \oplus W_1 \to V_{-1} \oplus W_{-1}$. For the reasons presented in the end of the previous section, we use finite element subspaces X_h and Y_h (same set, different inner product), in which we choose $\tilde{V}_X = V_1^h$, $\tilde{W}_X = W_1^h = V_1^{h\perp}$, $\tilde{V}_Y = V_{-1}^h$, $\tilde{W}_Y = W_{-1}^h = V_{-1}^{h\perp}$, which are ϵ-close to the original subspaces. We then move horizontally for the restriction of the tilted Lyapunov-Schmidt decomposition, $F_h : V_1^h \oplus W_1^h \to V_{-1}^h \oplus W_{-1}^h$, which exists for small ϵ due to Corollary 2.

4.1.2 Moving Along a Fiber

Once α_g is identified, the original problem reduces to a finite-dimensional issue: set $k = \dim V_1 = \dim V_{-1}$. In a nutshell, we need how to walk along a fiber. This is accomplished by implementing the identification $\eta : V_1 \to \alpha_g$ defined in Proposition 3. More precisely, given a height $v \in V_1$, we evolve along the horizontal affine subspace $v + W_1$ with Newton´s method until we reach the point $u = \eta(v) \in \alpha_g$ with height v.

Thus, we may think of solving the original equation as inverting the point Qg under the function from $V_1 \simeq \mathbb{R}^k$ to $V_{-1} \simeq \mathbb{R}^k$ given by $v \mapsto \eta(v) \mapsto QF(\eta(v))$.

Despite of the finite dimensionality, the remaining problem is not a trivial issue. In the examples below, we mostly consider the one dimensional case $k = 1$, where moving along a fiber essentially means going up and down. When $k = 2$, one might in principle make use of the algorithms presented in [13]: we intend to consider the matter in a forthcoming paper. In this work, instead, we show in Section 5.3 how simple arguments sometimes are enough to still obtain a few solutions to a PDE.

As usual, the more we know about F, the more we can say about the numerics. In the simplest case, when the nonlinearity f interacts with $J = \{1\}$ and $f'' > 0$, the Ambrosetti-Prodi case, fibers are taken by F to vertical lines in a very simple fashion: as a point moves in the fiber so that its height goes from $-\infty$ to ∞, the height of its image goes monotonically from $-\infty$ to a maximal point and then decreases monotonically to $-\infty$. Dropping convexity allows for

PUC-Rio - Certificação Digital Nº 0812261/CA

the loss of monotonicity, and hence, variation in the number of solutions, but not of asymptotic behavior.

There are theoretical results [18] that guarantee that under stringent hypotheses the Ambrosetti-Prodi pattern along fibers carry through. The numerics may be performed in more general conditions.

4.2
Finite Elements and Linear Algebra

In this section we describe in a detailed manner how we assemble and solve the linear algebra systems arising from Newton's steps in the algorithm above. We consider a rectangular domain Ω, with sizes chosen so that the resulting eigenvalues are all simple (concretely, we worked with a 1 by 2 rectangle). Our finite element X_h will be the standard $\mathcal{P}_1$ space (for definitions and notations, see A) on Ω. We use a uniform triangulation, with vertices $\nu_i,\ i = 1, \ldots, N$, and a nodal basis of functions ψ_i. All numerical work was performed using Matlab. The routines that generate the relevant matrices were written from scratch, but whenever possible were checked against the Matlab analogues. We also used Matlab PDE Toolbox to generate our triangulation.

Expanding $u \in X_h$ in the nodal basis we obtain a vector $\underline{u}$ of coordinates:

$$u(x) = \sum_j \underline{u}_j \psi_j(x).$$

We also introduce the notation $\hat{u}$ for the vector whose coordinates are the L^2 inner products of the function u and the nodal basis:$\hat{u}_i = \langle \psi_i, u \rangle_0$. The *mass matrix* M with entries $M_{ij} = \langle \psi_i, \psi_j \rangle_0$ changes coordinates:

$$M\,\underline{u} = \hat{u}.$$

With this notation, the (discrete) counterpart of Poisson's equation $-\Delta u = g$ with Dirichlet boundary conditions in the space X_h amounts to solving the linear system

$$K\,\underline{u} = \hat{g},$$

where the *stiffness matrix* K is given by $K_{ij} = \langle \psi_i, \psi_j \rangle_1$.

4.2.1
The Discrete Nonlinear Problem

We return to the nonlinear problem $F(u) = -\Delta u - f(u) = g$. In principle, $F(u)$ is an element of the dual space $H^{-1}(\Omega)$, but we will restrict ourselves to

PUC-Rio - Certificação Digital Nº 0812261/CA

right-hand sides g which are square integrable, so we have to solve

$$\begin{aligned}\hat{\mathrm{F}}_i &= \langle F(u), \psi_i\rangle = \langle u, \psi_i\rangle_1 - \langle f(u), \psi_i\rangle_0 \\ &= \sum_j \langle \psi_j, \psi_i\rangle_1 \underline{\mathrm{u}}_j - \langle f(u), \psi_i\rangle_0 = \mathrm{K}_i\, \underline{\mathrm{u}} - \langle f(u), \psi_i\rangle_0 = \hat{\mathrm{g}}_i.\end{aligned}$$

At this point we replace $f(u)$ by its linear interpolator $\sum_j f(\underline{\mathrm{u}}_j)\psi_j$, so that the algebraic nonlinear system becomes

$$\mathrm{K}\,\underline{\mathrm{u}} - \mathrm{M}\, f(\underline{\mathrm{u}}) = \hat{\mathrm{g}}, \tag{4-3}$$

where $f(\underline{\mathrm{u}})$ is the vector whose coordinates are $f(\underline{\mathrm{u}}_j)$.

4.2.2
Discretizing the Extended Projected Operator

We recall that to move horizontally towards a fiber α_g, we solve a continuous linear equation with the operator $L(u_0) : z \mapsto -\Delta z - P_{-1} f'(u_0) P_1 z$. In this subsection, we introduce its discrete counterpart L_0.

Denote by $w = L(u_0)\, z$, so that, taking the L^2-inner product with a nodal function ψ_i, we obtain scalar equations

$$\hat{\mathrm{w}}_i = \mathrm{K}_i\, \underline{\mathrm{z}} - \sum_j \langle P_{-1} f'(u_0) P_1 \psi_j, \psi_i\rangle \underline{\mathrm{z}}_j = \mathrm{K}_i\, \underline{\mathrm{z}} - \mathrm{A}_i\, \underline{\mathrm{z}} = (\mathrm{L}_0)_i\, \underline{\mathrm{z}}. \tag{4-4}$$

Here, B_i denotes the i-th row of matrix B. We now provide explicit formulae for the entries A_{ij}.

Using $\langle P_{-1} f'(u_0) P_1 \psi_j, \psi_i\rangle = \langle f'(u_0) P_1 \psi_j, P_1 \psi_i\rangle_0$ and $P = I - Q$, we have

$$\mathrm{A}_{ij} = \langle f'(u_0)\,(\psi_j - Q_1\psi_j)\,, (\psi_i - Q_1\psi_i)\rangle_0. \tag{4-5}$$

Consider now (H^1-norm) normalized eigenfunctions φ_k of the Dirichlet Laplacian and compute the vertical projections. For J a complete set,

$$\begin{aligned}\mathrm{A}_{ij} &= \langle f'(u_0)\,(\psi_j - Q_1\psi_j)\,, (\psi_i - Q_1\psi_i)\rangle_0 \\ &= \langle f'(u_0)(\psi_j - \sum_k \langle \psi_j, \varphi_k\rangle_1 \varphi_k), \psi_i - \sum_l \langle \psi_i, \varphi_l\rangle_1 \varphi_l\rangle_0 \\ &= \langle f'(u_0)\psi_j, \psi_i\rangle_0 - \sum_l \langle f'(u_0)\psi_j, \varphi_l\rangle_0 \langle \psi_i, \varphi_l\rangle_1 - \sum_k \langle f'(u_0)\psi_i, \varphi_k\rangle_0 \langle \psi_j, \varphi_k\rangle_1 \\ &+ \sum_{k,l} \langle f'(u_0)\varphi_k, \varphi_l\rangle_0 \langle \psi_j, \varphi_k\rangle_1 \langle \psi_i, \varphi_l\rangle_1,\end{aligned}$$

PUC-Rio - Certificação Digital Nº 0812261/CA

where the sums above are taken over J. Now, if we define

$$\begin{aligned}
\mathrm{M}_{0ij} &= \langle f'(u_0)\psi_j, \psi_i\rangle_0 \\
\mathrm{U}_{ki} &= \langle \psi_i, \varphi_k\rangle_1, \quad i = 1, \cdots, n,\ k \in J \\
\mathrm{V}_{0kj} &= \langle f'(u_0)\psi_j, \varphi_k\rangle_0, \quad j = 1, \cdots, n,\ k \in J \\
\mathrm{c}_{0pq} &= \langle f'(u_0)\varphi_p, \varphi_q\rangle_0, \quad j = 1, \cdots, n,\ p, q \in J,
\end{aligned}$$

we can write the matrix L_0 in (4-4) in a more compact way as

$$\mathrm{L}_0 = K - \mathrm{M}_0 + \sum_{k\in J}\left(\mathrm{U}_k\mathrm{V}_{0k}^{\mathrm{T}} + \mathrm{V}_{0k}\mathrm{U}_k^{\mathrm{T}}\right) - \sum_{k,l\in J}\mathrm{c}_{0kl}\left(\mathrm{U}_k\mathrm{U}_l^{\mathrm{T}}\right). \tag{4-6}$$

The subscript 0 should remind us that the given object depends on a function u_0.

We now say a few words on how we compute the matrices and vectors defined above. We are using elements in $\mathcal{P}_1$, so we have to integrate constant gradients to form the stiffness matrix K and any integration scheme will do the job. To compute M_0 we use the midpoint rule in each triangle. Pointers to our function f' are used to compute the exact derivative at each midpoint. Since $\langle\psi_i, \varphi_k\rangle_1 = \lambda_k\langle\psi_i, \varphi_k\rangle_0$, to compute U_k we only need to perform an L^2-inner product (we stored the first three eigenvalues for our simple rectangular domain).

4.2.3
The Horizontal Step

Now that we know the discrete counterpart of moving in a horizontal plane, for a given function u_0, which we store as a vector $\underline{\mathrm{u}}$, we update it by

$$\underline{\mathrm{u}} := \underline{\mathrm{u}} + \underline{\mathrm{h}},$$

where the vector $\underline{\mathrm{h}}$ is the solution of

$$\mathrm{L}_0\underline{\mathrm{h}} = (P_{-1}(g - F(u_0)))\hat{}. \tag{4-7}$$

Note that by Proposition 8 we can drop the projection above and project the final answer instead. That is

$$\mathrm{L}_0\underline{\eta} = \hat{g} - \hat{F}(u_0), \quad \underline{\mathrm{h}} = P_1\underline{\eta}. \tag{4-8}$$

To compute the discrete version of the projection P_1, recall that $P_1\underline{\mathrm{u}} = \underline{\mathrm{u}} - Q_1\underline{\mathrm{u}}$, where $Q_1\underline{\mathrm{u}} = \sum_k\langle u, \varphi_k\rangle_1\underline{\varphi_k} = \sum_k \lambda_k\langle u, \varphi_k\rangle_0\underline{\varphi_k}$. We can then choose between

PUC-Rio - Certificação Digital Nº 0812261/CA

the expressions

$$P_1\underline{u} = \underline{u} - \sum_k \langle \underline{u}, K\underline{\varphi_k}\rangle \underline{\varphi_k} \quad \text{and} \quad P_1\underline{u} = \underline{u} - \sum_k \lambda_k \langle \underline{u}, \hat{\varphi_k}\rangle \underline{\varphi_k},$$

where we have used that $\langle u, v\rangle_0 = \langle M\underline{u}, \underline{v}\rangle$ and $M\underline{u} = \hat{u}$, if u, v are finite elements.

4.2.4
The Vertical Step

We now provide some detail about moving along a one-dimensional fiber through a given point u_0, i.e. the preimage of the vertical line $F(u_0)+V_{-1}$, where V_{-1} is spanned by the simple eigenvector φ_k. We simply move up vertically in the domain, adding a small multiple of φ_k and then apply our correcting algorithm, which moves horizontally until reaching back the original fiber through u_0 (cf Figure 4.1).

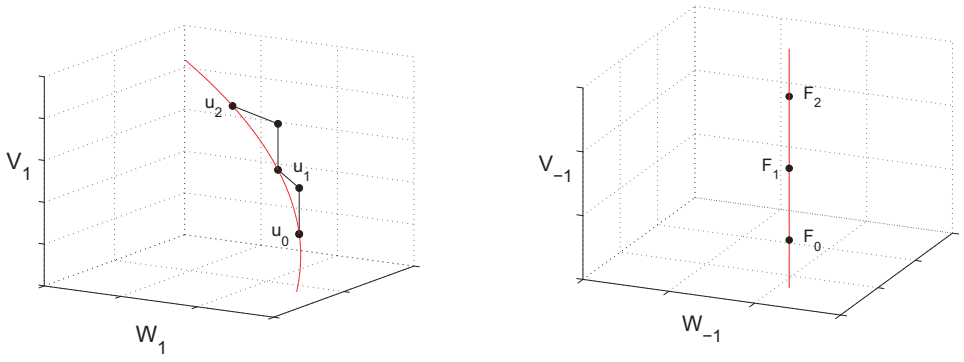

Figure 4.1: Mapping a 1-D fiber

When solving for $F(u) = g$, all me must then do is keep track of the vertical projections of our points u_j and that of their images $F(u_j)$ in the vertical subspace. The vertical projections of the points u_j are actually redundant, since the algorithm leaves that component unchanged; we perform it though in order to ensure that the computations are being properly done. All we have to do is to find a point u for which $F(u)$ and g have the same vertical projection.

PUC-Rio - Certificação Digital Nº 0812261/CA