Infinite time horizon spatially distributed optimal control problems with \texttt{pde2path} – algorithms and tutorial examples

Hannes Uecker$^1$, Hannes de Witt$^2$

$^1$ Institut für Mathematik, Universität Oldenburg, D26111 Oldenburg, hannes.uecker@uni-oldenburg.de
$^2$ Institut für Mathematik, Universität Oldenburg, D26111 Oldenburg, hannes.de.witt1@uni-oldenburg.de

March 23, 2020

Abstract

We use the continuation and bifurcation package \texttt{pde2path} to numerically analyze infinite time horizon optimal control problems for parabolic systems of PDEs. The basic idea is a two step approach to the canonical systems, derived from Pontryagin’s maximum principle. First we find branches of steady states or time–periodic states of the canonical systems, i.e., canonical steady states (CSS) respectively canonical periodic states (CPS), and then use these results to compute time-dependent canonical paths connecting to a CSS or a CPS with the so called saddle point property. This is a (high dimensional) boundary value problem in time, which we solve by a continuation algorithm in the initial states. We first explain the algorithms and then the implementation via some example problems and associated \texttt{pde2path} demo directories. The first two examples deal with the optimal management of a distributed shallow lake, and of a vegetation system, both with (spatially, and temporally) distributed controls. These examples show interesting bifurcations of so called patterned CSS, including patterned optimal steady states. As a third example we discuss optimal boundary control of a fishing problem with coastal catch. For the case of CPS–targets we first focus on an ODE toy model to explain and validate the method, and then discuss an optimal pollution mitigation PDE model.

Contents

1 Introduction 2

1.1 Pontryagin’s Maximum Principle and the canonical system 3
1.2 The general setup and the algorithms for canonical paths 5

2 Examples for CPs to CSSs 11

2.1 Optimal distributed control of the phosphorus in a shallow lake 12
  2.1.1 Canonical steady states 13
  2.1.2 Canonical paths 15
  2.1.3 Main \texttt{oclib} functions 17
  2.1.4 Skiba points 20
2.2 Optimal harvesting patterns in a vegetation model 21
2.3 Optimal coastal catch as an example of boundary control 23

3 Examples for CPs to CPSs 26

3.1 An ODE toy problem 26
  3.1.1 Preliminary analytical remarks 27
  3.1.2 \texttt{pde2path} implementation and results 28
3.2 Optimal pollution mitigation 30
1 Introduction

We consider optimal control (OC) problems for partial differential equations (PDEs) of the form

$$\partial_t v = -G_1(v,q) := D\Delta v + g_1(v,q), \text{ in } \Omega \subset \mathbb{R}^d \text{ (a bounded domain)},$$  \hfill (1a)

with initial condition $v|_{t=0} = v_0$, and suitable boundary conditions. Here $v : \Omega \times [0,\infty) \to \mathbb{R}^N$ denotes a (vector of) state variables, $q : \Omega \times [0,\infty) \to \mathbb{R}$ is a (distributed) control, $D \in \mathbb{R}^{N \times N}$ is a diffusion matrix, and $\Delta = \partial_{x_1}^2 + \ldots + \partial_{x_d}^2$ is the Laplacian. The goal is to find

$$V(v_0) := \max_{q(\cdot,\cdot)} J(v_0(\cdot),q(\cdot,\cdot))$$  \hfill (1b)

for the discounted time integral

$$J(v_0(\cdot),q(\cdot,\cdot)) := \int_0^\infty e^{-\rho t} J_{ca}(v(t),q(t)) \, dt,$$  \hfill (2)

where $J_{ca}(v(\cdot,t),q(\cdot,t)) = \frac{1}{\Omega} \int_\Omega J_c(v(x,t),q(x,t)) \, dx$ is the spatially averaged current value function, with the local current value $J_c : \mathbb{R}^{N+1} \to \mathbb{R}$ a given function, and $\rho > 0$ is the discount rate which corresponds to a long-term investment rate. The discounted time integral $J$ is typical for economic problems, where “profits now” weight more than mid or far future profits. The max (formally sup) in (1b) runs over all admissible controls $q$; this will be specified in more detail in the examples below. Additionally, we also give one example of a boundary control, where $q : \Omega_q \subset \partial \Omega \to \mathbb{R}$, and where (1) and (2) are modified accordingly. In applications, $J_c$ and $G_1$ of course often also depend on a number of parameters, which however for simplicity we do not display here.\footnote{\textcite{Uec16} can in fact be of a much more general form, but for simplicity here we stick to (1).}

In this tutorial we explain by means of four examples how to numerically study problems of type (1) with \texttt{pde2path}\footnote{see \texttt{UWR14} for background, and \texttt{Uec20} for download of the package, demo files, and various documentation and tutorials, including a quick start guide also giving installation instructions}. The examples are from \textcite{Uec16, GU17, GUU19, Uec19a}, and we mostly refer to these works and the references therein for modeling background and (bioeconomic) interpretation of the results, and for general references and comments on OC for PDE problems, here keeping these aspects to a minimum. In the first example, $q : \Omega \times [0,\infty) \to \mathbb{R}$ is the phosphate load in a model describing the phosphorus contamination of a shallow lake by a scalar PDE (1b) with homogeneous Neumann BCs $\partial_\nu v = 0$, $\nu$ the outer normal. Similarly, in the second example, $q : \Omega \times [0,\infty) \to \mathbb{R}$ is the harvesting effort in a vegetation-water system, such that (1b) is a two component reaction diffusion, again with homogeneous Neumann BCs, while in the third example $q : \partial \Omega_q \times [0,\infty) \to \mathbb{R}^2$ with $\Omega_q \subset \partial \Omega$ is a boundary control, namely the fishing effort on (part of) the shore of a lake. In these examples, so-called canonical steady states (CSSs), i.e., steady states of the so-called canonical system (see below) play an important role. The fourth example considers optimal pollution (mitigation), where the states are the emissions of some firms and the pollution stock, and the control is the pollution abatement investment. Here, canonical periodic states (CPSs) play an important role. To keep this tutorial simple, for all examples we restrict to the 1D case of $\Omega \subset \mathbb{R}$ an interval. Generalizations to domains in $\mathbb{R}^2$ are straightforward, and also straightforward to implement in \texttt{pde2path}, and for the shallow lake and the vegetation systems have also been studied in \textcite{Uec16, GU17} respectively.
In the remainder of this introduction, we first briefly review the derivation of the canonical system as a necessary first order optimality condition from (1) via Pontryagin’s maximum principle. Then we explain the basic algorithms to treat the canonical system, i.e., to first find CSSs and CPSs and then canonical paths (CPs), i.e., solutions of the canonical system which connect some given initial states to some CSS or CPS with the so–called saddle point property. These CPs yield candidates for solutions of (1). Subsequently, additional information such as concavity of $J_c$, or uniqueness of CPs, or obvious modelling considerations, often yield the optimality (or non-optimality) of the computed CPs.

In §2 we present the example problems and the pde2path implementation details for controlling to a CSS, and in §3 for controlling to a CPS. In §4 we close with a brief summary and outlook. We assume that the reader has a basic knowledge of Matlab, has installed pde2path (see also [dWDR+20]), and also has some previous experience with the software. If this is not the case, then we recommend to at least briefly look at one of the simpler problems discussed in, e.g., [RU19].

Remark 1.1. The pde2path library described here is in libs/oclib. There is also an older OC lib, namely libs/oc, which we mainly keep for downward compatibility, and the associated old demos are in ocdemos/legacy. The upgrade of oc to oclib includes the computation of CPs to CPSs, and the option to free the truncation time $T$, and we strongly recommend to switch to this setting.

1.1 Pontryagin’s Maximum Principle and the canonical system

We first consider the case of spatially distributed controls $q : \Omega \times [0, \infty) \to \mathbb{R}$, and assume homogeneous Neumann BCs for $v$, i.e., $\partial_\nu v = 0$ on $\partial \Omega$, $\nu$ the outer normal, and introduce the costates $\lambda : \Omega \times (0, \infty) \to \mathbb{R}^N$ and the (local current value) Hamiltonian

$$\mathcal{H} = \mathcal{H}(v, \lambda, q) = J_c(v, q) + \lambda^T (D \Delta v + g_1(v, q)).$$ (3)

By Pontryagin’s Maximum Principle (see Remarks 1.2 and 1.3) for the intertemporal Hamiltonian $\mathcal{H} = \int_0^\infty e^{-\rho t} \mathcal{H}(t) \, dt$ with the spatial integral

$$\mathcal{H}(t) = \int_{\Omega} \mathcal{H}(v(x, t), \lambda(x, t), q(x, t)) \, dx,$$ (4)

an optimal solution $(v, \lambda)$ (or equivalently $(v, q) : \Omega \times [0, \infty) \to \mathbb{R}^{N+1}$) has to solve the canonical system (CS)

$$\partial_t v = \partial_\lambda \mathcal{H} = D \Delta v + g_1(v, q), \quad v|_{t=0} = v_0,$$ (5a)
$$\partial_t \lambda = \rho \lambda - \partial_v \mathcal{H} = \rho \lambda + g_2(v, q) - D \Delta \lambda,$$ (5b)

where $q = \arg\max_{\tilde{q}} \mathcal{H}(v, \lambda, \tilde{q})$, which generally we assume to be obtained from solving

$$\partial_q \mathcal{H}(v, \lambda, q) = 0.$$ (5c)

Under suitable concavity conditions on $J_c$ this holds due to the absence of control constraints. The costates $\lambda$ also fulfill zero flux BCs, and in the derivation of (5) we imposed the so called inter–temporal transversality condition

$$\lim_{t \to \infty} e^{-\rho t} \int_{\Omega} \langle v(t, x), \lambda(t, x) \rangle \, dx = 0.$$ (5d)
In principle we want to solve (5) for \( t \in [0, \infty) \), but in (5b) we have initial data for only half the variables, and in (5d) we have anti–diffusion, such that (5) is ill–posed as an initial value problem. For convenience we set \( \tilde{u} \).

\[
\begin{align*}
u(t, \cdot) := \begin{pmatrix} v(t, \cdot) \\ \lambda(t, \cdot) \end{pmatrix} : \Omega \to \mathbb{R}^{2N},
\end{align*}
\]

and write (5) as

\[
\partial_t u = -G(u, \eta) := \mathcal{D}\Delta u + f(u, \eta), \quad \mathcal{D} = \begin{pmatrix} D & 0 \\ 0 & -D \end{pmatrix}, \quad f(u, \eta) = \begin{pmatrix} g_1(u, \eta) \\ g_2(u, \eta) \end{pmatrix},
\]

with BCs \( \partial_n u = 0 \), where \( \eta \in \mathbb{R}^p \) stands for parameters present, which for instance include the discount rate \( \rho \). A solution \( u \) of the canonical system (7) is called a canonical path (CP), a fixed point of (7) (which automatically fulfills (5d)) is called a canonical steady state (CSS) and a time–periodic solution of (7) is called canonical periodic states (CPS). With a slight abuse of notation we also call \((v, q)\) with \( q \) given by (5c) a canonical path.

**Remark 1.2.** For general background on OC in an ODE setting with a focus on the infinite time horizon see [GCF08] or [Taut15]. For the PDE see, [Trö10] and the references therein, or specifically [RZ99a] [RZ99b] and [AAC11], Chapter 5, for Pontryagin’s maximum principle for OC problems for semi-linear diffusive models. However, these works are in a finite time horizon setting, and often the objective function is linear in the control and there are control constraints, e.g., \( q(x, t) \in Q \) with some bounded set \( Q \). Therefore \( q \) is not obtained from the analogue of (5c), but rather takes the values from \( \partial Q \), which is often called bang–bang control. Here we do not (yet) consider (active) control or state constraints, and no terminal time, but the infinite time horizon. Our distributed OC models are motivated by [BX08] [BX10], which also discuss Pontryagin’s maximum principle in this setting.

**Remark 1.3.** The use of the Hamiltonian \( \mathcal{H} \) is the standard way of dealing with intertemporal OC problems in economics. Equivalently, the canonical system (5) is formally obtained as the first variation of the Lagrangian

\[
\mathcal{L} = \frac{1}{|\Omega|} \int_0^\infty e^{-\rho t} \left( \int_\Omega J_e(v, q) - \langle \lambda, \partial_t v + G_1(v, q) \rangle \, dx \right) \, dt,
\]

where \( G_1(v, q) = -D\Delta v - g_1 \), and where \( \lambda = (\lambda_1, \ldots, \lambda_N) \) can be identified as Lagrange multipliers to the constraint (1b), i.e., \( \partial_v + G_1(v, q) = 0 \). Using integration by parts in \( x \) with the Neumann BCs \( \partial_n v = 0 \) and \( \partial_n \lambda = 0 \) we have \( \int_\Omega \langle \lambda, D\Delta v \rangle \, dx = \int_\Omega \langle D\Delta \lambda, v \rangle \, dx \), and using integration by parts in \( t \) with transversality condition (5d) yields \( \int_0^\infty e^{-\rho t} \int_\Omega \langle \lambda, \partial_t v \rangle \, dx \, dt = \int_\Omega \langle \lambda(x, 0), v(x, 0) \rangle \, dx + \int_0^\infty e^{-\rho t} \langle \partial_t \lambda - \rho \lambda, v \rangle \, dx \, dt \). Thus, \( \mathcal{L} \) can also be written as

\[
\mathcal{L} = \frac{1}{|\Omega|} \left[ \int_\Omega \langle \lambda(x, 0), v(x, 0) \rangle \, dx 
+ \int_0^\infty e^{-\rho t} \left( \int_\Omega J_e(v, q) + \langle \partial_t \lambda + \rho \lambda + D\Delta \lambda, v \rangle + \langle \lambda, g_1(v, q) \rangle \, dx \right) \, dt \right],
\]

and (5) are the first variations of \( \mathcal{L} \) with respect to \( \lambda \) (using (9)) and \( v \) (using (9)) with \( v(0, x) = v_0(x) \). Both computations (with \( \mathcal{H} \) and \( \mathcal{L} \) are somewhat formal, and in particular the necessity of the

\footnote{The notation \( u = (v, \lambda) \) for the vector of state and costate variables is used here as \( u \) generally denotes the vector of unknowns in pde2path: in optimal control \( u \) is often used as the notation for the control, which here we denote by \( q \).}
transversality condition (5d), is subject to active research. See also [GUU19] and the references therein for a discussion of rigorous results for infinite time horizon OC problems with PDE constraints.

1.2 The general setup and the algorithms for canonical paths

To study (7) we proceed in two steps, which can be seen as a variant of the “connecting orbit method”, see, e.g., [BPS01], [Gra15, Chapter 7] and Remark 1.5: first we compute (branches of) CSSs and CPSs, and second we compute CPs connecting some initial states to some CSSs or CPSs. Thus we take a somewhat broader perspective than aiming at computing just one optimal control, given an initial condition $v_0$. Instead, we aim to give a somewhat global picture by identifying the optimal CSS/CPS and their respective domains of attraction. Here, a CSS $\hat{u}$ is called locally optimal if for each $v(0)$ in a neighborhood of $\hat{u}$ it is optimal to control the system to $\hat{u}$, and the set of all $v(0)$ such that the optimal path goes to $\hat{u}$ is called the domain of attraction. This definition extends to CPSs in a natural way.

(a) Branches of CSSs and CPSs. We compute (approximate) CSSs of (7), i.e., solutions $\hat{u}$ of

$$G(u, \eta) = 0,$$

(10)

together with the spatial BCs, by discretizing (10) via the finite element method (FEM) and then treating the discretized system as a continuation/bifurcation problem. This gives branches $s \mapsto (\hat{u}(\eta), \eta(s))$ of solutions, parameterized by a (pseudo-) arclength, which is in particular useful to possibly find several solutions $\hat{u}^{(l)}(\eta)$, $j = l, \ldots, m$ at fixed $\eta$. CPS are usually not computed directly, but via Hopf bifurcation from branches of CSS. Thus, after finding such Hopf bifurcations, we do a branch switching with an appropriate initial guess for the rescaled problem

$$\partial_t u = -T_p G(u, \eta),$$

$$u(0) = u(1),$$

(11a)

(11b)

where the period $T_p$ becomes an additional unknown, see [Uec19a].

By computing the associated $J_{ca}(\hat{v}, \hat{q})$ we can identify which of the CSSs and CPSs is optimal amongst the CSSs and CPSs. Given a CSS $\hat{u}$, for simplicity we also write $J_{ca}(\hat{u}) := J_{ca}(\hat{v}^{(l)}, \hat{q}^{(l)})$, and moreover have, by explicit evaluation of the time integral,

$$J(\hat{u}) = J_{ca}(\hat{u})/\rho.$$

(12)

For a CPS $\hat{u}$ with period length $T_p$ we evaluate the time integral

$$J(\hat{u}) = \int_0^\infty e^{-\rho t} J_{ca}(\hat{u}(t))dt = \frac{1}{1 - e^{-\rho T_p}} \int_0^{T_p} e^{-\rho T_p} J_{ca}(\hat{u}(t))dt.$$

(13)

Due to the discounting this integral may highly depend on the phase of the CPS, and thus we do not have a single objective value for a CPS, but a continuum of objective values. However, when computing CPs to a CPS it generally turns out that the values of the CPs are independent of the chosen phase of the CPS, see Remark 3.1 below.

4The $\hat{\cdot}$ notation is often used in OC for CSS, and is not related to Fourier transform in any way; we use the notation $\hat{\cdot}$ for CPS $t \mapsto \hat{u}(t)$ in an analogous sense.
(b) Canonical paths to canonical steady states. In a second step, using the results from (a), we compute CPs connecting chosen initial states to a CSS $\hat{u}$ (or a CPS $\hat{u}$, see below), and the objective values of the canonical paths. For paths to a CSS we choose a truncation time $T$ and modify (5d) to the condition that $u(T) \in W_s(\hat{u})$ and near $\hat{u}$, where $W_s(\hat{u})$ denotes the stable manifold of $\hat{u}$. In practice, we approximate $W_s(\hat{u})$ by the stable eigenspace $E_s(\hat{u})$, and thus consider the time-rescaled BVP

\[
\begin{align*}
\frac{\partial_t u}{dt} &= -TG(u), \\
v|_{t=0} &= v_0, \\
u(1) &\in E_s(\hat{u}),
\end{align*}
\]

and $\|u(1) - \hat{u}\|$ small in a suitable sense, further discussed below. If the mesh in the FEM discretization from (a) consists of $n$ nodes, then $u(t) \in \mathbb{R}^{2Nn}$, and (14a) yields a system of $2Nn$ ODEs in the form (with a slight abuse of notation)

\[
M \frac{d}{dt} u = -TG(u),
\]

while the initial and transversality conditions become

\[
\begin{align*}
v|_{t=0} &= v_0, \\
\Psi(u(1) - \hat{u}) &= 0.
\end{align*}
\]

Here, $M \in \mathbb{R}^{2Nn \times 2Nn}$ is the mass matrix of the FEM mesh, (15b) consists of $Nn$ initial conditions for the states, while the costates $\lambda$ (and hence the control $q$) are free, and $\Psi \in \mathbb{R}^{Nn \times 2Nn}$ defines the projection onto the unstable eigenspace $E_u(\hat{u})$, where due to the convention that $\partial_t u = -TG(u)$, the stable eigenspace is spanned by the (generalized) eigenvectors of $\partial_u G(u)$ to eigenvalues $\mu$ with positive real parts. Thus, to have $2Nn$ BCs altogether we need $\dim E_s(\hat{u}) = Nn$. On the other hand, we always have $\dim E_s(\hat{u}) \leq Nn$, see [GU17, Appendix A]. We define the defect

\[
d(\hat{u}) := \dim E_s(\hat{u}) - Nn
\]

and call a CSS $\hat{u}$ with $d(\hat{u}) = 0$ a CSS with the saddle–point property (SPP). At first sight it may appear that $d(\hat{u})$ depends on the spatial discretization, i.e., on the number of $n$ of nodes. However, $d(\hat{u})$ remains constant for finer and finer meshes, see [GU17, Appendix A] for further comments.

For $\hat{u} = (\hat{v}, \hat{\lambda})$ with the SPP, and $\|v_0 - \hat{v}\|$ sufficiently small, we may expect the existence of a solution $u$ of (15), which moreover can be found from a Newton loop for (15) with initial guess $u(t) \equiv \hat{u}$. Here, as a first guess for the truncation time $T$ we may use the longest decay length of the stable directions, i.e.,

\[
T = \text{Re}(\mu_2)^{-1},
\]

where $\mu_2$ is the stable eigenvalue with the smallest real part.

For larger $\|v_0 - \hat{v}\|$ a solution of (15) may not exist, or a good initial guess may be hard to find, and therefore we use a continuation process for (15). In the simplest setting, assume that for some

---

5recall that we put the truncation time $T$ into $\partial_t u = -TG(u)$ such that the end point of a CP is at $t = 1$
\( \alpha \in [0, 1] \) we have a solution \( u_\alpha \) of (15) with (15b) replaced by
\[
v(0) = \alpha v_0 + (1 - \alpha) \hat{v},
\] (18)
(e.g., for \( \alpha = 0 \) we have \( u = \hat{u} \)). We then increase \( \alpha \) by some stepsize \( \delta \alpha \) and use \( u_\alpha \) as initial guess for (15a), (15c) and (18), ultimately aiming at \( \alpha = 1 \). To ensure that \( \|u(1) - \hat{u}\| \) is small, the truncation time \( T \) may be set free if
\[
\|u(1) - \hat{u}\|_\infty \leq \varepsilon_\infty
\] (19)
is violated, and the additional boundary condition
\[
\|u(1) - \hat{u}\|_2^2 = \varepsilon^2
\] (20)
with fixed \( \varepsilon \) is added, where \( \|u\|_2 = \left( \frac{1}{n_u} \sum_{i=1}^{n_u} u_i^2 \right)^{1/2} \) is a weighted (discrete) \( L^2 \) norm, and we initialize
\[
\varepsilon = \frac{1}{10} \|u(1) - \hat{u}\|_\infty,
\] (21)
which of course is only a rough estimate and highly problem dependent, and, like all numerical parameters, can be reset by the user. See Remark 1.4 for further comments.

To discretize in time and then solve (15a), (15c) and (18) (including (20) if \( T \) is free) we use the BVP solver TOM [MS02, MT04, MST09] in a version \texttt{mtom} which accounts for the mass matrix \( M \) on the lhs of (15a). This predictor \((u_\alpha) - \text{corrector (mtom for } \alpha + \delta \alpha \text{)}\) continuation method corresponds to a “natural” parametrization of a canonical paths branch by \( \alpha \). We also give the option to use a secant predictor
\[
u^j(t) = u^{j-1}(t) + \delta \alpha \tau(t), \quad \tau(t) = \left( u^{(j-1)}(t) - u^{(j-2)}(t) / \|u^{(j-1)}(\cdot) - u^{(j-2)}(\cdot)\|_2 \right., \] (22)
where \( u^{j-2} \) and \( u^{j-1} \) are the two previous steps. However, the corrector still works at fixed \( \alpha \), in contrast to the arclength predictor–corrector described next.

It may happen that no solution of (15a), (15c) and (18) is found for \( \alpha > \alpha_0 \) for some \( \alpha_0 < 1 \), i.e., that the continuation to the chosen initial states fails. In that case, often the CP branch shows a fold in \( \alpha \), and we use a modified continuation process, letting \( \alpha \) be a free parameter and using a pseudo–arclength parametrization by \( \sigma \) in the BCs at \( t = 0 \). We set \( \alpha \) free and add BCs at continuation step \( j \),
\[
\langle s, (u(0) - u^{(j-1)}(0)) + s_\alpha (\alpha - \alpha^{(j-1)}) \rangle = \sigma,
\] (23)
with \((u^{(j-1)}(\cdot), \alpha^{j-1})\) the solution from the previous step, and \((s, s_\alpha) \in \mathbb{R}^{2n} \times \mathbb{R} \) appropriately chosen with \( \| (s, s_\alpha) \|_* = 1 \), where \( \| \cdot \|_* \) is a suitable norm in \( \mathbb{R}^{2n+1} \), which may contain different weights of \( v \) and \( v_\alpha \). For \( s = 0 \) and \( s_\alpha = 1 \) we find natural continuation with stepsize \( \delta \alpha = \sigma \) again. To get

\[\text{see also www.dm.uniba.it/~mazzia/mazzia/?page_id=433}\]
\[\text{and which is also used for finding time-periodic orbits in pde2path, see [Uec19a, Uec19b]}\]
\[\text{We also use a simple BVP solver bvphdw, which we mainly set up for testing, but which is sometimes more robust than the sophisticated methods (error estimation and mesh refinement) of mtom. In the following discussion, we write mtom for the BVP solver, but bvphdw can similarly be used.}\]
around folds we may use the secant
\[
s := \xi \left( u^{(j-1)}(0) - u^{(j-2)}(0) \right) / \left\| u^{(j-1)}(0) - u^{(j-2)}(0) \right\|_2 \quad \text{and} \quad s_\alpha = 1 - \xi
\]
with small \( \xi \), and also a secant predictor
\[
(u^j, \alpha^j)_{\text{pred}} = (u^{j-1}, \alpha^{j-1}) + \sigma \tau
\]
for \( t \mapsto u^j(t) \) with
\[
\tau = \xi \left( u^{(j-1)}(\cdot) - u^{(j-2)}(\cdot) \right) / \left\| u^{(j-1)}(\cdot) - u^{(j-2)}(\cdot) \right\|_2 \quad \text{and} \quad \tau_\alpha = 1 - \xi.
\]
This essentially follows [GCF+08, §7.2].

Finally, given \( \hat{u} \), to calculate \( \Psi \), at startup we solve the generalized adjoint eigenvalue problem
\[
\partial_u G(\hat{u})^T \Phi = \Lambda M \Phi
\]
for the eigenvalues \( \Lambda \) and (adjoint) eigenvectors \( \Phi \), which also gives the defect \( d(\hat{u}) \) by counting the negative eigenvalues in \( \Lambda \). If \( d(\hat{u}) = 0 \), then from \( \Phi \in \mathbb{C}^{2N \times 2N} \) we generate a real base of \( E_u(\hat{u}) \) which we sort into the matrix \( \Psi \in \mathbb{R}^{N \times 2N} \). Algorithm 1 summarizes our method to compute a CP to a CSS.

Algorithm 1: Continuation algorithm to compute CPs to a CSS \( \hat{u} \). Input \( \hat{u} = (\hat{v}, \hat{\lambda}) \), initial states \( v_0^\ast \), vector \( \vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m) \) of \( \alpha \) values for the 'natural' continuation with \( m \) steps, or \( (\alpha_1, \alpha_2) \) and number \( m \) of arclength steps. Optionally truncation time guess \( T \). For consecutive calls, Step 0 is omitted, and the new predictor is generated from the already computed data. Furthermore, let \( \text{arc}=0,1 \) be the switch controlling whether natural or arclength continuation is used.

0. Preparation. Compute \( \Psi \) and the defect \( d \) at the CSS. If \( d \neq 0 \), then return (not a saddle point).

Otherwise, set \( j = 0 \), and, if no initial guess for \( T \) is given, compute \( T \) from (17).

Repeat until \( \alpha = 1 \) or \( j = m \) or until convergence failure.

1. BVP solution. Solve (15) and check (19). If (19) is violated (or from the start), then augment (15) by (20), free \( T \), and solve again.

2. Next prediction (or stop). If \( \alpha = 1 \) or \( j = j_{\text{max}} \) then stop and return solution \( u \).

If no solution found:

If \( \text{arc}=0 \), then stop and return the last solution.

If \( \text{arc}=1 \) and \( \delta > \delta_{\text{min}} \) (else stop and return the last solution), then decrease \( \delta \) and go to 1 with new predictor from (24).

If solution found:

Let \( j = j + 1 \).

If \( \text{arc}=0 \), then let \( \alpha = \alpha_{j} \), let \( v_0 = \alpha v_0^\ast + (1 - \alpha)\hat{v} \), \( u_{\text{guess}} = u^{(j-1)} \) or set \( u_{\text{guess}} \) according to (24) (secant predictor), and go to 1.

If \( \text{arc}=1 \), then make a new \( (\alpha, u) \) predictor via (24), set \( v_0 = \alpha v_0^\ast + (1 - \alpha)\hat{v} \), and go to 1.

Remark 1.4. Writing (the discretized version of) (15) (and, if switched on, (20)) as
\[
H(U) = 0, \quad U = (u, T, \alpha),
\]
u is a (numerical) solution of (27) if \( \| H(U) \|_\ast \leq \text{tol} \), where the \( M_u \) component of \( H(U) \) is essentially measured in the \( \| \cdot \|_\infty \) norm (for \text{mtom} we use the relative error), and thus we also use \( \| \cdot \|_\infty \)
in (19). On the other hand, for the active condition (20) we choose the euclidean norm with derivative \( \frac{2}{n_u}(u(1) - \hat{u}) \) (as a row vector) instead of the at first sight more natural condition \( \|u(1) - \hat{u}\|_2^2 = 0 \), because we thus altogether obtain a well conditioned Jacobian \( \partial U H(\hat{U}) \) for the extended system (see (31)). For \( \varepsilon \) on the order of \( 10^{-1} \) and \( \|u(1) - \hat{u}\|_2^2 = 0 \) are essentially equivalent, but the additional flexibility via \( \varepsilon \) (and \( \varepsilon \) derived from \( \varepsilon_\infty \)) with for instance \( \varepsilon_\infty \) on the order of \( 10^{-3}\|\hat{u}\|_\infty \) turns out to be useful to obtain fast and robust results. Finally, starting with such a possibly rather large \( \varepsilon \) then also allows to decrease \( \varepsilon \) a posteriori in a few steps, see the examples in [2]. In detail, we solve

\[
H(U) = \begin{pmatrix} \Phi(U) \\ \Theta(U) \\ G(u, T) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{n_u m + k_1 + k_2},
\]

where \( n_u = 2N n_p \) is the number of spatial degrees of freedom (\( N = \)number of states, \( n_p = \)number of spatial discretization points), where \( m \) denotes the number of time steps, where the boundary conditions are written as

\[
\Phi = (v|_{t=0} - v_0) \text{, where } \Phi_2(u) = \begin{cases} \Psi(u(1) - \hat{u}) & \in \mathbb{R}^{n_u/2}, \text{ CSS, fixed } T, \\ \Psi(u(1) - \hat{u}) & \in \mathbb{R}^{n_u/2+1}, \text{ CSS, free } T, \\ ||u(1) - \hat{u}||^2 - \varepsilon^2 & \in \mathbb{R}^{n_u/2+1}, \text{ CPS, see (33)}, \end{cases}
\]

where \( \Theta \) contains the arclength boundary condition (23), if switched on, and where \( G \) is the discretization of (15). Thus, \( k_1 = 0 \) for CPs to CSSs with fixed time, and \( k_1 = 1 \) for CPs to CSSs with free time or CPs to CPSSs, and \( k_2 = 0 \) (natural continuation) or \( k_2 = 1 \) (arclength). To solve (28), given a guess \( U_0 \) from a previous step, we use Newton’s Method, i.e.,

\[
U_{j+1} = U_j - \mathcal{A}(U_j)^{-1}H(U_j),
\]

where the first line consist of \( n_u/2 \) rows, where the second line consists of \( n_u/2 + k_1 \) rows, and the \([s(1, \ldots , 1) \ 0 \ldots 0 \ s_\alpha] \)—row and the last column of \( \mathcal{A} \) are only present in the arclength setting.\(^9\) Moreover, \( M_j = h_j^{-1} M + \frac{1}{2} TG(u_j) \), \( H_j = -h_j^{-1} M + \frac{1}{2} TG(u_{j+1}) \), \( (\partial_T G)_j = \frac{1}{2}(G(u_j) + G(u_{j+1})) \), where \( h_j \) is the \( j \)-th time step, and \( u_j \) the field at time \( t_j \). Thus, for \( T \) free, \( D_u \Phi_2 \) contains the row \( 2(u(1) - \hat{u}) \), which would be ill conditioned if \( u(1) = \hat{u} \), and (20) is much more robust. Of course, \( \mathcal{A}^{-1} \) in (30) is never computed and only stands for the linear system solver used.

\(^9\)We only indicate in the first line the dimension of the 0.
(c) Canonical paths to canonical periodic states. For CPs to CPSs the basic idea is a BVP of style (14) again. However (14c) has to be adapted to the CPS case. The theoretical truncated BC is
\[ u(1) \in W_s(\hat{u}) \text{ (and } \|u(1) - \hat{u}_0\| \text{ small)}, \quad (32) \]
where \( W_s(\hat{u}) \) is the stable manifold of the CPS \( \hat{u} \), and \( \hat{u}_0 \) is some point on \( \hat{u} \). In practice, to have a boundary condition analogous to (15c), we fix an end-point \( \hat{u}_0 \) on the CPS. We then want a boundary condition of the form
\[ P(u(1) - \hat{u}_0) = 0, \quad (33) \]
with \( P \in \mathbb{R}^{(n_u/2+1) \times n_u} \) to yield \( n_u/2+1 \) boundary conditions. For a CPS, the analog to the linearization at a CSS is the monodromy matrix \( M_p \), which describes the linear effect of small deviations with respect to one period. It corresponds to the time \( T_p \) (period of the CPS) map of the variational equation
\[ \partial_t v = -\partial_u G(\hat{u}(t))v, \quad v(0) = v_0. \quad (34) \]
The eigenvalues of \( M_p \) are called Floquet multipliers, and are independent of the choice of \( \hat{u}_0 \), but the eigenvectors depend on \( \hat{u}_0 \). Since in (11a) we start with an autonomous system, we always have the (trivial) multiplier \( \gamma_1 = 1 \), which corresponds to a time shift of \( \hat{u} \), and this trivial multiplier can be used for assessing the numerics.\(^{10}\)

The monodromy matrix \( M_p \) can be computed (approximated) as the product of the linearizations of (14a) at every \( t \)-mesh point of \( \hat{u} \). However, for OC problems, in particular PDE OC problems, we often have both very large (due to the anti–diffusion in the co-states) and very small (due to diffusion in the states) multipliers\(^{11}\), and thus we need a particularly stable method for their computation. Here we use a periodic Schur decomposition, see [Kre01] and [Uec19a] and the references therein for details. This produces a set of matrices
\[ M_p = EDE^T \quad (35) \]
with an orthogonal matrix \( E \) and an upper triangular matrix \( D \) with the multipliers on the diagonal. Moreover, the adjoint monodromy matrix can be computed without extra effort, and sorting of the multipliers in \( D \) is possible, and as the first \( k \) columns of \( E \) are a basis of the span of the first \( k \) eigenvectors, we can compute projections on eigenspaces this way.

The projection \( P \) in (33) needs to provide \( n_u/2 + 1 \) boundary conditions by projecting onto the center–unstable eigenspace, associated with multipliers \( \gamma \) with \( |\gamma| \geq 1 \), where the translational eigenspace associated with the trivial multiplier \( \gamma = 1 \) is included because we want to fix the truncation point \( \hat{u}_0 \) on the CPS. As for the CSS case, the dimension of the center–stable eigenspace is at most \( Nn \) and thus this is the only case in which a canonical path can be computed. This is called saddle point property (SPP) of the CPS, see [GCF+08]. Given a CPS with the SPP we thus compute the matrix \( P \) as the projection on the center-unstable eigenspace, i.e. onto the eigenspace spanned by Floquet-multipliers \( \gamma \) with \( |\gamma| \geq 1 \).

Thus, altogether we have \( n_u/2 \) BCs (14b) for the initial states, \( n_u/2 + 1 \) BCs (33), and \( n_u \) ODEs

\(^{10}\)For instance, we give a warning if for the trivial multiplier we numerically have \( |\gamma_1 - 1| > \text{tolfloq} \), with default setting \( \text{tolfloq} = 10^{-8} \).

\(^{11}\)For instance \( |\gamma| \approx 10^{80} \) for the largest multiplier in a small scale PDE problem.
(14a) for the $n_u$ unknowns ($u_i$), and the free truncation time $T$ is the $(n_u + 1)$th unknown, i.e.,

$$\begin{align*}
\partial_t u &= -TG(u), \\
v|_{t=0} &= v_0, \\
u(1) &\in E_u(\hat{u}_0).
\end{align*}$$

(36a)\hspace{1cm} (36b)\hspace{1cm} (36c)

Moreover, as in (19) we additionally require, for a given $\varepsilon_\infty > 0$,

$$\|u(1) - \hat{u}_0\| < \varepsilon_\infty.$$

(37)

The continuation method in the initial states is the same as for CPs to CSS, i.e. we have natural parametrization with and without secant predictor, and arclength parametrization. This includes the adaptation of the truncation time $T$ to ensure (36c). Similar to (17), an estimate for $T$ can be obtained from the largest (in modulus) stable multiplier $\gamma$, which we denote by $\gamma_2$, with the trivial multiplier denoted by $\gamma_1 = 1$. In the linear regime (small deviation from $\hat{u}$) we then have

$$\|u(1) - \hat{u}_0\| \sim |\gamma_2|^T/T \|v(0) - \hat{v}_0\|.$$  

(38)

However, for small $\alpha$ we may use $T = lT_p$ as default initialization, with rather small $l$ ($l = 2, 3$) and then for larger $\alpha$ add periods of the CPS at the end of the canonical path within the continuation process if necessary, i.e., if the deviation from the CPS becomes to large. In detail, to ensure (37) (which is never used to extend (36)) we use an add hoc step additional to the discretization in time of (36) and the solution of the obtained algebraic system by Newton’s method:

After solving (36) we check (37). If (37) is violated, then we add multiples of the period $T_p$ of the CPS to the truncation time $T$, extend the last computed CP by extra copies of the CPS, and run the Newton loop again.

(39)

This appears to be a new idea, which allows to start with rather small initial $T$, and works very well in all our applications, see §3 for details and examples.

**Remark 1.5.** There are further (and more sophisticated) methods for computing connecting orbits, including (homo- and) heteroclinic orbits which also converge to some prescribed solutions as $t \to -\infty$. See, e.g., [Bey90] for a detailed analysis of the ‘standard’ projection boundary condition, [Pam01] and [BPS01] for the so-called boundary corrector method, and [DKvVK08, DKvVK09] for the special case of connecting orbits in $\mathbb{R}^3$. In particular, for connecting orbits to cycles (periodic orbits) these methods use a free truncation time $T$ together with certain phase conditions to ensure that $\|u(1) - \hat{u}\|$ is small, where $\hat{u}$ may vary on the cycle.

Here, we fix $\hat{u}$ and thus the (asymptotic) phase, and use (39) to ensure $\|u(1) - \hat{u}\| < \varepsilon$. From the application point of view, it is important to keep the defining systems for CPs as small as possible, and in particular to put the computation of the CPS and the projections at some target point $\hat{u}$ on the CPS into a preparatory step. Algorithm 2 summarizes our method to compute a CP to a CPS.

### 2 Examples for CPs to CSSs

To explain how to use pde2path to compute CSS and canonical paths we consider three example problems of type (1): The sloc (shallow lake OC) problem from [GU17], the vegoc (vegetation OC) problem from [Uec16], and the boundary fishing problem lvoc (Lotka-Volterra OC) from [GUU19].
Algorithm 2: Continuation algorithm to compute CPs to a CPS \( \hat{u} \). Input argument CPS \( \hat{u} = (\hat{v}, \hat{\lambda}) \), otherwise as for Algorithm 1. Again, for consecutive calls, Step 0 is omitted, and the new predictor is generated from the already computed data.

0. Preparation. Choose a point \( \hat{u}_0 = (\hat{v}_0, \hat{\lambda}_0) \) on the CPS and compute the projection \( P \) onto the center-unstable eigenspace of the monodromy matrix at \( \hat{u}_0 \) via (33), i.e. the eigenspace associated to Floquet-multiplier \( \gamma \) with \( |\gamma| \geq 1 \); this also yields the defect \( d \).

If \( d \neq 0 \), then return (not a saddle point CPS).

Let \( v_0 = \alpha v_0^* + (1 - \alpha)\hat{v}_0 \), and compute a guess for a canonical path to \( \hat{u}_0 \), initially \( (l \) copies of, if \( T = IT_p \) the CPS itself. Set \( j = 0 \).

Repeat until \( \alpha = 1 \) or \( j = m \) or until convergence failure.

1. BVP solution. Solve (36) for \( u \).

2. Target check. Check (37). If (37) is violated, then proceed as in (39), i.e., extend \( T \) and go to 1.

3. Next prediction (or stop). If \( \alpha = 1 \) or \( j = j_{\text{max}} \) then stop.

   If no solution found:
   - If arc=0, then stop and return the last solution.
   - If arc=1 and \( \delta > \delta_{\text{min}} \) (else stop and return the last solution), then decrease \( \delta \) and go to 1, with new predictor from (24).

   If solution found:
   - Let \( j = j + 1 \).
   - If arc=0, then let \( \alpha = \alpha_j \), let \( v_0 = \alpha v_0^* + (1 - \alpha)\hat{v}_0 \), \( u_{\text{guess}} = u^{(j-1)} \) or set \( u_{\text{guess}} \) according to (24) (secant predictor), and go to 1.
   - If arc=1, then make a new \((\alpha, u)\) predictor via (24), set \( v_0 = \alpha v_0^* + (1 - \alpha)\hat{v} \), and go to 1.

For all examples we first briefly sketch the models, and then summarize the contents of the respective demo folder and explain the most important files in some detail. For the first model we also explain the general setup how to initialize the spatial domain and discretization, the rhs, the computation of CSS, and the OC related routines of pde2path. For all models we give some results, but for details and interpretation of the results we refer to the respective papers.

### 2.1 Optimal distributed control of the phosphorus in a shallow lake

Following [BX08], in [GU17] we consider a model for phosphorus \( v = v(x, t) \) in a shallow lake, and phosphate load \( q = q(x, t) \) as a control. In 0D, i.e., in the ODE setting, this has been analyzed in detail for instance in [KW10]. Here we explain how we set up the spatial problem in pde2path. The model reads

\[
V(v_0(\cdot)) = \max_{q(\cdot)} J(v_0(\cdot), q(\cdot)), \quad J(v_0(\cdot), q(\cdot)) := \int_0^\infty e^{-\mu t} J_{ca}(v(t), q(t)) \, dt \quad (40a)
\]

where \( J_c(v, q) = \ln q - \gamma v^2 \), \( J_{ca}(v(t), q(t)) = \frac{1}{|\Omega|} \int_\Omega J_c(v(x, t), q(x, t)) \, dx \) as in (1b), and \( v \) fulfills the PDE

\[
\partial_t v(x, t) = D\Delta v(x, t) + q(x, t) - bv(x, t) + \frac{v(x, t)^2}{1 + v(x, t)^2}, \quad (40b)
\]

\[
\partial_n v(x, t)|_{\partial \Omega} = 0, \quad v(x, t)|_{t=0} = v_0(x), \quad x \in \Omega \subset \mathbb{R}^d. \quad (40c)
\]

The parameter \( b > 0 \) is the phosphorus degradation rate, and \( \gamma > 0 \) are ecological costs of the phosphorus contamination \( v \). One wants a low \( v \) for ecological reasons, but for economical reasons
Table 1: Scripts and functions in *ocdemos/sloc* (for the 1D case; some additional functions for the 2D case are also in the folder).

<table>
<thead>
<tr>
<th>Script/Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdcmds1D</td>
<td>script to compute bifurcation diagrams of CSS</td>
</tr>
<tr>
<td>cpdemo1D</td>
<td>script to compute CPs</td>
</tr>
<tr>
<td>skibademo</td>
<td>script to compute Skiba paths, see §2.1.4</td>
</tr>
<tr>
<td>slinit</td>
<td>init routine; set the <code>pde2path</code> parameters to standard values, then <em>some</em> parameters to problem specific values; initialize the domain, set an initial guess $u$, and find a first steady state by a Newton loop</td>
</tr>
<tr>
<td>oosetfemops</td>
<td>set FEM matrices (stiffness $K$ and mass $M$)</td>
</tr>
<tr>
<td>slsG; slsGjac</td>
<td>$G(u)$ resp. the Jacobian $\partial_u G(u)$ for (42)</td>
</tr>
<tr>
<td>slcon; sljcf</td>
<td>extract control from states/costates; compute the current value $J_c$</td>
</tr>
<tr>
<td>cssvalf; psol3D</td>
<td>print CSS value and characteristics; mod of psol3D to plot several solutions in one fig.</td>
</tr>
</tbody>
</table>

A high phosphate load $q$, for instance from fertilizers used by farmers. Thus, the objective function consists of the concave increasing function $\ln q$, and the concave decreasing function $-\gamma v^2$. In the demo directory `sloc` we consider the parameters $D = 0.5$, $\rho = 0.03$, $\gamma = 0.5$, $b \in (0.5, 0.8)$ (primary bif. param.).

With the co-state $\lambda$, the canonical system for (40) becomes, with $q(x,t) = -\frac{1}{\lambda(x,t)}$,

$$
\partial_t v(x,t) = q(x,t) - bv(x,t) + \frac{v(x,t)^2}{1 + v(x,t)^2} + D \Delta v(x,t),
$$

(42a)

$$
\partial_t \lambda(x,t) = 2\gamma v(x,t) + \lambda(x,t) \left(\rho + b - \frac{2v(x,t)}{(1 + v(x,t)^2)^2}\right) - D \Delta \lambda(x,t),
$$

(42b)

$$
\partial_\nu v = \partial_\nu \lambda = 0 \text{ on } \partial \Omega,
$$

(42c)

$$
v(x,t)|_{t=0} = v_0(x), \quad x \in \Omega.
$$

(42d)

### 2.1.1 Canonical steady states

To compute CSS we use a standard `pde2path` setup for (the steady version) of (42). As an overview, Table 1 lists the scripts and functions in *ocdemos/sloc*; these will be explained in more detail below, but for new users of `pde2path` we refer to [RU19] for an introduction into the basic `pde2path` data structures and setup of elliptic systems.

The `pde2path` FEM setup converts the PDE (42) into the ODE system (or algebraic system for steady states)

$$
M \frac{d}{dt} u = -G(u), \text{ where } G(u) = -K u - M f(u).
$$

(43)

In (43), $M$ is the mass matrix of the FEM, $K = \begin{pmatrix} K & 0 \\ 0 & -K \end{pmatrix}$ is the stiffness matrix, where $K$ is the 1–component stiffness matrix corresponding to the scalar (Neumann–)Laplacian, or, more precisely, $M^{-1}K \approx -\Delta$, and we put ‘everything but diffusion’ into the ‘nonlinearity’ $f$. As usual, our basic structure is a Matlab struct `p` as in problem, which has a number of fields (and subfields), e.g., `p.fuha`, `p.pdeo`, `p.u`, `p.hopf`, `p.file`, `p.plot`, `p.sw`, `p.nc` which contain for instance function handles to the right hand side (`p.fuha`), the FEM mesh (`p.pdeo`), the current solution `p.u`, data for Hopf
orbits (CPS, \(p.\text{hopf}\)), filenames/counters (\(p.\text{file}\)), plotting controls (\(p.\text{plot}\)), and switches (\(p.\text{sw}\)) and numerical constants (\(p.\text{nc}\)) used in the numerical solution, such as \(p.\text{nc}.\text{tol}\), where \(u\) is taken as a solution of \(G(u) = 0\) if \(\|G(u)\|_\infty < p.\text{nc}.\text{tol}\). However, most of these can be set to standard values by calling \(p = \text{stanparam}(p)\). In standard problems the user only has to provide:

1. The geometry of the domain \(\Omega\), and in the \texttt{OOPDE} setting used here a function \texttt{oosetfemops} used to generate the needed FEM matrices.

2. Function handles \(sG\) implementing \(G\), and, for speedup, \(sGjac\), implementing the Jacobian.

3. An initial guess for a solution \(u\) of \(G(u) = 0\), i.e., an initial guess for a CSS.

Typically, the steps 1-3 are put into an init routine, here \(p = \text{slinit}(p, lx, ly, nx, sw, ndim)\), where \(lx, ly, nx\) and \(ndim\) are parameters to describe the domain size and discretization\(^{12}\), and \(sw\) is used to set up different initial guesses, see Listing 1. For CSS computations the only additions/modifications to the standard \texttt{pde2path} setting are as follows: (the additional function handle) \(p.\text{fuha}.\text{jcf}\) should be set to the local current value objective function, here \(p.\text{fuha}.\text{jcf} = @\text{sljcf}\) (see Listing 2), and \(p.\text{fuha}.\text{outfu}\) to \texttt{ocbra}, i.e., \(p.\text{fuha}.\text{outfu} = @\text{ocbra}\). This automatically puts \(J_{\text{ca}}(u)\) at position 4 of the calculated output-branch. Finally, it is useful (for instance for plotting) to set \(p.\text{fuha}.\text{con} = @\text{slcon}\), where \(q = \text{slcon}(p, u)\) (see Listing 2.1.1) extracts the control \(q\) from the states \(v\), costates \(\lambda\) and parameters \(\eta\), all contained in the vector \(u\).\(^{13}\)

Listing 1: First 4 lines of \texttt{sloc/slinit.m}, which collects some typical initialization commands. \(p = \text{stanparam}(p)\) in line 2 sets the \texttt{pde2path} parameters, switches and numerical constants to standard values; these can always be overwritten afterwards, and some typically are. For instance, in line 3, besides setting the function handles to the rhs (necessarily problem dependent), here we overwrite the standard branch-output \(p.\text{fuha}.\text{outfu} = @\text{stanbra}\) with the OC standard output \texttt{ocbra}. The remainder of \texttt{slinit} follows the general rules of initialization in the \texttt{OOPDE} setting, explained in \cite{RU19}. We only comment that while we here no further discuss the 2D case, the same init-file is used for the 1D and 2D cases, controlled via the input argument \(ndim\), and that via the switch \(sw\) the user can choose an initial guess \(u\) near one of the two spatially homogeneous branches: \(sw=1\) leads to the so called ‘flat state clean’ (FSC) which turns into ‘flat state intermediate’ (FSI), and \(sw=2\) leads to ‘flat state muddy’ (FSM); see \cite{GU17} for this nomenclature.

Function \texttt{p=slinit(p, lx, ly, nx, sw, ndim)} % init - routine
\begin{verbatim}
p=stanparam(p); % set generic parameters to standard , if needed reset below ..
p.fuha.sG= @slsG; p.fuha.sGjac= @slsGjac; p.fuha.outfu= @ocbra; % rhs and branch
p.fuha.jcf= @sljcf; p.fuha.con= @slcon; % current -val , and fun to get k from u
\end{verbatim}
Listing 2: \texttt{oosetfemops.m}, \texttt{slcon.m} and \texttt{sljcf.m} from \texttt{sloc}. \texttt{oosetfemops} assembles and stores the needed FEM matrices, \texttt{slcon} computes the control (here very simple), and \texttt{sljcf} computes the current value \(J\) from the states/costates.

Function \texttt{p=oosetfemops(p)} % set FEM matrices (stiffness \(K\) and mass \(M\)) \begin{verbatim}
[K,M,""] = p.pdeo.fem.assema(p.pdeo.grid,1,1,1); % 'scalar' (1-component) matrices
p.mat.K = [(K,0*K);[0*K,-K]]; p.mat.M = [[M,0*M];[0*M,M]];
\end{verbatim}

Function \texttt{q=slcon(p,u)} % compute control from states/costates \begin{verbatim}
q=-1./u(p.np+1:p.nu);
\end{verbatim}

Function \texttt{jc=sljcf(p,u)} % current value \(J\) \begin{verbatim}
cp=u(p.nu+3); v=u(1:p.np); q=-1./u(p.np+1:p.nu); jc=log(q)-cp*v.^2;
\end{verbatim}
Listing 2: \texttt{oosetfemops.m}, \texttt{slcon.m} and \texttt{sljcf.m} from \texttt{sloc}. \texttt{oosetfemops} assembles and stores the needed FEM matrices, \texttt{slcon} computes the control (here very simple), and \texttt{sljcf} computes the current value \(J\) from the states/costates.

Function \texttt{r=slsG(p,u)} % rhs for SL \(v_t = D*\text{lap} \ v - 1/1-b*v+v^2/(1+v^2)\) \begin{verbatim}
1_t = -D*\text{lap} \ 1+2cp*v+1*(\rho+bp-2*v/(1+v^2)^2;
\end{verbatim}
\(^{12}\)ly (irrelevant) and \texttt{ndim} (=1) play no role in this tutorial, but we kept them in \texttt{slinit} because the same init routine is also used in 2D
\(^{13}\)We do not use \texttt{slcon} in \texttt{slsG}. However, putting this function into \texttt{p} has the advantage that for instance plotting and extracting the value of the control can easily be done by calling some convenience functions of \texttt{p2poc}.
par = u(p.nu + 1:end); r = par(1); bp = par(2); cp = par(3); D = par(4); % extract pars
v = u(1:p.np); l = u(p.np + 1:2*p.np); % extract sol components
f1 = -1./l - bp*v + v.^2./(1 + v.^2); % nonlin., first component
f2 = 2*cp*v + l.*(r + bp - 2*v./(1 + v.^2).^2); % 2nd component
f = [f1; f2];
r = D*p.mat.K*u(1:p.nu) - p.mat.M*f; % residual

Listing 3: sloc/slsG.m. The rhs of (42a,b): first we extract the parameters (line 3) and the fields \(v, \lambda\) (line 4) (with \(l\) for \(\lambda\)) from the full solution vector \(u\) which contains the pde variables \((v, \lambda)\) and the parameters. Then we compute the nonlinearity \(f\) (here and usually containing everything but diffusion), and then we compute the residual using the pre-assembled stiffness and mass matrices in \(p.mat.K\) and \(p.mat.M\), see oosetfemops.m. The Jacobian in sloc/slsGjac.m works accordingly.

At the end of slinit, we call a Newton–loop to converge to a (numerical) CSS, which is called ‘flat’, i.e., spatially homogeneous. By calling \(p = \text{cont}(p)\) we can continue such a state in some parameter. If \(p.sw.bifcheck > 0\), then pde2path detects, localizes and saves to disk bifurcation points on the branch. Afterwards, the bifurcating branches can be computed by calling swibra and \(\text{cont}\) again. These (and other) pde2path commands (continuation, branch switching, and plotting) are typically put into a script file, here bdcmds1D.m, see Listing 4 which we recommend to organize into cells. There are some modifications to the standard pde2path plotting commands, see, e.g., plot1D.m, to plot \(v\) and \(q\) simultaneously. These work as usual by overloading the respective pde2path functions by putting the adapted file in the current directory. See Fig. 1 for example results of running bdcmds1D.

\[
% FSC/FSI branch; init, then cont to find bif.points
lx = 2*pi/0.44; ly = 1; nx = 20; sw = 1; ndim = 1; p = slinit(p, lx, ly, nx, sw, ndim);
p = setfn(p, ‘f1’); p.nc.nsteps = 10; screenlayout(p); p = cont(p, 100);
% FFS branch
sw = 2; p = slinit(p, lx, ly, nx, sw, ndim); p = setfn(p, ‘f2’);
p.nc.dsmax = 0.2; p.sol.ds = 0.1; p = cont(p, 25);
% bif from f1 (set bpt* and p* and repeat as necessary)
p = swibra(‘f1’, ‘bpt1’, ‘p1’, -0.05); p.nc.dsmax = 0.3; p = cont(p, 150);
\]

Listing 4: sloc/bdcmds.m (first 8 lines), following standard pde2path principles. The remainder of bdcmds.m deals with plotting, see the source code.

2.1.2 Canonical paths

For OC problems, the computation of CSS is just the first step. The next goal is to calculate CPs from some starting state \(v(0)\) to a CSS \(\hat{u}_1\) with the SPP. For this we use the continuation algorithm isc which is essentially a wrapper for the BVP solvers mtom and bvphdw, and which for CSS as targets implements Algorithm 1. The data is again stored in a problem structure \(p\) which has a number of general options/parameters in \(p.oc\), options specific to the behavior of the BVP solvers stored in \(p.tomopt\), and solution data stored in \(p.cp\). In particular, the oclib routines re–use the data and functions (FEM data, function handles) already set up for the computation of the CSS (or CPS), and no new functions need to be set up. The convenience function ocinit sets most OC parameters to standard values and, if provided with the corresponding data, the starting states and end point of the canonical path, i.e., the target CSS \(\hat{u}_1\), or the target \(\hat{u}_0\) on a CPS \(\hat{u}\). This function is the analog of stanparam in the CSS setting. For a first call the user only has to set the parameters at the top of Table 2 the estimated truncation time and the (initial) number of mesh points. However, as usual, the user can, and sometimes has to, change some of the standard options.

After setting up the data structures (via ocinit or modifications and possibly further commands) in the struct \(p\), the computation of CPs is started by a call of \(p = \text{isc}(p, alvin, varargin)\) with default input parameters \(p\) and alvin, where alvin is a vector of continuation steps. For arclength
Figure 1: Example bifurcation diagrams and solution plots from running `bdcmds.m`. For $b < b_{\text{fold}} \approx 0.73$ there are three branches of FCSS, here called FSC (Flat State Clean, low $v$), FSI (Flat State Intermediate), and FSM (Flat State Muddy, high $v$). On FSC∪FSI there are a number of bifurcations to patterned CSS branches. See [GU17] for further discussion.

continuation, the third input may contain the number of arclength continuation steps. For consecutive arclength calls, one can also set `alvin=[]` to directly start with arclength continuation. However a secant and two values for $\alpha$ have to be given via `p.oc.usec` and `p.hist.alpha` (if natural continuation was done before, these fields are filled). See Section 2.1.3 for examples and further description of `isc`, other OC related `pde2path` functions, and the parameters in `p.oc` and `p.tomopt`. Here we continue with a brief description of CP results for the SLOC problem. The canonical path related computations are done in the command files `cpdemo*`. We first compute paths from a patterned CSS to a flat CSS and vice versa in `cpdemo1D`, see Listing 5 and Figure 2, while `cpdemo2D` does the same in 2D. The file `skibademo`, to be run after `cpdemo1D`, computes some Skiba paths, see §2.1.4.

Listing 5: Cells 1 and 2 from `sloc/cpdemo1D.m` (to be run after `bdcmds1D.m`). We compute a CP from the states of `p3/pt19` to the CSS `f1/pt13` by “natural continuation” in the initial states. In the remainder of `cpdemo1D` we illustrate arclength continuation (also in preparation of `skibademo`), and the extraction of data from the continuation history.
Figure 2: Example outputs from cpdemo1D.m. (a) shows $P, q$ on the CP from the patterned CSS $p3/pt19$ (see Fig. 1) to the ‘Clean Flat State’ FSC, and (b) shows typical associated ‘diagnostics’, namely the convergence behavior to $\hat{u}$ (in $\|\cdot\|_{\infty}$ norm), the current value, and the discounted current value along the CP. On top we plot the value $J_0$ of the ‘starting’ CSS (from which we take the states), the value $J$ of the path, and the value $J_1$ of the target CSS. (c) shows essentially the same as (b), but starting with a rather small truncation time $T$, and then adapted during the continuation in the initial states. (d) shows a case where a CP from some states (here taken from $p1/pt68$) to a target CSS (here again the FSC) does not seem to exist, or at least cannot be computed by continuation in $\alpha$, due to a fold.

2.1.3 Main oclib functions

The functions ocinit and isc are the main user interface functions for CP numerics. Essentially, after having set up $p$ as in §2.1.1 for the CSS, including $p.fuha.jcf$, the user does not need to set up any additional functions to calculate canonical paths and their values. We give the signatures and some general remarks on the arguments and behavior of ocinit and isc, with the Cells referring to Listing 5.

**p=ocinit(p, varargin)** (Cell 1). Convenience function (similar to $p=stanparam(p)$) to generate a standard problem structure $p$ where most parameters are set to standard values. These are parameters for mtom (see tom/tomset.m), and parameters for isc, see Table 2 for an overview. If varargin=$\{sd0, sp0, sd1, sp1\}$, then ocinit also sets the states of the solution stored in $sd0/sp0$ as the starting states and the solution stored in $sd1/sp1$ as the aimed CSS/CPS. Typically some of the options should be overwritten in the further setup.

**p=isc(p, alvin, varargin)** (Cell 1). $p$ is the problem structure containing the options/parameters described above in $p.oc$ and $p.tomopt$, see Tables 2 and 3 and the solution in $p.cp$. alvin is the vector of desired $\alpha$ values for the continuation, for instance $alvin=[0.25 \ 0.5]$. If varargin=$nsteps$ is given as a third input, arclength continuation is started after the last
Table 2: Data in the struct $p$ for computing CPs. Most parameters are set to standard values via $\text{ocinit}$, but some are problem specific and have to be set explicitly. Others can and some usually have to be overwritten for the specific problem. Some options are not listed here or in Table 3, which are for internal or expert use only. See $\text{ocinit}$ and $\text{isc}$ for further comments and details.

<table>
<thead>
<tr>
<th>struct/varname</th>
<th>default</th>
<th>description/comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{oc}$/varname</td>
<td></td>
<td>struct with controls for $\text{isc}$ external to the BVP solvers $\text{mtom}$ and $\text{bvphdw}$</td>
</tr>
<tr>
<td>nti</td>
<td></td>
<td>Initial number of mesh points. Highly problem dependent, so should be set by user. $\text{mtom}$ has automatic mesh refinement, so rather try a small nti, while for $\text{bvphdw}$ a somewhat larger nti should be used.</td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>first guess for truncation time - if empty it is set by $\text{isc}$</td>
</tr>
<tr>
<td>nTp</td>
<td>2</td>
<td>for setting $T=nTp\cdot T_p$, as a guess for $T$ for CPs to a CPS, $T_p$=period of CPS</td>
</tr>
<tr>
<td>freeT</td>
<td>0</td>
<td>if 1, then truncation time is set free for CPs to CSS, and (20) is included in the BVP, with $\varepsilon$ in $\text{oc.tadev2}$</td>
</tr>
<tr>
<td>tv</td>
<td></td>
<td>initial t-mesh if not empty, otherwise generated by $\text{isc}$</td>
</tr>
<tr>
<td>retsw</td>
<td>0</td>
<td>return-switch for $\text{isc}$: 0: only final soln, 1: solutions for all $\alpha$</td>
</tr>
<tr>
<td>rhoi</td>
<td>1</td>
<td>$\text{pde2path}$ index of the discount rate $\rho$</td>
</tr>
<tr>
<td>tadevs</td>
<td>inf</td>
<td>target error in sup-norm, i.e., $\varepsilon_\infty$ in [19]. Can be set initially, but we recommend a first step without $T$ adaptation (small $\alpha$), i.e., also freeT=0.</td>
</tr>
<tr>
<td>tadev2</td>
<td>*</td>
<td>target error in euclidean norm, i.e., $\varepsilon$ in (20), initialized by (21) once (19) is violated. Can be reset later to decrease the target deviation.</td>
</tr>
<tr>
<td>mtom</td>
<td>1</td>
<td>switch between $\text{mtom}$ and $\text{bvphdw}$ (in-house bvp solver)</td>
</tr>
<tr>
<td>sig</td>
<td>0.1</td>
<td>stepsize for arclength continuation</td>
</tr>
<tr>
<td>signin/sigmax</td>
<td>1e-2,10</td>
<td>minimal and maximal stepsizes for arclength continuation</td>
</tr>
<tr>
<td>fn</td>
<td></td>
<td>file-names, for instance for the initial and target states of the CP</td>
</tr>
<tr>
<td>u0</td>
<td></td>
<td>initial states; can be provided by file (see examples), or be set after $\text{ocinit}$</td>
</tr>
<tr>
<td>s1</td>
<td></td>
<td>classical $\text{pde2path}$ data struct which will typically contain the data for the target $\hat{u}$ (CSS in s1.u, or CPS in s1.hopf)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\text{tomopt}$/varname</th>
<th>default</th>
<th>description/comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>lu</td>
<td>0</td>
<td>if 0, then use \ (usually faster) instead of $LU$ decomposition in $\text{mtom}$</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>Standard $\text{mtom}$-parameters can be given here, e.g. Itlimax, Itnlimax and Nmax for maximal number of linear and nonlinear iterations and maximal number of mesh points. See $\text{mtom}$ documentation.</td>
</tr>
<tr>
<td>tol, maxIt</td>
<td>1e-8,10</td>
<td>tolerance (in $|\cdot|_\infty$ norm) and max nr of iterations in $\text{bvphdw}$.</td>
</tr>
</tbody>
</table>

value of $\text{alvin}$ with $\text{nsteps}$ steps, or until $\alpha = 1$ is reached. If previous calls of $\text{isc}$ are present, then we can directly start arclength continuation by calling $\text{isc}$ with empty $\text{alvin}$. After the first call of $\text{isc}$ some additional fields are set in $\text{p.oc}$, containing, e.g., the current secant, the last starting point in the continuation, and, if desired via $\text{p.oc.retsw}=1$, the solutions at different continuation steps, see Table 3.

**Remark 2.1.** Concerning the original TOM options we typically run $\text{isc}$ with the fastest monitor and order options, i.e., $\text{tomopt}\cdot\text{Monitor}=3; \text{tomopt}\cdot\text{order}=2$. Once continuation is successful (or also if it fails at some $\alpha$), we can always postprocess by calling $\text{mtom}$ again with a higher order, stronger error requirements, and different monitor options. See the original TOM documentation. The most convenient way to do so is to call $\text{isc}$ with $\text{alvin}=1$ again after resetting TOM-related options.

There are a number of additional functions for internal use, and some convenience functions, which we briefly review as follows:

$[\Psi, \text{mu, d, t}]=\text{getPsi(s1)}$. For CPs to CSSs only: compute $\Psi$, the eigenvalues $\text{mu}$, the defect $d$, and a suggestion for $T$. This becomes expensive with large $2nN$ (number of spatial DoF).
Table 3: Additional fields in \texttt{p}, typically set/maintained/updated by \texttt{isc} (\texttt{p.hist} only maintained/updated if \texttt{oc.retsw}=1).

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{cp.u}</td>
<td>solution (CP) generated by \texttt{isc}; used as initial guess for next continuation step, if already set by previous call to \texttt{isc} or if set externally.</td>
</tr>
<tr>
<td>\texttt{cp.t}</td>
<td>time mesh generated by \texttt{isc} (or set externally)</td>
</tr>
<tr>
<td>\texttt{cp.par}</td>
<td>solution parameters, i.e., truncation time (T) in \texttt{par}(1), current (\alpha) in \texttt{par}(2) (for arclength)</td>
</tr>
<tr>
<td>\texttt{hist.alpha}</td>
<td>vector of the (\alpha) values in the continuation</td>
</tr>
<tr>
<td>\texttt{hist.vv}</td>
<td>vector of the objective values of the canonical path for the (\alpha) stored in \texttt{hist.alpha}</td>
</tr>
<tr>
<td>\texttt{hist.u}</td>
<td>CPs at continuation steps stored in \texttt{hist.alpha}</td>
</tr>
<tr>
<td>\texttt{hist.t}</td>
<td>time-meshes (normalized to ([0,1])) of continuation steps</td>
</tr>
<tr>
<td>\texttt{hist.par}</td>
<td>parameter values ((T) and (\alpha)) of the continuation steps</td>
</tr>
</tbody>
</table>

\([\text{Fu1,Fu2,d}]=\text{floqpsmatadj}(\text{p.opt.s1}).\) For CPs to CPSs only: computes (by periodic Schur decomposition) the projection on the center-unstable eigenspace in \texttt{Fu2} and the Floquet-multiplicators in \texttt{d}. Expensive, but has to be done only once for each CPS.

\([\text{sol,info}]=\text{mtom}(\text{ODE,BC,solinit,opt,varargin}).\) Modification of TOM, which allows for \(M\) in \((15a).\) Extra arguments \(M\) and \(lu,vsw\) in \texttt{opt}. If \texttt{opt.lu}=0, then \(\bslash\) is used for solving linear systems instead of an LU–decomposition, which becomes too slow when \(2nN \times m\) becomes too large. See the TOM documentation for all other arguments included in \texttt{opt}, and note that the modifications in \texttt{mtom} can be identified by searching “HU” in \texttt{mtom.m}. Of course \texttt{mtom} (as any other function) can also be called directly, which for instance can be useful to postprocess the output of some continuation by changing parameters by hand.

\texttt{sol}=\text{bvphdw}(\text{ODE,BC,solinit,tmopopt,opt}).\) A simple Newton solver for CPs, which was mainly used for testing but is sometimes more robust than the sophisticated methods (error estimation and mesh refinement) of \texttt{mtom}.

\(f=\text{mrhs}(t,u,q,opt);\ J=\text{fjac}(t,u,opt)\); and \(f=\text{mrhse}(t,u,q,opt);\ J=\text{fjace}(t,u,opt).\) The rhs and its Jacobian to be called within \texttt{mtom} resp. \texttt{bvphdw} (see the respective wrapper files). These are wrappers which calculate \(f\) and \(J\) by calling the resp. functions in the \texttt{pde2path–struct \texttt{p.opt.s1}}, which were already set up and used to calculate the CSS/CPS. Similar remarks apply to \texttt{mrhs} and \texttt{fjac} for the arclength continuation.

\(bc=\text{cbcf}(ya,yb,opt);[ja,jb]=\text{cbcjac}(ya,yb,opt)\); \(bc=\text{cbcfe}(ya,yb);[ja,jb]=\text{cbcjace}(ya,yb).\) The boundary conditions (in time) for \((15)\) resp. \((33)\) and the associated Jacobians again with wrappers to be used for \texttt{mtom} and \texttt{bvphdw} at once. The \texttt{*e} (as in \texttt{extended}) versions are for arclength continuation again.

\([jval,jcav,jcavd]=\text{jcaiT(s1,cp,rho)}\) and \(\text{djca=disjcaT(s1,cp,rho)}.\) Computes the value

\[ jval = J(u) = \int_0^T e^{-\rho t} J_ca(v(t,\cdot),q(t,\cdot)) \, dt \quad (44) \]

of the solution \(u\) in \texttt{cp} (with \(J_c\) taken from \texttt{s1.fuha.jcf}), and also returns \(J_{ca}\) and \(e^{-\rho t} J_{ca}\) along the CP for easy plotting, cf. Fig. 2(b).

\([\text{di,d2}]=\text{tadev(p)}\). Deviations (in sup norm and euclidean norm) of endpoint of CP from target, see \((19)\) and \((20)\).

**Remark 2.2.** There are also some plotting function in \texttt{oclib}, which however should be seen as templates for plotting of canonical paths, including diagnostic plots to check the convergence behavior of the canonical path as \(t \to T\), cf. (d),(f) in Fig. 2. The function \texttt{s1solplot(p,view)} in the \texttt{sloc} demo directory can serve as a template how to set up such plots.
In ODE OC applications, if there are several locally stable CSS, then often an important issue is to identify their domains of attractions. These are separated by so called threshold or Skiba–points (if $N = 1$) or Skiba–manifolds (if $N > 1$), see [Ski78] and [GCF+08, Chapter 5]. Roughly speaking, these are initial states from which there are several optimal paths with the same value but leading to different CSS. Here we give an example for the SLOC model how to compute a patterned Skiba point between FSC and FSM.

In Cell 3 of cpdemo1D.m we attempt to find a path from $v_{PS}$ given by $p1/pt68$ to $(v,q)_{FSC}$ given by FSC/pt13; this fails due to the fold in $\alpha$. However, for given $\alpha$ we can also try to find a path from the initial state $v_0(0) := \alpha v_{PS} + (1 - \alpha)v_{FSC}$ to the FSM, and compare to the path to the FSC. For this, we can use the problem structure $p$ computed in Cell 3. The initial states for the $k$'th $\alpha$ value of $p.hist.alpha$ are, due to $p.opt.retsw=1$, stored in $p.hist.u{k}{:,1}$ i.e. as the starting point of the canonical path associated to the given $\alpha$ value.

In skibademo.m (Listing[8], which is a rather elaborate application of the OC facilities of pde2path, and can be skipped on first reading) we find paths from these initial states from cpdemo1D.m to the other flat steady state with the SPP, namely FSM, and compare the values with the values of the paths to FSC, stored in $p.hist.vv$. See Fig. 3 for illustration.

(a) A Skiba point at $\alpha = 0.454$ (b) Paths to FSC (blue) and FSM (red)

Figure 3: Example outputs from skibademo.m. For the ($\alpha$–dependent) initial states from the path from $p1/pt68$ to the FSC from cpdemo1D (blue curve in (a)), we compute the CPs to the FSM, yielding the red curve in (a). Near $\alpha = 0.43$ the two CPs have the same values, which makes these initial states a so–called Skiba candidate. (b) shows the two associated CPs of equal value.
Listing 6: sloc/skibademo.m. Using the data stored in p.hist.* in Cell 3 of cpdemo1D.m (for b = 0.65), we compute canonical paths from the initial states to the FSM at b = 0.65, and compare the objective values with those stored in p.hist.vv (for the path to the FSC). If both are sufficiently close, then we have a good approximation of a Skiba point. Thus, in line 6 we start a loop over the α values generated in cpdemo1D.m; in line 8 we put the associated initial states into p2.opt.s0, i.e. overwrite the loaded point, and then (line 10) find the canonical path to the FSM. In line 14 we check if we found a Skiba point approximation, in which case we plot both paths. Also the second cell then deals with plotting.

Remark 2.3. The directory slocdemo also contains the script files bdcmds2D.m and cpdemo2D.m, used to compute CSS and canonical paths for (42) over the 2D domain Ω = (−L, L) × (−L^2, L^2) (based on exactly the same init file slinit.m), and some modified plotting functions plotsolf.m and plotsolfu.m, see, e.g., [GU17, Fig. 4,5] for some 2D results.

2.2 Optimal harvesting patterns in a vegetation model

Our second example, from [Uec16], concerns the optimal control of a reaction diffusion system used to model harvesting (or grazing by herbivores) in a system for biomass (vegetation) v and soil water w, following [BX10]. Denoting the harvesting (grazing) effort as the control by E, we consider

\[
V(v_0, w_0) = \max_{E(\cdot)} J(v_0, w_0, E), \quad (45a)
\]

\[
\partial_t v = d_1 \Delta v + [g w p^\eta - d(1 + \delta v)]v - H, \quad (45b)
\]

\[
\partial_t w = d_2 \Delta w + R(\beta + \xi v) - (r_u v + r_w)w, \quad (45c)
\]

with harvest \( H = v^\alpha E^{1-\alpha} \), and current value objective function \( J_c = J_c(v, E) = pH - cE \), which thus depends on the price \( p \), the costs \( c \) for harvesting/grazing, and \( v, E \) in a classical Cobb–Douglas form with elasticity parameter \( 0 < \alpha < 1 \). Furthermore, we have the boundary conditions and initial conditions

\[
\partial_\nu v = \partial_\nu w = 0 \quad \text{on} \quad \partial \Omega, \quad (v, w)|_{t=0} = (v_0, w_0). \quad (45d)
\]

Again we want to maximize the discounted profit

\[
J = \int_0^\infty e^{-\rho t} J_c(a(v, E)) \, dt. \quad (46)
\]

For the modeling, and the meaning and values of the parameters \( (g, \eta, d, \delta, \beta, \xi, r_u, r_w, d_{1,2}) \) we refer to [BX10, Uec16], and here only remark that the model aims at a realistic description of certain semi–arid systems, that, e.g., the discount rate \( \rho = 0.03 \) is in the pertinent economic regime, and that, like in most studies of semi–arid systems, we take the rainfall \( R \) as the main bifurcation parameter.

Denoting the co-states by \( (\lambda, \mu) \) we obtain the canonical system

\[
\partial_t v = d_1 \Delta v + [g w p^\eta - d(1 + \delta v)]v - H, \quad (47a)
\]

\[
\partial_t w = d_2 \Delta w + R(\beta + \xi v) - (r_u v + r_w)w, \quad (47b)
\]

\[
\partial_t \lambda = \rho \lambda - p \alpha v^{\alpha-1} E^{1-\alpha} - \lambda[g(\eta + 1)w v^\eta - 2d\delta v - d - \alpha v^{\alpha-1} E^{1-\alpha}] - \mu(R \xi - r_u)w - d_1 \Delta \lambda, \quad (47c)
\]
\[ \partial_t \mu = \rho \mu - \lambda g v^{\eta+1} + \mu (r_u v + r_w) - d_2 \Delta \mu, \]

(47d)

where \( E \left( \frac{c}{(p - \lambda)(1 - \alpha)} \right)^{-1/\alpha} v. \)

The system (47) has a similar structure as (42), with the immediate difference that (47) has four components and many parameters, and thus looks somewhat complicated. However, it is still convenient to implement in pde2path, and leads to many patterned optimal steady states, see [Uec16] for further discussion. Thus, besides documenting the implementation of (47) underlying the results in [Uec16], our aim here is to illustrate that also rather complicated systems can be implemented and studied in the pde2path OC setting in a simple way. Writing (47) as \( \partial_t u = -G(u), u = (v, w, \lambda, \mu), \) we basically need to set up the domain, \( G \) and the BCs, and the objective function. Table 4 lists and comments on the scripts and functions in ocdemos/vegoc. The implementation of (47) follows the general pde2path settings with the OC related modifications already explained in §2.1, and thus we only give a few remarks in the Listing captions.

Table 4: Scripts (1D) and functions in ocdemos/vegoc: oosetfemops and veginit (which also contains the parameter values) as usual; the bottom part contains 'helper' functions for plotting.

<table>
<thead>
<tr>
<th>script/function</th>
<th>purpose, remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdcmds, cpcmds</td>
<td>scripts to compute CSS and canonical paths</td>
</tr>
<tr>
<td>efu, vegjcf</td>
<td>functions to compute ([E,H]) from (u), and the current value (J_c) (by calling efu)</td>
</tr>
<tr>
<td>vegsolplot, vegdiagn</td>
<td>functions to plot CPs, and compute and plot diagnostics for CPs</td>
</tr>
<tr>
<td>vegcm.asc, watcm.asc</td>
<td>colormaps for CP plots (and state plots in 2D)</td>
</tr>
</tbody>
</table>

Listing 7: vegoc/efu.m and vegsG. efu computes the harvesting effort (control) \( E \), the harvest \( h \) and the current value \( J \), hence also called in vegoc/vegjcf.m. In vegsG we first extract the parameters and the solution components, and compute \( H \) (from efu). Then we implement the 'nonlinearity' in a straightforward way, and compute the residual \( G \) using the preassembled stiffness and mass matrix (see oosetfemops).

Figure 4 shows a basic bifurcation diagram of CSS in 1D with \( \Omega = (-L, L) \), \( L = 5 \), from the script file bdcmds.m, which follows the same principles as the one for the SLOC demo. Again we start with a spatially flat (i.e., homogeneous) canonical steady FSS (black), on which we find a...
number of Turing-like bifurcations. The blue branch in (a) represents the primary bifurcation of PCSS (patterned canonical steady states), which for certain $R$ have the SPP, and, moreover, turn out to be POSS (patterned optimal steady states). See [Uec16] for more details, including a comparison with the uncontrolled case of so called “private optimization”, and 2D results for $\Omega = (-L, L) \times (-\sqrt{3}L/2, \sqrt{3}L/2)$ yielding various POSS, including hexagonal patterns.

The script files cpcmds.m for CPs, and skibacmds.m for a Skiba point between the flat optimal steady state $FSS/pt13$ and the POSS $p1/pt34$, again follow the same principles as in the the SLOC demo. See Figure 5 for an example output. We use customized colormaps for vegetation (green) and water (blue), which are provided as vegcm.asc, watcm.asc and whitecm.asc, respectively.

![Figure 4](image)

**Figure 4:** Example outputs of bdcmds.m. (a),(b) bifurcation diagrams of CSS in 1D; (c) example solutions.

2.3 Optimal coastal catch as an example of boundary control

Our third example, taken from [GUU19], considers the optimization of the discounted fishing profit

$$J = \int_0^\infty e^{-\rho t} J_c(v(t), q(t)) dt, \quad J_c(v, q) = \sum_{j=1}^2 p_j h_j(v_j, q_j) - c_j q_j.$$ 

Here $v = (v_1, v_2)$ are the populations...
of two fish species ($v_1 =$prey, $v_2 =$predator) in a (1D) lake or ocean $\Omega = (0, l_x)$, $q = (q_1, q_2)$ are the fishing (harvesting) efforts (controls) of $v_1$ and $v_2$, respectively, at the shore, and $p_{1,2}$ and $c_{1,2}$ are the prices for the fishes and the costs for fishing, respectively, and we again choose a Cobb–Douglas form for the harvests $h_j(v_j, q_j) = v_j^{\alpha_j}q_j^{1-\alpha_j}$, with parameters $\alpha_j \in (0, 1)$. We assume that the fish populations evolve according to a standard Lotka-Volterra model, namely

$$
\begin{align*}
\frac{\partial t}{\partial t} v_1 &= d_1 \Delta v_1 + (1 - \beta v_1 - v_2)v_1, \\
\frac{\partial t}{\partial t} v_2 &= d_2 \Delta v_2 + (v_1 - 1)v_2,
\end{align*}
$$

or $\frac{\partial v}{\partial t} = -G_1(v) = D \Delta v + f(v)$, (48)

in $\Omega$, with $D = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}$ and growth function $f(v) = \begin{pmatrix} (1 - \beta v_1 - v_2)v_1 \\ (v_1 - 1)v_2 \end{pmatrix}$, with parameter $\beta > 0$.

The controls $q_{1,2}$ (fishing efforts for species $v_{1,2}$, respectively) occur in the BCs for (48), i.e., we assume the Robin BCs

$$
\begin{align*}
d_j \partial_n v_j &= -g_j := -\gamma_j h_j, & j = 1, 2, \text{ at } x = 0,
\end{align*}
$$

and zero flux BCs $\partial_n v_j = 0$ at $x = l_x$. Thus, in contrast to the sloc and vegoc examples we no longer have a spatially distributed control, but a (two component) boundary control.

Introducing the co-states $\lambda_{1,2} : \Omega \to \mathbb{R}$, Pontryagin’s maximum principle yields the evolution and the BCs of the co-states (combining with (48), to have it all together)

$$
\begin{align*}
\frac{\partial t}{\partial t} v &= D \Delta v + f(v), \\
\frac{\partial t}{\partial t} \lambda &= \rho \lambda - D \Delta \lambda - (\partial_v f(v))^T \lambda \\
D \partial_n v + g &= 0, \\
D \partial_n \lambda + \partial_v g(v)\lambda - \partial_v J_c &= 0,
\end{align*}
$$

in $\Omega = (0, l_x)$, (50a)

on the left boundary $x = 0$, (50b)

on the right boundary $x = l_x$, (50c)

and

$$
q_j = \left(\frac{1 - \alpha_j}{c_j} - \gamma_j \lambda_j\right)^{1/\alpha_j} v_j, \text{ evaluated at the left boundary } x = 0, \quad j = 1, 2.
$$

(50d)

See [GUU19] for details on the derivation of (50).

Thus, we again have a four–component reaction diffusion system (50a-c) for the states $v$ and the costates $\lambda$, but now the controls live on the boundary at $x = 0$, leading to nonlinear flux boundary conditions. Also, from the modeling point of view the pertinent questions for (50), are slightly different than for (47), since for (50) we are not so much interested in bifurcations and pattern formation (which do not occur for the parameters chosen below), but rather in the dependence of the (unique) CSS on the parameters, and mostly in the canonical paths leading to these CSS.

In any case, we now focus on how to put the nonlinear BCs into pde2path. Table 5 lists the scripts and functions in ocdemos/lvoc, and some further comments are given in the listing captions below.
Table 5: Scripts and functions in ocdemos/lvoc; the first four follow closely the same pde2path principles as the respective functions in sloc and vegoc. lvG is a rather non–generic implementation of the rhs of (50), and in particular strongly uses the 1D nature of the problem. The others are naturally also problem specific, and in particular the last three overload some pde2path standard functions.

<table>
<thead>
<tr>
<th>script/function</th>
<th>purpose, remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdcmds,cpcmds</td>
<td>scripts to compute CSS and CPs</td>
</tr>
<tr>
<td>lvinit,oosetfemops</td>
<td>init routine, and setting of FEM matrices</td>
</tr>
<tr>
<td>lvG</td>
<td>the rhs for (50a), also explicitly implementing the BC (50b), while (50c) are naturally fulfilled with K the Neumann Laplacian.</td>
</tr>
<tr>
<td>lvbra</td>
<td>branch-output, here substantially modified from stanocbra, i.e., writing specific data like profits/harvest/controls per fish-species on the branch</td>
</tr>
<tr>
<td>hfu</td>
<td>returns harvest ( h \in \mathbb{R}^2 ), and derivatives (needed for lvG) ( \partial_v h \in \mathbb{R}^{2 \times 2} ), ( \partial_k h \in \mathbb{R}^{2 \times 2} ); straightforward implementation.</td>
</tr>
<tr>
<td>lvjcf</td>
<td>returns ( J_c ), as required in the standard pde2path setup for OC problems, but also the individual profits ( J_{1,2} ) per fish-species.</td>
</tr>
<tr>
<td>jca</td>
<td>( J_c ) average, overloaded here since ( J_c ) is on boundary, hence the default averaging by ( 1/</td>
</tr>
<tr>
<td>lvdiagn</td>
<td>diagnostics functions for canonical paths</td>
</tr>
<tr>
<td>stanpdeo1D&lt;pde</td>
<td>classdef, overloads the pde2path classdef stanpdeo1D in order to have ( \Omega = (0, l_x) ) instead of ( \Omega = (-l_x, l_x) ) (standard).</td>
</tr>
<tr>
<td>plotsol, psol3D</td>
<td>some more overloads of pde2path standard functions for plotting.</td>
</tr>
</tbody>
</table>

Figure 6 gives some example plots from running bdcmds.m. This script is rather lengthy, with the main point here to illustrate how to plot various data of interest by first putting it on the branch (here via lvbra.m) and then choosing the pertinent component for plotting. The script cpcmds.m for computing canonical paths follows the same outline as those for the sloc and vegoc examples. Figure 7 shows a canonical path to the CSS at \( c = (0.1, 0.1) \), starting from the homogeneous fixed point \( V^* = (1 - \beta) \) of (48). As already said, we refer to [GUU19] for discussion of the results.

(a) \( J, v \) and \( q \) at \( x = 0 \); continuation in \( c_1 \)

(b) CSS at \( c = (0.1, 0.1) \)

![Figure 6](image)

function \( r = \text{lvG}(p,u) \) % rhs for lvoc.

\[
\begin{align*}
\text{n} &= \text{p.np}; \text{par} = \text{u}(\text{p.nu}+1:\text{end}); \beta = \text{par}(1); \text{d1} = \text{par}(4); \text{d2} = \text{par}(5); \\
\text{g1} &= \text{par}(6); \text{g2} = \text{par}(7); \rho = \text{par}(8); \text{p1} = \text{par}(11); \text{p2} = \text{par}(12); \% \text{extract pars} \\
\text{v1} &= \text{u}(1:\text{n}); \text{v2} = \text{u}(\text{n}+1:2*\text{n}); \text{l1} = \text{u}(2*\text{n}+1:3*\text{n}); \text{l2} = \text{u}(3*\text{n}+1:4*\text{n}); \% \text{extract fields} \\
\text{f1} &= \text{v1}.*(1-\beta*\text{v1}-\text{v2}); \text{f2} = (\text{v1}-1).*\text{v2}; \% \text{bulk nonlin.} \\
\text{a11} &= 1-2*\beta*\text{v1}-\text{v2}; \text{a12} = -\text{v1}; \text{a21} = \text{v2}; \text{a22} = \text{v1}-1; \\
\text{f3} &= \text{rho}.*\text{l1}-\text{a11}.*\text{l1}-\text{a21}.*\text{l2}; \text{f4} = \text{rho}.*\text{l2}-\text{a12}.*\text{l1}-\text{a22}.*\text{l2}; \\
\text{f} &= [\text{f1}; \text{f2}; \text{f3}; \text{f4}]; \text{F} = \text{p.mat}.*\text{f}; \% \text{bulk finished, no go to BCs}
\end{align*}
\]
Figure 7: A CP starting from the spatially homogeneous steady state $V^*$ of (48) to the CSS at $(c_1,c_2) = (0.1,0.1)$. Note the logarithmic scales in the time-series of the values at the left boundary.

Listing 8: lvoc/lvsG.m. As usual we first extract the relevant parameters and fields from $u$, and compute the 'bulk' nonlinearity $f$ (i.e., the nonlinearity in $\Omega$). Then, to compute the BCs we first extract values of the co-states on the boundary, and the associated harvests and their derivatives. These are needed to compute the BCs $g_j$, $j=1,...,4$ (lines 10,11), which can then directly be added to the rhs on the boundary (lines 13,14).

Here we strongly use the 1D setup, i.e., that the left boundary value of component $j$ is at $u((j-1)*np+1)$, where $np$ is the number of spatial discretization points. In line 16 we then assemble the system $K$. Compared to the setup in vegoc/vegsG this has the advantage that the diffusion constants $d_1,d_2$ can be used like any other parameter at this point. The actual computation of the residual then works as usual.

3 Examples for CPs to CPSs

The basic idea for the computation of CPSs and of CPs to CPSs is similar to that for the computation of CSSs and CPs to CSSs. We first search for CPSs, usually via Hopf bifurcations from CSSs, and then aim to compute CPs $u$ to such CPSs with the SPP, again using the main pde2path OC user interface isc, which now implements Algorithm 2. To illustrate the setup we first discuss an ODE toy model. Subsequently we come to a PDE model for pollution mitigation.

3.1 An ODE toy problem

We start with the ODE toy model

\[ \dot{x}_1 = \rho \left( -x_1 - \frac{\theta x_2}{\rho} + x_1 y_1 r^2 \right), \quad \dot{x}_2 = \rho \left( -x_2 + \frac{\theta x_1}{\rho} + x_2 y_1 r^2 \right), \]  
\[ \dot{y}_1 = \omega y_2, \quad \dot{y}_2 = \omega \sin(2\pi y_1), \]  

with parameters $\rho, \omega, \theta > 0$ and $r = \sqrt{x_1^2 + x_2^2}$. Although (51) is not derived as a canonical system for an OC problem, we interpret $x = (x_1,x_2)$ as states and $y = (y_1,y_2)$ as costates and call a time
Concerning our points of interest, the model (51) can almost completely be treated analytically and thus can be used to test our numerical methods. For fixed $y_1 > 0$, the nonlinear system (51a) has the unstable periodic orbit $r = 1/\sqrt{y_1}$ of period $2\pi/\theta$, and (51a) is coupled to (or driven by) the nonlinear pendulum (51b). In detail, by polar coordinates in $(x_1, x_2)$, (51) transforms to

$$\dot{r} = \rho(-r + y_1 r^3), \quad \dot{\phi} = \theta,$$

$$\dot{y}_1 = \omega y_2, \quad \dot{y}_2 = \omega \sin(2\pi y_1),$$

with $\phi = \text{arg}(x_1, x_2)$ and $r = \sqrt{x_1^2 + x_2^2}$, with the phase portraits of the $r$ ODE (for fixed $y_1$) and the $y$ system sketched in Fig. 8(a, b). Thus, to find a CPS we look for CSS of the reduced system

$$\dot{r} = \rho(-r + y_1 r^3),$$

$$\dot{y}_1 = \omega y_2, \quad \dot{y}_2 = \omega \sin(2\pi y_1).$$

The costates are independent of the states and the $\dot{y}$ system has the first integral $E(y_1, y_2) = \frac{1}{2} y_2^2 + \frac{\omega^2}{2\pi} \cos(2\pi y_1)$, i.e., solutions of the system lie on contour lines of $E$, see Fig. 8(a). We choose $y_1 \in \mathbb{N}$, $y_2 = 0$ and $r = \frac{1}{\sqrt{y_1}}$, which yields a $2\pi$ periodic solution in the full system. We fix $(\rho, \theta) = (1, 1)$ and one of these CPS, namely

$$\hat{u}(t) = (r(t), \varphi(t), y_1(t), y_2(t)) = (1, t, 1, 0)$$

as our CPS of interest and aim to compute canonical paths to $\hat{u}$.

Figure 8: (a) Phase portraits of the “control system” for $y$ with the heteroclinic orbits from $(0, 0)$ to $(1, 0)$ and from $(2, 0)$ to $(1, 0)$ (red orbit, partial plot). (b) The behavior of the $r$ system. (c) Sketch of the expected CPs in the $y_1-r$–plane (using actual numerics for parameter values $(\omega, \rho, \theta) = (1, 1, 1)$ and initial state $(x_1, x_2)(0) = (4, 0)$. The red line shows the nullcline $r = 1/\sqrt{y_1}$, above (below) which he have $\dot{r} > 0$ ($\dot{r} < 0$). As $r(0) = \|x(0)\| > 1$ we must start below the red–line by choice of $0 < y_1(0) < 1$. Together, the costates $(y_1, y_2)(0)$ must be on the heteroclinic to $y = (1, 0)$ such that $y_1(t)$ increases in $t$ in such a way that $r(t) \to 1$ as $t \to \infty$.

Now given an initial state $(x_1, x_2)(0)$ with, e.g., $\|x\| > 1$ and aiming at a CP to $\hat{u}$, i.e., the CPS associated with $(r, y_1) = (1, 1)$, we have the situation sketched in Fig. 8(c). The only possible co-state choice to end in the CPS lies on the heteroclinic connection from $(y_1, y_2) = (0, 0)$ to $(1, 0)$. Thus, $y_1$
which is the only costate which influences the states, can take values in \([0, 2]\). The state dynamics are sketched by the red triangles for \( \dot{r} < 0 \) (\( \dot{r} > 0 \)) for \( r < 1/\sqrt{y_1} \) (\( r > \sqrt{y_1} \)). Since \( r(0) > 1 \) we need to choose \( y_1(0) > 0 \) sufficiently small such that \( r(0) < 1/\sqrt{y_1(0)} \) to have \( \dot{r} < 0 \), and at the same time we need \( (y_1(0), y_2(0)) \) on the black \( y \)-heteroclinic to \((1, 0)\). The argument for \( \frac{1}{\sqrt{2}} < \|x_0\| < 1 \) works similarly.

We can also explicitly compute the Floquet multipliers of \( \hat{u} \) and the associated projections. In polar coordinates, the variational equation \((34)\) is autonomous, namely

\[
\dot{v} = J_f(\hat{u}) v, \quad J_f(\hat{u}) = \begin{pmatrix}
2\rho & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & \omega \\
0 & 0 & 2\pi\omega & 0
\end{pmatrix}, \tag{54}
\]

with eigenvalues \( \mu = (0, -\sqrt{2}\pi\omega, 2\rho, \sqrt{2}\pi\omega) \). Since \((54)\) is autonomous we obtain the multipliers by exponentiation of \( \mu \), namely

\[
\gamma = (\gamma_1, \gamma_2, \gamma_3, \gamma_4) = (1, \exp(-\sqrt{2}\pi\omega/\theta), \exp(4\pi\rho/\theta), \exp(-\sqrt{2}\pi\omega/\theta)). \tag{55}
\]

Clearly, additional to the trivial multiplier we have one stable multiplier \( \gamma_2 \) and two unstable multipliers \( \gamma_3, \gamma_4 \). Similarly, we can also compute the projection \( P \) onto the center unstable eigenspace in Cartesian coordinates (which depends on the target point \( \hat{u}_0 \)) analytically. Moreover,

\[
\gamma_2 \nearrow 1 \text{ as } \omega \to 0 \text{ or } \theta \to \infty,
\]

and conversely \( \gamma_2 \searrow 0 \) as \( \omega \to \infty \) or \( \theta \to 0 \), and this (and the analytical projection, implemented in a testing function \texttt{anaproj}) can be used to tune and test the convergence behavior of the CPs in the numerics.

### 3.1.2 pde2path implementation and results

Table 6 lists the main files for the implementation of \((51)\). Even though we do not need the bifurcation methods of pde2path, we implement the ODE \((51)\) as a pde2path problem via the convenience function \texttt{toyinit}, because the OC routines reuse these basic pde2path data structures. To show the setup, in \texttt{cmds_basic} (Listing 9), we compute some CPs with easy parameter settings, namely \((\rho, \omega, \theta) = (1, 1, 1)\), which yields \( \gamma_2 \approx 10^{-7} \) for the “leading” Floquet multiplier, and hence fast convergence to the CPS. See Fig. 9 for some basic results, which were also used to generate the blue curve in Fig. 8(c). For convenience we outsourced the main setup in \texttt{ocinit_sp}, see Listing 10, which also recalls the meaning of the most important parameters. Additionally, there are some helper functions for plotting.

<table>
<thead>
<tr>
<th>Table 6: Main scripts and functions in \texttt{oc demos/toy}.</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{cmds_basic}</td>
</tr>
<tr>
<td>\texttt{cmds_advanced}</td>
</tr>
<tr>
<td>\texttt{toyinit}</td>
</tr>
<tr>
<td>\texttt{ocinit_sp}</td>
</tr>
<tr>
<td>\texttt{sG, sGjac}</td>
</tr>
<tr>
<td>\texttt{anaproj}</td>
</tr>
</tbody>
</table>
% Basic canonical paths of ODE toy system, Cell1, initialization
p=[]; poc=[]; nt1=40; nt2=4*nt1; % # of time-slices for CPS and CP (initial)
om=1; rho=1; th=1; par=[om; rho; th]; % parameters
p=toyinit(p,2,nt1,par); % construction of explicit CPS; now initialize CP data,
% with mtom (0/1), end-error (derr) and length wT in multiples of CP-length
mtom=1; tadevs=1e-4; poc=ocinit_sp(poc,p,[4;0;0;0],nt2,mtom,tadevs);
% Cell 2: CP from states (4,0) to the CPS at states (1,0)
% Here no time adaption is necessary, i.e. this is the easiest case
poc2=isc(poc,0:0.1:1) ; toyplot (poc2);
% Cell 3: CP from states (4,4) to the CPS at states (1,0). The path only
% needs 7/8 of a circle, i.e. truncation time needs to be adapted.
poc3=poc3; poc3.oc.s0.u(1:4)=[4;4;0;0]; % overwrite starting point
poc3.oc.mtom=0; poc3=isc(poc3,0:0.2:1); toyplot(poc3);
% Cell 4: CP from states (4,0) to the CPS with different base-point
poc4=poc; s1=poc4.oc.s1;
s1.hopf.y(1:2,1:end-1)=circshift(s1.hopf.y(1:2,1:end-1),300,2); % shift CPS
s1.hopf.y(1:2,end)=s1.hopf.y(1:2,1); poc4.oc.s1=s1;
poc4=isc(poc4,0:0.1:1); toyplot(poc4);

Listing 9: Selection from ochopftriv/cmds-basic.m. Cell 1: Initialization. Cell 2: A first canonical path,
see Figure 9 for solution plots. Cell 3: A path which needs to adapt the truncation time T. A full circle
around the origin needs time 2π independent of the other states and thus a start in (4,4,0,0) to the fixed
end point (1,0,1,0) has a truncation time of 7/8π + 2πN. In Cell 4 we shift the target point on the CPS and
see similar behavior. The omitted rest of the script is plotting.

function poc=ocinit_sp(poc,p,u0,nti,mtom,tadevs)
poc.oc.s0=p; % set starting point problem structure
poc.oc.s1=p; % set end point problem structure
poc.oc.s0.hopf.y(:,end)=u0; % overwrite starting point
poc=ocinit(poc); % set standard parameters
poc.oc.s1.fuha.jcf=@(p,u) zeros(size(u)); % dummy objective function
poc.oc.rhoi=1; % set addition parameters, first the (here dummy) discount-index
poc.oc.mtom=mtom; % if 1, then use MTOM, else use bvphdw
poc.oc.nti=nti; poc.tomopt.Nmax=500; % initial and max # of points in t
poc.tomopt.err=1e-6; % max tolerance for (discrete) ODE solution
poc.tomopt.tadevs=tadevs; % max L-infty distance of endpoint of CP from CPS
poc.tomopt.M=spye(4); % set mass matrix

Listing 10: toy/ocinit_sp.m, initialization for the computation of canonical paths adapted to this problem.
First runs ocinit, which sets most values to defaults, but sets start and end-point manually because no
structure via bifurcation analysis has been constructed before.

![Graphs](image-url)

Figure 9: Canonical path from \((x_1,x_2) = (4,0)\) to \(\hat{u}_0 = (1,0,1,0)\) on CPS \(\hat{u}\), \((\rho,\omega,\theta) = (1,1,1)\).

In **cmds_advanced** we essentially decrease \(\omega\) (to 0.04) which makes the problem more expensive
due to slow convergence to \(\hat{u}\), cf. (56). For \(\omega = 0.04\) we obtain \(\gamma_2 \approx 0.52\) for the leading stable
multiplier. Intuitively, a change of \(\omega\) correspond to a rescaling of time in the costates by \(1/\omega\), i.e., small
As an example for an OC problem with Hopf bifurcations we consider optimal pollution mitigation.

### 3.2 Optimal pollution mitigation

As an example for an OC problem with Hopf bifurcations we consider

\[
V(v_0(\cdot)) = \max_{q(\cdot)} J(v_0(\cdot), q(\cdot)), \quad J(v_0(\cdot), q(\cdot)) := \int_0^\infty e^{-\rho t} J_{ca}(v(t), q(t)) \, dt, \tag{57a}
\]

with discount rate $\rho > 0$, and where $J_{ca}(v(\cdot), q(\cdot)) = \frac{1}{|\Omega|} \int_{\Omega} J_c(v(x, t), q(x, t)) \, dx$ as in §2.1 and $J_c$ is the spatially averaged current value function, with here $J_c(v, q) = pv_1 - \beta v_2 - C(q)$ the local current value, $C(q) = q + \frac{1}{2} q^2$. The state evolution is

\[
\partial_t v_1 = -q + d_1 \Delta v_1, \quad \partial_t v_2 = v_1 - \alpha(v_2) + d_2 \Delta v_2, \tag{57b}
\]

with Neumann BCs $\partial_n v = 0$ on $\partial \Omega$, where $v_1 = v_1(t, x)$ models the emissions of some firms, and $v_2 = v_2(t, x)$ is the pollution stock, while the control $q = q(t, x)$ models the firms’ abatement policies. In $J_c$, $pv_1$ and $\beta v_2$ are the firms’ value of emissions and costs of pollution, and $C(q)$ are the costs for abatement, and $\alpha(v_2) = v_2(1 - v_2)$ in (57b) is the recovery function of the environment. Again, the max in (57a) runs over all admissible controls $q$, meaning that $q \in L^\infty((0, \infty) \times \Omega, \mathbb{R})$, and we do not consider active control or state constraints. The associated ODE OC problem (no $x$-dependence of $v, q$) was set up and analyzed in [TW96 Wir00]; in suitable parameter regimes it shows Hopf bifurcations of periodic orbits for the associated canonical (ODE) system. See also, e.g., [DF91 HMN92 Wir96 KGF+02 GCF+08] for results about the occurrence of Hopf bifurcations and optimal periodic solutions in ODE OC problems.
Setting $D = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}$, $g_1(v, q) = \begin{pmatrix} -q \\ v - \alpha(w) \end{pmatrix}$, and introducing the co-states (Lagrange multipliers)

$$\lambda : \Omega \times (0, \infty) \rightarrow \mathbb{R}^2,$$

and the (local current value) Hamiltonian $\mathcal{H} = \mathcal{H}(v, \lambda, q) = J_c(v, q) + \langle \lambda, D \Delta v + g_1(v, q) \rangle$, by Pontryagins Maximum Principle we obtain

$$\begin{align*}
\partial_t v &= \partial_\lambda \mathcal{H} = D \Delta v + g_1(v, q), \quad v|_{t=0} = v_0, \\
\partial_t \lambda &= \rho \lambda - \partial_v \mathcal{H} = \rho \lambda + g_2(v, \lambda) - D \Delta \lambda,
\end{align*}$$

(58a)

where $\partial_n \lambda = 0$ on $\partial \Omega$, and

$$q = q(\lambda_1) = -(1 + \lambda_1)/\gamma. \quad (59)$$

Finally we set $u(t, \cdot) := (v(t, \cdot), \lambda(t, \cdot)) : \Omega \rightarrow \mathbb{R}^4$, and write (58a) as

$$\partial_t u = -G(u) := D \Delta u + f(u),$$

(60)

where $D = \text{diag}(d_1, d_2, -d_1, -d_2)$, $f(u) = \begin{pmatrix} -q, v_1 - \alpha(v_2), \rho \lambda_1 - p - \lambda_2, (\rho + \alpha'(v_2)) \lambda_2 + \beta \end{pmatrix}^T$.

For all parameter values, (60) has the spatially homogeneous CSS

$$u^* = (z_*(1 - z_*), z_*, -1, -(p + \rho)), \quad \text{where} \quad z_* = \frac{1}{2} \left( 1 + \rho - \frac{\beta}{p + \rho} \right).$$

We use similar parameter ranges as in Wir00, namely

$$(p, \beta, \gamma) = (1, 0.2, 300), \quad \text{and} \quad \rho \in [0.5, 0.65] \text{ as a continuation parameter}, \quad (61)$$

consider (60) over $\Omega = (-\pi/2, \pi/2)$, and set the diffusion constants to $d_1 = 0.001, d_2 = 0.2$.

**Remark 3.1.**

a) The motivation for the choice of $d_{1,2}$ is to have the first (for increasing $\rho$) Hopf bifurcation to a spatially patterned branch, and the second to a spatially uniform Hopf branch, because the former is more interesting from the PDE point of view. We use that the Hopf bifurcations for the model (60) can be analyzed by a simple modification of Wir00 Appendix A]. We find that for branches with spatial wave number $l \in \mathbb{N}$ the necessary condition for Hopf bifurcation, $K > 0$ from Wir00 (A.5)], becomes $K = - (\alpha' + d_2 l^2)(\rho + \alpha' + d_2 l^2) - d_1 l^2(\rho + d_1 l^2) > 0$. Since $\alpha' = \alpha'(z_*) < 0$, a convenient way to first fulfill $K > 0$ for $l = 1$ is to choose $0 < d_1 \ll d_2 < 1$, such that for $l = 0, 1$ the factor $\rho + \alpha' + d_2 l^2$ is the crucial one.

b) Even though we do not specify the units, $\rho \in [0.5, 0.65]$ may be considered quite large, in the following sense. Typical periods of the CPS will be between 20 and 40, and, moreover, CPs starting not close to these CPSs will need times scales $T \geq 100$ (and larger) for convergence to the CPSs, but $\rho > 0.5$ means that the large time ($T \geq 100$) behavior of a CP hardly plays a role for the value of the CP, as the discounted current value drops to $e^{-\rho t} J_c(t) < e^{-50}$. Thus, our example turns out to be somewhat academic, but nevertheless it will show the robustness of our approach.

 c) In the literature, most of the (ODE) OC examples with canonical periodic states show these at rather large discount rates, see, e.g., HMN92 KGF+02. An exception is for instance the resource
management model in [BPS01], where (ODE) CPSs are found at discount rates near $\rho = 0.1$. We have also implemented this example, including a PDE setting, but its main drawback, already hinted at in [BPS01], is that already in the ODE setting it is extremely rich in CPSs, which undergo several period doubling and fold bifurcations, and the smaller $\rho$ is again offset by rather long periods (between 20 and 60). In summary, we focus on the pollution example because it gives a clear and robust bifurcation picture.

Figure 11: (a) full spectrum of the linearization of (60) around $u^*$ at $\rho = 0.5$ on a coarse mesh with $n_p = 21$. (b) Bifurcation diagram, value $J$ over $\rho$. Black: $u^*$; blue: h1, red: h2, $J(u^H;0)$ (full line) and $J(u^H;T/2)$ (dashed line). (c) Time series of a spatially homogeneous solution, including current value $J_c$, control $q$, and co-states $\lambda_1, 2$. (d) Example plots of $u_H$ at $h1/pt8$. (e)-(g) Floquet multipliers at selected CPS, with $\rho = 0.56$ at $h1/pt10$ and at $h2/pt17$ at $h1/pt8$. The largest $\gamma$ in (f) is $\gamma_{84} \approx 10^{79}$.

The implementation of (60) works as usual, and, moreover, the computation of the bifurcation diagram of CSS and CPS, and of the Floquet multipliers, is already explained in [Uec19a, Uec19c], and the novelty here is the computation of CPs to the CPSs in a full PDE setting. Table 7 gives a few comments on the used files. In Figure 11 (essentially already contained in [Uec19a]) we give some basic results for (60) with a coarse spatial discretization of $\Omega$ by only $n_p = 21$ points (and thus $n_u = 84$). (a) shows the full spectrum of the linearization of (60) around $u^*$ at $\rho = 0.5$. (b) shows a basic bifurcation diagram. At $\rho = \rho_1 \approx 0.53$ there bifurcates a Hopf branch h1 with spatial wave number $l = 1$, and at $\rho = \rho_2 \approx 0.58$ a spatially homogeneous ($l = 0$) Hopf branch h2 bifurcates.
bifurcation diagram of CSS and CPS, and computation of some CPs.
plot of CPs, computation and plotting of diagnostics for CPs.

For $|\gamma_1 - 1| < 10^{-8}$ for all computations, i.e., a very accurate trivial multiplier, and hence we trust it. The large multipliers are those at $\rho \approx 2$, and larger, even for the coarse space discretization, as should be expected from the spectrum in (60). If $u_H$ is a $T_p$ periodic solution of (60), then, for $\phi \in [0, T_p)$, we consider

$$J(u_H; \phi) := \int_0^\infty e^{-\rho t} J_{c,a}(u_H(t + \phi)) \, dt = \frac{1}{1 - e^{-\rho T_p}} \int_0^{T_p} e^{-\rho t} J_{c,a}(u_H(t + \phi)) \, dt,$$

which in general may depend on the phase, and for $h2$ in (c) we plot $J(u_H; \phi)$ for $\phi = 0$ (full red line) and $\phi = T/2$ (dashed red line). For the spatially periodic branch $h1$, $J_{c,a}(t)$ averages out in $x$ and hence $J(u_H; \phi)$ only weakly depends on $\phi$. Thus, we first conclude that for $\rho \in (\rho_1, \rho_f)$ the spatially patterned periodic orbits from $h1$ give the highest $J$, while for $\rho \geq \rho_f$ this is obtained from $h2$ with the correct phase. The example plots (d) at $h1/pt8$ illustrate the spatio-temporal dependence of $q$, $v$, and $J_c$ on the patterned CPS.

It remains to
- compute the defects $d(u^*)$ of the CSS and $d(u_H)$ of periodic orbits on the bifurcating branches,
- compute CPs to saddle point CSSs and CPSs.

For $d(u^*)$ we find that it starts with 0 at $\rho = 0.5$, and, as expected, increases by 2 at each Hopf point. Below we shall focus on CPs to the CSSs $h1/pt8$ and $h2/pt17$, and in Fig. 11(e–g) we illustrate typical multiplier spectra, computed with pqzschr, which yields $|\gamma_1 - 1| < 10^{-8}$ for all computations, i.e., a very accurate trivial multiplier, and hence we trust it. The large multipliers are very large, i.e., $10^{40}$ and larger, even for the coarse space discretization, as should be expected from the spectrum in Fig. 11(a).

On $h1$ we find $d(u_H) = 0$ up to $pt9$, see (e) for the $n_u/2$ smallest multipliers at $pt8$, and (f) for $|\gamma_j|$ for the large ones, which are mostly real. For larger $\rho$ the $h1$ branch looses stability by a (second) multiplier going through 1, and in fact at $h1/pt8$ we have $\gamma_2 \approx 0.948$, which suggests a slow convergence of CPs to the CPS. On $h2$ we start with $d(u_H) = 3$, but $d(u_H) = 0$ after the fold until $\rho = \rho_1 \approx 0.6$, after which $d(u_H)$ increases again by multipliers going through 1. At $pt17$ we have $\gamma_2 \approx 0.905$, again suggesting slow convergence.

Nevertheless, the computation of CPs (from various initial states) to the CPSs works quite robustly, and in Fig. 12 we present some sample results from cpcmds, and from pol10DE/cpmdsode, which treats the same problem as an ODE. The idea is that the behavior of the spatially homogeneous CPs can be studied much faster in the ODE setting due to much less DoF. Also, for instance the ODE multipliers $\gamma$ at $h2/pt17$ are $\gamma \approx (0.303, 1, 1.012 \times 10^{10}, 3.789 \times 10^{10})$, i.e., $\gamma_2 = 0.303$. The associated ODE paths then also exist in the PDE as spatially homogeneous paths, and show the same convergence behavior as long as the instability which yields the existence of the patterned CPS $h1$ plays no role.

In Fig. 12(a)–(d) we show CPs to the CSS at $\rho = 0.55$ (starting from two different initial states), and to the homogeneous CPS $h2/pt17$ at $\rho = 0.57$. The convergence to the CSS is very slow as we are close to the Hopf bifurcation and hence the slowest decay rate is $\mu = 0.0059$. For $(v_0, w_0) = (0.4, 0.4)$ (significant initial emissions and pollution) we obtain a negative value $J \approx -0.1297$, as initially the

<table>
<thead>
<tr>
<th>Table 7: Main scripts and functions in ocdemos/pollution. Additionally, we locally modify some standard pde2path functions for convenience (plotting).</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcdmds, cpcmnds</td>
</tr>
<tr>
<td>cpplot, polldiagn</td>
</tr>
<tr>
<td>bifurcation diagram of CSS and CPS, and computation of some CPs.</td>
</tr>
<tr>
<td>plot of CPs, computation and plotting of diagnostics for CPs.</td>
</tr>
</tbody>
</table>
emission are strongly reduced and the initial abatement investments are high. For \((v_0, w_0) = (0, 0)\) (no initial emissions and pollution) we have \(J \approx 0.0202\) due to increasing initial emission and negative initial abatement investments. To compute CPs to the CPS, we start with a rather small \(T = 2T_p\) and for \(\|u(1) - \hat{u}_0\| > \varepsilon_{\infty} = 10^{-2}\) use the extension of the CP by copies of the CPS during the continuation in \(\alpha\). This leads to the extension of \(T\) to about \(10T_p\), and to \(\|u(1) - \hat{u}_0\|_{\infty} \approx 10^{-4}\) for the final deviation from the target \(\hat{u}_0\), and we also illustrate how to a posteriori decrease this deviation to \(10^{-6}\). We also run a few further tests, for instance computing CPs from the same ICs to shifted CPSs, e.g., \(h_2\) shifted by half a period. As expected, shifting the base point \(\hat{u}_0\) on the CPS just expands or shortens the truncation time \(T\) by half a period, see Fig. 12(d).

![Figure 12: CPs to the CSS and to different CPS from different initial states.](image)

(a) \(J = 0.1297\)

(b) \(J = 0.02015\)

(c) \(J = 0.1178\)

(d) \(J = -0.1178\)

(e) \(J = 0.1019\)

Figure 12: CPs to the CSS and to different CPS from different initial states. (a,b) CPs to the CSS at \(\rho = 0.55\), ICs \(v \equiv (0.4, 0.4)\) in (a), with zoom of the initial phase on the right, and ICs \(v \equiv (0, 0)\) in (b) showing the different initial behavior, and hence a different value. (c) CP to the homogeneous CPS at \(\rho = 0.57\), ICs \(v \equiv (0.4, 0.4)\). (d) Same as (c), but with different target \(\hat{u}_0\) on the CPS (phase shift by half a period). In (b–d) we start with a short initial \(T = 2T_p\) and set \(\varepsilon_{\infty} = 10^{-2}\), leading to repeated extension of the CPs by the CPS \(\hat{u}\) via (39). (e) CP to the patterned CPS \(h_1/pt8\) at \(\rho = 0.56\), starting close \((\alpha = 0.975)\) to the states \(v \equiv (0.205, 0.72)\) which is near the CSS at \(\rho = 0.56\). The patterning instability of the CSS then only manifests after a rather long transient. See text for further discussion.

In Fig. 12(e) we give one exemplary CP to the inhomogeneous CPS at \(\rho = 0.56\), starting with ICs \(0.975 \ast (0.21, 0.71) + 0.025 \ast \hat{u}_0\). The ICs are thus quite close to the CSS, which is stable in the ODE, but (very weakly) unstable in the PDE, as we are beyond the primary Hopf bifurcation. Consequently, the associated CP transiently decays towards the CSS, before the inhomogeneous instability manifests and the CP converges to the inhomogeneous CPS. In summary, these examples show that our algorithms allow a robust control towards CSSs and CPSs with the SPP.
4 Summary and outlook

We explained how to study OC problems of class [1] in pde2path. The class [1] is quite general, and with the pde2path machinery we have a powerful tool to first study the bifurcations of CSS/CPS. For the computation of canonical paths to CSSs and CPSs, our Algorithms 1 and 2 implement for the class [1] variants of the connecting orbits methods explained for ODE problems in [GCF+08, Chapter 7]. For the CPS case, because of the very small and very large multipliers present due diffusion and anti-diffusion, an important technical issue is the use of pqzschur to compute the projection onto the center–unstable eigenspace. Similarly, the idea to start with a rather small truncation time $T$ and then using (39), i.e., adding copies of the CPS to the CP to ensure convergence, seems crucial to have a fast and robust algorithm.

There also is a number of issues we do not address (yet), for instance inequality constraints that frequently occur in OC problems. In our examples we can simply check the natural constraints (such as $v, q \geq 0$ in the sloc example) a posteriori and find them to be always fulfilled, i.e., inactive. If such constraints become active the problem becomes much more complicated.

References


