**Vegetation community patterns**,

Michel Ferré[1], Induja Pavithran[1] and Hannes Uecker[2],

[1] BGU, michel.ferre.diaz@gmail.com and indujap2013@gmail.com, [2] Institut für Mathematik, Universität Oldenburg hannes.uecker@uol.de,

**Abstract**

This document presents the `pde2path` implementation of the vegetation community pattern formation problem discussed in reference FPBUM24 [?]. The spatial and trait spaces are discretized by a finite element and finite difference approach, respectively. Additionally, we discuss the implementation of the single-species case, where the Busse Balloon is calculated through bifurcation-point continuation.

## Contents

**1 Introduction.** **1**

**2 Community model: `pde2path` implementation.** **2**

2.1 Periodic branches. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2.2 Brute Force Busse-Balloon. . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

**3 Single species model.** **15**

3.1 Periodic branches. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.2 Busse-Balloon: Bifurcation point continuation. . . . . . . . . . . . . . . . . 20

## 1 Introduction.

This tutorial describes the `pde2path` implementation of the community model FPBUM24 [?]. The community model studies the evolution between different plant species under dryland conditions, where plant species compete for water and light. The community is divided into $N$ functional groups of plant species that differentiate between fast-growing and water stress tolerance species, being quantified by the dimensionless trait space, $0 = \chi_0 < \chi_1 < \cdots < \chi_i < \cdots < \chi_N = 1$, where $\chi_i = i\Delta\chi$ depict the $i$th functional group. The spatiotemporal community evolution is described by N continuous fields, $B_i = B_i(\chi_i, x, t)$, with $(i = 1, \dots, N)$. The below-ground and above-ground water are modeled by continuous fields $W = W(x,t)$ and $H = H(x,t)$, respectively. Two spatial dependent feedbacks, infiltration and below-ground water diffusion, are considered in their equations. The explicit form of the community model is eq:Comm

$$\partial_t B_i = \Lambda W B_i - M_i B_i + D_B \partial_x^2 B_i + D_\chi \partial_\chi^2 B_i, \tag{1a}$$
$$\partial_t W = IH - LW - \Gamma W \bar{B} + D_W \partial_x^2 W, \tag{1b}$$
$$\partial_t H = P - IH + D_H \partial_x^2 H, \tag{1c}$$

where the growth rate of the $i$th functional group, $\Lambda_i$, the infiltration rate of above-ground water into soil, $I$, and the evaporation rate, $L$, are written as,

$$\Lambda_i \equiv \frac{\Lambda_0 K_i}{\bar{B} + K_i}, \quad I \equiv \frac{A\left(\bar{\bar{B}} + fQ\right)}{\bar{\bar{B}} + Q}, \quad L \equiv \frac{L_0}{1 + R\bar{B}}$$

being $\bar{B} \equiv \sum_{i=1}^{N} B_i$ and $\bar{\bar{B}} \equiv \sum_{i=1}^{N} Y_i B_i$; terms that models the interaction the vegetation community and the above/below ground water content. $\partial_\chi^2 B \equiv N^2 \left(B_{i+1} - 2B_i + B_{i-1}\right)$, with $D_\chi$ corresponds to the mutation rate.

## 2 Community model: `pde2path` implementation.

### 2.1 Periodic branches.

subsec:PB

`pde2path` is a MATLAB package that proves a general tool for numerical continuation and bifurcation of nonlinear PDEs combining finite element methods for the spatial discretization with numerical algorithms for continuation and bifurcation methods. The numerical implementation of the system of equations eq:Comm (1) was based on a finite element approach implemented in `pde2path`—for detailed background ecker2021numerical [Uec21] and a basic introduction see reference rademacher2018oopde,uecker2019pattern [RU18, Uec19]. The basic setup is made through the files `oosetfemops.m`, file that compute the `FEM` matrices needed, `sG.m` and `sGjac.m`, functions that compute the right-hand side of equations eq:Comm (1) and its respective (analytical) Jacobian. Additionally, for simplicity, we included a temporal solver based on implicit Euler to find stationary solutions.

We customize the continuation output display windows through the functions `nplotbra.m` and `cswibra.m`, making use of the functions given by `userplot.m` and their functions related. During continuation additionally, in `pde2path`, it is possible to calculate on-demand quantifiers such as, in our case, $\chi_{max}$, a quantifier that characterizes the community composition (see reference FPBUM24 [?]). Table tb:community 1 gives an overview of the community model directory.

Table 1: Scripts and functions in `bwhComm`.

| file | purpose, remarks |
|---|---|
| `bwhinit` | initialization of problem struct `p` with standard parameter values, call of `stanpdeo1D` to generate a 1D PDE object (interval, with mesh), initialization of $u$ with $u^*$, call of `oosetfemops` to generate the FEM matrices, and finally resetting of some `pde2path` parameters to problem-adapted values. |
| `oosetfemops` | assemble and store the mass matrix $M$, and the (1-component) Neumann-Laplacian $K$. |
| `nodalf` | "nonlinearity", i.e., terms without spatial derivatives. |
| `sG,sGjac` | rhs of eq:Comm (1), and Jacobian; these here have a simple standard structure. |
| `sgbra` | output function that provides different functions to be plotted during continuation like $\chi_{max}$. |
| `userplot` | function that defines the plotting solution windows during continuation |
| `bbdns` | ... |
| `cmds1` | main script, containing basic `pde2path`commands, to be run cell-by-cell |

tb:community

Let us take a look at the `pde2path` implementation. Listing sG 1 shows the implementation of the residual. Matrices `p.mat.K` and `p.mat.M` here, which are the stiff and mass matrices, are coded in `oosetfemops.m`, see Listing ooset 3. The `FEM` formulation is then written as,

$$0 = -\mathcal{M}F(U) + \mathcal{K}U; \tag{2}$$

where $U := \left(\tilde{B}_1, \ldots, \tilde{B}_N, \tilde{W}, \tilde{H}\right)^T \in \mathbb{R}^{n_u}$ the state vector, $\tilde{B}_1 = \left(B_{1,1}, \ldots, B_{1,n_p}\right)$ and $n_u = (N+2)n_p$, being $n_p$ the first discretized plant species and the number of nodes in the `FEM` formulation, respectively. Upper tilde notation $\tilde{(\cdot)}$ depicts the discretized nature of the variable at which it is used. The nonlinear component of our system is written, component-wise,

$$\begin{aligned} F_i &:= \Lambda_i \tilde{W}\tilde{B}_i - M_i\tilde{B}_i + D_\chi N^2\left(\tilde{B}_{i+1} - 2\tilde{B}_i + \tilde{B}_{i-1}\right), \quad i = (1,\dots,N), \\ F_{N+1} &:= \tilde{I}\tilde{H} - L\tilde{W} - \Gamma\tilde{W}\tilde{B}, \\ F_{N+2} &:= P - \tilde{I}\tilde{H}. \end{aligned}$$

To approximate the nonlinear term in the FEM formulation we follow [Uec21], with $M \in \mathbb{R}^{n_p\times n_p}$ mass matrix given by `p.mat.M`, by $\mathcal{M}F(U)$. The stiff matrix, given by `oosetfemops.m`, approximate the Laplacian operator by $\mathcal{K}U$. The explicit form of $\mathcal{M}$ and $\mathcal{K}$ are given as,

$$\mathcal{M} := \begin{pmatrix} M & \dots & 0 & 0 & 0 \\ \vdots & \ddots & 0 & 0 & 0 \\ 0 & 0 & M & 0 & 0 \\ 0 & 0 & 0 & M & 0 \\ 0 & 0 & 0 & 0 & M \end{pmatrix}, \quad \mathcal{K} := \begin{pmatrix} D_BK & \dots & 0 & 0 & 0 \\ \vdots & \ddots & 0 & 0 & 0 \\ 0 & 0 & D_BK & 0 & 0 \\ 0 & 0 & 0 & D_WK & 0 \\ 0 & 0 & 0 & 0 & D_HK \end{pmatrix}$$

where $M \in \mathbb{R}^{n_p\times n_p}$ and $K \in \mathbb{R}^{n_p\times n_p}$. The code for `sG.m` read as,

sG

```
function r=sG(p,u) % RHS of bwh-community
N=p.N; n=p.np; b=reshape(u(1:N*n),n,N);      % field assign.
w=u(N*n+1:(N+1)*n); h=u((N+1)*n+1:(N+2)*n);
par=u(p.nu+1:end);pp=par(1);Lam0=par(2);Ga=par(3); % syst. params
A=par(4);R=par(5);L0=par(6);f=par(7);Q=par(8);
Kmin=par(9);Kmax=par(10); Mmin=par(11); Mmax=par(12);
Ymin=par(13);Ymax=par(14);Db=par(15);Dw=par(16);
Dh=par(17);Dchi=par(18);chimin=par(19);chimax=par(20);
K=p.mat.K; M=p.mat.M(1:n,1:n); % Stiffness and Mass matrices
ov=ones(n,1);bt=zeros(n,1); btt=bt;% b-tilde and b-tilde-tilde
chii=linspace(chimin,chimax,N);dchi=chii(2)-chii(1);
for i=1:N % fill bt and btt
    bt=bt+b(:,i);
    Yi=Ymax+chii(i)*(Ymin-Ymax);
    btt=btt+b(:,i)*Yi;
end
I=A*(btt+f*Q)./(btt+Q); L=ov*L0./(1+R*bt); % infil. and evapo. funcs
for i=1:N    %loop that assemble the finite-difference scheme
    Ki=Kmax+chii(i)*(Kmin-Kmax);
    Mi=Mmax+chii(i)*(Mmin-Mmax);
    Lami=Lam0*Ki./(bt+Ki);
    bi=b(:,i);
    switch i  % chi-diffusion terms, i=1 and i=N with Neumann BCs
        case 1; bcc=(b(:,2)-2*b(:,1))/dchi^2;
        case N;  bcc=(-2*b(:,N)+b(:,N-1))/dchi^2;
        otherwise; bcc=(b(:,i+1)-2*b(:,i)+b(:,i-1))/dchi^2;
    end
    r1=-M*(Lami.*w.*bi-Mi*bi+Dchi*bcc)+Db*K*bi;
    r((i-1)*n+1:i*n)=r1;
end
r2=-M*(I.*h-L.*w-Ga.*bt.*w)+Dw*K*w;
r3=-M*(pp-I.*h)+Dh*K*h;
```

```
r=[r;r2;r3];
```

Listing 1: `c2Dm/bwhcom/sG.m` provides the calculation of the complete RHS of the system of equations (1) eq:Comm in finite element formulation. Notice that in line 9, the diffusion coefficients are provided accordingly.

The Jacobian is created in the same way, see Listing 2 sGjac

sGjac

```
function J=sGjac(p,u)
N=p.N;n=p.np;ng=(N+2)*n;    % field assign.
w=u(N*n+1:(N+1)*n);h=u((N+1)*n+1:(N+2)*n);
par=u(p.nu+1:end);pp=par(1);Lam0=par(2);Ga=par(3); % syst. params
A=par(4);R=par(5);L0=par(6);f=par(7);Q=par(8);
Kmin=par(9);Kmax=par(10);Mmin=par(11);Mmax=par(12);
Ymin=par(13);Ymax=par(14);Db=par(15);Dw=par(16);
Dh=par(17);Dchi=par(18); chimin=par(19); chimax=par(20);
K=p.mat.K; M=p.mat.M(1:n,1:n); % Stiffness and Mass matrices
bt=zeros(n,1); btt=bt; Yi=zeros(N,1); % b-tilde and b-tilde-tilde init.
Ki=Yi;Mi=Yi; chii=linspace(chimin,chimax,N);dchi=chii(2)-chii(1);
for i=1:N % b-tilde and b-tilde-tilde assig.
    Bi=u((i-1)*n+1:i*n);
    bt=bt+Bi;
    Yi(i)=Ymax+chii(i)*(Ymin-Ymax);
    Ki(i)=Kmax+chii(i)*(Kmin-Kmax);
    Mi(i)=Mmax+chii(i)*(Mmin-Mmax);
    btt=btt+Yi(i)*Bi;
end
ov=ones(n,1);I=A*(btt+f*Q)./(btt+Q); % infil. and evapo. funcs and dev.
L=ov*L0./(1+R*bt);dL=-L0*R*ov./(1+R*bt).^2;
if p2pglob.nzi==0; J=zeros(ng); else J=p2pglob.gu; end % for allocation!
for i=1:N  % finite-difference scheme for the Jacobian.
    Bi    = u((i-1)*n+1:i*n);
    Lami  = Lam0*Ki(i)*ov./(bt+Ki(i));
    dLami = -Lam0*Ki(i)*ov./(bt+Ki(i)).^2;
    dI    = (A*Yi(i)*ov./(btt+Q) - A*Yi(i)*(btt+f*Q)./(btt+Q).^2);
    for j=1:N % D_j f_i
     djfi=dLami.*w.*Bi+(Lami.*w-Mi(i))*(i==j);
     J((i-1)*n+1:i*n,(j-1)*n+1:j*n)=-M*spdiags(djfi,0,n,n)+(i==j)*Db*K;
    end
    dwfi=Lami.*Bi; % w-derivative of fi
    J((i-1)*n+1:i*n,ng-2*n+1:ng-n)=-M*spdiags(dwfi,0,n,n);
    dbfw=dI.*h-dL.*w-Ga*w;
    J(N*n+1:(N+1)*n,(i-1)*n+1:i*n)=-M*spdiags(dbfw,0,n,n); % b-der. of fw
    dbfh=-dI.*h;
    J((N+1)*n+1:(N+2)*n,(i-1)*n+1:i*n)=-M*spdiags(dbfh,0,n,n); % b-der. of fh
    switch i  % add chi-diffusion, distinguish bulk from boundaries
  case 1; J(1:n,1:n)=J(1:n,1:n)-(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
      J(1:n,n+1:2*n)=J(1:n,n+1:2*n)+(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
  case N;  J((N-1)*n+1:N*n,(N-2)*n+1:(N-1)*n)=...
          J((N-1)*n+1:N*n,(N-2)*n+1:(N-1)*n)-(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
      J((N-1)*n+1:N*n,(N-1)*n+1:N*n)=...
          J((N-1)*n+1:N*n,(N-1)*n+1:N*n)+(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
  otherwise;
     J((i-1)*n+1:i*n,(i-2)*n+1:(i-1)*n)=...
      J((i-1)*n+1:i*n,(i-2)*n+1:(i-1)*n)-(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
     J((i-1)*n+1:i*n,(i-1)*n+1:i*n)=...
```

```
        J((i-1)*n+1:i*n,(i-1)*n+1:i*n)+2*(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
      J((i-1)*n+1:i*n,i*n+1:(i+1)*n)=...
        J((i-1)*n+1:i*n,i*n+1:(i+1)*n)-(Dchi/dchi^2)*M*spdiags(ov,0,n,n);
      end
end
dwfw=-(L+Ga*bt);
J(N*n+1:(N+1)*n,N*n+1:(N+1)*n)=Dw*K-M*spdiags(dwfw,0,n,n); % w-der of fw
dhfw=I;
J(N*n+1:(N+1)*n,(N+1)*n+1:(N+2)*n)=-M*spdiags(dhfw,0,n,n); % w-der of fw
dhfh=-I;
J((N+1)*n+1:(N+2)*n,(N+1)*n+1:(N+2)*n)=Dh*K-M*spdiags(dhfh,0,n,n); % h-der of f
if p2pglob.nzi==0; J=sparse(J); p2pglob.gu=J; p2pglob.nzi=1; end
end
```

Listing 2: `c2Dm/bwhcom/sGjac.m` computes the Jacobian associated to `sG.m` .

When coding `sG.m` and `sGjac.m` it is advised to separate the components of the unknow function and the parameters as done here. Except `p.mat.K` and `p.mat.M`, the model is completely implemented now and these two are initialized by `oosetfemops.m`, see Listing ooset 3.

ooset

```
1  function p=oosetfemops(p) % in problem-dir, since highly problem dependent
2  [K,M,~]=p.pdeo.fem.assema(p.pdeo.grid,1,1,1);
3  p.mat.K=K; p.mat.M=kron(diag(ones(1,p.N+2)),M); % scalar Lapl., full M
```

Listing 3: `c2Dm/bwhcom/oosetfemops.m` collect the FEM matrices from the PDE object allocated in p.pdeo.

Line 3 introduces the stiffness matrix `K` individually and the mass matrix `M` globally since we need to introduce the diffusion coefficients later in `sGjac.m` file. Notice that we use homogeneous Neumman boundary conditions by only call `assema`.

Once the community model is fully implemented, to start the continuation, we need to set the `pde2path` environment, model parameters, and initial steady state guess. Listing bwhinit 4 gives a basic setting for the present problem.

bwhinit

```
function p=bwhinit(lx,nx,N,par,dir,aux) % bwh-com init function
p=[]; p=stanparam(p); % p-structure creation and basic init command
p=setfn(p,dir); p.fuha.outfu=@sgbra; % folder creation and output cont. file
pde=stanpdeo1Db(0,lx,lx/nx); % standard PDE object 1D
p.np=pde.grid.nPoints;p.pdeo=pde;n=p.np; % set array-struct dimensions
p.N=N;p.nc.neq=N+2;p.ndim=1;p.vol=lx;p.nu=p.np*p.nc.neq;p.sol.xi=0.1/p.nu;
ov=ones(n,1);b=zeros(n*N,1); % initial conditions switch
x=getpte(p); chimin=par(19); chimax=par(20); delchi=(chimax-chimin)/(N-1);
switch aux.sw;
    case 1; iv=1:N;
        for i=iv; chi=chimin+(i-1)*delchi;
            b((i-1)*n+1:i*n)=10*sech(28*(chi-0.825)).^2; end
        w=0.9*ov; h=3.5*ov;
    case 2; iv=1:N;
        for i=iv; chi=chimin+(i-1)*delchi;
            b((i-1)*n+1:i*n)=8*sech(28*(chi-0.36)).*cos((2*pi*10.8/p.vol)*x).^2; end
            %b((i-1)*n+1:i*n)=0.8*sech(27*(chi-0.88)); end
        w=15*ov; h=28*ov;
    case 3; iv=1:N;
        for i=iv; chi=chimin+(i-1)*delchi;
            b((i-1)*n+1:i*n)=0.5*sech(20*(chi-0.5)).^2; end
```

```
22         w=35*ov; h=125*ov;
23 end
24 p.u=[b; w; h; par]; % concatenation of bwh variables and system parameters
25 p.sw.sfem=-1; p=oosetfemops(p); % use OOPDE, generate FEM matrices
26 p.plot.pstyle=-1; % naturally, bwh requires special plots, via userplot and
       p2pglob.ps
27 p.plot.bpcmp=2; p.plot.pcmp=2; % component branch and sol plotting
28 p.nc.ilam=1; % continue in par(1)
```

Listing 4: `cm2D/bwhcom/bwhinit.m` collects the minimal initialization commands needed.

Please revise the function `stanparam` (line 2) to obtain a detailed list of initialization commands needed for the use of `pde2path`. Line 4 generates the 1D PDE object by proving the domain length and spacing between mesh points. Line 10, through `p.sw.sfem=-1` assigns the calculation of the FEM matrices by `oossetfemops`, and then calculates them. Lines 11 and 12 set up the bifurcation check method, the use of the explicit Jacobian provided by `sGjac` function and the continuation parameter `par(1)=P`, respectively.

Now we have the basic setting done, we can start the continuation. The file `cmds1.m` give us the basic continuation for the generation of Figure 3 in reference FPBUM24 [?].

cmds1

```
close all; keep p2phome;
global p2pglob; p2pglob.gu=[]; p2pglob.nzi=0;p2pglob.ps=1;
%% parameters and folder
lx=90; nx=350; N=33;  % domain parameters
aux.solve=1;  aux.sw=1; % init solution
pp=300; Lam0=8;Ga=10; A=3000;L0=200; f=0.01; % system params.
Q=12; R=0.7;chimin=0; chimax=1;
Kmin=6.7; Kmax=35.599; Mmin=14.15; Mmax=22.515; Ymin=0.069; Ymax=0.11463;
Dchi=1e-4;Db=1; Dw=80; Dh=1800;
par=[pp; Lam0; Ga; A; R; L0; f; Q; Kmin; Kmax; Mmin; Mmax; Ymin; Ymax; Db;
    Dw; Dh; Dchi;chimin; chimax]; % system params vector.
dir0='comm'; dir=char([dir0 '/hom']); % principal and homog. sol. directory
p=bwhinit(lx,nx,N,par,dir,aux); % p-struct initialization

%% time integration for initial cont. guess
t1=0;  ts=[]; dt=2e-3; nt=4e4; nc=0; pmod=nt/20; smod=pmod;
[p,t1,ts,nc]=tintxs(p,t1,ts,dt,nt,nc,pmod,smod,@sGdns);
%% newton-iteration
p.nc.tol=1e-11;
[p.u,p.r,iter]=nloop(p,p.u); fprintf('res=%g, iter=%i\n',norm(p.r,Inf),iter
    ); plotsol(p);
%% homogeneous continuation branch.
p.nc.ilam=1; p.sol.ds=-1; % cont. param. and init. continuation step
p.sw.bifcheck=2;p.nc.tol=1e-10; % bif. detection criteria and residual tol.
p.sw.verb=2;
tic; p=cont(p,30); toc

%% cont. from Turing using cswibra
aux=[]; aux.m=3; aux.besw=0;  p0=cswibra(dir,'bpt1',aux); % basic setting
p0.nc.eigref=-3;p0.nc.neig=3; % ref. and number of eigval to calc.
p0.nc.dsmin=1e-6; p0.nc.dsmax=2; % min. and max. cont. step size
p0.nc.tol=1e-7; % residual tolerance.

%% T1
dirT=char([dir0 '/T1']);p=gentau(p0,[1]); p=setfn(p,dirT); % dir and comp.
    projection to cont.
p.sol.ds=1e-1; p=cont(p,300); % cont. step size and cont.
```

```

%% T2
dirT=char([dir0 '/T2']); p=gentau(p0,[0 1]); p=setfn(p,dirT); % dir and
    comp. projection to cont.
p.sol.ds=1e-1; p=cont(p,300); % cont. step size and cont.

%% plotting solution branches
plotbra('comm/hom','cl',[0 0.6 0],'tyun','--','cmp',3,'bplab',1,'lab',20) %
     homog. solution branch
plotbra('comm/T1','cl',[0 0.6 1],'tyun','--','cmp',3,'bplab',[1 2],'lab'
    ,260) % First Turing pattern
plotbra('comm/T2','cl','r','tyun','--','cmp',3,'bplab',[1 2],'lab',260) %
    Second Turing pattern
ylabel('$\langle \chi_{max} \rangle$','Interpreter','latex');xlabel('$P$','
    Interpreter','latex')

%% plotting solution states
plotsol('comm/hom','pt20','pfig',1);
plotsol('comm/T1','pt260','pfig',2);
plotsol('comm/T2','pt260','pfig',3);
```

Listing 5: c2Dm/bwhcom/cmds1.m Basic continuation comands.

The previous commands produce the following results for the homogeneous solution branch,

```
>> cmds1
Problem directory name: comm/hom
creating directory comm/hom
inires=62.2966
res=5.28481e-12, iter=3
step   lambda      y-axis  residual  iter meth   ds       #-EV b(0)
   0  300.00000    0.65371 5.28e-12    0  nat  0.000e+00   0  1.023e-18
   1  299.90000    0.65386 5.39e-12    2  nat   -0.10000   0  9.903e-19
   .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
  21  279.90005    0.68657 6.69e-12    2  nat   -1.00000   0  1.237e-20
  22  278.90005    0.68837 6.22e-12    2  nat   -1.00000   0  9.656e-21
   1 possible bifurcation between 278.9 and 277.9, om=0
 checking lam=278.4 ... ok, ineg=2
 checking lam=278.65 ... ok, ineg=0
 checking lam=278.525 ... ok, ineg=1
 checking lam=278.588 ... ok, ineg=1
 checking lam=278.619 ... ok, ineg=0
 mu_r=0.00015167, mu_i=0
<phi,psi>=-1.6187e-12,BP
  23  2.78619e+02 (BP, saved to comm/hom/bpt1.mat) bisection steps 5, last ds -0.015625
  24  277.90005    0.69019 7.75e-12    2  nat   -1.00000   3  7.514e-21
  25  276.90005    0.69203 6.54e-12    2  nat   -1.00000   3  5.828e-21
Timing: total=41.6571, av.step=1.49552, av.Newton=0.829873, av.spcalc=0.182782
Elapsed time is 41.681327 seconds.
```

Calculating the first Turing mode

```
lam=278.6188; 8 smallest eigenvalues: 0.000152   0.00562    0.0181    0.0328    0.0578
```

```
using m=3
Problem directory name: comm/T1
step   lambda      y-axis  residual  iter meth   ds       #-EV b(0)
   stepsizecontrol: dlam=1.27284, res=6.78391e-12, reducing ds to 0.005
   stepsizecontrol: dlam=0.311202, res=3.0601e-09, increasing ds to 0.01
   1 possible bifurcation between 278.619 and 278.93, om=0
 checking lam=278.774 ... ok, ineg=1
 checking lam=278.656 ... ok, ineg=1
 checking lam=278.638 ... ok, ineg=1
 checking lam=278.628 ... ok, ineg=1
 checking lam=278.623 ...nloop: damp alpha=0.5, res=0.000765044
No convergence, localization might be poor ...
 checking lam=278.623 ...nloop: damp alpha=0.5, res=0.000765044
No convergence, localization might be poor ...
 checking lam=278.623 ...nloop: damp alpha=0.5, res=0.000765044
No convergence, localization might be poor ...
mu_r=-0.00119937, mu_i=0, no convergence
   1  278.93000    0.68802 3.06e-09    2  arc    0.00500   1  9.275e-20
   stepsizecontrol: dlam=0.0099996, res=1.22785e-09, increasing ds to 0.02
   1 possible bifurcation between 278.93 and 278.94, om=0
 checking lam=278.935 ... ok, ineg=2
 checking lam=278.933 ... ok, ineg=2
 checking lam=278.931 ... ok, ineg=2
 checking lam=278.931 ... ok, ineg=2
 checking lam=278.93 ... ok, ineg=2
 checking lam=278.93 ... ok, ineg=2
 checking lam=278.93 ... ok, ineg=2
mu_r=0.0106918, mu_i=0, no convergence
   2  278.94000    0.68799 1.23e-09    1  nat    0.01000   2  2.778e-21
   stepsizecontrol: dlam=0.0199992, res=1.73947e-08, increasing ds to 0.04
   3  278.96000    0.68794 1.74e-08    1  nat    0.02000   2  1.281e-20
   stepsizecontrol: dlam=0.0399985, res=5.22826e-12, increasing ds to 0.08
   4  279.00000    0.68783 5.23e-12    2  nat    0.04000   2  1.212e-20
   stepsizecontrol: dlam=0.0799973, res=1.82201e-11, increasing ds to 0.16
   1 possible bifurcation between 279 and 279.08, om=0
 checking lam=279.04 ... ok, ineg=2
 checking lam=279.06 ... ok, ineg=3
 checking lam=279.05 ... ok, ineg=2
 checking lam=279.055 ... ok, ineg=2
 checking lam=279.057 ... ok, ineg=3
 checking lam=279.056 ... ok, ineg=2
 checking lam=279.057 ... ok, ineg=3
 mu_r=-1.25474e-05, mu_i=0
<phi,psi>=5.95988e-08,BP
   5  2.79057e+02 (BP, saved to comm/T1/bpt1.mat) bisection steps 7, last ds -0.0003125
   also saved to comm/T1/pt5.mat
   6  279.08000    0.68761 1.82e-11    2  nat    0.08000   3  1.289e-20
   stepsizecontrol: dlam=0.0799976, res=6.83653e-12, increasing ds to 0.16
   7  279.15999    0.68739 6.84e-12    2  nat    0.08000   3  1.371e-20
   stepsizecontrol: dlam=0.159996, res=3.29977e-10, increasing ds to 0.32
   8  279.31999    0.68695 3.30e-10    2  nat    0.16000   3  1.547e-20
   stepsizecontrol: dlam=0.319993, res=9.48978e-09, increasing ds to 0.64
   9  279.63998    0.68607 9.49e-09    2  nat    0.32000   3  1.959e-20
  10  280.27997    0.68434 6.06e-12    3  nat    0.64000   3  3.039e-20
  11  280.91996    0.68262 6.80e-09    2  nat    0.64000   3  4.610e-20
  12  281.55995    0.68091 8.76e-10    2  nat    0.64000   3  6.758e-20
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 299  383.47742    0.43435 1.06e-11    2  nat    0.64000   0  6.241e-08
 300  382.83742    0.43430 1.25e-11    2  nat    0.64000   0  6.274e-08
Timing: total=540.207, av.step=1.71267, av.Newton=0.794449, av.spcalc=0.301111
```

Calculating the second Turing mode

```
Problem directory name: comm/T2
creating directory comm/T2
step   lambda      y-axis  residual  iter meth   ds       #-EV b(0)

nloopext: damp alpha=0.5, res=19.3513, ds=0.1
   stepsizecontrol: dlam=-3.77511, res=15.383, reducing ds to 0.05

nloopext: damp alpha=0.5, res=55.4696, ds=0.05
   stepsizecontrol: dlam=15.7879, res=1.47795, reducing ds to 0.025
   stepsizecontrol: dlam=10.1701, res=2.13706e-11, reducing ds to 0.0125
   stepsizecontrol: dlam=1.97532, res=1.09421e-11, reducing ds to 0.00625
   stepsizecontrol: dlam=0.388289, res=1.63365e-09, increasing ds to 0.0125
   1 possible bifurcation between 278.619 and 279.007, om=0
 checking lam=278.813 ... ok, ineg=2
 checking lam=278.621 ...nloop: damp alpha=0.5, res=0.00783928
No convergence, localization might be poor ...
 checking lam=278.621 ...nloop: damp alpha=0.5, res=0.00783928
No convergence, localization might be poor ...
 checking lam=278.621 ...nloop: damp alpha=0.5, res=0.00783928
No convergence, localization might be poor ...
 checking lam=278.621 ...nloop: damp alpha=0.5, res=0.00783928
No convergence, localization might be poor ...
 checking lam=278.621 ...nloop: damp alpha=0.5, res=0.00783928
No convergence, localization might be poor ...
 checking lam=278.621 ...nloop: damp alpha=0.5, res=0.00783928
No convergence, localization might be poor ...
mu_r=0.00610572, mu_i=0, no convergence
   1  279.00709    0.68774 1.63e-09    2  arc    0.00625   2  3.450e-19
   stepsizecontrol: dlam=0.0124997, res=6.78514e-10, increasing ds to 0.025
   1 possible bifurcation between 279.007 and 279.02, om=0
 checking lam=279.013 ... ok, ineg=3
 checking lam=279.01 ... ok, ineg=3
 checking lam=279.009 ... ok, ineg=3
 checking lam=279.008 ... ok, ineg=3
 checking lam=279.007 ... ok, ineg=3
 checking lam=279.007 ... ok, ineg=3
 checking lam=279.007 ... ok, ineg=3
mu_r=-0.0179504, mu_i=0, no convergence
   2  279.01959    0.68771 6.79e-10    1  nat    0.01250   3  5.335e-20
   stepsizecontrol: dlam=0.0249994, res=9.86429e-09, increasing ds to 0.05
   3  279.04459    0.68764 9.86e-09    1  nat    0.02500   3  8.893e-21
   stepsizecontrol: dlam=0.0499987, res=6.59872e-12, increasing ds to 0.1
   4  279.09459    0.68751 6.60e-12    2  nat    0.05000   3  1.378e-20
   stepsizecontrol: dlam=0.0999976, res=7.68363e-12, increasing ds to 0.2
   5  279.19458    0.68725 7.68e-12    2  nat    0.10000   3  1.490e-20
   stepsizecontrol: dlam=0.199996, res=3.31784e-10, increasing ds to 0.4
   6  279.39458    0.68672 3.32e-10    2  nat    0.20000   3  1.723e-20
   stepsizecontrol: dlam=0.399993, res=9.9949e-09, increasing ds to 0.8
   7  279.79457    0.68566 9.99e-09    2  nat    0.40000   3  2.284e-20
   8  280.59456    0.68357 6.06e-12    3  nat    0.80000   3  3.817e-20
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 174  413.39413    0.46274 2.26e-09    2  nat    0.80000   3  4.420e-09
 175  414.19412    0.46081 2.99e-08    2  nat    0.80000   3  5.290e-09
 176  414.99411    0.45841 1.01e-11    3  nat    0.80000   3  6.712e-09
nloop: damp alpha=0.5, res=0.0513855
   stepsizecontrol: dlam=0.799971, res=0.0513855, reducing ds to 0.4
   stepsizecontrol: dlam=0.399986, res=2.66083e-11, increasing ds to 0.8
 177  415.39409    0.45671 2.66e-11    3  nat    0.40000   3  7.832e-09
nloop: damp alpha=0.5, res=0.188948
   stepsizecontrol: dlam=0.799932, res=0.188948, reducing ds to 0.4
nloop: damp alpha=0.5, res=0.0464601
   stepsizecontrol: dlam=0.399966, res=0.0464601, reducing ds to 0.2
   stepsizecontrol: dlam=0.199983, res=1.53913e-08, increasing ds to 0.4
 178  415.59408    0.45528 1.54e-08    3  nat    0.20000   3  8.919e-09
nloop: damp alpha=0.5, res=0.124661
   stepsizecontrol: dlam=0.399841, res=0.124661, reducing ds to 0.2
nloop: damp alpha=0.5, res=0.0638903
   stepsizecontrol: dlam=0.199921, res=0.0638903, reducing ds to 0.1
nloop: damp alpha=0.5, res=0.0158279
   stepsizecontrol: dlam=0.0999603, res=0.0158279, reducing ds to 0.05
nloop: damp alpha=0.5, res=0.000460114
   stepsizecontrol: dlam=0.0499801, res=0.000460114, reducing ds to 0.025
   stepsizecontrol: dlam=0.0249901, res=3.12767e-08, increasing ds to 0.05
 179  415.61907    0.45495 3.13e-08    2  nat    0.02500   3  9.149e-09
nloop: damp alpha=0.5, res=0.0140019
   stepsizecontrol: dlam=0.0499587, res=0.0140019, reducing ds to 0.025
nloop: damp alpha=0.5, res=0.000801025
   stepsizecontrol: dlam=0.0249793, res=0.000801025, reducing ds to 0.0125
   stepsizecontrol: dlam=0.0124897, res=9.64975e-10, increasing ds to 0.025
 180  415.63156    0.45472 9.65e-10    3  nat    0.01250   3  9.377e-09
nloop: damp alpha=0.5, res=0.00441373
   stepsizecontrol: dlam=0.0249585, res=0.00441373, reducing ds to 0.0125
nloop: damp alpha=0.5, res=0.000450962
   stepsizecontrol: dlam=0.0124793, res=0.000450962, reducing ds to 0.00625
   stepsizecontrol: dlam=0.00623964, res=1.74337e-08, increasing ds to 0.0125
 181  415.63779    0.45455 1.74e-08    2  nat    0.00625   3  9.503e-09
nloop: damp alpha=0.5, res=0.00251158
   stepsizecontrol: dlam=0.0124538, res=0.00251158, reducing ds to 0.00625
nloop: damp alpha=0.5, res=0.000469488
   stepsizecontrol: dlam=0.00622688, res=0.000469488, reducing ds to 0.003125
   stepsizecontrol: dlam=0.00311344, res=3.03109e-08, increasing ds to 0.00625
   1 possible bifurcation between 415.638 and 415.641, om=0
 checking lam=415.639 ... ok, ineg=3
 checking lam=415.64 ... ok, ineg=3
 checking lam=415.641 ... ok, ineg=2
 checking lam=415.64 ... ok, ineg=3
 checking lam=415.64 ... ok, ineg=2
 checking lam=415.64 ... ok, ineg=3
 checking lam=415.64 ... ok, ineg=3
 mu_r=-3.82211e-06, mu_i=0
<phi,psi>=-2.32234e-07,BP
 182  4.15640e+02 (BP, saved to comm/T2/bpt1.mat) bisection steps 7, last ds 1.2207e-05
 183  415.64091    0.45441 3.03e-08    2  nat    0.00313   2  9.619e-09
nloop: damp alpha=0.5, res=0.000423671
   stepsizecontrol: dlam=0.00309418, res=0.000423671, reducing ds to 0.0015625
   stepsizecontrol: dlam=0.00154709, res=8.7473e-10, increasing ds to 0.003125
   1 possible bifurcation between 415.641 and 415.642, om=0
 checking lam=415.642 ... ok, ineg=2
 checking lam=415.642 ... ok, ineg=2
 checking lam=415.642 ... ok, ineg=2
 checking lam=415.642 ... ok, ineg=2
 checking lam=415.642 ... ok, ineg=2
 checking lam=415.642 ... ok, ineg=2
 checking lam=415.642 ... ok, ineg=1
 mu_r=1.25927e-06, mu_i=0
<phi,psi>=-1.8246e-10,BP
 184  4.15642e+02 (BP, saved to comm/T2/bpt2.mat) bisection steps 7, last ds -6.10352e-
 185  415.64246    0.45429 8.75e-10    3  nat    0.00156   1  9.739e-09
nloop: damp alpha=0.5, res=0.000403631
   stepsizecontrol: dlam=0.00146613, res=0.000403631, reducing ds to 0.00078125
nloop: damp alpha=0.5, res=0.000163929
   stepsizecontrol: dlam=0.000733066, res=0.000163929, reducing ds to 0.000390625
nloop: damp alpha=0.5, res=4.09644e-05
   stepsizecontrol: dlam=0.000366533, res=4.09644e-05, reducing ds to 0.000195313
   stepsizecontrol: dlam=0.000183267, res=4.83095e-09, increasing ds to 0.000390625
 186  415.64264    0.45426 4.83e-09    2  nat    0.00020   1  9.765e-09
```

```
nloop: damp alpha=0.5, res=0.000105241
   stepsizecontrol: dlam=0.000325408, res=0.000105241, reducing ds to 0.000195313
nloop: damp alpha=0.5, res=2.63406e-05
   stepsizecontrol: dlam=0.000162704, res=2.63406e-05, reducing ds to 9.76563e-05
   stepsizecontrol: dlam=8.13519e-05, res=5.56829e-08, increasing ds to 0.000195313
 187  415.64272    0.45421 5.57e-08    4  nat    0.00010   1  9.801e-09
   stepsizecontrol: dlam=-0.000272382, res=1.42743e-08, increasing ds to 0.000390625
   1 possible bifurcation between 415.643 and 415.642, om=0
 checking lam=415.643 ... ok, ineg=0
 checking lam=415.643 ... ok, ineg=0
 checking lam=415.643 ... ok, ineg=0
 checking lam=415.643 ... ok, ineg=0
 checking lam=415.643 ... ok, ineg=0
 checking lam=415.643 ... ok, ineg=0
 checking lam=415.643 ... ok, ineg=0
 mu_r=0.000163737, mu_i=0
<phi,psi>=-0.0444386, Fold
 188  415.64245    0.45414 1.43e-08    1  arc    0.00020   0  9.919e-09
   stepsizecontrol: dlam=-0.000183715, res=4.36202e-11, increasing ds to 0.000390625
 189  415.64226    0.45411 4.36e-11    2  nat    0.00020   0  9.891e-09
   stepsizecontrol: dlam=-0.000376292, res=2.66905e-10, increasing ds to 0.00078125
 190  415.64189    0.45408 2.67e-10    2  nat    0.00039   0  9.913e-09
   stepsizecontrol: dlam=-0.000765194, res=9.64048e-10, increasing ds to 0.0015625
 191  415.64112    0.45403 9.64e-10    2  nat    0.00078   0  9.952e-09
   stepsizecontrol: dlam=-0.00154559, res=2.54436e-09, increasing ds to 0.003125
 192  415.63958    0.45395 2.54e-09    2  nat    0.00156   0  1.001e-08
   stepsizecontrol: dlam=-0.00310778, res=5.73319e-09, increasing ds to 0.00625
 193  415.63647    0.45385 5.73e-09    2  nat    0.00313   0  1.009e-08
   stepsizecontrol: dlam=-0.00623277, res=1.20292e-08, increasing ds to 0.0125
 194  415.63024    0.45370 1.20e-08    2  nat    0.00625   0  1.021e-08
   stepsizecontrol: dlam=-0.0124829, res=2.45026e-08, increasing ds to 0.025
 195  415.61775    0.45349 2.45e-08    2  nat    0.01250   0  1.039e-08
   stepsizecontrol: dlam=-0.0249832, res=4.93441e-08, increasing ds to 0.05
 196  415.59277    0.45320 4.93e-08    2  nat    0.02500   0  1.063e-08
   stepsizecontrol: dlam=-0.0499837, res=9.89227e-08, increasing ds to 0.1
 197  415.54279    0.45280 9.89e-08    2  nat    0.05000   0  1.098e-08
   stepsizecontrol: dlam=-0.0999843, res=1.08713e-11, increasing ds to 0.2
 198  415.44280    0.45225 1.09e-11    3  nat    0.10000   0  1.179e-08
   stepsizecontrol: dlam=-0.199985, res=9.9476e-12, increasing ds to 0.4
 199  415.24282    0.45150 9.95e-12    3  nat    0.20000   0  1.263e-08
   stepsizecontrol: dlam=-0.399987, res=8.04334e-12, increasing ds to 0.8
 200  414.84283    0.45052 8.04e-12    3  nat    0.40000   0  1.384e-08
 201  414.04284    0.44926 1.15e-11    3  nat    0.80000   0  1.555e-08
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 300  334.84276    0.43776 9.88e-12    2  nat    0.80000   0  4.402e-08
Timing: total=613.351, av.step=1.9546, av.Newton=0.854463, av.spcalc=0.430008
```

The final lines of the file `cmds1.m` produces the results given in figure (1) fig:min_bif

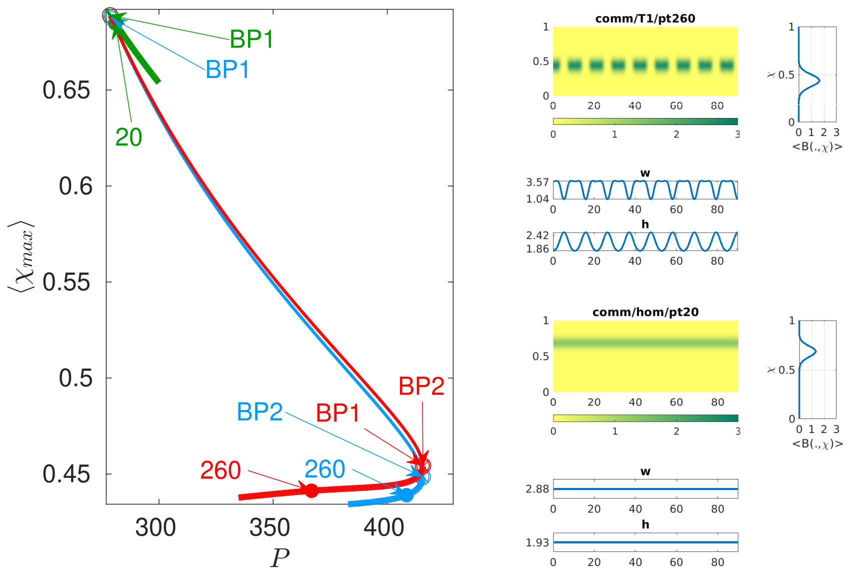

Figure 1: A minimal bifurcation diagram depicting three states by their respective numbers. Bifurcation points are labeled by BP. fig:min_bif

## 2.2 Brute Force Busse-Balloon.

In reference FPBUM24[?], we discuss the plant community evolution under precipitation variation, showing the community composition shifts to stress-tolerance species as precipitation decreases for homogeneous solutions, while patterns, due to the increase of water availability, shift back to fast-growing species exhibiting *patch thinning* along the periodic branch as precipitation further decreases. Such dynamic is plotted in Figure 3 in reference FPBUM24[?]. Additionally, to compare the plant community pattern solutions to the single plant species pattern solutions—for $\chi = 1$ and $\chi = 0$—we characterize the Busse-Balloon for each and present them in Figure (4) en reference FPBUM24[?]. Usually, Busse-Balloon calculation can be done by bifurcation point continuation (see next section sec:BBs3.2), which requires extending the discretized system size up to double. This limits us from working on the current community model due to the technical issue (computer resource-wise); consequently, we went for a brute force approach, i.e., calculating the periodic branches, saving the stable ranges, and interpolating the boundaries to generate a smoother result. The brute-force Bussee-Balloon code is given as follows,

```
function [kv, pv]=bfbb(kv,pv,dir,k0) % Brute Force Busse Ballon
% scans branch in dir for stable solns and returns the par-values
% wave-nr k user supplied!  (could be obtained from soln via FFT,
% but here we keep life simple)
p=loadpp(dir); pv0=p.branch(4,:); inv=p.branch(3,:); % take data from
    branch
lb=length(pv0);
for i=1:lb;    % if stable, then add point (and wave-nr) to list
    if inv(i)<1; kv=[kv k0]; pv=[pv pv0(i)]; end
end
```

Listing 6: c2Dm/bwhcom/bfbb.m.

By considering the original data of reference FPBUM24[??], the following file,

```
close all; clc;clear all;
keep p2phome; global p2pglob; p2pglob.gu=[]; p2pglob.nzi=0;p2pglob.ps=1;
```

```
%% ------- brute force Busse ballon, extracting stability info from computed branches, chi=0.88
% here passing wave-nr to bfbb; to improve: compute more T-branches,
kvcm0=[]; pvcm0=[];

[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T1',0.58643) ;[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T2',0.56549);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T3',0.54454) ;[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T4',0.62832);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T5',0.60737) ;[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T6',0.5236);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T7',0.64926) ;[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T8',0.67021);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T9',0.73304) ;[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T10',0.77493);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T11',0.81681);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T12',0.87965);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T13',0.90059);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T14',0.46077);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T15',0.43982);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T16',0.41888);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T17',0.39794);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T18',0.79587);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T19',0.83776);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T20',0.8587);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T21',0.75398);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T22',0.71209);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T23',0.48171);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T24',0.69115);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T25',0.50265);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T26',0.35605);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T27',0.3351); [kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T28',0.31416);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T29',0.27227);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T30',0.23038);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T31',0.1885); [kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T32',0.14661);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T33',0.10472);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T34',0.083776);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T35',0.20944);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T36',0.16755);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T37',0.12566);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T38',0.25133);
[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T39',0.29322);[kvcm0 pvcm0]=bfbb(kvcm0,pvcm0,'11/T40',0.37699);

%% plot the BB
mclf(17);
hold on
plot(pvcm0,kvcm0,'x','Color','m');
plot(pp0,0.07854*l0,'LineWidth',2,'Color','b');
plot(pp1,0.07854*l1,'LineWidth',2,'Color','r');

ylim([0,1.2])
xlabel('P'); ylabel('k'); title('BB','Interpreter','latex');
set(gca,'fontsize',14);
```
Listing 7: c2Dm/bwhcom/cmds2.m.

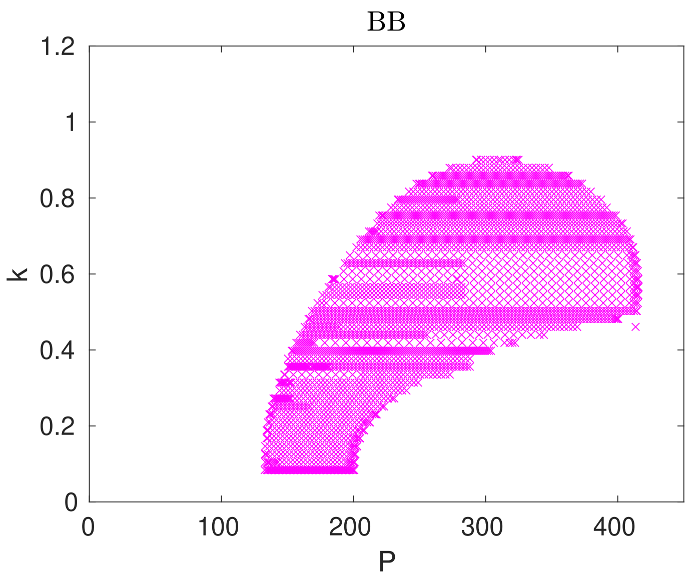

Figure 2: Brute Force Bussee-Balloon using the original data of reference [FPBUM24]. fig:brut_for_bb

produces a brute fore Bussee-Balloon (see Figure (2)). fig:brut_for_bb

## 3 Single species model.

### 3.1 Periodic branches.

The single species models (SSM) eq:SSM

$$\partial_t B_i = \Lambda_i W B_i - M_i B_i + D_B \partial_x^2 B_i, \tag{3a}$$
$$\partial_t W = IH - LW - \Gamma W \bar{B} + D_W \partial_x^2 W, \tag{3b}$$
$$\partial_t H = P - IH + D_H \partial_x^2 H \tag{3c}$$

is a standard semilinear reaction-diffusion system and hence can be treated similarly to the models in ueckeR2021numerical [Uec21, Ch.9], and associated demos available at p2phome [Uec23a]. We also refer to ueckeR2021numeri [Uec21, Uec19] and the tutorials at p2phome [Uec23a] for general background and usage of `pde2path`. The `bwhsingle` folder comes with the files needed to calculate a minimal version of the bifurcation diagram exhibited in Figure 1 in reference FPBUM24 [?] and the ones necessary for the Busse Ballon calculation via branch point continuation (BPC). Similar treatment as the `pde2path` implementation treatment gived in section subsec:PB (2.1) is given. A list of the pertinent files is given in Table tab1 2.

Table 2: Scripts and functions in `bwhsingle`. Associated to most `cmds*`–scripts are `cmds*plot` scripts for plotting; all figure numbers refer to [?]FPBUM24. 1st two blocks: scripts; 3rd block: problem describing functions and overloads of `pde2path` library functions and convenience functions. tab1

| file | purpose, remarks |
|---|---|
| `cmds1` | starting script ... Fig.4 |
| `bwhinit` | initialization of problem struct `p` with standard parameter values, call of `stanpdeo1D` to generate a 1D PDE object (interval, with mesh), initialization of $u$ with $u^*$, call of `oosetfemops` to generate the FEM matrices, and finally resetting of some `pde2path` parameters to problem-adapted values. |
| `oosetfemops` | assemble and store the mass matrix $M$, and the (1-component) Neumann-Laplacian $K$. |
| `nodalf` | "nonlinearity", i.e., terms without spatial derivatives, called in `hotintxs`. |
| `sG,sGjac` | rhs of (3)eq:SSM, and Jacobian; these here have a simple standard structure. |
| `bpjac` | implements $\partial_u(G_u^T\psi)$ for BPC, see [Uec21, §3.6.1]uecker2021numerical. |
| `sgbra` | mod of library function `stanbra`; |

The script files are presented in detail as follows. `bwhinit.m` set up the continuation environment. Please revise the function `stanparam` (line 2) to obtain a detailed list of initialization commands needed for the use of `pde2path`. Line 4 generates the 1D PDE object by proving the domain length and spacing between mesh points, while we assign the calculation of the FEM matrices for `oossetfemops` through `p.sw.sfem=-1` in Line 10, and then calculates them. Lines 11 and 12 set up the bifurcation check method, the use of the explicit Jacobian provided by `sGjac` function (`p.sw.jac=1`) and the continuation parameter `par(1)=P` by `p.sw.bifchek=2`.

```
1  function p=bwhinit(lx,nx,par,b0,w0,h0,dir) % bwh-single species init
      function
2  p=[]; p=stanparam(p);    % p-structure creation and basic init command
3  p=setfn(p,dir); % init. dir.
4  p.fuha.outfu=@sgbra;  % set output quantities from continuation
5  pde=stanpdeo1Db(0,lx,lx/nx);    % standard PDE objects 1D
6  p.pdeo=pde;p.vol=lx; p.np=pde.grid.nPoints;
7  n=p.np;p.ndim=1;p.nc.neq=3;   % set array-struct dimensions
8  p.nu=p.np*p.nc.neq;p.sol.xi=0.1/p.nu;
9  b=b0*ones(n,1);w=w0*ones(n,1);h=h0*ones(n,1);
10 p.u=[b; w;h; par];               % init sol with parameters appended
11 p.sw.sfem=-1;p=oosetfemops(p); % use OOPDE, generate FEM matrices
12 p.sw.bifcheck=2;p.sw.jac=1;     % set bp-detection and numerical Jac
13 p.nc.ilam=1; % continue in par(1)
14 p.sol.ds=0.01;p.nc.dsmin=0.01;p.nc.dsmax=3; %  set arc-length cont params
15 p.sw.qjac=0; % bif. point. cont. pure numeric.
```

Listing 8: `cm2D/bwhsingle/bwhinit.m` Minimal initialization commands.

In `oosetfemops.m`the `FEM` matrices from the PDE object allocated in `p.pde` are obtained and assigned to the `p.mat.K` for the stiffness matrix and `p.mat.M` for the mass matrix. Notice that only the mass matrix is allocated from `oosetfemops.m` since we need to introduce the diffusion coefficient later in `sGjac.m`.

```
function p=oosetfemops(p) % set FEM operators, homog. Neuman BC by default
[K,M,~]=p.pdeo.fem.assema(p.pdeo.grid,1,1,1); % FEM matrices
p.mat.K=K; p.mat.M=kron(diag([1,1,1]),M); % scalar Lapl., full M
```

Listing 9: `cm2D/bwhsingle/oosetfemops.m` collect the FEM matrices.

As in the previous section (subsec:PB 2.1), we implement the nonlinear terms through the `nodalf.m` file, where the $\chi$-dependent terms are calculated accordingly[1]. In line 6, the diffusion coefficients are provided accordingly.

```
 1 function r=sG(p,u) % for bwh-singles species spatial implementation
 2 n=p.np; b=u(1:n); w=u(n+1:2*n); h=u(2*n+1:3*n); % nodes and field
     assignation
 3 par=u(p.nu+1:end); pp=par(1); Lam0=par(2); Ga=par(3); %params.
 4 A=par(4); R=par(5); L0=par(6); f=par(7); Q=par(8);
 5 Kmin=par(9); Kmax=par(10); Mmin=par(11); Mmax=par(12); % trait-depent.
     params1
 6 Ymin=par(13); Ymax=par(14);
 7 Db=par(15); Dw=par(16); Dh=par(17); % diff. coeff.
 8 chi=par(18); l=par(19); % trait and spatial scl.
 9 K=p.mat.K; M=p.mat.M(1:n,1:n); ov=ones(n,1); % stiff and Mass FEM
10
11 Yi=Ymax+chi*(Ymin-Ymax); % trait-depent. params2
12 Mi=Mmax+chi*(Mmin-Mmax);
13 Ki=Kmax+chi*(Kmin-Kmax);
14 Lam=Lam0*ov*Ki./(b+Ki);  % growth-rate
15 I=A*(Yi*b+f*Q)./(Yi*b+Q); % infilt.
16 L=L0*ov./(1+R*b);         % evap.
17
18 r1=-M*(Lam.*w.*b-Mi*b)+l^2*Db*K*b;
19 r2=-M*(I.*h-L.*w-Ga*w.*b)+l^2*Dw*K*w;
20 r3=-M*(pp-I.*h)+l^2*Dh*K*h;
21
22 r=[r1;r2;r3];
```

Listing 10: `c2Dm/bwhsingle/sG.m`

Notice we can make the continuation based on a numerical jacobian; however, for numerical performance, we decided to provide the jacobian explicitly through `p.sw.jac=1`, being the Jacobian calculated by `sGjac.m`,

```
 1 function Gu=sGjac(p,u)
 2 global p2pglob % used for sparsity pattern
 3 n = p.np;
 4 par=u(p.nu+1:end);
 5 Db=par(15); Dw=par(16); Dh=par(17);l=par(19);
 6 [f1b,f1w,f1h,f2b,f2w,f2h,f3b,f3w,f3h] = njac(p,u);
 7 Fu=[[spdiags(f1b,0,n,n),spdiags(f1w,0,n,n),spdiags(f1h,0,n,n)];
 8     [spdiags(f2b,0,n,n),spdiags(f2w,0,n,n),spdiags(f2h,0,n,n)];
 9     [spdiags(f3b,0,n,n),spdiags(f3w,0,n,n),spdiags(f3h,0,n,n)]];
10 Gu= kron([[l^2*Db,0,0];[0,l^2*Dw,0];[0,0,l^2*Dh]],p.mat.K) - p.mat.M*Fu;
11 end
12
13 function [f1b,f1w,f1h,f2b,f2w,f2h,f3b,f3w,f3h] = njac(p,u)
14 %Jacobian, nodal version
15 n = p.np;
16 b=u(1:n);
17 w=u(n+1:2*n); h=u(2*n+1:3*n);
18 par=u(p.nu+1:end); pp=par(1); Lam0=par(2); Ga=par(3); A=par(4); R=par(5);
     L0=par(6);
19 f=par(7); Q=par(8); Kmin=par(9); Kmax=par(10); Mmin=par(11); Mmax=par(12);
20 Ymin=par(13); Ymax=par(14); Db=par(15); Dw=par(16); Dh=par(17); chi=par(18)
     ;
```

[1]We implement the FEM similarly as the subsection (subsec:PB 2.1)

```
ov=ones(n,1);
Yi = Ymax + chi*(Ymin-Ymax);
Ki = Kmax + chi*(Kmin-Kmax);
Mi = Mmax + chi*(Mmin-Mmax);
I = A*(Yi*b+f*Q)./(Yi*b+Q);
L=L0*ov./(1+R*b);
Lam=Lam0*Ki*ov./(b+Ki);
dI = A*Yi*ov./(Yi*b+Q) - A*Yi*(Yi*b+f*Q)./(Yi*b+Q).^2;
dLam=-Lam0*Ki*ov./(b+Ki).^2;
dL =-L0*R*ov./(1+R*b).^2;
f1b = dLam.*w.*b + Lam.*w - Mi; f1w = Lam.*b; f1h =0*b;
f2b = dI.*h-Ga*w-dL.*w; f2w = -L - Ga*b; f2h = I;
f3b = -dI.*h; f3w = 0*b ; f3h = -I;
end
```

Listing 11: `c2Dm/bwhsingle/sGjac.m`

Now we have all set up to continuate some solutions branches by running `cmds1.m`,

```
close all; keep p2phome; % script for 1D
global p2pglob; p2pglob.ps=1;   % plotstyle
%% parameters and folder
pp=300;lx=85; nx=300; % domain parameters
Lam0=8;Ga=10; A=3000;L0=200; f=0.01;  % system params.
Q=12; R=0.7;chimin=0.0; chimax=1;
Kmin=6.7; Kmax=28.93; Mmin=14.15; Mmax=20.585; Ymin=0.069; Ymax=0.1041;
Dchi=1e-4;Db=1; Dw=80; Dh=1800;
chi=1;  % \chi value assigned manually
l=1;   % fictitious spatial scale param.
par=[pp; Lam0; Ga; A; R; L0; f; Q; Kmin; Kmax; Mmin; Mmax; Ymin; Ymax; Db;
    Dw; Dh; chi;l]; % params vector
b0=5.25;w0=3.15;h0=pp*(Yi*b0+Q)/(A*(Yi*b0+f*Q)); % initial homog. sol.
    guess
dir0='aaa';dir=char([dir0 '/hom']); % principal and homog. sol directories
p=bwhinit(lx,nx,par,b0,w0,h0,dir0); % p-struct init

%% homog. solution continuation
p.plot.bpcmp=1; % component plotted during continuation
p.nc.eigref=-0.3;p.nc.neig=3; % eigenval. reference and number of eigenvals
    .
p.sol.ds=-0.01; p.nc.ilam=1; % initial cont. step and 1 param. cont.
p=cont(p,10); % continuation

%% cont. from Turing using cswibra
aux=[]; aux.m=3; aux.besw=0; p0=cswibra(dir,'bpt1',aux); p0.sw.bifcheck=2;
p0.nc.neig=5; p0.nc.eigref=-0.5; p0.sw.spcalc=1;
p0.nc.dsmin=1e-6; p0.nc.dsmax=5; p.nc.tol=1e-11;

%% T1
dirT=char([dir0 '/T1']);p=gentau(p0,[1]); % 6
p=setfn(p,dirT);p.sol.ds=1e-3;p=cont(p,10);

%% T2
dirT=char([dir0 '/T2']);p=gentau(p0,[0 1]); % 6
p=setfn(p,dirT);p.sol.ds=1e-3;p=cont(p,10);


%%
plotbra(char([dir0 '/hom']),'cl','b','tyun','--','cmp',1)
plotbra(char([dir0 '/T1']),'cl','b','tyun','--','cmp',1)
plotbra(char([dir0 '/T2']),'cl','b','tyun','--','cmp',1)
```

Listing 12: `c2Dm/bwhsingle/cmds1.m`

he previous commands produce the following results,

```
    >> cmds1
Problem directory name: s1/hom
step   lambda      y-axis  residual  iter meth   ds       #-EV b(0)
   0  300.00000    5.23399 7.35e-09    2  nat  0.000e+00   0   49.65399
   1  299.90000    5.23124 8.80e-12    2  nat   -0.10000   0   49.62790
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  59  270.89969    4.34205 6.44e-12    2  nat   -0.50000   0   41.19233
  60  270.39968    4.32456 7.37e-12    2  nat   -0.50000   0   41.02642
   1 possible bifurcation between 270.4 and 269.9, om=0
 mu_r=-8.52986e-07, mu_i=0
<phi,psi>=6.71159e-10,BP
  61  2.70372e+02 (BP, saved to s1/hom/bpt1.mat) bisection steps 10, last ds 0.00024414
  62  269.89968    4.30697 7.80e-12    2  nat   -0.50000   3   40.85954
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  79  260.89966    3.96985 7.42e-12    2  nat   -1.00000   3   37.66128
  80  259.89966    3.92952 8.41e-12    2  nat   -1.00000   3   37.27868
Timing: total=5.30086, av.step=0.0302019, av.Newton=0.00210061, av.spcalc=0.00616449
```

First Turing branch

```
    lam=270.3719; smallest eigenvalues: -8.53e-07    0.0161    0.0251
using m=3
Problem directory name: s1/T1
creating directory s1/T1
step   lambda      y-axis  residual  iter meth   ds       #-EV b(0)
   1 possible bifurcation between 270.372 and 271.193, om=0
 mu_r=3.70903e-05, mu_i=0
<phi,psi>=-3.14952e-10,BP
   1  2.70495e+02 (BP, saved to s1/T1/bpt1.mat) bisection steps 10, last ds 2.44141e-05
   2  271.19298    4.34117 7.47e-12    3  arc    0.05000   3   41.25940
   3  271.24295    4.34223 8.68e-12    2  nat    0.05000   3   41.27410
   4  271.34291    4.34436 7.08e-12    2  nat    0.10000   3   41.30347
   1 possible bifurcation between 271.343 and 271.543, om=0
 mu_r=-2.80281e-05, mu_i=0
<phi,psi>=-3.26133e-12,BP
   5  2.71347e+02 (BP, saved to s1/T1/bpt2.mat) bisection steps 10, last ds -9.76563e-0
   also saved to s1/T1/pt5.mat
   6  271.54284    4.34862 6.09e-11    2  nat    0.20000   4   41.36214
   7  271.74277    4.35288 1.86e-11    2  nat    0.20000   4   41.42072
   1 possible bifurcation between 271.743 and 272.143, om=0
 mu_r=5.64688e-05, mu_i=0
<phi,psi>=1.48108e-07,BP
   8  2.71887e+02 (BP, saved to s1/T1/bpt3.mat) bisection steps 10, last ds 0.00019531
   9  272.14267    4.36136 1.20e-09    2  nat    0.40000   5   41.53761
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 106  341.34391    5.46895 1.10e-11    3  nat    0.02500   5   58.73031
   1 possible bifurcation between 341.344 and 341.294, om=0
 mu_r=1.05115e-05, mu_i=0
<phi,psi>=4.9079e-10,BP
 107  3.41316e+02 (BP, saved to s1/T1/bpt4.mat) bisection steps 10, last ds -2.44141e-0
 108  341.29403    5.46407 1.13e-11    3  nat    0.05000   0   58.73642
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 150  310.14468    4.58636 1.20e-11    2  nat    0.80000   0   53.91918
Timing: total=8.83578, av.step=0.0318982, av.Newton=0.00333239, av.spcalc=0.00495385
```

Second Turing branch

```
    Problem directory name: s1/T2
creating directory s1/T2
step   lambda      y-axis  residual  iter meth   ds       #-EV b(0)
   1 possible bifurcation between 270.372 and 270.903, om=0
mu_r=-0.0123881, mu_i=0, no convergence
   1  270.90260    4.33107 1.10e-09    2  arc    0.05000   3   41.16361
   1 possible bifurcation between 270.903 and 270.953, om=0
mu_r=0.0280418, mu_i=0, no convergence
   2  270.95257    4.33207 7.47e-12    2  nat    0.05000   4   41.17817
   3  271.05252    4.33406 7.59e-12    2  nat    0.10000   4   41.20726
   4  271.25243    4.33805 1.19e-10    2  nat    0.20000   4   41.26538
   1 possible bifurcation between 271.252 and 271.652, om=0
 mu_r=-6.60466e-05, mu_i=0
<phi,psi>=4.38396e-07,BP
   5  2.71272e+02 (BP, saved to s1/T2/bpt1.mat) bisection steps 10, last ds -0.00019531
   also saved to s1/T2/pt5.mat
   6  271.65228    4.34602 5.34e-09    2  nat    0.40000   5   41.38138
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 110  338.57778    5.34937 1.33e-11    3  nat    0.10000   5   58.10432
   1 possible bifurcation between 338.578 and 338.378, om=0
 mu_r=-5.69923e-06, mu_i=0
<phi,psi>=-1.26009e-08,BP
 111  3.38525e+02 (BP, saved to s1/T2/bpt2.mat) bisection steps 10, last ds 9.76563e-05
 112  338.37789    5.33730 1.11e-11    3  nat    0.20000   0   58.09821
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 149  309.77825    4.53834 1.08e-11    2  nat    0.80000   0   53.60794
 150  308.97824    4.51805 1.08e-11    2  nat    0.80000   0   53.46958
Timing: total=9.31093, av.step=0.0361994, av.Newton=0.00311805, av.spcalc=0.00558257
```

We plot the results in Figure (3)fig:min_bif_ss.

## 3.2 Busse-Balloon: Bifurcation point continuation. sec:BBs

Important special points of solutions branches of (3)eq:SSM are such fold points (FPs), branch points (BPs), and Hopf points (HPs), and a useful feature of a numerical continuation and bifurcation

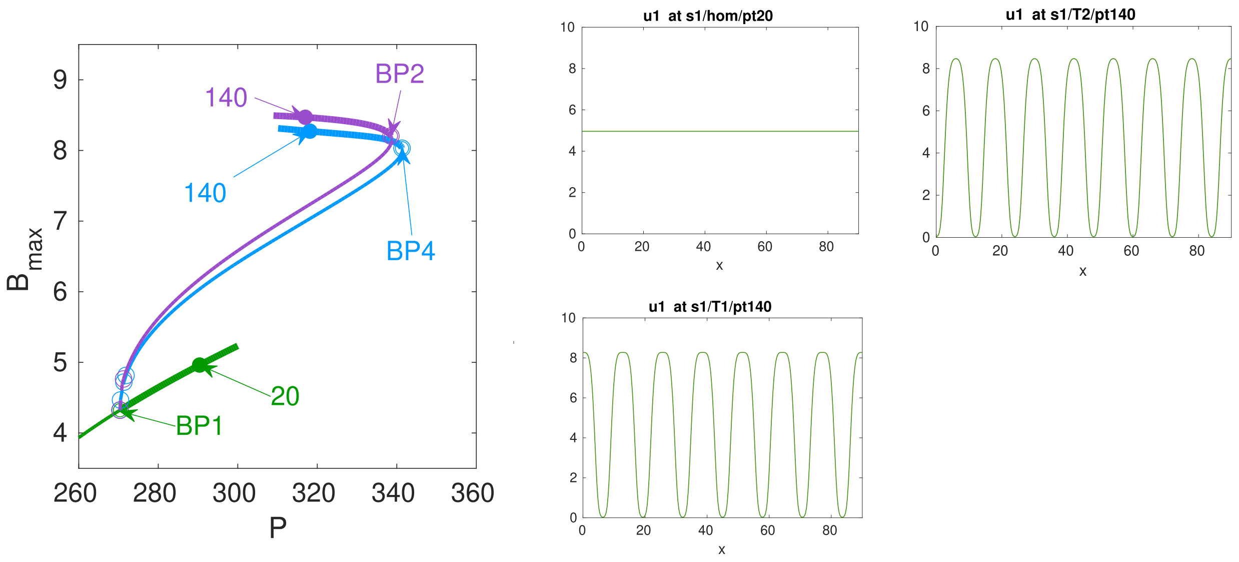

Figure 3: A minimal bifurcation diagram depicting three states by their respective numbers. Bifurcation points are labeled by BP.
fig:min_bif_ss

package is the option of FP-, BP-, and HP-continuation, fow which one hast to free an additional parameter. Besides being important themselves due to topological changes occurring there (and new branches bifurcating at BPs and HPs), FPs, BPs of HPs for instance also often delimit stability regions of solutions branches, and hence FPC, BPC and HPC can be used to efficiently compute such stability regions in dependence of a second parameter. For detailed explanation and background, see references [Uec21, Uec23b] uecker2021numerical,uecker2023continuation

The numerical continuation of the bifurcation points

$$H(U)=\begin{pmatrix} G(u,\lambda)+\mu M_d\psi \\ G_u^T(u,w)\Psi \\ \parallel\psi\parallel_2^2-1 \\ \langle\psi,G_\lambda(u,w)\rangle \end{pmatrix}=\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\quad U=(u,\psi,w), \tag{4}$$

where $(u,\lambda)$ is a simple BP (for the continuation in $\lambda$), $\psi$ is an adjoint kernel vector, and $w=(\lambda,\mu)$ with $w_1=\lambda$ the primary active parameter and $w_2=\mu$ as additional active parameter.

The main task then is to set up $\partial_u(G_u^T\psi)$ for the Jacobian of $H$, e.i.,

$$J_H=\begin{pmatrix} G_u & \mu\mathcal{M} & G_\lambda & \mathcal{M}\psi \\ \partial_u(G_u^T\psi) & G_u^T & \partial_{\lambda(}G_u^T\psi) & 0 \\ 0 & 2\psi^T & 0 & 0 \\ \psi^T\partial_\lambda G_u^T & G_\lambda^T & \psi^T\partial_\lambda G_\lambda & 0 \end{pmatrix} \tag{5}$$

As (3) eq:SSM is a semilinear problem, we can proceed similarly to the examples given in [Uec23a] p2phome by implementing $\partial_u(G_u^T\psi)$ explicitly

$$\partial_u(G_u^T\psi) = \partial_u \begin{pmatrix} f_{1,u_1}\psi_1 + f_{2,u_1}\psi_2 \\ f_{2,u_1}\psi_1 + f_{2,u_2}\psi_2 \end{pmatrix} \mathcal{M}^T \tag{6}$$

$$= \begin{pmatrix} f_{1,u_1u_1}\psi_1 + f_{2,u_1u_1}\psi_2 & f_{1,u_1u_2}\psi_1 + f_{2,u_1u_2}\psi_2 \\ f_{1,u_1u_2}\psi_1 + f_{2,u_2u_1}\psi_2 & f_{1,u_2u_2}\psi_1 + f_{2,u_2u_2}\psi_2 \end{pmatrix} \mathcal{M}^T \tag{7}$$

through `bpjac.m`, given explicitly as follows.

```
function duGuph=bpjac(p,u) % second derivative for BP continuation
n=p.np; b=u(1:n);w=u(n+1:2*n);h=u(2*n+1:3*n);par=u(p.nu+1:end);
pp=par(1); Lam0=par(2); Ga=par(3); A=par(4); R=par(5); L0=par(6);
f=par(7); Q=par(8); Kmin=par(9); Kmax=par(10); Mmin=par(11); Mmax=par(12);
Ymin=par(13); Ymax=par(14); Db=par(15); Dw=par(16); Dh=par(17); chi=par(18)
    ;
ov=ones(n,1);Yi=Ymax+chi*(Ymin-Ymax);Ki=Kmax+chi*(Kmin-Kmax);Mi=Mmax+chi*(
    Mmin-Mmax);

f1bb=-2*(Lam0*w*Ki^2)./(b+Ki).^3;
f1wb=(Lam0*Ki^2)*ov./(b+Ki).^2;
f2bb=-2*Lam0*R^2*w./(1+R*b).^3 + 2*A*Q*Yi^2*(-1+f)*h./(Q+Yi*b).^3;
f2bw=-ov*Ga + L0*R*ov./(1+R*b).^2;
f2bh=-A*Q*Yi*(-1+f)*ov./(Q+Yi*b).^2;
f3bb=-2*A*(-1+f)*Q*Yi^2*h./(Q+Yi*b).^3;
f3bh=A*(-1+f)*Q*Yi*ov./(Q+Yi*b).^2;

ph1=u(p.nu+1:p.nu+p.np); ph2=u(p.nu+p.np+1:p.nu+2*p.np);ph3=u(p.nu+2*p.np
    +1:p.nu+3*p.np);

M1=spdiags(f1bb.*ph1+f2bb.*ph2+f3bb.*ph3,0,n,n);
M2=spdiags(f1wb.*ph1+f2bw.*ph2,0,n,n);
M3=spdiags(f2bh.*ph2+f3bh.*ph3,0,n,n);

duGuph = - [[M1 M2 M3];[M2 0*M2 0*M2];[M3 0*M3 0*M3]]*p.mat.M;
```

Listing 13: `c2Dm/bwhsingle/bpjac.m`. This function provides the component $\partial_u(G_u\phi)$ used for bifurcation-point continuation.

Once $\partial_u(G_u^T\psi)$ is implemented, we use the following minimal commands to produce the first step in the Busse-Balloon calculation

```
close all; keep p2phome; % script for 1D
%% continuation of the subcritical BP for T1 up
p=bpcontini('s1/T1','bpt4',19,'bps1a',3e-3); % bif. point. cont
p.plot.bpcmp=5; p.sw.spjac=0; p.nc.dsmax=0.1;p.nc.lammin=0;p.nc.del=4e-3;
p.nc.dsmin=1e-5; p.fuha.spjac=@bpjac; p.sw.spcalc=0; p.sw.bifcheck=0;
p.nc.almine=0.4;p.nc.tol=9e-9; p.file.smod=10;
huclean(p); p=cont(p,20);
%% continuation in the reverse direction
p=loadp('bps1a','pt0');p.sol.ds=-3*p.sol.ds;p=setfn(p,'bps1b');p=cont(p,20)
    ;
%% branch point continuation data
p0=loadpp('bps1a'); pp0=p0.branch(11,:);l0=p0.branch(4,:);
p1=loadpp('bps1b'); pp1=p1.branch(11,:);l1=p1.branch(4,:);

%% plot of branch point cont. results
% wl=lx/n; k=2*pi/wl; our case wl=90/7;k=2*pi/wl=0.48869
```

```

hold on
plot(pp0,0.48869*l0,'LineWidth',2,'Color','b');
plot(pp1,0.48869*l1,'LineWidth',2,'Color','b');
ylim([0,0.85])
xlabel('P'); ylabel('k'); title('Busee-Balloon Single Species','Interpreter
    ','latex');
set(gca,'fontsize',14);
```

Listing 14: `c2Dm/bwhsingle/cmds2.m`

Producing the following results for the first continuation (`p.sol.ds>0`)

```
Warning: Computation may be slow as some pde derivatives are computed numerically.
Approx. zero eigenvalue=1.05115e-05.
New active parameters and their values:
aux vars of p (is point of type 1)
p.nc.ilam  values
  19      1
  1     341.316
  20      1.05115e-05
Problem directory name: bps1a
creating directory bps1a
BP continuation. Use p=bpcontexit(p) to return to normal continuation.
step   lambda      y-axis  residual  iter meth   ds       b(0)
   0    1.00000  341.31572 3.87e-09    1  nat  0.000e+00   58.73440
   1    1.00030  341.30625 1.15e-09    2  nat    0.00030   58.73226
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  19    1.60194  269.85979 3.47e-09    5  arc    0.02400   41.66268
  20    1.60778  265.87779 8.62e-10    4  arc    0.02400   40.66906
Timing: total=16.0294, av.step=0.747914, av.Newton=0.730686, av.spcalc=0
```

Results for the reverse direction (`p.sol.ds<0`) we obtain,

```
Problem directory name: bps1b
creating directory bps1b
BP continuation. Use p=bpcontexit(p) to return to normal continuation.
step   lambda      y-axis  residual  iter meth   ds       b(0)
   0    1.00000  341.31572 3.87e-09    0  nat  0.000e+00   58.73440
   1    0.99910  341.34404 3.05e-09    2  nat   -0.00090   58.74081
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  19    0.70554  344.21361 5.88e-12    5  nat   -0.01800   59.49595
  20    0.70553  344.21359 5.80e-09   10  arc   -0.00001   59.49595
Timing: total=33.7603, av.step=1.64145, av.Newton=1.62148, av.spcalc=0
```

Producing the following plot (see Figure (4)) fig:busse_ball_ss

## References

rademacher2018oopde

[RU18] Jens DM Rademacher and Hannes Uecker. The oopde setting of pde2path–a tutorial via some allen-cahn models. 2018.

uecker2019pattern

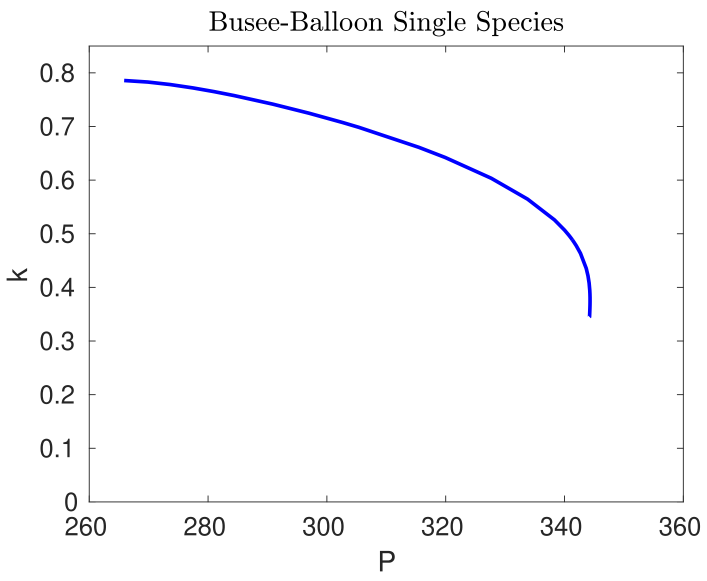

Figure 4: Few of bifurcation point continuation results from BP4 at the folder s1/T1. fig:busse_ball_ss

[Uec19] Hannes Uecker. Pattern formation with pde2path–a tutorial. *arXiv preprint arXiv:1908.05211*, 2019. uecker2021numerical

[Uec21] Hannes Uecker. *Numerical continuation and bifurcation in Nonlinear PDEs*. SIAM, 2021. p2phome

[Uec23a] H. Uecker. pde2path – a matlab package for continuation and bifurcation in systems of pdes, v3.1, 2023. uecker2023continuation

[Uec23b] Hannes Uecker. Continuation of fold points, branch points, and hopf points with constraints in pde2path. 2023.