Nonlinear Systems

ME 3295-001/ME 5895-001/ECE 6095-004

Chang Liu

School of Mechanical, Aerospace, and Manufacturing Engineering

University of Connecticut

chang_liu@uconn.edu

Numerically searching for Lyapunov functions and bifurcation analysis

Numerically searching for Lyapunov functions

Region of attraction

- How to tune P and Q matrice to enlarge region of attraction estimation?
- How to work on high-order systems?
- How to use high-order polynomials as Lyapunov functions?
- If all techniques are limited to 2D ODE toy model...?

Numerically searching for Lyapunov functions

Course example code

https://github.com/cliu124/Nonlinear_Systems

Syllabus

Enabled: Statistics Tracking

Attached Files: Syllabus Nonlinear Systems Fall 2024 ME 3295-001 5895-001.pdf (197.207 KB)

Update 09/17/2024: Add the link to course example code on https://github.com/cliu124/Nonlinear_Systems

cliu124 Add ReadMe — d570bae · 2 hours ago — 12 Commits

1_Lyapunov_method_YALMIP	Update pde2path	15 hours ago
2_Bifurcation_pde2path	Update pde2path	3 hours ago
.gitattributes	Initial commit	last week
ReadMe.md	Add ReadMe	2 hours ago
install.m	Update reorganize	2 days ago

Installation of YALMIP, SeDuMi, and pde2path

HW 3 will use some of these code
Can be used for course project

Numerically searching for Lyapunov functions

ReadMe

This repository is example code for Nonlinear Systems ME 3295-001/ME 5895-001/ECE 6095-004 at University of Connecticut, taught by Dr. Chang Liu (https://changliulab.engineering.uconn.edu/).

This provide examples to use YALMIP (https://yalmip.github.io/) for searching Lyapunov function and pde2path (https://www.staff.uni-oldenburg.de/hannes.uecker/pde2path/) to conduct bifurcation analysis.

These softwares have been already downloaded and unzipped.

Running install.m will finish the installation of YALMIP, SeDuMi and pde2path.

For YALMIP, this repository already has SeDuMi (https://sedumi.ie.lehigh.edu/?page_id=58) as semi-definite programming solver.

The Mosek (https://www.mosek.com/) solver is required for A_growth_rate.m, C_region_of_attraction.m, D_sum_of_squares.m, E_growth_rate_time_varying.m. This is NOT contained here and it needs to install Mosek following the instructions in the link (https://docs.mosek.com/10.2/install/installation.html).

Using edu email can get a free academic license of Mosek.

Numerically searching for Lyapunov functions

Mosek

4.2.3 Windows, MSI installer

1. Make the right choice between the 32bit and 64bit versions. In most cases it is recommend to use the 64bit version.
2. Download the Windows 32bit x86 or Windows 64bit x86 **MOSEK** Optimization Suite MSI installer from https://mosek.com/downloads/.
3. Run the installer to complete the installation.
4. Check that the path

```
<MSKHOME>\mosek\10.2\tools\platform\<PLATFORM>\bin
```

was added to the OS variable PATH, where <MSKHOME> is the directory where **MOSEK** was installed and <PLATFORM> is win64x86 or win32x86 depending on the version of **MOSEK** installed. This is necessary for Windows to locate the **MOSEK** shared libraries.

Academic free license of Mosek

This email contains a Personal Academic License file valid until **2024-dec-05** for the MOSEK Optimization Tools version 10. This license is backward compatible, meaning it supports both current and earlier versions of MOSEK.

Although your license expires on the date above, Mosek allows you to submit a request for a new license up to 30 days before this date.

The license file should be placed inside a folder called "mosek" under the user's home directory ($HOME/mosek/mosek.lic or %USERPROFILE%\mosek\mosek.lic). In most typical cases that will be:

```
/home/YOUR_USER_NAME/mosek/mosek.lic (Linux)
/Users/YOUR_USER_NAME/mosek/mosek.lic (OSX)
C:\Users\YOUR_USER_NAME\mosek\mosek.lic (Windows)
```

Where *YOUR_USER_NAME* is your user ID on the computer.

Numerically searching for Lyapunov functions

within Folder: 1_Lyapunov_method_YALMIP

SeDuMi_1_3	Update reorganize	2 days ago
YALMIP-master	Update reorganize	2 days ago
A_growth_rate.m	Update reorganize	2 days ago
B_transient_growth.m	Update reorganize	2 days ago
C_region_of_attraction.m	Update stable version.	20 hours ago
D_sum_of_squares.m	Update pde2path	15 hours ago
E_growth_rate_time_varying.m	Update SOS example	20 hours ago
F_transient_growth_time_varying.m	Update SOS example	20 hours ago
SeDuMi_1_3.zip	Update reorganize	2 days ago
YALMIP-master.zip	Update reorganize	2 days ago

Numerically searching for Lyapunov functions

Local bounds of nonlinearity

$$\dot{x}_1 = -x_2$$
$$\dot{x}_2 = x_1 + \left(x_1^2 - 1\right) x_2$$

$$\dot{x} = Ax + g(x), \text{ where } A = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix}, \; g(x) = \begin{bmatrix} 0 \\ x_1^2 x_2 \end{bmatrix}$$

Further write it into $\dot{x} = Ax + Bu$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and $u = x_1^2 x_2$

$|u| \leq |x_1^2 x_2| \leq \delta^2 |x_2|$, when $|x| \leq \delta$, not unique, depends on your choice.

$|u| \leq \delta^2 K \begin{bmatrix} |x_1| \\ |x_2| \end{bmatrix}$ if $|x| \leq \delta$, where $K = \begin{bmatrix} 0 & 1 \end{bmatrix}$

Or in quadratic form: $u^2 \leq \delta^4 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T K^T K \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Numerically searching for Lyapunov functions

s-procedure

$$\dot{x} = Ax + Bu$$

$$V = x^T P x,$$

$$\dot{V} = \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} < 0$$

Not feasible

$$\dot{V} = \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + s \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} \delta^4 K^T K & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} < 0$$

$s \geq 0$ + within region $|x| \leq \delta$

+

(Lagragian multiplier)

$$\text{Thus, } \dot{V} < 0 \text{ within } |x| \leq \delta$$

Numerically searching for Lyapunov functions

Numerical tool: YALMIP

```
%We try to find Lyapunov function V=x^T*P*x, and let \dot{V}\leq 0 within a
%local region |x|^2\leq \delta^2 using s-procedure, such that
%\dot{V}+s*(\delta^4* (K*x)^T*(K*x) -u^2)\leq 0
%As normal, we require V is positive definite, and without loss of
%generality, we can require P>=I.



A=[0,-1;
    1,-1];

B=[0;1]; % write nonlinear system as dx=A*x+B*u;

%-----------------------bound of forcing term
K1=[0,1]; % bound this forcing term such that u^2\leq \delta^4* (K*x)^T*(K*x)

I=eye(2,2); %identify matrix
```

C_region_of_attraction.m

Numerically searching for Lyapunov functions

Numerical tool: YALMIP

```
%-----------------------formulate linear matrix inequalities
%define variables to be optimized
P=sdpvar(2,2); %weighting matrix for Lyapunov function
s=sdpvar(1,1); %s is a non-negative value to enforce that dV is negative semidefi
delta4=sdpvar(1,1); %delta^4 going to be optimized

dV_constraint=[A'*P+P*A+s*delta4*K1'*K1, P*B;
    B'*P, -s];
constraint=[P>=I, dV_constraint<=0, s>=0];

sdp_option=sdpsettings('solver','sedumi'); %This solver can be modified as mosek

bisection(constraint, -delta4, sdp_option); %negative sign means maximize delta4.

delta=value(delta4)^(1/4); %get delta such that |x|<=\delta
```

C_region_of_attraction.m

Numerically searching for Lyapunov functions

Numerical tool: YALMIP

```
%get V function as a symbolic function
x_sym=sym('x',[2,1]);
V_poly=simplify(transpose(x_sym)*value(P)*x_sym);%construct the V function
V_fun=matlabFunction(V_poly);%convert V_poly to a MATLAB function.

%get beta=min V within |x|=\delta.
theta_list=linspace(0,2*pi,200);
for theta_ind=1:length(theta_list)
    theta=theta_list(theta_ind);
    x_delta=delta*[cos(theta);sin(theta)];
    V_val_delta(theta_ind)=V_fun(x_delta(1),x_delta(2));
end
[beta,theta_ind]=min(V_val_delta);

%evaluate V contour for x\in [-3,3] and y\in [-3,3];
x=linspace(-3,3,1000);
y=linspace(-3,3,1000);
[X,Y]=meshgrid(x,y);
V_val=V_fun(X,Y);
V_val(find(V_val>beta))=NaN;% Only consider V\leq \beta
V_val(find(X.^2+Y.^2>value(delta^2)))=NaN; %only consider V that is within
pcolor(x,y,V_val); shading interp;
```

C_region_of_attraction.m

Numerically searching for Lyapunov functions

Numerical tool: YALMIP

```
%--------------------
%plot the trajectories from inverse time van der pol oscillator
theta_list=linspace(0,2*pi,30);
for theta_ind=1:length(theta_list)
    theta=theta_list(theta_ind);
    x0=0.001*[cos(theta);sin(theta)];%initial conditions.
    [t,z]=ode45(@(t,z) [z(2); -z(1)-(z(1)^2-1)*z(2)],[0,20],x0);
    plot(z(:,1),z(:,2),'k','LineWidth',1); hold on;
end
xlabel('x_1'); ylabel('x_2');
```

C_region_of_attraction.m

Numerically searching for Lyapunov functions

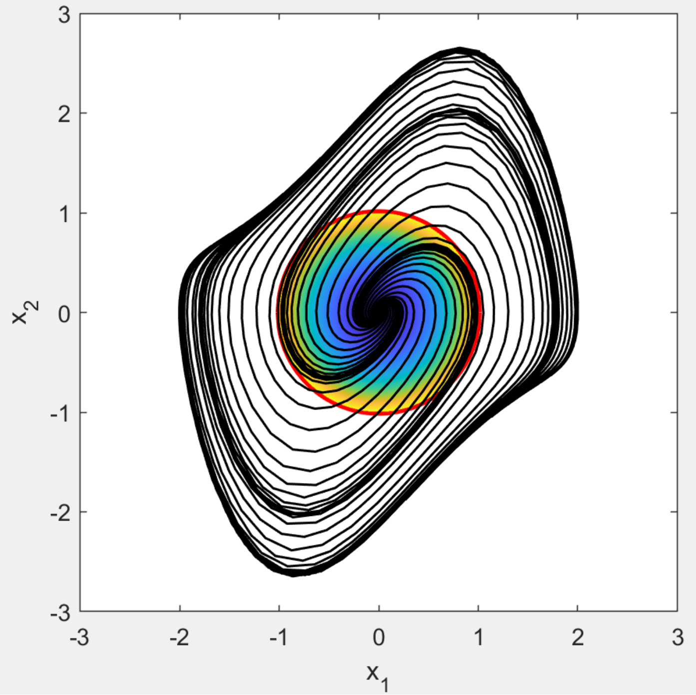
Region of attraction

Running C_region_of_attraction.m

Linear matrix inequalities formulation with bounds on cubic nonlinearity

$$\dot{V} <= 0 \text{ for } |x| \leq 1$$

$$V = 1.8825(x_1^2 + x_2^2)$$



Numerically searching for Lyapunov functions

Sum of squares programming

$$V = x^T P x,\ P = I,\ \text{then } V = x_1^2 + x_2^2 \text{ is sum of squares.}$$

```
x = sdpvar(1,1);y = sdpvar(1,1);
p = (1+x)^4 + (1-y)^2;
F = sos(p);
solvesos(F);
```

How about higher-order polynomials

The sum-of-squares decomposition is extracted with the command sosd.

```
h = sosd(F);
sdisplay(h)

ans =
  '-1.203-1.9465x+0.22975y-0.97325x^2'
  '0.7435-0.45951x-0.97325y-0.22975x^2'
  '0.0010977+0.00036589x+0.0010977y-0.0018294x^2'
```

https://yalmip.github.io/tutorial/sumofsquaresprogramming/

Numerically searching for Lyapunov functions

Sum of squares programming

```
x = sdpvar(1,1);y = sdpvar(1,1);
p = (1+x)^4 + (1-y)^2;
```

Introduce a decomposition $p = v^T Q v$

$$v = \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix}$$

```
v = monolist([x y],degree(p)/2);
Q = sdpvar(length(v));
p_sos = v'*Q*v;
```

The polynomials have to match, hence all coefficient in the polynomial describing the difference of the two polynomials have to be zero. With that and the crcual constraint that the matrix used in the decomposition is positive semi-definite, we have defined the full sos-problem and can solve it.

```
F = [coefficients(p-p_sos,[x y]) == 0, Q >= 0];
optimize(F)
```

Numerically searching for Lyapunov functions

Sum of squares programming

```
degree=4; %polynomial degree of V function

x=sdpvar(2,1);
m = monolist(x,degree/2); %define monomials

P = sdpvar(length(m)-1); %V=m'*P*m, and we only consider the second
V_poly=m(2:end)'*P*m(2:end);

R = sdpvar(length(m)); %This is a SOS multiplier.
R_poly=m'*R*m;

I = eye(length(m)-1); %identity matrix.

delta2=sdpvar(1,1); %\dot{V} is negative within |x|^2\leq \delta^2.

f=([-x(2);x(1)+(x(1)^2-1)*x(2)]);%RHS of nonlinear term.

dV=jacobian(V_poly, x)*f;%compute dV/dt.
```

D_sum_of_squares.m

Numerically searching for Lyapunov functions

Sum of squares programming

```
    m_p = monolist([x],(degree+2)/2);
    Q_V=sdpvar(length(m_p));
    Q_dV = sdpvar(length(m_p));

    constraint=[P>=I,...%V is positive definite
        coefficients(dV + (delta2-x'*x) * R_poly -m_p'*Q_dV*m_p,m_p)==0, ...
        Q_dV<=0,... %dV is negative definite within a local region |x|^2\leq delta2.
        R>=0 %make sure R_poly is non-negative multiplier.
        ]

    sol=bisection(constraint, -delta2, sdp_option); %negative sign means maximize delta4.
```

Then conduct the same thing to get the largest level set of Lyapunov function run simulations with reversed time

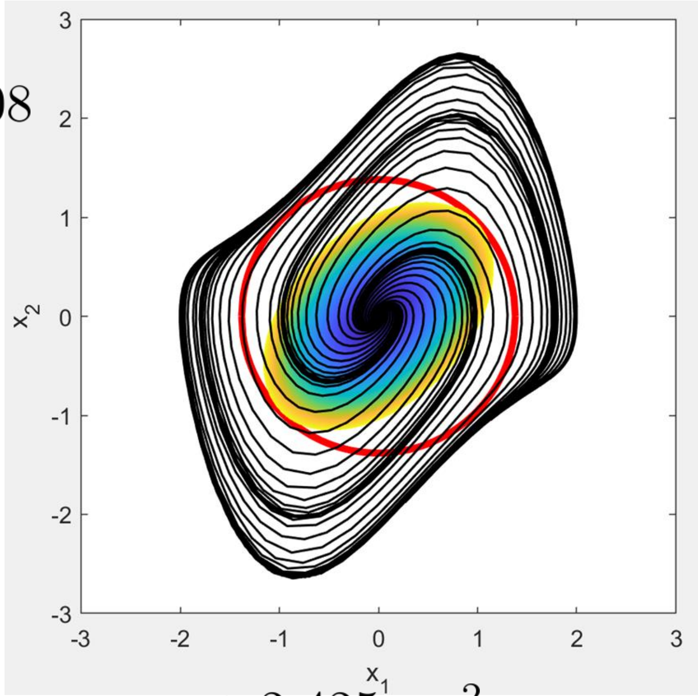
D_sum_of_squares.m

Numerically searching for Lyapunov functions

Region of attraction

Running D_sum_of_squares.m with V polynomial of order 2

$$\dot{V} <= 0 \text{ for } |x| \leq 1.3808$$



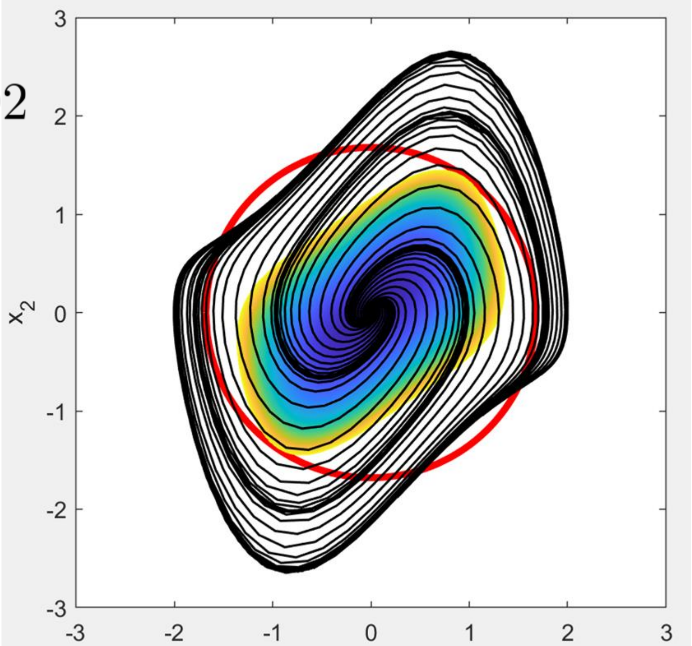
$$V = 2.386 * x_1^2 - 1.994 * x_1 * x_2 + 2.425 * x_2^2$$

Numerically searching for Lyapunov functions

Region of attraction

Running D_sum_of_squares.m with V polynomial of order 4

$$\dot{V} <= 0 \text{ for } |x| \leq 1.6792$$

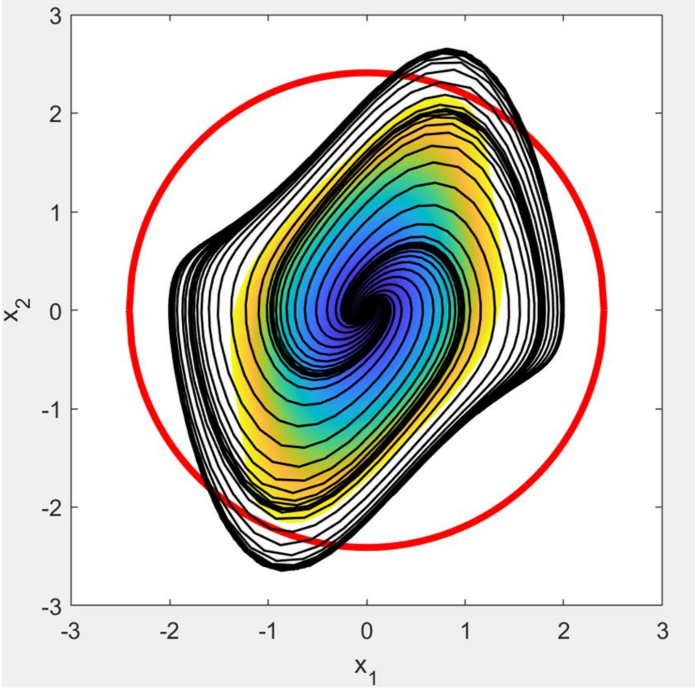


$$V = 1.43\times10^6 x_1^4 + 8.27\times10^6 x_1^3 x_2 + 2.18\times10^{-13} x_1^3 + 8.47\times10^6 x_1^2 x_2^2 + 1.04\times10^{-12} x_1^2 x_2 + 1.67\times10^7 x_1^2 - 9.1\times10^6 x_1 x_2^3 - 1.51\times10^{-12} x_1 x_2^2 - 2.78\times10^7 x_1 x_2 + 4.96\times10^6 x_2^4 - 2.8\times10^{-13} x_2^3 + 1.81\times10^7 x_2^2$$

Numerically searching for Lyapunov functions

Region of attraction

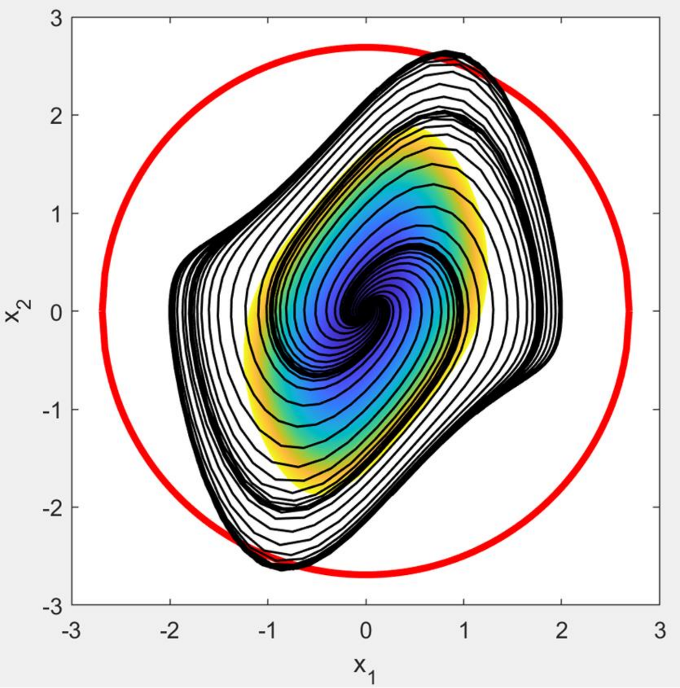
Running D_sum_of_squares.m with V polynomial of order 6



Numerically searching for Lyapunov functions

Region of attraction

Running D_sum_of_squares.m with V polynomial of order 8



Numerically searching for Lyapunov functions

Upper bound of growth rate

Program A_growth_rate.m

$$\dot{x} = Ax$$

$$\min \lambda, \text{ s.t.}$$
$$I \preceq P, \; A^T P + PA \preceq 2\lambda P$$

$$\dot{V} \leq 2\lambda V$$

λ is the upper bound of growth rate, i.e., $\|x(t)\| \leq Ce^{\lambda t}\|x(0)\|$

```
lambda=sdpvar(1,1); %upper bound of growth rate to be optimized
I=eye(2,2); %identity matrix
P=sdpvar(2,2);
constraint=[P>=I,P*A+A'*P<=2*lambda*P];

objective=lambda; %set objective of optimization as lambda. In default, it
sdp_option=sdpsettings('solver','sedumi'); %This solver can be modified as
results=bisection(constraint,objective,sdp_option); %call YALMIP function
```

Numerically searching for Lyapunov functions

Upper bound of growth rate

Program A_growth_rate.m

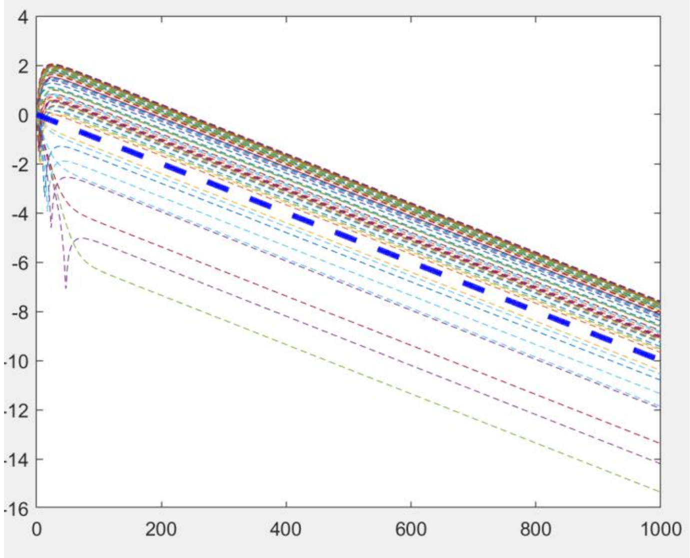
$$\dot{x} = Ax$$

$$\min\ \lambda,\ \text{s.t.}$$
$$I \preceq P,\ A^T P + PA \preceq 2\lambda P$$

$$\dot{V} \le 2\lambda V$$

λ is the upper bound of growth rate, i.e., $\|x(t)\| \le Ce^{\lambda t}\|x(0)\|$

Blue thick dashed line: $Ce^{\lambda t}\|x(0)\|$
Other lines: simulation with random I.C.



Numerically searching for Lyapunov functions

Upper bound of transient growth

Program B_transient_growth.m

$$\dot{x} = Ax$$

$$\dot{V} \leq 0$$

$$\min \gamma, \text{ s.t.}$$
$$I \preceq P \preceq \gamma I, \; A^T P + PA \preceq 0$$

γ is the upper bound of trasient growth, i.e., $G(t) = \frac{\|x(t)\|_2^2}{\|x(0)\|_2^2} \leq \gamma$

```
gamma=sdpvar(1,1); %upper bound of transient growth to be optimi
I=eye(2,2); %identity matrix
P=sdpvar(2,2);
constraint=[I<=P<=gamma*I,P*A+A'*P<=0]; %constraint: I<=P<=gamma
objective=gamma; %set objective of optimization as gamma. In def
sdp_option=sdpsettings('solver','sedumi'); %This solver can be m
results=optimize(constraint,objective,sdp_option); %call YALMIP
```

Numerically searching for Lyapunov functions

Upper bound of transient growth

Program B_transient_growth.m

$$\dot{x} = Ax$$

$$\min \gamma, \text{ s.t.}$$
$$I \preceq P \preceq \gamma I, \; A^T P + PA \preceq 0$$

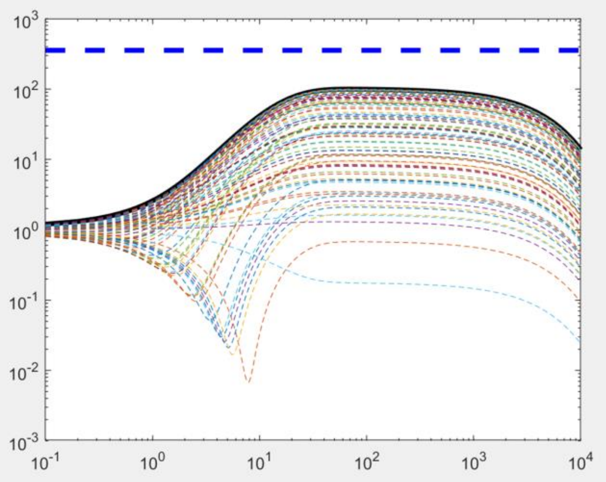
$$\dot{V} \leq 0$$

$$\gamma \text{ is the upper bound of transient growth, i.e., } G(t) = \frac{\|x(t)\|_2^2}{\|x(0)\|_2^2} \leq \gamma$$

Blue thick dashed line: γ

Other lines: simulation with random I.C.

Black solid line: envelope of simulations



Numerically searching for Lyapunov functions

Upper bound of growth rate of time-varying systems

Program E_growth_rate_time_varying.m

$$\dot{x} = A(t)x$$
$$V = x^T P(t) x$$

$$\min \lambda, \text{ s.t.}$$
$$I \preceq P(t),\ A(t)^T P(t) + P(t)A(t) + \dot{P}(t) \preceq 2\lambda P(t)$$

$\dot{V} \leq 2\lambda V$ $\quad$ λ is the upper bound of growth rate, i.e., $\|x(t)\| \leq Ce^{\lambda t}\|x(0)\|$

```
lambda=sdpvar(1,1); %upper bound of growth rate to be optimized
I=eye(2,2); %identity matrix
constraint=[]; %constraint list.
for t_ind=1:sample_num% we only need to go through one period as this A is periodic.
    P{t_ind}=sdpvar(2,2); %a time-varying Lyapunov function
    constraint=[constraint,P{t_ind}>=I]; %add constraint that I<=P(t), for any t
end
P{sample_num+1}=P{1}; %enforce that P is also periodic. This is only suitable for periodic A(t)

for t_ind=1:sample_num
    dP=(P{t_ind+1}-P{t_ind})/(dt); %use finite difference to approximate  dP/dt
    constraint=[constraint,P{t_ind}*A(:,:,t_ind)+A(:,:,t_ind)'*P{t_ind}+dP<=2*lambda*P{t_ind}];
end

objective=lambda; %set objective of optimization as lambda. In default, it will minimize this o
sdp_option=sdpsettings('solver','sedumi'); %This solver can be modified as mosek (https://www.m
results=bisection(constraint,objective,sdp_option); %call YALMIP function optimize to solve thi
```

Numerically searching for Lyapunov functions

Upper bound of growth rate of time-varying systems

Program E_growth_rate_time_varying.m

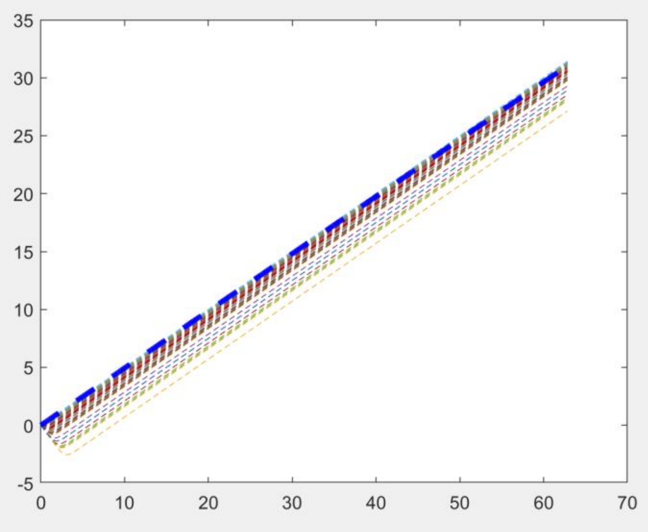
$$\dot{x} = A(t)x$$
$$V = x^T P(t) x$$

$$\min\ \lambda,\ \text{s.t.}$$
$$I \preceq P(t),\ A(t)^T P(t) + P(t)A(t) + \dot{P}(t) \preceq 2\lambda P(t)$$

$\dot{V} \leq 2\lambda V$ $\quad$ λ is the upper bound of growth rate, i.e., $\|x(t)\| \leq Ce^{\lambda t}\|x(0)\|$

Blue thick dashed line: $Ce^{\lambda t}\|x(0)\|$

Other lines: simulation with random I.C.



Numerically searching for Lyapunov functions

Upper bound of transient growth of time-varying systems

Program F_transient_growth_time_varying.m

$$\dot{x} = A(t)x$$
$$V = x^T P(t) x$$

$$\min \lambda, \text{ s.t.}$$
$$I \preceq P(t) \preceq \gamma I, \; A(t)^T P(t) + P(t)A(t) + \dot{P} \preceq 0$$

$\dot{V} \leq 0$ $\quad \gamma$ is the upper bound of transient growth, i.e., $G(t) = \frac{\|x(t)\|_2^2}{\|x(0)\|_2^2} \leq \gamma$

```
gamma=sdpvar(1,1); %upper bound of transient growth to be optimized
I=eye(2,2); %identity matrix
constraint=[]; %constraint list.
for t_ind=1:length(t_list)
    P{t_ind}=sdpvar(2,2); %a time-varying Lyapunov function
    constraint=[constraint,P{t_ind}>=I,P{t_ind}<=gamma*I]; %add constraint that
end

for t_ind=1:length(t_list)-1
    dP=(P{t_ind+1}-P{t_ind})/(dt); %use finite difference to approximate dP/dt
    constraint=[constraint,P{t_ind}*A(:,:,t_ind)+A(:,:,t_ind)'*P{t_ind}+dP<=0];
end

objective=gamma; %set objective of optimization as gamma. In default, it will mi
sdp_option=sdpsettings('solver','sedumi'); %This solver can be modified as mosek
results=optimize(constraint,objective,sdp_option); %call YALMIP function optimiz
```

Numerically searching for Lyapunov functions

Upper bound of transient growth of time-varying systems

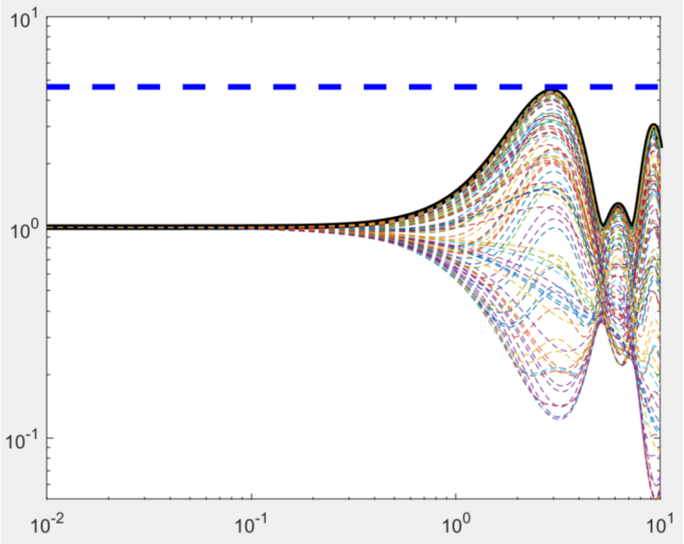
Program F_transient_growth_time_varying.m

$$\dot{x} = A(t)x$$
$$V = x^T P(t) x$$

$$\min \lambda, \text{ s.t.}$$
$$I \preceq P(t) \preceq \gamma I,\ A(t)^T P(t) + P(t)A(t) + \dot{P} \preceq 0$$

$$\dot{V} \leq 0$$

γ is the upper bound of transient growth, i.e., $G(t) = \frac{\|x(t)\|_2^2}{\|x(0)\|_2^2} \leq \gamma$



Blue thick dashed line: γ

Other lines: simulation with random I.C.

Black solid line: envelope of simulations

Bifurcation analysis

Numerical tool: pde2path

https://www.staff.uni-oldenburg.de/hannes.uecker/pde2path/

pde2path - a Matlab package for continuation and bifurcation in systems of PDEs, v3.1

Currently maintained by A. Meiners, J. Rademacher, and H. Uecker

Former developers include H. de Witt, T. Dohnal, and D. Wetzel

Home | Tutorials and Demos | Movies | Applications| Backlog

New version pde2path 3.1 (September 2023). Download: pde2path (software and demos) tar.gz or zip.
Latest updates:

- September 2023: some bugfixes, and new tutorial Continuation of fold points, branch points and Hopf points with constraints
- July 2023: version 3.1. First version of library Xcont and demos geomtut for Differential geometric bifurcation problems in pde2path
- June 2021: version 3.0, stable version, associated to the book *Numerical continuation and bifurcation in Nonlinear PDEs*, SIAM.
- January 2021: New tutorial pde2path without FEM, explaining mods to run pde2path on "general" right hand sides.
- September 2020: most of pde2path now also runs under octave. See README in the folder octave of the pde2path download. Otherwise: bug--fixes, and additional demos (chtor, schnakcone) for problems Pattern formation tutorial [Â§5 and 6].

For documentation, see the Quickstart guide and reference card and the Tutorials section. For a quick look, here are some movies. For older versions see the Backlog.

pde2path is currently maintained by: Alexander Meiners, Jens Rademacher and Hannes Uecker. Former co--developers: Hannes deWitt, Tomas Dohnal, and Daniel Wetzel.

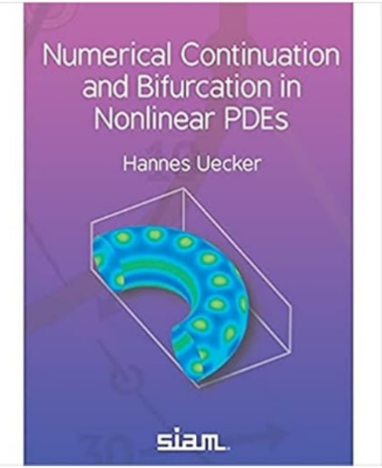
Many thanks to: Francesca Mazzia for TOM, to Uwe PrÃ¼fert for providing OOPDE, to Daniel Kressner for pqzschur, to Kristian Ejlebjaerg Jensen for trullekrul, to Alec Jacobson for the gptoolbox, and to all pe lend (public domain) code, some just small snippets, some large, see also "Licence" below.

For bugs, questions or remarks please write to: hannes.uecker -- at -- uol.de, and/or one of the other current maintainers. Any feedback is welcome.

Abstract. *pde2path* is a continuation/bifurcation package for systems of PDEs over bounded d-dimensional domains, d=1,2,3, including features such as nonlinear boundary conditions, cylinder and torus geomet boundary conditions), and a general interface for adding auxiliary equations like mass conservation or phase equations for continuation of traveling waves. The original version 1.0 was for elliptic systems in 2D a pdetoolbox, which since v2.3 has been more or less replaced by the free package OOPDE. Recent additions (v2.5 and v2.6) include the handling of multiple steady bifurcation points, Branch point continuation ar continuation via extended systems, continuation of relative equilibria (e.g., traveling waves and rotating waves), branch switching from periodic orbits (Hopf pitchfork/transcritical bifurcation, and period doublin instance on pattern formation on spheres and tori (Pattern formation tutorial, § 6), and on the computation of coefficients of amplitude equations for Turing bifurcations (ampsys tutorial, standalone version of amp

Bifurcation analysis

Numerical tool: pde2path



Click image to open expanded view

Numerical Continuation and Bifurcation in Nonlinear PDEs

by Hannes Uecker (Author)

See all formats and editions

Partial differential equations (PDEs) are the main tool to describe spatially and temporally extended systems in nature. PDEs usually come with parameters, and the study of the parameter dependence of their solutions is an important task. Letting one parameter vary typically yields a branch of solutions, and at special parameter values, new branches may bifurcate. Numerical Continuation and Bifurcation in Nonlinear PDEs • presents hands-on approach to numerical continuation and bifurcation for nonlinear PDEs, in 1D, 2D and 3D, • provides a concise but sound review of analytical background and numerical methods, • explains the use of the free MATLAB package pde2path via a large variety of examples with ready to use code, and • contains demo codes that can be easily adapted to the reader's given problem. This book will be of interest to applied mathematicians and scientists from physics, chemistry, biology, and economics interested in the numerical solution of nonlinear PDEs, particularly the parameter dependence of solutions. It is appropriate for the following courses: Advanced Numerical Analysis, Special Topics on Numerical Analysis, Topics on Data Science, Topics on Numerical Optimization, and Topics on Approximation Theory

^ Read less

Bifurcation analysis

Numerical tool: pde2path

https://github.com/cliu124/Nonlinear_Systems

Syllabus

Enabled: Statistics Tracking

Attached Files: Syllabus Nonlinear Systems Fall 2024 ME 3295-001 5895-001.pdf (197.207 KB)

Update 09/17/2024: Add the link to course example code on https://github.com/cliu124/Nonlinear_Systems

cliu124 Add ReadMe	d570bae · 2 hours ago	12 Commits
1_Lyapunov_method_YALMIP	Update pde2path	15 hours ago
2_Bifurcation_pde2path	Update pde2path	3 hours ago
.gitattributes	Initial commit	last week
ReadMe.md	Add ReadMe	2 hours ago
install.m	Update reorganize	2 days ago

Installation of YALMIP, SeDuMi, and pde2path

HW 3 will use some of these code
Can be used for course project

Bifurcation analysis

Numerical tool: pde2path

Within 2_Bifurcation_pde2path folder

1_supercritical_pitchfork	Update 20240917	last week
2_subcritical_pitchfork	Update 20240917	last week
3_supercritical_Hopf	Update 20240917	last week
4_subcritical_Hopf	Update 20240917	last week
pde2path	Update pde2path	last week
pde2path.zip	Update reorganize	last week

Bifurcation analysis

Numerical tool: pde2path

Within 1_supercritical_pitchfork folder

- cmds1.m
- init.m
- oosetfemops.m
- sG.m
- sGjac.m

Within 3_supercritical_Hopf folder

- bradat.m
- cmds1.m
- init.m
- nodalf.m
- oosetfemops.m
- sG.m
- sGjac.m

Bifurcation analysis

Numerical tool: pde2path

Within 1_supercritical_pitchfork: cmds1.m, main command, run this!

cmds1.m

```
%% demo for supercritical Pitchfork bifurcation on Exercise 2.27(a) of Khal
% This is a demo for
% \dot{x}_1=x2
% \dot{x}_2=\mu*(x1+x2)-x2-x1^3-3*x1^2*x2

close all; keep pphome;
%% cell 1: init and cont of trivial branch
p=[]; par=[-2]; % set initial value of parameter mu=-2.
p=init(p,par); p=setfn(p,'tr'); p=cont(p); %continuation of trivial branch.

%% cell 2: switch to first bifurcating branches and continue
p=swibra('tr','bpt1','b1',0.1); p=cont(p);

%% cell 3: plot the bifurcation diagram.
plotbra('tr'); plotbra('b1');
```

Bifurcation analysis

Numerical tool: pde2path

Within 1_supercritical_pitchfork: init.m and oosetfemops.m

cmds1.m | init.m | oosetfemops.m

```
function p=init(p,par)  % init routine for AC on interva
p.np=2; %dimension of ODE systems.

%% default setup, no need to change.
p=stanparam(p); screenlayout(p); p.sw.sfem=-1;
p.fuha.sG=@sG; p.fuha.sGjac=@sGjac;
pde=stanpdeo1D(1,4/p.np); p.pdeo=pde; % domain and mesh
p.nu=p.np; p.sol.xi=1/(p.nu); [po,t,e]=getpte(p);
p.mat.M=eye(p.np);

%% define the trivial solution. In most cases just zero.
p.u=zeros(p.np,1);
p.u=[p.u; par']; % initial guess (here 0, explicitly kno

p.sw.foldcheck=1;
p.plot.auxdict={'mu'}; %for plotting to show lambda
p.plot.pstyle=1; %defaulty
p.nc.nsteps=200; %default continuation step
p.sw.bifcheck=2; %bifcheck=2: check the bifurcation poin
p.nc.ilam=1; %select ilam th parameter as the continuati
p.nc.lammax=2; %the maximum lambda you want to continue
p.sol.ds=0.001; %the initial step size of numerical cont
p.nc.dsmax=0.05; %the maximal step size of numerical con
```

Modify this if you compute ODE with dimension >2.

cmds1.m | init.m | oosetfemops.m

```
function p=oosetfemops(p)
%mass matrix is identity matr
%system.
p.mat.M=eye(p.np);
```

No need to change these two functions in most cases

Bifurcation analysis

Numerical tool: pde2path

Within 1_supercritical_pitchfork: sG.m and sGjac.m

```
function r=sG(p,u)  % AC with periodic BC
par=u(p.nu+1:end); up=u(1:p.nu); % params, and u on

%get parameter mu and state variable x1 and x2.
mu=par(1);
x1=up(1);
x2=up(2);

% Right hand side of
% \dot{x}_1=x2
% \dot{x}_2=\mu*(x1+x2)-x2-x1^3-3*x1^2*x2
r(1,1)=x2;
r(2,1)=mu*(x1+x2)-x2-x1^3-3*x1^2*x2;

r=-r; %reverse the sign due to pde2path convention
```

```
function Gu=sGjac(p,u)  % PDE Jacobian for AC with pB
par=u(p.nu+1:end); up=u(1:p.nu); % params, and u on p

%get parameter mu and state variable x1 and x2.
mu=par(1);
x1=up(1);
x2=up(2);

% Gu is Jacobian matrix associated with nonlinear sys
% \dot{x}_1=x2
% \dot{x}_2=\mu*(x1+x2)-x2-x1^3-3*x1^2*x2
Gu(1,1)=0;
Gu(1,2)=1;
Gu(2,1)=mu-3*x1^2-6*x1*x2;
Gu(2,2)=mu-1-3*x1^2;

Gu=-Gu; %reverse the sign due to pde2path convention
```

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \mu(x_1 + x_2) - x_2 - x_1^3 - 3x_1^2 x_2$$

Modify these two functions will change to a different dynamical system

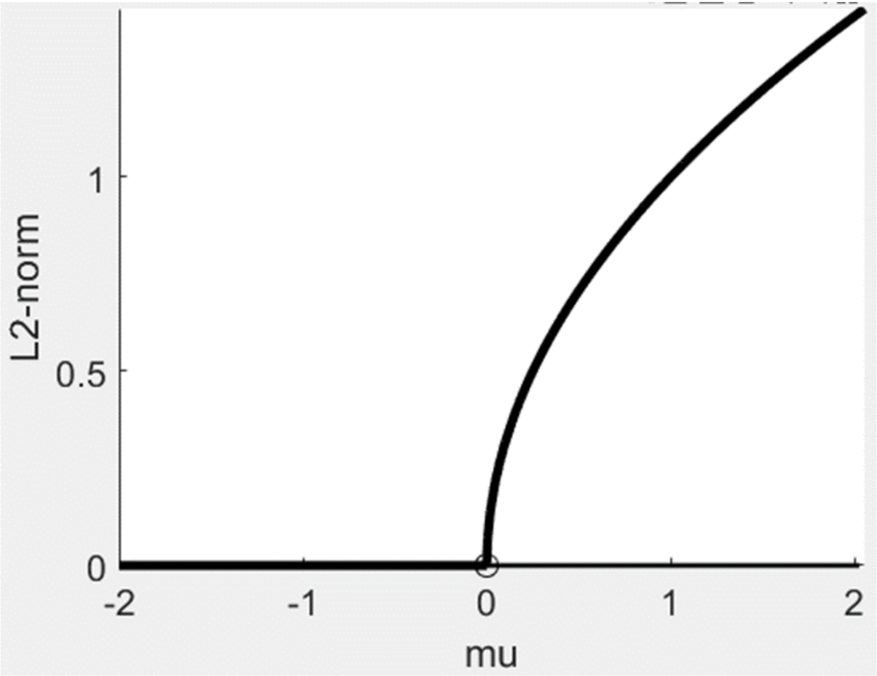
Bifurcation analysis

Numerical tool: pde2path

- Supercritical Pitchfork
- Within 1_supercritical_pitchfork

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \mu(x_1 + x_2) - x_2 - x_1^3 - 3x_1^2 x_2$$

Exercise 2.27(1)

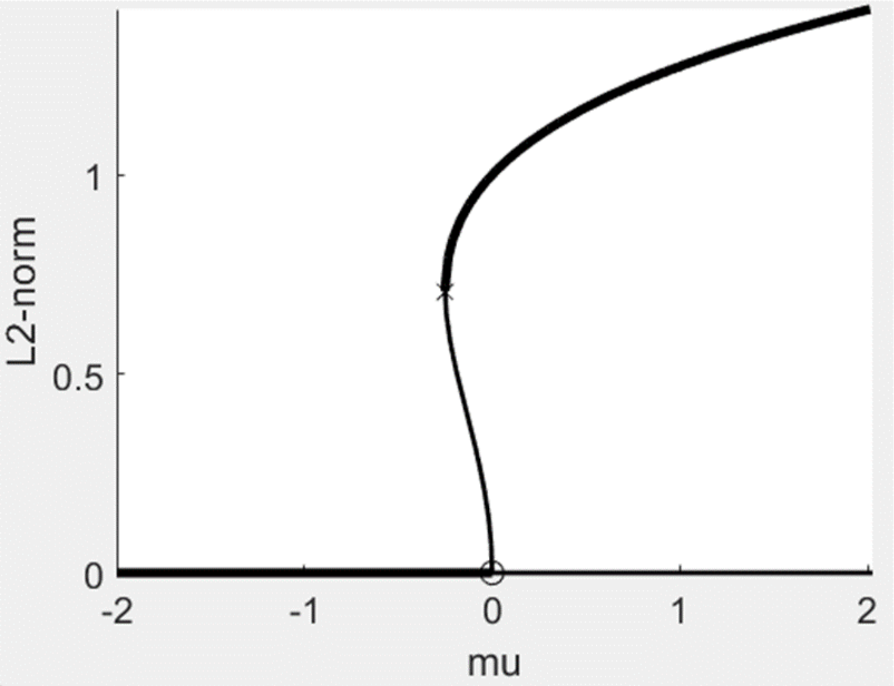


Bifurcation analysis

Numerical tool: pde2path

- Subcritical Pitchfork
- Within 2_subcritical_pitchfork

$$\dot{x}_1 = \mu x_1 + x_1^3 - x_1^5$$
$$\dot{x}_2 = -x_2$$



Bifurcation analysis

Numerical tool: pde2path

Within 3_supercritical_Hopf: cmds1.m

```
close all; keep pphome;
%% cell 1: init and cont of trivial branch
p=[]; par=[-2]; % set initial value of parameter mu=-2.
p=init(p,par); p=setfn(p,'tr'); p=cont(p);

%% cell 2: switch to Hopf bifurcating branches and continue
para=4; ds=0.1; figure(2); clf; aux=[];
p=hoswibra('tr','hpt1',ds,para,'hpt1',aux); nsteps=200;%switch to Hopf bifu

%% cell 3: continue in Hopf branch
p.hopf.nfloq=2;
p.hopf.jac=1; p.nc.dsmax=0.05; p.hopf.xi=0.05; p.file.smod=5; p.sw.verb=2;
p.hopf.flcheck=1; % switch for Floquet-comp (stability of periodic orbit):
p.sw.bifcheck=1;  % switch for bifurcation detection: 0:off, 1:on
p=hocont(p,nsteps);

%% cell 4: plot bifurcation diagram.
plotbra('tr'); plotbra('hpt1');
```

Bifurcation analysis

Numerical tool: pde2path

Within 3_supercritical_Hopf: sG.m and sGjac.m

```matlab
function r=sG(p,u)  % AC with periodic BC
par=u(p.nu+1:end); up=u(1:p.nu); % params, and u on

%get parameter mu and state variable x1 and x2.
mu=par(1);
x1=up(1);
x2=up(2);

% RHS of the nonlinear system
% \dot{x}_1=x1*[\mu-(x1^2+x2^2)]-x2
% \dot{x}_2=x2*[\mu-(x1^2+x2^2)]+x1
r(1,1)=x1*(mu-x1^2-x2^2)-x2;
r(2,1)=x2*(mu-x1^2-x2^2)+x1;

r=-r; %reverse the sign due to pde2path convention
```

```matlab
function Gu=sGjac(p,u)  % PDE Jacobian for AC with pE
par=u(p.nu+1:end); up=u(1:p.nu); % params, and u on p

%get parameter mu and state variable x1 and x2.
mu=par(1);
x1=up(1);
x2=up(2);

% Jacobian matrix of the nonlinear system
% \dot{x}_1=x1*[\mu-(x1^2+x2^2)]-x2
% \dot{x}_2=x2*[\mu-(x1^2+x2^2)]+x1
Gu(1,1)=mu-x1^2-x2^2-2*x1^2;
Gu(1,2)=-2*x2*x1-1;
Gu(2,1)=-2*x1*x2+1;
Gu(2,2)=mu-x1^2-x2^2-2*x2^2;

Gu=-Gu; %reverse the sign due to pde2path convention
```

$$\dot{x}_1 = x_1[\mu - (x_1^2 + x_2^2)] - x_2$$
$$\dot{x}_2 = x_2[\mu - (x_1^2 + x_2^2)] + x_1$$

Bifurcation analysis

Numerical tool: pde2path

Within 3_supercritical_Hopf: nodalf.m, the same as sG.m, needs to be modified together

init.m   nodalf.m

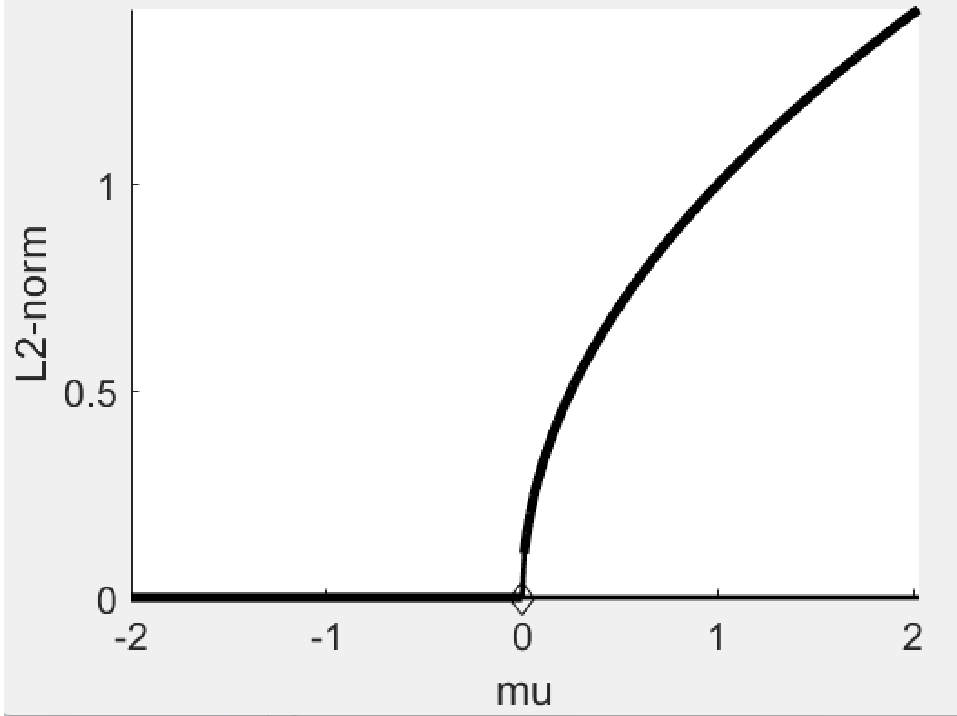
```
function f=nodalf(p,u) % nonlinearity for
par=u(p.nu+1:end); up=u(1:p.nu); % params,
mu=par(1);
x1=up(1);
x2=up(2);
f(1,1)=x1*(mu-x1^2-x2^2)-x2;
f(2,1)=x2*(mu-x1^2-x2^2)+x1;

f=-f;
```

Bifurcation analysis

Numerical tool: pde2path

- Supercritical Hopf
- Within 3_supercritical_Hopf

$$\dot{x}_1 = x_1[\mu - (x_1^2 + x_2^2)] - x_2$$
$$\dot{x}_2 = x_2[\mu - (x_1^2 + x_2^2)] + x_1$$



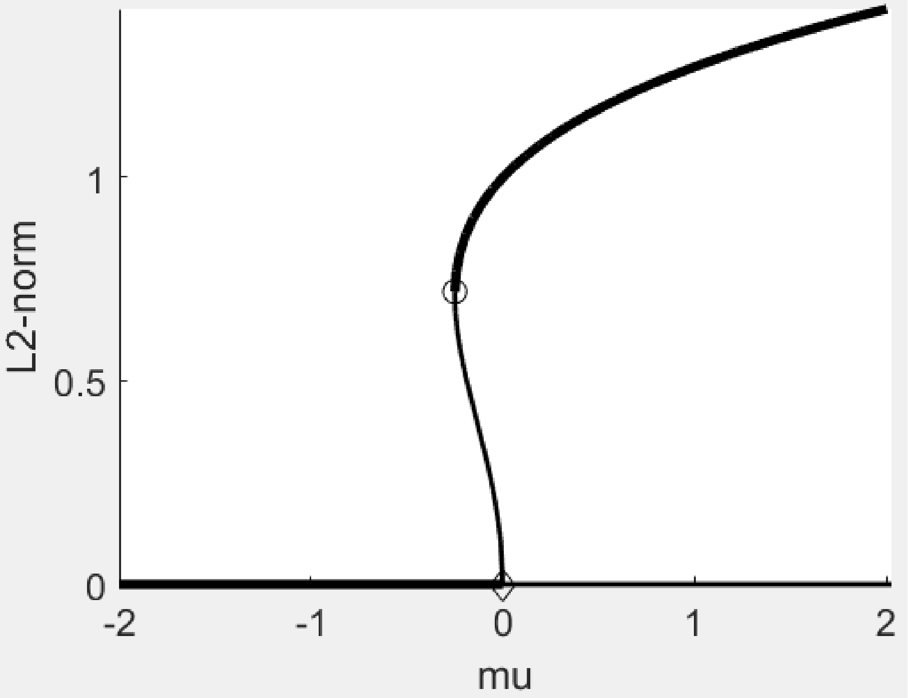
Page 73 of Khalil

Bifurcation analysis

Numerical tool: pde2path

- Subcritical Hopf
- Within 4_subcritical_Hopf

$$\dot{x}_1 = x_1[\mu + (x_1^2 + x_2^2) - (x_1^2 + x_2^2)^2] - x_2$$
$$\dot{x}_2 = x_2[\mu + (x_1^2 + x_2^2) - (x_1^2 + x_2^2)^2] + x_1$$



Page 74 of Khalil

Common problems of installing software

Mosek

The license file should be placed inside a folder called "mosek" under the user's home directory ($HOME/mosek/mosek.lic or %USERPROFILE%\mosek\mosek.lic). In most typical cases that will be:

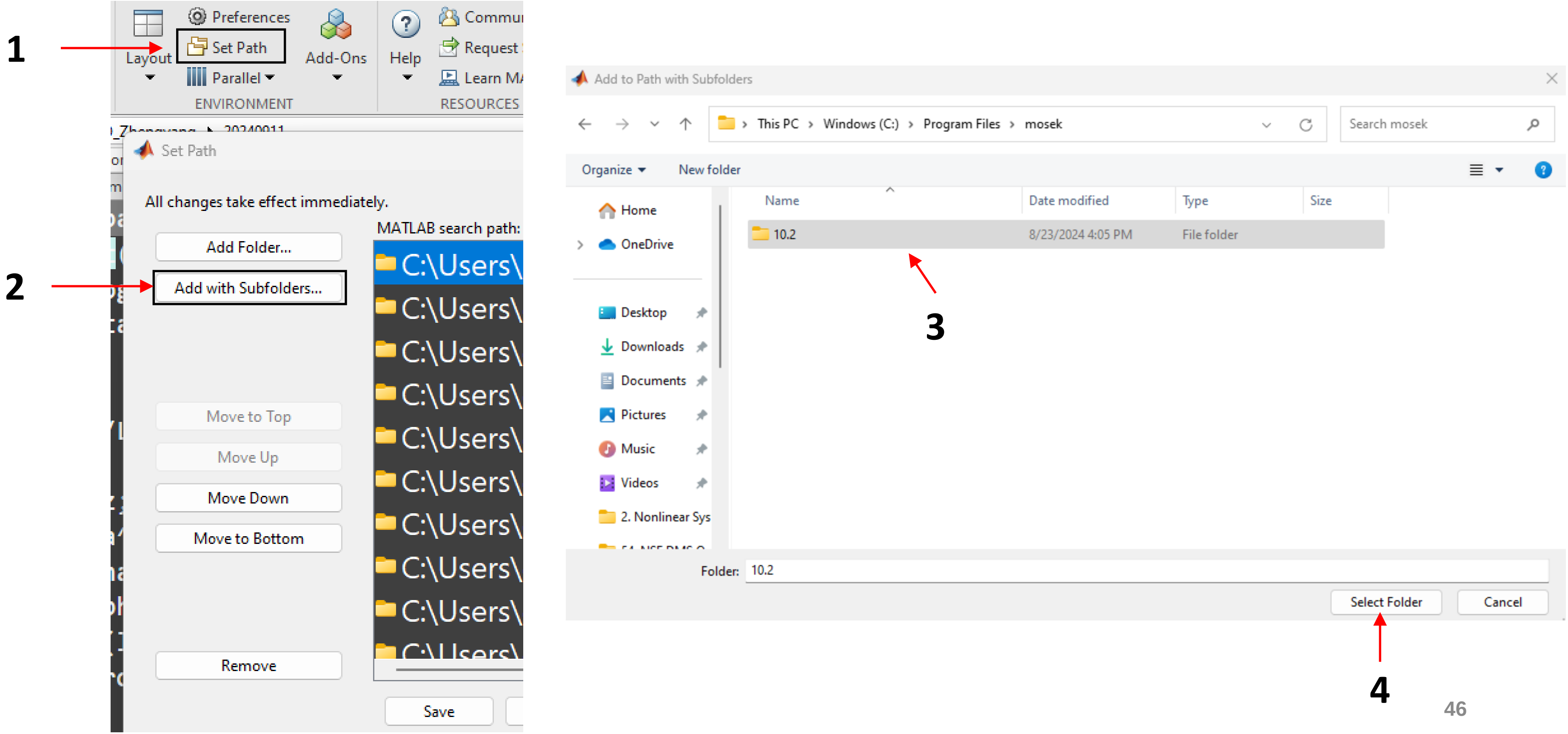
```
/home/YOUR_USER_NAME/mosek/mosek.lic (Linux)
/Users/YOUR_USER_NAME/mosek/mosek.lic (OSX)
C:\Users\YOUR_USER_NAME\mosek\mosek.lic (Windows)
```

Where *YOUR_USER_NAME* is your user ID on the computer.

Create a folder called 'mosek' if you do not have a folder called mosek there.
Need to put the license in EXACTLY the same place

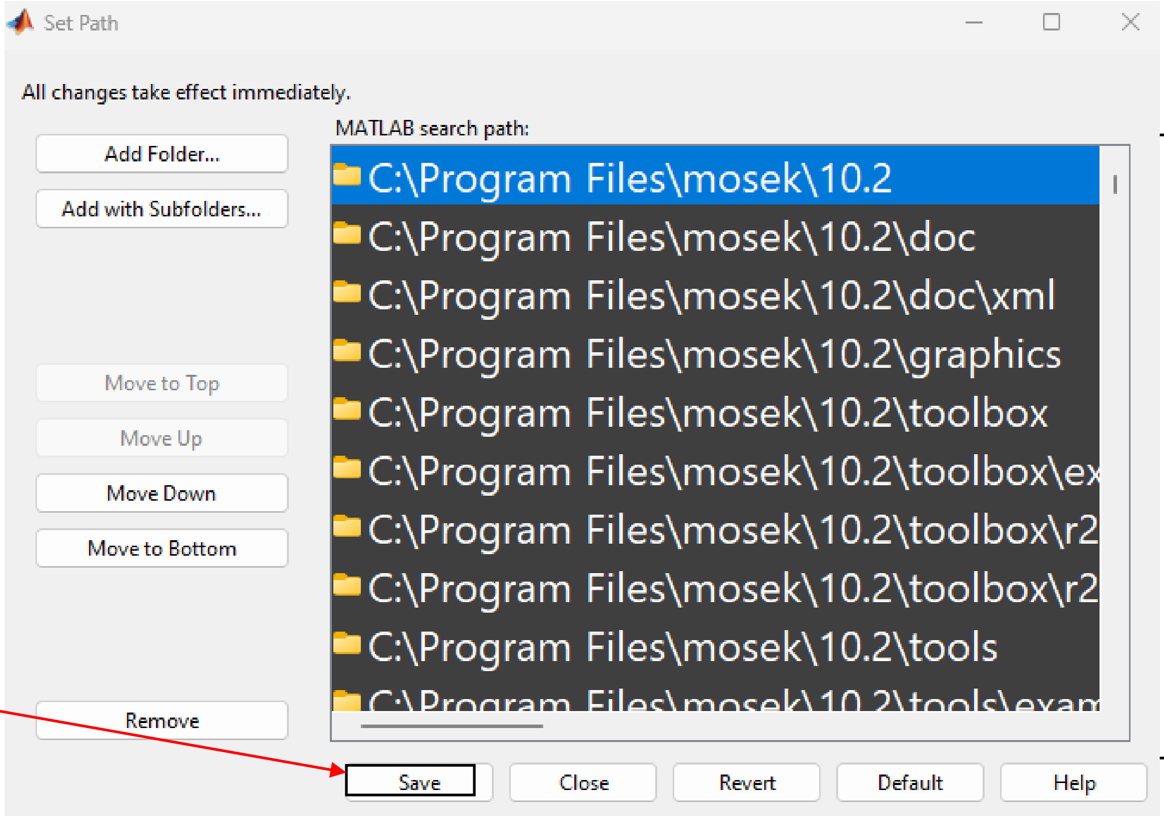
Common problems of installing software

Mosek



Common problems of installing software

Mosek

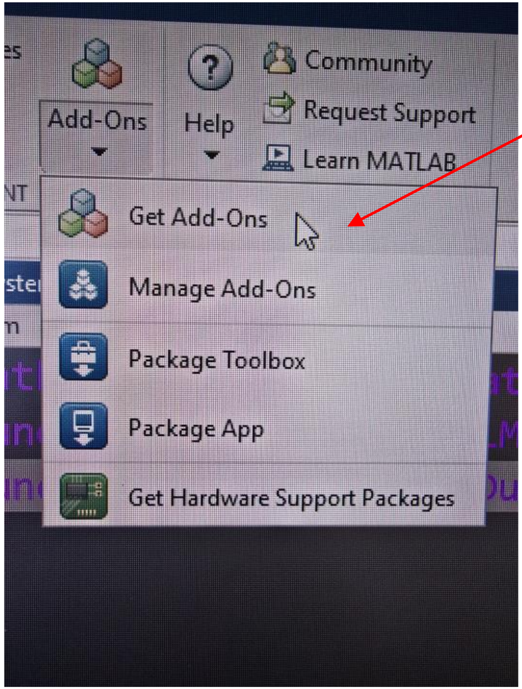


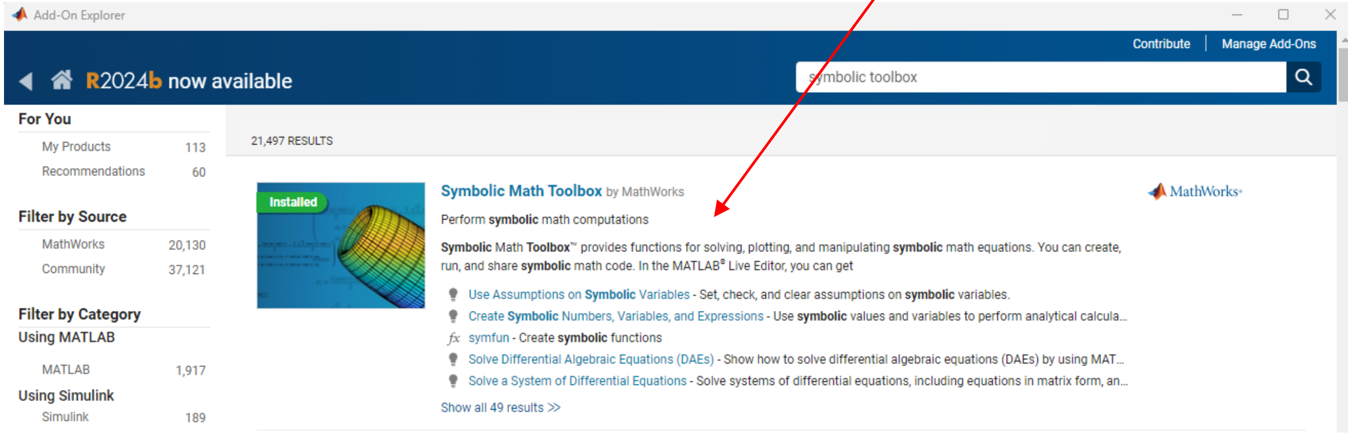
Check all subfolders are here

5

Common problems of installing software

Symbolic Math toolbox





Common problems of installing software

pde2path

- Make sure to run ‘install.m’ before running code within 2_Bifurcation_pde2path.

Editor - C:\Users\chl23026\Nutstore\1\Nutstore\Workspace\1_Nonlinear_Systems\Nonlinear_Systems\install.m

init.m | nodalf.m | sG.m | sGjac.m | SOS_4th_order.m | install.m | +

```
run('./2_Bifurcation_pde2path/pde2path/setpde2path.m');
addpath(genpath(['./1_Lyapunov_method_YALMIP/YALMIP-master']));
addpath(genpath(['./1_Lyapunov_method_YALMIP/SeDuMi_1_3']));
```

- No SetPath as mosek is needed.

- HW3 due is extended to October 12 (Saturday) 11:59pm.