Numerical bifurcation analysis of delay differential equations

Dirk Roose

Dept. of Computer Science

K.U.Leuven

Dirk.Roose@cs.kuleuven.be

Acknowledgements

Many thanks to

- Koen Engelborghs
- Tatyana Luzyanina
- Koen Verheyden
- Giovanni Samaey
- Kirk Green
- Robert Szalai

because they did the work ...

Overview

- *Lecture 1*
  - numerical methods for delay differential equations (DDEs) with constant pointwise delays
    - stability analysis of steady state solutions
  - short introduction to software package DDE-BIFTOOL

- *Practical session*
  - Demo & hands-on experience with DDE-BIFTOOL

Overview (2)

- *Lecture 2*
  - computation & stability analysis of periodic solutions
  - computation of connecting orbits (homo- & heteroclinic orbits)
  - short introduction to software package PDDE-CONT for continuation and bifurcation analysis of periodic solutions of DDEs

- *Practical session*
  - Demo & hands-on experience with DDE-BIFTOOL and PDDE-CONT

Delay Differential Equations

http://www.scholarpedia.org/article/Delay-differential_equations

Delay differential equations (DDEs) differ from ODEs in that the derivative at any time depends on the solution at prior times (and in the case of *neutral* equations on the derivative at prior times)

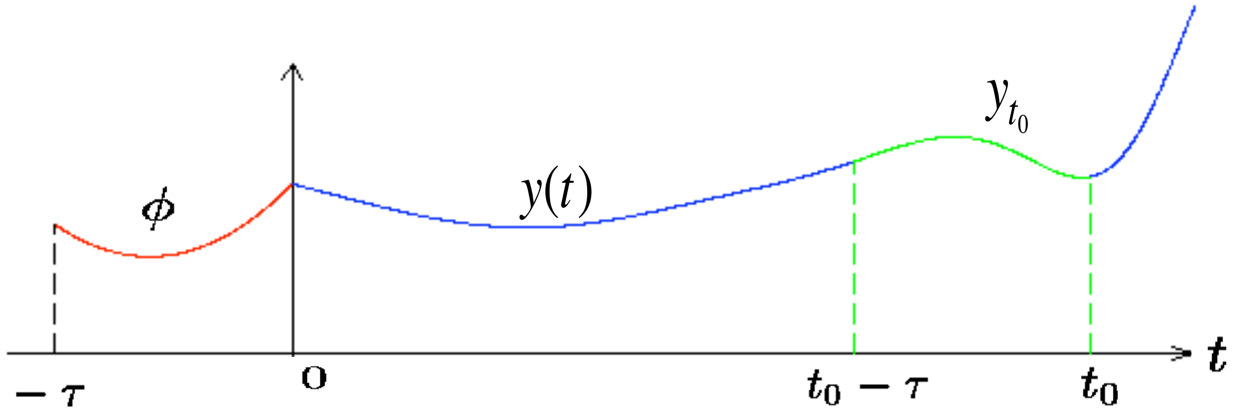
$$\frac{d}{dt}y(t) = f(y(t), y(t-\tau_1), y(t-\tau_2), \ldots, y(t-\tau_m))$$

constant or state-dependent 'point' delays $\tau_i\,(y(t))$

DDEs often arise when traditional pointwise modeling assumptions are replaced by more realistic distributed assumptions, for example, when the birth rate of predators is affected by prior levels of predators or prey rather than by only the current levels in a predator-prey model.

Initial Function

Because the derivative *y'(t)* depends on the solution at previous time(s), it is necessary to provide an *initial history function* to specify the value of the solution before time *t = 0*. In many common models the history is a constant; but non constant history functions are encountered routinely.



For most problems there is a jump derivative discontinuity at the inital time.

Discontinuity propagation

In most models, the DDE and the initial function are incompatible: for some derivative order, usually the first, the left and right derivatives at *t = 0* are not equal.

$y'(t) = y(t-1)$ const. history $y'(0^+) = 1 \neq y'(0^-) = 0.$

Fascinating property: how such derivative discontinuities are propagated in time.

For the equation and history just described, for example, the initial first discontinuity is propagated as a second degree discontinuity at time t = 1, as a third degree discontinuity at time t = 2, and, more generally, as a discontinuity in the (n+1)st derivative at time t = n.

Discontinuity propagation

This behavior is typical of that for a wide class of delay differential equations: generalized smoothing occurs as the initial derivative discontinuity is propagated successively to higher order derivatives.

Smoothing need not occur for neutral equations or for non-neutral equations with vanishing delays.

Introduction to DDEs

used in various application areas

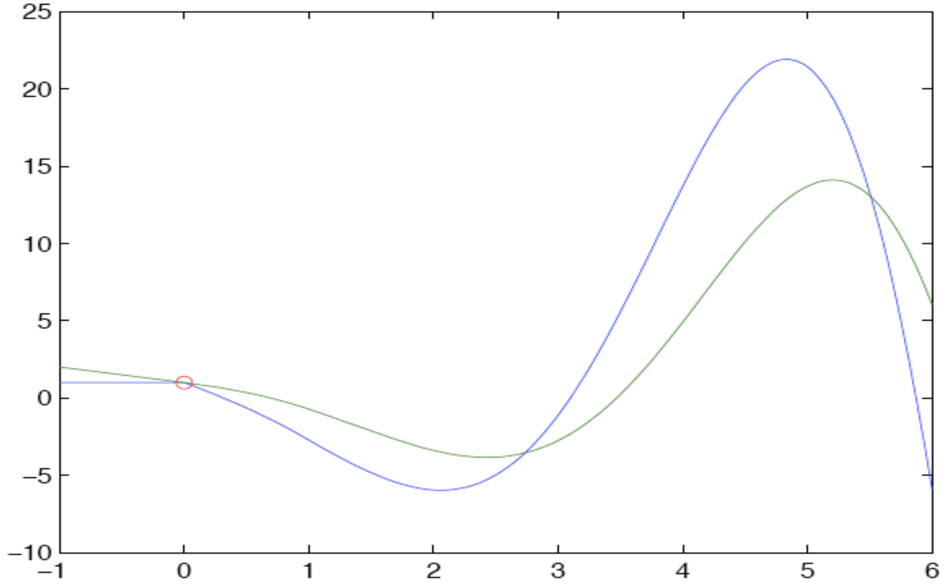
- Biology, physiology
- population dynamics (cells, viruses)
- control systems
- semiconductor lasers (optical feedback)
- high-speed cutting, milling & drilling
- congestion control in communication networks
- car following models

Example: climate modeling

An early model of the El Niño/Southern Oscillation phenomenon with a physical parameter $\alpha > 0$ is

$$T'(t) = T(t) - \alpha T(t-\tau)$$

$\alpha = 2; \tau = 1$



A nonlinear ENSO model with periodic forcing is

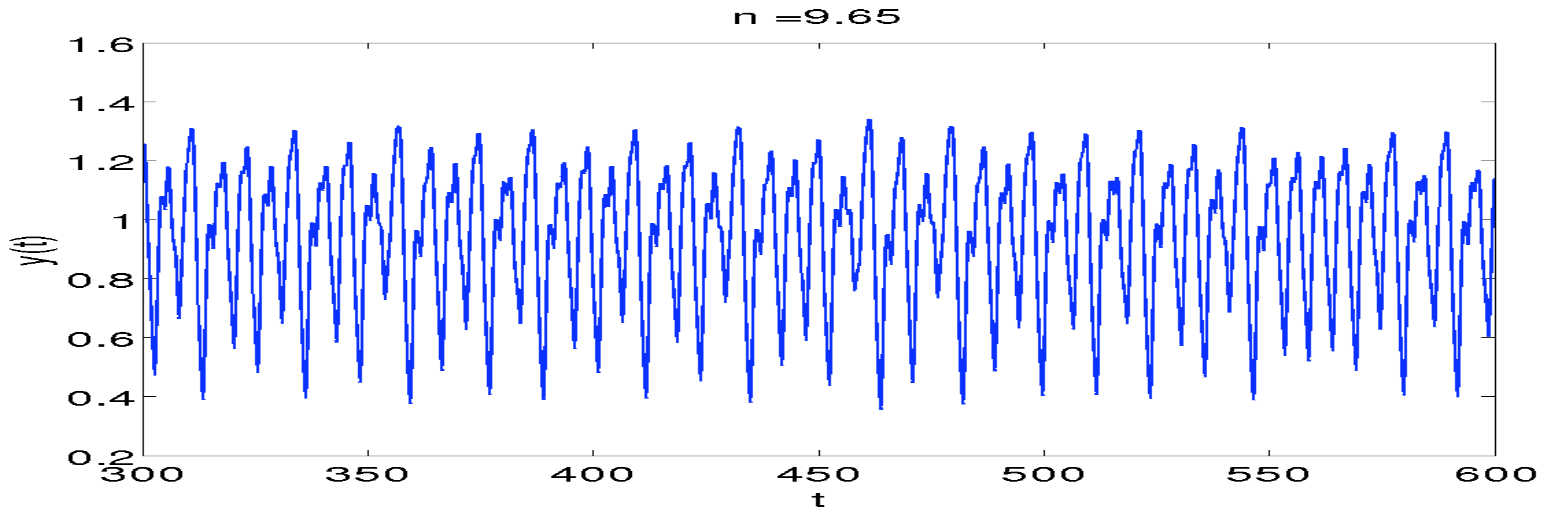
$$h'(t) = -a\ \tanh[\kappa\, h(t-\tau)] + b\ \cos(2\pi\, \omega\, t)$$

Example: physiology

- Mackey-Glass equation
- physiological control system (feedback system) breathing control
- control variable is sensed and appropriate changes are made in the rates of production and/or decay.

  control variable: e.g. concentration blood cells

$$\dot{x}(t) = ax(t) + b\frac{x(t-\tau)}{1+x^{10}(t-\tau)}$$



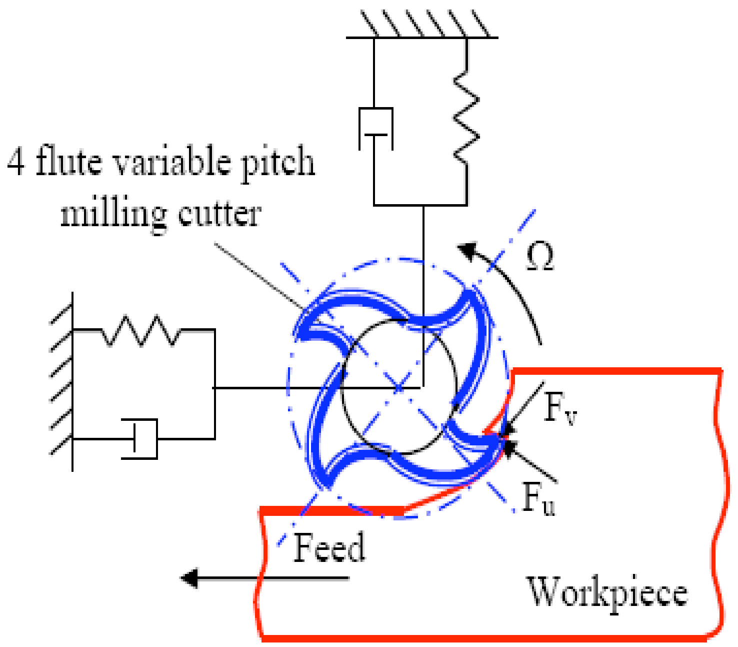
Example: laser with optical feedback

- In many laser systems delay arises due to the finite travel time of light between components of the system and may lead to different types of dynamic behaviour including chaos
- DDE model of a semiconductor laser with filtered optical feedback

$$
\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}t} &= (1+i\alpha)N(t) + \kappa F(t),\\
T\frac{\mathrm{d}N}{\mathrm{d}t} &= P - N(t) - (1+2N(t))|E(t)|^2,\\
\frac{\mathrm{d}F}{\mathrm{d}t} &= \Lambda E(t-\tau)\mathrm{e}^{-iC_p} + (i\Delta - \Lambda)F(t),
\end{aligned}
$$

E: complex optical field, *N:* population inversion of the laser, *F:* complex optical field of filter, τ : delay (external feedback)

Example: milling machine



Regenerative effect: cutting tool cuts a surface that was produced by the same tool some time ago.
The cutting forces nonlinearly depend on the chip geometry, which in turn depends on the current and a delayed tool position.
Rotation of each tooth

$\Rightarrow$ periodic coefficients

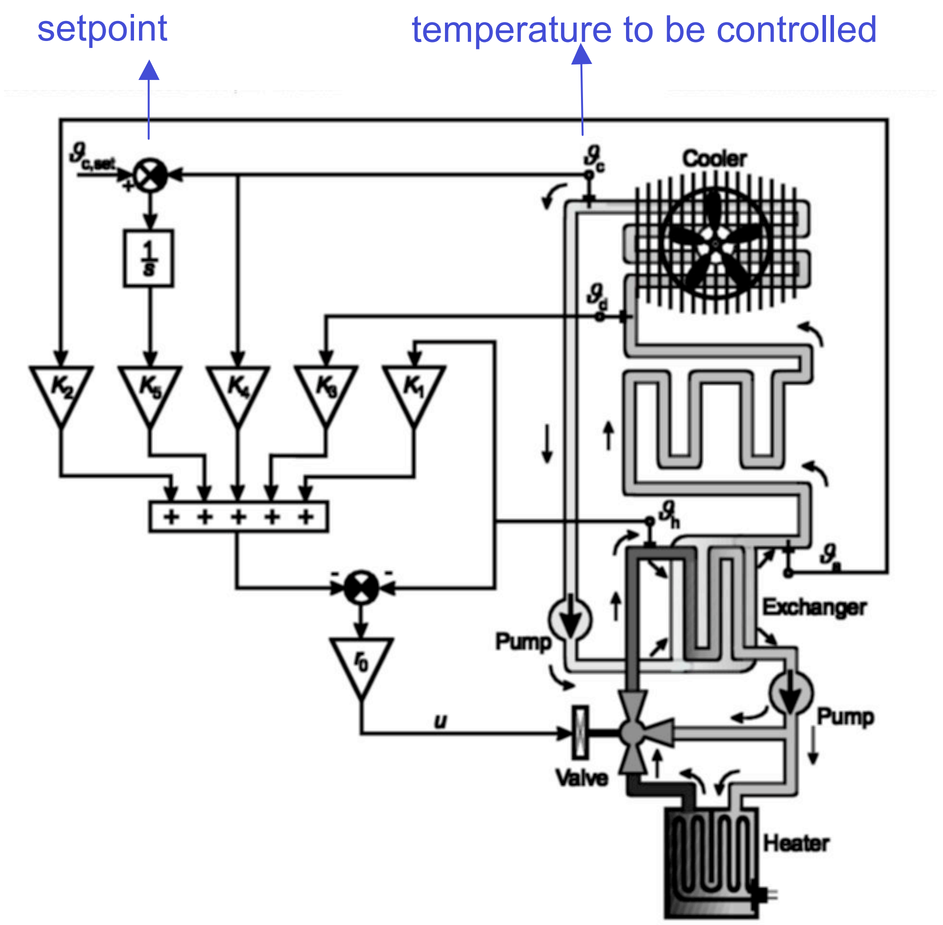
Delay is inversely proportional to speed

$$\dot{x}(t) = F(x(t)) + B(\omega t)\,(x(t) - x(t - \tau(t)))$$

$$\begin{aligned}&\ddot{x}(t) + 2\xi\dot{x}(t) + x(t) = g(t)\hat{w}(\cos 2\pi t/T + 0.3\sin 2\pi t/T)\times\\ &\times[H(1 + x(t-2T) - x(t-T))F_c((1 + x(t-T) - x(t))\sin 2\pi t/T) \qquad (46)\\ &+H(x(t-T) - x(t-2T) - 1)F_c((2 + x(t-T) - x(t))\sin 2\pi t/T)],\end{aligned}$$

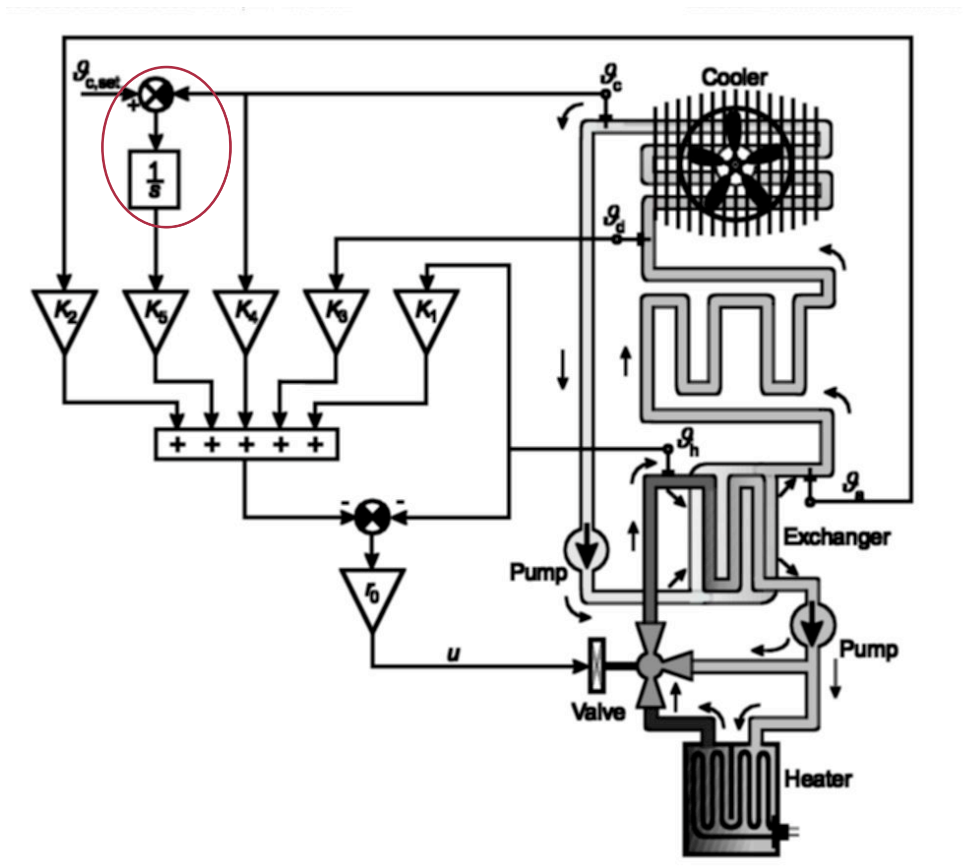
Example: control of heating system

Lab. Tomas Vyhlidal, CTU Prague



Example: heating system (2)

$$\begin{cases} T_h \dot{x}_h(t) = -x_h(t-\eta_h) + K_b x_a(t-\tau_b) + K_u x_{h,set}(t-\tau_u) \\ T_a \dot{x}_a(t) = -x_a(t) + x_c(t-\tau_e) + K_a \left(x_h(t) - \frac{1+q}{2} x_a(t) - \frac{1-q}{2} x_c(t-\tau_e) \right) \\ T_d \dot{x}_d(t) = -x_d(t) + K_d x_a(t-\tau_d) \\ T_c \dot{x}_c(t) = -x_c(t-\eta_c) + K_c x_d(t-\tau_c) \\ \dot{x}_e(t) = x_{c,set}(t) - x_c(t) \end{cases}$$



7 delays !

Control law (PI+ state feedback)

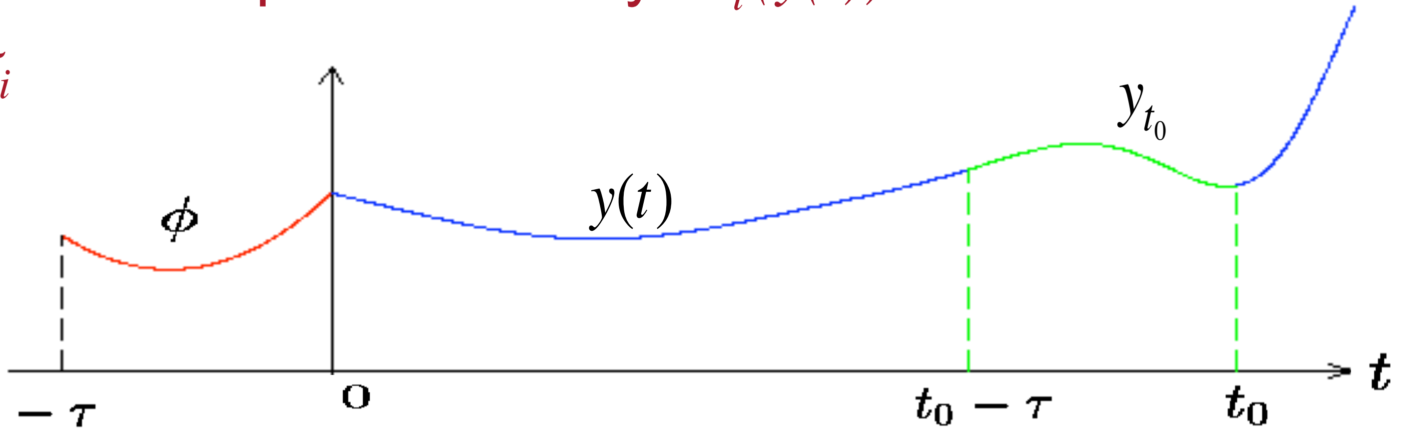
$$x_{h,set} = K \left[x_h \ x_a \ x_d \ x_c \ x_e \right]^T$$

Introduction to DDEs

$$\frac{d}{dt}y(t) = f(y(t), y(t-\tau_1), y(t-\tau_2), \dots, y(t-\tau_m))$$

constant or state-dependent delays $\tau_i\,(y(t))$

τ : max τ_i



- initial function segment $\phi(\theta), \theta \in [-\tau, 0]$ has to be specified
- state at $t=t_0$: function segment $y_{t_0}(\theta), \theta \in [-\tau, 0]$

  $\Rightarrow$ *infinite dimensional state space* $\Rightarrow$ *analytical & numerical calculations more difficult than for ODEs*

Steady state solutions

$$\frac{d}{dt}y(t) = f(y(t), y(t-\tau_1), y(t-\tau_2), \ldots, y(t-\tau_m))$$

Computation $\quad y^* \in R^n \to f(y^*, y^*, \ldots, y^*) = 0$

Stability: constant delays: consider variational equation

$$\frac{\mathrm{d}y}{\mathrm{d}t}(t) = A_0 y(t) + A_1 y(t-\tau_1) + \ldots + A_m y(t-\tau_m)$$

$$A_i \equiv \left.\frac{\partial f}{\partial y^i}\right|_{\left(y^0, y^1, \ldots, y^m\right) = \left(y^*, y^*, \ldots, y^*\right)}$$

determine roots λ of characteristic equation

$$\det\left(\lambda I - A_0 - \sum_{i=1}^{m} A_i \mathrm{e}^{-\lambda\tau_i}\right) = 0$$

nonlinear generalised eigenvalue problem

state-dependent delay: linearisation: τ can be treated as constant

Characteristic equation

[Shampine `http://faculty.smu.edu/shampine/Read1.pdf`]

Linear, homogeneous, constant-coefficient ODEs have solutions of the form $y(t) = e^{\lambda t}$. Any root λ of the **characteristic equation** provides a solution. This polynomial equation has a finite number of roots.

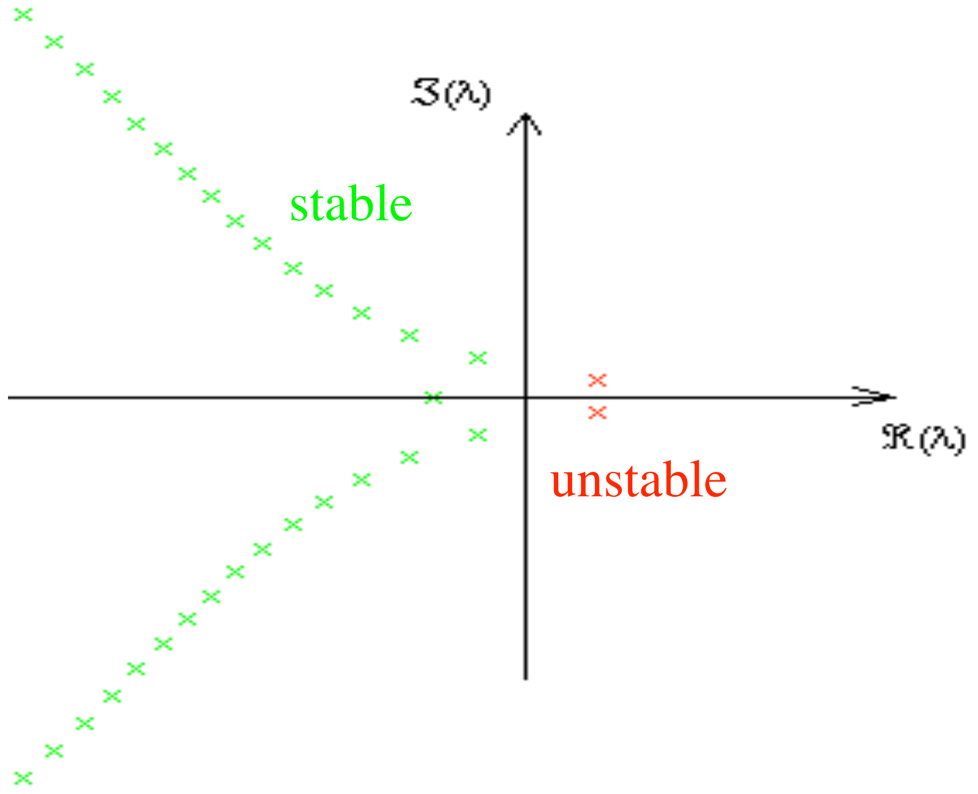
The characteristic equation for linear, homogeneous, constant-coefficient DDEs is **transcendental**. Generally there are infinitely many roots λ. El'sgol'ts and Norkin give asymptotic expressions for these roots.

The differential equation is **stable** if all roots of the characteristic equation satisfy $Re(\lambda) \leq \beta < 0$.

It is **unstable** if for some root, $Re(\lambda) > 0$.

Stability of steady state solutions

$$\det\left(\lambda I - A_0 - \sum_{i=1}^{m} A_i e^{-\lambda \tau_i}\right) = 0$$



infinite number roots of characteristic equation ('eigenvalues')

only finite number of eigenvalues with $\mathrm{Re}(\lambda) > r$

Numerical stability analysis

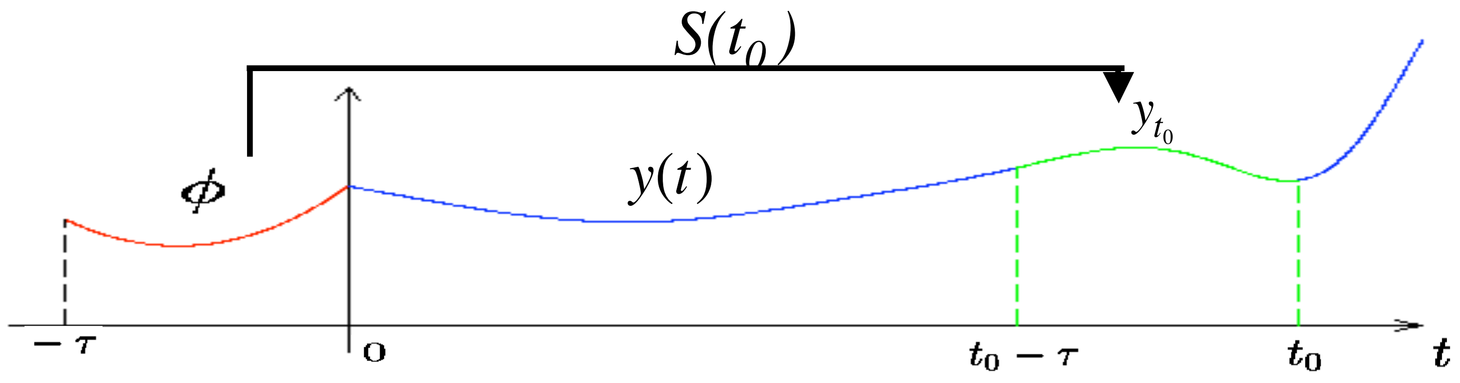
numerical methods to determine stability of a *steady state solution* by computing the ‘rightmost eigenvalues’

- based on discretization of solution operator of variational equation via
  - time integration (e.g. linear multistep method (LMS))
  - pseudospectral discretization
- based on discretization of infinitisimal generator via
  - time integration
  - pseudospectral discretization

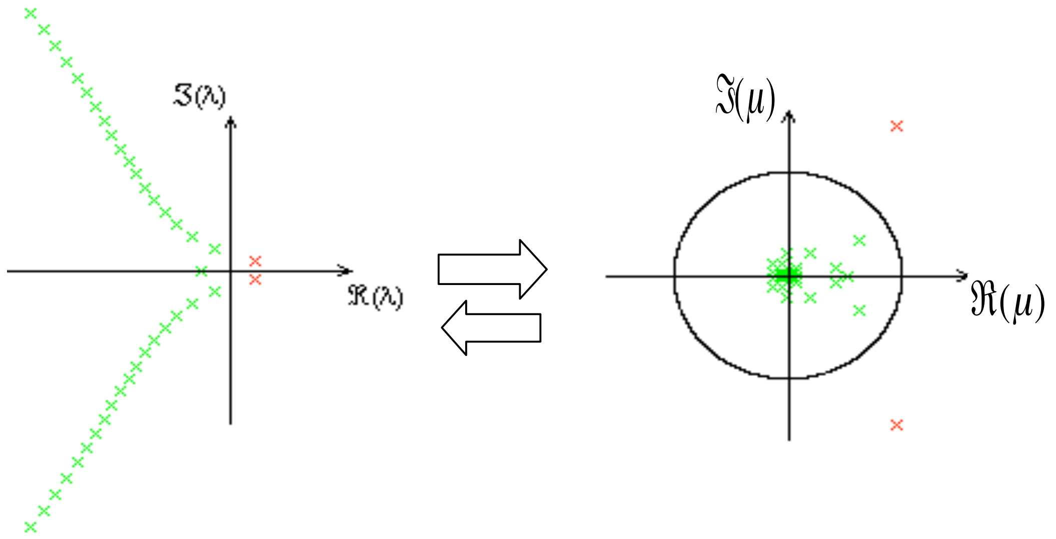
Discretization of solution operator

$S(t)$: solution operator of variational equation

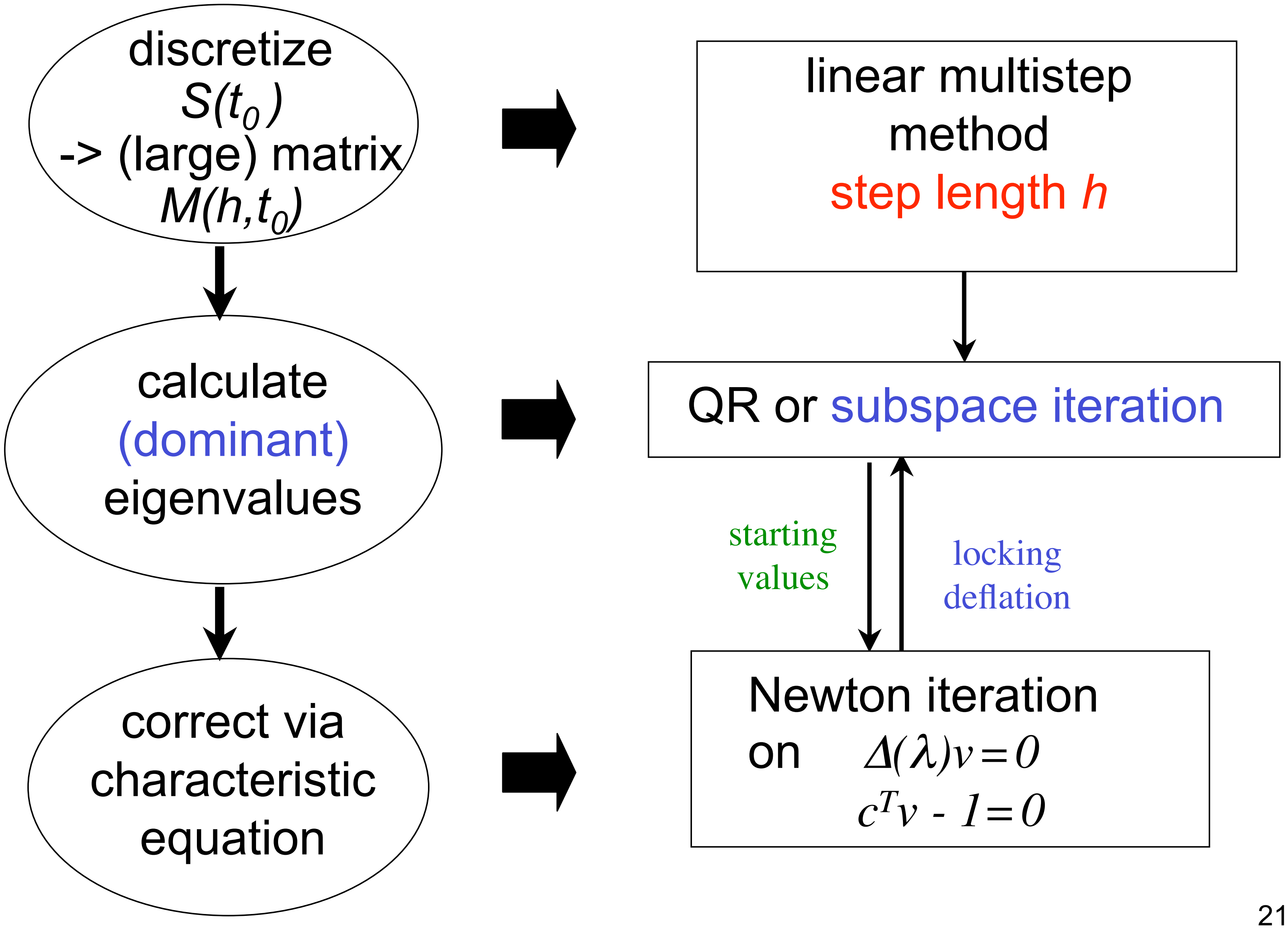
$$\mathcal{S}(t)y(\cdot)(\theta) = y(t+\theta), \quad -\tau \le \theta \le 0,\ t \ge 0.$$



eigenvalues of
solution operator $S(t_0)$
$\mu = \exp(\lambda t_0)$



Computation of dominant eigenvalues of $S(t_0)$



discretize $S(t_0)$ -> (large) matrix $M(h,t_0)$
calculate (dominant) eigenvalues
correct via characteristic equation
linear multistep method
step length h
QR or subspace iteration
starting values
locking deflation
Newton iteration on $\Delta(\lambda)v = 0$
$c^T v - 1 = 0$

Computation of dominant eigenvalues of $S(t_0)$

- matrix $M(h) = M(h,t_0)$: approximation / discretisation of $S(t_0)$

  eigenvalues $\mu = \exp(\lambda t_0)$

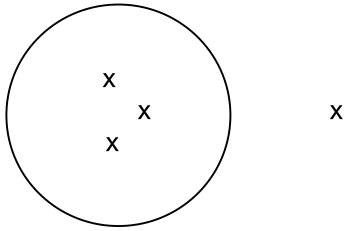
  – dimension of M = # mesh points in $[-\tau,0]$ x # eqs

  – preferably: dominant eigenvalues easy to compute

- choice of t_0

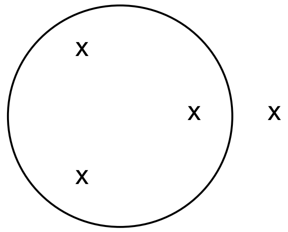
  t_0 large : + : expensive time integration

  - : well separated eigenvalues



  t_0 small : + : integration over short time interval

  - : μ's not well separated $\Rightarrow$ use QR



we use $t_0 = h$ (= step length) !!!

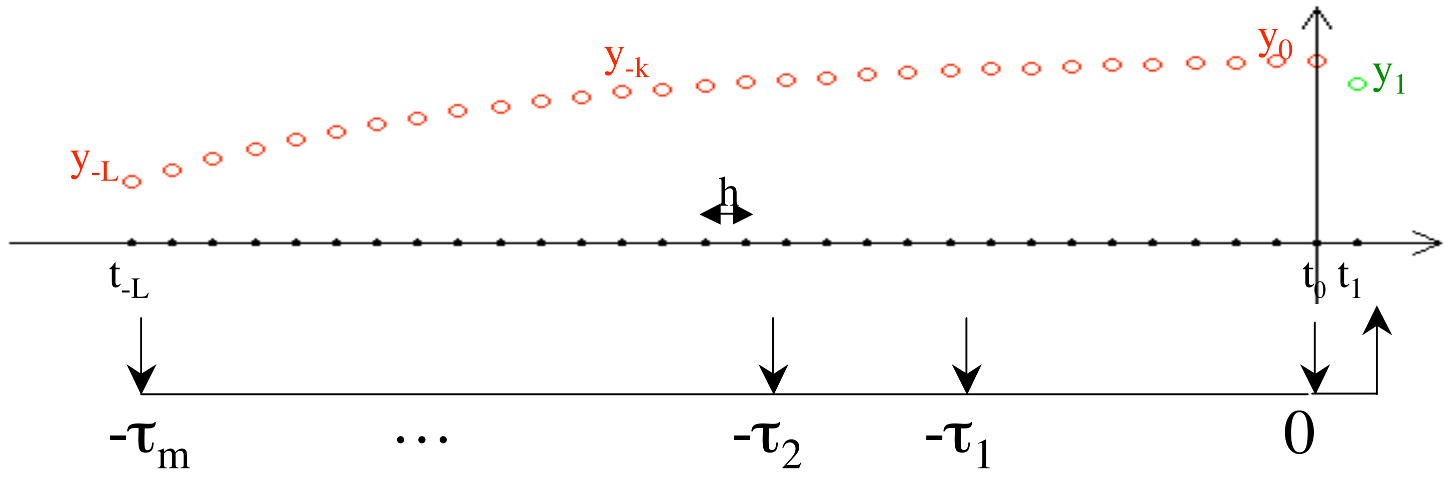
Discretization of solution operator

- (extended) delay interval discretized by equidistant mesh with spacing *h*
- solution represented by $y_i = y(t_i), t_i = ih$ *i=-L,...,0*
- LMS method: e.g. $y_1 = y_0 + h\left(A_0 y_0 + \sum_{j=1}^{m} A_j \tilde{y}(t_0 - \tau_j) \right)$

  approximations $\tilde{y}(t_0 - \tau_j)$ by interpolation
- discretization of solution operator

$$\left[y_{-L+1} \ldots\ y_0\ y_1 \right]^T = M(h) \left[y_{-L} \ldots\ y_{-1}\ y_0 \right]^T$$

Construction of matrix *M(h)*



$$\begin{bmatrix} y_{-L+1} \cdots y_0 \, y_1 \end{bmatrix}^T = M(h)\begin{bmatrix} y_{-L} \cdots y_{-1} \, y_0 \end{bmatrix}^T$$

$$\begin{bmatrix} 0 & 1 & & \\ & & \ddots & \\ & & & 1 \\ \blacksquare & \blacksquare & \cdots & \blacksquare \end{bmatrix} \begin{bmatrix} y_{-L} \\ y_{-L+1} \\ \vdots \\ y_0 \end{bmatrix} = \begin{bmatrix} y_{-L+1} \\ y_{-L+2} \\ \vdots \\ y_1 \end{bmatrix}$$

Matrix $M(h)$: dimension $N = n(L+1) \approx n\frac{\tau_{\max}}{h}$

Computation of rightmost eigenvalues

Compute eigenvalues of *M(h)* by QR-algorithm (‘eig’ in Matlab)

-> approximate (dominant) eigenvalues of solution operator

recover roots λ from eigenvalues μ

h should be chosen that all ‘rightmost’ roots λ with real part > *r* are approximated accurately

‘steplength heuristic’ in DDE-BIFTOOL

approximate eigenvalues can be corrected by Newton’s method applied to characteristic equation

$$\left(\lambda I - A_0 - \sum_{j=1}^{m} A_j \mathrm{e}^{-\lambda\tau_j}\right)v = 0$$
$$v_0{}^T v = 1$$

Reliable stability computation

- approximate all roots λ with $Re(\lambda) > 0$ accurately
  - suppose that steady state solution of DDE is delay-independent stable (sol. variational eq. stable for all τ)
  - determine region enclosing all λ with $Re(\lambda) > r$
  - determine 'radius' of LMS stability region ρ_{LMS} such that stability of solution of DDE and of LMS integrator 'coincide'

  $\Rightarrow$ heuristic for *h* : $h = 0.9 \dfrac{\rho_{LMS}}{\sum_{i=0}^{m} \|A_i\|}$

- approximate all eigenvalues λ with $Re(\lambda) > r$ accurately

$$h = 0.9 \frac{\rho_{LMS}}{\| A_0 \| + | r | + \sum \| A_i \| \exp(-r\tau_i)}$$

Stability of solution of DDE

Characteristic equation $\rightarrow$ $\quad \lambda \in \sigma(A_0 + \sum_{j=1}^{m} A_j e^{-\lambda\tau_j})$

define $\Sigma_\tau(\lambda) = \sigma(A_0 + \sum_{j=1}^{m} A_j e^{-\lambda\tau_j})$; λ is root iff $\lambda \in \Sigma_\tau(\lambda)$

if $\mathbb{C}^+ \cap \Sigma_\tau(\mathbb{C}^+) = \emptyset$ then solution is stable

$$\max |\Sigma_\tau(\mathbb{C}^+)| \leq \sum_{j=0}^{m} \| A_j \|$$

all roots with pos. real part lie in circle with radius $\sum_{j=0}^{m} \| A_j \|$

$\rho_{\mathrm{LMS},\epsilon}$: radius of disc in which imaginary axis is approximated by LMS(i[0,2π]) ‘up to ε’

Stability of solution of discrete system

Characteristic equation for discrete system can be written as

$$\frac{1}{h}\,\mathrm{LMS}(\tilde{\lambda}h) \in \sigma\Big(A_0 + \sum_{i=1}^{m} A_j \mathrm{e}^{-\frac{\tau_j}{h}\mathrm{Int}_j(\tilde{\lambda}h)}\Big)$$

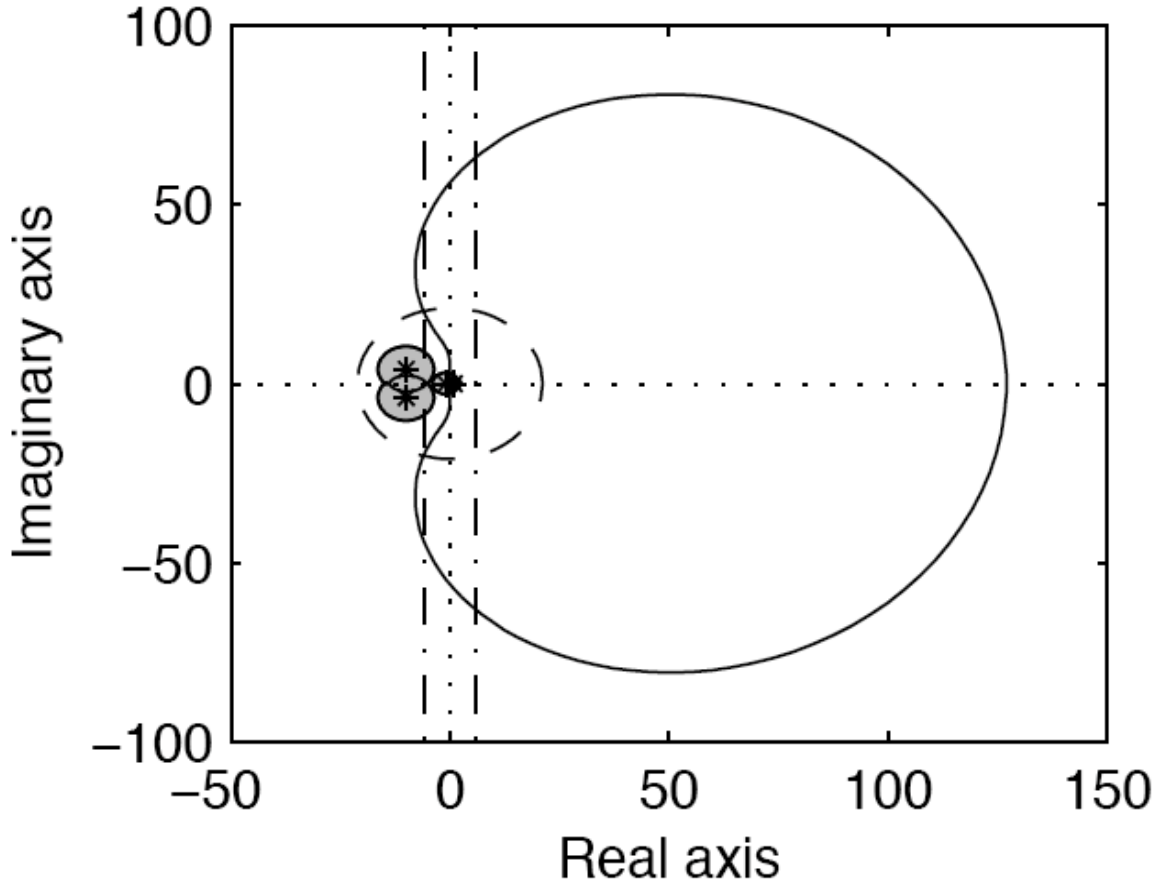
$$\mathrm{LMS}(z) := \frac{\alpha(\mathrm{e}^z)}{\beta(\mathrm{e}^z)} \qquad \alpha(\tilde{\mu}) := \sum_{i=0}^{k} \alpha_i \tilde{\mu}^i \quad \text{and} \quad \beta(\tilde{\mu}) := \sum_{i=0}^{k} \beta_i \tilde{\mu}^i$$

$$\tilde{\mu} = \exp(\tilde{\lambda}h)$$

Step length heuristic

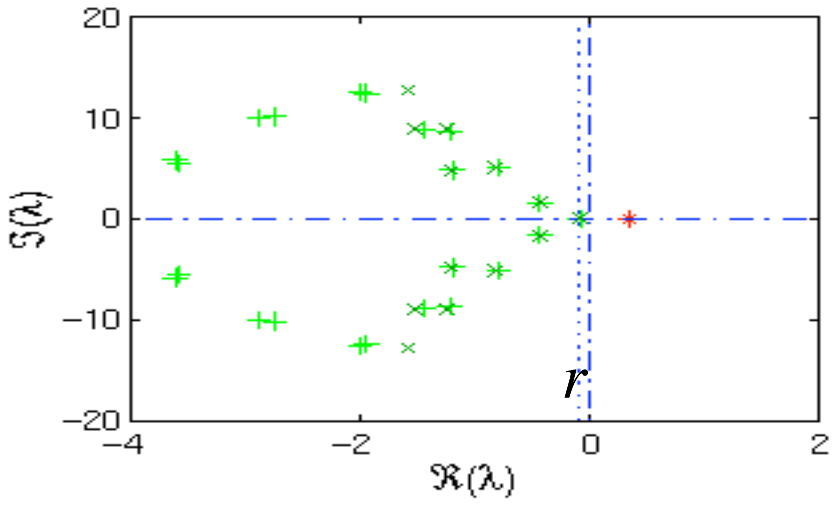
[Engelborgs & R., 2002] if step size of LMS method < h

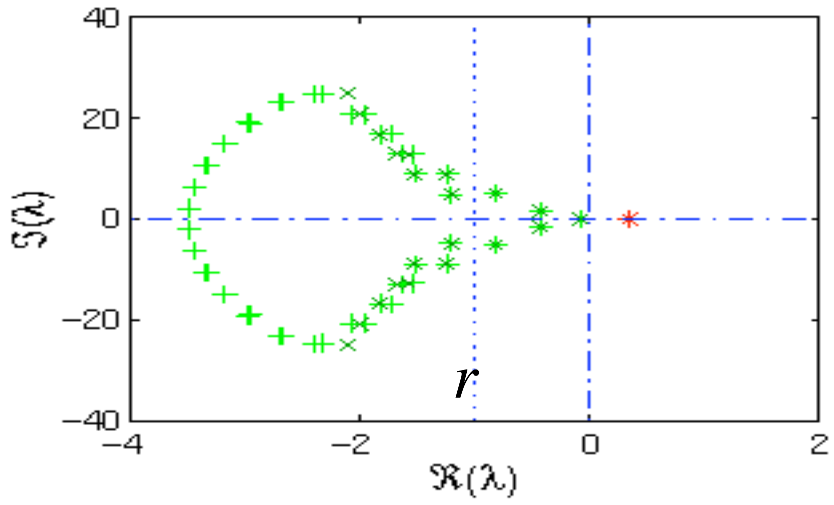
then delay independent stability is preserved in the discrete system ‘up to ε’

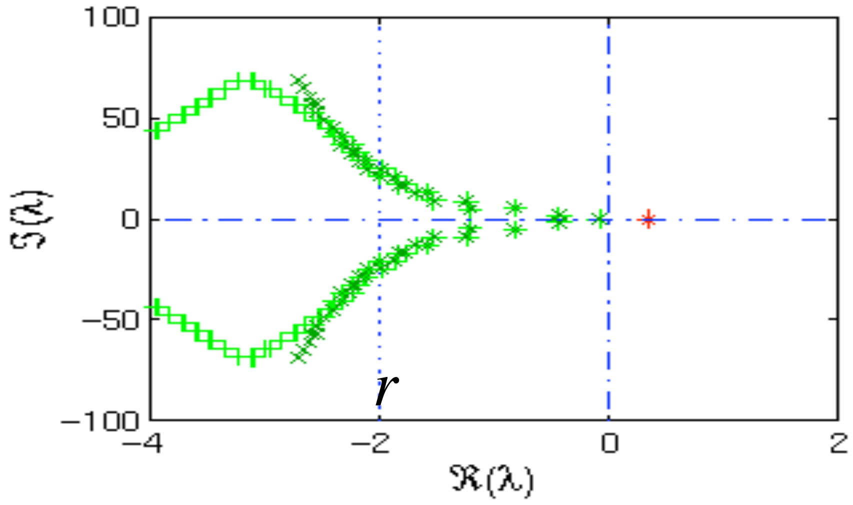


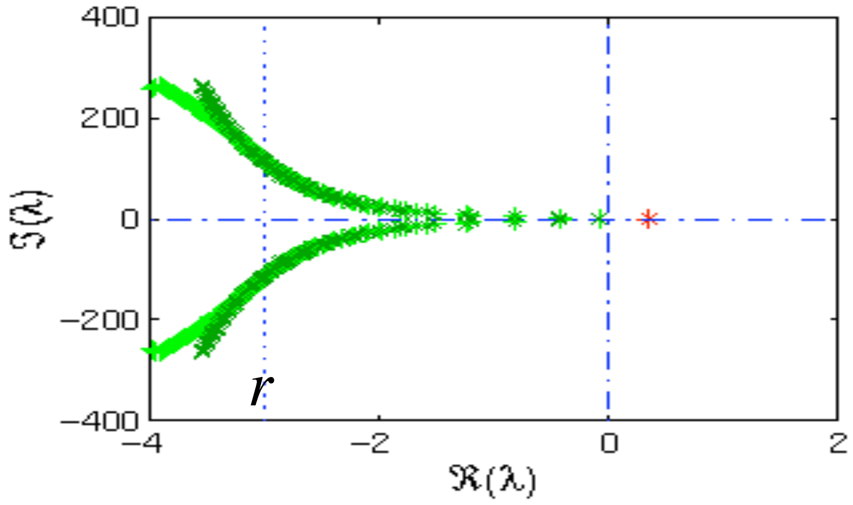
Computed eigenvalues: examples

x: exact eigenvalues +: computed eigenvalues









Improved step length heuristic

- Heuristic implemented in original version of DDE-BIFTOOL:

  robust, but too conservative
  too many roots are computed accurately
  $\rightarrow$ *h* too small, large eigenvalue problem, expensive

- *Current version of* DDE-BIFTOOL*: improved heuristic:*

  larger h, cheaper procedure

Improved step length heuristic

$$h = 0.9 \frac{\rho_{LMS}}{\| A_0 \| + | r | + \sum \| A_i \| \exp(-r\tau_i)}$$

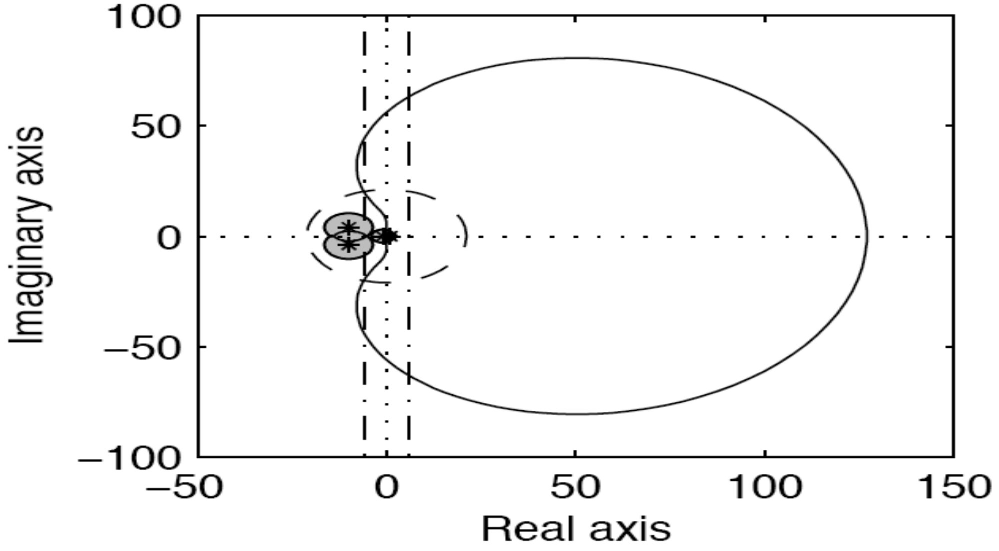
Towards larger h

- Numerator: region in which eigenvalues are preserved by LMS
  time integration properties not important
  $\rightarrow$ *special purpose LMS methods (of maximal order)*

- Denominator : boundary of region enclosing all λ with *Re(*λ*)* > *r*
  often large overestimation, especially when DDE system
  is discretization of PDE with delay ($\rightarrow$ ‘long tail’)
  $\rightarrow$ *more realistic bound*

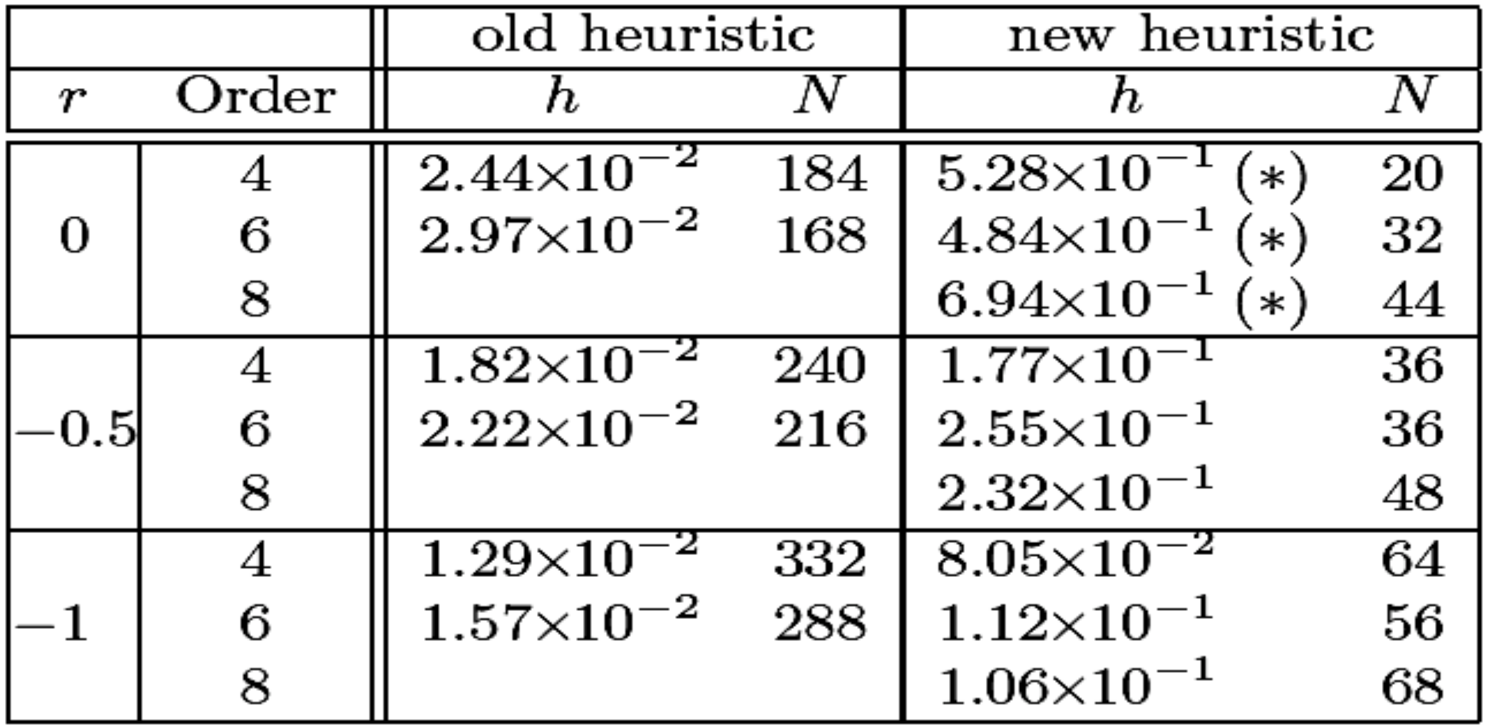
Example: 4 DDEs with 1 delay

$$A_0 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -10 & -4 \\ 0 & 0 & 4 & -10 \end{bmatrix}, \qquad A_1 = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 0 & -1.5 & 0 & 0 \\ 0 & 0 & 3 & -5 \\ 0 & 5 & 5 & 5 \end{bmatrix}$$



	$r=0$	$r=-0.5$	$r=-1$
$\lVert A_0\rVert + \lvert r\rvert + \sum_{j=1}^{m} \lVert A_j\rVert e^{-r\tau_j}$	21.1	28.3	39.9
$\max \lvert \Omega(r\vec{\tau}) \cap (\mathbb{C}^+ + r - \Delta r)\rvert$	2.86	8.58	18.8

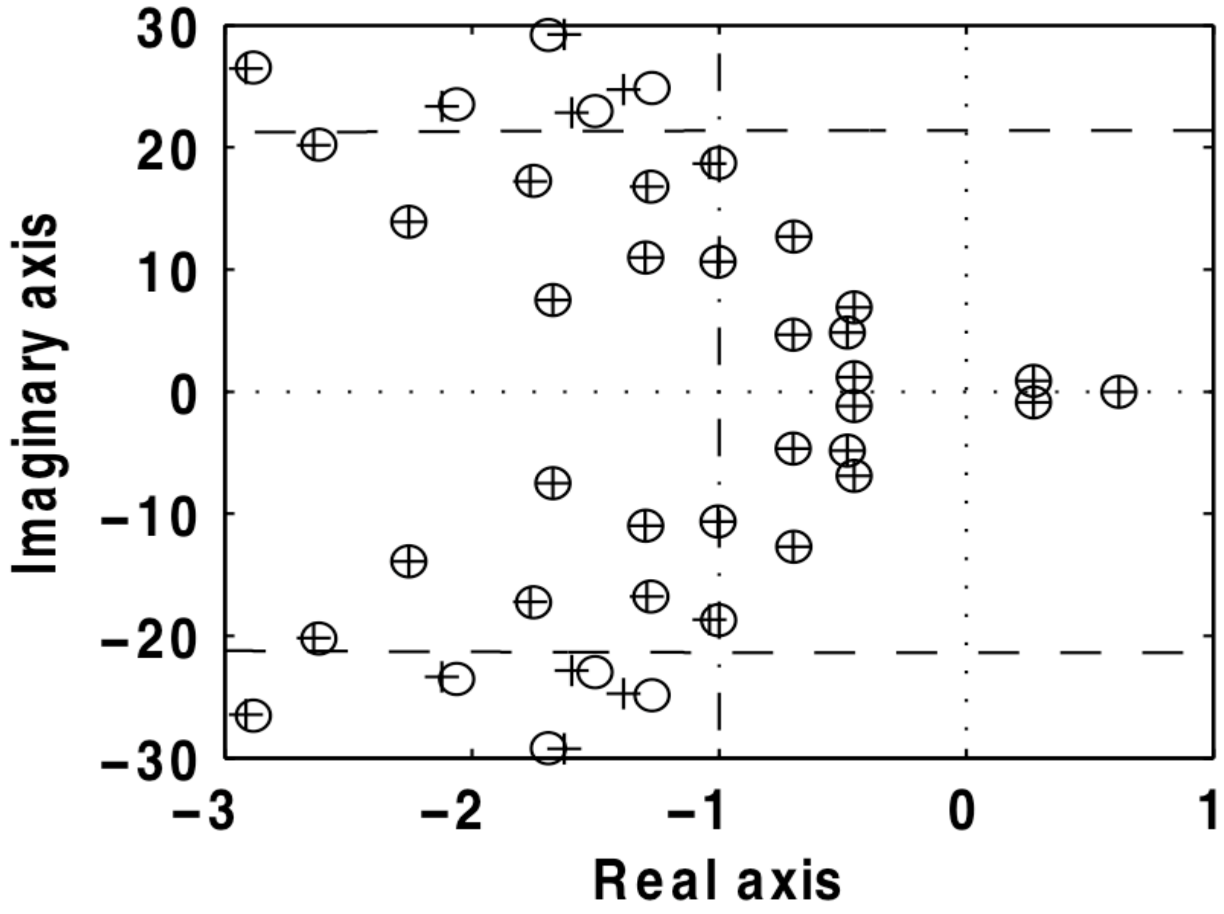
Example: 4 DDEs with 1 delay



		old heuristic		new heuristic	
r	Order	h	N	h	N
0	4	2.44×10^{-2}	184	5.28×10^{-1} $(*)$	20
	6	2.97×10^{-2}	168	4.84×10^{-1} $(*)$	32
	8			6.94×10^{-1} $(*)$	44
-0.5	4	1.82×10^{-2}	240	1.77×10^{-1}	36
	6	2.22×10^{-2}	216	2.55×10^{-1}	36
	8			2.32×10^{-1}	48
-1	4	1.29×10^{-2}	332	8.05×10^{-2}	64
	6	1.57×10^{-2}	288	1.12×10^{-1}	56
	8			1.06×10^{-1}	68

Example: 4 DDEs with 1 delay

Computed approximate roots and corrected roots (Newton on characteristic equation)



8^{th} order, $r = -1$, $h = 1.06 \times 10^{-1}$

Large scale DDE

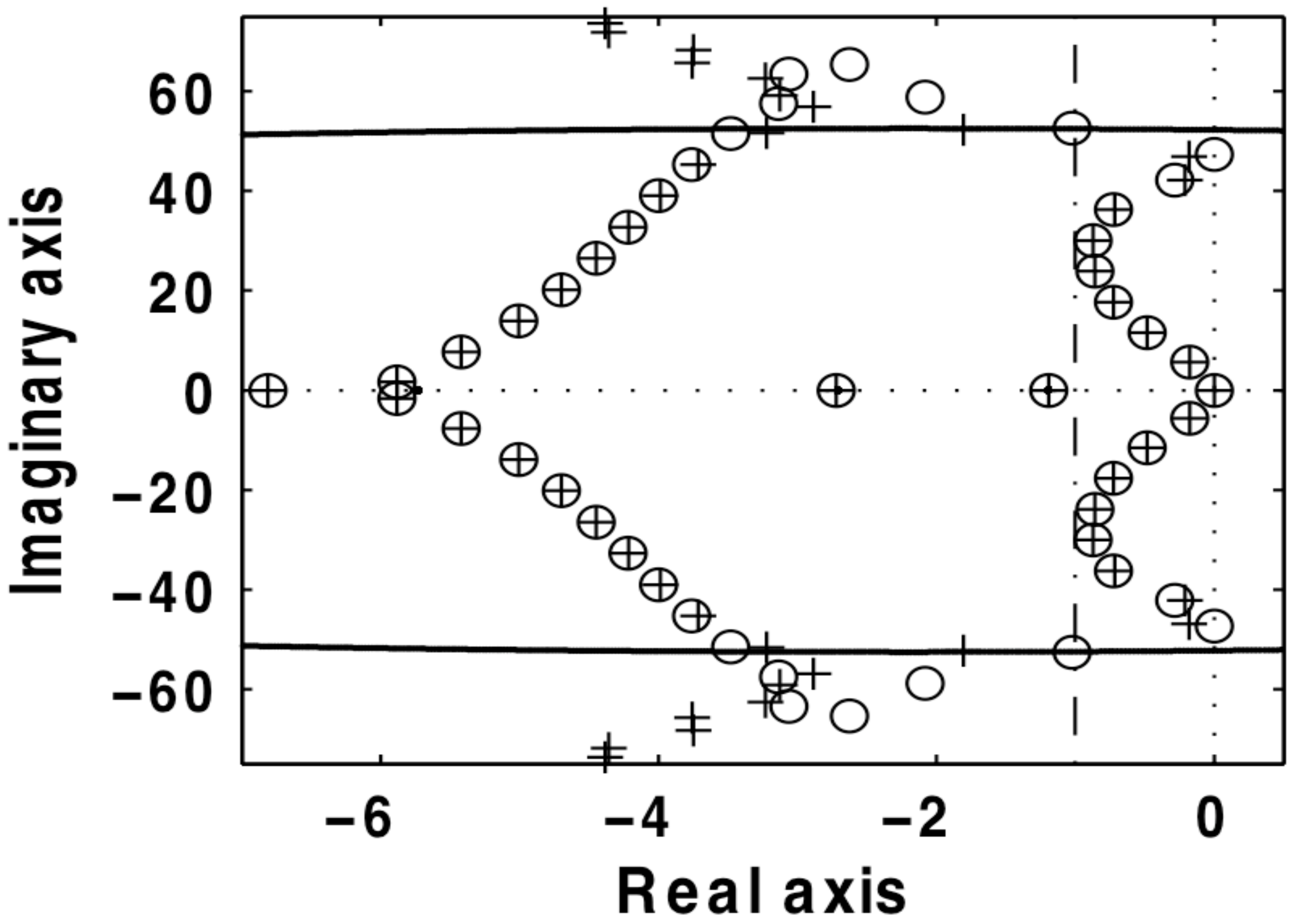
- system of DDE and PDE (laser dynamics)

$$\begin{aligned}
\frac{\mathrm{d}A(t)}{\mathrm{d}t} &= (1-\mathrm{i}\alpha)A(t)\zeta(t)+\eta A(t-\tau)e^{-\mathrm{i}\phi}-\mathrm{i}bA(t) \\
T\frac{\partial Z(x,t)}{\partial t} &= d\frac{\partial^2 Z(x,t)}{\partial x^2}-Z(x,t)+P(x) \\
&\quad -F(x)(1+2Z(x,t))|A(t)|^2.
\end{aligned}$$

- spatial variable *x* — functions $\zeta(t)$, $P(x)$ and $F(x)$
- 2d order finite diff. discretization in space
- resulting DDE system: dimension *n* =131
- parameters such that close to Hopf bifurcation (with large ω)
- spectrum : long tail !
  old heuristic with BDF order 6 : very small *h*
  $\rightarrow$ size eigenvalue problem *N* = ± 1 000 000
- new heuristic : *N* = ± 3 500

Large scale DDE

- Computed approximate roots and corrected roots (Newton on characteristic equation)



Other approaches

B) discretization of solution operator
using pseudospectral approximation

C) discretization of infinitisimal generator
– using time integrator (LMS or Runge-Kutta)
– using pseudospectral approximation

Spectral discretization

- solution operator can be discretized by pseudospectral discretization (polynomial of high degree instead of points on uniform mesh) $\rightarrow$ matrix eigenvalue problem
- *asymptotic* convergence properties better than for LMS methods
- for relatively low accuracy: both methods lead to a matrix eigenvalue problem of similar size
- but no automatic selection of appropriate degree of polynomial

Infinitisimal generator

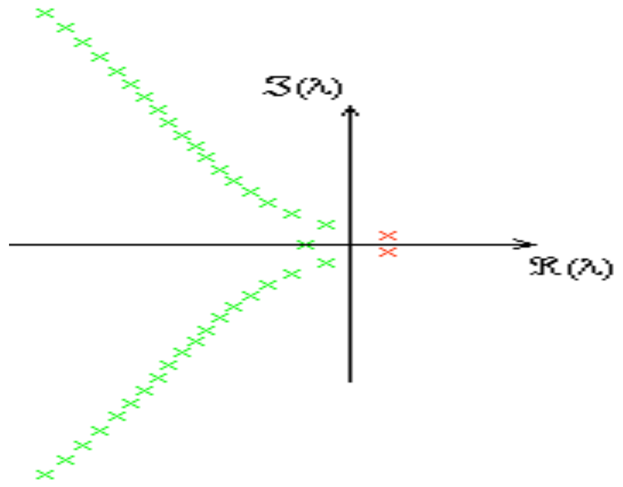
Since $S(t)$ is a strongly continuous semi-group, one can define the corresponding infinitesimal generator A by

$$\mathcal{A}y = \lim_{t \to 0+} \frac{\mathcal{S}(t)y - y}{t}$$

For variat. eq. the infinitesimal generator becomes

$$\mathcal{A}y(\theta) = y'(\theta), y \in \mathcal{D}(\mathcal{A})$$
$$\mathcal{D}(\mathcal{A}) = \{y \in C : y' \in C \quad \text{and} \quad y'(0) = \textstyle\sum_{j=0}^{m} A_j y(-\tau_j)\}$$

eigenvalues of A ≡
roots of characteristic eq.



)

Computation of eigenvalues of *A*

- discretise A into matrix $A(h)$
- calculate (rightmost) eigenvalues of $A(h)$
- (correct via Newton on characteristic equation)

Discretisation of *A* [Breda, Maset and Vermiglio]

- discretise C into vector space X_N
  mesh: equidistant or not
- approximate d$y\,/d\theta$
  - pseudo-spectral discretisation
  - time integration methods
    - LMS (k steps BDF)
    - Runge-Kutta (Radau II)

Pseudo-spectral discretization

Breda et al.: pseudo-spectral discretization of the infinitesimal generator.
An eigenfunction of the infinitesimal generator $ve^{\lambda t}$, *t in* $[-\tau, 0]$, is approximated by a polynomial $P(t)$ of degree p.
Collocation for the eigenvalue problem for the infinitisimal generator leads to an equation of the form

$$P'(t_i) = \lambda P(t_i),$$

collocation points t_i, $i = 1 \ldots p$ are chosen as the shifted and scaled roots of an (orthogonal) polynomial of degree p.
System-specific information

$$A_0 P(0) + \sum_{j=1}^{m} A_j P(\tau_j) = \lambda P(0),$$

Pseudo-spectral discretization

The resulting matrix eigenvalue problem has size $n(p+1)$

The matrix is full but can be of much smaller size than in the previous case,due to the ‘spectral accuracy’ convergence

Pseudo-spectral discretisation (cont.)

Convergence analysis

$$\max_{1\le i\le \nu} \left|\lambda_{exact} - \lambda_i\right| = O\left(\left(\frac{C}{N}\right)^{\frac{p}{\nu}}\right) = O\left(\left(\frac{Ch}{\tau}\right)^{\frac{p}{\nu}}\right)$$

ν : multiplicity of λ_{exact}

with	$p = k$	BDF
	$p = 2s\text{-}1$	Runge-Kutta
	$p = N$	Pseudo-spectral

Software packages

• DDE-BIFTOOL	K. Engelborghs et al
• PDDE-CONT	R. Szalai

DDE-BIFTOOL

- Functionality
  - *no time integration (use Matlab dde23 or ARCHI, DKLAG6, XPPAUT, DDVERK, ...)*
  - continuation of steady state & periodic solutions of DDEs with constant & state-dependent delays (no branch switching)
  - computation of stability of solutions monitoring of relevant eigenvalues

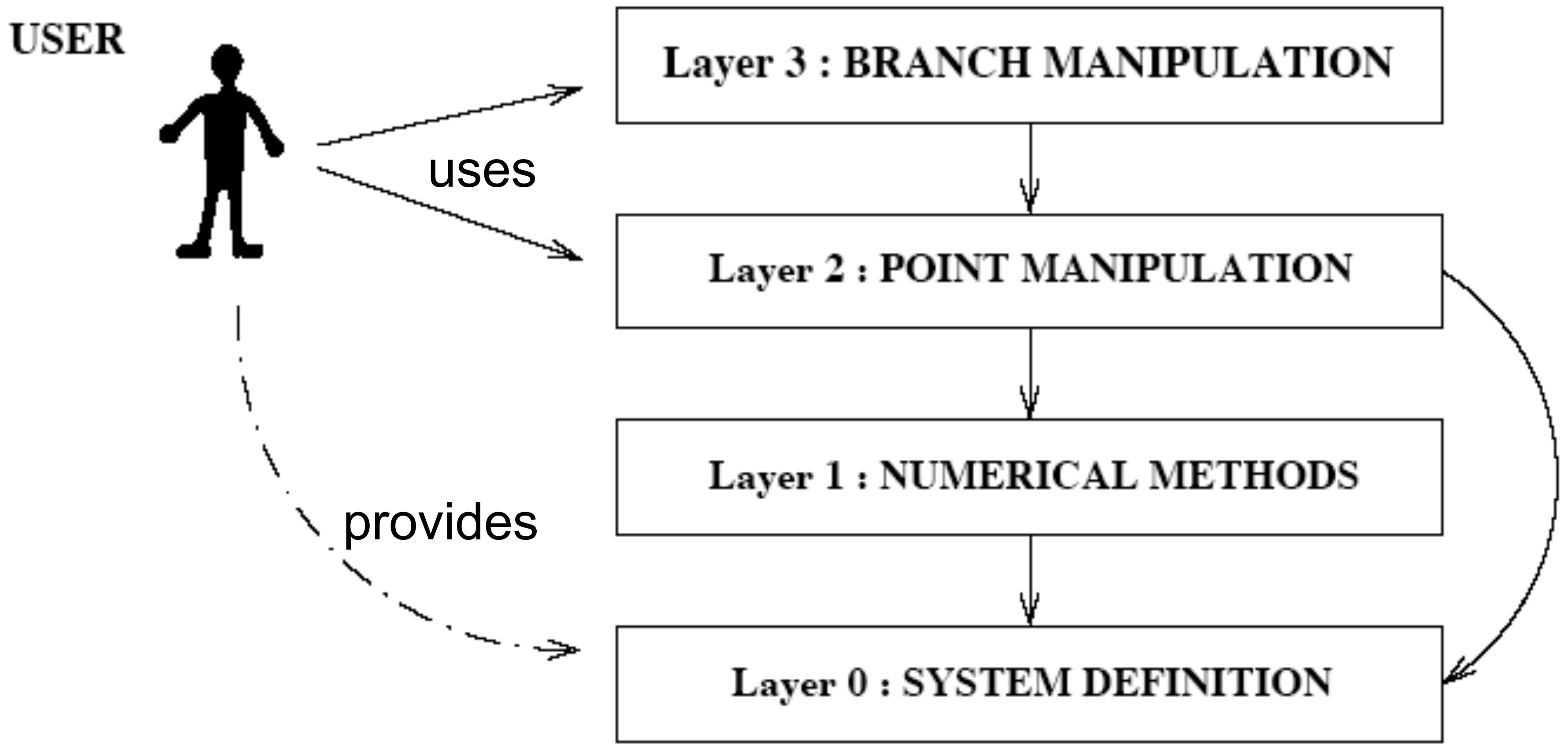
    (no automatic detection of bifurcation points)
  - continuation of fold and Hopf points
  - continuation of homo- & heteroclinic orbits
  - no normal forms ...

DDE-BIFTOOL

- Implementation
  - a set of Matlab routines
  - can be adapted and extended easily

  - no GUI, ‘command line’ Matlab commands
  - graphical output from Matlab
  - user has to provide the system equations (and derivatives) and to write (interactively)
    a ‘high level’ program
- Availability
  - free for research purposes

Structure of DDE-BIFTOOL


USER
uses
provides
Layer 3 : BRANCH MANIPULATION
Layer 2 : POINT MANIPULATION
Layer 1 : NUMERICAL METHODS
Layer 0 : SYSTEM DEFINITION

DDE-BIFTOOL: numerical methods

- stability of *steady states*: discretization of solution operator by LMS method; automatic procedure to approx. accurately all eigenvalues with real part > r (r : user defined)

  approximate eigenvalues corrected by Newton iteration on characteristic equation

- *periodic solutions* and stability: based on collocation

- *continuation*:
  secant prediction, pseudo-arclength, Newton correction, steplength strategy based on extrapolations/interpolations

- *determining systems* for fold, Hopf, ...

Usage of DDE-BIFTOOL

Layer 0 system definitions: user provides

- sys_init.m (path)
- sys_rhs.m (system eqs.)
- sys_deri.m (derivatives, or use sys_deriv.m)
- sys_tau.m

only in special cases

- sys_cond.m (extra conditions, e.g. to enforce unique solution)

only for state-dependent delays

- sys_ntau.m

Structure of DDE-BIFTOOL

Layer 2 routines to manipulate individual points

- point types: determines which information is stored
  - stst (steady state): parameter, state
  - hopf : parameter, state, ω, eigenvector
  - fold
  - psol : periodic orbit : ...., degree of collocation polynomial, mesh
  - hcli : homoclinic or heteroclinic orbit
- additional : stability information
- routines to correct points, compute & plot stability, convert type and correct

immediate acess to all 'points' via matlab command line

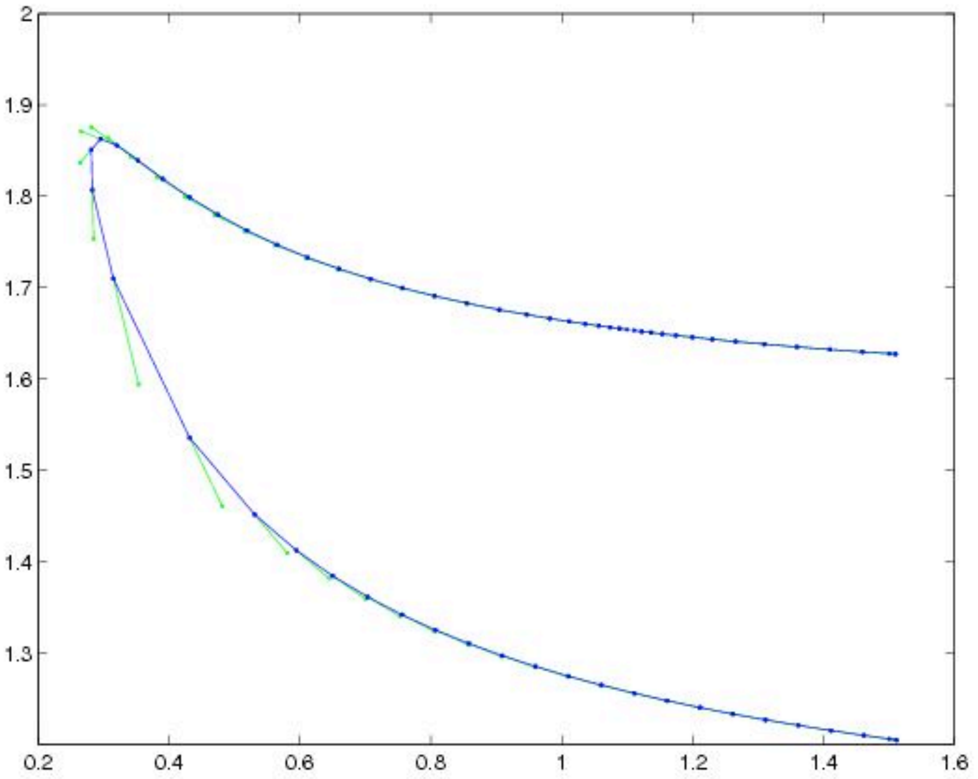
Structure of DDE-BIFTOOL

Layer 3 branches

- branch = array of points; method parameters; free parameters
- method parameters : data structure, 3 substructures:
  - point : *st.st.* : max. Newton iterations, accuracy, ...
    periodic: extra: phase cond., collocation par.
  - continuation strategy
  - stability computation : '*r*' (eigenvalues real part > *r*)
- free parameters: parameters bounds; max. step sizes
- routines to extend branch, to compute stability, to visualize branch & stability

Example of DDE-BIFTOOL output

One parameter branch of steady state solutions

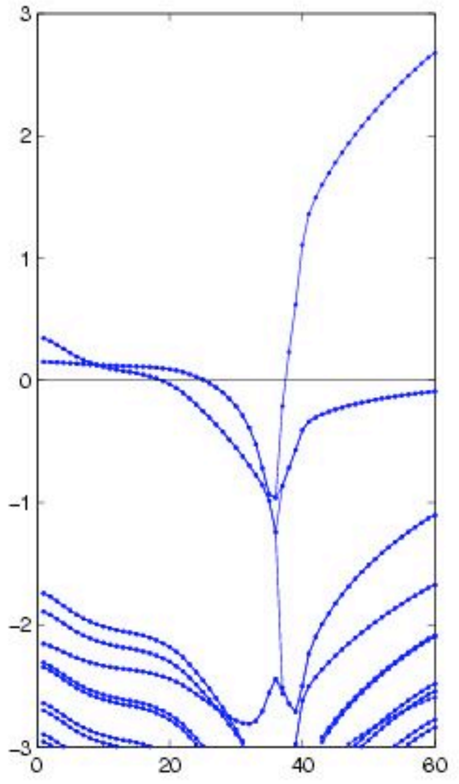


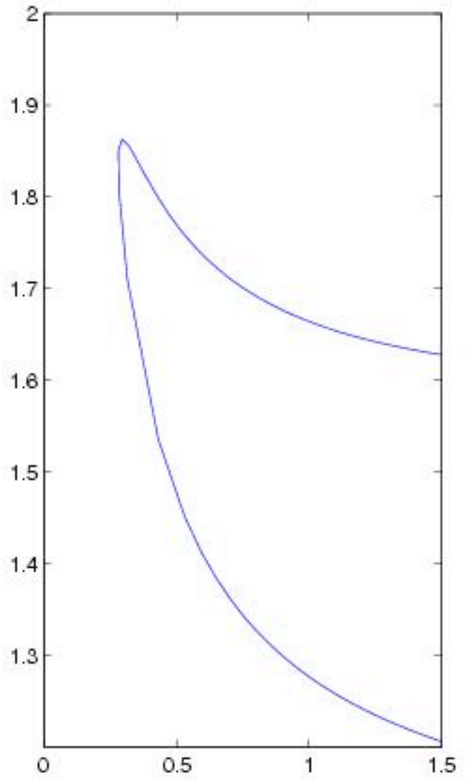
Prediction steps shown in green; corrected points in blue

Example of DDE-BIFTOOL output

Stability along the branch can then be computed

real part of roots

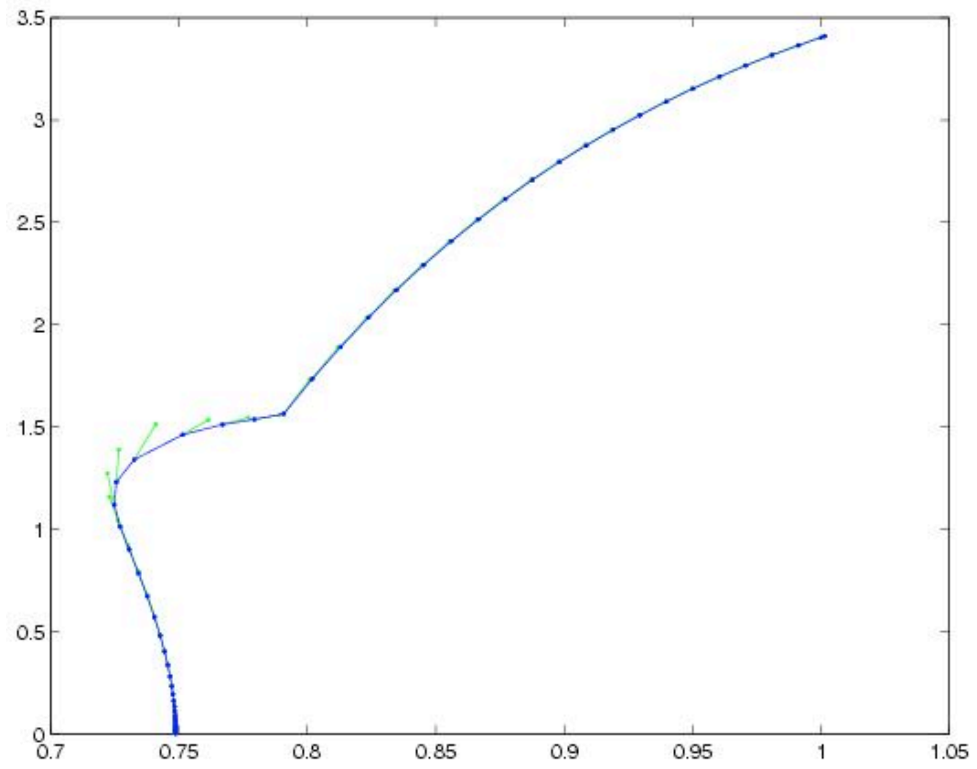




Crossings of the imaginary axis show bifurcations against point number (left) along the branch (right); stabilising Hopf bifurcation at point 26.

Example of DDE-BIFTOOL output

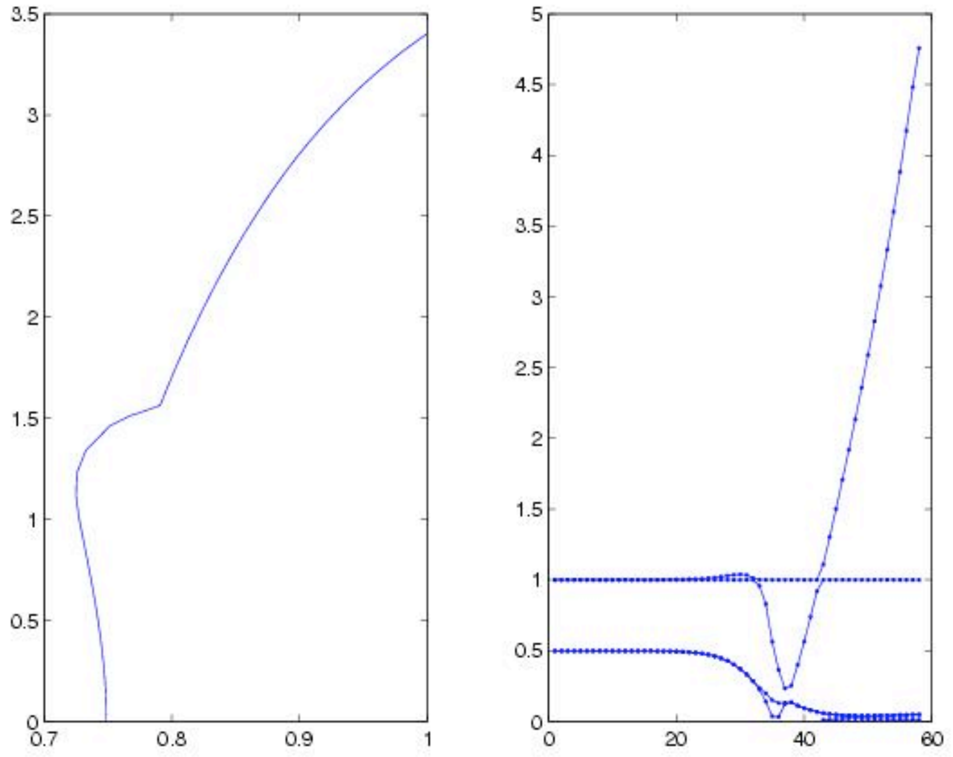
From this Hopf bifurcation, a one parameter branch of periodic solutions can be computed



(The shape of this branch indicates the Hopf bifurcation was subcritical)

Example of DDE-BIFTOOL output

Again, stability along the branch can then be computed

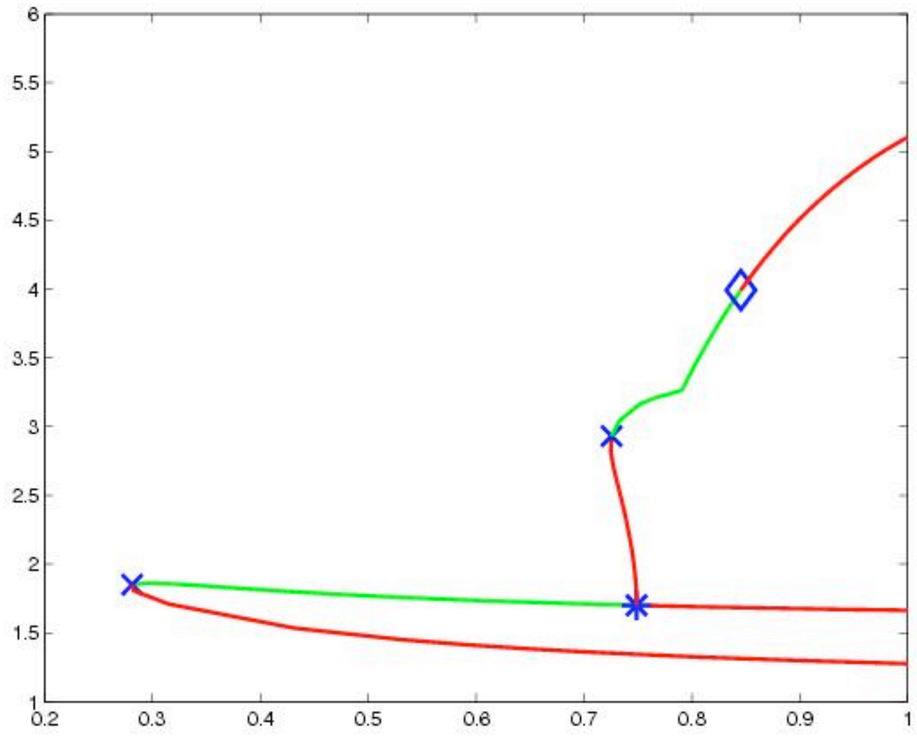


trivial Floquet multiplier

Crossings of the unit circle show bifurcations against point number (right) along the branch (left); stabilising saddle-node bifurcation at point 33.

Example of DDE-BIFTOOL output

Final branch plotted manually



Lower branch shows stable steady state (green), born in a saddle-node bifurcation (x), destabilised in a Hopf bifurcation (*). Initially unstable branch of periodic solutions stabilised in saddle-node bifurcation of limit cycles (x) and destabilised in a period-doubling bifurcation (diamond).

DDE-BIFTOOL run

branch of steady state solutions

- generate 1st point (build data structure)
  set method parameters
  correct steady state solution
  [set method parameters & compute stability]
- copy 1st point into 2d point
  change parameter
  correct steady state solution
- build brach with 2 points
  set method parameters (incl. continuation parameter)