A projection method for the eigenvalues of a delay-differential equation

8th GAMM Workshop - Applied and numerical linear algebra
Hamburg, 11-12th September, 2008

Elias Jarlebring
Institut *Computational Mathematics*
Technische Universität Braunschweig

Delay-differential equations (DDEs)

Retarded DDE with a single delay

$$\Sigma = \begin{cases} \dot{x}(t) = A_0 x(t) + A_1 x(t-\tau), \ t \geq 0 \\ x(t) = \varphi(t), \ t \in [-\tau, 0] \end{cases} \quad (1)$$

where $A_0, A_1 \in \mathbb{R}^{n \times n}$ and $\varphi : [-\tau, 0] \to \mathbb{R}^n$.

Delay-differential equations (DDEs)

Retarded DDE with a single delay

$$\Sigma = \begin{cases} \dot{x}(t) = A_0 x(t) + A_1 x(t-\tau), \ t \geq 0 \\ x(t) = \varphi(t), \ t \in [-\tau, 0] \end{cases} \tag{1}$$

where $A_0, A_1 \in \mathbb{R}^{n\times n}$ and $\varphi : [-\tau, 0] \to \mathbb{R}^n$.
Ansatz: $x(t) = ve^{st} \Rightarrow$

$$\left(-sI + A_0 + A_1 e^{-s\tau}\right) v = 0$$

Delay-differential equations (DDEs)

Retarded DDE with a single delay

$$\Sigma = \begin{cases} \dot{x}(t) = A_0 x(t) + A_1 x(t-\tau),\ t \geq 0 \\ x(t) = \varphi(t),\ t \in [-\tau, 0] \end{cases} \quad (1)$$

where $A_0, A_1 \in \mathbb{R}^{n\times n}$ and $\varphi : [-\tau, 0] \to \mathbb{R}^n$.
Ansatz: $x(t) = ve^{st} \Rightarrow$

$$\left(-sI + A_0 + A_1 e^{-s\tau}\right) v = 0$$

Definition

Spectrum: $\sigma(\Sigma) := \{s \in \mathbb{C} :\ \det\left(-sI + A_0 + A_1 e^{-s\tau}\right) = 0\}$

The hot-shower example

- $x(t) =$ Temp diff from optimal Temp

The hot-shower example

- $x(t) =$ Temp diff from optimal Temp
- $\alpha =$ Human sensitivity

The hot-shower example

- $x(t) =$ Temp diff from optimal Temp
- $\alpha =$ Human sensitivity
- $\tau = \frac{\text{Length of pipe}}{\text{Speed of water}}$

The hot-shower example

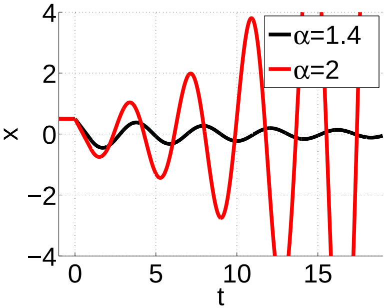
- $x(t) =$ Temp diff from optimal Temp
- $\alpha =$ Human sensitivity
- $\tau = \frac{\text{Length of pipe}}{\text{Speed of water}}$

Delayed negative feedback: $\dot{x}(t) = -\alpha x(t - \tau)$

The hot-shower example

- $x(t) =$ Temp diff from optimal Temp
- $\alpha =$ Human sensitivity
- $\tau = \frac{\text{Length of pipe}}{\text{Speed of water}}$

Delayed negative feedback: $\dot{x}(t) = -\alpha x(t-\tau)$, $\tau = 1$

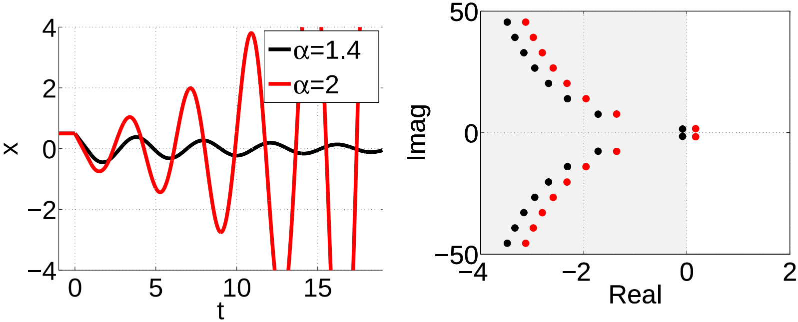


Solution $x(t)$

The hot-shower example

- $x(t)$ = Temp diff from optimal Temp
- α = Human sensitivity
- $\tau = \frac{\text{Length of pipe}}{\text{Speed of water}}$

Delayed negative feedback: $\dot{x}(t) = -\alpha x(t-\tau)$, $\tau = 1$



Solution $x(t)$

Spectrum
$\sigma(\Sigma) = \{s \in \mathbb{C} : -s - \alpha e^{-\tau s} = 0\}$

Computing the spectrum

Example ([Ebenbauer, Allgöwer ’06])

Computing the spectrum

Example ([Ebenbauer, Allgöwer '06])

Spectrum $\sigma(\Sigma)$: $s \in \mathbb{C}$ such that

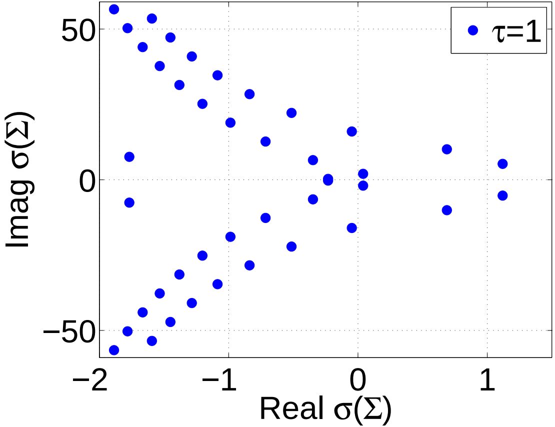
$$\det\left(-sI + \begin{pmatrix} -1 & 13.5 & -1 \\ -3 & -1 & -2 \\ -2 & -1 & -4 \end{pmatrix} + \begin{pmatrix} -5.9 & 7.1 & -70.3 \\ 2 & -1 & 5 \\ 2 & 0 & 6 \end{pmatrix} e^{-\tau s}\right) = 0$$

Computing the spectrum

Example ([Ebenbauer, Allgöwer '06])

Spectrum $\sigma(\Sigma)$: $s \in \mathbb{C}$ such that

$$\det\left(-sI + \begin{pmatrix} -1 & 13.5 & -1 \\ -3 & -1 & -2 \\ -2 & -1 & -4 \end{pmatrix} + \begin{pmatrix} -5.9 & 7.1 & -70.3 \\ 2 & -1 & 5 \\ 2 & 0 & 6 \end{pmatrix} e^{-\tau s}\right) = 0$$



Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Method 1: Lambert W

Analytic method for scalar and simultaneously triangularizable systems.

Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Method 1: Lambert W

Analytic method for scalar and simultaneously triangularizable systems.

Method 2: Solution operator

Numerical method based on the discretization of the solution operator.

Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Method 1: Lambert W

Analytic method for scalar and simultaneously triangularizable systems.

Method 2: Solution operator

Numerical method based on the discretization of the solution operator.

Method 3: Infinitesimal generator

Numerical method based on the discretization of the infinitesimal generator.

Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Method 1: Lambert W

Analytic method for scalar and simultaneously triangularizable systems.

Method 2: Solution operator

Numerical method based on the discretization of the solution operator.

Method 3: Infinitesimal generator

Numerical method based on the discretization of the infinitesimal generator.

Method 4: Subspace accelerated residual inverse iteration

A projection method based on residual inverse iteration.

Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Method 1: Lambert W

Analytic method for scalar and simultaneously triangularizable systems.

Method 2: Solution operator

Numerical method based on the discretization of the solution operator.

Method 3: Infinitesimal generator

Numerical method based on the discretization of the infinitesimal generator.

Method 4: Subspace accelerated residual inverse iteration

A projection method based on residual inverse iteration.

- Two examples indicate that Method 4 should be used for large problems.

Methods for $\sigma(\Sigma) := \{s \in \mathbb{C} : \det(-sI + A_0 + A_1 e^{-s\tau}) = 0\}$

Method 1: Lambert W

Analytic method for scalar and simultaneously triangularizable systems.

Method 2: Solution operator

Numerical method based on the discretization of the solution operator.

Method 3: Infinitesimal generator

Numerical method based on the discretization of the infinitesimal generator.

Method 4: Subspace accelerated residual inverse iteration

A projection method based on residual inverse iteration.

- Two examples indicate that Method 4 should be used for large problems.
- Previously in the literature $n = 131$, Method 4: $n = 10^6$.

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Multiply with $\tau e^{\tau s - \tau a_0}$ and rearrange terms $\Rightarrow$

$$\tau a_1 e^{-\tau a_0} = \tau(s - a_0)e^{\tau(s-a_0)}$$

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Multiply with $\tau e^{\tau s - \tau a_0}$ and rearrange terms $\Rightarrow$

$$\tau a_1 e^{-\tau a_0} = \tau(s - a_0) e^{\tau(s-a_0)}$$

Lambert W: Multivalued inverse of we^w, i.e., $W_k(x)e^{W_k(x)} = x$

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Multiply with $\tau e^{\tau s - \tau a_0}$ and rearrange terms $\Rightarrow$

$$\tau a_1 e^{-\tau a_0} = \tau(s - a_0)e^{\tau(s-a_0)}$$

Lambert W: Multivalued inverse of we^w, i.e., $W_k(x)e^{W_k(x)} = x$
Matlab & Maple:`lambertw`
[Corless et al '96][Jeffrey et al 96]

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Multiply with $\tau e^{\tau s - \tau a_0}$ and rearrange terms $\Rightarrow$

$$\tau a_1 e^{-\tau a_0} = \tau(s - a_0)e^{\tau(s-a_0)}$$

Lambert W: Multivalued inverse of we^w, i.e., $W_k(x)e^{W_k(x)} = x$
Matlab & Maple:`lambertw`
[Corless et al '96][Jeffrey et al 96]

$$W_k(\tau a_1 e^{-\tau a_0}) = W_k\left(\tau(s - a_0)e^{\tau(s-a_0)}\right)$$

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Multiply with $\tau e^{\tau s - \tau a_0}$ and rearrange terms $\Rightarrow$

$$\tau a_1 e^{-\tau a_0} = \tau(s - a_0)e^{\tau(s-a_0)}$$

Lambert W: Multivalued inverse of we^w, i.e., $W_k(x)e^{W_k(x)} = x$
Matlab & Maple:`lambertw`
[Corless et al '96][Jeffrey et al 96]

$$W_k(\tau a_1 e^{-\tau a_0}) = W_k\left(\tau(s - a_0)e^{\tau(s-a_0)}\right) = \tau(s - a_0)$$

Method 1: The Lambert W function

Scalar single delay:

$$-s + a_0 + a_1 e^{-s\tau} = 0$$

Multiply with $\tau e^{\tau s - \tau a_0}$ and rearrange terms $\Rightarrow$

$$\tau a_1 e^{-\tau a_0} = \tau(s - a_0)e^{\tau(s-a_0)}$$

Lambert W: Multivalued inverse of we^w, i.e., $W_k(x)e^{W_k(x)} = x$
Matlab & Maple:`lambertw`
[Corless et al '96][Jeffrey et al 96]

$$W_k(\tau a_1 e^{-\tau a_0}) = W_k\left(\tau(s - a_0)e^{\tau(s-a_0)}\right) = \tau(s - a_0) \Rightarrow$$

Well known closed form: $s = \frac{1}{\tau} W_k(\tau a_1 e^{-\tau a_0}) + a_0$
[Corless et al '96], [Shinozaki, Mori '06]

Problem

Does the formula

$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \frac{1}{\tau} W_k(\tau a_1 e^{-\tau a_0}) + a_0$$

hold for systems with $n > 1$?

We have shown in [J.,Damm '07]:

Problem

Does the formula

$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \frac{1}{\tau} W_k(\tau a_1 e^{-\tau a_0}) + a_0$$

hold for systems with $n > 1$?

We have shown in [J.,Damm '07]:

Problem

Does the formula

$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \frac{1}{\tau} W_k(\tau a_1 e^{-\tau a_0}) + a_0$$

hold for systems with $n > 1$?

We have shown in [J.,Damm '07]:

i) Answer: Yes, if A_0, A_1 are *simultaneously triangularizable* or in particular if $A_0 A_1 = A_1 A_0$.

Problem

Does the formula

$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \frac{1}{\tau} W_k(\tau a_1 e^{-\tau a_0}) + a_0$$

hold for systems with $n > 1$?

We have shown in [J.,Damm '07]:

i) Answer: Yes, if A_0, A_1 are *simultaneously triangularizable* or in particular if $A_0A_1 = A_1A_0$.
ii) Answer: No, for the general case.

Problem

Does the formula

$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \frac{1}{\tau} W_k(\tau a_1 e^{-\tau a_0}) + a_0$$

hold for systems with $n > 1$?

We have shown in [J.,Damm '07]:

i) Answer: Yes, if A_0, A_1 are *simultaneously triangularizable* or in particular if $A_0 A_1 = A_1 A_0$.

ii) Answer: No, for the general case. Other literature [Asl, Ulsoy '03], with citation count 12, is not valid in the stated generality.

The Lambert W formula

Theorem ([J., Damm '07])

If A_0 and A_1 are simultaneously triangularizable, then

$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \sigma\left(\frac{1}{\tau}W_k(A_1\tau e^{-A_0\tau}) + A_0\right) .$$

The Lambert W formula

Theorem ([J., Damm ’07])

If A_0 and A_1 are simultaneously triangularizable, then

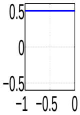
$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \sigma\left(\frac{1}{\tau} W_k(A_1 \tau e^{-A_0\tau}) + A_0\right) .$$

Corollary

If $A_0 A_1 = A_1 A_0$, then

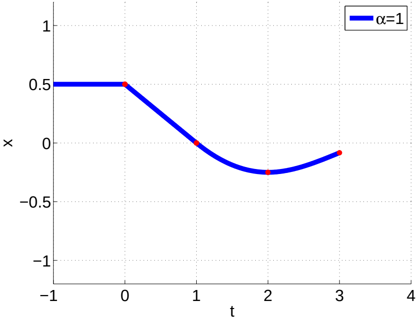
$$\sigma(\Sigma) = \bigcup_{k\in\mathbb{Z}} \sigma\left(\frac{1}{\tau} W_k(A_1 \tau e^{-A_0\tau}) + A_0\right) .$$

Method 2: Solution operator discretization (SOD)



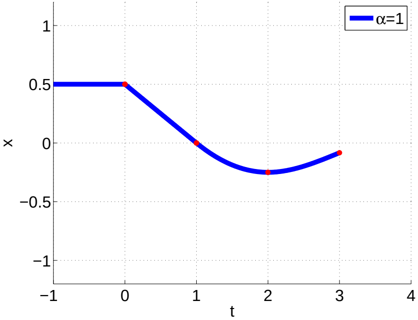
$$y^{(0)} = \varphi$$

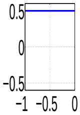
$$\dot{x}(t) = -\alpha x(t-1),\ \varphi(t) = 0.5$$



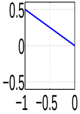
Method 2: Solution operator discretization (SOD)

$$\dot{x}(t) = -\alpha x(t-1),\ \varphi(t) = 0.5$$





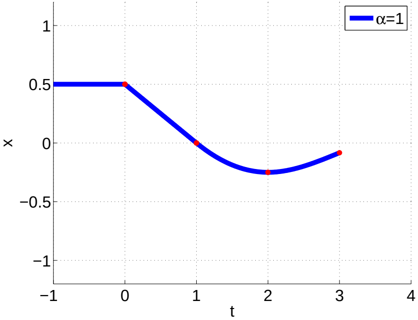
$$y^{(0)} = \varphi$$

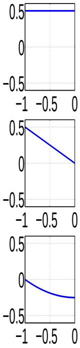


$$y^{(1)} = T(1)\varphi$$

Method 2: Solution operator discretization (SOD)

$$\dot{x}(t) = -\alpha x(t-1),\ \varphi(t) = 0.5$$





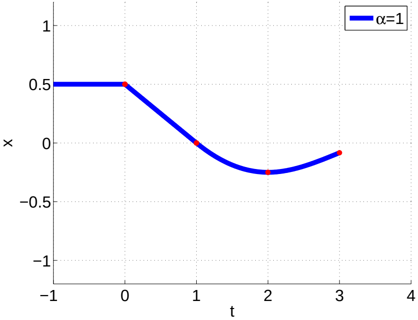
$$y^{(0)} = \varphi$$

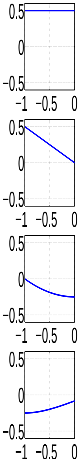
$$y^{(1)} = T(1)\varphi$$

$$y^{(2)} = T(2)\varphi$$

Method 2: Solution operator discretization (SOD)

$$\dot{x}(t) = -\alpha x(t-1),\ \varphi(t) = 0.5$$





$$y^{(0)} = \varphi$$

$$y^{(1)} = T(1)\varphi$$

$$y^{(2)} = T(2)\varphi$$

$$y^{(3)} = T(3)\varphi$$

Hot shower problem:

$$T(1)y = -\alpha \int_{-1}^{t} y(t)\, dt + y(0)$$

Hot shower problem:

$$T(1)y = -\alpha \int_{-1}^{t} y(t)\,dt + y(0)$$

More generally if $h \leq \tau$,

$$(T(h)\varphi)(\theta) = \begin{cases} \psi(\theta) = \varphi(\theta - h) & \theta \leq -h \\ \text{Solution of } \dot{\psi}(\theta) = A_0\psi(\theta) + A_1\varphi(\theta + h - \tau) & \theta > -h, \end{cases}$$

Discretize with Runge-Kutta or Linear Multistep, stepsize h
$\Rightarrow S_N \in \mathbb{R}^{nN \times nN}$.

Hot shower problem:

$$T(1)y = -\alpha \int_{-1}^{t} y(t)\,dt + y(0)$$

More generally if $h \leq \tau$,

$$(T(h)\varphi)(\theta) = \begin{cases} \psi(\theta) = \varphi(\theta - h) & \theta \leq -h \\ \text{Solution of } \dot{\psi}(\theta) = A_0\psi(\theta) + A_1\varphi(\theta + h - \tau) & \theta > -h, \end{cases}$$

Discretize with Runge-Kutta or Linear Multistep, stepsize h
$\Rightarrow S_N \in \mathbb{R}^{nN \times nN}$.

Hot shower problem:

$$T(1)y = -\alpha \int_{-1}^{t} y(t)\,dt + y(0)$$

More generally if $h \leq \tau$,

$$(T(h)\varphi)(\theta) = \begin{cases} \psi(\theta) = \varphi(\theta - h) & \theta \leq -h \\ \text{Solution of } \dot{\psi}(\theta) = A_0\psi(\theta) + A_1\varphi(\theta + h - \tau) & \theta > -h, \end{cases}$$

Discretize with Runge-Kutta or Linear Multistep, stepsize h
$\Rightarrow S_N \in \mathbb{R}^{nN \times nN}$.

Hot shower problem:

$$T(1)y = -\alpha \int_{-1}^{t} y(t)\,dt + y(0)$$

More generally if $h \leq \tau$,

$$(T(h)\varphi)(\theta) = \begin{cases} \psi(\theta) = \varphi(\theta - h) & \theta \leq -h \\ \text{Solution of } \dot{\psi}(\theta) = A_0\psi(\theta) + A_1\varphi(\theta + h - \tau) & \theta > -h, \end{cases}$$

Discretize with Runge-Kutta or Linear Multistep, stepsize h
$\Rightarrow S_N \in \mathbb{R}^{nN \times nN}$.

$$\Rightarrow \text{ If } s \in \frac{1}{h}\ln(\sigma(S_N)) \text{ and discretization sufficiently fine}$$

$$\text{then } s \approx z \text{ for some } z \in \sigma(\Sigma)$$

Hot shower problem:

$$T(1)y = -\alpha \int_{-1}^{t} y(t)\,dt + y(0)$$

More generally if $h \leq \tau$,

$$(T(h)\varphi)(\theta) = \begin{cases} \psi(\theta) = \varphi(\theta - h) & \theta \leq -h \\ \text{Solution of } \dot{\psi}(\theta) = A_0\psi(\theta) + A_1\varphi(\theta + h - \tau) & \theta > -h, \end{cases}$$

Discretize with Runge-Kutta or Linear Multistep, stepsize h
$\Rightarrow S_N \in \mathbb{R}^{nN\times nN}$.

$$\Rightarrow \text{ If } s \in \frac{1}{h}\ln(\sigma(S_N)) \text{ and discretization sufficiently fine}$$

$$\text{then } s \approx z \text{ for some } z \in \sigma(\Sigma)$$

LMS, DDE-BIFTOOL [Engelborghs, Roose '99, '02], RK [Breda '05]

SOD(LMS) is a Padé approximation of the logarithm

By change of variables:

$$\det(-sI + A_0 + A_1 e^{-\tau s}) = 0$$

SOD(LMS) is a Padé approximation of the logarithm

By change of variables:

$$\det(-sI + A_0 + A_1 e^{-\tau s}) = 0$$

$$\Leftrightarrow$$

$$\det\left(-\frac{1}{h}\mu^N \ln\mu + A_0\mu^N + A_1\right) = 0 \qquad (2)$$

where $\mu = e^{hs}$ and $h = \tau/N$.

SOD(LMS) is a Padé approximation of the logarithm

By change of variables:

$$\det(-sI + A_0 + A_1 e^{-\tau s}) = 0$$

$$\Leftrightarrow$$

$$\det\left(-\frac{1}{h}\mu^N \ln\mu + A_0\mu^N + A_1\right) = 0 \qquad (2)$$

where $\mu = e^{hs}$ and $h = \tau/N$.
Approximate $\ln(\mu)$ with (2,2)-Padé approximation in $\mu = 1$

$$\frac{\alpha(\mu)}{\beta(\mu)} = \frac{\mu^2 - 1}{\frac{1}{3}\mu^2 + \frac{4}{3}\mu + \frac{1}{3}}$$

$\Rightarrow$ (2) turns into *polynomial eigenvalue problem*

SOD(LMS) is a Padé approximation of the logarithm

By change of variables:

$$\det(-sI + A_0 + A_1 e^{-\tau s}) = 0$$

$$\Leftrightarrow$$

$$\det\left(-\frac{1}{h}\mu^N \ln\mu + A_0\mu^N + A_1\right) = 0 \qquad (2)$$

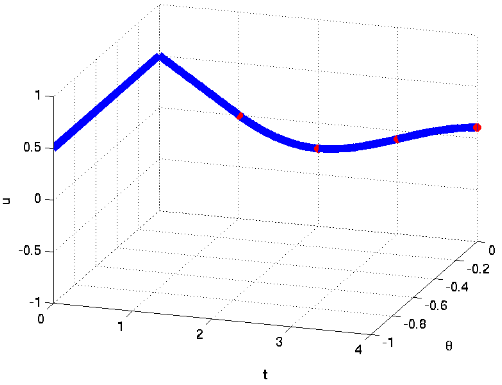
where $\mu = e^{hs}$ and $h = \tau/N$.
Approximate $\ln(\mu)$ with (2,2)-Padé approximation in $\mu = 1$

$$\frac{\alpha(\mu)}{\beta(\mu)} = \frac{\mu^2 - 1}{\frac{1}{3}\mu^2 + \frac{4}{3}\mu + \frac{1}{3}}$$

$\Rightarrow$ (2) turns into *polynomial eigenvalue problem*
Companion linearization $\Rightarrow$ LMS(Milne-Simpson) used in DDE-BIFTOOL.

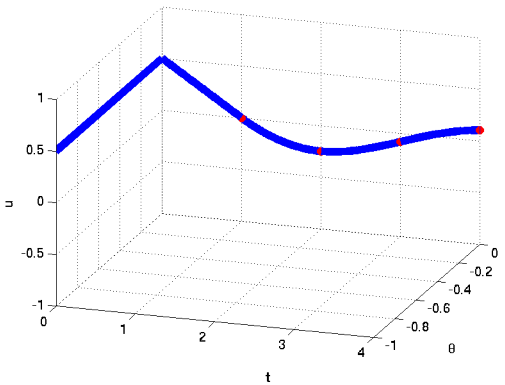
Method 3: Infinitesimal generator discretization (IGD)

PDE-representation: e.g. [Diekman, van Gils '95]



Method 3: Infinitesimal generator discretization (IGD)

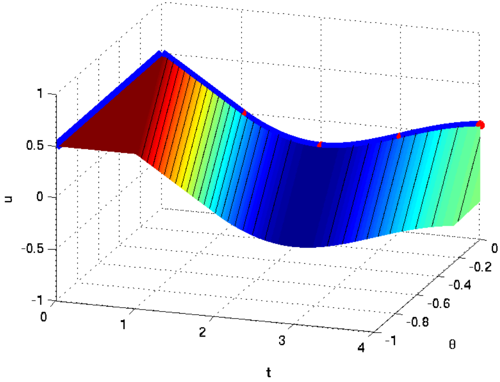
PDE-representation: e.g. [Diekman, van Gils '95]



$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial \theta}$$

Method 3: Infinitesimal generator discretization (IGD)

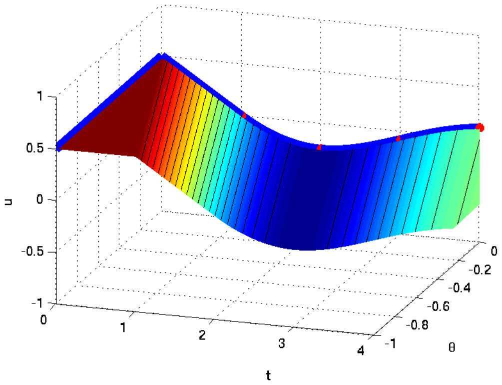
PDE-representation: e.g. [Diekman, van Gils '95]



$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial \theta}$$

Method 3: Infinitesimal generator discretization (IGD)

PDE-representation: e.g. [Diekman, van Gils '95]



$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial \theta}$$

IV: $u(0,\theta) = \varphi(\theta)$

BC: $\frac{\partial u}{\partial t}(t,0) = A_0 u(t,0) + A_1 u(t,-\tau)$

PDE:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial u}{\partial \theta} \\ \text{IV:} u(0,\theta) = \varphi(\theta) \\ \text{BC:} \dfrac{\partial u}{\partial t}(t,0) = A_0 u(t,0) + A_1 u(t,-\tau) \end{cases}$$

PDE:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial u}{\partial \theta} \\ \text{IV:} u(0,\theta) = \varphi(\theta) \\ \text{BC:} \dfrac{\partial u}{\partial t}(t,0) = A_0 u(t,0) + A_1 u(t,-\tau) \end{cases}$$

Semi-discretization with
Euler [Bellen, Maset '00],

PDE:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial u}{\partial \theta} \\ \text{IV:} u(0,\theta) = \varphi(\theta) \\ \text{BC:} \dfrac{\partial u}{\partial t}(t,0) = A_0 u(t,0) + A_1 u(t,-\tau) \end{cases}$$

Semi-discretization with
Euler [Bellen, Maset '00], or
Runge-Kutta [Breda, Maset, Vermiglio '03]

PDE:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial u}{\partial \theta} \\ \text{IV:} u(0,\theta) = \varphi(\theta) \\ \text{BC:} \frac{\partial u}{\partial t}(t,0) = A_0 u(t,0) + A_1 u(t,-\tau) \end{cases}$$

Semi-discretization with
Euler [Bellen, Maset '00], or
Runge-Kutta [Breda, Maset, Vermiglio '03]

Alternative derivation for [Bellen, Maset '00]:
Replace $e^{-\tau s}$ in $\sigma(\Sigma)$ with $(1+\tau s/N)^{-N}$ and *companion linearization*.

Method 4:Subspace accelerated residual inverse iteration

In the context of nonlinear eigenvalue problems (NLEVPs)

$$0 = (-sI + A_0 + A_1 e^{-\tau s})v = T(s)v.$$

Method 4:Subspace accelerated residual inverse iteration

In the context of nonlinear eigenvalue problems (NLEVPs)

$$0 = (-sI + A_0 + A_1 e^{-\tau s})v = T(s)v.$$

Residual inverse iteration [Neumaier '85]

Input: $v_0 \in \mathbb{C}^n$ shift $\mu \in \mathbb{C}$

1: **for** $k = 1 \ldots$ convergence **do**
2: Solve $v_k^* T(s) v_k = 0$ numerically
3: $v_{k+1} = v_k - T(\mu)^{-1} T(s) v_k$ and normalize
4: **end for**

Method 4:Subspace accelerated residual inverse iteration

In the context of nonlinear eigenvalue problems (NLEVPs)

$$0 = (-sI + A_0 + A_1 e^{-\tau s})v = T(s)v.$$

Residual inverse iteration [Neumaier '85]

Input: $v_0 \in \mathbb{C}^n$ shift $\mu \in \mathbb{C}$

for $k = 1 \ldots$ convergence **do**

  Solve $v_k^* T(s) v_k = 0$ numerically

  $v_{k+1} = v_k - T(\mu)^{-1} T(s) v_k$ and normalize

end for

Idea: Expand a subspace [Voss '04] [Meerbergen '01]

Subspace accelerated residual inverse iteration (SRI)

for all $k = 1 \ldots$ convergence **do**

  Solve $V^* T(s) V y = 0$ numerically

  $v = T(\mu)^{-1} T(s) V y$ and orthogonalize

  $V = [V, v/\|v\|]$

end for

Full featured SRI

Factorize $LU = PT(\sigma)Q$
Compute initial vectors V
for $m = 1 \ldots m_{max}$ **do**
  while *relres* $>$ *RESTOL* **do**
    Update *Aproj* such that $Aproj(k) = V^* A_k V$ for all k.
    Solve projected nonlinear eigenvalue problem

$$[\mu, y, \mu_{next}, y_{next}] = \mathrm{projectedsolve}(V, Aproj, \mu, \sigma, \sigma_t, V_l, \mu_l)$$

    where (μ, y) is the “best solution candidate”.
    Compute $v = T(\sigma)^{-1} T(\mu) V y$ with factorization of $T(\sigma)$.
    Orthogonalize v against V
    Expand search space $V = [V, v/\|v\|]$
  end while
  Store locked values $V_l = [V_l, v]$, $\mu_l = [\mu_l, \mu]$.
  If dimension $V >$ RESTARTDIMENSION restart:

$$[V, \mu] = \mathrm{restart}(V, V_l, \mu, y, \mu_{next}, y_{next})$$

end for

Example 1

A PDE with delay,

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + a_0(x)u + a_1(x)u(x, t-\tau), \\ \qquad u(0,t) = u(\pi, t) = 0,\ t \geq 0 \end{cases} \quad (3)$$

where $a_0(x) = a_0 + \alpha_0 \sin(x)$,
$a_1(x) = a_1 + \alpha_1 x(1 - e^{x-\pi}) + \alpha_2 x(\pi - x)$.

Example 1

A PDE with delay,

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + a_0(x)u + a_1(x)u(x, t-\tau), \\ \qquad\qquad u(0,t) = u(\pi,t) = 0,\ t \geq 0 \end{cases} \tag{3}$$

where $a_0(x) = a_0 + \alpha_0 \sin(x)$,
$a_1(x) = a_1 + \alpha_1 x(1 - e^{x-\pi}) + \alpha_2 x(\pi - x)$.
Central difference and uniform stepsize $\Rightarrow$

$$\dot{v}(t) = \frac{(n+1)^2}{\pi^2} \begin{pmatrix} -2 & 1 & \\ 1 & \ddots & 1 \\ & 1 & -2 \end{pmatrix} v(t) +$$

$$\begin{pmatrix} a_0(x_1) & 0 & \\ 0 & \ddots & 0 \\ & 0 & a_0(x_n) \end{pmatrix} v(t) + \begin{pmatrix} a_1(x_1) & 0 & \\ 0 & \ddots & 0 \\ & 0 & a_1(x_n) \end{pmatrix} v(t-\tau)$$

Example 1: CPU consumption

Method	$n = 100$		$n = 10^3$		$n = 10^4$		$n = 10^6$	
	nof. eigs	CPU	nof. eigs	CPU	nof. eigs	CPU	nof. eigs	CPU
DDEBIFTOOL 2.03 minrealpart=-20	5	18.0s	**MEMERR**					
SOD(MS,$N = 4$)	6	0.2s	6	3.3s	**MEMERR**			
SOD(MS,$N = 8$)	10	0.4s	9	6.1s	**MEMERR**			
IGD(PS,$N = 5$)	5	0.3s	5	2.6s	**MEMERR**			
IGD(PS,$N = 10$)	12	0.4s	12	27.4s	**MEMERR**			
IGD(Euler,$N = 10$)	4	0.1s	4	1.2s	**MEMERR**			
IGD(Euler,$N = 100$)	5	0.9s	5	12.9s	**MEMERR**			
SRI(RI) restart=0	12	1.3s	12	0.5s	12	2.8s	12	361.7s
SRI(RI) restart=20	12	0.3s	12	0.3s	12	1.9s	12	306.5s
SRI(RI) restart=∞	12	0.2s	12	0.3s	12	1.9s	12	**292.5s**

Example 1: CPU consumption

Method	$n = 100$		$n = 10^3$		$n = 10^4$		$n = 10^6$	
	nof. eigs	CPU	nof. eigs	CPU	nof. eigs	CPU	nof. eigs	CPU
DDEBIFTOOL 2.03 minrealpart=-20	5	18.0s	**MEMERR**					
SOD(MS,$N = 4$)	6	0.2s	6	3.3s	**MEMERR**			
SOD(MS,$N = 8$)	10	0.4s	9	6.1s	**MEMERR**			
IGD(PS,$N = 5$)	5	0.3s	5	2.6s	**MEMERR**			
IGD(PS,$N = 10$)	12	0.4s	12	27.4s	**MEMERR**			
IGD(Euler,$N = 10$)	4	0.1s	4	1.2s	**MEMERR**			
IGD(Euler,$N = 100$)	5	0.9s	5	12.9s	**MEMERR**			
SRI(RI) restart=0	12	1.3s	12	0.5s	12	2.8s	12	361.7s
SRI(RI) restart=20	12	0.3s	12	0.3s	12	1.9s	12	306.5s
SRI(RI) restart=∞	12	0.2s	12	0.3s	12	1.9s	12	**292.5s**

In the literature: Large scale DDE: $n = 131$ [Verheyden, Green, Roose, '04].

Example 2

$$\dot{x}(t) = A_0 x(t) + A_1 x(t-1),\ t > 0$$
$$x(t) = \varphi(t),\ t \in [-1, 0]$$

Random matrices:

```
rand('seed',0);
A0=alpha*sprandn(n,n,beta);
A1=alpha*sprandn(n,n,beta);
```

Example 2

$$\dot{x}(t) = A_0 x(t) + A_1 x(t-1),\ t > 0$$
$$x(t) = \varphi(t),\ t \in [-1, 0]$$

Random matrices:

```
rand('seed',0);
A0=alpha*sprandn(n,n,beta);
A1=alpha*sprandn(n,n,beta);
```

Method	$n=100$, $\alpha=0.7$, $\beta=0.1$		$n=1000$, $\alpha=6$, $\beta=0.1$		$n=2000$, $\alpha=9$, $\beta=0.1$		$n=4000$, $\alpha=10$, $\beta=0.05$		$n=10000$, $\alpha=15$, $\beta=0.001$	
	Accuracy	CPU	Accuracy	CPU	Accuracy	CPU	Accuracy	CPU	Accuracy	CPU
SOD(MS,$N=2$)	4.8e-05	0.3s	8.4e-06	19.3s	8.5e-06	109.7s	**MEMERR**			
SOD(MS,$N=4$)	6.1e-07	0.3s	1.0e-07	38.5s	1.0e-07	193.2s	**MEMERR**			
IGD(PS,$N=3$)	6.3e-03	0.3s	5.5e-03	9.8s	7.7e-03	47.8s	4.9e-03	321.5s	**MEMERR**	
IGD(PS,$N=5$)	2.8e-06	**0.1s**	2.3e-06	11.7s	3.7e-06	105.4s	**MEMERR**			
IGD(Euler,$N=10$)	1.0e-02	0.6s	1.1e-02	131.1s	1.2e-02	399.0s	**MEMERR**			
IGD(Euler,$N=100$)	1.0e-03	5.4s	1.1e-03	680.8s	**MEMERR**					
SRI(IGD) restart=0	3.8e-04	14.0s	2.6e-05	55.3s	1.0e-04	69.3s	1.1e-04	225.0s	1.1e-03	523.0s
SRI(IGD) restart=50	3.8e-04	29.9s	4.5e-05	91.2s	1.1e-04	75.3s	4.6e-05	245.5s	2.0e-04	521.4s
SRI(IGD) restart=∞	3.8e-04	38.1s	4.5e-05	113.5s	1.1e-04	91.0s	2.3e-01	286.3s	2.0e-04	**518.7s**

Thanks for your attention!