AEROSPACE ENGINEERING
TEXAS A&M UNIVERSITY

Lecture 3
PICARD-CHEBYSHEV METHODS

JOHN L. JUNKINS & ROBYN M. WOOLLANDS

Five Part Lecture Series

Picard-Chebyshev Numerical Integration Applications in Astrodynamics

Texas A&M University
Department of Aerospace Engineering
College Station, TX 77840

Spring 2017

FIVE PART LECTURE SERIES

Lecture	Title	Presenter
1	**Orthogonal Approximation**	Junkins
2	**Numerical Quadrature**	Junkins
3	**Picard-Chebyshev Methods & Theoretical Convergence**	Woollands
4	**Accelerated Picard Iteration & Adaptive Segmentation**	Woollands
5	**Gravity Approximations**	Junkins

CONTENTS

Picard iteration

- Picard iteration is a ***successive path approximation*** technique for solving differential equations.

Least Squares

- Review of least squares from ***lecture 1*** (vector problem)
- Discuss the least squares operator

Picard-Chebyshev Initial Value Problem Derivation/Algorithm (First Order)

- Thoroughly derive the Picard-Chebyshev first order IVP algorithm
- Discuss the first integration operator (P_1)
- Present two examples to demonstrate the method (MATLAB code is available)

Picard-Chebyshev Initial Value Problem Derivation/Algorithm (Second Order)

- Derive the Picard-Chebyshev second order IVP algorithm
- Discuss the second integration operator (P_2)
- Present two examples to demonstrate the method (MATLAB code is available)

Picard-Chebyshev Boundary Value Problem Derivation/Algorithm

- Three types of BVPs
- Derive the Picard-Chebyshev second order BVP algorithm
- Present three examples to demonstrate the three methods (MATLAB code is available)

Convergence Picard-Chebyshev Algorithm

- Discuss convergence for the IVP and TPBVP algorithms (MATLAB code is available)

PICARD ITERATION

What is Picard iteration?

• Picard iteration is a ***successive path approximation*** technique for solving differential equations of the form:

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(t) \in R^{1\times n}.$$

• This can be rearranged without approximation to the following ***integral equation***:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(s, \boldsymbol{x}(s))ds.$$

• A ***series*** of trajectory approximations (Picard iteration) can be generated by:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(s, \boldsymbol{x}^{i-1}(s))ds, \quad i = 1, 2, \ldots$$

Picture Credit: Wikipedia

Charles Emile Picard (1856-1941)

Picard Convergence Theorem

• If there is a time interval $|t - t_0| < \delta$ and a starting trajectory $\boldsymbol{x}^0(t)$ satisfying $\|\boldsymbol{x}(t) - \boldsymbol{x}^0(t)\| < \Delta$, for suitable finite bounds (δ, Δ), then the Picard sequence converges.

PICARD–CHEBYSHEV METHODS

Picard-Chebyshev methods combine the techniques of two great mathematicians…

Picture Credit: Wikipedia

Charles Emile Picard
(1856-1941)

Developed the ***path approximation*** method for solving differential equations

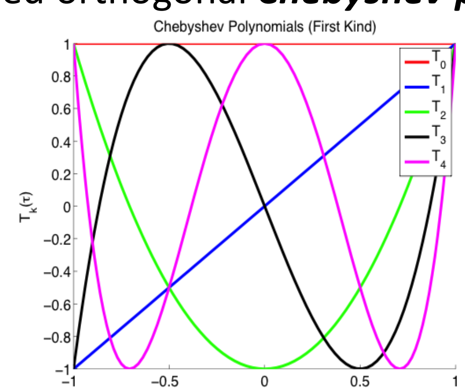
$$\boldsymbol{x}^{i}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \underbrace{\boldsymbol{f}(s, \boldsymbol{x}^{i-1}(s))}\, ds, \quad i = 1, 2, \ldots$$

Chebyshev polynomials are used to ***approximate the integrate*** in the Picard iteration sequence.

Picture Credit: Wikipedia

Pafnuty Chebyshev
(1821-1894)

Developed orthogonal ***Chebyshev polynomials***



Recall Lecture 1

$$T_k(\tau) = \cos(k \arccos(\tau)).$$

FIRST ORDER METHOD

Recall Picard Iteration

$$\underbrace{\boldsymbol{x}^{i}(t)=\boldsymbol{x}(t_0)+\int_{t_0}^{t}\boldsymbol{f}(s,\boldsymbol{x}^{i-1}(s))ds,\quad i=1,2,\ldots}_{} \qquad \boldsymbol{x}^{i}(t)=\boldsymbol{x}(-1)+\int_{-1}^{\tau}\sum_{k=0}^{N-1}\boldsymbol{a}_k^{i-1}T_k(s)ds,\quad i=1,2,\ldots$$

(Lecture 1)

Approximation with Chebyshev polynomials

N = order of Chebyshev series
n = # of state variables
$M+1$ = # of sample (node) points

Recall Least Squares Approximation

- Scalar system:

$$\frac{dx(t)}{dt}=f(t,x(t)),\quad x(t_0)=x_0,\quad x(t)\in R^{1\times 1}. \xrightarrow{\text{Least squares coefficients}} \boldsymbol{a}^{i-1}=\left(\Phi^{\mathrm{T}}W\Phi\right)^{-1}\Phi^{\mathrm{T}}W\boldsymbol{f}^{\,i-1}=V\Phi^{\mathrm{T}}W\boldsymbol{f}^{\,i-1}=A\boldsymbol{f}^{\,i-1},$$

where
$$\boldsymbol{a}^{i-1}=\begin{Bmatrix}a_0^{i-1}\\ a_1^{i-1}\\ \vdots\\ a_{N-1}^{i-1}\end{Bmatrix},\quad \boldsymbol{f}^{\,i-1}=\begin{Bmatrix}f(\tau_0,x^{i-1}(\tau_0))\\ f(\tau_1,x^{i-1}(\tau_1))\\ \vdots\\ f(\tau_M,x^{i-1}(\tau_M))\end{Bmatrix},\quad [\Phi]=[T]=\begin{bmatrix}T_0(\tau_0)&\cdots&T_{N-1}(\tau_0)\\ T_0(\tau_1)&\cdots&T_{N-1}(\tau_1)\\ \vdots&\ddots&\vdots\\ T_0(\tau_M)&\cdots&T_{N-1}(\tau_M)\end{bmatrix}$$
and
$$\tau(t)=-1+2\frac{(t-t_0)}{(t_f-t_0)},$$
$$t=\frac{(t_f+t_0)}{2}+\frac{(t_f-t_0)}{2}\tau,$$
$$=w_1+w_2\tau.$$

- n-dimensional system (use row vectors)

$$\frac{d\boldsymbol{x}(t)}{dt}=\boldsymbol{f}(t,\boldsymbol{x}(t)),\quad \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\quad \boldsymbol{x}(t)\in R^{1\times n}. \xrightarrow{\text{Least squares coefficients}} G^{i-1}=\left(\Phi^{\mathrm{T}}W\Phi\right)^{-1}\Phi^{\mathrm{T}}WF^{i-1}=V\Phi^{\mathrm{T}}WF^{i-1}=AF^{i-1},$$

$$[G^{i-1}]=\begin{Bmatrix}\boldsymbol{a}_0^{i-1}\\ \boldsymbol{a}_1^{i-1}\\ \vdots\\ \boldsymbol{a}_{N-1}^{i-1}\end{Bmatrix}=\begin{bmatrix}a_{01}^{i-1}&\cdots&a_{0n}^{i-1}\\ a_{11}^{i-1}&\cdots&a_{1n}^{i-1}\\ \vdots&\ddots&\vdots\\ a_{(N-1)1}^{i-1}&\cdots&a_{(N-1)n}^{i-1}\end{bmatrix},\quad [F^{i-1}]=\begin{Bmatrix}\boldsymbol{f}(\tau_0,\boldsymbol{x}^{i-1}(\tau_0))\\ \boldsymbol{f}(\tau_1,\boldsymbol{x}^{i-1}(\tau_1))\\ \vdots\\ \boldsymbol{f}(\tau_M,\boldsymbol{x}^{i-1}(\tau_M))\end{Bmatrix}=\begin{bmatrix}f_1(\tau_0,\boldsymbol{x}^{i-1}(\tau_0))&\cdots&f_n(\tau_0,\boldsymbol{x}^{i-1}(\tau_0))\\ f_1(\tau_1,\boldsymbol{x}^{i-1}(\tau_1))&\cdots&f_n(\tau_1,\boldsymbol{x}^{i-1}(\tau_1))\\ \vdots&\ddots&\vdots\\ f_1(\tau_M,\boldsymbol{x}^{i-1}(\tau_M))&\cdots&f_n(\tau_M,\boldsymbol{x}^{i-1}(\tau_M))\end{bmatrix}.$$

FIRST ORDER METHOD

From the previous slide…

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{a}_k^{i-1} T_k(s)ds, \;\; i = 1, 2, \ldots$$

Expand

- Expanding the Chebyshev series above leads to:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + \boldsymbol{a}_0^{i-1}\int_{-1}^{\tau} T_0(s)ds + \boldsymbol{a}_1^{i-1}\int_{-1}^{\tau} T_1(s)ds + \ldots + \boldsymbol{a}_{N-2}^{i-1}\int_{-1}^{\tau} T_{N-2}(s)ds + \boldsymbol{a}_{N-1}^{i-1}\int_{-1}^{\tau} T_{N-1}(s)ds.$$

Integrate

- The following relationship exists for ***Chebyshev polynomials*** and their ***integrals***:

$$\int T_0(s)ds = T_1(s), \quad \int T_1(s)ds = \frac{1}{4}\big(T_2(s) + T_0(s)\big), \quad \int T_k(s)ds = \frac{1}{2}\left(\frac{T_{k+1}(s)}{k+1} + \frac{T_{k-1}(s)}{k-1}\right), k \geq 2.$$

- Substituting the above leads to

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + \left[\boldsymbol{a}_0^{i-1} T_1(s) + \frac{1}{4}\boldsymbol{a}_1^{i-1}\big(T_2(s) + T_0(s)\big) + \frac{1}{2}\boldsymbol{a}_2^{i-1}\left(\frac{T_3(s)}{3} - T_1(s)\right) + \ldots\right.$$

$$\left.\cdots + \frac{1}{2}\boldsymbol{a}_{N-2}^{i-1}\left(\frac{T_{N-1}(s)}{N-1} + \frac{T_{N-3}(s)}{N-3}\right) + \frac{1}{2}\boldsymbol{a}_{N-1}^{i-1}\left(\frac{T_N(s)}{N} + \frac{T_{N-2}(s)}{N-2}\right)\right]\Bigg|_{-1}^{\tau}$$

FIRST ORDER METHOD

From the previous slide…

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + \left[\boldsymbol{a}_0^{i-1} T_1(s) + \frac{1}{4}\boldsymbol{a}_1^{i-1}\left(T_2(s)+T_0(s)\right) + \frac{1}{2}\boldsymbol{a}_2^{i-1}\left(\frac{T_3(s)}{3} - T_1(s)\right) + \ldots + \frac{1}{2}\boldsymbol{a}_{N-2}^{i-1}\left(\frac{T_{N-1}(s)}{N-1} + \frac{T_{N-3}(s)}{N-3}\right) + \frac{1}{2}\boldsymbol{a}_{N-1}^{i-1}\left(\frac{T_N(s)}{N} + \frac{T_{N-2}(s)}{N-2}\right)\right]\Bigg|_{-1}^{\tau}$$

Matrix Form

N = order of Chebyshev series
n = # of state variables
$M+1$ = # of sample (node) points

Solution

$$\underbrace{\boldsymbol{x}^i(t)}_{1\times n} = \underbrace{\boldsymbol{x}(-1)}_{1\times n} + \underbrace{\left[T_0(s) \quad \cdots \quad T_N(s)\right]}_{1\times(N+1)}\Big|_{-1}^{\tau} \cdot \underbrace{\begin{bmatrix} \frac{1}{4}\boldsymbol{a}_0^{i-1} \\ \frac{1}{2}\left(2\boldsymbol{a}_0^{i-1} - \boldsymbol{a}_2^{i-1}\right) \\ \frac{1}{4}\left(\boldsymbol{a}_1^{i-1} - \boldsymbol{a}_3^{i-1}\right) \\ \vdots \\ \frac{1}{2k}\left(\boldsymbol{a}_{k-1}^{i-1} - \boldsymbol{a}_{k+1}^{i-1}\right) \\ \vdots \\ \frac{1}{2(N-1)}\boldsymbol{a}_{N-2}^{i-1} \\ \frac{1}{2N}\boldsymbol{a}_{N-1}^{i-1} \end{bmatrix}}_{(N+1)\times n}$$

Initial condition vector

Let us look more closely at this matrix of the integrand fit least squares coefficients…

FIRST ORDER METHOD

Matrix of Least Squares Coefficients

N = order of Chebyshev series
n = # of state variables
$M+1$ = # of sample (node) points

$$
\underset{(N+1)\times n}{\begin{bmatrix} \frac{1}{4}\boldsymbol{a}_0^{i-1} \\ \frac{1}{2}\left(2\boldsymbol{a}_0^{i-1}-\boldsymbol{a}_2^{i-1}\right) \\ \frac{1}{4}\left(\boldsymbol{a}_1^{i-1}-\boldsymbol{a}_3^{i-1}\right) \\ \vdots \\ \frac{1}{2k}\left(\boldsymbol{a}_{k-1}^{i-1}-\boldsymbol{a}_{k+1}^{i-1}\right) \\ \vdots \\ \frac{1}{2(N-1)}\boldsymbol{a}_{N-2}^{i-1} \\ \frac{1}{2N}\boldsymbol{a}_{N-1}^{i-1} \end{bmatrix}}
=
\underbrace{\underset{(N+1)\times N}{\begin{bmatrix}
\frac{1}{4} & 0 & & & \cdots & & & & 0 \\
1 & 0 & -1/2 & 0 & & & & & \\
0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & & \\
& 0 & \ddots & \ddots & \ddots & 0 & & & \\
\vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & & \vdots & \vdots \\
& & & & \ddots & \ddots & & \ddots & 0 \\
& & & & & \frac{1}{2(N-2)} & 0 & & \frac{-1}{2(N-1)} \\
& & & & & 0 & \frac{1}{2(N-1)} & & 0 \\
0 & & & & \cdots & & 0 & & \frac{1}{2N}
\end{bmatrix}}}_{[S]}
$$

$t \to \tau$ scale factor

$$
[W_2] = \begin{bmatrix} w_2 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_2 \end{bmatrix} = \begin{bmatrix} \frac{t_f-t_0}{2} & 0 & 0 \\ 0 & \frac{t_f-t_0}{2} & 0 \\ 0 & 0 & \frac{t_f-t_0}{2} \end{bmatrix}
$$

$$
[G^{i-1}][W_2] = [S][A][F^{i-1}][W_2]
$$

$N \times (M+1)$

LEAST SQUARES OPERATOR

$$
\underset{N\times n}{[G^{i-1}]} = \begin{Bmatrix} \boldsymbol{a}_0^{i-1} \\ \boldsymbol{a}_1^{i-1} \\ \vdots \\ \boldsymbol{a}_{N-1}^{i-1} \end{Bmatrix} = \underbrace{\left(\Phi^{\mathrm{T}} W \Phi\right)^{-1} \Phi^{\mathrm{T}} W}_{\equiv A} [F^{i-1}] = A[F^{i-1}], \quad \underset{(M+1)\times n}{[F^{i-1}]} = \begin{bmatrix} f_1(\tau_0, \boldsymbol{x}^{i-1}(\tau_0)) & \cdots & f_n(\tau_0, \boldsymbol{x}^{i-1}(\tau_0)) \\ f_1(\tau_1, \boldsymbol{x}^{i-1}(\tau_1)) & \cdots & f_n(\tau_1, \boldsymbol{x}^{i-1}(\tau_1)) \\ \vdots & \ddots & \vdots \\ f_1(\tau_M, \boldsymbol{x}^{i-1}(\tau_M)) & \cdots & f_n(\tau_M, \boldsymbol{x}^{i-1}(\tau_M)) \end{bmatrix}.
$$

FIRST ORDER METHOD

Matrix Form

N = *order of Chebyshev series*
n = *# of state variables*
$M+1$ = *# of sample (node) points*

$$\overset{1\times n}{\boldsymbol{x}^i(t)} = \overset{1\times n}{\boldsymbol{x}(-1)} + \overset{1\times (N+1)}{\left[T_0(s) \quad \cdots \quad T_N(s)\right]\Big|_{-1}^{\tau}} \overset{(N+1)\times n}{\begin{bmatrix} \frac{1}{4}\boldsymbol{a}_0^{i-1} \\ \frac{1}{2}\left(2\boldsymbol{a}_0^{i-1} - \boldsymbol{a}_2^{i-1}\right) \\ \frac{1}{4}\left(\boldsymbol{a}_1^{i-1} - \boldsymbol{a}_3^{i-1}\right) \\ \vdots \\ \frac{1}{2k}\left(\boldsymbol{a}_{k-1}^{i-1} - \boldsymbol{a}_{k+1}^{i-1}\right) \\ \vdots \\ \frac{1}{2(N-1)}\boldsymbol{a}_{N-2}^{i-1} \\ \frac{1}{2N}\boldsymbol{a}_{N-1}^{i-1} \end{bmatrix}}$$

LEAST SQUARES OPERATOR

$$\overset{1\times n}{\boldsymbol{x}^i(t)} = \overset{1\times n}{\boldsymbol{x}(-1)} + \overbrace{\left[\left[T_0(\tau) \quad \cdots \quad T_N(\tau)\right] - \left[T_0(-1) \quad \cdots \quad T_N(-1)\right]\right]}^{1\times (N+1)} \overset{(N+1)\times N}{[S]} \underset{N\times (M+1)}{[A]} \overset{(M+1)\times n}{[F^{i-1}][W_2]}$$

FIRST ORDER METHOD

Chebyshev Series for L.H.S

$$\boldsymbol{x}^i(t) = \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{a}_k^{i-1} T_k(s)ds$$

Matrix Form

$$\overset{1\times n}{\boldsymbol{x}^i(t)} = \overset{1\times(N+1)}{\begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau)\end{bmatrix}} \overset{(N+1)\times n}{\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix}} = \overset{1\times n}{\boldsymbol{x}(-1)} + \overbrace{\left[\begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau)\end{bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1)\end{bmatrix}\right]}^{1\times(N+1)} \overset{(N+1)\times N}{[S]} \overset{N\times(M+1)}{[A]} \overset{(M+1)\times n}{[F^{i-1}][W_2]}.$$

Solution coefficients

Constant terms

$$\boldsymbol{C}(-1) = \boldsymbol{x}(-1) - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1)\end{bmatrix}[S][A][F^{i-1}][W_2]$$

$$\boldsymbol{x}^i(t) = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau)\end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \boldsymbol{C}(-1) + \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau)\end{bmatrix}[S][A][F^{i-1}][W_2].$$

FIRST ORDER METHOD

Absorb Constant

$$\boldsymbol{x}^i(t) = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_0^i - \boldsymbol{C}(-1) \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} [S][A][F^{i-1}][W_2].$$

Equate coefficients of T_k

$$\begin{bmatrix} \boldsymbol{\beta}_0^i - \boldsymbol{C}(-1) \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = [S][A][F^{i-1}][W_2] \xrightarrow{W_2} \begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \begin{Bmatrix} \boldsymbol{C}(-1) \\ 0 \\ \vdots \\ 0 \end{Bmatrix} + [S][A][F^{i-1}][W_2].$$

Recall Constant $\boldsymbol{C}(-1) = \boldsymbol{x}(-1) - [T_0(-1) \quad \cdots \quad T_N(-1)][S][A]\left[F^{i-1}\right][W_2]$

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1(-1) & \cdots & \boldsymbol{x}_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} [S][A][F^{i-1}][W_2] + [S][A][F^{i-1}][W_2].$$

FIRST ORDER METHOD

From the previous slide…

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \begin{Bmatrix} \boldsymbol{x}(-1) \\ 0 \\ \vdots \\ 0 \end{Bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} [S][A][F^{i-1}][W_2] + [S][A][F^{i-1}][W_2].$$

Final Expression for Coefficients

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \boldsymbol{X}_0 + \big[[I]-[L]\big][S][A][F^{i-1}][W_2] = \boldsymbol{X}_0 + [P_1][A][F^{i-1}][W_2],$$

$$\text{where } \boldsymbol{X}_0 = \begin{bmatrix} \boldsymbol{x}_1(-1) & \cdots & \boldsymbol{x}_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \ [L] = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

FIRST ORDER METHOD

Recall the summation equation

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{a}_k^{i-1} T_k(s)\,ds, \quad i = 1, 2, \ldots$$

Expression for Coefficients

$$\underset{(N+1)\times n}{\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix}} = \overset{N+1\times n}{\boldsymbol{X}_0} + \overbrace{[\underbrace{[I]-[L]}_{(N+1)\times(N+1)}]\underset{(N+1)\times N}{[S]}}^{[P_1]\ (N+1)\times N}\overset{N\times(M+1)}{[A]}\overset{(M+1)\times n}{[F^{i-1}]}[W_2] = \boldsymbol{X}_0 + \underset{\text{INTEGRATION OPERATOR}}{[P_1]}\overset{\text{LEAST SQUARES OPERATOR}}{[A]}[F^{i-1}][W_2], \text{ where}$$

$$\overset{N+1\times n}{\boldsymbol{X}_0} = \begin{bmatrix} \boldsymbol{x}_1(-1) & \cdots & \boldsymbol{x}_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},$$

$$\overset{(N+1)\times(N+1)}{[L]} = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

Matrix Representation

$$\boldsymbol{\beta}^i = \boldsymbol{X}_0 + [P_1][A][F^{i-1}][W_2], \qquad \boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\beta}^i.$$

FIRST ORDER ALGORITHM

Dynamics & Initial Conditions

$$\frac{d\boldsymbol{x}}{dt}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0,\ t_0 = 0,\ t_f = T.$$

Time and τ

$$\tau_j = -\cos\left(\frac{j\pi}{M}\right),\ \ j = 0, 1, 2, \ldots, M.$$

$$t = \frac{(t_f + t_0)}{2} + \frac{(t_f - t_0)}{2}\tau = w_1 + w_2\tau.$$

Picard Iteration

$$\frac{d\boldsymbol{x}(t)}{d\tau} = w_2 \boldsymbol{f}(t, \boldsymbol{x}(t))$$

$$\boldsymbol{\beta}^i = \boldsymbol{X}_0 + [P_1][A][F^{i-1}][W_2]$$

$$\boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\beta}^i$$

Iterate

Convergence

$$e = \max\left(\frac{\left|\boldsymbol{x}^i(t) - \boldsymbol{x}^{i-1}(t)\right|}{\left|\boldsymbol{x}^i(t)\right|}\right)$$

Update

$$\boldsymbol{x}^{i+1}(t) = \boldsymbol{x}^i(t)$$

$$\boldsymbol{X}_0 = \begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},$$

$$[L] = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

$$[P_1] = [[I] - [L]][S],$$

$$T(\tau) = \begin{bmatrix} T_0(\tau_0) & \cdots & T_N(\tau_0) \\ T_0(\tau_1) & \cdots & T_N(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_N(\tau_M) \end{bmatrix}.$$

$M = N$

$$A = \begin{bmatrix} \frac{1}{2}\frac{1}{M}T_0(\tau_0) & \frac{1}{M}T_0(\tau_1) & \cdots & \frac{1}{M}T_0(\tau_{M-1}) & \frac{1}{2}\frac{1}{M}T_0(\tau_M) \\ \frac{1}{2}\frac{2}{M}T_1(\tau_0) & \frac{2}{M}T_1(\tau_1) & \cdots & \frac{2}{M}T_1(\tau_{M-1}) & \frac{1}{2}\frac{2}{M}T_1(\tau_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2}\frac{2}{M}T_{N-2}(\tau_0) & \frac{2}{M}T_{N-2}(\tau_1) & \cdots & \frac{2}{M}T_{N-2}(\tau_{M-1}) & \frac{1}{2}\frac{2}{M}T_{N-2}(\tau_M) \\ \frac{1}{2}\frac{1}{M}T_{N-1}(\tau_0) & \frac{1}{M}T_{N-1}(\tau_1) & \cdots & \frac{1}{M}T_{N-1}(\tau_{M-1}) & \frac{1}{2}\frac{1}{M}T_{N-1}(\tau_M) \end{bmatrix}.$$

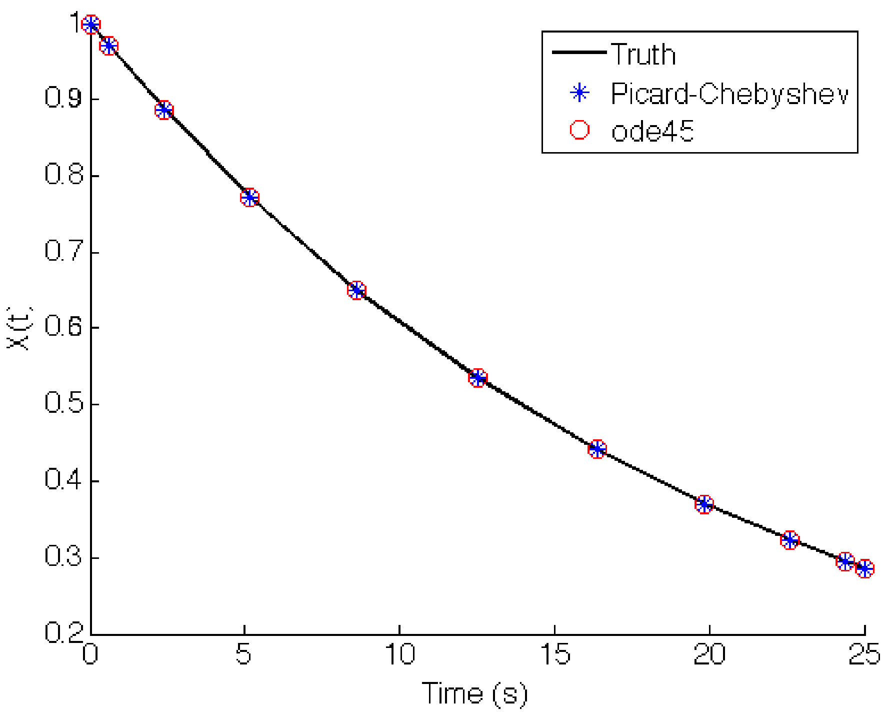
$M > N$

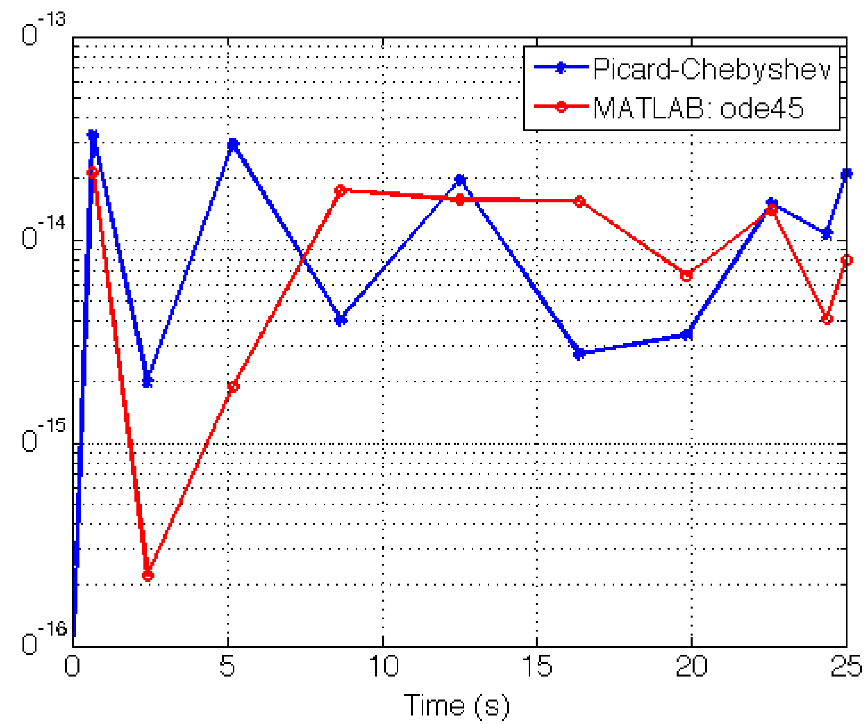
$$A = \begin{bmatrix} \frac{1}{2}\frac{1}{M}T_0(\tau_0) & \frac{1}{M}T_0(\tau_1) & \cdots & \frac{1}{M}T_0(\tau_{M-1}) & \frac{1}{2}\frac{1}{M}T_0(\tau_M) \\ \frac{1}{2}\frac{2}{M}T_1(\tau_0) & \frac{2}{M}T_1(\tau_1) & \cdots & \frac{2}{M}T_1(\tau_{M-1}) & \frac{1}{2}\frac{2}{M}T_1(\tau_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2}\frac{2}{M}T_{N-2}(\tau_0) & \frac{2}{M}T_{N-2}(\tau_1) & \cdots & \frac{2}{M}T_{N-2}(\tau_{M-1}) & \frac{1}{2}\frac{2}{M}T_{N-2}(\tau_M) \\ \frac{1}{2}\frac{2}{M}T_{N-1}(\tau_0) & \frac{2}{M}T_{N-1}(\tau_1) & \cdots & \frac{2}{M}T_{N-1}(\tau_{M-1}) & \frac{1}{2}\frac{2}{M}T_{N-1}(\tau_M) \end{bmatrix}.$$

$$[S] = \begin{bmatrix} \frac{1}{4} & 0 & & & \cdots & & & 0 \\ 1 & 0 & -1/2 & 0 & & & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & \\ & 0 & \ddots & \ddots & \ddots & 0 & & \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \frac{1}{2(N-2)} & 0 & \frac{-1}{2(N-1)} \\ & & & & & 0 & \frac{1}{2(N-1)} & 0 \\ 0 & & & & \cdots & & 0 & \frac{1}{2N} \end{bmatrix}$$

EXAMPLE 1

First Order Example: $\dot{x}(t) = -0.05x$

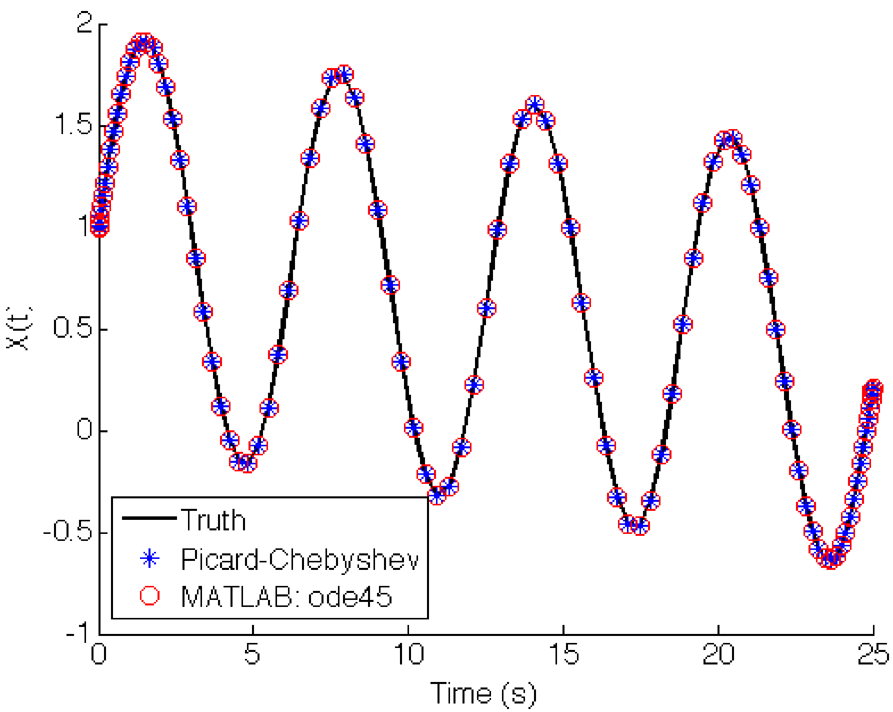


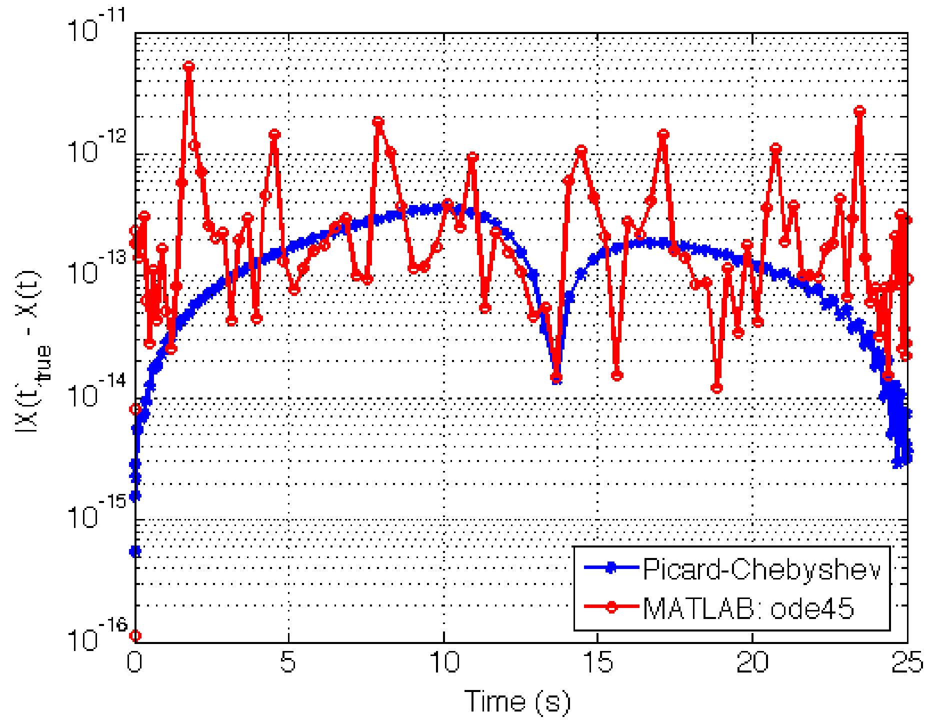


This simple first order example is solved using the Picard-Chebyshev technique and MATLAB's ode45. The is an analytic solution to the problem that we use to check the accuracy of the integrators. The code required for generating the above figures is available for use as a learning tool: **run_lecture3_example1a_ivpl.m** and **run_lecture3_example1b_fvpl.m**.

EXAMPLE 2

First Order Example: $\dot{x}(t) = \cos(t + 0.05x)$





This simple first order example is solved using the Picard-Chebyshev technique and MATLAB's ode45. The is an analytic solution to the problem (see Bai's PhD) that we use to check the accuracy of the integrators. The code required for generating the above figures is available for use as a learning tool: **run_lecture3_example2_ivpI.m**.

SECOND ORDER METHOD

Second Order Differential Equation

$$\ddot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{v}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{v}(t_0) = \boldsymbol{v}_0, \quad \boldsymbol{x}(t), \boldsymbol{v}(t) \in R^{1\times n}.$$

Velocity Approximation

$$\boldsymbol{v}^i(\tau) = \boldsymbol{v}(-1) + \int_{-1}^{s} \boldsymbol{f}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq.$$

Similar to the first order case, the velocity can be written in terms of a Chebyshev series, allowing the $\boldsymbol{\beta}$ coefficients to computed in terms of the least squares $\boldsymbol{a}$ coefficients:

$$\boldsymbol{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s) = \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq.$$

Position Approximation

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boldsymbol{v}(-1) + \int_{-1}^{s} \boldsymbol{f}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq \right\} ds.$$

The position can also be written in terms of a Chebyshev series, where the position coefficients ($\boldsymbol{\alpha}$) can be determined in terms of the least squares coefficients ($\boldsymbol{a}$),

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\},$$

however, it is more convenient to compute the position coefficients ***directly*** from the velocity coefficients by applying the ***integration operator*** twice. More on this to follow.

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s)ds.$$

SECOND ORDER METHOD: STEP 1

Velocity Integration

$$\boldsymbol{v}^i(t)=\sum_{k=0}^{N-1}\boldsymbol{\beta}_k^i T_k(\tau)=\boldsymbol{v}(-1)+\int_{-1}^{\tau}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)dq.$$

Expression for Velocity Coefficients

LEAST SQUARES OPERATOR

$$\overset{N\times n}{\begin{bmatrix}\boldsymbol{\beta}_0^i\\ \boldsymbol{\beta}_1^i\\ \vdots\\ \boldsymbol{\beta}_{N-2}^i\\ \boldsymbol{\beta}_{N-1}^i\end{bmatrix}}=\overset{N\times n}{\boldsymbol{V}_0}+\overbrace{[\underbrace{[I]-[L]}_{N\times N}]\underset{N\times(N-1)}{[S]}}^{\substack{[P_1]\\ N\times(N-1)}}\overset{(N-1)\times(M+1)}{[A]}\overset{(M+1)\times n}{[F^{i-1}]}[W_2]=\boldsymbol{V}_0+[P_1][A][F^{i-1}][W_2] \text{ where}$$

INTEGRATION OPERATOR

$$\overset{N\times n}{\boldsymbol{V}_0}=\begin{bmatrix}v_1(-1) & \cdots & v_n(-1)\\ 0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix},\quad \overset{N\times N}{[L]}=\begin{bmatrix}T_0(-1) & \cdots & T_N(-1)\\ 0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}.$$

Matrix Representation for Velocity

$$\boldsymbol{\beta}^i=\boldsymbol{V}_0+[P_1][A][F^{i-1}][W_2],\quad \boldsymbol{v}^i(t)=T(\tau)\boldsymbol{\beta}^i.$$

SECOND ORDER METHOD: STEP 2

Position Integration

$$\mathbf{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \mathbf{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s)ds.$$

Expression for Position Coefficients

INTEGRATION OPERATOR

$$\underbrace{\begin{bmatrix} \boldsymbol{\alpha}_0^i \\ \boldsymbol{\alpha}_1^i \\ \vdots \\ \boldsymbol{\alpha}_{N-1}^i \\ \boldsymbol{\alpha}_N^i \end{bmatrix}}_{(N+1)\times n} = \underbrace{\boldsymbol{X}_0}_{(N+1)\times n} + \overbrace{\underbrace{[[I]-[L]]}_{(N+1)\times(N+1)}\underbrace{[S]}_{(N+1)\times N}}^{[P_2]\ (N+1)\times N} \underbrace{\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-2}^i \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix}}_{N\times n} [W_2] = \boldsymbol{X}_0 + [P_2]\boldsymbol{\beta}^i[W_2] \text{ where}$$

Velocity Coefficients

$$\underbrace{\boldsymbol{X}_0}_{(N+1)\times n} = \begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},$$

$$\underbrace{[L]}_{(N+1)\times(N+1)} = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

Matrix Representation for Position

$$\boldsymbol{\alpha}^i = \boldsymbol{X}_0 + [P_2]\boldsymbol{\beta}^i[W_2], \quad \mathbf{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i.$$

SECOND ORDER ALGORITHM

Dynamics & Initial Conditions

$$\frac{d^2\boldsymbol{x}}{dt^2}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)),\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0,\ t_0 = 0,\ t_f = T.$$

Time and τ

$$\tau_j = -\cos\left(\frac{j\pi}{M}\right),\ \ j = 0,1,2,\ldots,M.$$

$$t = \frac{(t_f + t_0\ )}{2} + \frac{(t_f - t_0\ )}{2}\tau = w_1 + w_2\tau.$$

Picard Iteration

$$\frac{d^2\boldsymbol{x}(t)}{d^2\tau} = w_2^2\, \boldsymbol{f}(t, \boldsymbol{x}(t))$$

$$\boldsymbol{\beta}^i = \boldsymbol{V}_0 + [P_1][A][F^{i-1}][[W_2]],\ \boldsymbol{v}^i(t) = T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{\alpha}^i = \boldsymbol{X}_0 + [P_2]\boldsymbol{\beta}^i[[W_2]],\ \boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i.$$

Iterate

Convergence

$$e = \max\left(\left[\max\left(\frac{\left|\boldsymbol{x}^i(t) - \boldsymbol{x}^{i-1}(t)\right|}{\left|\boldsymbol{x}^i(t)\right|}\right), \max\left(\frac{\left|\boldsymbol{v}^i(t) - \boldsymbol{v}^{i-1}(t)\right|}{\left|\boldsymbol{v}^i(t)\right|}\right)\right]\right)$$

Update

$$\boldsymbol{x}^{i+1}(t) = \boldsymbol{x}^i(t),\ \boldsymbol{v}^{i+1}(t) = \boldsymbol{v}^i(t).$$

Velocity Matrices

$$[P_1] = [[I] - [L]][S]$$

$$\boldsymbol{V}_0 = \begin{bmatrix} \boldsymbol{v}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad [L] = \begin{bmatrix} T(-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$T(\tau) = \begin{bmatrix} T_0(\tau_0) & \cdots & T_{N-1}(\tau_0) \\ T_0(\tau_1) & \cdots & T_{N-1}(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_{N-1}(\tau_M) \end{bmatrix}$$

$$[S] = \begin{bmatrix} \frac{1}{4} & 0 & & & \cdots & & & 0 \\ 1 & 0 & -1/2 & 0 & & & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & \\ & 0 & \ddots & \ddots & \ddots & 0 & & \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \frac{1}{2(N-3)} & 0 & \frac{-1}{2(N-2)} \\ & & & & & 0 & \frac{1}{2(N-2)} & 0 \\ 0 & & & & \cdots & & 0 & \frac{1}{2(N-1)} \end{bmatrix}$$

Position Matrices

$$[P_2] = [[I] - [L]][S]$$

$$\boldsymbol{X}_0 = \begin{bmatrix} \boldsymbol{x}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad [L] = \begin{bmatrix} T(-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

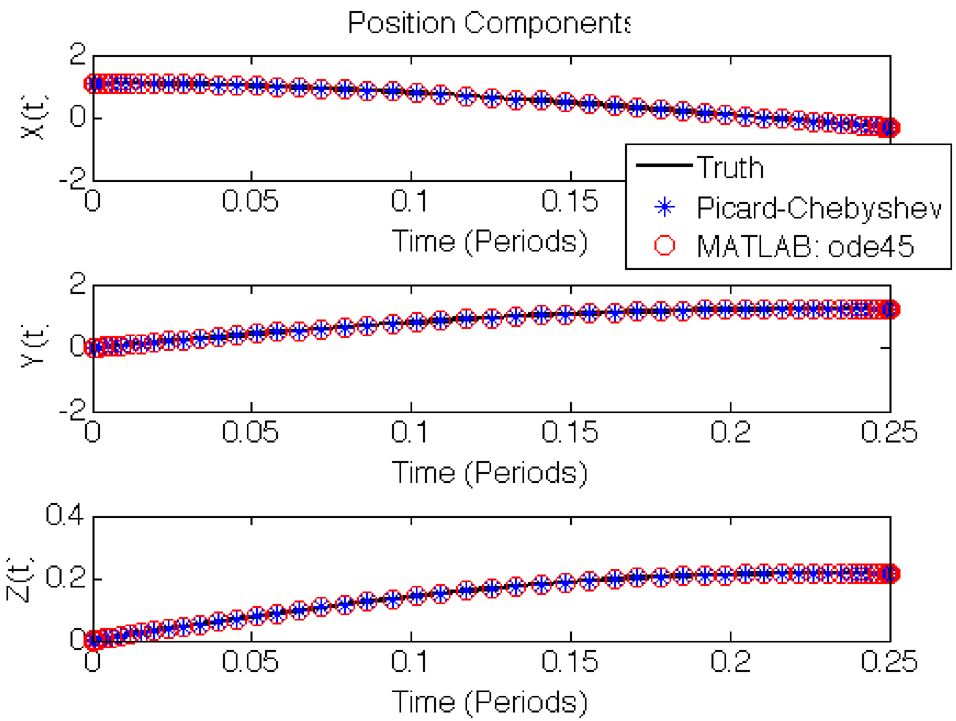
$$T(\tau) = \begin{bmatrix} T_0(\tau_0) & \cdots & T_N(\tau_0) \\ T_0(\tau_1) & \cdots & T_N(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_N(\tau_M) \end{bmatrix}$$

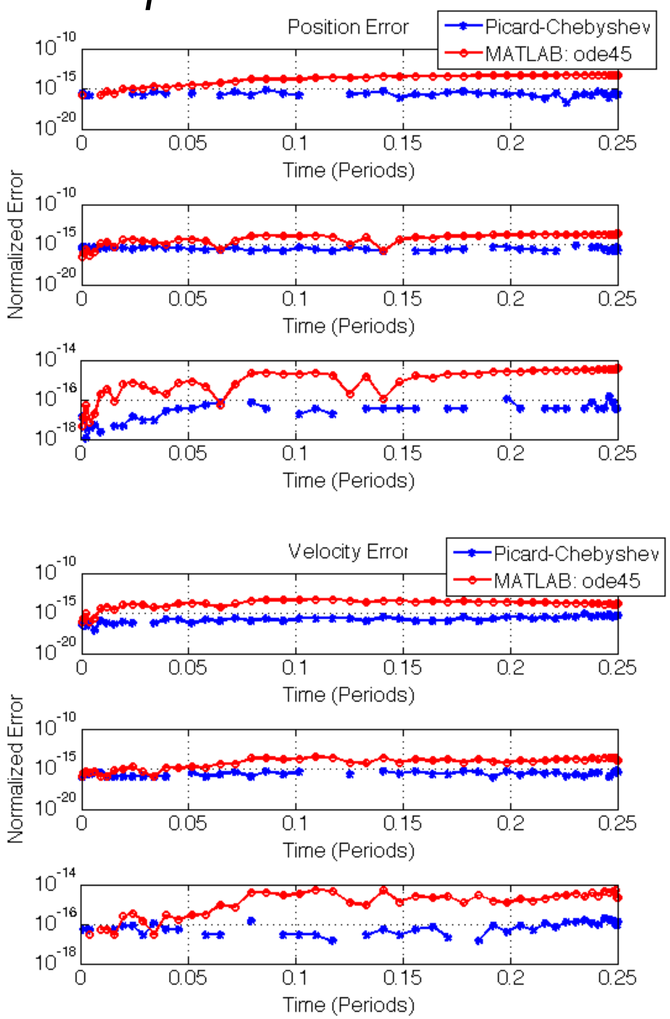
$$[S] = \begin{bmatrix} \frac{1}{4} & 0 & & & \cdots & & & 0 \\ 1 & 0 & -1/2 & 0 & & & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & \\ & 0 & \ddots & \ddots & \ddots & 0 & & \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \frac{1}{2(N-2)} & 0 & \frac{-1}{2(N-1)} \\ & & & & & 0 & \frac{1}{2(N-1)} & 0 \\ 0 & & & & \cdots & & 0 & \frac{1}{2N} \end{bmatrix}$$

EXAMPLE 3

Second Order Example: Two-body Problem

$$\ddot{\boldsymbol{r}}(t) = -\frac{\mu}{r^3}\boldsymbol{r}$$



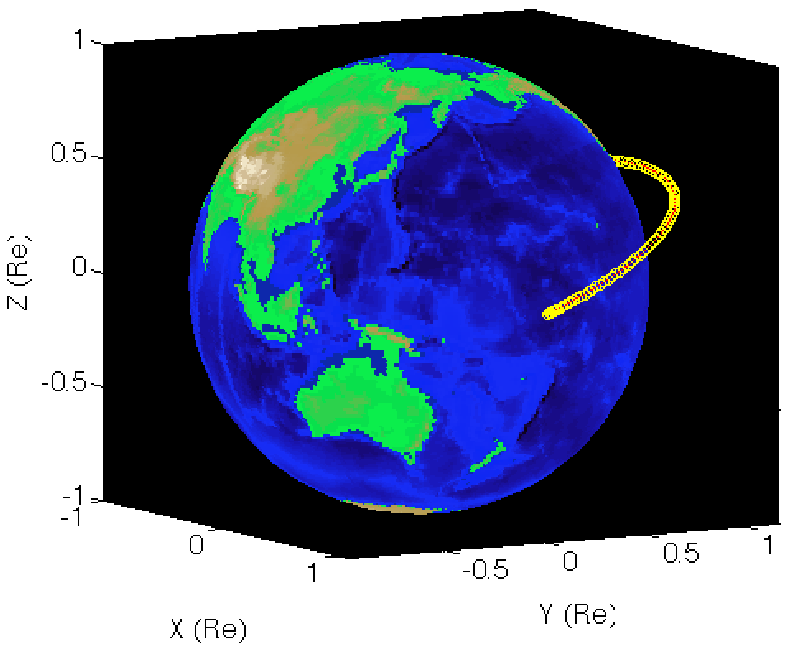


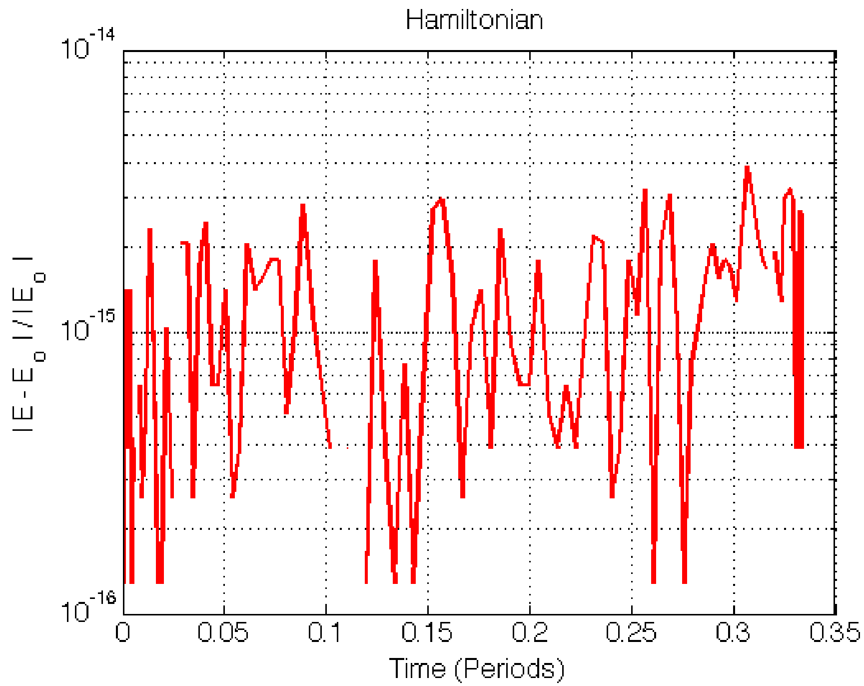
There is an analytical solution to the two-body problem in Celestial mechanics. In this example we demonstrate how the second order Picard-Chebyshev technique is used to integrate a second order system of differential equations. The code for generating the above figures is available for use as a learning tool: **run_lecture3_example3_ivpII.m**.

EXAMPLE 4

Second Order Example: Perturbed Two-body Problem

$$\ddot{\boldsymbol{r}}(t) = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_d(x,y,z)$$

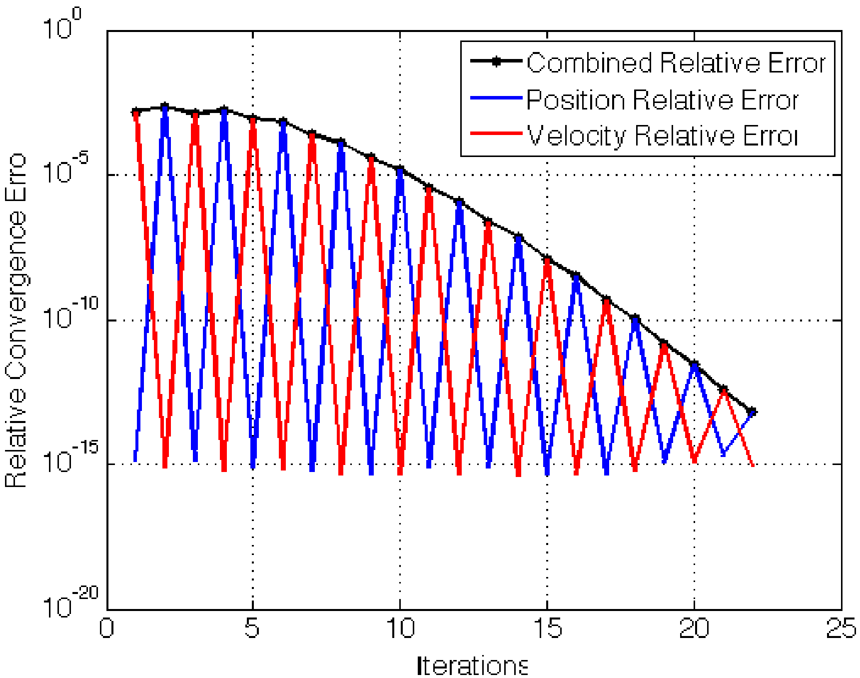




There is ***no analytical solution*** to the ***perturbed problem.*** *We used a* ***spherical harmonic degree & order 40 gravity model***. One approach to check the solution accuracy is to compute the ***Hamiltonian*** at each point and check if it is ***conserved*** to the desired tolerance over the orbit (near machine precision of 15 digits in this case). For a ***non-conservative system*** other methods such as the ***reverse test*** and ***Zadunaisky's technique*** (Berry & Healy 2003) must be utilized. The code for generating the above figures is available for use as a learning tool: **run_lecture3_example4b_ivpII.m** and **run_lecture3_example4c_fvpII.m**.

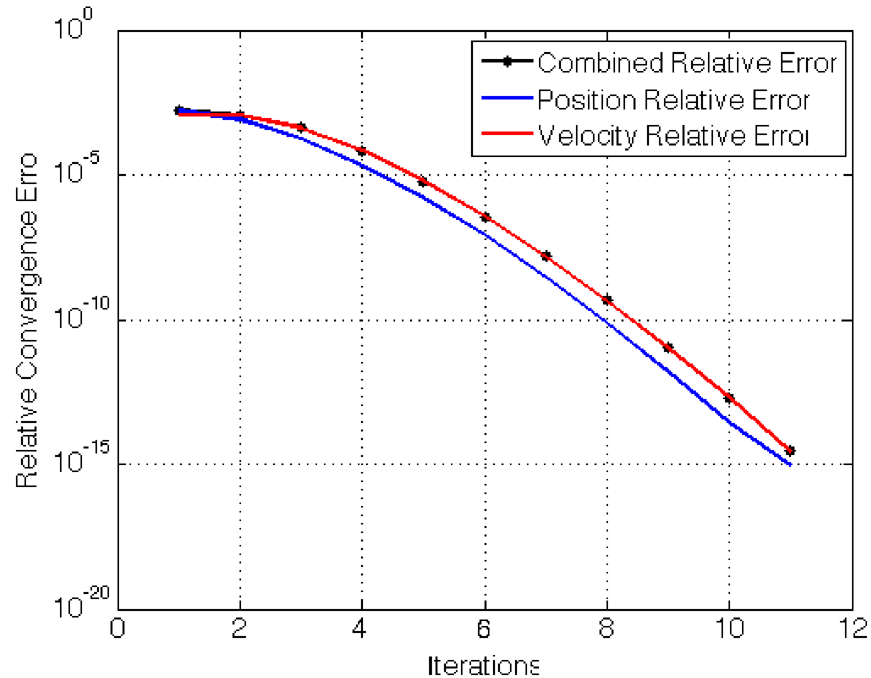
First Order vs Second Order

First Order Algorithm



$$\dot{\boldsymbol{r}}(t) = \boldsymbol{v}(t), \quad \dot{\boldsymbol{v}}(t) = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_d(x, y, z).$$

Second Order Algorithm



$$\ddot{\boldsymbol{r}}(t) = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_d(x, y, z)$$

The ***naturally*** second order system is solved in ***first order*** form using the first order Picard-Chebyshev algorithm (left). Position and velocity are updated on ***alternate iterations*** and thus the total number of iterations is about ***twice*** as many as solving the naturally second order system with the second order Picard-Chebyshev algorithm (right). **run_lecture3_example4a_ivpI.m**.

BOUNDARY VALUE PROBLEMS

Types of Boundary Value Problems

- ***BVP of the first kind***: $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ are specified.
  - Also known as a two-point boundary value problem (TPBVP).
  - In Celestial mechanics this is often referred to as ***Lambert's problem***.
  - Special Picard-Chebyshev formulation that does not require a Newton-like shooting method.

- ***BVP of the second kind***: $\boldsymbol{x}_0$ and $\boldsymbol{v}_f$ are specified.
  - Combination of initial value problem (IVP) and final value problem (FVP).
  - Special Picard-Chebyshev formulation that does not require a Newton-like shooting method.

- ***BVP of the third kind***: $\boldsymbol{x}_f$ and $\boldsymbol{v}_0$ are specified.
  - Combination of FVP and IVP.
  - Special Picard-Chebyshev formulation that does not require a Newton-like shooting method.

TWO-POINT BOUNDARY VALUE PROBLEM

Second Order Differential Equation

$$\ddot{x}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{v}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boxed{\boldsymbol{x}(t_f) = \boldsymbol{x}_f,} \quad \boldsymbol{x}(t), \boldsymbol{v}(t) \in R^{1\times n}.$$

Pseudo Velocity Approximation

The velocity can be written in terms of a Chebyshev series, and similar to the second order IVP, the $\boldsymbol{\beta}$ coefficients can be computed in terms of the least squares $\boldsymbol{a}$ coefficients. The ***initial velocity*** is unknown and the resulting ***pseudo velocity*** is correct to within the ***constant of integration***. This constant only effects the $\boldsymbol{\beta}_0$ coefficient. All other coefficients are correct.

$$\boldsymbol{v}_{pseudo}{}^{i}(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}^{i}_{k\,pseudo} T_k(s) = \cancel{\boldsymbol{v}_0} + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq.$$

Position Approximation

The position can also be written in terms of a Chebyshev series. It is clear that the unknown ***integration constant*** at the velocity level is ***contained*** within the $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ position coefficients. These must be determined using some ***other information***.

$$\begin{aligned} \boldsymbol{x}^{i}(\tau) &= \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boxed{\boldsymbol{v}_0} + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\} ds \\ &= \boldsymbol{x}(-1) + \boxed{w_2\, \boldsymbol{v}_0 s\Big|_{-1}^{\tau}} + \int_{-1}^{\tau} \left\{ \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\} ds \\ &= \boldsymbol{x}(-1) + \boxed{w_2 \boldsymbol{v}_0(\tau+1)} + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}^{i}_{k\,pseudo} T_k(s)ds. \end{aligned}$$

AEROSPACE ENGINEERING
TEXAS A&M UNIVERSITY

STEP 1: PSEUDO VELOCITY

Pseudo Velocity (correct to within a constant)

$$\boldsymbol{v}^i_{pseudo}(t)=\sum_{k=0}^{N-1}\boldsymbol{\beta}^i_{k\,pseudo}T_k(\tau)=\int_{-1}^{\tau}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)dq.$$

Expression for Pseudo Velocity Coefficients

LEAST SQUARES OPERATOR

$$\overbrace{\begin{bmatrix}\boldsymbol{\beta}^i_{0\,pseudo}\\ \boldsymbol{\beta}^i_{1\,pseudo}\\ \vdots\\ \boldsymbol{\beta}^i_{N-2\,pseudo}\\ \boldsymbol{\beta}^i_{N-1\,pseudo}\end{bmatrix}}^{N\times n}=\overbrace{[\underbrace{[I]-[L]}_{N\times N}]\underbrace{[S]}_{N\times(N-1)}}^{[P_1]\ N\times(N-1)}\overbrace{[A]}^{(N-1)\times(M+1)}\overbrace{[F^{i-1}]}^{(M+1)\times n}[[W_2]]=[[W_2]][P_1][A][F^{i-1}]\text{ where }[L]=\overbrace{\begin{bmatrix}T_0(-1)&\cdots&T_N(-1)\\0&\cdots&0\\\vdots&\ddots&\vdots\\0&\cdots&0\end{bmatrix}}^{N\times N}.$$

INTEGRATION OPERATOR

Matrix Representation for Pseudo Velocity

$$\boldsymbol{\beta}^i_{pseudo}=[P_1][A][F^{i-1}][[W_2]],\ \boldsymbol{v}^i_{pseudo}(t)=T(\tau)\boldsymbol{\beta}^i_{pseudo}.$$

STEP 2: PSEUDO POSITION

Pseudo Position (linearly contained integration constant)

$$\boldsymbol{x}^{i}_{pseudo}(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}^{i}_{k\,pseudo} T_k(\tau) = \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}^{i}_{k\,pseudo} T_k(s)ds.$$

Expression for Pseudo Position Coefficients

$$\overset{(N+1)\times n}{\begin{bmatrix} \boldsymbol{\alpha}^{i}_{0\,pseudo} \\ \boldsymbol{\alpha}^{i}_{1\,pseudo} \\ \vdots \\ \boldsymbol{\alpha}^{i}_{N-1\,pseudo} \\ \boldsymbol{\alpha}^{i}_{N\,pseudo} \end{bmatrix}} = \overbrace{\left[\underbrace{[I]-[L]}_{(N+1)\times(N+1)}\right]\underset{(N+1)\times N}{[S]}}^{\substack{(N+1)\times N \\ [P_2]}} \overset{N\times n}{\begin{bmatrix} \boldsymbol{\beta}^{i}_{0\,pseudo} \\ \boldsymbol{\beta}^{i}_{1\,pseudo} \\ \vdots \\ \boldsymbol{\beta}^{i}_{N-2\,pseudo} \\ \boldsymbol{\beta}^{i}_{N-1\,pseudo} \end{bmatrix}} [[W_2]] = \overset{\text{INTEGRATION OPERATOR}}{[P_2]} \underset{\text{Pseudo Velocity Coefficients}}{\boldsymbol{\beta}^{i}_{pseudo}} [[W_2]] \text{ where } [L] = \overset{(N+1)\times(N+1)}{\begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}.$$

Matrix Representation for Pseudo Position

$$\boldsymbol{\alpha}^{i}_{pseudo} = [P_2]\boldsymbol{\beta}^{i}_{pseudo}[[W_2]],\ \boldsymbol{x}^{i}_{pseudo}(t) = T(\tau)\boldsymbol{\alpha}^{i}_{pseudo}.$$

STEP 3: VELOCITY & POSITION

Position Trajectory

$$\boldsymbol{x}^{i}(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boldsymbol{v}_0 + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\} ds$$

Note that after integration the unknown initial velocity $(\boldsymbol{v}_0)$ is multiplied by the scalar w_2 scale factor

$$\boldsymbol{x}^{i}(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boldsymbol{v}_0 + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\} ds = \boldsymbol{x}(-1) + w_2 \boldsymbol{v}_0(\tau+1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}_{k\,pseudo}^{i} T_k(s)ds$$

Vector-matrix form

$$\boldsymbol{x}(\tau) = \boldsymbol{x}(-1) + w_2 \boldsymbol{v}_0(\tau+1) + [T(\tau)] \begin{bmatrix} \boldsymbol{\alpha}_{0\,pseudo}^{i} \\ \vdots \\ \boldsymbol{\alpha}_{N\,pseudo}^{i} \end{bmatrix} \xrightarrow{\tau = 1} \boldsymbol{x}(1) = \boldsymbol{x}(-1) + (t_f - t_0)\boldsymbol{v}_0 + T(1) \begin{bmatrix} \boldsymbol{\alpha}_{0\,pseudo}^{i} \\ \vdots \\ \boldsymbol{\alpha}_{N\,pseudo}^{i} \end{bmatrix}$$

The ***position*** at the ***final time*** $(\tau = 1)$ is ***known*** and thus the equation can be rearranged to obtain the initial velocity.

$$\boldsymbol{v}_0 = \frac{\boldsymbol{x}(1) - \boldsymbol{x}(-1)}{t_f - t_0} - \frac{1}{t_f - t_0} [T(1)] \begin{bmatrix} \boldsymbol{\alpha}_{0\,pseudo}^{i} \\ \vdots \\ \boldsymbol{\alpha}_{N\,pseudo}^{i} \end{bmatrix}$$

Velocity Coefficients

$$\boldsymbol{\beta}^{i} = \boldsymbol{V}_0 + \boldsymbol{\beta}_{pseudo}^{i}, \ \boldsymbol{v}^{i}(t) = T(\tau)\boldsymbol{\beta}^{i}.$$

Position Coefficients

$$\boldsymbol{\alpha}^{i} = \boldsymbol{X}_0 + [P_2]\boldsymbol{\beta}^{i}[[W_2]], \ \boldsymbol{x}^{i}(t) = T(\tau)\boldsymbol{\alpha}^{i}.$$

ALTERNATE METHOD FOR COEFFICIENTS

Position Solution

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \boldsymbol{v}_0 s\Big|_{-1}^{\tau} + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}^i_{k\,pseudo} T_k(s)ds.$$

The first two position coefficients, $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$, can be determined using knowledge of both the terminal boundary conditions.

Boundary Conditions

The left boundary condition can be written as:

$$\boldsymbol{x}(-1) = \sum_{k=0}^{N} \boldsymbol{\alpha}^i_k T_k(-1) = \boldsymbol{\alpha}^i_0 - \boldsymbol{\alpha}^i_1 + \boldsymbol{\alpha}^i_2 + \ldots + (-1)^N \boldsymbol{\alpha}^i_N.$$

We use this approach later for the Picard-Chebyshev convergence analysis!

The right boundary condition can be written as:

$$\boldsymbol{x}(1) = \sum_{k=0}^{N} \boldsymbol{\alpha}^i_k T_k(1) = \boldsymbol{\alpha}^i_0 + \boldsymbol{\alpha}^i_1 + \boldsymbol{\alpha}^i_2 + \ldots + \boldsymbol{\alpha}^i_N.$$

This produces ***two equations*** and ***two unknowns***, allowing $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ to be computed in terms is ***the known initial*** and ***final position***, and the other ***known coefficients***.

$$\boldsymbol{\alpha}^i_0 = \frac{\boldsymbol{x}(1) + \boldsymbol{x}(-1)}{2} - \left(\boldsymbol{\alpha}^i_2 + \boldsymbol{\alpha}^i_4 + \boldsymbol{\alpha}^i_6 \ldots\right),$$

$$\boldsymbol{\alpha}^i_1 = \frac{\boldsymbol{x}(1) - \boldsymbol{x}(-1)}{2} - \left(\boldsymbol{\alpha}^i_3 + \boldsymbol{\alpha}^i_5 + \boldsymbol{\alpha}^i_7 \ldots\right).$$

ALTERNATE METHOD FOR COEFFICIENTS

Position Coefficients

We use this approach later for the Picard-Chebyshev convergence analysis!

$$\underset{(N+1)\times n}{\begin{bmatrix} \alpha_0^i \\ \alpha_1^i \\ \vdots \\ \alpha_{N-1}^i \\ \alpha_N^i \end{bmatrix}} = \underset{(N+1)\times n}{X_{0f}} + \overbrace{\begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & \cdots \\ 0 & 0 & 0 & -1 & 0 & -1 & \cdots \\ 0 & 0 & 1 & 0 & \cdots & & 0 \\ & & \cdots & \ddots & \cdots & & 0 \\ \vdots & \vdots & \cdots & 0 & 1 & 0 & 0 \\ & & \cdots & & 0 & 1 & 0 \\ 0 & & \cdots & & & 0 & 1 \end{bmatrix}}^{\substack{(N+1)\times N \\ [P_B]}} \underset{N\times n}{\begin{bmatrix} \alpha_{pseudo0}^i \\ \alpha_{pseudo1}^i \\ \vdots \\ \alpha_{pseudo\,N-1}^i \\ \alpha_{pseudo\,N}^i \end{bmatrix}} = X_{0f} + [P_B]\alpha_{pseudo}^i, \quad X_{0f} = \underset{(N+1)\times n}{\begin{bmatrix} \dfrac{x_1(1)+x_1(-1)}{2} & \cdots & \dfrac{x_n(1)+x_n(-1)}{2} \\ \dfrac{x_1(1)-x_1(-1)}{2} & \cdots & \dfrac{x_n(1)-x_n(-1)}{2} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}.$$

Pseudo Position Coefficients

Matrix Representation for Position

$$\alpha^i = X_{0f} + [P_B]\alpha_{pseudo}^i, \qquad x^i(t) = T(\tau)\alpha^i.$$

TPBVP ALGORITHM

Dynamics & Initial Conditions

$$\frac{d^2\boldsymbol{x}}{dt^2}(t)=\boldsymbol{f}(t,\boldsymbol{x}(t)),\ \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\ t_0=0,\ t_f=T.$$

Time and τ

$$\tau_j=-\cos\left(\frac{j\pi}{M}\right),\quad j=0,1,2,\ldots,M.$$

$$t=\frac{(t_f+t_0)}{2}+\frac{(t_f-t_0)}{2}\tau=w_1+w_2\tau.$$

Picard Iteration

$$\frac{d^2\boldsymbol{x}(t)}{d^2\tau}=w_2^2\,\boldsymbol{f}(t,\boldsymbol{x}(t))$$

$$\boldsymbol{\beta}^i_{pseudo}=[P_1][A][F^{i-1}][[W_2]],\ \boldsymbol{\alpha}^i_{pseudo}=[P_2]\boldsymbol{\beta}^i_{pseudo}[[W_2]].$$

$$\boldsymbol{v}_0=\frac{\boldsymbol{x}(1)-\boldsymbol{x}(-1)}{t_f-t_0}-\frac{1}{t_f-t_0}[T(1)]\begin{bmatrix}\boldsymbol{\alpha}^i_{0\,pseudo}\\ \vdots\\ \boldsymbol{\alpha}^i_{N\,pseudo}\end{bmatrix}$$

Iterate

$$\boldsymbol{\beta}^i=\boldsymbol{V}_0+\boldsymbol{\beta}^i_{pseudo},\ \boldsymbol{v}^i(t)=T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{\alpha}^i=\boldsymbol{X}_0+[P_2]\boldsymbol{\beta}^i[[W_2]],\ \boldsymbol{x}^i(t)=T(\tau)\boldsymbol{\alpha}^i.$$

Convergence

$$e=\max\left(\left[\max\left(\frac{\left|\boldsymbol{x}^i(t)-\boldsymbol{x}^{i-1}(t)\right|}{\left|\boldsymbol{x}^i(t)\right|}\right),\max\left(\frac{\left|\boldsymbol{v}^i(t)-\boldsymbol{v}^{i-1}(t)\right|}{\left|\boldsymbol{v}^i(t)\right|}\right)\right]\right)$$

Update

$$\boldsymbol{x}^{i+1}(t)=\boldsymbol{x}^i(t),\ \boldsymbol{v}^{i+1}(t)=\boldsymbol{v}^i(t).$$

Velocity Matrices

$$[P_1]=[[I]-[L]][S]$$

$$\boldsymbol{V}_0=\begin{bmatrix}\boldsymbol{v}_0\\0\\\vdots\\0\end{bmatrix}\quad [L]=\begin{bmatrix}T(-1)\\0\\\vdots\\0\end{bmatrix}$$

$$T(\tau)=\begin{bmatrix}T_0(\tau_0)&\cdots&T_{N-1}(\tau_0)\\T_0(\tau_1)&\cdots&T_{N-1}(\tau_1)\\\vdots&\ddots&\vdots\\T_0(\tau_M)&\cdots&T_{N-1}(\tau_M)\end{bmatrix}$$

$$[S]=\begin{bmatrix}\frac{1}{4}&0&&&\cdots&&&0\\1&0&-1/2&0&&&&\\0&\frac{1}{4}&0&-\frac{1}{4}&0&&&\\&0&\ddots&\ddots&\ddots&0&&\\\vdots&\vdots&\vdots&\frac{1}{2k}&0&-\frac{1}{2k}&\vdots&\vdots\\&&&&\ddots&\ddots&\ddots&0\\&&&&&\frac{1}{2(N-3)}&0&\frac{-1}{2(N-2)}\\&&&&&0&\frac{1}{2(N-2)}&0\\0&&&&\cdots&&0&\frac{1}{2(N-1)}\end{bmatrix}$$

Position Matrices

$$[P_2]=[[I]-[L]][S]$$

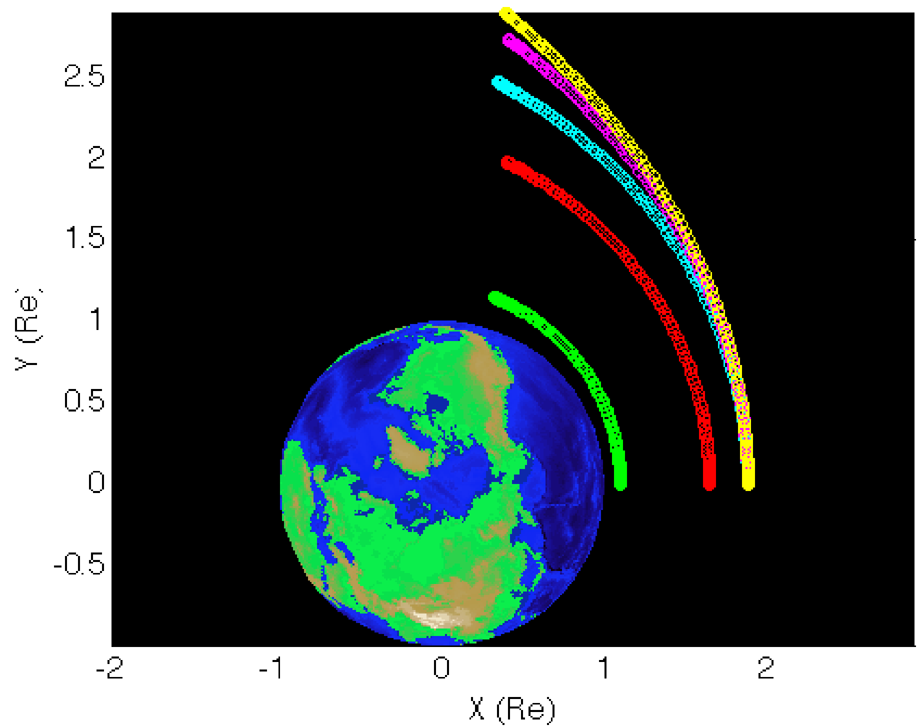
$$\boldsymbol{X}_0=\begin{bmatrix}\boldsymbol{x}_0\\0\\\vdots\\0\end{bmatrix}\quad [L]=\begin{bmatrix}T(-1)\\0\\\vdots\\0\end{bmatrix}$$

$$T(\tau)=\begin{bmatrix}T_0(\tau_0)&\cdots&T_N(\tau_0)\\T_0(\tau_1)&\cdots&T_N(\tau_1)\\\vdots&\ddots&\vdots\\T_0(\tau_M)&\cdots&T_N(\tau_M)\end{bmatrix}$$

$$[S]=\begin{bmatrix}\frac{1}{4}&0&&&\cdots&&&0\\1&0&-1/2&0&&&&\\0&\frac{1}{4}&0&-\frac{1}{4}&0&&&\\&0&\ddots&\ddots&\ddots&0&&\\\vdots&\vdots&\vdots&\frac{1}{2k}&0&-\frac{1}{2k}&\vdots&\vdots\\&&&&\ddots&\ddots&\ddots&0\\&&&&&\frac{1}{2(N-2)}&0&\frac{-1}{2(N-1)}\\&&&&&0&\frac{1}{2(N-1)}&0\\0&&&&\cdots&&0&\frac{1}{2N}\end{bmatrix}$$

EXAMPLE 5

TPBVP Example: Perturbed Two-body Problem



$$\ddot{\boldsymbol{r}}(t) = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_d(x,y,z)$$

-o- a = 8000 km, e = 0.125, $t_f \approx$ 20 mins
-o- a = 15,000 km, e = 0.3, $t_f \approx$ 39 mins
-o- a = 20,000 km, e = 0.4, $t_f \approx$ 51 mins
-o- a = 30,000 km, e = 0.6, $t_f \approx$ 51 mins
-o- a = 40,000 km, e = 0.7, $t_f \approx$ 53 mins

The Picard-Chebyshev TPBVP algorithm only converges over a fraction of an orbit (as seen above). The arcs were computed with a ***spherical harmonic degree & order 40 gravity model.*** Although the Picard-Chebyshev TPBVP has a relatively small domain of convergence (compared with the IVP) is not a Newton-like shooting method and does not require a state transition matrix. As a result it is very fast and is ideal for solving "short range" type problems. The code for generating the above figure is available for use as a learning tool: **run_lecture3_example5_tpbvpII.m**.

BVP Type II & Type III

BVP Type II ($\boldsymbol{x}_0$ and $\boldsymbol{v}_f$)

Second order system & boundary conditions

$$\ddot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{v}(t)),$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{v}(t_f) = \boldsymbol{v}_f, \quad \boldsymbol{x}(t), \boldsymbol{v}(t) \in R^{1\times n}.$$

Compute Velocity: ***Final Value Problem***

$$\boldsymbol{v}^i(\tau) = \boldsymbol{v}(1) + \int_s^1 \boldsymbol{f}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq.$$

$$\boldsymbol{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s) = \boldsymbol{v}(1) + \int_s^1 \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq.$$

Compute Position: ***Initial Value Problem***

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boldsymbol{v}(1) + \int_s^1 \boldsymbol{f}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq \right\} ds.$$

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \left\{ \boldsymbol{v}(1) + \int_s^1 \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\}.$$

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s)ds.$$

BVP Type III ($\boldsymbol{x}_f$ and $\boldsymbol{v}_0$)

Second order system & boundary conditions

$$\ddot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{v}(t)),$$

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f, \quad \boldsymbol{v}(t_0) = \boldsymbol{v}_0, \quad \boldsymbol{x}(t), \boldsymbol{v}(t) \in R^{1\times n}.$$

Compute Velocity: ***Initial Value Problem***

$$\boldsymbol{v}^i(\tau) = \boldsymbol{v}(-1) + \int_{-1}^{s} \boldsymbol{f}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq.$$

$$\boldsymbol{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s) = \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq.$$

Compute Position: ***Final Value Problem***

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}(1) + \int_{\tau}^{1} \left\{ \boldsymbol{v}(-1) + \int_{-1}^{s} \boldsymbol{f}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq \right\} ds.$$

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(1) + \int_{\tau}^{1} \left\{ \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{a}_k^{i-1} T_k(q)dq \right\}.$$

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(1) + \int_{\tau}^{1} \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s)ds.$$

BVP Type II & Type III

BVP Type II ($\boldsymbol{x}_0$ and $\boldsymbol{v}_f$)

Velocity $\mathbf{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s) = \mathbf{v}(1) + \int_s^1 \sum_{k=0}^{N-2} \mathbf{a}_k^{i-1} T_k(q)dq.$

FVP: $\tau(t) = 1 - 2\frac{(t-t_0)}{(t_f-t_0)}$, if $t=t_0$, $\tau=1$, if $t=t_f$, $\tau=-1$ and $\frac{d\tau}{dt} = -\frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix} = \boldsymbol{V}_f + [[U]-[I]][S][A][F^{i-1}][-W_2],\ \mathbf{v}^i(t) = T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{V}_f = \begin{bmatrix} \mathbf{v}_1(1) & \cdots & \mathbf{v}_n(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},\ U = \begin{bmatrix} T_0(1) & \cdots & T_N(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

Position $\mathbf{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \mathbf{x}(-1) + \int_{-1}^{\tau} \left\{ \mathbf{v}(1) + \int_s^1 \sum_{k=0}^{N-2} \mathbf{a}_k^{i-1} T_k(q)dq \right\}.$

IVP: $\tau(t) = -1 + 2\frac{(t-t_0)}{(t_f-t_0)}$, if $t=t_0$, $\tau=-1$, if $t=t_f$, $\tau=1$ and $\frac{d\tau}{dt} = \frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \boldsymbol{\alpha}_0^i \\ \vdots \\ \boldsymbol{\alpha}_N^i \end{bmatrix} = \boldsymbol{X}_0 + [[I]-[L]][S]\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix}[W_2],\ \mathbf{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i.$$

$$\boldsymbol{X}_0 = \begin{bmatrix} \mathbf{x}_1(-1) & \cdots & \mathbf{x}_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},\ L = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

BVP Type III ($\boldsymbol{x}_f$ and $\boldsymbol{v}_0$)

Velocity $\mathbf{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s) = \mathbf{v}(-1) + \int_{-1}^s \sum_{k=0}^{N-2} \mathbf{a}_k^{i-1} T_k(q)dq.$

IVP: $\tau(t) = -1 + 2\frac{(t-t_0)}{(t_f-t_0)}$, if $t=t_0$, $\tau=-1$, if $t=t_f$, $\tau=1$ and $\frac{d\tau}{dt} = \frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix} = \boldsymbol{V}_0 + [[I]-[L]][S][A][F^{i-1}][W_2],\ \mathbf{v}^i(t) = T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{V}_0 = \begin{bmatrix} \mathbf{v}_1(-1) & \cdots & \mathbf{v}_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},\ L = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

Velocity $\mathbf{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \mathbf{x}(1) + \int_{\tau}^{1} \left\{ \mathbf{v}(-1) + \int_{-1}^s \sum_{k=0}^{N-2} \mathbf{a}_k^{i-1} T_k(q)dq \right\}.$

FVP: $\tau(t) = 1 - 2\frac{(t-t_0)}{(t_f-t_0)}$, if $t=t_0$, $\tau=1$, if $t=t_f$, $\tau=-1$ and $\frac{d\tau}{dt} = -\frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \boldsymbol{\alpha}_0^i \\ \vdots \\ \boldsymbol{\alpha}_N^i \end{bmatrix} = \boldsymbol{X}_f + [[U]-[I]][S]\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix}[-W_2],\ \mathbf{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i.$$

$$\boldsymbol{X}_f = \begin{bmatrix} \mathbf{x}_1(1) & \cdots & \mathbf{x}_n(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},\ U = \begin{bmatrix} T_0(1) & \cdots & T_N(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

BVP II ALGORITHM

Dynamics & Initial Conditions

$$\frac{d^2\boldsymbol{x}}{dt^2}(t)=\boldsymbol{f}(t,\boldsymbol{x}(t)),\ \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\ t_0=0,\ t_f=T.$$

Time and τ

$$\tau_j=-\cos\left(\frac{j\pi}{M}\right),\ \ j=0,1,2,\dots,M.$$

$$t=\frac{(t_f+t_0)}{2}+\frac{(t_f-t_0)}{2}\tau=w_1+w_2\tau.$$

Picard Iteration

$$\frac{d^2\boldsymbol{x}(t)}{d^2\tau}=w_2^2\,\boldsymbol{f}(t,\boldsymbol{x}(t))$$

$$\boldsymbol{\beta}^i=\boldsymbol{V}_f+[P_1][A][F^{i-1}][-W_2],\ \boldsymbol{v}^i(t)=T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{\alpha}^i=\boldsymbol{X}_0+[P_2]\boldsymbol{\beta}^i[W_2],\ \boldsymbol{x}^i(t)=T(\tau)\boldsymbol{\alpha}^i.$$

Iterate

Convergence

$$e=\max\left(\left[\max\left(\frac{\left|\boldsymbol{x}^i(t)-\boldsymbol{x}^{i-1}(t)\right|}{\left|\boldsymbol{x}^i(t)\right|}\right),\max\left(\frac{\left|\boldsymbol{v}^i(t)-\boldsymbol{v}^{i-1}(t)\right|}{\left|\boldsymbol{v}^i(t)\right|}\right)\right]\right)$$

Update

$$\boldsymbol{x}^{i+1}(t)=\boldsymbol{x}^i(t),\ \boldsymbol{v}^{i+1}(t)=\boldsymbol{v}^i(t).$$

Velocity Matrices

$$[P_1]=[[U]-[I]][S]$$

$$\boldsymbol{V}_f=\begin{bmatrix}\boldsymbol{v}_f\\0\\\vdots\\0\end{bmatrix}\qquad [U]=\begin{bmatrix}T(1)\\0\\\vdots\\0\end{bmatrix}$$

$$T(\tau)=\begin{bmatrix}T_0(\tau_0)&\cdots&T_{N-1}(\tau_0)\\T_0(\tau_1)&\cdots&T_{N-1}(\tau_1)\\\vdots&\ddots&\vdots\\T_0(\tau_M)&\cdots&T_{N-1}(\tau_M)\end{bmatrix}$$

$$[S]=\begin{bmatrix}\frac{1}{4}&0&&&\cdots&&&&0\\1&0&-1/2&0&&&&&\\0&\frac{1}{4}&0&-\frac{1}{4}&0&&&&\\&0&\ddots&\ddots&\ddots&0&&&\\\vdots&\vdots&\vdots&\frac{1}{2k}&0&-\frac{1}{2k}&&\vdots&\vdots\\&&&&\ddots&\ddots&&\ddots&0\\&&&&&\frac{1}{2(N-3)}&&0&\frac{-1}{2(N-2)}\\&&&&&0&&\frac{1}{2(N-2)}&0\\0&&&&\cdots&&&0&\frac{1}{2(N-1)}\end{bmatrix}$$

Position Matrices

$$[P_2]=[[I]-[L]][S]$$

$$\boldsymbol{X}_0=\begin{bmatrix}\boldsymbol{x}_0\\0\\\vdots\\0\end{bmatrix}\qquad [L]=\begin{bmatrix}T(-1)\\0\\\vdots\\0\end{bmatrix}$$

$$T(\tau)=\begin{bmatrix}T_0(\tau_0)&\cdots&T_N(\tau_0)\\T_0(\tau_1)&\cdots&T_N(\tau_1)\\\vdots&\ddots&\vdots\\T_0(\tau_M)&\cdots&T_N(\tau_M)\end{bmatrix}$$

$$[S]=\begin{bmatrix}\frac{1}{4}&0&&&\cdots&&&&0\\1&0&-1/2&0&&&&&\\0&\frac{1}{4}&0&-\frac{1}{4}&0&&&&\\&0&\ddots&\ddots&\ddots&0&&&\\\vdots&\vdots&\vdots&\frac{1}{2k}&0&-\frac{1}{2k}&&\vdots&\vdots\\&&&&\ddots&\ddots&&\ddots&0\\&&&&&\frac{1}{2(N-2)}&&0&\frac{-1}{2(N-1)}\\&&&&&0&&\frac{1}{2(N-1)}&0\\0&&&&\cdots&&&0&\frac{1}{2N}\end{bmatrix}$$

BVP III ALGORITHM

Dynamics & Initial Conditions

$$\frac{d^2\boldsymbol{x}}{dt^2}(t) = \boldsymbol{f}(t,\boldsymbol{x}(t)),\ \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\ t_0=0,\ t_f=T.$$

Time and τ

$$\tau_j = -\cos\left(\frac{j\pi}{M}\right),\ \ j=0,1,2,\dots,M.$$

$$t = \frac{(t_f+t_0\ )}{2} + \frac{(t_f-t_0\ )}{2}\tau = w_1 + w_2\tau.$$

Picard Iteration

$$\frac{d^2\boldsymbol{x}(t)}{d^2\tau} = w_2^2\,\boldsymbol{f}(t,\boldsymbol{x}(t))$$

$$\boldsymbol{\beta}^i = \boldsymbol{V}_0 + [P_1][A][F^{i-1}][W_2],\ \ \boldsymbol{v}^i(t) = T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{\alpha}^i = \boldsymbol{X}_f + [P_2]\boldsymbol{\beta}^i[-W_2],\ \ \boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i.$$

Iterate

Convergence

$$e = \max\left(\left[\max\left(\frac{\left|\boldsymbol{x}^i(t)-\boldsymbol{x}^{i-1}(t)\right|}{\left|\boldsymbol{x}^i(t)\right|}\right), \max\left(\frac{\left|\boldsymbol{v}^i(t)-\boldsymbol{v}^{i-1}(t)\right|}{\left|\boldsymbol{v}^i(t)\right|}\right)\right]\right)$$

Update

$$\boldsymbol{x}^{i+1}(t) = \boldsymbol{x}^i(t),\ \ \boldsymbol{v}^{i+1}(t) = \boldsymbol{v}^i(t).$$

Velocity Matrices

$$[P_1] = [[I]-[L]][S]$$

$$\boldsymbol{V}_0 = \begin{bmatrix} \boldsymbol{v}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad [L] = \begin{bmatrix} T(-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$T(\tau) = \begin{bmatrix} T_0(\tau_0) & \cdots & T_{N-1}(\tau_0) \\ T_0(\tau_1) & \cdots & T_{N-1}(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_{N-1}(\tau_M) \end{bmatrix}$$

$$[S] = \begin{bmatrix} \frac{1}{4} & 0 & & & \cdots & & & 0 \\ 1 & 0 & -1/2 & 0 & & & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & \\ & 0 & \ddots & \ddots & \ddots & 0 & & \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \frac{1}{2(N-3)} & 0 & \frac{-1}{2(N-2)} \\ & & & & & 0 & \frac{1}{2(N-2)} & 0 \\ 0 & & & & \cdots & & 0 & \frac{1}{2(N-1)} \end{bmatrix}$$

Position Matrices

$$[P_2] = [[U]-[I]][S]$$

$$\boldsymbol{X}_f = \begin{bmatrix} \boldsymbol{x}_f \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad [U] = \begin{bmatrix} T(1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$T(\tau) = \begin{bmatrix} T_0(\tau_0) & \cdots & T_N(\tau_0) \\ T_0(\tau_1) & \cdots & T_N(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_N(\tau_M) \end{bmatrix}$$

$$[S] = \begin{bmatrix} \frac{1}{4} & 0 & & & \cdots & & & 0 \\ 1 & 0 & -1/2 & 0 & & & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & \\ & 0 & \ddots & \ddots & \ddots & 0 & & \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \frac{1}{2(N-2)} & 0 & \frac{-1}{2(N-1)} \\ & & & & & 0 & \frac{1}{2(N-1)} & 0 \\ 0 & & & & \cdots & & 0 & \frac{1}{2N} \end{bmatrix}$$

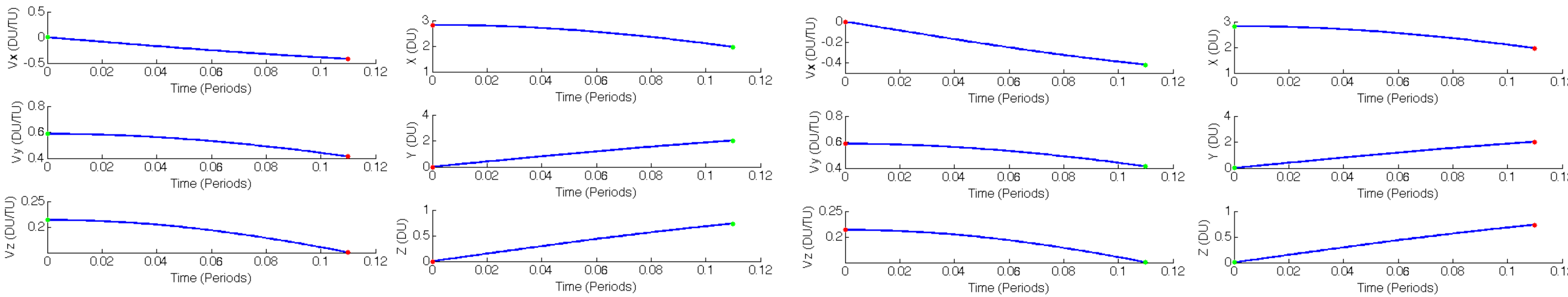
EXAMPLE: BVP Type II & III

BVP Type II ($\boldsymbol{x}_0$ and $\boldsymbol{v}_f$)

$$\boxed{\ddot{\boldsymbol{r}}(t) = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_d(x,y,z)}$$

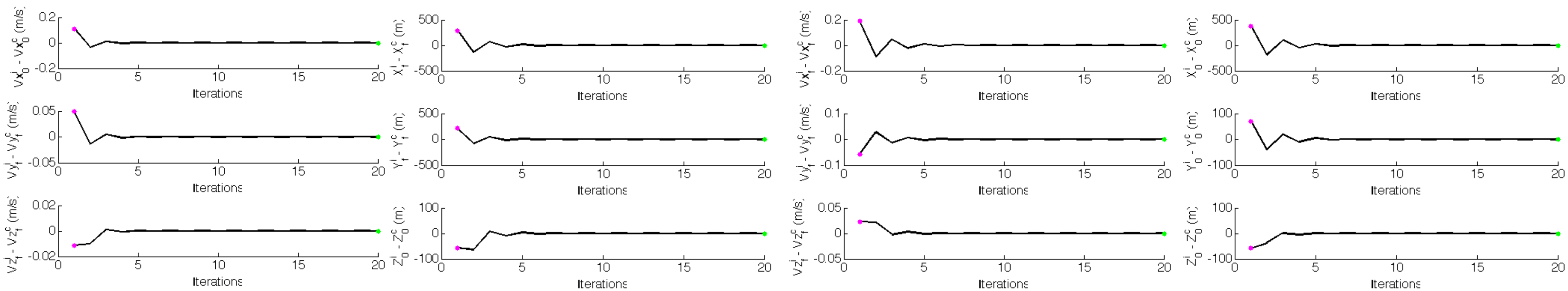
BVP Type III ($\boldsymbol{x}_f$ and $\boldsymbol{v}_0$)

Velocity Components **Position Components** **Velocity Components** **Position Components**



LEGEND: ● Specified (fixed) boundary condition ● Converged boundary condition ● Two-body initial guess

Initial Velocity **Final Position** **Final Velocity** **Initial Position**



The Picard-Chebyshev BVP algorithm also converges over a fraction of an orbit (slightly less that the TPBVP). As with the TPBVP it is very fast as it is not a Newton-like shooting method and it does not require a state transition matrix. The code for generating the above figures is available for use as a learning tool: **run_lecture3_example5_ivpII_fvpII.m**.

PICARD-CHEBYSHEV CONVERGENCE: FIRST ORDER

Scalar Problem

- Consider the first order linear differential equation: consider the simplest case, x is a scalar.

$$\frac{d\boldsymbol{x}(t)}{dt} = c\boldsymbol{x}(t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{x} \in R^{1\times n}.$$

$$\boxed{\begin{aligned} &\tau = -1 + 2(t - t_0)/(t_f - t_0); \quad -1 \le \tau \le 1 \\ &t = t_0 + (\tau + 1)(t_f - t_0)/2; \quad t_0 \le t \le t_f \end{aligned}}$$

Picard-Chebyshev Vector Matrix Notation with *t* => τ

$$\boldsymbol{x}^i = \left(\frac{t_f - t_0}{2}\right) \overset{(N+1)\times(N+1)}{T} \overset{\substack{\text{integration operator} \\ (N+1)\times(N+1)}}{P_1} \overset{\substack{\text{least square operator} \\ (N+1)\times(N+1)}}{A} \left(c\boldsymbol{x}^{i-1}\right) + \boldsymbol{x}_0;$$

where $\boldsymbol{x} = [x(\tau_0)\cdots x(\tau_N)]^T;$ $\quad T = \begin{bmatrix} T_0(\tau_0) & \cdots & T_N(\tau_0) \\ \vdots & \ddots & \vdots \\ T_0(\tau_N) & \cdots & T_N(\tau_N) \end{bmatrix}$

or

$$\boxed{\boldsymbol{x}^i = c\left(\frac{t_f - t_0}{2}\right) \overset{\substack{constant \equiv M \\ (N+1)\times(N+1)}}{[TP_1A]} \boldsymbol{x}^{i-1} + \boldsymbol{x}_0};$$

$$\boldsymbol{x}_0 = [x_0 \quad 0 \ \cdots \ 0]^T$$

- If max eigenvalue of $M < 1$, Picard converges over finite interval (analogous to diff eqs).
- Max eigenvalues are scaled by the time of flight $t_f - t_0$ and c:

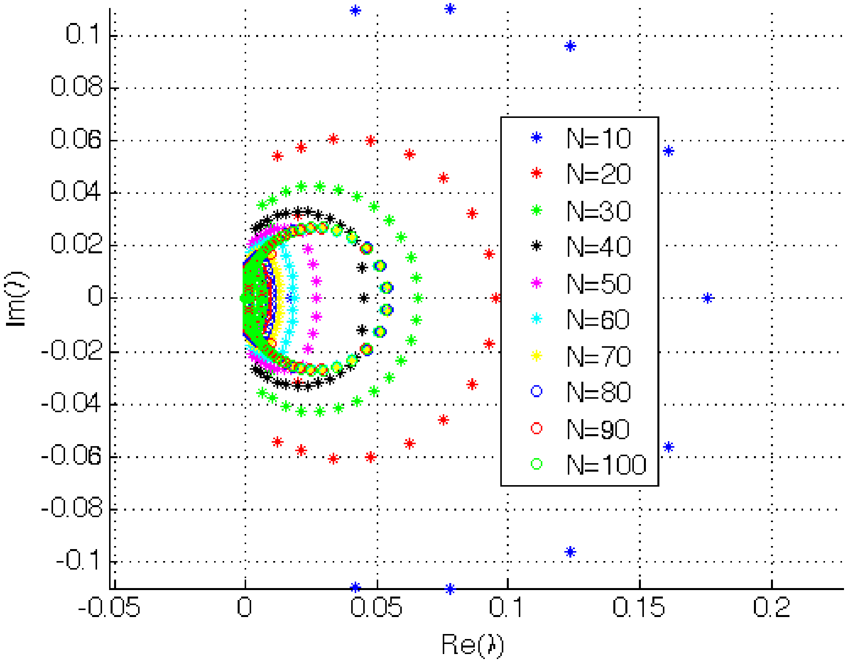
$$\boxed{\left|\left(\frac{c(t_f - t_0)}{2}\right)\lambda_{\max}[M]\right| < 1 \quad \text{or} \quad |t_f - t_0| < \frac{2}{|c\lambda_{\max}[M]|}}$$

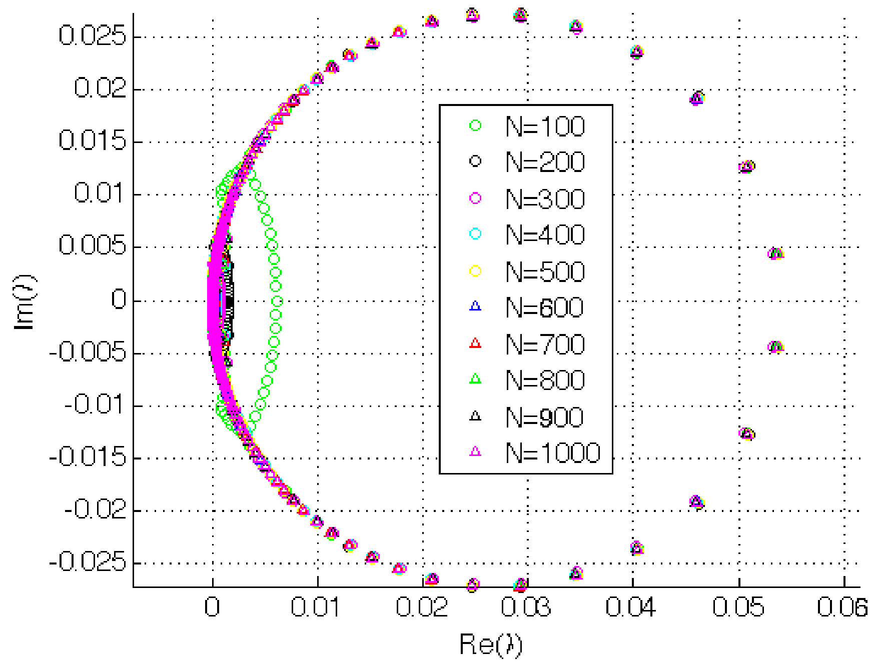
Note

- For a linear system, given c we can directly compute the domain of convergence $t_f - t_0$.

EIGENVALUE ANALYSIS: FIRST ORDER

Root Locus Plots for $\lambda_{max}(TP_1A)$

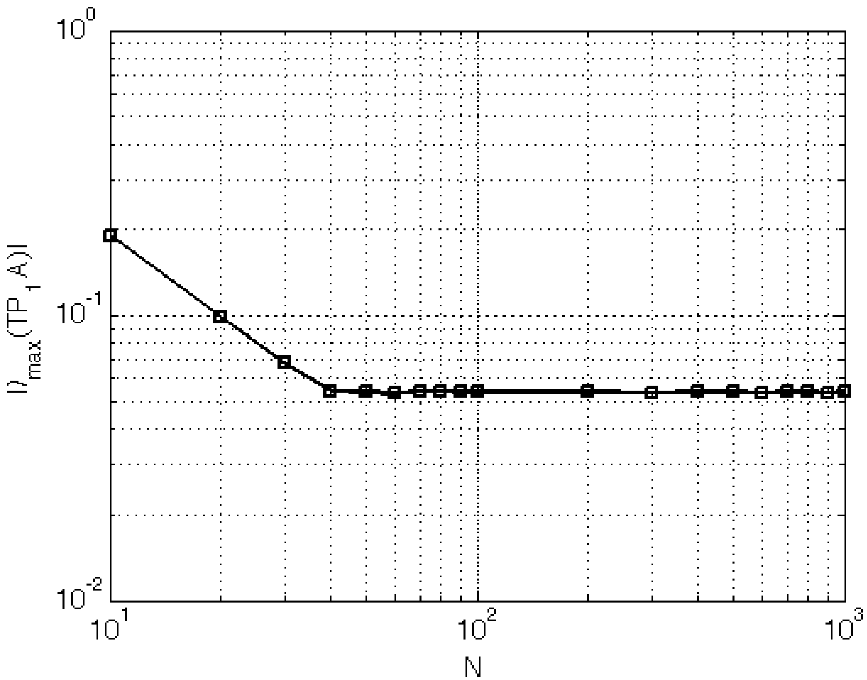




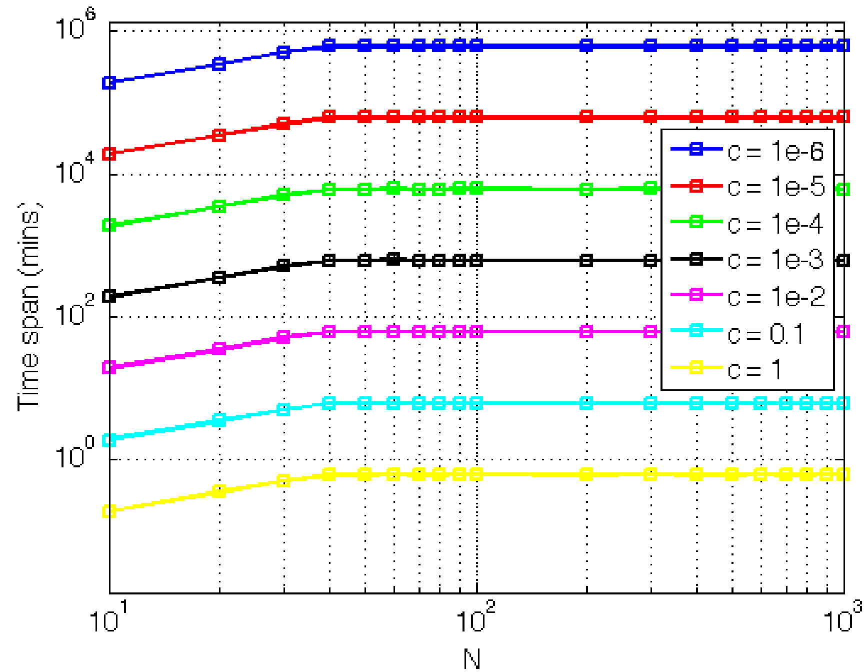
- For $N > 40$, the ***maximum eigenvalue*** of the matrix product [TP_1A] or $\lambda_{max}([TP_1A])$ is attracted to a ***fixed point*** on the root locus plots above.
- The code for generating the above figures is available for use as a learning tool: **run_lecture3_example6_ivpI_conv.m**.

EIGENVALUE ANALYSIS: FIRST ORDER

MAX EIGENVALUE VS N



DOMAIN OF CONVERGENCE



- The left figure shows that increasing N beyond 40 will ***not increase*** the ***convergence rate***, however, it may ***improve*** the ***accuracy*** of the fit/solution for functions requiring $M >= N >$ 40 to capture the higher frequency behavior accurately over a specific time interval.
- Increasing N beyond 40 also ***does not increase*** the ***theoretical time interval*** over which Picard-Chebyshev numerical integration will converge for a given c.
- The code for generating the above figures is available for use as a learning tool: **run_lecture3_example6_ivpI_conv.m**.

PICARD-CHEBYSHEV CONVERGENCE: SECOND ORDER

Scalar Problem

- Consider the first order linear differential equation: $\dfrac{d\boldsymbol{x}^2(t)}{dt^2} = c\boldsymbol{x}(t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{v}(t_0) = \boldsymbol{v}_0, \quad \boldsymbol{x} \in R^{1\times n}.$

Picard-Chebyshev Vector Matrix Notation

$$\text{Solution} \longrightarrow \boldsymbol{x}^i = \underbrace{\left(\frac{t_f - t_0}{2}\right)^2}_{t \to \tau \text{ time transformation}} \overset{\substack{\text{Chebyshev matrix}\\ \downarrow}}{T(\tau)} \underbrace{P_2 P_1}_{\text{Integration operators: I \& II}} \overset{\substack{\text{Least squares operator}\\ \downarrow}}{A} \underbrace{\left(c\boldsymbol{x}(t)\right)}_{\text{Forcing function}} + \boldsymbol{x}_0 \longleftarrow \text{Initial condition vector}$$

- If the max eigenvalue < 1, Picard sequence converges (analogous to difference equations).
- The matrix product $T(\tau)P_2P_1A$ is constant once N is selected.
- Max eigenvalues are scaled by the time of flight $t_f - t_0$ and c.

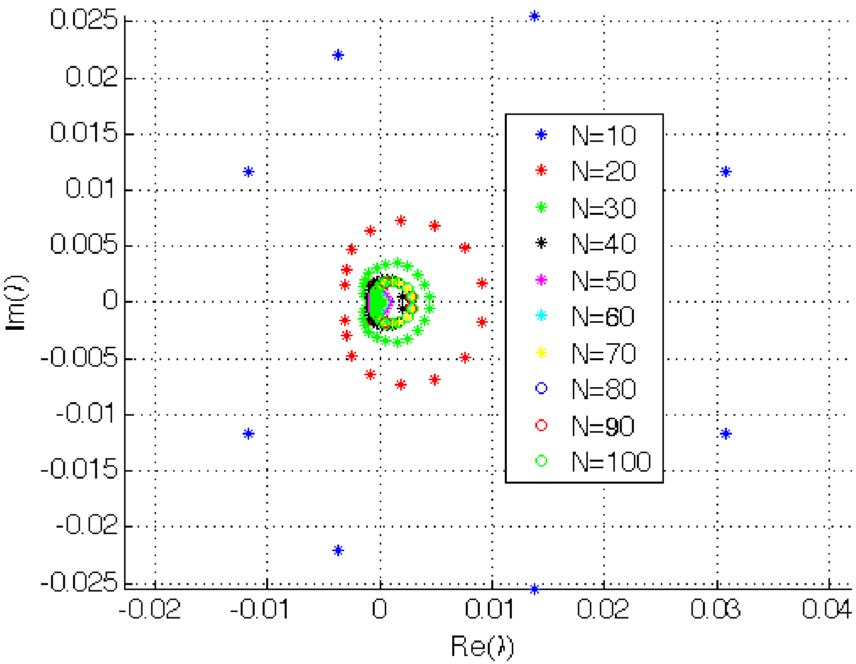
$$\left|\left(\frac{t_f - t_0}{2}\right)^2 c\lambda_{\max}\left(T(\tau)P_2P_1A\right)\right| < 1 \quad \text{or} \quad \left|t_f - t_0\right| < \frac{2}{\sqrt{\left|c\lambda_{\max}\left(T(\tau)P_2P_1A\right)\right|}}$$

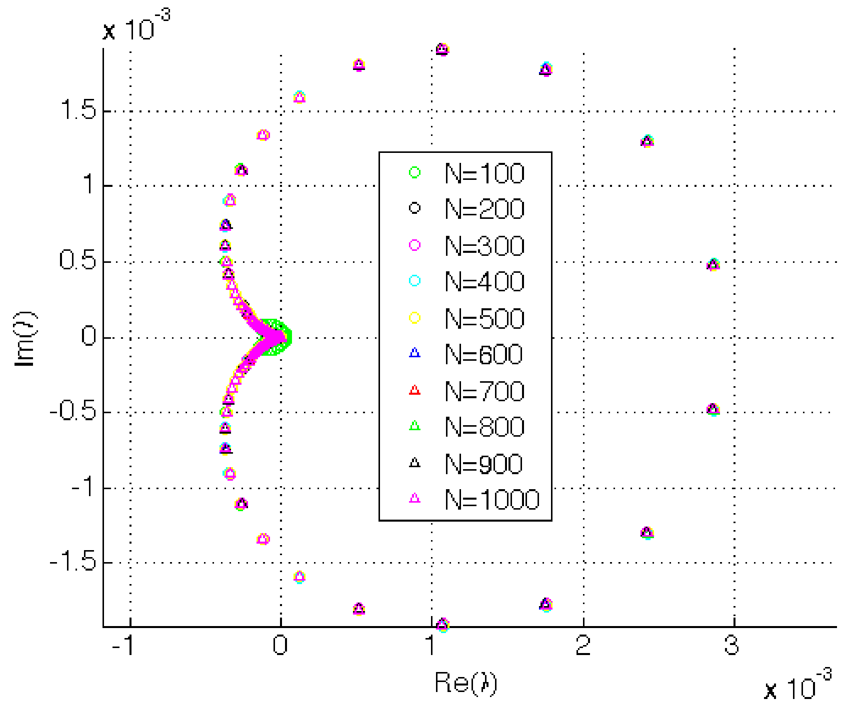
Notation

- For a linear system, given c we can directly compute the domain of convergence $t_f - t_0$.

EIGENVALUE ANALYSIS: SECOND ORDER

Root Locus Plots for $\lambda_{max}(TP_2P_1A)$

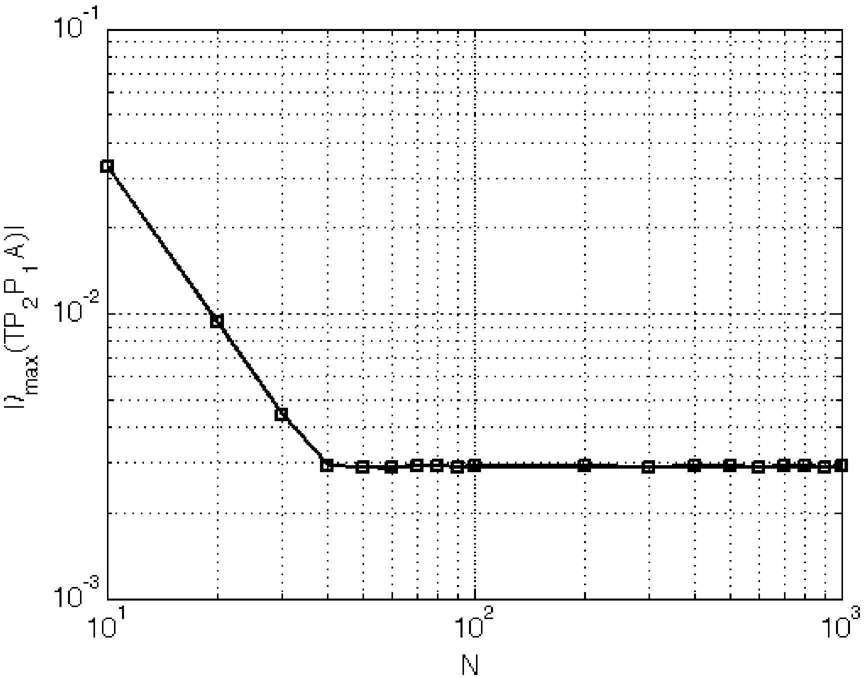




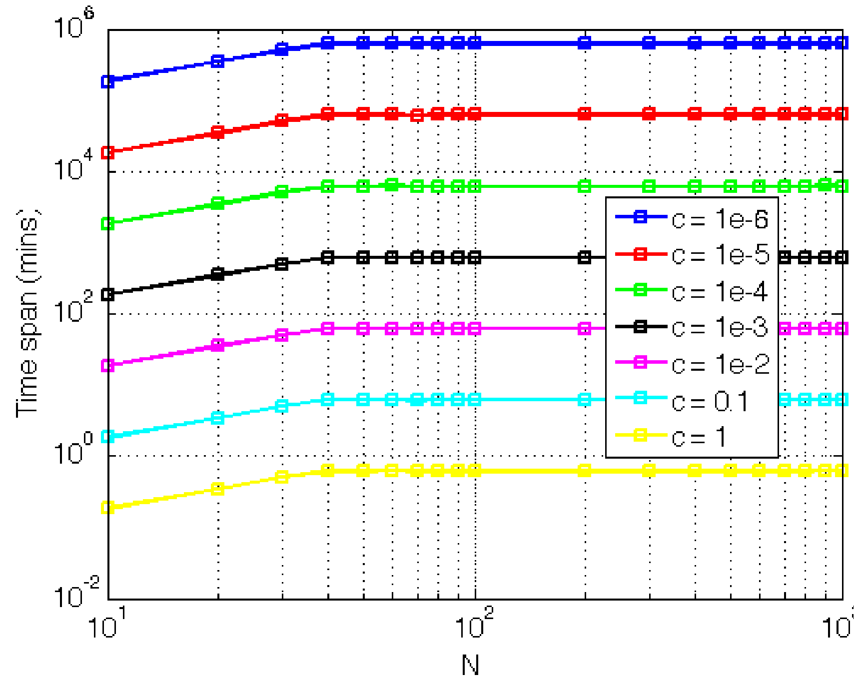
- For $N > 40$, the ***maximum eigenvalue*** of the matrix product $[TP_2P_1A]$ or $\lambda_{max}([TP_2P_1A])$ is attracted to a ***fixed point*** on the root locus plots above.
- Note that $\lambda_{max}([TP_2P_1A])$ for the ***second order system*** is much ***smaller*** than for the first order system, however, it is the ***square root*** of this value (as shown in the denominator on the previous slide) that is used for computing the ***theoretical convergence***.
- The code for generating the above figures is available for use as a learning tool: **run_lecture3_example7_ivpII_conv.m**.

EIGENVALUE ANALYSIS: SECOND ORDER

MAX EIGENVALUE VS N



DOMAIN OF CONVERGENCE



- The left figure shows that increasing N beyond 40 will ***not increase*** the ***convergence rate***, however, it may ***improve*** the ***accuracy*** of the fit/solution for functions requiring $M >= N >$ 40 to capture the higher frequency behavior accurately over a specific time interval.
- Increasing N beyond 40 also ***does not increase*** the ***theoretical time interval*** over which Picard-Chebyshev numerical integration will converge for a given c.
- The code for generating the above figures is available for use as a learning tool: **run_lecture3_example7_ivpII_conv.m**.

PICARD-CHEBYSHEV CONVERGENCE: TPBVP

Scalar Problem

- Consider the first order linear differential equation: $\frac{d\mathbf{x}^2(t)}{dt^2} = c\mathbf{x}(t),\quad \mathbf{x}(t_0) = \mathbf{x}_0,\quad \mathbf{x}(t_f) = \mathbf{x}_f,\quad \mathbf{x} \in R^{1\times n}.$

Picard-Chebyshev Vector Matrix Notation

$$\text{Solution} \rightarrow \mathbf{x}^i = \underbrace{\left(\frac{t_f - t_0}{2}\right)^2}_{t \rightarrow \tau \text{ time transformation}} \overset{\substack{\text{Chebyshev}\\ \text{matrix}\\ \downarrow}}{T(\tau)} \underbrace{P_B P_2 P_1}_{\text{Integration operators: I, II \& } P_B} \overset{\substack{\text{Least squares}\\ \text{operator}\\ \downarrow}}{A} \underbrace{\left(c\mathbf{x}(t)\right)}_{\substack{\text{Forcing}\\ \text{function}}} + \mathbf{x}_{0f} \leftarrow \substack{\text{Initial condition}\\ \text{vector}}$$

- If the max eigenvalue < 1, Picard sequence converges (analogous to difference equations).
- The matrix product $T(\tau)P_B P_2 P_1 A$ is constant once N is selected.
- Max eigenvalues are scaled by the time of flight $t_f - t_0$ and c.

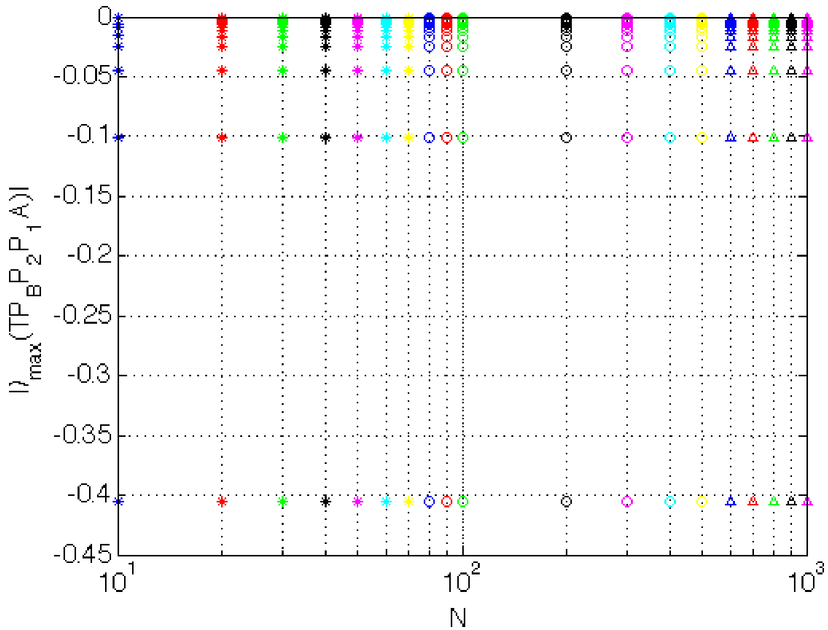
$$\left|\left(\frac{t_f - t_0}{2}\right)^2 c\lambda_{\max}\left(T(\tau)P_B P_2 P_1 A\right)\right| < 1 \quad \text{or} \quad \left|t_f - t_0\right| < \frac{2}{\sqrt{\left|c\lambda_{\max}\left(T(\tau)P_B P_2 P_1 A\right)\right|}}$$

Notation

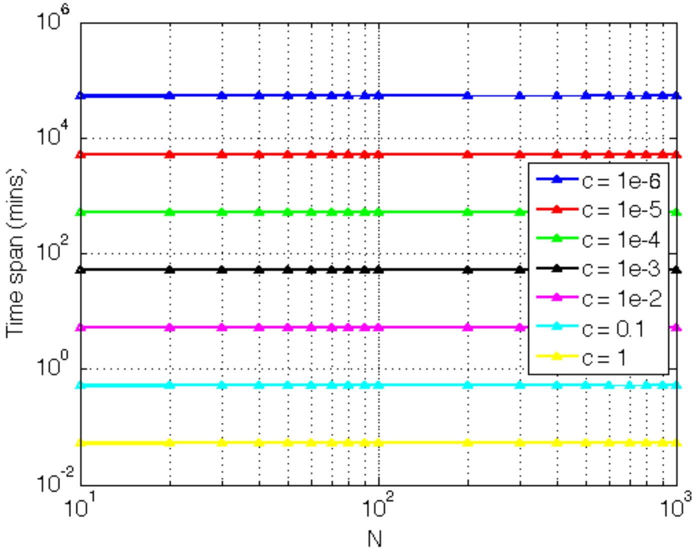
- For a linear system, given c we can directly compute the domain of convergence $t_f - t_0$.

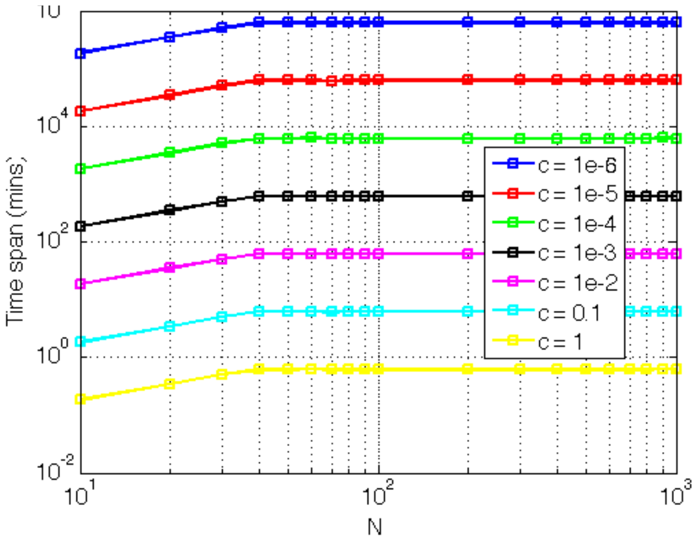
EIGENVALUE ANALYSIS: TPBVP

MAX EIGENVALUE VS N



DOMAIN OF CONVERGENCE





The ***maximum eigenvalue*** of the matrix product $\lambda_{max}([TP_BP_2P_1A])$ is constant (≈ 0.405) for increasing N, and as expected the ***theoretical convergence domain*** is also constant (top right). Note that the TPBVP theoretical domain of convergence is about an order of magnitude ***smaller*** than that for the second order IVP (bottom right).

run_lecture3_example8_tpbvp_conv.m

CONCLUSION

Picard iteration

- Picard iteration is a ***successive path approximation*** technique for solving differential equations.

Least Squares

- Reviewed of least squares from ***lecture 1*** (vector problem)
- Discussed the least squares operator

Picard-Chebyshev Initial Value Problem Derivation/Algorithm (First Order)

- Thoroughly derived the Picard-Chebyshev first order IVP algorithm
- Discussed the first integration operator (P_1)
- Presented two examples to demonstrate the method (MATLAB code is available)

Picard-Chebyshev Initial Value Problem Derivation/Algorithm (Second Order)

- Derived the Picard-Chebyshev second order IVP algorithm
- Discussed the second integration operator (P_2)
- Presented two examples to demonstrate the method (MATLAB code is available)

Picard-Chebyshev Boundary Value Problem Derivation/Algorithm

- Three types of BVPs
- Derived the Picard-Chebyshev second order BVP algorithm
- Presented three examples to demonstrate the three methods (MATLAB code is available)

Convergence Picard-Chebyshev Algorithm

- Discussed convergence for the IVP and TPBVP algorithms (MATLAB code is available)

REFERENCES

1. Bai, X., *Modified Chebyshev-Picard Iteration Methods for Solution of Initial and Boundary Value Problems*. PhD Dissertation, Texas A&M University, 2010.
2. Bai, X. and Junkins, J., *Modified Chebyshev-Picard Iteration Methods for Orbit Propagation*, Journal of the Astronautical Sciences, 2011.
3. Budd, N. and Junkins, J., *Picard-Chebyshev Integration Operator Informal Seminar Notes and Discussions*, 2016.
4. Fukushima, T., Picard iteration method, Chebyshev polynomial approximation, and global numerical integration of dynamical motions, The Astronomical Journal, 1997.
5. Bai, X. and Junkins, J., *Modified Chebyshev-Picard Iteration for Solution of Boundary Value Problems*, Journal of the Astronautical Sciences, 2012.
6. Bani-Younes, A., *Orthogonal Polynomial Approximation in Higher Dimensions: Applications in Astrodynamics*. PhD thesis, Department of Aerospace Engineering, Texas A&M University, College Station, TX, 2013.
7. Woollands, R., *Regularization and Computational Methods for Precise Solution of Perturbed Orbit Transfer Problems*. PhD thesis, Department of Aerospace Engineering, Texas A&M University, College Station, TX, 2016.
8. Schaub, H. and Junkins, J., *Analytical Mechanics of Space Systems 3rd Ed.*, AIAA Education Series, 2014.
9. Berry, M. and Healy, L., *Comparison of Accuracy Assessment Techniques for Numerical Integrator Comparison*, Advances of the Astronautical Sciences, 2003.