



Lecture 1

ORTHOGONAL APPROXIMATION

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

CONTACT INFORMATION



JOHN L. JUNKINS

Distinguished Professor of Aerospace
Engineering

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

junkins@tamu.edu



ROBYN M. WOOLLANDS

Postdoctoral Research Associate

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

robyn.woollands@gmail.com

MOTIVATION

• During recent years significant progress has been made on iterative path approximation methods for solving nonlinear ordinary differential equations that arise in mechanics and control. Several of **John Junkins**' recent PhD students have made significant contributions:

- **Xiaoli Bai, 2010** (Assist. Prof at Rutgers University)
- **Ahmad Bani Younes, 2013** (Assist Prof. at Khalifa University)
- **Donghoon Kim, 2013** (Postdoctoral Researcher at Oklahoma State University)
- **Tarek Elgohary, 2015** (Assist. Prof at University of Central Florida)
- **Brent Macomber, 2015** (GNC Engineer at SpaceX)
- **Robyn Woollands, 2016** (GNC Engineer at NASA's Jet Propulsion Lab, May 2017-)
- **Julie Read, 2016** (GNC Engineer at NASA's Johnson Space Center)
- **Austin Probe, PhD Candidate** (GNC Engineer at Emergent Technologies, Inc.)

• Our effort has led to multiple formulations and algorithms for specific circumstances/applications. These lectures provide a tutorial explanation and not-too-detailed *summary of some of these methods and algorithms as we make our **Picard Adaptive Path Approximation (PAPA)** Software Release 1.0.*

• The ***path approximation differential equation solvers*** to be presented rely on certain results from ***function approximation theory*** and ***approximation of the approximated functions' integrals and derivatives using orthogonal polynomials***. So the first two lectures address these basic issues. Examples are given throughout. While we focus mainly on one dimensional approximation, we include some extensions to n dimensions.

FUNCTION APPROXIMATION

What is function approximation?

- Function approximation can be approached by sampling the **true function** at a discrete number of points (nodes) and introducing a linear combination of basis functions to approximate the given function within a specified tolerance.

Why is this useful?

- As one example, when a function is **not analytically integrable**, but an accurate combination of **basis functions is integrable**, then the advantage is obvious.

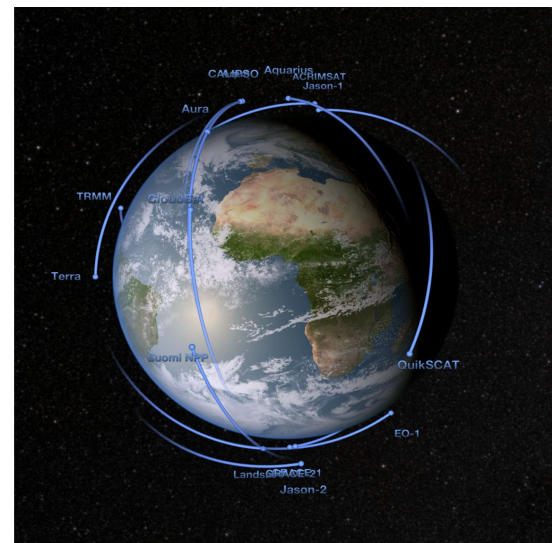
- Consider the second order, nonlinear, ordinary differential equation:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3} \mathbf{r} + \mathbf{a}_d(\mathbf{r}, \mathbf{v}, t), \quad \mathbf{v} = \dot{\mathbf{r}} \quad (1)$$

- The term $\mathbf{a}_d(\mathbf{r}, \mathbf{v}, t)$ represents perturbations: e.g., spherical harmonic gravity, drag, etc. (**More in Lecture 5**)

- Given an initial position $\mathbf{r}(t_0)$ and velocity $\mathbf{v}(t_0)$ this eqn may be solved numerically to compute $\mathbf{r}(t)$ and $\mathbf{v}(t)$.

- A remarkably stable and efficient approach (compared to existing methods), a recursive, accelerated Picard method will be introduced that allows, on the k^{th} iteration, analytical integration of the RHS of **Eq. (1)** along the $(k-1)^{\text{th}}$ path approximation.



NASA/Goddard Space Flight Center
Scientific Visualization Studio

APPROXIMATION AND RESIDUAL ERRORS

How to approximate a function?

- Approximating a function is done in a way the **minimizes SOME MEASURE of the error** (or residual) between the true and approximate functions.

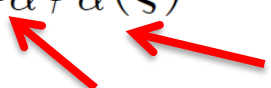
Approximation

- The function, $f(\xi)$, is approximated by a sum of $N+1$ polynomials basis functions $\{\phi_0(\xi), \phi_1(\xi), \dots, \phi_M(\xi)\}$ as shown:

$$f(\xi) \approx \sum_{\alpha=0}^N a_{\alpha} \phi_{\alpha}(\xi)$$

Basis Function
 (e.g., **Chebyshev Polynomials**)

Coefficients
 (**Chebyshev Coefficients**)




Residuals

- The residual errors are simply the difference between the function computed at each of the $M+1$ nodes and the approximated values (for specified N and basis function the residuals are a function only of the coefficients $\{a_0, a_1, \dots, a_N\}$):

$$r_j = f(\xi_j) - \sum_{\alpha=0}^N a_{\alpha} \phi_{\alpha}(\xi_j); \quad j = 0, 1, \dots, M.$$

$\xi_0, \xi_1, \dots, \xi_M$
Discrete Sample Points (Nodes)
 (e.g., **Cosine Sampling or Uniform Sampling**)

Residual **Function Evaluated at Sample Points**



VECTOR MATRIX & LEAST SQUARES

Vector Matrix Form

- Approximating a function is frequently done in a way that **minimizes a chosen norm of the residual error r** between the true and approximate functions.

$$\mathbf{r} = \mathbf{f} - \Phi \mathbf{a} \quad (2)$$

$$\mathbf{f} = \begin{bmatrix} f(\xi_0) \\ f(\xi_1) \\ \vdots \\ f(\xi_M) \end{bmatrix}, \quad \Phi = \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \cdots & \phi_N(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \cdots & \phi_N(\xi_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\xi_M) & \phi_1(\xi_M) & \cdots & \phi_N(\xi_M) \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$$

Least Squares Approximation

- Consider the usual linear equation $\mathbf{b} = A\mathbf{x}$. Solving this equation for \mathbf{x} is a standard problem in linear algebra; there may be a **unique solution** for \mathbf{x} , an **infinity of solutions**, or **no exact solution**, depending on the dimensions and rank of A .
- For the over determined case (the number of eqs exceeds the number of unknowns), we expect no exact solution, in which case, the classical approach is to find \mathbf{x} to minimize $J = \frac{1}{2} \mathbf{e}^T \mathbf{e}$; the sum of square of the residual error $\mathbf{e} = \mathbf{b} - A\mathbf{x}$. By analogy, **Eq. (2)** invites us to minimize $\frac{1}{2} \mathbf{r}^T \mathbf{r}$ to estimate \mathbf{a} .

LEAST SQUARES APPROXIMATION

Error / Residuals

- Starting with $Ax = b$, the **error** (e) that we seek to minimize is given | $e = b - Ax$.

Cost Function

- The cost function (J) measures the fit quality is $J = \frac{1}{2}e^T e = \frac{1}{2}(b - Ax)^T(b - Ax)$.

Minimization of $J = \frac{1}{2}e^T e$: To find the judicious x , we expand J and simplify to

$$J = \frac{1}{2}b^T b - b^T Ax + \frac{1}{2}x^T A^T Ax.$$

The necessary conditions are: $\frac{\partial J}{\partial x} = 0 - A^T b + A^T Ax = 0 \Rightarrow A^T Ax = A^T b$ (3)

The sufficient condition is: The matrix $A^T A$ must be positive definite.

- The **normal equations**, **Eqs (3)** can be solved for x via a variety of ways, including “brute fore inversion” (or we can use the QR, SVD, or related algorithms) as

$$x = (A^T A)^{-1} A^T b \quad \text{or, with a weight matrix:} \quad x = (A^T W A)^{-1} A^T W b.$$

MATRIX INVERSE

From the previous slide...

$$\mathbf{x} = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} \mathbf{b}$$

- Consistent with the above, minimization of $\frac{1}{2} \mathbf{r}^T \mathbf{r}$ gives: $\mathbf{a} = (\Phi^T \mathbf{W} \Phi)^{-1} \Phi^T \mathbf{W} \mathbf{f}$

Matrix Inverse

- Note that a **matrix inverse (or some other more robust but expensive and potentially troublesome method)** is required for least squares approximation. The requirement to numerically invert the normal equations is computationally expensive, especially as the polynomial order (N) increases.

Diagonal Matrix \Leftrightarrow Orthogonal Basis Function Approximation

- It turns out that if make judicious choices (consequence of orthogonality):...
 - the basis functions judiciously (e.g., **Chebyshev Polynomials**),
 - the function to be approximated is sampled using a **cosine distribution** of nodes,
 - the **weight matrix** is chosen to be $\mathbf{W} = \text{diag}[\frac{1}{2}, 1, 1, \dots, 1, 1, \frac{1}{2}]$.

Then the matrix $\Phi^T \mathbf{W} \Phi$ to be inverted is **diagonal** and the inverse is trivial (and machine precision solution for the minimizing coefficient matrix \mathbf{a} is possible).

Furthermore, for a **complete set of orthogonal basis functions**, N & M can be made sufficiently large to allow the **residual error norm to be zero, to machine precision**.

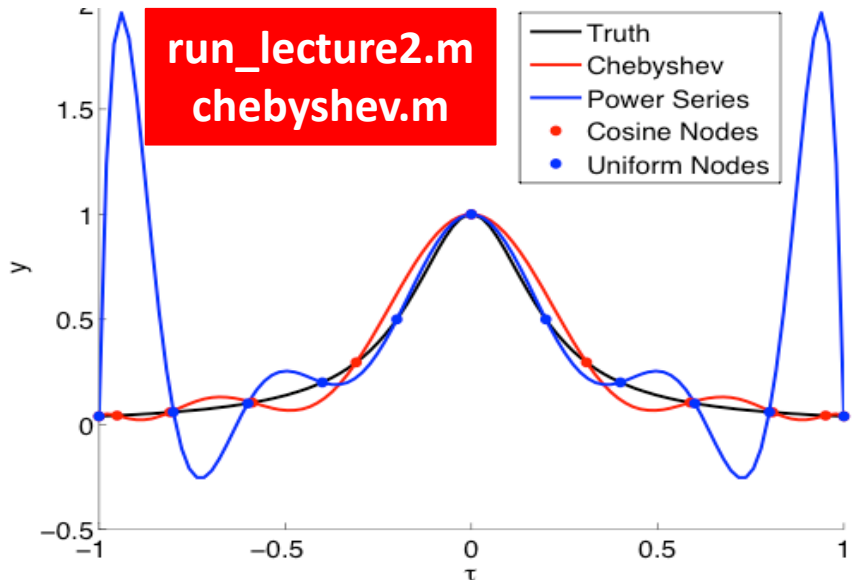
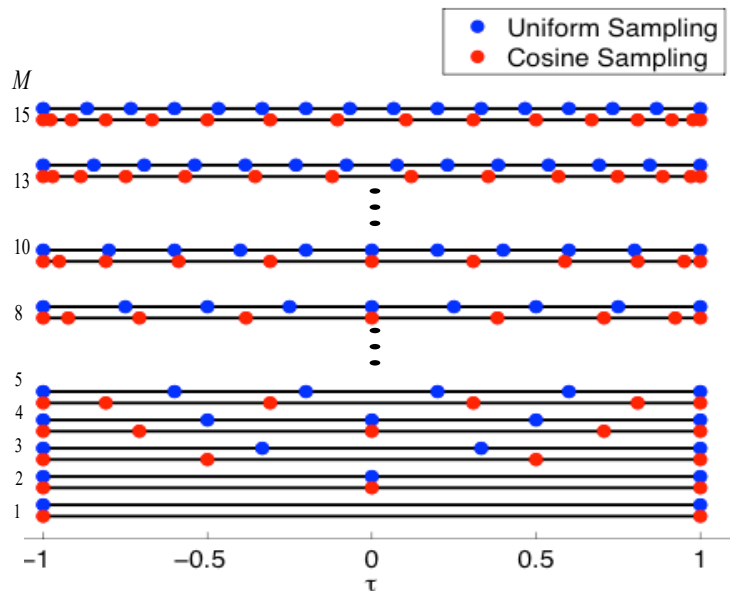
Sample Points

- Sample points (of the forcing function to be fitted) are computed using a cosine node distribution as follows:

$$\xi_j = -\cos(j\pi/M), \quad j = 0, 1, 2, \dots, M.$$

Runge Effect

- Note **clustering** of cosine nodes near the ± 1 boundaries. **Uniform sampling** leads to the “**Runge effect**” (large oscillatory errors between nodes near the boundaries due to **lack of support** outside the interval). **Denser interior** nodes helps **compensate** for absence of support to the right and left of the approximation interval. Qualitatively, cosine nodes lead to nearly uniform approximation errors.



Chebyshev Polynomials

Definition

- Chebyshev polynomials are defined on the range $T(\tau) \in [-1, 1]$
- Two ways to generate Chebyshev polynomials

1. **Recursively** $T_0(\tau) = 1, \quad T_1(\tau) = \tau,$

$$T_{k+1}(\tau) = 2\tau T_k(\tau) - T_{k-1}(\tau). \quad \tau \in [-1, 1]$$

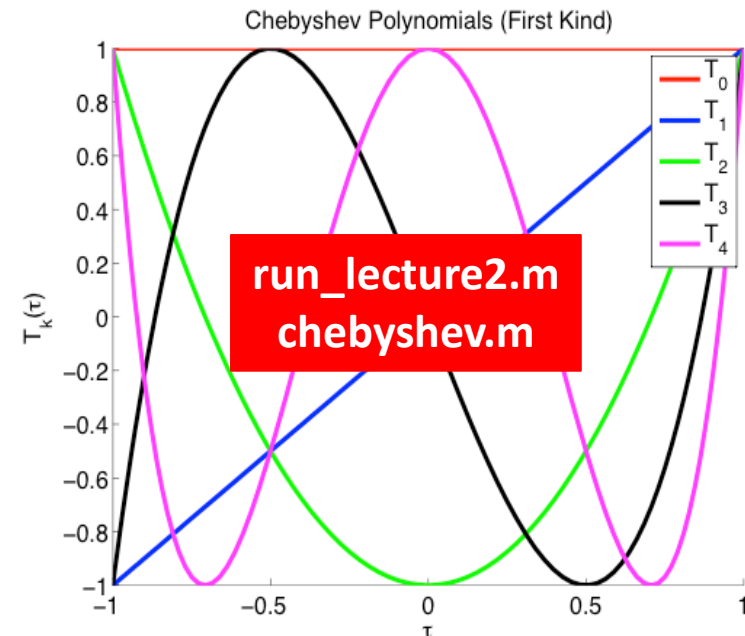
2. **Trigonometrically**

$$T_k(\tau) = \cos(k \arccos(\tau)). \quad \tau \in [-1, 1]$$



Picture Credit: Wikipedia

Pafnuty Chebyshev
(1821-1894)



Independent Variable Transformation

- The independent variable (t) is transformed into (τ : $-1 \leq \tau \leq +1$) as follows

$$\frac{df}{d\tau} = \frac{df}{dt} \frac{dt}{d\tau}$$

- One suggestion is: $t = t_0 + (\tau + 1)(t_f - t_0)/2$
 $\tau = -1 + 2(t - t_0)/(t_f - t_0)$

so $\frac{dt}{d\tau} = (t_f - t_0)/2$

WEIGHT MATRIX EXAMPLE

Example for $M = 2$

- Consider the following

$$m_{\alpha\beta} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} = \Phi^T W \Phi,$$

where $\Phi^T W \Phi$ is given by

$$\Phi^T W \Phi = \begin{bmatrix} \phi_0(\xi_0) & \phi_0(\xi_1) & \phi_0(\xi_2) \\ \phi_1(\xi_0) & \phi_1(\xi_1) & \phi_1(\xi_2) \\ \phi_2(\xi_0) & \phi_2(\xi_1) & \phi_2(\xi_2) \end{bmatrix} \begin{bmatrix} W_0 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_2 \end{bmatrix} \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \phi_2(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \phi_2(\xi_1) \\ \phi_0(\xi_2) & \phi_1(\xi_2) & \phi_2(\xi_2) \end{bmatrix}.$$

- The matrix multiplication computation leads to the following:

$$m_{00} = W_0 \phi_0^2(\xi_0) + W_1 \phi_0^2(\xi_1) + W_2 \phi_0^2(\xi_2),$$

$$m_{11} = W_0 \phi_1^2(\xi_0) + W_1 \phi_1^2(\xi_1) + W_2 \phi_1^2(\xi_2),$$

$$m_{22} = W_0 \phi_2^2(\xi_0) + W_1 \phi_2^2(\xi_1) + W_2 \phi_2^2(\xi_2),$$

$$m_{01} = m_{10} = W_0 \phi_0(\xi_0) \phi_1(\xi_0) + W_1 \phi_0(\xi_1) \phi_1(\xi_1) + W_2 \phi_0(\xi_2) \phi_1(\xi_2),$$

$$m_{02} = m_{20} = W_0 \phi_0(\xi_0) \phi_2(\xi_0) + W_1 \phi_0(\xi_1) \phi_2(\xi_1) + W_2 \phi_0(\xi_2) \phi_2(\xi_2),$$

$$m_{12} = m_{21} = W_0 \phi_1(\xi_0) \phi_2(\xi_0) + W_1 \phi_1(\xi_1) \phi_2(\xi_1) + W_2 \phi_1(\xi_2) \phi_2(\xi_2).$$

- If $\{\phi_0(\xi), \phi_1(\xi), \phi_2(\xi)\}$ are an orthogonal set, $m_{01} = m_{10} = 0$, $m_{02} = m_{20} = 0$ and $m_{12} = m_{21} = 0$. All that remains are the diagonal terms of matrix m .

WEIGHT MATRIX

Coefficients

- Recall the Chebyshev fit coefficients: $\mathbf{a} = (\Phi^T W \Phi)^{-1} \Phi^T W \mathbf{f}$

Diagonal Matrix

- If the matrix $\Phi^T W \Phi$ is diagonal then the inverse is simply the reciprocal of the diagonal elements.

$$(\Phi^T W \Phi)^{-1} = \text{diag} \left\{ 1 / (\Phi^T W \Phi)_{ii} \right\} = \text{diag} \left\{ 1 / m_{00} \quad 1 / m_{11} \quad \dots \quad 1 / m_{NN} \right\}$$

Inner Products

- The typical element of $\Phi^T W \Phi$ is a **discrete inner product** denoted $m_{\alpha\beta} = m_{\beta\alpha}$, and invoking the condition that $\Phi^T W \Phi$ be diagonal gives rise to the orthogonality conditions, requiring the typical pair of inner products to obey:

$$m_{\alpha\beta} = m_{\beta\alpha} \equiv \langle \phi_\alpha(\xi), \phi_\beta(\xi) \rangle \equiv \sum_{j=0}^M W_j \phi_\alpha(\xi_j) \phi_\beta(\xi_j) = \begin{cases} 0, & \text{for } \alpha \neq \beta \\ m_{\alpha\alpha} = c_\alpha > 0, & \text{for } \alpha = \beta \end{cases},$$

Note that the orthogonality conditions depend jointly on (i) the particular basis functions, (ii) the node locations, and (iii) the weight matrix.

WHY HALF?

Preserve Orthogonality

- At a fundamental level, once the basis functions and nodes have been chosen, the weights are ***what ever they have to be*** to ensure orthogonality conditions are | satisfied.

Proof

- Chebyshev polynomials may be computed as $\phi_k(\xi_j) = \cos(k \arccos(\xi_j))$, $\xi \in [-1, 1]$
- As part of the ***least squares*** analysis we showed that the ***inner product*** of two Chebyshev basis functions must be computed, i.e. $\phi_\alpha(\xi) = \cos(\alpha x)$ and $\phi_\beta(\xi) = \cos(\beta x)$, where $x_j = \arccos(\xi_j)$.
- Since $\xi_j = -\cos\left(\frac{j\pi}{M}\right)$, $x = \arccos(\xi_j) = \arccos\left(-\cos\left(\frac{j\pi}{M}\right)\right) = \frac{(M\pi - j\pi)}{M}$.

we can make use of the ***products-to-sums cosine trigonometric identity*** leads to the following:

$$\cos(\alpha x) \cos(\beta x) = \frac{1}{2} \left(\cos((\alpha + \beta)x) + \cos((\alpha - \beta)x) \right)$$

- This implies that the typical product of the two basis functions within the summation may be written purely as a sum of cosine terms. We proceed with an example.

EXAMPLE

Example: $M = 3, \alpha = 2, \beta = 3$

- From the previous slide: $\cos(\alpha x)\cos(\beta x) = \frac{1}{2}(\cos((\alpha + \beta)x) + \cos((\alpha - \beta)x))$
- The summation of the inner products, using $M = 3, \alpha = 2, \beta = 3$, results in

$$\sum_{j=0}^M \phi_{\alpha}(\xi_j)\phi_{\beta}(\xi_j) = \sum_{j=0}^M \frac{1}{2}(\cos((\alpha + \beta)x_j) + \cos((\alpha - \beta)x_j)),$$

which becomes

$$\sum_{j=0}^3 \phi_2(\xi_j)\phi_3(\xi_j) = \frac{1}{2}(\cos((2+3)x_0) + \cos((2-3)x_0) + \cos((2+3)x_1) + \cos((2-3)x_1) + \dots \\ \dots + \cos((2+3)x_2) + \cos((2-3)x_2) + \cos((2+3)x_3) + \cos((2-3)x_3)).$$

- More specifically,

$$\sum_{j=0}^3 \phi_2(\xi_j)\phi_3(\xi_j) = \frac{1}{2} \left(\underbrace{\cos\left((2+3)\frac{(3\pi-0\pi)}{3}\right)}_{=0} + \underbrace{\cos\left((2-3)\frac{(3\pi-0\pi)}{3}\right)}_{=0} + \underbrace{\cos\left((2+3)\frac{(3\pi-\pi)}{3}\right)}_{=\frac{10\pi}{3}=\theta_1} + \underbrace{\cos\left((2-3)\frac{(3\pi-\pi)}{3}\right)}_{=-\frac{2\pi}{3}=\theta_2} + \dots \\ \dots + \underbrace{\cos\left((2+3)\frac{(3\pi-2\pi)}{3}\right)}_{=\frac{5\pi}{3}=2\theta_1} + \underbrace{\cos\left((2-3)\frac{(3\pi-2\pi)}{3}\right)}_{=-\frac{\pi}{3}=2\theta_2} + \dots + \underbrace{\cos\left((2+3)\frac{(3\pi-3\pi)}{3}\right)}_{=5\pi=3\theta_1} + \underbrace{\cos\left((2-3)\frac{(3\pi-3\pi)}{3}\right)}_{=-\pi=3\theta_2} \right).$$

- For orthogonality, the inner product of the two basis functions (ϕ_2 and ϕ_3) must be zero. Thus the above equation must equal zero. The purpose of the lower braces will soon become apparent.

EXAMPLE CONTINUED

Example: $M = 3, \alpha = 2, \beta = 3$

- Consider another trigonometric identity that is the sum of cosine terms:

$$\frac{1}{2} + \cos(\theta) + \cos(2\theta) + \dots + \cos((M-1)\theta) + \frac{1}{2}\cos(M\theta) \equiv \frac{1}{2}\sin(M\theta)\cot\left(\frac{1}{2}\theta\right)$$

- It is clear that when $\theta = M\pi$ the RHS become zero.
- This trigonometric identity is similar to the cosine sum that results from the inner product in the last equation on the previous slide, if θ is set equal to $(\alpha + \beta)\frac{\pi}{3}$. For $\theta = (\alpha - \beta)\frac{\pi}{3}$ we have another form of the identity above.
- So in fact the last equation on the previous slide is the sum of twice the above identity, one with $\theta = (\alpha + \beta)\frac{\pi}{3}$ and one with $\theta = (\alpha - \beta)\frac{\pi}{3}$. This is shown below with the θ_1 colored red to aid in identifying the two series.

$$\frac{1}{2} + \frac{1}{2} + \cos\theta_1 + \cos\theta_2 + \cos 2\theta_1 + \cos 2\theta_2 + \frac{1}{2}\cos 3\theta_1 + \frac{1}{2}\cos 3\theta_2 = 0$$

- Independently comparing the red parts of the above equation with the red parts in the last equation on the previous slide, it is clear that the only difference is that the first and last terms are $1/2$ the size of those on the previous slide. Thus a weight matrix of $W = \text{diag}[\frac{1}{2}, 1, 1, \dots, 1, 1, \frac{1}{2}]$ is required to achieve orthogonality.

Chebyshev Coefficients

Since we know the weight matrix for orthogonality, now we obtain the needed inner products:

- Recall the least squares coefficients: $\mathbf{a} = (\Phi^T W \Phi)^{-1} \Phi^T W \mathbf{f}$.
- Substituting in Chebyshev polynomials, utilizing **cosine sample points** and the **correct weight matrix**, satisfies the orthogonality conditions allows the coefficients to be computed as:

$$a_\alpha = \langle T_\alpha(\xi), f(\xi) \rangle \equiv \frac{1}{c_\alpha} \left\{ \sum_{j=0}^M W_j T_\alpha(\xi_j) f(\xi_j) \right\} = \frac{1}{c_\alpha} \left\{ \frac{1}{2} T_\alpha(\xi_0) f(\xi_0) + \dots + T_\alpha(\xi_{M-1}) f(\xi_{M-1}) + \frac{1}{2} T_\alpha(\xi_M) f(\xi_M) \right\},$$

where the denominators c_α are **positive constants**.

$$c_\alpha = \langle T_\alpha(\xi), f(\xi) \rangle \equiv \sum_{j=0}^M W_j T_\alpha^2(\xi_j) = \left\{ \frac{1}{2} T_\alpha^2(\xi_0) + T_\alpha^2(\xi_1) + \dots + T_\alpha^2(\xi_{M-1}) + \frac{1}{2} T_\alpha^2(\xi_M) \right\}, \alpha = 0, 1, \dots, N$$

- It can be verified (next slide) that the **denominator inner products** reduce to simply:

$$c_0 = \langle T_0(\xi), T_0(\xi) \rangle = M$$

$$c_\alpha = \langle T_\alpha(\xi), T_\alpha(\xi) \rangle = M / 2, \quad \alpha = 1, 2, \dots, N-1$$

$$c_N = \langle T_N(\xi), T_N(\xi) \rangle = M, \quad \text{if } M = N \text{ (interpolation case)}$$

$$c_N = \langle T_N(\xi), T_N(\xi) \rangle = M / 2, \quad \text{if } M > N \text{ (least squares case)}$$

- The final coefficients are computed directly from the discrete inner products as:

$$a_0 = \frac{\langle T_0(\xi), f(\xi) \rangle}{\langle T_0(\xi), T_0(\xi) \rangle} = \frac{1}{M} \left\{ \frac{1}{2} T_0(\xi_0) f(\xi_0) + \dots + T_0(\xi_{M-1}) f(\xi_{M-1}) + \frac{1}{2} T_0(\xi_M) f(\xi_M) \right\}$$

$$a_\alpha = \frac{\langle T_\alpha(\xi), f(\xi) \rangle}{\langle T_\alpha(\xi), T_\alpha(\xi) \rangle} = \frac{2}{M} \left\{ \frac{1}{2} T_\alpha(\xi_0) f(\xi_0) + \dots + T_\alpha(\xi_{M-1}) f(\xi_{M-1}) + \frac{1}{2} T_\alpha(\xi_M) f(\xi_M) \right\}, \quad \alpha = 1, 2, \dots, N-1$$

$$a_N = \frac{\langle T_N(\xi), f(\xi) \rangle}{\langle T_N(\xi), T_N(\xi) \rangle} = \frac{1}{c_N} \left\{ \frac{1}{2} T_N(\xi_0) f(\xi_0) + \dots + T_N(\xi_{M-1}) f(\xi_{M-1}) + \frac{1}{2} T_N(\xi_M) f(\xi_M) \right\}, \quad \begin{cases} c_N = M, & M = N \\ c_N = \frac{M}{2}, & M > N \end{cases}$$

VERIFICATION OF c_0, c_α, c_N

$$c_0 = \langle T_0(\xi), T_0(\xi) \rangle \quad (M > N, N = 0)$$

- Consider the case for $N = 0, M = 4$. Performing the inner product and including the correct weight matrix leads to

$$c_0 = \frac{1}{2}T_0^2(\xi_0) + T_0^2(\xi_1) + T_0^2(\xi_2) + T_0^2(\xi_3) + \frac{1}{2}T_0^2(\xi_4).$$

- The zeroth order Chebyshev polynomial is $T_0 = 1$, which leads to $c_0 = \left\{ \frac{1}{2} + 1 + 1 + 1 + \frac{1}{2} = 4 \right\} = M$.

$$c_\alpha = c_1 = \langle T_1(\xi), T_1(\xi) \rangle \quad (M > N)$$

- Consider the case for $N = 1, M = 4$. Carrying out the inner product as above gives

$$c_1 = \frac{1}{2}T_1^2(\xi_0) + T_1^2(\xi_1) + T_1^2(\xi_2) + T_1^2(\xi_3) + \frac{1}{2}T_1^2(\xi_4).$$

- The cosine sample points are calculated as follows: $\xi_0 = -\cos\left(\frac{0\pi}{4}\right)$, $\xi_1 = -\cos\left(\frac{\pi}{4}\right)$, $\xi_2 = -\cos\left(\frac{2\pi}{4}\right)$, $\xi_3 = -\cos\left(\frac{3\pi}{4}\right)$, $\xi_4 = -\cos\left(\frac{4\pi}{4}\right)$. Substituting these into the first Chebyshev polynomial, $T_1 = \xi$, results in

$$c_\alpha = c_1 = \left\{ \frac{1}{2} + \frac{1}{2} + 0 + \frac{1}{2} + \frac{1}{2} = 2 \right\} = \frac{M}{2}.$$

$$c_\alpha = c_2 = \langle T_2(\xi), T_2(\xi) \rangle \quad (M > N)$$

- Similarly for $N = 2, M = 4$, and the second Chebyshev polynomial, $T_2 = 2\xi^2 - 1$, with the above sample points, we can quickly verify

$$c_\alpha = c_2 = \left\{ \frac{1}{2} + 0 + 1 + 0 + \frac{1}{2} = 2 \right\} = \frac{M}{2}.$$

VERIFICATION OF c_0, c_α, c_N CONTINUED

$$c_\alpha = c_3 = \langle T_3(\xi), T_3(\xi) \rangle \quad (M > N)$$

- Similarly for $N = 3, M = 4$, and the third Chebyshev polynomial, $T_3 = 4\xi^3 - 3\xi$, with the sample points shown on the previous slide, we have

$$c_\alpha = c_3 = \left\{ \frac{1}{2} + \frac{1}{2} + 0 + \frac{1}{2} + \frac{1}{2} = 2 \right\} = \frac{M}{2}.$$

$$c_N = c_4 = \langle T_4(\xi), T_4(\xi) \rangle \quad (M = N)$$

- Similarly for $N = 4, M = 4$, and the fourth Chebyshev polynomial, $T_4 = 8\xi^4 - 8\xi^2$, with the sample points shown on the previous slide, we have

$$c_N = c_4 = \left\{ \frac{1}{2} + 1 + 1 + 1 + \frac{1}{2} = 4 \right\} = M.$$

$$c_N = c_4 = \langle T_4(\xi), T_4(\xi) \rangle \quad (M > N)$$

- For $N = 4, M = 5$, the sample points are given by:

$$\xi_0 = -\cos\left(\frac{0\pi}{5}\right), \xi_1 = -\cos\left(\frac{\pi}{4}\right), \xi_2 = -\cos\left(\frac{2\pi}{4}\right), \xi_3 = -\cos\left(\frac{3\pi}{4}\right), \xi_4 = -\cos\left(\frac{4\pi}{4}\right), \xi_5 = -\cos\left(\frac{5\pi}{4}\right).$$

- Substituting these sample points into the fourth Chebyshev polynomial gives

$$c_N = c_4 \approx \left\{ \frac{1}{2} + 0.6545 + 0.0995 + 0.6545 + \frac{1}{2} = \frac{5}{2} \right\} = \frac{M}{2}.$$

VECTOR MATRIX NOTATION

Coefficients

- The least squares solution may be expressed in vector matrix form:

where $\boxed{\mathbf{a} = (T^T W T)^{-1} T^T W \mathbf{f} = V T^T W \mathbf{f} = A \mathbf{f}}$, $(T^T W T)^{-1} = V = \text{diag} \left\{ \frac{1}{c_0}, \frac{1}{c_\alpha}, \frac{1}{c_\alpha}, \dots, \frac{1}{c_\alpha}, \frac{1}{c_\alpha}, \frac{1}{c_N} \right\}$.

- The least squares operator (A) is simply given by

$$A = \begin{matrix} & \begin{matrix} M=N \end{matrix} & \\ \begin{bmatrix} \frac{1}{2} \frac{1}{M} T_0(\xi_0) & \frac{1}{M} T_0(\xi_1) & \cdots & \frac{1}{M} T_0(\xi_{M-1}) & \frac{1}{2} \frac{1}{M} T_0(\xi_M) \\ \frac{1}{2} \frac{2}{M} T_1(\xi_0) & \frac{2}{M} T_1(\xi_1) & \cdots & \frac{2}{M} T_1(\xi_{M-1}) & \frac{1}{2} \frac{2}{M} T_1(\xi_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2} \frac{2}{M} T_{N-1}(\xi_0) & \frac{2}{M} T_{N-1}(\xi_1) & \cdots & \frac{2}{M} T_{N-1}(\xi_{M-1}) & \frac{1}{2} \frac{2}{M} T_{N-1}(\xi_M) \\ \frac{1}{2} \frac{1}{M} T_N(\xi_0) & \frac{1}{M} T_N(\xi_1) & \cdots & \frac{1}{M} T_N(\xi_{M-1}) & \frac{1}{2} \frac{1}{M} T_N(\xi_M) \end{bmatrix} & \begin{matrix} M>N \end{matrix} & \\ \begin{bmatrix} \frac{1}{2} \frac{1}{M} T_0(\xi_0) & \frac{1}{M} T_0(\xi_1) & \cdots & \frac{1}{M} T_0(\xi_{M-1}) & \frac{1}{2} \frac{1}{M} T_0(\xi_M) \\ \frac{1}{2} \frac{2}{M} T_1(\xi_0) & \frac{2}{M} T_1(\xi_1) & \cdots & \frac{2}{M} T_1(\xi_{M-1}) & \frac{1}{2} \frac{2}{M} T_1(\xi_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2} \frac{2}{M} T_{N-1}(\xi_0) & \frac{2}{M} T_{N-1}(\xi_1) & \cdots & \frac{2}{M} T_{N-1}(\xi_{M-1}) & \frac{1}{2} \frac{2}{M} T_{N-1}(\xi_M) \\ \frac{1}{2} \frac{2}{M} T_N(\xi_0) & \frac{2}{M} T_N(\xi_1) & \cdots & \frac{2}{M} T_N(\xi_{M-1}) & \frac{1}{2} \frac{2}{M} T_N(\xi_M) \end{bmatrix} \end{matrix}$$

In general, matrix A is an $[N+1] \times [M+1]$ matrix. The number of **rows** is determined by the **degree** of the polynomial and the number of **columns** is determined by the number of **sample points**.

FUNCTION APPROXIMATION

Return to Function Approximation:

- Using summation notation the function is approximated in terms of the matrices T , V , W and f as follows:

$$f(\xi) \approx \sum_{\alpha=0}^N a_{\alpha} \phi_{\alpha}(\xi) = \sum_{\alpha=0}^N \left\{ \underbrace{\sum_{j=0}^M V_{\alpha j} W_{\alpha j} T_{\alpha}(\xi_j) f(\xi_j)}_{\text{Least squares fit to discrete acceleration samples to obtain coefficients for acceleration}} \right\} T_{\alpha}(\xi).$$

Least squares fit to discrete acceleration samples to obtain coefficients for acceleration

Function approximation

Chebyshev basis functions

We have now approximated the function. We can show via examples and arguments based on completeness, that almost any smooth function can be approximated, in an at worst a piecewise continuous fashion, with near machine precision ...

APPROXIMATION EXAMPLE

Perturbed Two-body Acceleration

- Given orbit initial conditions:

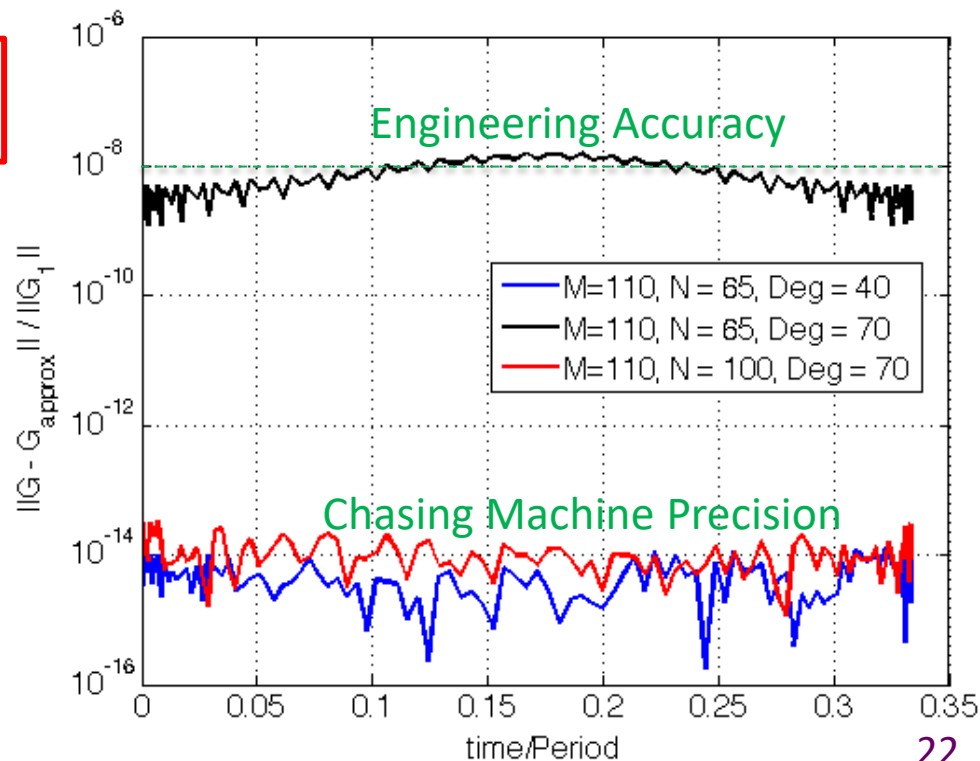
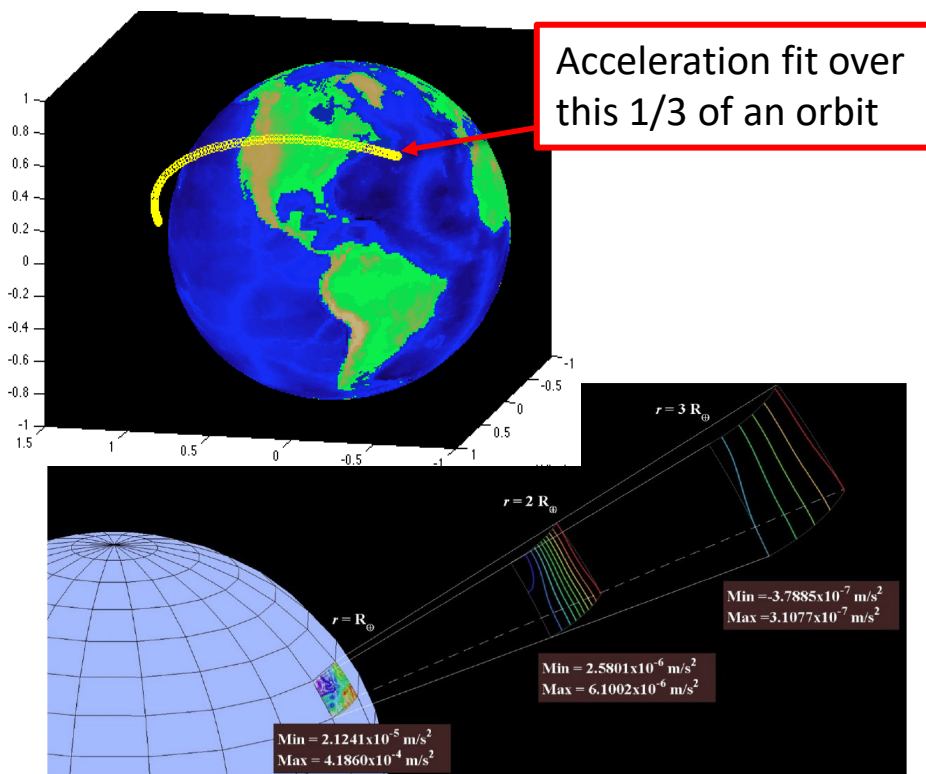
$$\mathbf{r}_0 = \{7000 \quad 0 \quad 0\}^T \text{ km}$$

$$\mathbf{v}_0 = \{ \quad 0 \quad 5.335 \quad 5.335\}^T \text{ km/s.}$$

- Fit perturbed acceleration along an ideal elliptical orbit, using the spherical harmonic gravity model for degree = 40 and 70. Note that more nodes are required to get a machine precision fit for the higher degree gravity model.

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3} \mathbf{r} + \underbrace{\mathbf{a}_d(\mathbf{r}, \mathbf{v}, t)}_{\text{Gravity}}, \quad \mathbf{v} = \dot{\mathbf{r}}$$

Gravity is modeled by infinite spherical harmonic series as fct (\mathbf{r}, t) .





Lecture 2

NUMERICAL QUADRATURE

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

NEWTON-COTES AND GAUSSIAN QUADRATURE

- **Newton-Cotes Formula** for approximating Integral has the form of a weighted average:

$$\mathcal{J}(f(x), a, b, c_1, c_2, \dots, c_M, x_1, x_2, \dots, x_M) \cong \int_a^b f(x) dx \cong \sum_{i=1}^M c_i f(x_i)$$

- In the classical Newton-Cotes formulation, the nodes are uniformly spaced, and this rules out taking advantage of the truth that higher accuracy can be obtained via judicious use of non-uniform nodes.
 - Note, there are $2M$ unknowns $(c_1, c_2, \dots, c_M, x_1, x_2, \dots, x_M)$, and the resulting equations for a given $f(x)$ contain the (x_1, x_2, \dots, x_M) non linearly, so in general, we may anticipate solving a set of $2M$ coupled nonlinear algebraic equations.
- **Gaussian Quadrature** ... is a special case of the Newton-Cotes approach wherein, we select judicious nodes (“quadrature points”) chosen such that a polynomial of a given degree is integrated exactly by the Newton-Cotes formula. This will lead to non-uniform nodes for each degree polynomial.
 - Other intervals than $[a, b]$ can be used, and we can subdivide the given interval into many smaller intervals in order to improve accuracy, as needed.
 - Usually, the theoretical developments are done on the interval $[-1, 1]$, and of course, we can always transform $[a, b]$ onto $[-1, 1]$ with a linear variable change.

GAUSSIAN QUADRATURE

Consider the Newton-Cotes formula

$$\mathcal{J}(f(x), -1, 1, c_1, c_2, \dots, c_M, x_1, x_2, \dots, x_M) \cong \int_{-1}^1 f(x) dx \cong \sum_{i=1}^M c_i f(x_i)$$

Let us set $M=2$, and seek to determine the four unknowns (c_1, c_2, x_1, x_2) such that the Newton-Cotes formula is exact for the four polynomials:

$$(1) f(x) = 1; \quad (2) f(x) = x; \quad (3) f(x) = x^2; \quad (4) f(x) = x^3$$

Note that these four requirements will give us four equations in four unknowns:

$$(1) f(x) = 1 \Rightarrow \int_{-1}^1 1 dx = 2 \quad \Rightarrow \quad c_1 + c_2 = 2$$

$$(2) f(x) = x \Rightarrow \int_{-1}^1 x dx = 0 \quad \Rightarrow \quad c_1 x_1 + c_2 x_2 = 0$$

$$(3) f(x) = x^2 \Rightarrow \int_{-1}^1 x^2 dx = \frac{2}{3} \quad \Rightarrow \quad c_1 x_1^2 + c_2 x_2^2 = \frac{2}{3}$$

$$(4) f(x) = x^3 \Rightarrow \int_{-1}^1 x^3 dx = 0 \quad \Rightarrow \quad c_1 x_1^3 + c_2 x_2^3 = 0$$

These equations are satisfied by $c_1 = 1, c_2 = 1, x_1 = -\frac{1}{\sqrt{3}}, x_2 = \frac{1}{\sqrt{3}}$, so this **Gaussian**

Quadrature formula exactly integrates a cubic polynomial over the interval $[-1, 1]$:

$$\Rightarrow \mathcal{J}(f(x)) = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

Likewise a Gaussian Quadrature formula that exactly integrates

a quartic polynomial over the interval $[-1, 1]$ can be shown to be:

$$\Rightarrow \mathcal{J}(f(x)) = \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right)$$

PATH APPROXIMATION VS NUMERICAL QUADRATURE

In a significant family of problems, we have a differential equation of the form

$$\dot{x} = \frac{dx}{dt} = f(t), \quad x(t_0) = a$$

Obviously, for an integrable $f(t)$, the differential equation is fully equivalent to the integral equation

$$x(t) = a + \int_a^t f(\tau) d\tau$$

Thus we see that solving a differential equation where $f(t)$ is a general function (or an accurate approximation of a given single valued general function) is closely related to the foregoing developments on quadrature. However, it is significant that we may have a variable upper limit, rather than a fixed limit, and that instead of just a **numerical value to approximate the integral over fixed limits**, we may desire an analytically attractive approximation of the **trajectory** $x(t)$ expressed in terms of “nice” algebraic functions.

Obtaining not just a numerical value of the integral (i.e., numerical quadrature), but rather an algebraic expression that accurately approximates the integral for all infinity of t values in the interval of interest is central to the path approximation methods we discuss below. Obviously, specific t values can be substituted and a path approximation formula will give numerical quadrature result over any interior subinterval as a special case.

PATH APPROXIMATION VIA CHEBYSHEV POLYNOMIAL QUADRATURE

Consider the problem posed on the previous page with the interval $[-1 \leq t \leq 1]$, we have a differential equation of the form

$$\dot{x} = f(t), \quad x(-1) = x_0 \quad \Rightarrow \quad x(t) = x_0 + \int_{-1}^t f(\tau) d\tau$$

Suppose $f(t)$ is smooth and single-valued, but is a messy (“ugly”) function not found in your standard integral tables (or is not integrable by your favorite symbol manipulator).

We can approximate the integrand in an infinity of ways as a linear combination of basis functions that are easily integrated. If we do least square approximation to determine the coefficients, of the basis functions we in general must invert a matrix that may be high dimensioned if we desire high accuracy. To avoid this dilemma, we can utilize any set of orthogonal functions. One attractive set are the Chebyshev orthogonal polynomials.

So we write the approximation of the integrand and integrate term-by-term to get

$$f(\tau) = \sum_{k=0}^{N-1} a_k T_k(\tau) \quad \Rightarrow \quad x(t) = x_0 + \int_{-1}^t f(\tau) d\tau = x_0 + \sum_{k=0}^{N-1} a_k \int_{-1}^t T_k(\tau) d\tau$$

Now, we know from Lecture 1 that

$$\mathbf{a} = \begin{Bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{Bmatrix} = \begin{bmatrix} N \times M \\ \text{Least Square} \\ \text{Operator Matrix} \end{bmatrix} \begin{Bmatrix} f(t_0) \\ f(t_1) \\ \vdots \\ f(t_M) \end{Bmatrix}, \text{ and } \int_{-1}^t T_k(\tau) d\tau = \frac{1}{2} \left(\frac{T_{k+1}(\tau)}{k+1} - \frac{T_{k-1}(\tau)}{k-1} \right) \Big|_{-1}^t, \quad k > 1; \begin{cases} \int_{-1}^t T_0(\tau) d\tau = [T_1(\tau)]_{-1}^t \\ \int_{-1}^t T_1(\tau) d\tau = \frac{1}{4} [T_2(\tau) + 1]_{-1}^t \end{cases}$$

PATH APPROXIMATION VIA CHEBYSHEV POLYNOMIAL QUADRATURE

As developed in Lecture 1, the resulting series for $x(t)$ can also be written as a Chebyshev polynomial

$$x(t) = x_0 + \int_{-1}^t f(\tau) d\tau \cong x_0 + \sum_{k=0}^{N-1} a_k \int_{-1}^t T_k(\tau) d\tau \equiv \sum_{k=0}^N \beta_k T_k(\tau)$$

The coefficient vector $\beta(t)$ can be compactly expressed in matrix notation as

$$\beta = X_0 + [P_1][A]f,$$

where

$$X_0^T = [x_0 \quad 0 \quad \dots \quad 0],$$

$$a = (T^T W T)^{-1} T^T W f = V T^T W f = A f \quad [S] =$$

$$A \equiv V T^T W$$

$$[L] = \begin{bmatrix} T_0 & T_1(-1) & \dots & T_N(-1) \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Indefinite Integral Operator:

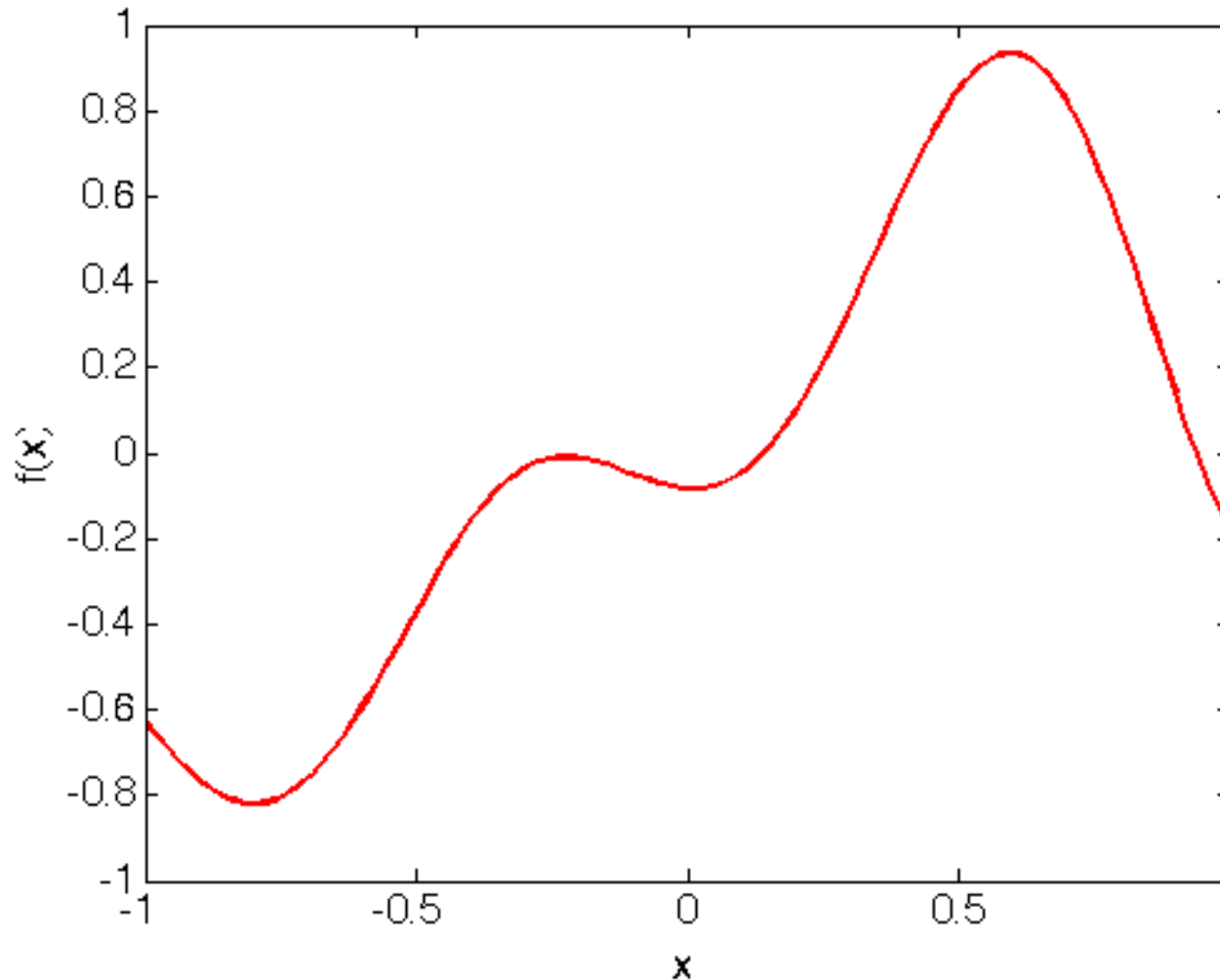
$$[S] = \begin{bmatrix} 1/4 & 0 & \dots & 0 \\ 1 & 0 & -1/2 & 0 \\ 0 & 1/4 & 0 & -1/4 & 0 & \vdots & \vdots \\ & 0 & 1/2k & 0 & -1/2k & 0 & \\ & & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \frac{1}{2(N-2)} & 0 & \frac{-1}{2(N-2)} \\ & & & & 0 & \frac{1}{2(N-1)} & 0 \\ 0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

Definite Integral Operator:

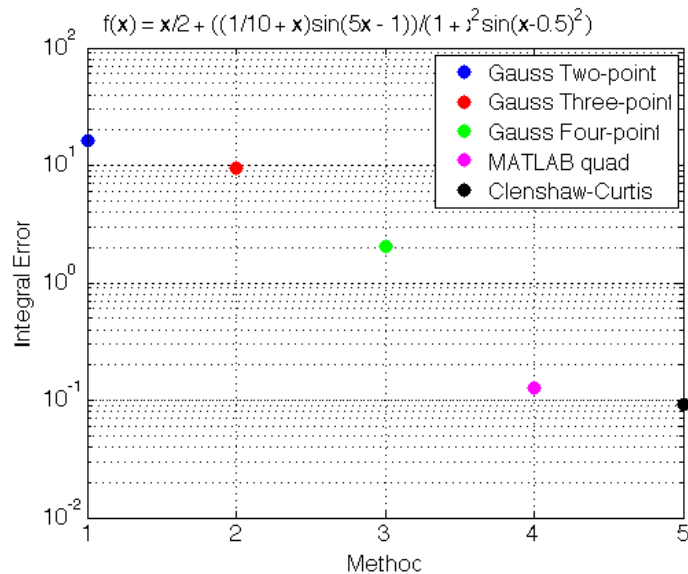
$$[P_1] = [[I] - [L]][S]$$

THE “UGLY” TEST FUNCTION 1

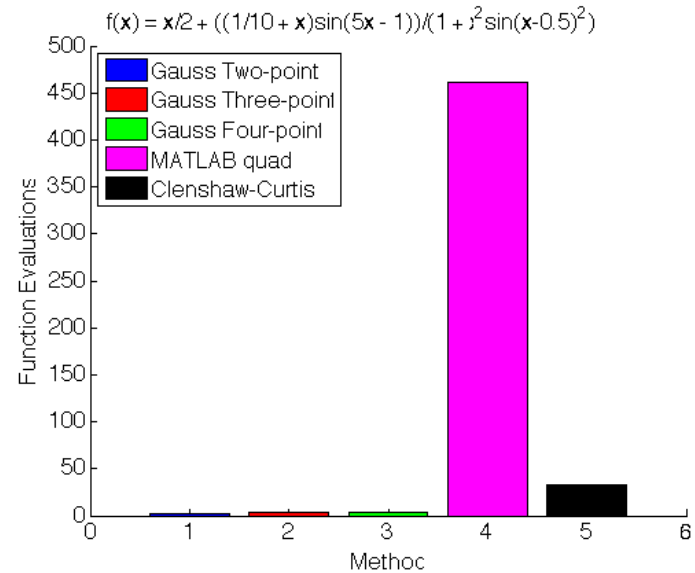
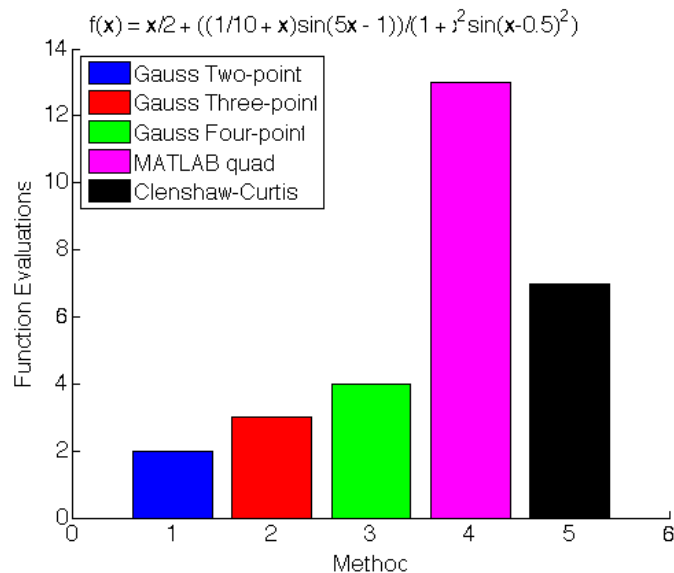
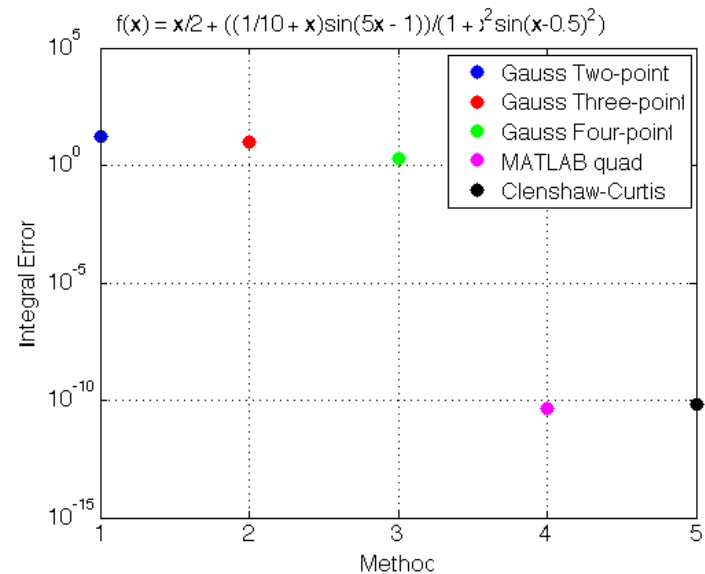
$$f(x) = \frac{x/2 + ((1/10 + x)\sin(5x + 1))}{(1 + x^2)\sin^2(x - 1/2)}$$



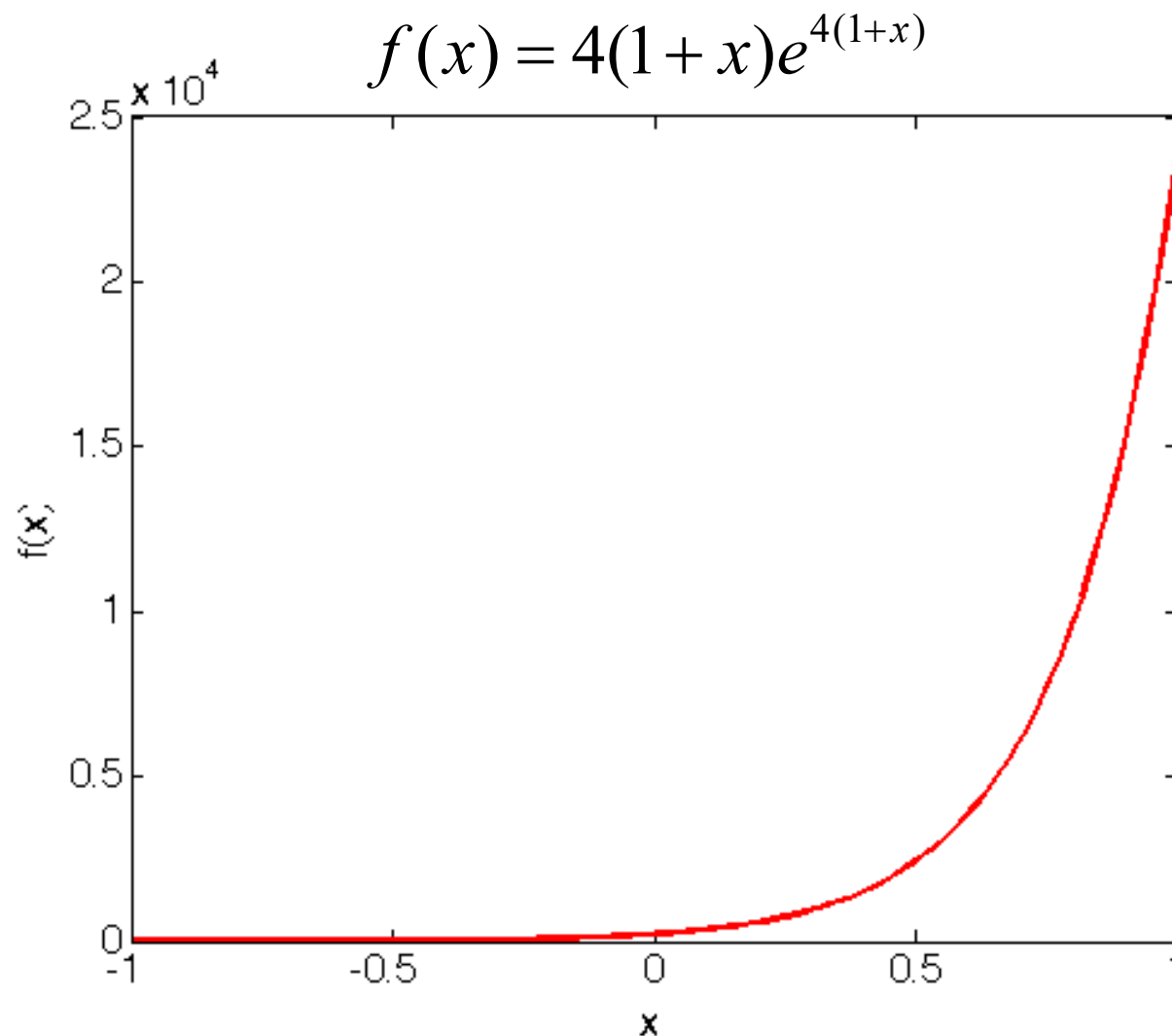
Low Accuracy



High Accuracy



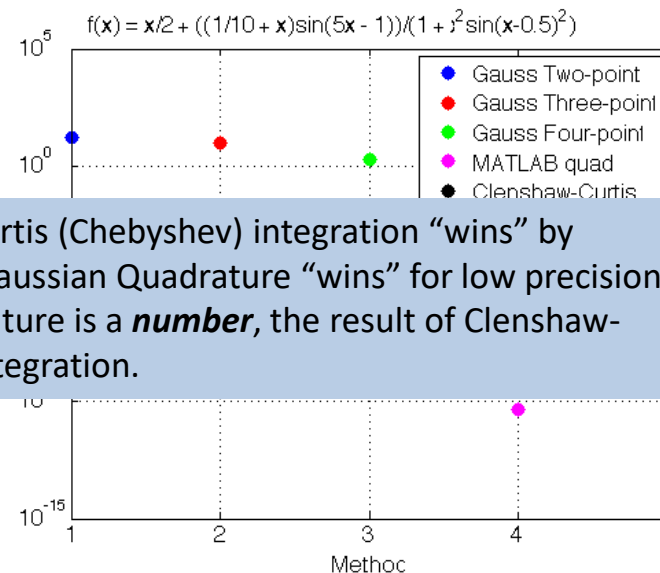
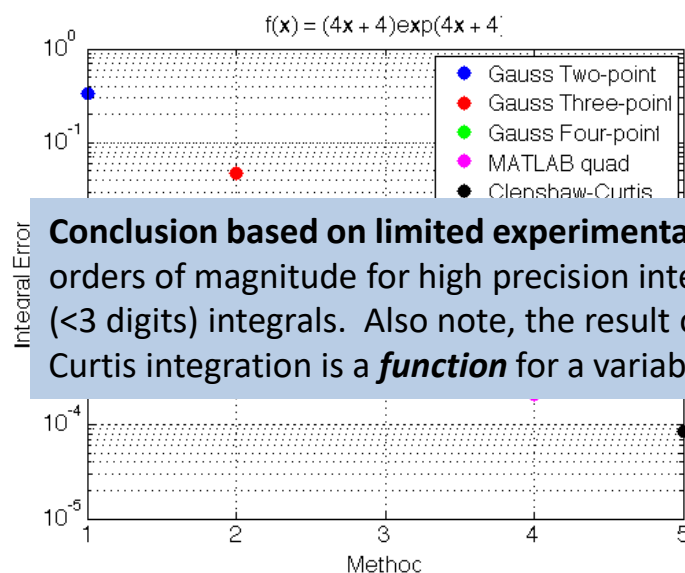
QUADRATURE COMPARISON 2: EASIER FUNCTION



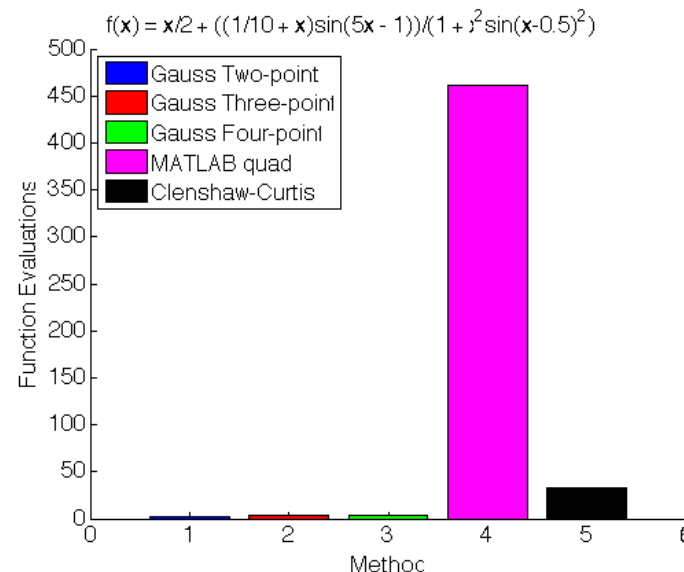
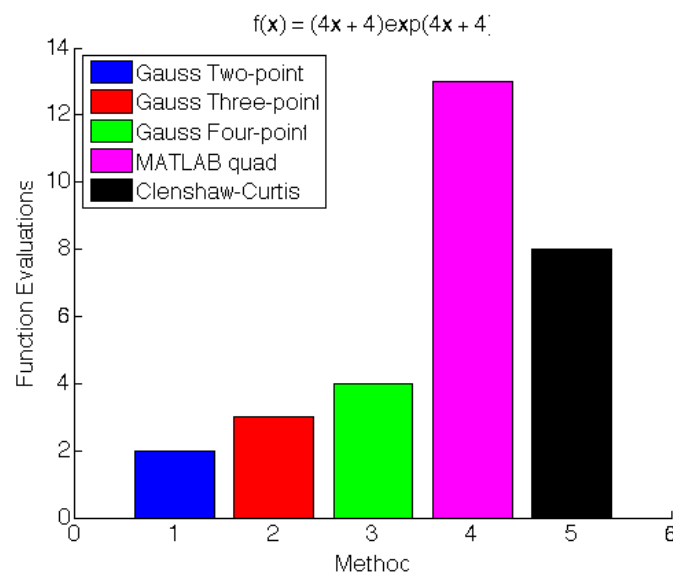
QUADRATURE COMPARISON 2 – LOW AND HIGH ACCURACY

Low Accuracy

High Accuracy



Conclusion based on limited experimentation: Clenshaw-Curtis (Chebyshev) integration “wins” by orders of magnitude for high precision integrals (>6 digits), Gaussian Quadrature “wins” for low precision (<3 digits) integrals. Also note, the result of Gaussian Quadrature is a **number**, the result of Clenshaw-Curtis integration is a **function** for a variable upper limit of integration.



REFERENCES

1. Boyd, John P., *Chebyshev and Fourier spectral methods*, Courier Corporation, 2001.
2. Hildebrand, Francis Begnaud, *Introduction to numerical analysis*, Courier Corporation, 1987.
3. Ma, J., Rokhlin, V., and Wandzura, S., *Generalized Gaussian quadrature rules for systems of arbitrary functions*, SIAM Journal on Numerical Analysis 33.3 (1996): 971-996.
4. Piessens, Robert, *Modified Clenshaw-Curtis integration and applications to numerical computation of integral transforms*, Numerical integration, Springer Netherlands, 1987: 35-51.
5. Junkins, J., Bani Younes, A., Woollands, R., Bai, B. *Orthogonal Approximation in Higher Dimensions: Applications in Astrodynamics*, ASS 12-634, JN Juang Astrodynamics Symp, 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.



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

INTRODUCTION

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:

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

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

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

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

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

Picard Convergence Theorem

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



Picture Credit: Wikipedia

Charles Emile Picard
(1856-1941)

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

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

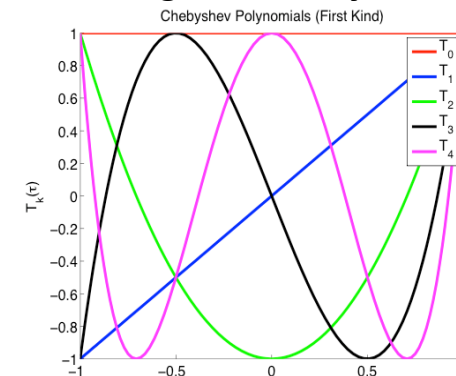
Chebyshev polynomials are used to **approximate the integrate** in the Picard iteration sequence.



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

$$\mathbf{x}^i(t) = \mathbf{x}(t_0) + \int_{t_0}^t \underbrace{\mathbf{f}(s, \mathbf{x}^{i-1}(s))}_{\text{(Lecture 1)}} ds, \quad i = 1, 2, \dots$$

$$\mathbf{x}^i(t) = \mathbf{x}(-1) + \int_{-1}^{\tau} \underbrace{\sum_{k=0}^{N-1} \mathbf{a}_k^{i-1} T_k(s)}_{\text{Approximation with Chebyshev polynomials}} ds, \quad i = 1, 2, \dots$$

Recall Least Squares Approximation

• Scalar system:

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

where $\mathbf{a}^{i-1} = \begin{bmatrix} a_0^{i-1} \\ a_1^{i-1} \\ \vdots \\ a_{N-1}^{i-1} \end{bmatrix}$, $\mathbf{f}^{i-1} = \begin{bmatrix} f(\tau_0, x^{i-1}(\tau_0)) \\ f(\tau_1, x^{i-1}(\tau_1)) \\ \vdots \\ f(\tau_M, x^{i-1}(\tau_M)) \end{bmatrix}$, $[\Phi] = [T] = \begin{bmatrix} T_0(\tau_0) & \cdots & T_{N-1}(\tau_0) \\ T_0(\tau_1) & \cdots & T_{N-1}(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_{N-1}(\tau_M) \end{bmatrix}$ 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)

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

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

FIRST ORDER METHOD

From the previous slide...

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

Expand

- Expanding the Chebyshev series above leads to:

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

Integrate

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

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

- Substituting the above leads to

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

FIRST ORDER METHOD

From the previous slide...

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

Matrix Form

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

Solution \rightarrow $\mathbf{x}^i(t) = \mathbf{x}(-1) + \begin{matrix} 1 \times n & 1 \times n & 1 \times (N+1) \end{matrix} [T_0(s) \quad \dots \quad T_N(s)] \Big|_{-1}^{\tau}$

Initial condition vector \uparrow

$$\begin{bmatrix} \frac{1}{4} \mathbf{a}_0^{i-1} \\ \frac{1}{2} (2\mathbf{a}_0^{i-1} - \mathbf{a}_2^{i-1}) \\ \frac{1}{4} (\mathbf{a}_1^{i-1} - \mathbf{a}_3^{i-1}) \\ \vdots \\ \frac{1}{2k} (\mathbf{a}_{k-1}^{i-1} - \mathbf{a}_{k+1}^{i-1}) \\ \vdots \\ \frac{1}{2(N-1)} \mathbf{a}_{N-2}^{i-1} \\ \frac{1}{2N} \mathbf{a}_{N-1}^{i-1} \end{bmatrix} \cdot$$

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

FIRST ORDER METHOD

Matrix of Least Squares Coefficients

$(N+1) \times n$

$(N+1) \times N$

$$\begin{bmatrix} \frac{1}{4}a_0^{i-1} \\ \frac{1}{2}(2a_0^{i-1} - a_2^{i-1}) \\ \frac{1}{4}(a_1^{i-1} - a_3^{i-1}) \\ \vdots \\ \frac{1}{2k}(a_{k-1}^{i-1} - a_{k+1}^{i-1}) \\ \vdots \\ \frac{1}{2(N-1)}a_{N-2}^{i-1} \\ \frac{1}{2N}a_{N-1}^{i-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{4} & 0 & \cdots & 0 \\ 1 & 0 & -1/2 & 0 \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \frac{1}{2(N-2)} & 0 & \frac{-1}{2(N-1)} \\ 0 & \vdots & \vdots & 0 & \frac{1}{2(N-1)} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \frac{1}{2N} \end{bmatrix}}_{[S]}$$

$$[G^{i-1}] = \begin{Bmatrix} a_0^{i-1} \\ a_1^{i-1} \\ \vdots \\ a_{N-1}^{i-1} \end{Bmatrix} = \underbrace{(\Phi^T W \Phi)^{-1} \Phi^T W}_{\equiv A} [F^{i-1}] = A[F^{i-1}], \quad [F^{i-1}] = \begin{bmatrix} f_1(\tau_0, \mathbf{x}^{i-1}(\tau_0)) & \cdots & f_n(\tau_0, \mathbf{x}^{i-1}(\tau_0)) \\ f_1(\tau_1, \mathbf{x}^{i-1}(\tau_1)) & \cdots & f_n(\tau_1, \mathbf{x}^{i-1}(\tau_1)) \\ \vdots & \ddots & \vdots \\ f_1(\tau_M, \mathbf{x}^{i-1}(\tau_M)) & \cdots & f_n(\tau_M, \mathbf{x}^{i-1}(\tau_M)) \end{bmatrix}$$

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

$t \rightarrow \tau$ 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**

FIRST ORDER METHOD

Matrix Form

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

$$\mathbf{x}^i(t) = \mathbf{x}(-1) + \left[\overset{1 \times n}{T_0(s)} \cdots \overset{1 \times n}{T_N(s)} \right]_{-1}^t \overset{1 \times (N+1)}$$

$$\begin{bmatrix} \overset{(N+1) \times n}{\frac{1}{4} a_0^{i-1}} \\ \frac{1}{2} (2a_0^{i-1} - a_2^{i-1}) \\ \frac{1}{4} (a_1^{i-1} - a_3^{i-1}) \\ \vdots \\ \frac{1}{2k} (a_{k-1}^{i-1} - a_{k+1}^{i-1}) \\ \vdots \\ \frac{1}{2(N-1)} a_{N-2}^{i-1} \\ \frac{1}{2N} a_{N-1}^{i-1} \end{bmatrix}$$

**LEAST SQUARES
OPERATOR**

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

FIRST ORDER METHOD

Chebyshev Series for L.H.S

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

Matrix Form

Diagram illustrating the matrix form of the Chebyshev series for the L.H.S, with dimensions and annotations:

$$\mathbf{x}^i(t) = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} \begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \mathbf{x}(-1) + \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \end{bmatrix} \begin{bmatrix} S \\ A \\ F^{i-1} \end{bmatrix} W_2$$

Annotations and Dimensions:

- $\mathbf{x}^i(t)$: $1 \times n$ (purple)
- $\begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix}$: $1 \times (N+1)$ (orange)
- $\begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix}$: $(N+1) \times n$ (cyan)
- $\mathbf{x}(-1)$: $1 \times n$ (green circle)
- $\begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix}$: $1 \times (N+1)$ (orange bracket)
- $\begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \end{bmatrix}$: $(N+1) \times N$ (green circle)
- $\begin{bmatrix} S \\ A \\ F^{i-1} \end{bmatrix}$: $N \times (M+1)$ (red arrow)
- W_2 : $(M+1) \times n$ (purple)

Solution coefficients (cyan text)

Constant terms (green text)

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

$$\mathbf{x}^i(t) = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} \begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = C(-1) + \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} \begin{bmatrix} S \\ A \\ F^{i-1} \end{bmatrix} W_2$$

FIRST ORDER METHOD

Absorb Constant

$$\mathbf{x}^i(t) = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} \begin{bmatrix} \beta_0^i - C(-1) \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} [S][A][F^{i-1}][W_2].$$

Equate coefficients of T_k

$$\begin{bmatrix} \beta_0^i - C(-1) \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = [S][A][F^{i-1}][W_2] \xrightarrow{W_2} \begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \begin{Bmatrix} C(-1) \\ 0 \\ \vdots \\ 0 \end{Bmatrix} + [S][A][F^{i-1}][W_2].$$

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

$$\begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1(-1) & \cdots & \mathbf{x}_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} [S][A][F^{i-1}][W_2] + [S][A][F^{i-1}][W_2].$$

FIRST ORDER METHOD

From the previous slide...

$$\begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \begin{Bmatrix} \mathbf{x}(-1) \\ 0 \\ \vdots \\ 0 \end{Bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} [S][A][F^{i-1}][W_2] + [S][A][F^{i-1}][W_2].$$

Final Expression for Coefficients

$$\begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \mathbf{X}_0 + [[I] - [L]][S][A][F^{i-1}][W_2] = \mathbf{X}_0 + [P_1][A][F^{i-1}][W_2],$$

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

FIRST ORDER METHOD

Recall the summation equation

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

Expression for Coefficients

Diagram illustrating the matrix representation of the Picard-Chebyshev method:

The coefficient vector $\begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix}$ (size $(N+1) \times n$) is related to the initial condition \mathbf{X}_0 (size $N+1 \times n$) and the integral term via the equation:

$$\begin{bmatrix} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-1}^i \\ \beta_N^i \end{bmatrix} = \mathbf{X}_0 + \underbrace{[I] - [L]}_{(N+1) \times (N+1)} \underbrace{[S]}_{(N+1) \times N} \underbrace{[A]}_{(N+1) \times (M+1)} \underbrace{[F^{i-1}]}_{N \times (M+1)} \underbrace{[W_2]}_{(M+1) \times n} = \mathbf{X}_0 + [P_1][A][F^{i-1}][W_2], \text{ where}$$

The matrices are defined as:

- $\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_1(-1) & \dots & \mathbf{x}_n(-1) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$ (size $N+1 \times n$)
- $[L] = \begin{bmatrix} T_0(-1) & \dots & T_N(-1) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$ (size $(N+1) \times (N+1)$)

The diagram also shows the application of the **LEAST SQUARES OPERATOR** (red box) and the **INTEGRATION OPERATOR** (blue box) to the integral term.

Matrix Representation

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

FIRST ORDER ALGORITHM

Dynamics & Initial Conditions

$$\frac{dx}{dt}(t) = f(t, x(t)), \quad x(t_0) = x_0, \quad t_0 = 0, \quad t_f = T.$$

Time and τ

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

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

Picard Iteration

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

$$\beta^i = X_0 + [P_1][A][F^{i-1}][W_2]$$

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

Convergence

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

Update

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

Iterate

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

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

$$X_0 = \begin{bmatrix} x_1(-1) & \dots & x_n(-1) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix},$$

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

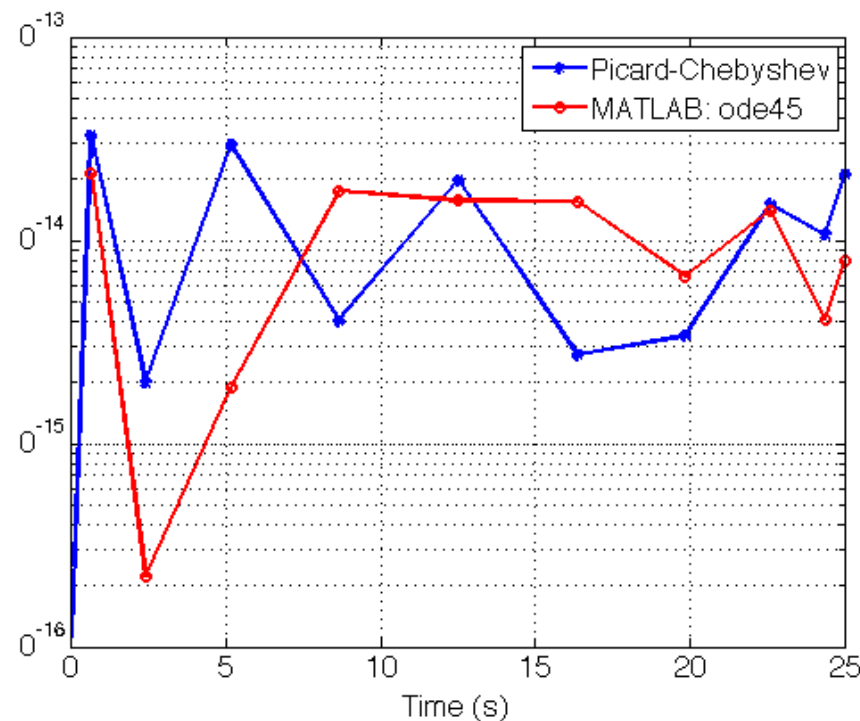
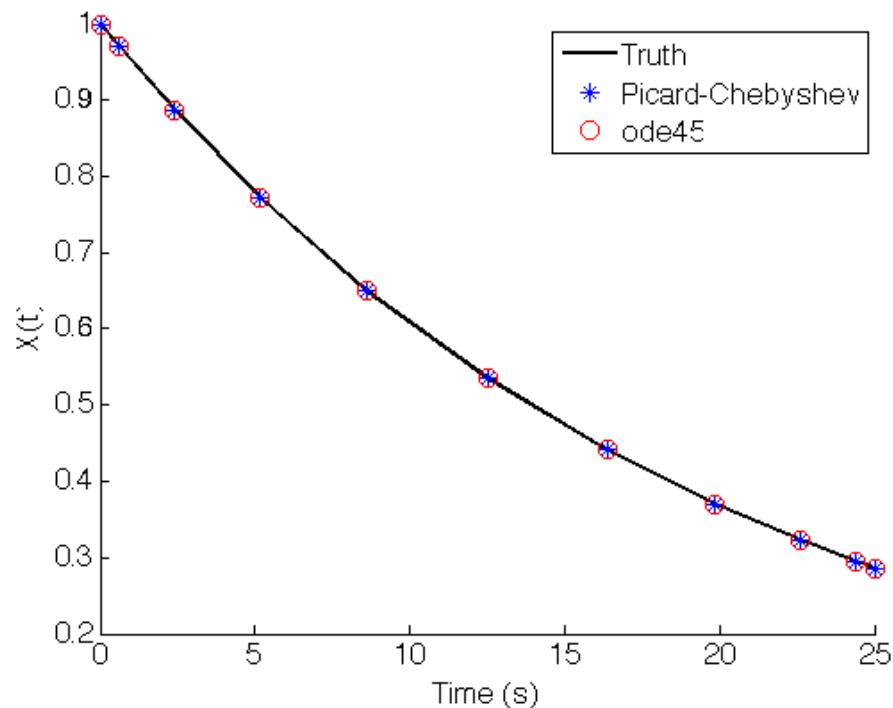
$$[P_1] = [I] - [L][S],$$

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

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

EXAMPLE 1

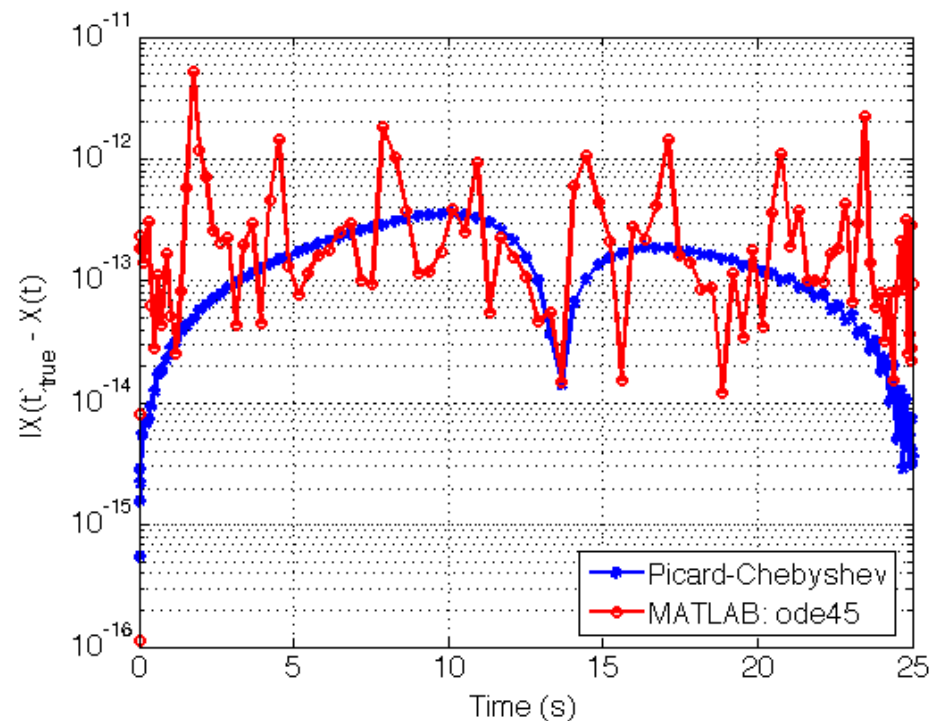
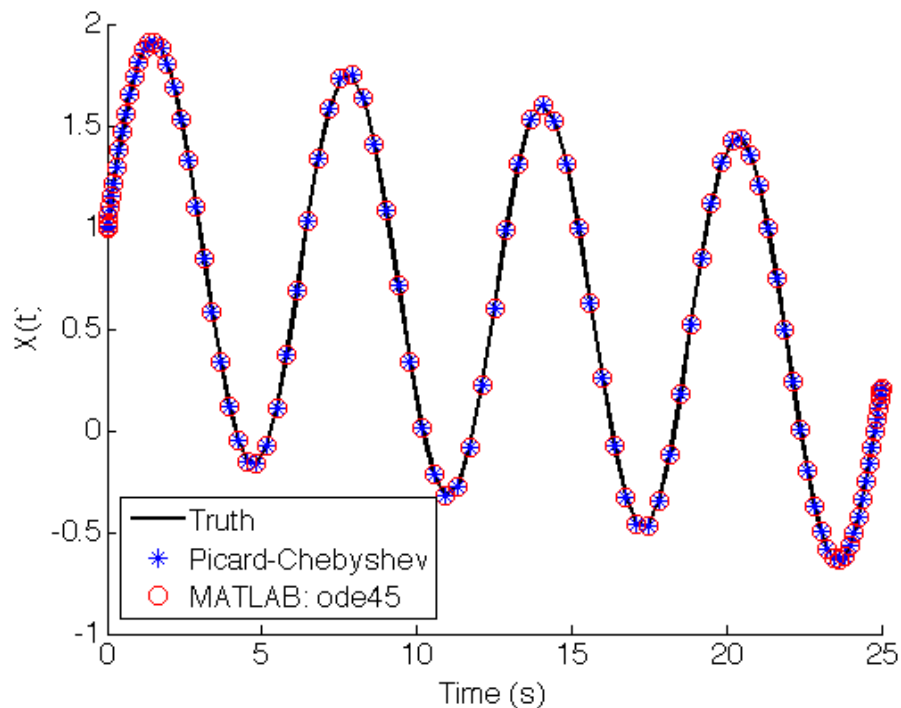
First Order Example: $\dot{x}(t) = -0.05x$



This simple first order example is solved using the Picard-Chebyshev technique and MATLAB's ode45. There 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. There 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_ivpl.m](#).

SECOND ORDER METHOD

Second Order Differential Equation

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

Velocity Approximation

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

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

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

Position Approximation

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

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

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

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.

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

SECOND ORDER METHOD: STEP 1

Velocity Integration

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

Expression for Velocity Coefficients

$N \times n$

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-2}^i \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix} = \mathbf{V}_0 + \underbrace{[\mathbf{I}] - [\mathbf{L}]}_{N \times N} \underbrace{[\mathbf{S}]}_{N \times (N-1)} \underbrace{[\mathbf{A}]}_{(N-1) \times (M+1)} \underbrace{[\mathbf{F}^{i-1}]}_{(M+1) \times n} [\mathbf{W}_2] = \mathbf{V}_0 + [\mathbf{P}_1][\mathbf{A}][\mathbf{F}^{i-1}][\mathbf{W}_2] \text{ where}$$

LEAST SQUARES OPERATOR

INTEGRATION OPERATOR

$N \times n$

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

$N \times N$

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

Matrix Representation for Velocity

$$\boldsymbol{\beta}^i = \mathbf{V}_0 + [\mathbf{P}_1][\mathbf{A}][\mathbf{F}^{i-1}][\mathbf{W}_2], \quad \mathbf{v}^i(t) = T(\tau) \boldsymbol{\beta}^i.$$

SECOND ORDER METHOD: STEP 2

Position Integration

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

Expression for Position Coefficients

**INTEGRATION
OPERATOR**

↓

↑

Velocity Coefficients

$$\begin{aligned}
 \begin{matrix} (N+1) \times n \\ \left[\begin{array}{c} \alpha_0^i \\ \alpha_1^i \\ \vdots \\ \alpha_{N-1}^i \\ \alpha_N^i \end{array} \right] \end{matrix} &= \begin{matrix} (N+1) \times n \\ X_0 \end{matrix} + \underbrace{\begin{matrix} (N+1) \times N \\ [P_2] \end{matrix}}_{\substack{(N+1) \times (N+1) \\ \uparrow \\ (N+1) \times N}} \begin{matrix} N \times n \\ \left[\begin{array}{c} \beta_0^i \\ \beta_1^i \\ \vdots \\ \beta_{N-2}^i \\ \beta_{N-1}^i \end{array} \right] \end{matrix} \\
 &= X_0 + \underbrace{[[I] - [L]]}_{(N+1) \times (N+1)} [S]
 \end{aligned}$$

$[W_2] = X_0 + [P_2] \beta^i [W_2]$ where

$(N+1) \times n$

$X_0 = \begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix},$

$(N+1) \times (N+1)$

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

Matrix Representation for Position

$$\alpha^i = X_0 + [P_2] \beta^i [W_2], \quad \mathbf{x}^i(t) = T(\tau) \alpha^i.$$

SECOND ORDER ALGORITHM

Dynamics & Initial Conditions

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

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

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

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

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

Convergence

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

Update

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

Velocity Matrices

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

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

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

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

Position Matrices

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

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

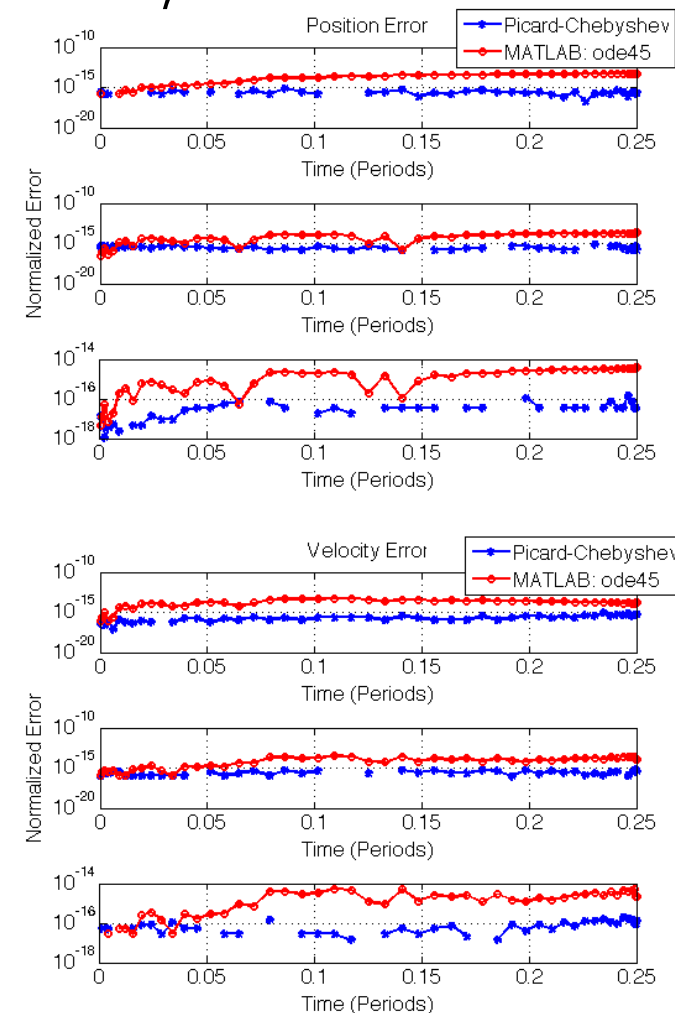
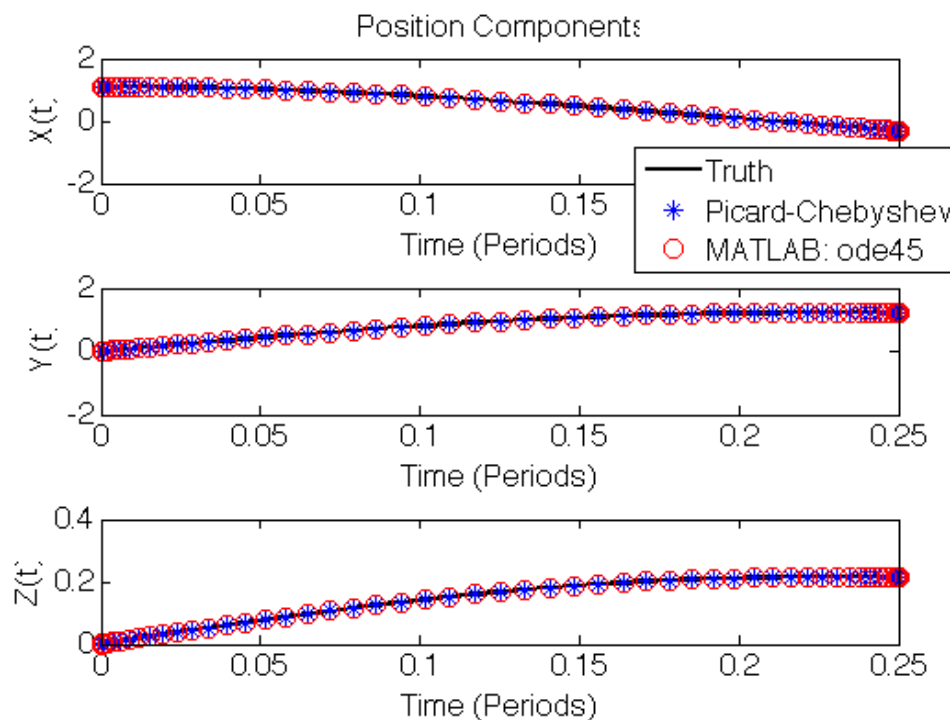
$$T(\tau) = \begin{bmatrix} T_0(\tau_0) & \cdots & T_N(\tau_0) \\ T_0(\tau_1) & \cdots & T_N(\tau_1) \\ \vdots & \ddots & \vdots \\ T_0(\tau_M) & \cdots & T_N(\tau_M) \end{bmatrix}$$

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

EXAMPLE 3

Second Order Example: Two-body Problem

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

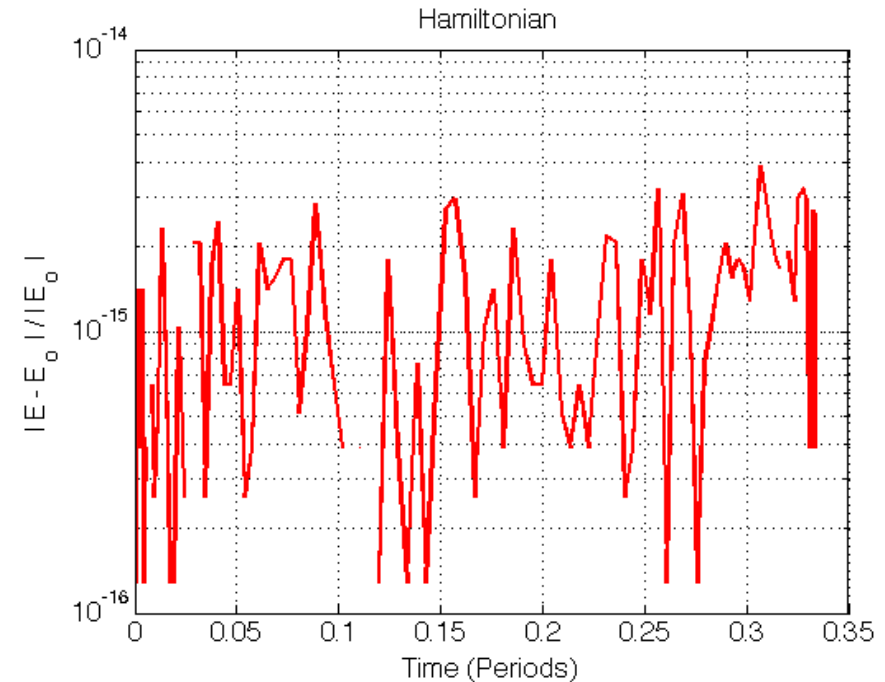
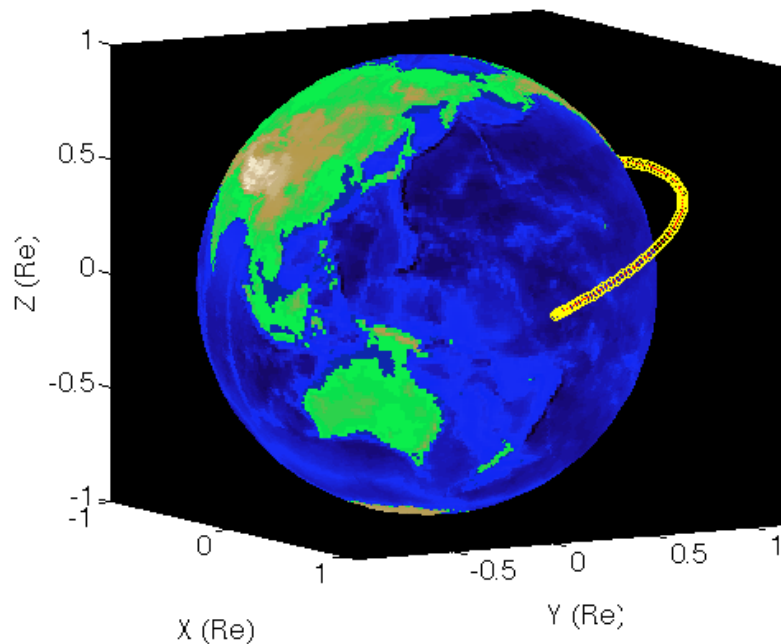


There is an analytical solution to the two-body problem in Celestial mechanics. In this example we demonstrate how the second order Picard-Chebyshev technique is used to integrate a second order system of differential equations. The code for generating the above figures is available for use as a learning tool: [run_lecture3_example3_ivp11.m](#).

EXAMPLE 4

Second Order Example: Perturbed Two-body Problem

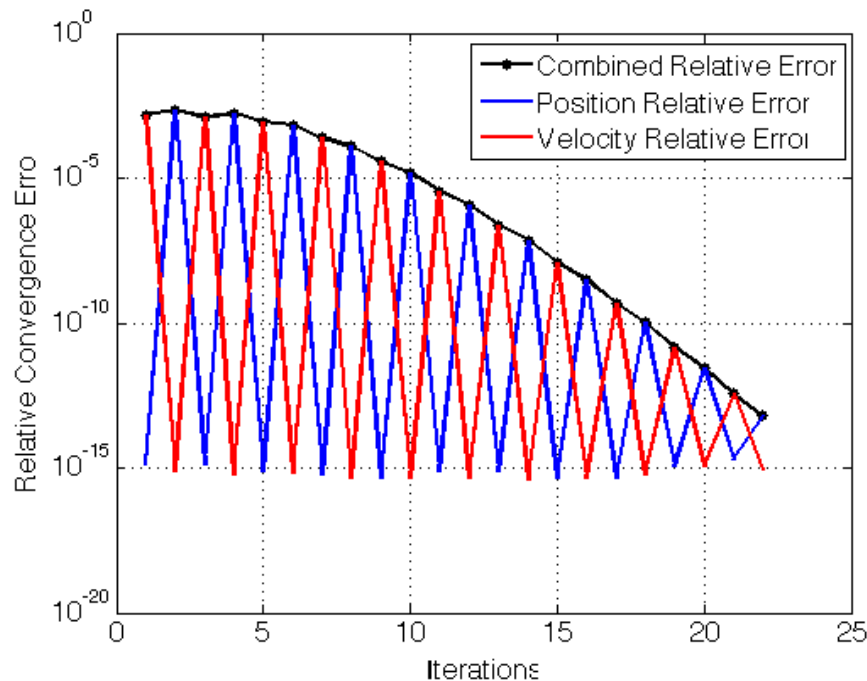
$$\ddot{\mathbf{r}}(t) = -\frac{\mu}{r^3} \mathbf{r} + \mathbf{a}_d(x, y, z)$$



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](#).

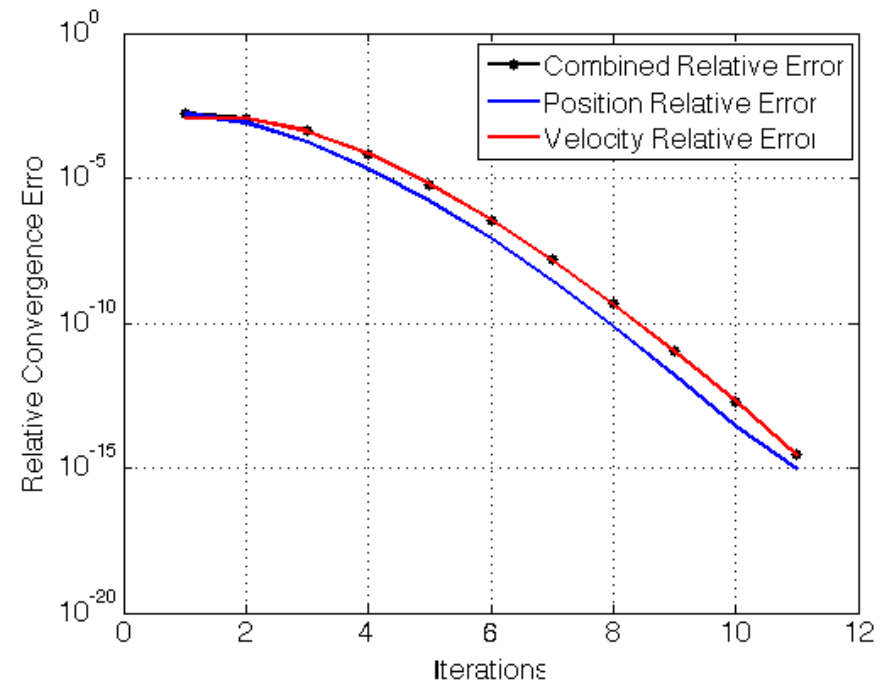
First Order vs Second Order

First Order Algorithm



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

Second Order Algorithm



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

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_ivpl.m](#).

BOUNDARY VALUE PROBLEMS

Types of Boundary Value Problems

- ***BVP of the first kind:*** x_0 and 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:*** x_0 and 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:*** x_f and 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

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

Pseudo Velocity Approximation

The velocity can be written in terms of a Chebyshev series, and similar to the second order IVP, the β coefficients can be computed in terms of the least squares α coefficients. The **initial velocity** is unknown and the resulting **pseudo velocity** is correct to within the **constant of integration**. This constant only effects the β_0 coefficient. All other coefficients are correct.

$$\mathbf{v}_{pseudo}^i(\tau) = \sum_{k=0}^{N-1} \beta_{k pseudo}^i T_k(s) = \cancel{\mathbf{v}_0} + \int_{-1}^s \sum_{k=0}^{N-2} \alpha_k^{i-1} T_k(q) dq.$$

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 α_0 and α_1 position coefficients. These must be determined using some **other information**.

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

STEP 1: PSEUDO VELOCITY

Pseudo Velocity (correct to within a constant)

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

Expression for Pseudo Velocity Coefficients

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

LEAST SQUARES OPERATOR

INTEGRATION OPERATOR

Matrix Representation for Pseudo Velocity

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

STEP 2: PSEUDO POSITION

Pseudo Position (linearly contained integration constant)

$$\mathbf{x}_{pseudo}^i(\tau) = \sum_{k=0}^N \alpha_{k pseudo}^i T_k(\tau) = \int_{-1}^{\tau} \sum_{k=0}^{N-1} \beta_{k pseudo}^i T_k(s) ds.$$

Expression for Pseudo Position Coefficients

The diagram illustrates the relationship between the pseudo position coefficients α_{pseudo}^i and the pseudo velocity coefficients β_{pseudo}^i through an integration operator. The pseudo position coefficients are represented as a column vector of size $(N+1) \times n$. This vector is equal to the product of a matrix $[I] - [L]$ of size $(N+1) \times (N+1)$ and a matrix $[S]$ of size $(N+1) \times N$, multiplied by a column vector of pseudo velocity coefficients β_{pseudo}^i of size $N \times n$. The matrix $[P_2]$ of size $(N+1) \times N$ is also shown. The integration operator is represented by a blue box labeled "INTEGRATION OPERATOR". The pseudo velocity coefficients are represented as a column vector of size $N \times n$. The matrix $[L]$ is defined as $[L] = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$ of size $(N+1) \times (N+1)$.

$$\begin{bmatrix} \alpha_{0 pseudo}^i \\ \alpha_{1 pseudo}^i \\ \vdots \\ \alpha_{N-1 pseudo}^i \\ \alpha_{N pseudo}^i \end{bmatrix} = \begin{bmatrix} [I] - [L] \end{bmatrix} \begin{bmatrix} [S] \end{bmatrix} \begin{bmatrix} \beta_{0 pseudo}^i \\ \beta_{1 pseudo}^i \\ \vdots \\ \beta_{N-2 pseudo}^i \\ \beta_{N-1 pseudo}^i \end{bmatrix}$$

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

Matrix Representation for Pseudo Position

$$\alpha_{pseudo}^i = [P_2] \beta_{pseudo}^i [[W_2]], \quad \mathbf{x}_{pseudo}^i(t) = T(\tau) \alpha_{pseudo}^i.$$

STEP 3: VELOCITY & POSITION

Position Trajectory

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

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

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

Vector-matrix form

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

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

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

Velocity Coefficients

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

Position Coefficients

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

ALTERNATE METHOD FOR COEFFICIENTS

Position Solution

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

The first two position coefficients, α_0 and α_1 , can be determined using knowledge of both the terminal boundary conditions.

Boundary Conditions

The left boundary condition can be written as:

$$\mathbf{x}(-1) = \sum_{k=0}^N \alpha_k^i T_k(-1) = \alpha_0^i - \alpha_1^i + \alpha_2^i + \dots + (-1)^N \alpha_N^i.$$

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

The right boundary condition can be written as:

$$\mathbf{x}(1) = \sum_{k=0}^N \alpha_k^i T_k(1) = \alpha_0^i + \alpha_1^i + \alpha_2^i + \dots + \alpha_N^i.$$

This produces **two equations** and **two unknowns**, allowing α_0 and α_1 to be computed in terms of **the known initial and final position**, and the other **known coefficients**.

$$\alpha_0^i = \frac{\mathbf{x}(1) + \mathbf{x}(-1)}{2} - (\alpha_2^i + \alpha_4^i + \alpha_6^i \dots),$$

$$\alpha_1^i = \frac{\mathbf{x}(1) - \mathbf{x}(-1)}{2} - (\alpha_3^i + \alpha_5^i + \alpha_7^i \dots).$$

ALTERNATE METHOD FOR COEFFICIENTS

Position Coefficients

(N+1) × N
[P_B]

(N+1) × n

(N+1) × n

N × n

(N+1) × n

Pseudo Position Coefficients

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

$$\begin{bmatrix} \alpha_0^i \\ \alpha_1^i \\ \vdots \\ \alpha_{N-1}^i \\ \alpha_N^i \end{bmatrix} = X_{0f} + \begin{bmatrix} 0 & 0 & -1 & 0 & -1 & 0 & \dots \\ 0 & 0 & 0 & -1 & 0 & -1 & \dots \\ 0 & 0 & 1 & 0 & \dots & & 0 \\ & & \dots & \ddots & \dots & & 0 \\ \vdots & \vdots & \dots & 0 & 1 & 0 & 0 \\ & & \dots & & 0 & 1 & 0 \\ 0 & \dots & & & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{pseudo0}^i \\ \alpha_{pseudo1}^i \\ \vdots \\ \alpha_{pseudoN-1}^i \\ \alpha_{pseudoN}^i \end{bmatrix} = X_{0f} + [P_B] \alpha_{pseudo}^i, \quad X_{0f} = \begin{bmatrix} \frac{x_1(1) + x_1(-1)}{2} & \dots & \frac{x_n(1) + x_n(-1)}{2} \\ \frac{x_1(1) - x_1(-1)}{2} & \dots & \frac{x_n(1) - x_n(-1)}{2} \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}.$$

Matrix Representation for Position

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

TPBVP ALGORITHM

Dynamics & Initial Conditions

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

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

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

$$\beta_{pseudo}^i = [P_1][A][F^{i-1}][[W_2]], \alpha_{pseudo}^i = [P_2]\beta_{pseudo}^i[[W_2]].$$

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

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

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

Convergence

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

Update

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

Velocity Matrices

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

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

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

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

Position Matrices

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

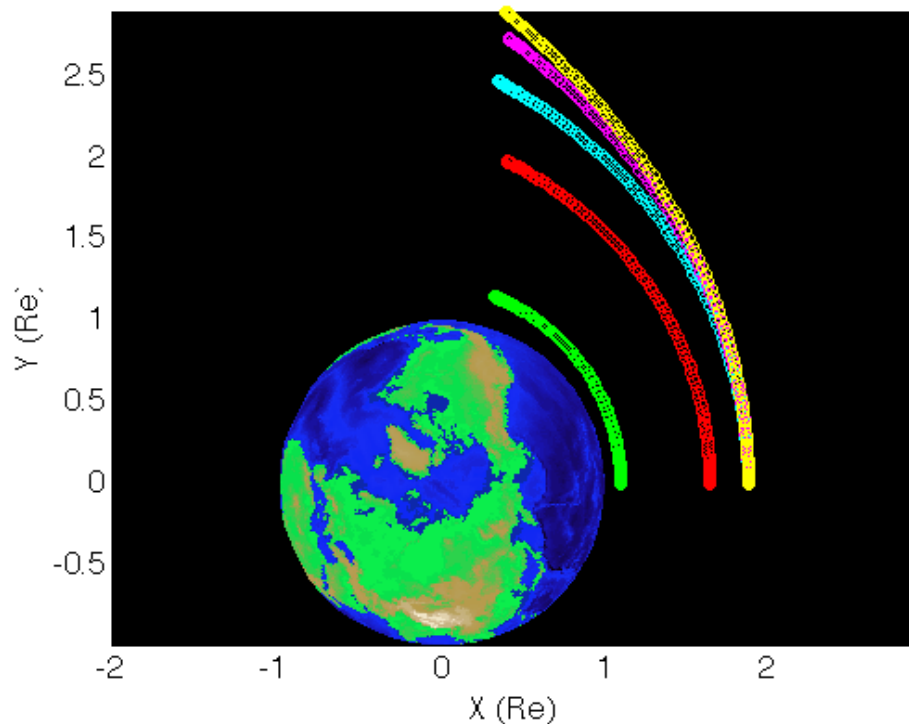
$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad [L] = \begin{bmatrix} T(-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

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

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

EXAMPLE 5

TPBVP Example: Perturbed Two-body Problem



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

- 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 (x_0 and v_f)

Second order system & boundary conditions

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

$$x(t_0) = x_0, \quad v(t_f) = v_f, \quad x(t), v(t) \in R^{1 \times n}.$$

Compute Velocity: **Final Value Problem**

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

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

Compute Position: **Initial Value Problem**

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

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

$$x^i(\tau) = \sum_{k=0}^N \alpha_k^i T_k(\tau) = x(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \beta_k^i T_k(s) ds.$$

BVP Type III (x_f and v_0)

Second order system & boundary conditions

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

$$x(t_f) = x_f, \quad v(t_0) = v_0, \quad x(t), v(t) \in R^{1 \times n}.$$

Compute Velocity: **Initial Value Problem**

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

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

Compute Position: **Final Value Problem**

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

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

$$x^i(\tau) = \sum_{k=0}^N \alpha_k^i T_k(\tau) = x(1) + \int_{\tau}^1 \sum_{k=0}^{N-1} \beta_k^i T_k(s) ds.$$

BVP Type II & Type III

BVP Type II (x_0 and v_f)

Velocity $v^i(\tau) = \sum_{k=0}^{N-1} \beta_k^i T_k(s) = v(1) + \int_s^1 \sum_{k=0}^{N-2} a_k^{i-1} T_k(q) dq.$

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 $\frac{d\tau}{dt} = -\frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \beta_0^i \\ \vdots \\ \beta_{N-1}^i \end{bmatrix} = V_f + ([U] - [I])[S][A][F^{i-1}][-W_2], \quad v^i(t) = T(\tau)\beta^i.$$

$$V_f = \begin{bmatrix} v_1(1) & \cdots & v_n(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} T_0(1) & \cdots & T_N(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

Position $x^i(\tau) = \sum_{k=0}^N \alpha_k^i T_k(\tau) = x(-1) + \int_{-1}^{\tau} \left\{ v(1) + \int_s^1 \sum_{k=0}^{N-2} a_k^{i-1} T_k(q) dq \right\} d\tau.$

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 $\frac{d\tau}{dt} = \frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \alpha_0^i \\ \vdots \\ \alpha_N^i \end{bmatrix} = X_0 + ([I] - [L])[S] \begin{bmatrix} \beta_0^i \\ \vdots \\ \beta_{N-1}^i \end{bmatrix} [W_2], \quad x^i(t) = T(\tau)\alpha^i.$$

$$X_0 = \begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \quad L = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

BVP Type III (x_f and v_0)

Velocity $v^i(\tau) = \sum_{k=0}^{N-1} \beta_k^i T_k(s) = v(-1) + \int_{-1}^s \sum_{k=0}^{N-2} a_k^{i-1} T_k(q) dq.$

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 $\frac{d\tau}{dt} = \frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \beta_0^i \\ \vdots \\ \beta_{N-1}^i \end{bmatrix} = V_0 + ([I] - [L])[S][A][F^{i-1}][W_2], \quad v^i(t) = T(\tau)\beta^i.$$

$$V_0 = \begin{bmatrix} v_1(-1) & \cdots & v_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \quad L = \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

Velocity $x^i(\tau) = \sum_{k=0}^N \alpha_k^i T_k(\tau) = x(1) + \int_{\tau}^1 \left\{ v(-1) + \int_{-1}^s \sum_{k=0}^{N-2} a_k^{i-1} T_k(q) dq \right\} d\tau.$

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 $\frac{d\tau}{dt} = -\frac{2}{(t_f-t_0)}$.

$$\begin{bmatrix} \alpha_0^i \\ \vdots \\ \alpha_N^i \end{bmatrix} = X_f + ([U] - [I])[S] \begin{bmatrix} \beta_0^i \\ \vdots \\ \beta_{N-1}^i \end{bmatrix} [-W_2], \quad x^i(t) = T(\tau)\alpha^i.$$

$$X_f = \begin{bmatrix} x_1(1) & \cdots & x_n(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} T_0(1) & \cdots & T_N(1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

BVP II ALGORITHM

Dynamics & Initial Conditions

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

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

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

$$\boldsymbol{\beta}^i = \mathbf{V}_f + [P_1][A][F^{i-1}][-\mathbf{W}_2], \mathbf{v}^i(t) = T(\tau)\boldsymbol{\beta}^i.$$

$$\boldsymbol{\alpha}^i = \mathbf{X}_0 + [P_2]\boldsymbol{\beta}^i[\mathbf{W}_2], \mathbf{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i.$$

Convergence

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

Update

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

Velocity Matrices

$$[P_1] = [[\mathbf{U}] - [\mathbf{I}]] [\mathbf{S}]$$

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

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

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

Position Matrices

$$[P_2] = [[\mathbf{I}] - [\mathbf{L}]] [\mathbf{S}]$$

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

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

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

BVP III ALGORITHM

Dynamics & Initial Conditions

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

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

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

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

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

Convergence

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

Update

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

Velocity Matrices

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

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

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

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

Position Matrices

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

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

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

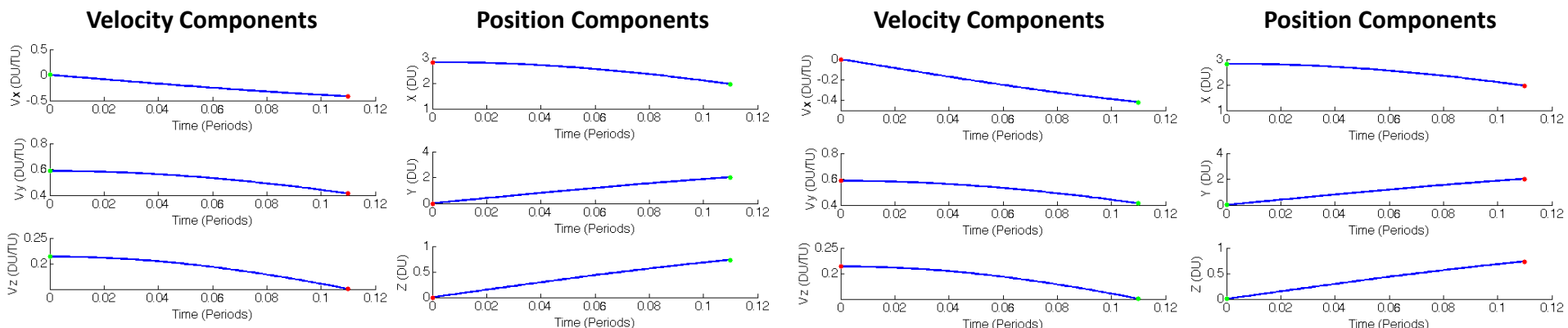
$$[S] = \begin{bmatrix} \frac{1}{4} & 0 & \cdots & & 0 \\ 1 & 0 & -1/2 & 0 & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & \\ & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & & \frac{1}{2(N-2)}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

EXAMPLE: BVP Type II & III

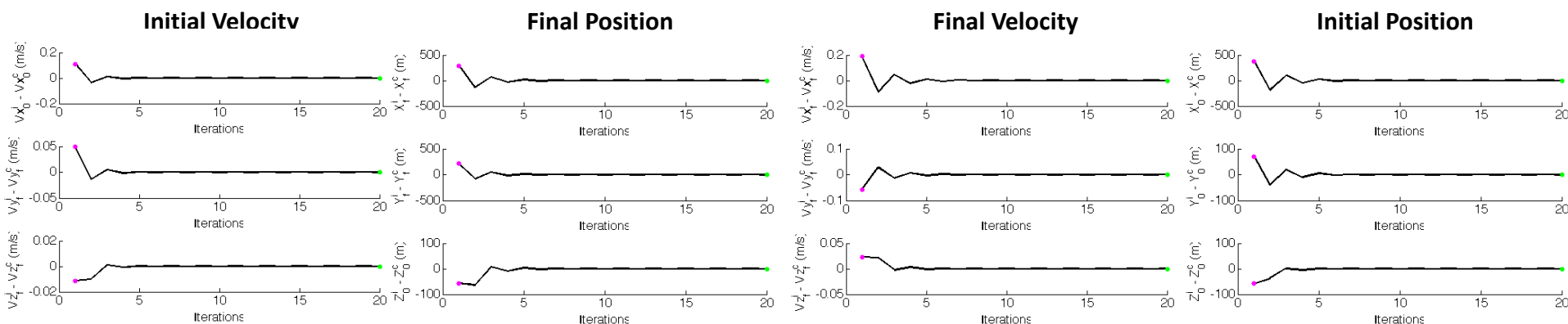
BVP Type II (x_0 and v_f)

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

BVP Type III (x_f and v_0)



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



The Picard-Chebyshev BVP algorithm also converges over a fraction of an orbit (slightly less than 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.

$$\frac{dx(t)}{dt} = cx(t), \quad x(t_0) = x_0, \quad x \in R^{1 \times n}.$$

Picard-Chebyshev Vector Matrix Notation with $t \Rightarrow \tau$

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

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

or

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

$$\boxed{\mathbf{x}^i = c \left(\frac{t_f - t_0}{2} \right)^{\overset{\text{constant} \equiv M}{(N+1) \times (N+1)}} [TP_1 A] \mathbf{x}^{i-1} + \mathbf{x}_0;}$$

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

- 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 :

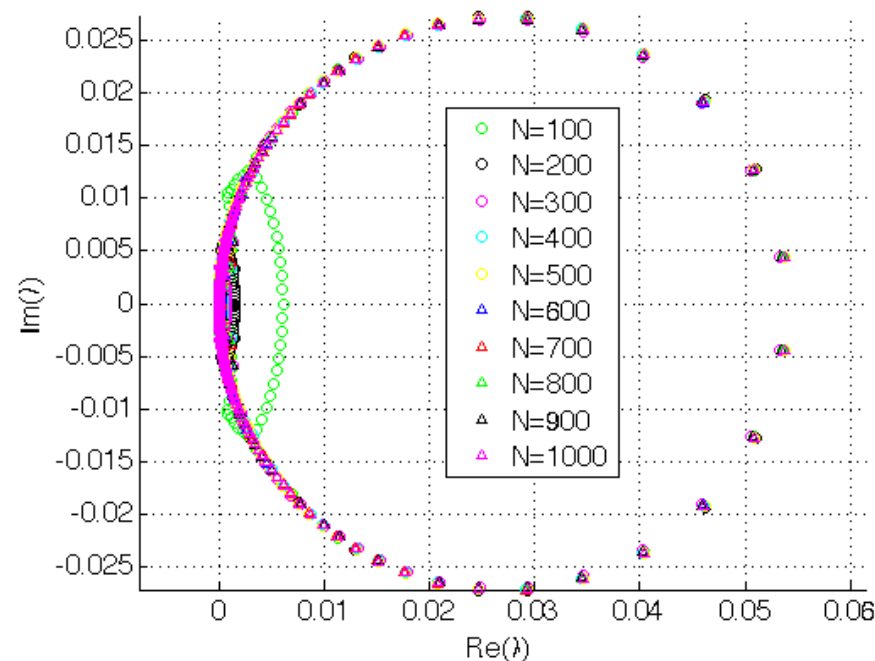
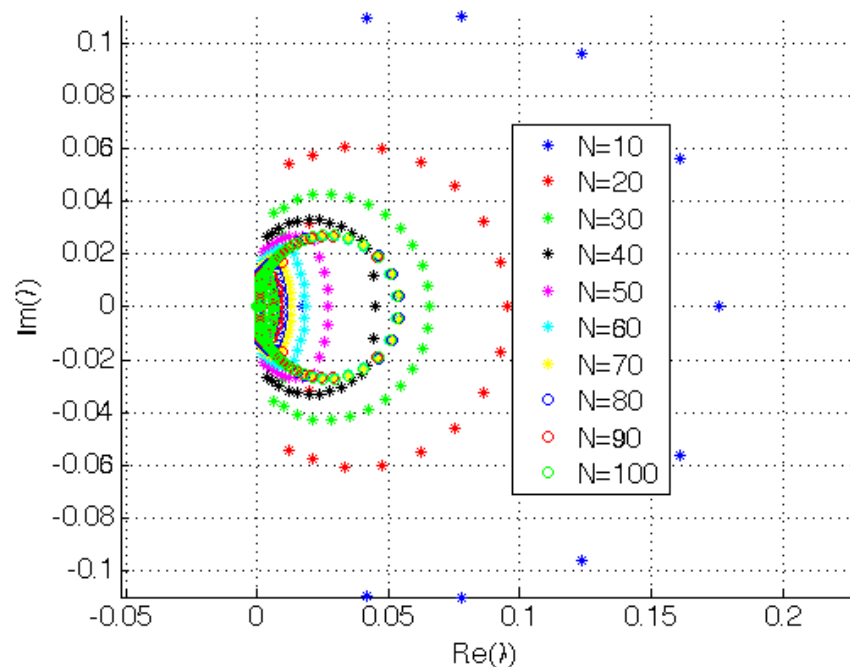
$$\left| \left(\frac{c(t_f - t_0)}{2} \right) \lambda_{\max}[M] \right| < 1 \quad \text{or} \quad |t_f - t_0| < \frac{2}{|c \lambda_{\max}[M]|}$$

Note

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

EIGENVALUE ANALYSIS: FIRST ORDER

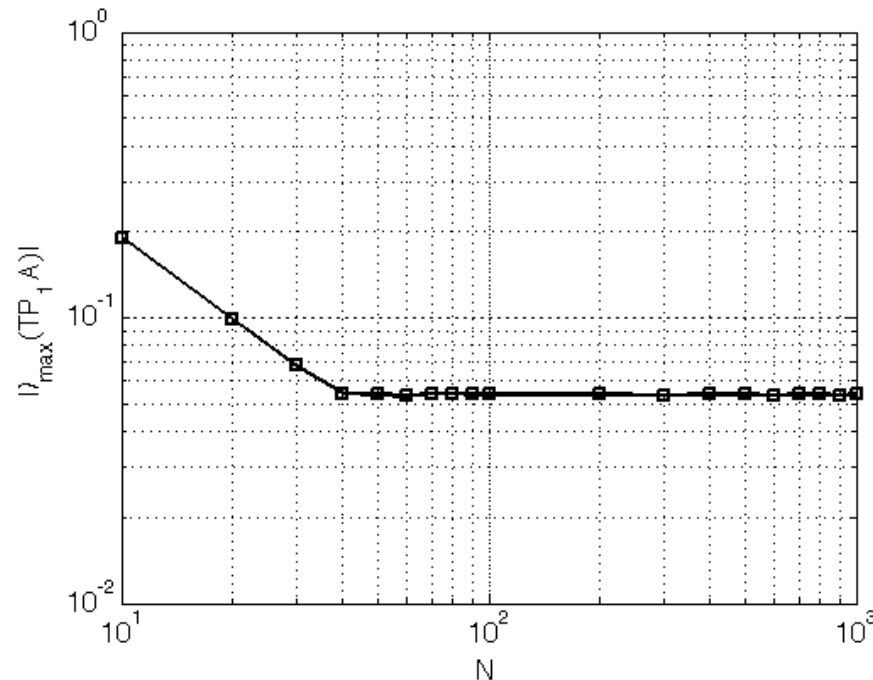
Root Locus Plots for $\lambda_{\max}(TP_1A)$



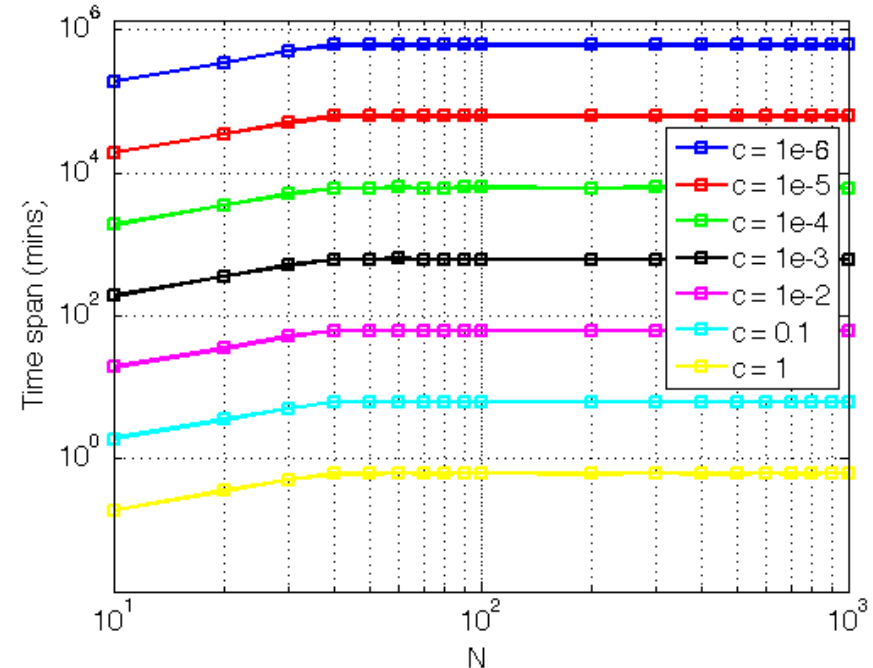
- For $N > 40$, the **maximum eigenvalue** of the matrix product $[TP_1A]$ or $\lambda_{\max}([TP_1A])$ is attracted to a **fixed point** on the root locus plots above.
- The code for generating the above figures is available for use as a learning tool:
[run_lecture3_example6_ivpl_conv.m](#).

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 $M \geq N > 40$ 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_ivpl_conv.m](#).

PICARD-CHEBYSHEV CONVERGENCE: SECOND ORDER

Scalar Problem

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

Picard-Chebyshev Vector Matrix Notation

$$\text{Solution} \rightarrow \mathbf{x}^i = \underbrace{\left(\frac{t_f - t_0}{2}\right)^2}_{\substack{\text{t} \rightarrow \tau \text{ time} \\ \text{transformation}}} \underbrace{T(\tau)P_2P_1A}_{\substack{\text{Chebyshev} \\ \text{matrix}}} \underbrace{(c\mathbf{x}(t))}_{\substack{\text{Least squares} \\ \text{operator}}} + \mathbf{x}_0 \leftarrow \text{Initial condition vector}$$

Integration operators: I & II Forcing function

- 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 .

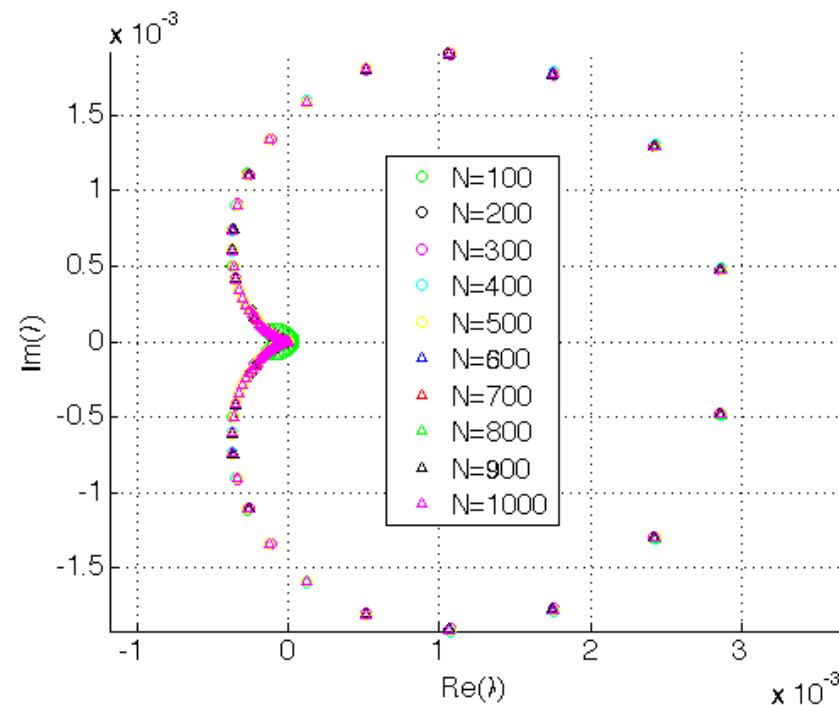
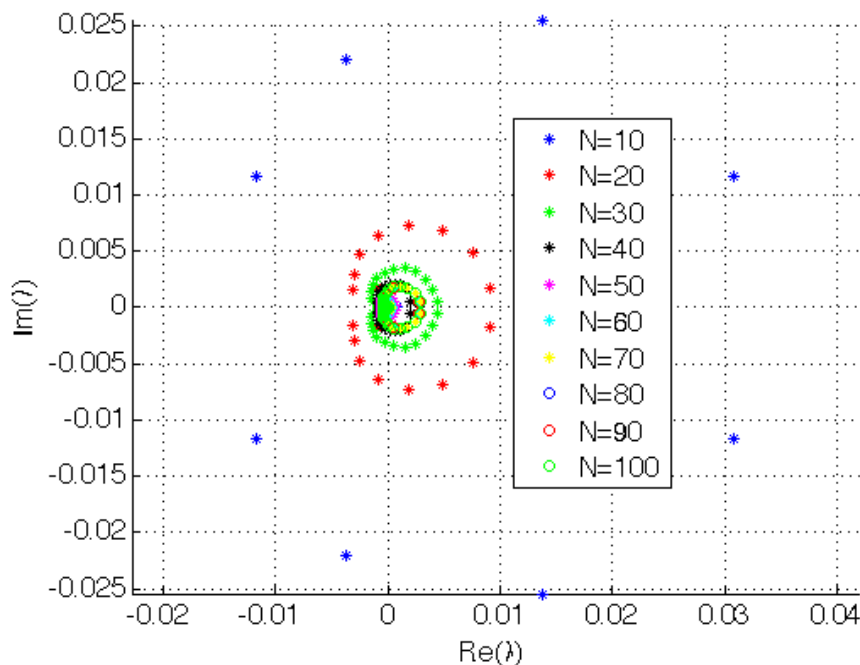
$$\left| \left(\frac{t_f - t_0}{2}\right)^2 c \lambda_{\max} (T(\tau)P_2P_1A) \right| < 1 \quad \text{or} \quad |t_f - t_0| < \frac{2}{\sqrt{|c \lambda_{\max} (T(\tau)P_2P_1A)|}}$$

Notation

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

EIGENVALUE ANALYSIS: SECOND ORDER

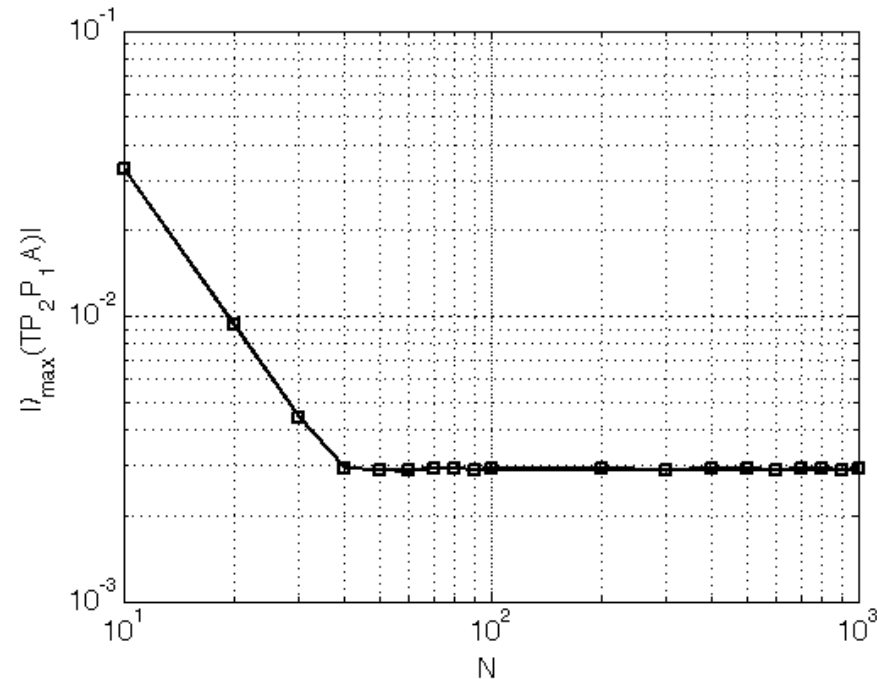
Root Locus Plots for $\lambda_{\max}(\text{TP}_2\text{P}_1\text{A})$



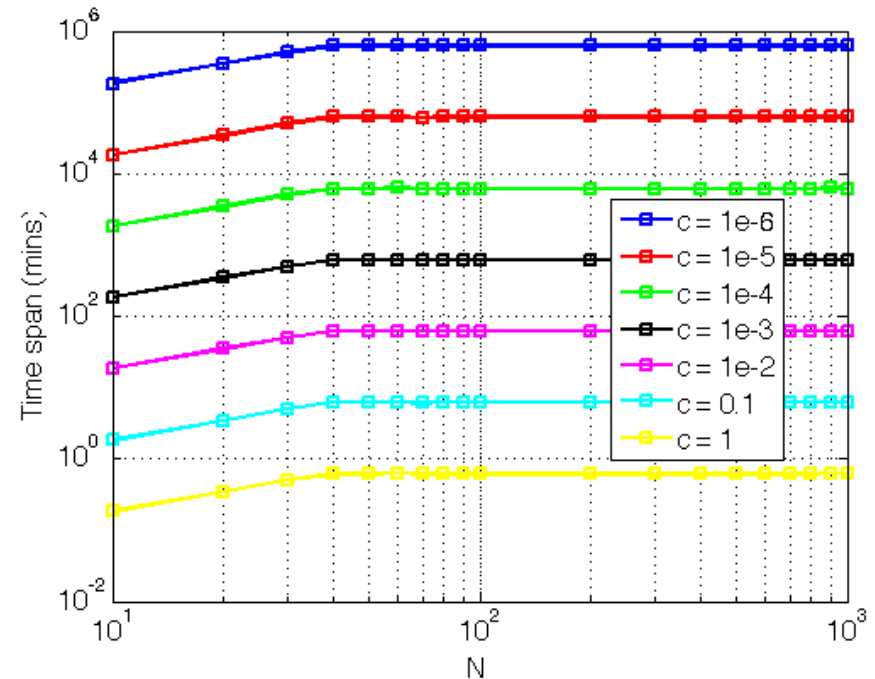
- For $N > 40$, the **maximum eigenvalue** of the matrix product $[\text{TP}_2\text{P}_1\text{A}]$ or $\lambda_{\max}([\text{TP}_2\text{P}_1\text{A}])$ is attracted to a **fixed point** on the root locus plots above.
- Note that $\lambda_{\max}([\text{TP}_2\text{P}_1\text{A}])$ for the **second order 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_ivpll_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 $M \geq N > 40$ 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: $\frac{dx^2(t)}{dt^2} = cx(t)$, $x(t_0) = x_0$, $x(t_f) = x_f$, $x \in R^{1 \times n}$.

Picard-Chebyshev Vector Matrix Notation

The diagram shows the equation $x^i = \left(\frac{t_f - t_0}{2} \right)^2 T(\tau) P_B P_2 P_1 A (cx(t)) + x_{0f}$ with various annotations:

- Solution** (blue arrow) points to x^i .
- $t \rightarrow \tau$ time transformation** (orange bracket) is under $\left(\frac{t_f - t_0}{2} \right)^2$.
- Chebyshev matrix** (green arrow) points to $T(\tau)$.
- Integration operators: I, II & P_B** (blue bracket) is under $P_B P_2 P_1$.
- Least squares operator** (red arrow) points to A .
- Forcing function** (purple bracket) is under $cx(t)$.
- Initial condition vector** (red arrow) points to x_{0f} .

- If the max eigenvalue < 1 , Picard sequence converges (analogous to difference equations).
- The matrix product $T(\tau)P_BP_2P_1A$ is constant once N is selected.
- Max eigenvalues are scaled by the time of flight $t_f - t_0$ and c .

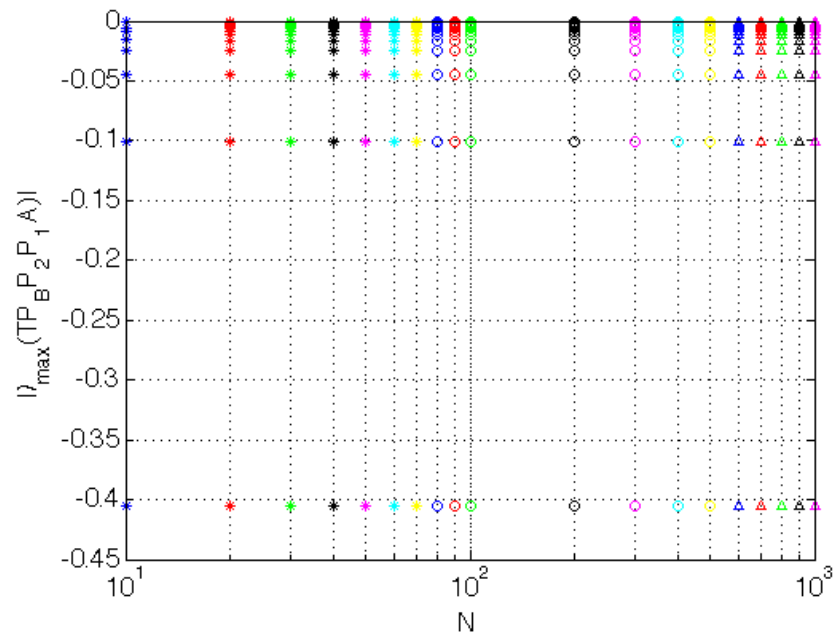
$$\left| \left(\frac{t_f - t_0}{2} \right)^2 c \lambda_{\max} (T(\tau)P_BP_2P_1A) \right| < 1 \quad \text{or} \quad |t_f - t_0| < \frac{2}{\sqrt{|c \lambda_{\max} (T(\tau)P_BP_2P_1A)|}}$$

Notation

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

EIGENVALUE ANALYSIS: TPBVP

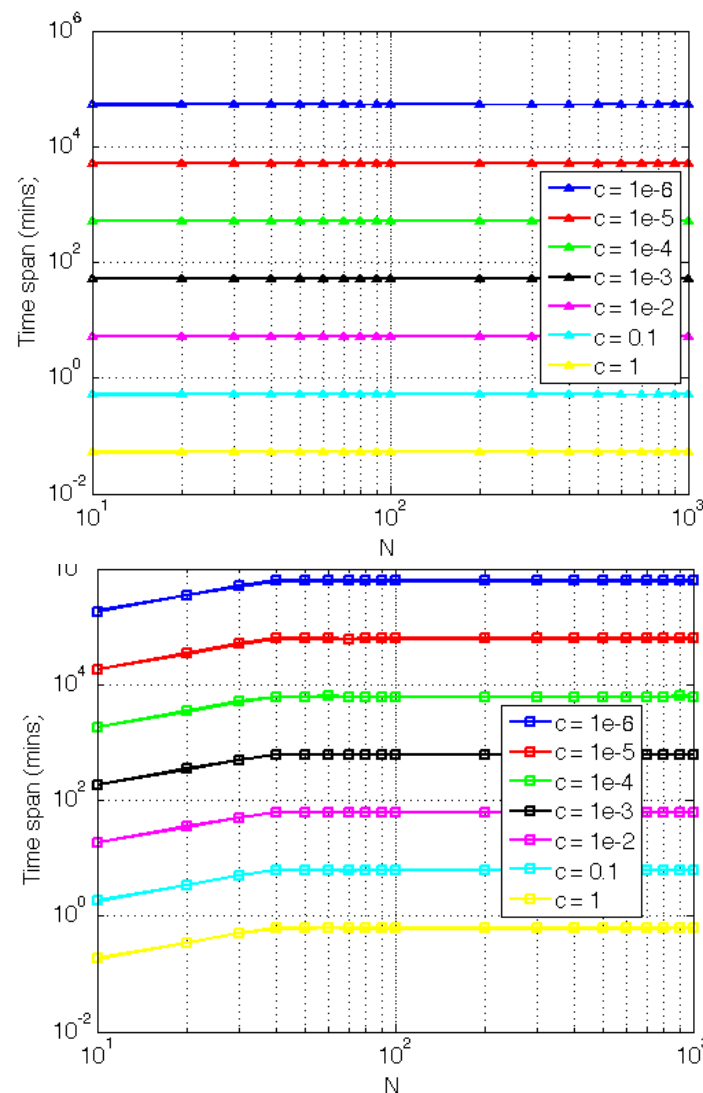
MAX EIGENVALUE VS N



The **maximum eigenvalue** of the matrix product $\lambda_{\max}(\text{TP}_B \text{P}_2 \text{P}_1 A)$ 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

DOMAIN OF CONVERGENCE



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.