## **Adaptive Picard-Chebyshev Iteration with Segmentation and Chebyshev Polynomial Approximation (APCI)**

**Purpose:** The **Adaptive Picard-Chebyshev Iteration** method is a numerical technique for solving ordinary differential equations (ODEs) that combines **Picard iteration** with **Chebyshev polynomial approximation** for high-precision orbital propagation. This method is particularly well-suited for problems like satellite orbit propagation, where accuracy is crucial over long time spans. The analysis extends this approach by incorporating an adaptive scheme, which determines the **degree of the Chebyshev polynomial** and **segmentation** based on the problem’s characteristics and tolerance requirements. This adaptive segmentation allows the method to manage large orbital periods effectively by breaking them into smaller, more manageable segments.

**Overview:** The **Picard-Chebyshev propagator** iteratively solves the ODE describing satellite motion by approximating the solution using Chebyshev polynomials over defined segments. The **degree of the Chebyshev polynomial** is determined based on the problem’s tolerance, ensuring that the approximation is accurate. The **segmentation** divides the total time span into smaller intervals, with each segment being solved using the Picard-Chebyshev method. This approach allows the method to adapt to the complexity of the solution, ensuring high accuracy while managing computational costs.

### Mathematical Formulas and Coefficients

**Picard Iteration**

Where:

* is the state vector (e.g., position and velocity).
* is the system of differential equations (the forces acting on the system, such as gravitational forces for satellite motion).

**Chebyshev Polynomial Approximation**

To approximate the function , a series of Chebyshev polynomials ( is used:

Where:

* ​ are the coefficients obtained through a least-squares fit of the function .
* are the Chebyshev polynomials, which form an orthogonal basis over the interval [−1,1].

**Chebyshev Polynomial Basis and Nodes**

The Chebyshev polynomials are sampled at the Chebyshev-Gauss-Lobatto nodes :

This ensures that the approximation is more accurate near the boundaries of the interval.

**Integral Form of Picard Iteration with Chebyshev Polynomials**

Once the approximation is represented as a Chebyshev series, the integral for the Picard iteration becomes:

Where:

* ​ are the Chebyshev coefficients from the previous iteration.
* The integral of the Chebyshev polynomials can be computed analytically.

**Adaptive Segmentation**

The total time interval is divided into smaller segments to ensure accuracy over long periods. Each segment is propagated individually using the Picard-Chebyshev iteration. The degree of the Chebyshev polynomial NNN and the segment length are adaptively chosen based on error criteria, often controlled by the magnitude of the last few Chebyshev coefficients:

Where is a small tolerance value.

**Error Feedback with Quasi-Linearization**

To accelerate convergence, an error feedback term is added to the Picard iteration:

Where:

* is the Jacobian of the system evaluated along the current approximation .
* This error feedback accelerates convergence, especially in the final iterations.

**Second-Order Systems (e.g., Satellite Motion)**

For second-order differential equations (such as those governing satellite motion), the system is written in a cascade form:

**,**

Picard iteration is applied to the velocity update first, and the position update is obtained by integrating the velocity:

this ensures kinematic consistency between the velocity and position.

**Node Adaptation**

The number of nodes N and segment size are adapted based on the nonlinearity of the system over the given segment, ensuring computational efficiency and precision. The coefficients are adjusted accordingly.

### Pseudocode:

**Adaptive\_Picard\_Chebyshev(r0, v0, t0, tf, dt, deg, tol, soln\_size, Feval, Soln)**

**// Step 1: Determine Degree and Segmentation**

**Call polydegree\_segments to compute polynomial degree (N), number of segments (seg), and time period (Period)**

**// Calculate coefficient array size based on segments and polynomial degree**

**coeff\_size = Calculate\_Coeff\_Size(tf, Period, seg, N)**

**// Step 2: Prepare Propagator**

**Initialize arrays for storing Chebyshev polynomials and time vectors**

**Call prepare\_propagator to setup matrices and segment times based on polynomial degree and segmentation**

**// Step 3: Picard-Chebyshev Propagation**

**Allocate memory for ALPHA and BETA coefficient arrays**

**Initialize total\_seg and segment\_times arrays**

**// Perform Picard iteration for each segment**

**Call picard\_chebyshev\_propagator to iterate and update coefficients for position and velocity**

**// Step 4: Interpolate the Solution**

**Call interpolate to compute the final solution at user-specified times using Chebyshev coefficients**

**Free allocated memory for ALPHA and BETA**

**End Function**

**Key Notes:**

**Functionality**:

**polydegree\_segments**: Determines polynomial degree and number of segments based on the problem's complexity and tolerance.

**prepare\_propagator**: Prepares the propagation environment, including Chebyshev polynomial calculations and segment times.

**picard\_chebyshev\_propagator**: Performs the core propagation using Picard-Chebyshev iterations.

**interpolate**: Uses the computed Chebyshev coefficients to interpolate the final solution.

**Memory Management**: The algorithm dynamically allocates memory for storing Chebyshev coefficients, which are used across different segments for iterative calculations.

**Adaptivity**: The algorithm adapts the segmentation and polynomial degree based on the tolerance specified by the user, ensuring both accuracy and computational efficiency.

**Important Note:**

This pseudocode provides a general overview of the algorithm's structure. The complete implementation includes additional details for managing memory, iterations, and specific calculations, which are too extensive to fully encapsulate here. However, this structure serves as the base for the Adaptive Picard-Chebyshev Numerical Integration process.

### Time Complexity:

* **Per Iteration:** O (N)

Each iteration involves computing Chebyshev polynomials and their coefficients over N nodes within a segment. This requires a constant number of operations per node, leading to linear complexity with respect to the number of nodes in the segment.

* **Total Complexity:** O (), where S is the number of segments, and N is the number of nodes (polynomial degree) in each segment.

**Explanation:** The total time complexity is determined by the segmentation of the time span and the polynomial degree for each segment. Since the method performs iterations over N nodes for each segment and propagates the solution across S segments, the total complexity scales with . The adaptive nature of the algorithm ensures that the method is applied over manageable segments, improving accuracy and computational efficiency.

### Space Complexity:

* **Overall:** O (), The space complexity is proportional to the number of nodes N in each segment and the number of segments S. Memory is required to store the Chebyshev coefficients and intermediate results for position and velocity values in each segment.

**Explanation:** The algorithm stores position and velocity values for each node within a segment, leading to a space complexity of O(N) per segment. Since the solution is propagated across S segments, the overall space complexity becomes O (), The storage requirements are primarily driven by the Chebyshev coefficients and the state variables (position and velocity) at each node. Additional memory for intermediate results, such as the Picard-Chebyshev coefficients, has minimal impact on the overall space complexity.

### Edge Cases and Limitations

When only one time point is provided, the method adjusts by expanding the time interval to allow for meaningful integration. The segmentation scheme helps handle long-duration propagation by breaking down the problem into smaller sections, improving accuracy while controlling computation time. However, the method may struggle with highly stiff systems, where changes occur rapidly over short periods, making adaptive methods with error control potentially more suitable. Furthermore, the degree of the Chebyshev polynomial must be chosen carefully to balance between computational cost and approximation accuracy.

**Conclusion:** The **Adaptive Picard-Chebyshev Iteration** method with **segmentation** and **Chebyshev polynomial approximation** is a highly accurate and efficient approach for solving ODEs, especially in scenarios like satellite motion. The method's adaptive nature, through segmentation and degree selection, ensures precise results even over long orbital periods. By using Chebyshev polynomials, the method achieves high accuracy with fewer iterations compared to traditional methods. However, it is best suited for problems with smooth dynamics, and alternative methods may be more appropriate for stiff systems or when extremely fine error control is required.