1 Some details in this Method

Consider the initial value problem:

\[
\begin{align*}
\varphi'(t) &= F(t, \varphi(t)), \quad t \in [a, b], \\
\varphi(a) &= \varphi_a. 
\end{align*}
\]  

(1.1)

where \( \varphi_a, \varphi(t) \in \mathbb{C}^n \) and \( F : \mathbb{R} \times \mathbb{C}^n \rightarrow \mathbb{C}^n \).

Let \( \{ t_n \}_{n=0}^N \) be equally spaced nodes in the interval \([a, b] \) with \( t_0 = a, t_N = b \). Let \( \{ t_{n+1} \}_j \) be the Legendre-Gauss-Lobatto nodes in the subinterval \([t_n, t_{n+1}] \) with \( t_0^0 = t_n, t_M^M = t_{n+1} \). Denote \( \Delta_n^j = t_n^{j+1} - t_n^j \).

1.1 Forward Euler Scheme

The Picard integral equation in each \([t_n, t_{n+1}] \) associated with (1.1) is

\[
\varphi(t) = \varphi(t_n) + \int_{t_n}^{t} F(s, \varphi(s))ds.
\]

(1.2)

Employing the Forward Euler Scheme on Eqn.(1.2), we can get an approximate solution \( \varphi^0(t) \):

\[
\varphi^0(t_{n+1}) = \varphi^0(t_n) + \Delta_n^j F(t_n^j, \varphi^0(t_n^j)).
\]

(1.3)

Then \( \varphi^0(t) \) and \( F(t, \varphi(t)) \) can be written in the linear combination of Legendre polynomials:

\[
\varphi^0(t) = \sum_{j=0}^{M} \varphi_j L_j(t), \quad t \in [t_n, t_{n+1}].
\]

(1.4a)

\[
F(t, \varphi^0(t)) = \sum_{j=0}^{M} f_j L_j(t), \quad t \in [t_n, t_{n+1}].
\]

(1.4b)
where the coefficients \( \{ \phi_j \}_{j=0}^M \) and \( \{ f_j \}_{j=0}^M \) can be computed by the Legendre transform.

The residual function is

\[
\varepsilon(t) = \phi(t_n) + \int_{t_n}^{t} F(s, \phi^0(s)) ds - \phi^0(t). \tag{1.5}
\]

Let the Integral Matrix be \( S \), then Eqn.(1.5) can be rewritten as the vector version:

\[
\bar{\varepsilon}(t^j_n) = \phi(t_n) + \frac{t_{n+1} - t_n}{2} S \times \bar{f}_j - \bar{\phi}^0(t^j_n). \tag{1.6}
\]

where \( S = (s_{ji}) \), \( s_{ji} = \int_{x_j}^{x_{j+1}} L_i(x) dx \). Note that \( \int_{t_n}^{t_{n+1}} L_i(t) dt = \frac{t_{n+1} - t_n}{2} \int_{-1}^{1} L_i(x) dx \).

Let \( \delta(t) = \phi(t) - \phi^0(t) \), then

\[
\delta(t) = \varepsilon(t) + \int_{t_n}^{t} \left( F(s, \phi^0(s) + \delta(s)) - F(s, \phi^0(s)) \right) ds. \tag{1.7}
\]

Similarly as (1.3), the solution of Eqn.(1.7) is given by

\[
\delta(t^j_{n+1}) = \delta(t^j_n) + \Delta^j_n G(t^j_n, \delta(t^j_n)) + (\varepsilon(t^j_{n+1}) - \varepsilon(t^j_n)), \tag{1.8}
\]

where \( G(t, \delta(t)) = F(t, \phi^0(t) + \delta(t)) - F(t, \phi^0(t)) \).

Then we can do the correction

\[
\phi^1(t) = \phi^0(t) + \delta(t). \tag{1.9}
\]

Now we can let \( \phi^0(t) = \phi^1(t) \) in Eqn.(1.4) and do the iteration.

In the numerical experiments, the stop criterion for the iteration is that

\[
\| \varepsilon(t^M_n) \|_\infty < tol, \tag{1.10}
\]

where \( tol \) is the preset tolerance error.

By the way, I compute the Integral Matrix \( S \) in Matlab by the recursive relation of Legendre polynomials and the tool function \( polyval.m \).

### 1.2 Semi-Implicit Scheme

If \( F(t, \phi(t)) \) in Eqn.(1.1) can be written as the sum of linear part and nonlinear part,

\[
F(t, \phi(t)) = L\phi(t) + N(t, \phi(t)), \tag{1.11}
\]

then the Semi-Implicit Scheme can be employed to solve \( \phi^0(t) \) and \( \delta(t) \).
Note that usually the linear part $L\varphi$ is a stiff term while the nonlinear part $N(t, \varphi(t))$ is a non-stiff term.

The Picard integral equation in each $[t_n, t_{n+1}]$ associated with (1.1) becomes

$$\varphi(t) = \varphi(t_n) + \int_{t_n}^{t} (L\varphi(s) + N(s, \varphi(s))) \, ds. \quad (1.12)$$

Employing the Semi-Implicit Scheme on (1.12), we can get an approximate solution $\varphi^0(t)$:

$$\varphi^0(t_{n+1}) = \varphi^0(t_n) + \Delta_n^j \left( L\varphi^0(t_n) + N(t_n, \varphi^0(t_n)) \right)$$

$$\Rightarrow \quad (1 - \Delta_n^j L) \varphi^0(t_{n+1}) = \varphi^0(t_n) + \Delta_n^j N(t_n, \varphi^0(t_n)) \quad (1.13)$$

Similarly as (1.4), $\varphi^0(t)$ and $N(t, \varphi(t))$ can be written in the linear combination of Legendre polynomials:

$$\varphi^0(t) = \sum_{j=0}^{M} \varphi_j L_j(t), \quad t \in [t_n, t_{n+1}]. \quad (1.14a)$$

$$N(t, \varphi^0(t)) = \sum_{j=0}^{M} f_j L_j(t), \quad t \in [t_n, t_{n+1}]. \quad (1.14b)$$

where the coefficients $\{\varphi_j\}_{j=0}^{M}$ and $\{f_j\}_{j=0}^{M}$ can be computed by the Legendre transform.

The residual function is

$$\varepsilon(t) = \varphi(t_n) + \int_{t_n}^{t} (L\varphi^0(s) + N(s, \varphi^0(s))) \, ds - \varphi^0(t). \quad (1.15)$$

Define $S$ as in Eqn.(1.6), then (1.15) can be rewritten as the vector version:

$$\varepsilon(t_n^j) = \varphi(t_n) + \frac{t_{n+1} - t_n}{2} S \times (L\varphi_j + \bar{f}_j) - \varphi_j^0(t_n). \quad (1.16)$$

Let $\delta(t) = \varphi(t) - \varphi^0(t)$, then

$$\delta(t) = \varepsilon(t) + \int_{t_n}^{t} (L\delta(s) + N(s, \varphi^0(s) + \delta(s)) - N(s, \varphi^0(s))) \, ds. \quad (1.17)$$

Similarly as (1.13), the solution of Eqn.(1.17) is given by

$$\delta(t_{n+1}^j) = \delta(t_n^j) + \Delta_n^j L\delta(t_{n+1}^j) + \Delta_n^j G(t_n^j, \delta(t_n^j)) + (\varepsilon(t_{n+1}^j) - \varepsilon(t_n^j))$$

$$\Rightarrow \quad (1 - \Delta_n^j L) \delta(t_{n+1}^j) = \delta(t_n^j) + \Delta_n^j G(t_n^j, \delta(t_n^j)) + (\varepsilon(t_{n+1}^j) - \varepsilon(t_n^j)) \quad (1.18)$$

where $G(t, \delta(t)) = N(t, \varphi^0(t) + \delta(t)) - N(t, \varphi^0(t))$.

The correction process and stop criterion are the same as Forward Euler Scheme.
2 Numerical results of ODE

2.1 Forward Euler Scheme

Consider the common model for non-stiff problems:

\[
\begin{align*}
\text{sn}'(t) &= \text{cn}(t) \cdot \text{dn}(t), \\
\text{cn}'(t) &= -\text{sn}(t) \cdot \text{dn}(t), \\
\text{dn}'(t) &= -\mu \cdot \text{sn}(t) \cdot \text{cn}(t),
\end{align*}
\]

where \( \mu = 0.5, \; t \in [0, 1], \) the initial data is \( \text{sn}(0) = 0, \text{cn}(0) = 1, \text{dn}(0) = 1. \)

Note that the norm of error is chosen as the infinity norm:

\[
\text{Error} = \| \text{U}_{\text{exact}} - \text{U}_{\text{numerical}} \|_{\infty}.
\]

2.1.1 The spectral accuracy with \( M \) (the number of Gaussian nodes)

Fig.1 shows that if \( N \) is fixed, the error will decay exponentially as the \( M \) increasing.

![Figure 1: Take various \( M \) and fix \( N = 2, tol = 1e - 15 \)](image-url)
Table 1: Compare the tolerance and iteration when $N = 4, M = 8$

<table>
<thead>
<tr>
<th>Tolerance</th>
<th>Error</th>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0e-03</td>
<td>5.0896e-05</td>
<td>3</td>
</tr>
<tr>
<td>1.0e-05</td>
<td>5.4263e-07</td>
<td>4</td>
</tr>
<tr>
<td>1.0e-07</td>
<td>7.2637e-09</td>
<td>6</td>
</tr>
<tr>
<td>1.0e-09</td>
<td>1.9666e-11</td>
<td>7</td>
</tr>
<tr>
<td>1.0e-11</td>
<td>3.5350e-13</td>
<td>9</td>
</tr>
<tr>
<td>1.0e-13</td>
<td>1.9984e-15</td>
<td>10</td>
</tr>
<tr>
<td>1.0e-15</td>
<td>1.1102e-16</td>
<td>11</td>
</tr>
</tbody>
</table>

2.1.2 The algebraic accuracy with $N$ (the number of time step in Euler)

Fig. 2 shows that if $M$ is fixed, the error will be the typically algebraic diminishing.

Figure 2: Take various $N$ and fix $M = 4, tol = 1e - 15$

2.1.3 The relation between tolerance error and iteration times

Table 1 illustrates the comparison of the tolerance error, the error of numerical solution and the iteration times, which suggests that we can obtain high actuary by setting the small tolerance error of the iteration even if both of the $N$ and $M$ are not large.
Table 2: Compare the results of SDM and RK4

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Error of SDM</th>
<th>Time of SDM</th>
<th>Error of RK4</th>
<th>Time of RK4</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1.0559e-11</td>
<td>0.0176</td>
<td>1.0067e-07</td>
<td>0.0028</td>
</tr>
<tr>
<td>25</td>
<td>1.3323e-15</td>
<td>0.0113</td>
<td>1.7077e-08</td>
<td>0.0015</td>
</tr>
<tr>
<td>36</td>
<td>0.0000e-00</td>
<td>0.0143</td>
<td>3.9947e-09</td>
<td>0.0019</td>
</tr>
<tr>
<td>49</td>
<td>1.1102e-16</td>
<td>0.0174</td>
<td>1.1678e-09</td>
<td>0.0025</td>
</tr>
<tr>
<td>64</td>
<td>2.2204e-16</td>
<td>0.0194</td>
<td>4.0216e-10</td>
<td>0.0031</td>
</tr>
<tr>
<td>81</td>
<td>2.2204e-16</td>
<td>0.0245</td>
<td>1.5697e-10</td>
<td>0.0039</td>
</tr>
<tr>
<td>100</td>
<td>2.2204e-16</td>
<td>0.0260</td>
<td>6.7642e-11</td>
<td>0.0044</td>
</tr>
<tr>
<td>225</td>
<td>3.3307e-16</td>
<td>0.0467</td>
<td>2.6458e-12</td>
<td>0.0091</td>
</tr>
<tr>
<td>400</td>
<td>1.1102e-16</td>
<td>0.0769</td>
<td>2.6390e-13</td>
<td>0.0165</td>
</tr>
<tr>
<td>900</td>
<td>2.2204e-16</td>
<td>0.1396</td>
<td>8.8818e-15</td>
<td>0.0384</td>
</tr>
</tbody>
</table>

2.1.4 Compare with 4 order Runge-Kutta method

Table 2 illustrates the comparison of the SDM (Spectral Deferred Method) and RK4 (4 order Runge-Kutta method). In SDM, $N = M = [4 : 30]$ while in RK4, the time step is chosen as $N \times M$ in order to use the same number of total nodes as SDM. The tolerance error in SDM is chosen as $tol = 1e^{-15}$.

From Table 2, we can see that SPD method can achieve high accuracy ($\approx$ tolerance error) just using relative few nodes while RK4 has to use more nodes in order to get the accuracy of $1e^{-15}$. Besides, the computational cost of both methods are acceptable.

2.2 Semi-Implicit Scheme

Consider the nonlinear problem:

$$\begin{align*}
   x'(t) &= x + x^2 + e^{2t} - e^{4t}, \\
   x(0) &= 1.
\end{align*}$$

(2.2)

It has exact solution as follows

$$x(t) = e^{2t}.$$ 

The FESDC (Forward Euler Spectral Deferred Correction method) and SISDC (Semi-Implicit Spectral Deferred Correction method) are employed to get the numerical results respectively. Note that in Table 3, $N = 10$, and $T = 1$.

Since both of the Forward Euler and Semi-Implicit are of first-order accuracy, the errors in Table 3 is almost the same. I choose this example just to verify my code.
Table 3: Compare the error and CPU time between the SDC and SISDC

<table>
<thead>
<tr>
<th>M</th>
<th>SDC error</th>
<th>SDC time</th>
<th>SISDC error</th>
<th>SISDC time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7.4099e-010</td>
<td>0.0211</td>
<td>7.4330e-010</td>
<td>0.0119</td>
</tr>
<tr>
<td>6</td>
<td>5.1603e-013</td>
<td>0.0166</td>
<td>9.7700e-014</td>
<td>0.0132</td>
</tr>
<tr>
<td>8</td>
<td>2.6734e-013</td>
<td>0.0178</td>
<td>1.6964e-013</td>
<td>0.0149</td>
</tr>
<tr>
<td>10</td>
<td>8.8818e-015</td>
<td>0.0208</td>
<td>3.7215e-013</td>
<td>0.0175</td>
</tr>
<tr>
<td>12</td>
<td>3.5527e-014</td>
<td>0.0237</td>
<td>2.3714e-013</td>
<td>0.0193</td>
</tr>
<tr>
<td>14</td>
<td>1.5721e-014</td>
<td>0.0234</td>
<td>1.6875e-013</td>
<td>0.0205</td>
</tr>
<tr>
<td>16</td>
<td>3.4195e-014</td>
<td>0.0292</td>
<td>1.6964e-013</td>
<td>0.0240</td>
</tr>
<tr>
<td>18</td>
<td>6.5192e-013</td>
<td>0.0297</td>
<td>9.9476e-014</td>
<td>0.0274</td>
</tr>
<tr>
<td>20</td>
<td>3.0198e-014</td>
<td>0.0338</td>
<td>1.7053e-013</td>
<td>0.0259</td>
</tr>
</tbody>
</table>

3 SISDC and RSC for Burger’s-Huxley equation

Consider the generalized Burger’s-Huxley equation of the form:

\[
\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} - \alpha u \frac{\partial u}{\partial x} + \beta u \left(1 - u^\delta\right) \left(u^\delta - \gamma\right),
\]

\[
D \equiv \{(x, t) \in [a, b] \times (0, T)\},
\]

\[
u(x, 0) = u_0(x), \quad x \in [a, b],
\]

\[
u(a, t) = u_a(t), \quad u(b, t) = u_b(t),
\]

where \(\alpha, \beta, \gamma, \delta, \varepsilon\) are parameters and \(\beta \geq 0, \delta > 0, \gamma \in (0, 1), \varepsilon > 0\).

The RSC (Rational Spectral Collocation method) is employed in the spatial direction while the SISDC is used to discretize the time variable.

3.1 No boundary layer case

Let \(\varepsilon = 1, [a, b] = [0, 1]\), the initial and boundary conditions are given by

\[
u(x, 0) = \left(\frac{\gamma}{2} + \frac{\gamma}{2} \tanh(A_1 x)\right)^{1/\delta},
\]

\[
u(0, t) = \left(\frac{\gamma}{2} + \frac{\gamma}{2} \tanh(-A_1 A_2 t)\right)^{1/\delta},
\]

\[
u(1, t) = \left(\frac{\gamma}{2} + \frac{\gamma}{2} \tanh(A_1 (1 - A_2 t))\right)^{1/\delta}.
\]

The exact solution is

\[
u(x, t) = \left(\frac{\gamma}{2} + \frac{\gamma}{2} \tanh(A_1 (x - A_2 t))\right)^{1/\delta},
\]
where

\[
A_1 = \frac{-\alpha \delta + \delta \sqrt{\alpha^2 + 4\beta(1 - \delta)}}{4(1 + \delta)},
\]

\[
A_2 = \frac{\gamma \alpha}{1 + \delta} - \frac{(1 + \delta - \gamma)(-\alpha + \delta \sqrt{\alpha^2 + 4\beta(1 - \delta)})}{2(1 + \delta)}.
\]