Affine Combination of Splitting Type Integrators, Implemented with Parallel Computing Methods

Adrian Alvarez, Diego Rial

Abstract—In this work we present a family of new convergent type methods splitting high order no negative steps feature that allows your application to irreversible problems. Performing affine combinations consist of results obtained with Trotter Lie integrators of different steps. Some examples where applied symplectic compared with methods, in particular a pair of differential equations semilinear. The number of basic integrations required is comparable compared with methods, in particular a pair of differential equations allows your application to irreversible problems. Performing affine combination of splitting type integrators, Irreversible Problems, Splitting methods without negative steps, MPI, HPC.

Keywords—Lie Trotter integrators, Irreversible Problems, Splitting Methods without negative steps, MPI, HPC.

I. INTRODUCTION

We consider the autonomous problem:

\[ \dot{x} = f_0(x) + f_1(x), \quad x(0) = x_0. \]  

With \( f_0 \) is a closed densely defined operator \( D(f_0) \subset H \), a Hilbert space, which generates a strongly continuous semigroup of operators.

The nonlinear term \( f_1 : H \rightarrow H \) is a smooth application with \( f_1(0) = 0 \).

In general numerical integration methods based on temporary partitions, splitting methods, take advantage of the ability to easily solve partial problems, finding approximate solutions of the problem (1) alternately applying the partial flows \( \phi_j \) with \( j = 0.1 \) associated with each subproblem:

\[ u_t = u_0, u_t = f_1(u) \]

II. BRIEF HISTORY

The first and simplest methods are proposed in [1] Lie-Trotter and [2] Strang given by:

\[ \phi_L^1 = \phi_1 - \frac{1}{h} \phi_j(h), \quad \phi_S^1 = \phi_j(h/2) \phi_{1-j}(h) \phi_j(h/2). \]

We say that a method \( \hat{\phi} \) is of order \( q \) sii local truncation error for \( 0 < h < h_0 \) holds:

\[ |\phi(h, x_0) - \hat{\phi}(h, x_0)| \leq C(f_0, f_1, x_0) h^{q+1} \]

where \( \phi \) is the flux equation (1). It is known that \( \phi_L \) is of order 1 and \( \phi_S \) of order 2. In [6], symplectic integrators how arise:

\[ \phi_{n,b}^j(h) = \phi_1(b_m h) \circ \phi_j(a_m h) \circ \cdots \circ \phi_1(b_1 h) \circ \phi_j(a_1 h) \]

Ruth gets in ( [8] ) A symplectic integrator \( \phi_{S,a}^j \) of third order, later Neri shown in [7] be a symplectic integrator \( \phi_{S,b}^j \) of fourth order. In [6], Yoshida presents a systematic way to obtain any even-order integrators, from Baker formula - Campbell - Hausdorff. We see [5] that \( q > 2 \) is necessary that some step is negative, which prevents its application in cases irreversible. Our goal is to provide stable high-order methods for such problems.

III. THE RELATED METHODS

Given the flow \( \phi_j \) associated with partial problems, we define applications:

\[ \phi^+(h) = \phi_1(h) \circ \phi_0(h), \quad \phi^-(h) = \phi_0(h) \circ \phi_1(h) \]

\[ \phi_{m}^+(h) = \phi^+(h) \circ \phi_{m-1}^+(h) \]

\[ \Phi(h) = \sum_{m=1}^{n} \gamma_m \phi_m^+(h/m) \] (asymetric),

\[ \Phi(h) = \sum_{m=1}^{n} \gamma_m (\phi_m^+(h/m) + \phi_m^-(h/m)) \] (symetric).

Reference [9] tested the consistency and stability of these methods, given by integrators given in (2a) and (2b) are convergent with order \( q \) where \( 2n = q \), if the coefficients \( \gamma_m \) verified respectively:

\[ 1 = \gamma_1 + \gamma_2 + \cdots + \gamma_s, \]

\[ 0 = \gamma_1 + 2^{-k} \gamma_2 + \cdots + s^{-k} \gamma_s, \]

\[ 1 \leq k \leq q - 1, \]
The above calculation to find the scalar necessary for integrators proposed family of related methods $\phi_{A,q}(h)$, is a method of order $q$, where:

1. If $q = 2$, $\gamma_1 = 1/2$.
2. If $q = 3$, $\gamma_1 = -1/6$, $\gamma_2 = 2/3$.
3. If $q = 4$, $\gamma_1 = -1/16$, $\gamma_2 = 0$, $\gamma_3 = 9/16$.
4. If $q = 5$, $\gamma_1 = 1/90$, $\gamma_2 = -2/9$, $\gamma_3 = 0$, $\gamma_4 = 32/45$.
5. If $q = 6$, $\gamma_1 = 1/144$, $\gamma_2 = -8/63$, $\gamma_3 = 4/63$, $\gamma_4 = 0$, $\gamma_5 = 0.125/336$, $\gamma_6 = 3.6/3780$.
6. If $q = 7$, $\gamma_1 = -1/1680$, $\gamma_2 = 1/15$, $\gamma_3 = -27/80$, $\gamma_4 = 0$, $\gamma_5 = 27/35$.
7. If $q = 8$, $\gamma_1 = -1/2560$, $\gamma_2 = 0$, $\gamma_3 = 0.45/14112$, $\gamma_4 = 2.2/128700$, $\gamma_5 = 8.16/4790016$, $\gamma_6 = 0$, $\gamma_7 = 117649/172800$.

IV. COMPARATIVE ANALYSIS

In [9] are calculated coefficients related to low orders. Table I shows the number of steps for different methods, the advantage of the

\[ \frac{S}{P} = \frac{2q}{q} - 2 \]

1 ≤ $k$ ≤ $n - 1$.

The technique a shared memory architecture using the API OpenMP, with the proposal is to use a distributed memory architecture, for which it becomes necessary using a cluster by the MPI interface proposed technical standard with Master Worker, Fig. 1, see [4] applied as outlined below, so as to provide scalability to the model, using the Master to maintain synchrony and also as Worker.

To compare with the serial implementation the Amdahl’s Law is taken:

The algorithm that decides the speed improvement

Increasing the number of processors from a value not reflected significant progress. This fact tells us that does not always serve the parallelization, so we show cases where we observed that improves significantly. It defines the Speed up $a$: $S_p = \frac{T_S}{T_T}$, in terms of efficiency it can be seen as: $E_p = \frac{S_p}{p}$, parallelizable algorithm fraction $\alpha$. Taking that:

\[ T_T = T_S(1 - \alpha) + \frac{T_S}{p}, \alpha \in [0, 1] \]

With what fraction will pay for parallelized:

\[ S_p \leq \frac{p}{p(1 - \alpha) + \alpha} \]

V. COMPUTATIONAL IMPLEMENTATION OF PARALLELING

To illustrate see the idea in case $q = 4$, it can implement a shared memory architecture using the API OpenMP, with the technique fork join as developed by [4]. To display the performance of the proposed methods compared the results with those obtained using symplectic integrators and applications exhibit 4 order using distributed memory clusters.

\[ \frac{1}{2} = \gamma_1 + \gamma_2 + \cdots + \gamma_s, \]

\[ 0 = \gamma_1 + 2^{-2k}\gamma_2 + \cdots + s^{-2k}\gamma_s, \]

\[ 1 \leq k \leq n - 1, \]

\]  

In the implementation for cases $q > 4$, you must increase the number of threads accordingly, will increase the parallelism of the application and the number of processors, the proposal is to use a distributed memory architecture, for which it becomes necessary using a cluster by the MPI interface proposed technical standard with Master Worker, Fig. 1, see [4] applied as outlined below, so as to provide scalability to the model, using the Master to maintain synchrony and also as Worker.

The linear flow is a clockwise rotation, the orbits are indicated in Fig. 3, by circles.

<table>
<thead>
<tr>
<th>Method</th>
<th>Order (q)</th>
<th>Number of steps S</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler–Trotter</td>
<td>1</td>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>Strang</td>
<td>2</td>
<td>3</td>
<td>–</td>
</tr>
<tr>
<td>Related 2</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Ruth</td>
<td>4</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>Related 3</td>
<td>5</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>Neri</td>
<td>6</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Related 4</td>
<td>7</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>Related 5</td>
<td>8</td>
<td>28</td>
<td>8</td>
</tr>
<tr>
<td>Yoshida 6</td>
<td>9</td>
<td>32</td>
<td>10</td>
</tr>
<tr>
<td>Related 6</td>
<td>10</td>
<td>48</td>
<td>12</td>
</tr>
<tr>
<td>Related 7</td>
<td>11</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>Related 8</td>
<td>12</td>
<td>52</td>
<td>14</td>
</tr>
</tbody>
</table>

In the implementation for cases $q > 4$, you must increase the number of threads accordingly, will increase the parallelism of the application and the number of processors, the proposal is to use a distributed memory architecture, for which it becomes necessary using a cluster by the MPI interface proposed technical standard with Master Worker, Fig. 1, see [4] applied as outlined below, so as to provide scalability to the model, using the Master to maintain synchrony and also as Worker.

To compare with the serial implementation the Amdahl’s Law is taken:

The algorithm that decides the speed improvement

Increasing the number of processors from a value not reflected significant progress. This fact tells us that does not always serve the parallelization, so we show cases where we observed that improves significantly. It defines the Speed up $a$: $S_p = \frac{T_S}{T_T}$, in terms of efficiency it can be seen as: $E_p = \frac{S_p}{p}$, parallelizable algorithm fraction $\alpha$. Taking that:

\[ T_T = T_S(1 - \alpha) + \frac{T_S}{p}, \alpha \in [0, 1] \]

With what fraction will pay for parallelized:

\[ S_p \leq \frac{p}{p(1 - \alpha) + \alpha} \]
That to have a notion of computational implementation show
Fig. 2:

Fig. 2 Affine integrator module parallelized order 4

The no linear flow

\[ \dot{x}_j = -\tan(x_j) \]

whose solution is:

\[ x_j(t) = \arcsin(e^{-t}\sin(x_{j,0})) \]

Lines that converge to the origin represent the trajectories of equations. Note that the solution it is not defined for:
\[ t < \ln|\sin(x_{j,0})| \leq 0 \]

which limits the value of \( h \) for symplectic integrators. In the same graph the exact solution with initial data \((1,3/2)\) for \( t \in [0,2] \) and the points earned with \( \Phi_{A,4} \) with \( h = 0.2 \) Note that the symplectic integrator fourth order proposed Neri (1987) [7] and Yoshida (1990) [6], is not defined for that value of \( h \). Then we display an evolution conducted in parallel indicated in Fig. 4.

B. Schrödinger Cubic 2D

\[
\begin{align*}
\frac{\partial}{\partial t} |u|^2 &\quad= \left\{ u_t = i(u_{xx} + u_{yy} + |u|^2 u), \\
\left. u(0) = u_0. \right. \end{align*}
\]

\( X = (x,y) \), can be calculated using algorithm Cooley and Tukey FFT, where \( \hat{U}_s \) is the discrete Fourier coefficient. Since \( \eta \) is an integer, where the execution time required to perform FFT, it is fastest for powers of two, resulted the number of operations is \( O(\eta^2 \times \log(\eta)) \). We exhibit pseudo Dirac delta evolution in parallel (a Gaussian with very small variance \( \sigma = 0.01 \) figure 5) and showing the result to 50 sec. in Fig. 6. We consider:

\[
\hat{U}_s(S) = \frac{1}{\eta^2} \sum_{r_1=0}^{\eta-1} \sum_{r_2=0}^{\eta-1} u(2\pi/\eta(r_1,r_2)) e^{-i2\pi/\eta(r_1s_2+r_1s_2)} \]

\[
I_\eta u(X) = \sum_{s_1=-l}^{l} \sum_{s_2=-l}^{l} \hat{U}_s e^{iS.X}.
\]

The nonlinear flow take the ordinary (parametric in \( X \))

\[ u_t = i|u|^2 u \]

being that:

\[ \partial_t |u|^2 = 4\Re|u|^4 = 0 \]

Where worth

\[ |u(X,t)|^2 = |u_0(X)|^2 \]

in which the flow is given by

\[ \Phi_1(h,u_0) = e^{i2h|u_0|^2} u_0 \]
Fig. 5 Initial solution for (5) Dirac delta

Fig. 6 Evolution for (5) in 50 seconds with the parallel module 4 order for the NLS 2D $S_p = 1.2373$

VII. GOALS

In this method, be emphasizes stability and the possibility of no having negative steps give to the Related Methods unbeatable features in situations where symplectic not work. Be stable high-order methods and have advantage of to use simplicity of partial problems. The possibility of parallelism is an additional advantage that reduces the total computation time if performed on multiple processors in parallel.

In this paper, we show some figures made parallelizations making figures with Matlab in the cluster of CNEA.

The idea is to use these methods to treat cases as to analyze the dynamics of Bose - Einstein rotating based on the Gross-Pitaevskii equation 2D, irreversible problem with a term that corresponds to the angular momentum of rotation, is applying the affine method to find the minimum of the Hamiltonian by gradient descent.

ACKNOWLEDGMENTS

The authors thank to Roberto Bebilaqua, because made possible the application in the cluster of the Department of High Performance Computing CNEA. We thank Pablo Amster, the boss of DM FCEyN UBA for the support that enables the communication of this work.

REFERENCES