Fortran codes for solving Example 1 and to get Figs. 3 and 4: Symplectic Splitting method versus Chebyshev and Taylor methods*

Sergio Blanes[1†] Fernando Casas[2‡] Ander Murua[3§]

February 17, 2015

1 Numerical example

We illustrate the use of the method jointly with Taylor and Chebyshev methods on a simple example that allows to recover the results from the paper as well as to use these methods in other more challenging problems.

Example 1. The problem consists in computing $u(t) = \exp(-itH)u_0$ with

$$H = \frac{1}{2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N\times N}. \tag{1}$$

The algorithms compute $u = \mathrm{e}^{-i\alpha t}\mathrm{e}^{-i\beta t\tilde{H}}u_0$ where

$$H = \alpha I + \beta\tilde{H}, \qquad \text{where} \qquad \alpha = \frac{E_{\max}+E_{\min}}{2}, \quad \beta = \frac{E_{\max}-E_{\min}}{2} \tag{2}$$

The eigenvalues of H verify $0 \le E_k \le 2$ for all k, so that we can take $E_{\min} = 0$, $E_{\max} = 2$, and thus $\alpha = \beta = 1$ in (2). For the numerical examples we take $N = 10000$ and u_0 a unitary random vector. The matrix H is defined in the subroutine '`ProdAv.f`' which computes the products Hu. The initial conditions are stored in a separate file, '`InitialConditions.dat`, so all methods always use the same initial conditions'.

There are three methods implemented: Taylor and Chebyshev methods, and the new splitting pseudo-method. All three methods require the same inputs and are implemented in real variables. These methods compute the product

*This document has to be read jointly with Ref. [1]. We show how to obtain the results shown in Figures 3 and 4 from Example 1 in the paper.

†Email: `serblaza@imm.upv.es`

‡Email: `Fernando.Casas@uji.es`

§Email: `Ander.Murua@ehu.es`

for the scaled and shifted exponential, and the main program makes the last shift. This main program also measures the energy at the initial instant as well as after each step and the norm so, one can measure the relative error in energy and norm (unitarity). All methods return `imet`, which is the number of vector-matrix products required, which we take as the cost of the methods.

Some commands shared by all three methods like

```
beta=(Emax-Emin)/2.d0;     rhoH=beta;
alpha=(Emax+Emin)/2.d0;  shift=alpha
theta=t*beta

qs =  cos(shift*t)*q + sin(shift*t)*p
ps = -sin(shift*t)*q + cos(shift*t)*p

q=qs;   p=ps
```

appear in the main program. We very briefly review the methods.

1.1 Taylor method: TaylorEx.f

First, the algorithm checks that $\tau\beta \leq 15$. If not, the effective time is divided such this conditions is satisfied (and as close as possible to this value), and the degree of the polynomial is chosen according the formula for the error bound.

Here, w_τ is computed by *Horner's algorithm*:

$$
\begin{aligned}
&y_0 = u_0 \\
&\textbf{do}\ \ k = 1, m \\
&\quad y_k = u_0 - i\frac{\tau\beta}{m+1-k}\tilde{H}y_{k-1} \\
&\textbf{enddo} \\
&w_\tau = y_m
\end{aligned}
\tag{3}
$$

This algorithm is computed using real variables (the real and imaginary parts are computed separately).

1.2 Chebyshev method: ChebyshevEx.f

The algorithm starts by computing the degree of the polynomial necessary to ensure an error bound below `tol`. Next, it computes the Bessel functions, $J_i \equiv J_i(\tau\,\beta)$, necessary for the algorithm. We found that the function '`dbesJn(ic,theta)`' defined in fortran produced large round off errors when $i \sim \tau\,\beta > 100$, which are usual values for the Chebyhsev method, and we decided to write our own stable routine. Then the *Clenshaw recursive algorithm*:

$$
\begin{aligned}
&d_{m+2} = 0, \qquad d_{m+1} = 0 \\
&\textbf{do}\ \ i = m, 0 \\
&\quad d_i = c_i u_0 + 2\tilde{H}d_{i+1} - d_{i+2} \\
&\textbf{enddo} \\
&u_1 \equiv P^C_{m-1}(\tau\beta\tilde{H})\,u_0 = d_0 - d_2
\end{aligned}
$$

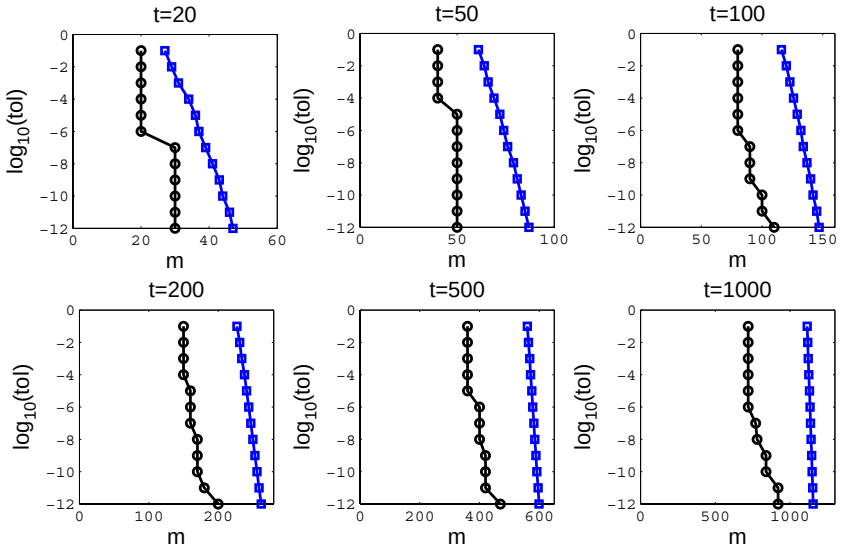

Figure 1: Degree m of the polynomials to achieve tolerances $\texttt{tol} = 10^{-k}$, $k = 1, 2, \ldots, 12$ for different values of $T\beta$ ($\beta = 1$ for this problem) as determined by the error bound formulas using the Chebyshev method (squares) and the algorithm based on splitting methods (circles).

is written in real variables as follows: $v_k = \mathrm{Re}(d_k)$ and $w_k = \mathrm{Im}(d_k)$

$$
\begin{array}{l}
v_{m+1} = 0, \qquad w_{m+1} = 0, \qquad v_m = 0, \qquad w_m = 0 \\
\textbf{do} \;\; i = m-1, 0 \\
\quad \textbf{if} \;\; \left(\lfloor i/2 \rfloor = \lfloor (i+1)/2 \rfloor \right) \qquad (\%\ if \;\; i \;\; is\ even\ \%) \\
\qquad call\ ProdAv(w_{i+1}, z) \\
\qquad v_i = (-1)^{i/2} J_i q_0 + 2z - v_{i+2} \\
\qquad call\ ProdAv(v_{i+1}, z) \\
\qquad w_i = (-1)^{i/2} J_i p_0 + 2z - w_{i+2} \\
\quad \textbf{else} \\
\qquad call\ ProdAv(w_{i+1}, z) \\
\qquad v_i = (-1)^{(i-1)/2} J_i p_0 + 2z - v_{i+2} \\
\qquad call\ ProdAv(v_{i+1}, z) \\
\qquad w_i = -(-1)^{(i-1)/2} J_i q_0 + 2z - w_{i+2} \\
\quad \textbf{endif} \\
\textbf{enddo} \\
u_1 = (q_1, p_1) \equiv P_{m-1}^C(\tau H)\, u_0 = (v_0 - v_2, w_0 - w_2)
\end{array}
$$

This algorithm needs seven real vectors in memory $(z, v_i, v_{i+1}, v_{i+2}, w_i, w_{i+1}, w_{i+2})$ (4 complex vectors using the algorithm in complex variables).

The program `MainExFigs34.f` computes the cost for different values of the tolerance as well as the error in energy and unitarity for all three methods. We compute the cost of the methods for different values of the tolerance. In particular, we take $\texttt{tol} = 10^{-k}$, $k = 1, 2, \ldots, 15$ and final integration times $T = 20, 50, 100, 200, 500, 1000$. Figure 1 (Figs. 3 in the paper) shows the

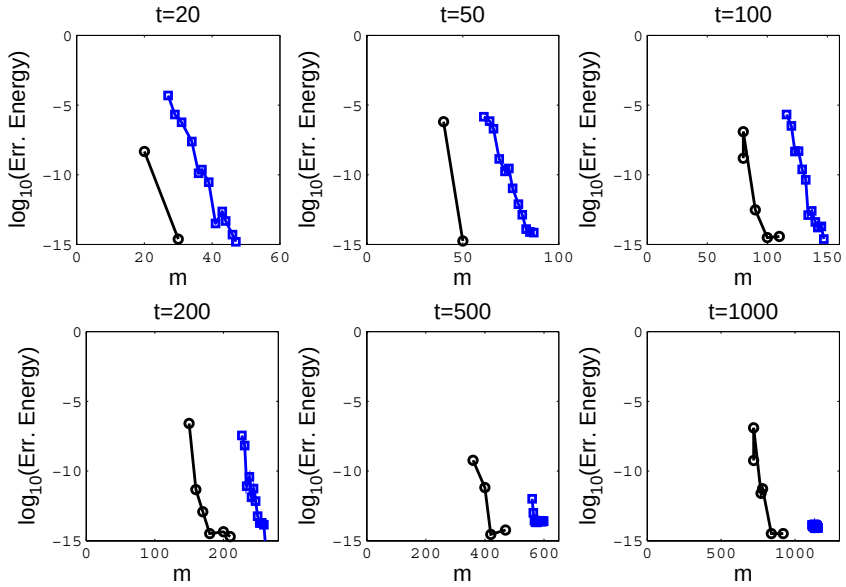

Figure 2: Same as Figure 1 but showing the relative error in energy.

results obtained with Chebyshev (line with squares) and the algorithm based on splitting schemes (line with circles) as a function of m. Even when high accuracy is required over long integration times (the most advantageous situation for Chebyshev approximations), the new algorithm requires a smaller value of m and therefore less computational effort. Notice how the algorithm selects the value of m to achieve the desired tolerance.

Figure 2 (Figs. 4 in the paper) shows the corresponding results for the relative error in energy versus m for the same example. Similar results are obtained for the error in unitarity or the two-norm error for which the error bounds apply (in this case one should compute numerically the exact solution and compare with the approximations obtained for each value of `tol`).

The results can be obtained by running the fortran file

`MainExFigs.f`

This program generate all data files for all tolerances and final times for all methods (Chebyshev, Taylor and Splitting). The Matlab files

`FigEx1a.m, FigEx1aa.m`

read all data and plot the figures (the results from Taylor are computed, but not plotted because they show a worse performance).

References

[1] S. Blanes, F. Casas, and A. Murua. An efficient algorithm based on splitting for the time integration of the Schrödinger equation. 2015. Submitted. Axiv.....