Symplectic splitting operator methods for the time-dependent Schrödinger equation

Sergio Blanes*
Instituto de Matemática Multidisciplinar, Universitat Politécnica de Valencia, E-46022 Valencia, Spain.

Fernando Casas†
Departament de Matemàtiques, Universitat Jaume I, E-12071 Castellón, Spain.

Ander Murua‡
Konputazio Zientziak eta A.A. saila, Informatika Fakultatea, EHU/UPV, Donostia/San Sebastián, Spain.
(Dated: April 3, 2006)

We present a family of symplectic splitting methods especially tailored to solve numerically the time-dependent Schrödinger equation. When discretised in time, this equation can be recast in the form of a classical Hamiltonian system with a Hamiltonian function corresponding to a generalized high-dimensional separable harmonic oscillator. The structure of the system allows us to build highly efficient symplectic integrators at any order. The new methods are accurate, easy to implement and very stable in comparison with other standard symplectic integrators.

PACS numbers:

I. INTRODUCTION

For understanding the basic atomic and molecular phenomena, the quantum mechanical treatment of molecular processes plays an essential role. This requires, in general, to solve the time-dependent Schrödinger equation ($\hbar = 1$)

$$i\frac{\partial}{\partial t}\psi(x,t) = \left(-\frac{1}{2\mu}\nabla^2 + V(x)\right)\psi(x,t), \qquad (1)$$

$\psi(x,0) = \psi_0(x)$, where $\psi : \mathbb{R}^d \times \mathbb{R} \longrightarrow \mathbb{C}$ is the wave function associated with the system. We can write (1) as

$$i\frac{\partial}{\partial t}\psi = (T(P) + V(X))\,\psi, \qquad (2)$$

with $T(P) = \dfrac{1}{2\mu}P^2$, and the operators X, P are defined by their actions on $\psi(x,t)$ as

$$X\psi(x,t) = x\psi(x,t), \qquad P\psi(x,t) = -i\nabla\psi(x,t). \qquad (3)$$

In most cases, the Schrödinger equation has to be solved numerically. For simplicity, let us consider the one-dimensional problem and suppose that it is defined in a given interval $x \in [x_0, x_N]$ ($\psi(x_0,t) = \psi(x_N,t) = 0$ or it has periodic boundary conditions). A common procedure consists in taking first a discrete spatial representation of the wave function $\psi(x,t)$: the interval is split in N parts of length $\Delta x = (x_N - x_0)/N$ and the vector $\mathbf{u} = (u_0, \ldots, u_{N-1})^T \in \mathbb{C}^N$ is formed, with $u_n = \psi(x_n, t)$ and $x_n = x_0 + n\Delta x$, $n = 0, 1, \ldots, N-1$. The partial differential equation (1) is then replaced by the N-dimensional linear ODE

$$i\frac{d}{dt}\mathbf{u}(t) = \mathbf{H}\,\mathbf{u}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0 \in \mathbb{C}^N, \qquad (4)$$

where $\mathbf{H} \in \mathbb{R}^{N\times N}$ represents the (in general Hermitian) matrix associated with the Hamiltonian [1]. The formal solution of equation (4) is given by $\mathbf{u}(t) = \mathrm{e}^{-it\mathbf{H}}\mathbf{u}_0$, but to exponentiate this $N \times N$ complex and full matrix can be prohibitively expensive for large values of N, so in practice other methods are preferred.

In general $\mathbf{H} = \mathbf{T} + \mathbf{V}$, where $\mathbf{V}$ is a diagonal matrix associated with the potential energy V and $\mathbf{T}$ is a full matrix related to the kinetic energy T. Their action on the wave function vector is obtained as follows. Since the potential operator is local in this representation, $(\mathbf{Vu})_n = V(x_n)u_n$ and thus the product $\mathbf{Vu}$ requires to compute N complex multiplications. Since periodic boundary conditions are assumed, for the kinetic energy one has $\mathbf{T}\,\mathbf{u} = \mathcal{F}^{-1}\mathbf{D}_T\mathcal{F}\mathbf{u}$, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ correspond to the forward and backward discrete Fourier transform, and $\mathbf{D}_T$ is local in the momentum representation (i.e., it is a diagonal matrix). The transformation $\mathcal{F}$ from the discrete coordinate representation to the discrete momentum representation (and back) is done via the fast Fourier transform (FFT) algorithm, requiring $\mathcal{O}(N \log N)$ operations.

Since the matrix-vector product $\mathbf{Hu}$ can be computed efficiently using FFTs, Fourier methods turn into very popular algorithms to approximate the solution $\mathrm{e}^{-it\mathbf{H}}\mathbf{u}_0$ [1–3]. Thus we can enumerate, in particular, the Chebyshev scheme, the short iterative Lanczos propagator, the second order differencing scheme, methods based on Krylov subspace techniques and splitting schemes. All these methods have obviously their merits and demerits. Their performances depend on each particular problem and even the computer where they are implemented, the

*Electronic address: serblaza@imm.upv.es
†Electronic address: Fernando.Casas@uji.es
‡Electronic address: ander@si.ehu.es

accuracy desired, how frequently the output is required, the storage requirements, etc. An extensive analysis can be found in the references [1–9]. These families of methods have been thoroughly studied indeed to build efficient algorithms for the numerical solution of the Schrödinger equation.

Most Fourier methods are based on approximating the solution $\mathbf{u}(t) = \mathrm{e}^{-it\mathbf{H}}\mathbf{u}_0$ by repeated products $\mathbf{H}\mathbf{u}$. A noteworthy exception is formed by *unitary split operator* algorithms (USO for short), which instead take advantage of the usual separability $\mathbf{H} = \mathbf{T} + \mathbf{V}$. Some of the most popular USO methods are the first order Lie–Trotter and the second order leapfrog composition [1],

$$\begin{aligned} \mathrm{e}^{\tau\mathbf{H}} &= \mathrm{e}^{\tau\mathbf{T}}\,\mathrm{e}^{\tau\mathbf{V}} + \mathcal{O}(\tau^2) && (5)\\ &= \mathrm{e}^{\tau/2\mathbf{V}}\,\mathrm{e}^{\tau\mathbf{T}}\,\mathrm{e}^{\tau/2\mathbf{V}} + \mathcal{O}(\tau^3), && (6) \end{aligned}$$

for a time step $\tau = -i\Delta t$. Since the last exponential in (6) can be evaluated together with the first one in the next step both schemes require the same computational cost. Here, clearly, $\left(\mathrm{e}^{\tau\mathbf{V}}\mathbf{u}\right)_i = \mathrm{e}^{\tau V(x_i)}u_i$ and for the kinetic part one has $\mathrm{e}^{\tau\mathbf{T}}\mathbf{u} = \mathcal{F}^{-1}\mathrm{e}^{\tau\mathbf{D}_T}\mathcal{F}\mathbf{u}$. Observe that $\mathrm{e}^{\tau\mathbf{V}}$ and $\mathrm{e}^{\tau\mathbf{T}}$ are both unitary transformations, hence the name of USO for this class of schemes.

There are other ways, however, of using splitting techniques in this context. For instance, in the so-called *symplectic split operator* schemes (SSO henceforth) the evolution operator is approximated by a composition of symplectic matrices [7, 10, 11]. This is in fact the approach followed here: we propose new families of SSO methods especially adapted to the numerical integration of the Schrödinger equation. They are based on the idea of processing and are constructed with two different goals in mind: to attain maximal stability and maximal accuracy. It turns out that by adding more stages than strictly necessary for a given order, it is possible to increase the order of the methods and design new algorithms that outperform other SSO schemes previously available by several orders of magnitude with the same computational cost. What is more striking, the extra stages can also be used to enlarge the stability interval and the accuracy of processed second-order methods so that the resulting schemes are highly efficient for all practical purposes. Although the new integrators involve a large number of stages, their implementation is not difficult and several practical algorithms are presented.

The structure of the paper is the following. In section II we briefly review different families of SSO methods, with particular emphasis on integrators using the processing technique. We also present new families of schemes especially well adapted to the numerical integration of equation (4) and discuss their practical implementation. In section III we illustrate the main features of the new methods on some relevant numerical examples. Finally, section IV contains a detailed theoretical analysis which justifies the particular choice of methods in section II and their behavior in practice.

II. COMPOSITION INTEGRATORS BASED ON SYMPLECTIC SPLIT OPERATOR BASIC SCHEMES

A. Basic symplectic split operators

Usually $\mathbf{H}$ in (4) is a real symmetric matrix, so that complex vectors can be avoided by writing $\mathbf{u} = \mathbf{q} + i\,\mathbf{p}$, with $\mathbf{q}, \mathbf{p} \in \mathbb{R}^N$. Equation (4) is then equivalent to [7, 10]

$$\frac{d}{dt}\mathbf{q} = \mathbf{H}\,\mathbf{p}, \qquad \frac{d}{dt}\mathbf{p} = -\mathbf{H}\,\mathbf{q}, \qquad (7)$$

where $\mathbf{H}\,\mathbf{q}$ and $\mathbf{H}\,\mathbf{p}$ require both a real-complex FFT and its inverse. In addition, system (7) can be seen as the classical evolution equations corresponding to the Hamiltonian function $\mathcal{H} = \frac{1}{2}\mathbf{p}^T\mathbf{H}\,\mathbf{p} + \frac{1}{2}\mathbf{q}^T\mathbf{H}\,\mathbf{q}$. Clearly, one may write

$$\frac{d}{dt}\left\{\begin{array}{c}\mathbf{q}\\ \mathbf{p}\end{array}\right\} = \left(\begin{array}{cc}\mathbf{0} & \mathbf{H}\\ -\mathbf{H} & \mathbf{0}\end{array}\right)\left\{\begin{array}{c}\mathbf{q}\\ \mathbf{p}\end{array}\right\} = (\mathbf{A}+\mathbf{B})\left\{\begin{array}{c}\mathbf{q}\\ \mathbf{p}\end{array}\right\}, \quad (8)$$

with the $2N \times 2N$ matrices $\mathbf{A}$ and $\mathbf{B}$ given by

$$\mathbf{A} \equiv \left(\begin{array}{cc}\mathbf{0} & \mathbf{H}\\ \mathbf{0} & \mathbf{0}\end{array}\right), \qquad \mathbf{B} \equiv \left(\begin{array}{cc}\mathbf{0} & \mathbf{0}\\ -\mathbf{H} & \mathbf{0}\end{array}\right).$$

The evolution operator corresponding to (8) is

$$\mathbf{O}(t) = \left(\begin{array}{cc}\cos(t\mathbf{H}) & \sin(t\mathbf{H})\\ -\sin(t\mathbf{H}) & \cos(t\mathbf{H})\end{array}\right), \qquad (9)$$

which is an orthogonal and symplectic $2N \times 2N$ matrix. As before, its evaluation is computationally very expensive and thus some approximation is required. The usual procedure is to split the whole time interval into M steps of length $h = t/M$, so that $\mathbf{O}(t) = [\mathbf{O}(h)]^M$, and then approximate $\mathbf{O}(h)$ acting on the initial condition at each step.

In this respect, observe that

$$\mathrm{e}^{\mathbf{A}} = \left(\begin{array}{cc}\mathbf{I} & \mathbf{H}\\ \mathbf{0} & \mathbf{I}\end{array}\right), \qquad \mathrm{e}^{\mathbf{B}} = \left(\begin{array}{cc}\mathbf{I} & \mathbf{0}\\ -\mathbf{H} & \mathbf{I}\end{array}\right)$$

and the cost of evaluating the action of $\mathrm{e}^{\mathbf{A}}$ and $\mathrm{e}^{\mathbf{B}}$ on $\mathbf{z} = (\mathbf{q}, \mathbf{p})^T$ is essentially the cost of computing the products $\mathbf{H}\,\mathbf{p}$ and $\mathbf{H}\,\mathbf{q}$, respectively. It makes sense, then, to use splitting operator methods similar to (5) and (6), which in this setting read

$$\begin{aligned} \mathbf{O}_1(h) &= \mathrm{e}^{h\mathbf{A}}\,\mathrm{e}^{h\mathbf{B}} && (10)\\ \mathbf{O}_2(h) &= \mathrm{e}^{h\mathbf{B}/2}\,\mathrm{e}^{h\mathbf{A}}\,\mathrm{e}^{h\mathbf{B}/2}. && (11) \end{aligned}$$

Here and in the sequel $\mathbf{O}_n(h)$ denotes an approximation of order n in the time step h to the exact operator (9), i.e., $\mathbf{O}_n(h) = \mathbf{O}(h) + \mathcal{O}(h^{n+1})$. As $\mathrm{e}^{\mathbf{A}}$ and $\mathrm{e}^{\mathbf{B}}$ are symplectic matrices, these schemes are referred to as symplectic split operator (SSO) methods. Unitarity is no longer preserved by these schemes, but in any case neither the average error in energy nor the norm of the solution increase

with time, since, as shown in [12], they are conjugate to unitary methods. The mechanism that takes place here is analogous to the propagation of the error in energy for symplectic integrators in classical mechanics.

On the other hand, the splitting (8) is general, in the sense that it can always be applied, even when the Hamiltonian cannot be split into two simple parts as is the case for USO methods.

Schemes (10) and (11) (of order 1 and 2, respectively) both require four real FFTs and thus their computational cost is similar to the USO methods (5) and (6). A possible implementation of the second order scheme requiring only one evaluation of $\mathrm{e}^{\mathbf{A}}$ and $\mathrm{e}^{\mathbf{B}}$ per step is presented as Algorithm 1 in table I, with $(\mathbf{q}(t_n), \mathbf{p}(t_n)) \simeq (\mathbf{q}_{out}(t_n), \mathbf{p}_{out}(t_n)$. This procedure needs only two products per step ($\mathbf{Hp}_{n-1/2}$, $\mathbf{Hq}_n$) and storing three vectors $(\delta\mathbf{p}, \mathbf{q}_{n-1}, \mathbf{p}_{n-1})$.

If output is not frequently required, one may consider another algorithm which only needs storing two vectors. Notice that from (10) and (11) it is clear that $\mathbf{O}_2 = e^{h\mathbf{B}/2}\,\mathbf{O}_1\,e^{-h\mathbf{B}/2}$ and so, after M steps,

$$[\mathbf{O}_2]^M = \mathrm{e}^{h\mathbf{B}/2}\,[\mathbf{O}_1]^M\,\mathrm{e}^{-h\mathbf{B}/2}, \tag{12}$$

which results in Algorithm 2 of table I.

TABLE I: Two algorithms for the numerical integration of (8) using M steps of length $h = t/M$ with the symmetric second-order method (11): (i) by storing three vectors per step (Algorithm 1), and (ii) by storing two vectors with one more exponential output, i.e. applying equation (12) (Algorithm 2).

Algorithm 1	**Algorithm 2**

$\mathbf{q}_0 = \mathbf{q}(0)$
$\mathbf{p}_0 = \mathbf{p}(0)$
$\delta\mathbf{p} = \mathbf{H}\mathbf{q}_0$
do $n = 1, M$
$\quad \mathbf{p}_{n-1/2} = \mathbf{p}_{n-1} - \frac{h}{2}\delta\mathbf{p}$
$\quad \mathbf{q}_n = \mathbf{q}_{n-1} + h\mathbf{H}\mathbf{p}_{n-1/2}$
$\quad \delta\mathbf{p} = \mathbf{H}\mathbf{q}_n$
$\quad \mathbf{p}_n = \mathbf{p}_{n-1/2} - \frac{h}{2}\delta\mathbf{p}$
If (**output**) **then**
$\quad \mathbf{q}_{out}(t_n) = \mathbf{q}_n$
$\quad \mathbf{p}_{out}(t_n) = \mathbf{p}_n$
endif
enddo
	$\mathbf{q}_0 = \mathbf{q}(0)$
	$\mathbf{p}_0 = \mathbf{p}(0) + \frac{h}{2}\mathbf{H}\mathbf{q}_0$
	do $n = 1, M$
	$\quad \mathbf{p}_n = \mathbf{p}_{n-1} - h\mathbf{H}\mathbf{q}_{n-1}$
	$\quad \mathbf{q}_n = \mathbf{q}_{n-1} + h\mathbf{H}\mathbf{p}_n$
	$\quad$ **If** (**output**) **then**
	$\quad\quad \mathbf{q}_{out}(t_n) = \mathbf{q}_n$
	$\quad\quad \mathbf{p}_{out}(t_n) = \mathbf{p}_n - \frac{h}{2}\mathbf{H}\mathbf{q}_n$
	$\quad$ **endif**
	enddo

B. A brief review of composition methods

Schemes (10) and (11) have good stability properties and minimal storage requirements. In addition, they are very easy to implement and are relatively fast to compute. However, their low order of approximation makes them useless when highly accurate results are desired.

One of the simplest techniques to build symplectic methods of any desired order is by composing a basic lower order integrator with different time-steps [13–15]. Thus, given a symmetric integrator $\mathbf{O}_{2n}$ of order $2n$, the composition

$$\mathbf{O}_{2n+2}(h) = \mathbf{O}_{2n}(\alpha_1 h)\mathbf{O}_{2n}(\alpha_0 h)\mathbf{O}_{2n}(\alpha_1 h) \tag{13}$$

with $\alpha_1 = 1/(2 - 2^{1/(2n+1)})$, $\alpha_0 = 1 - 2\alpha_1$, is a symmetric integrator of order $2n+2$ for (8). The iteration can be started, for example, with $\mathbf{O}_2$ as given in (11). This recursive procedure has, however, two main drawbacks: the large number of operators involved (making the algorithm expensive) and the presence of large positive and negative coefficients ($\alpha_1 > 1$ and $\alpha_0 < -1$ for all n), generating substantial error terms.

Other algorithms are obtained by considering the following composition (which includes (13) as a particular case):

$$\mathbf{O}_n(h) = \mathbf{O}_2(\beta_k h)\,\cdots\,\mathbf{O}_2(\beta_2 h)\,\mathbf{O}_2(\beta_1 h). \tag{14}$$

The structure of the nonlinear equations to be satisfied by the β_i coefficients for attaining order n (the so-called order conditions) is more intricate, but the resulting methods are generally more efficient. Schemes up to order 10 can be found in the literature (see [16–18] and references therein).

Even more general methods, which also include the former compositions as particular cases for the system (8), have the form

$$\mathbf{O}_n(h) = \mathrm{e}^{hb_k\mathbf{B}}\,\mathrm{e}^{ha_k\mathbf{A}}\;\cdots\;\mathrm{e}^{hb_1\mathbf{B}}\,\mathrm{e}^{ha_1\mathbf{A}}. \tag{15}$$

Now the number of exponentials k (and therefore the number of coefficients $\{a_i, b_i\}_{i=1}^k$) has to be sufficiently large to solve the corresponding order conditions. In practice, only methods up to order 6 have been constructed [16, 17, 19]. Nevertheless, when $\mathbf{A}$ and/or $\mathbf{B}$ have some special structure it is possible to design more efficient schemes [17, 19–21]. This is the case, in particular, for the problem defined by (8). If we denote by $[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}$ (the commutator of $\mathbf{A}$ and $\mathbf{B}$), then it is easy to verify that $[\mathbf{A}, [\mathbf{A}, [\mathbf{A}, \mathbf{B}]]] = [\mathbf{B}, [\mathbf{B}, [\mathbf{B}, \mathbf{A}]]] = \mathbf{0}$. As a consequence, the number of order conditions for the coefficients a_i, b_i reduces drastically.

Several methods with different orders have been constructed along these lines indeed [7, 11, 22]. Of particular relevance are the schemes presented in [7], since only $k = n$ exponentials $\mathrm{e}^{ha_i\mathbf{A}}$ and $\mathrm{e}^{hb_i\mathbf{B}}$ are used to achieve order n for $n = 4, 6, 8, 10$ and 12. By contrast, in a general composition (15) the minimum number k of exponentials $\mathrm{e}^{ha_i\mathbf{A}}$ and $\mathrm{e}^{hb_i\mathbf{B}}$ (or stages) required to attain order $n = 8, 10$ is $k = 15, 31$, respectively [16, 23]. Actually, in [7] the authors also consider the case $k > n$ as well as the processing technique in order to obtain new fourth order schemes ($n = 4$) with better stability properties.

C. New schemes based on processing

It is in fact possible to design extremely efficient integrators for the general separable linear differential equation

$$\dot{\mathbf{u}} = \mathbf{M}\,\mathbf{v}, \qquad \dot{\mathbf{v}} = \mathbf{N}\,\mathbf{u} \tag{16}$$

with $\mathbf{u} \in \mathbb{R}^N$, $\mathbf{v} \in \mathbb{R}^M$ and $\mathbf{M} \in \mathbb{R}^{N\times M}$, $\mathbf{N} \in \mathbb{R}^{M\times N}$ satisfying certain assumptions [12]. Observe that the Schrödinger equation in the form (8) is a particular example of (16) with $\mathbf{u} = \mathbf{q}$, $\mathbf{v} = \mathbf{p}$, $\mathbf{M} = -\mathbf{N} = \mathbf{H}$. This class of methods are based on the idea of processing.

In principle, the processing technique allows to reduce significantly the number of stages in a method of a given order. Here, however, we take a number larger than strictly necessary to solve all the order conditions to improve the efficiency and stability of the resulting schemes. In any case, their practical implementation can be carried out by slightly modifying Algorithm 2 of table I.

In the following we summarize the main features of the schemes and refer the reader to section IV and to [12] for technical details and a comprehensive theoretical study.

The processing technique is well illustrated by the symmetric composition $\mathbf{O}_2(h)$ of (11), which, as we noticed, can be written as (12) after M steps. Thus, with a simple correction to the first order method $\mathbf{O}_1(h)$ one is able to increase the order while preserving all its properties, and this is virtually cost free.

A *processed method* has the general structure

$$\mathbf{S}(h) = \mathbf{P}(h)\,\mathbf{K}(h)\,\mathbf{P}^{-1}(h). \tag{17}$$

Here $\mathbf{K}$ is called the *kernel* and $\mathbf{P}^{-1}$, $\mathbf{P}$ are the pre- and post-processor (also called corrector), respectively. We say that $\mathbf{K}$ has effective order n if there exists $\mathbf{P}$ such that $\mathbf{S}(h) = \mathbf{O}(h) + \mathcal{O}(h^{n+1})$.

One may replace $\mathrm{e}^{h\mathbf{B}/2}$ and $\mathrm{e}^{-h\mathbf{B}/2}$ in (12) by more general operators $\mathbf{P}$ and $\mathbf{P}^{-1}$, but no higher order methods are obtained. This can be accomplished only by replacing $\mathbf{O}_1(h)$ with more general kernels [24, 25].

Here we take as kernel $\mathbf{K}$ a composition of type (15). Then methods of order $n = 2k$ can in principle be constructed, as shown in [12]. In fact, many different solutions for the coefficients a_i, b_i exist, although all of them lead to processed methods with exactly the same performance. With this fact in mind, for simplicity we consider two different types of kernel:

(i) the $(2m)$-stage symmetric composition

$$\mathbf{K}(h) = \mathrm{e}^{ha_1\mathbf{A}}\,\mathrm{e}^{hb_1\mathbf{B}}\,\cdots\,\mathrm{e}^{hb_m\mathbf{B}}\,\mathrm{e}^{ha_{m+1}\mathbf{A}}\,\mathrm{e}^{hb_m\mathbf{B}}\,\cdots \cdots\,\mathrm{e}^{hb_1\mathbf{B}}\,\mathrm{e}^{ha_1\mathbf{A}} \tag{18}$$

determined by the $2m$ coefficients a_i, b_i, $i = 1,\ldots,m$, and

(ii) the $(2m-1)$-stage symmetric composition

$$\mathbf{K}(h) = \mathrm{e}^{ha_1\mathbf{A}}\,\mathrm{e}^{hb_1\mathbf{B}}\,\cdots\,\mathrm{e}^{ha_{m-1}\mathbf{A}}\,\mathrm{e}^{hb_m\mathbf{B}}\,\mathrm{e}^{ha_{m-1}\mathbf{A}}\,\cdots \cdots\,\mathrm{e}^{hb_1\mathbf{B}}\,\mathrm{e}^{ha_1\mathbf{A}} \tag{19}$$

defined by the $2m-1$ coefficients a_i, $i = 1,\ldots,m-1$, b_i, $i = 1,\ldots,m$.

Notice that $\mathbf{A}$ and $\mathbf{B}$ play similar roles, so that they can be interchanged. Also the last exponential $\mathrm{e}^{ha_1\mathbf{A}}$ is not counted in the total number of stages because it can be concatenated with the first exponential in the next step (this is sometimes called the First-Same-As-Last (FSAL) property [26]).

Usually, the efficiency of an integrator is characterized by its accuracy (in terms of some measure of the error terms) normalized by its computational cost (for instance, the number of stages). As is well known, by suitably increasing the number of stages it is possible to achieve a higher efficiency without raising the order [19, 27]. In other words, the extra cost can be used to reduce the size of the error terms. This is especially true for the problem at hand: it turns out that the improvement in efficiency takes place even when a really large number of stages is considered. Here we propose kernels with up to $2m-1 = 19$, $2m = 32$ and $2m = 38$ stages (other options are also possible), and for each kernel we select the corresponding coefficients a_i, b_i according to two different criteria. The first set of solutions is taken so as to provide methods of order $n = 10$, 16 and 20. The second set of coefficients bring highly accurate *second* order methods with an enlarged domain of stability. The resulting processed integrators are denoted by $\mathrm{P}_k n$, where k is the number of stages and n is the order. Thus, $\mathrm{P}_{38}2$ is a second order method with a kernel formed by 38 stages.

Once a kernel is built, we have to find appropriate pre- and post-processors. The kernel being symmetric, it can be shown that they may be taken as block diagonal matrices [12], i.e.,

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_1(h) & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2(h) \end{pmatrix}, \quad \mathbf{P}^{-1} = \begin{pmatrix} \mathbf{P}_2(h) & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_1(h) \end{pmatrix}$$

where $\mathbf{P}_2 = \mathbf{P}_1^{-1}$. In practice and for simplicity we may choose $\mathbf{P}_1$ and $\mathbf{P}_2$ as polynomial functions of $\mathbf{H}$,

$$\mathbf{P}_1(h) = \sum_{i=0}^{s} c_i (h\mathbf{H})^{2i}, \quad \mathbf{P}_2(h) = \sum_{i=0}^{s} d_i (h\mathbf{H})^{2i} \tag{20}$$

for a given s. In this way the constraint $\mathbf{P}_2 = \mathbf{P}_1^{-1}$ is relaxed to $\mathbf{P}_2 = \mathbf{P}_1^{-1} + \mathcal{O}(h^{2s+2})$. As a consequence, symplecticity is only preserved up to order $2s$, but the effect on the error is not propagated along the evolution [28].

Let us define the polynomial scalar function $p_1(z) = \sum_{i=0}^{s} c_i z^{2i}$. The coefficients $\mathbf{c} = (c_0,\ldots,c_s)$ are chosen to satisfy some order conditions (see section IV), whereas the coefficients d_i can be obtained by requiring that

$$p_2(z) = \sum_{i=0}^{s} d_i z^{2i} = \frac{1}{p_1(z)} + \mathcal{O}(h^{2s+2}). \tag{21}$$

In section IV we list the c_i coefficients corresponding to the method $\mathrm{P}_{38}2$.

For an efficient implementation of the resulting processed methods it is convenient to optimize the computation of the action of $\mathbf{P}_1$ and $\mathbf{P}_2$ on both $\mathbf{q}$ and $\mathbf{p}$. This can be done, for instance, with Horner's rule (to minimize the number of vector-matrix products or FFTs). Given $\mathbf{v} \in \mathbb{R}^N$ (corresponding to $\mathbf{q}$ or $\mathbf{p}$) and $\mathbf{z} = (z_0, z_1, \ldots, z_s)$ (corresponding to $\mathbf{c}$ or $\mathbf{d}$), the algorithm is given in table II as the function **Horner**.

TABLE II: Algorithms for the numerical integration of (8) using M steps of length $h = t/M$ by a processed method with kernel (15) and pre- and post-processors given by (20). We use Horner's rule optimize the evaluation of the polynomial function of $\mathbf{H}$.

Algorithm 3	**function Horner**
$\mathbf{q}_0 = \mathbf{Horner}(\mathbf{q}(0), \mathbf{d})$	
$\mathbf{p}_0 = \mathbf{Horner}(\mathbf{p}(0), \mathbf{c})$	
do $n = 1, M$	
do $i = 1, k$	**function Horner**$(\mathbf{v}, \mathbf{z})$
$\mathbf{q}_i = \mathbf{q}_{i-1} + a_i h \mathbf{H} \mathbf{p}_{i-1}$	$\mathbf{w} = z_s \mathbf{v}$
$\mathbf{p}_i = \mathbf{p}_{i-1} - b_i h \mathbf{H} \mathbf{q}_i$	**do** $i = 1, s$
enddo	$\mathbf{w} = z_{s-i}\mathbf{v} + (h\mathbf{H})^2 \mathbf{w}$
$\mathbf{q}_0 = \mathbf{q}_k$	**enddo**
$\mathbf{p}_0 = \mathbf{p}_k$	$\mathbf{v} = \mathbf{w}$
If (**output**) **then**	**end**
$\mathbf{q}_{out}(t_n) = \mathbf{Horner}(\mathbf{q}_0, \mathbf{c})$	
$\mathbf{p}_{out}(t_n) = \mathbf{Horner}(\mathbf{p}_0, \mathbf{d})$	
endif	
enddo	

Finally, the whole processed method can be used in practice as Algorithm 3 in table II. Observe that the differences with Algorithm 2 of table I are minimal and that the non-processed integrators of type (15) are included in this implementation by replacing the function **Horner** by the identity map.

The processed methods introduced here involve s products $\mathbf{H}\,\mathbf{q}$ and $\mathbf{H}\,\mathbf{p}$ for the pre-processor to start the algorithm, then $k = 2m - 1$ or $k = 2m$ products at each step for the kernel and finally s more products in the post-processor for getting output. As mentioned, the average relative error in energy remains, in general, constant and the error in phase space (in $\mathbf{q}, \mathbf{p}$) grows only linearly with time, i.e. $\mathcal{E} \sim \alpha\, t$. However, if we approximate the pre- and post-processor by a polynomial function as in (20), another source of error is introduced in the procedure. It turns out, however, that this second contribution to the error is of local character and does not grow with time [28], $\mathcal{E} \sim \beta(s)$, where $\beta(s) \to 0$ for $s \to \infty$ if one stays in the region of convergence. In this way the total error is given by

$$\mathcal{E}^{(P)} \sim \alpha\, t + \beta(s).$$

Since the value of α depends typically on the problem, the optimal choice of s in (20), namely the minimum value of s such that $\beta(s)$ does not dominate $\mathcal{E}^{(P)}$, depends both on the problem and the time interval of integration t.

On the other hand, if output is frequently required the efficiency of the algorithms can be reduced (even when $s' < s$ products are used for the post-processor). In this case, though, $\mathbf{P}(h)$ can be approximated by a linear combination of the internal stages of the kernel [12].

III. NUMERICAL EXAMPLES

To illustrate the performance of the new integrators proposed here we examine a simple problem which is frequently used as a test bench for numerical methods. Let us consider the one-dimensional Schrödinger equation (1) with the Morse potential $V(x) = D\left(1 - e^{-\alpha x}\right)^2$ as a good approximation for the study of the vibration states of a diatomic molecule [29]. The parameter D corresponds to the dissociation energy and α to the length parameter. The unperturbed system has 24 bounded states whose energy is given by

$$E_n = \left(n + \frac{1}{2}\right) w_0 - \left(n + \frac{1}{2}\right)^2 \frac{w_0^2}{4D}.$$

Here $w_0 = \alpha\sqrt{2D/\mu}$, and the ground state is described by the wave function

$$\phi(x) = \sigma \exp\big(-(\gamma - 1/2)\alpha\, x\big) \exp\big(-\gamma e^{-\alpha x}\big),$$

with $\gamma = 2D/w_0$ and σ is a normalizing constant. We take the following parameter values: $\mu = 1745$ *a.u.*, $D = 0.2251$ *a.u.* and $\alpha = 1.1741$ *a.u.*, corresponding to the HF molecule. As initial condition we take the Gaussian wave function

$$\psi(x, 0) = \rho \exp\big(-\beta(x - x_m)^2\big),$$

with $\beta = \sqrt{k\mu}/2$, $k = 2D\alpha^2$, ρ is a normalizing constant and $x_m = -1/10$.

We assume that the system is defined in the interval $x \in [-0.8, 4.32]$, which is split into $N = 64, 128$ parts of length $\Delta x = 0.08, 0.04$, and consider the finite dimensional linear equation (8) with periodic boundary conditions. In all cases we integrate along the interval $t \in [0, 20T]$, with $T = 2\pi/w_0$.

The following methods are compared:

- The second order method $\mathbf{O}_2$ of (11).
- The well known 3-stage fourth-order method ($\mathrm{YS}_3 4$) obtained from the recursion (13), and the 17-stage eighth-order method ($\mathrm{M}_{17} 8$) of type (14) whose coefficients can be found in [17, 30, 31] (very similar performance is attained with the coefficients given in [16, 23]). Both methods take $\mathbf{O}_2$ as basic scheme.
- The k-stage nth-order methods ($\mathrm{GM}_k n$) for $k = n = 4, 8, 12$ and $k = 6, n = 4$ given in [7].
- The following k-stage processed methods [34] of order n, $\mathrm{P}_k n$, built in this work: $\mathrm{P}_{19}10$, $\mathrm{P}_{32}16$, $\mathrm{P}_{38}20$, $\mathrm{P}_{19}2$, $\mathrm{P}_{32}2$ and $\mathrm{P}_{38}2$.

As a measure of the error we compare the wave functions at the final time obtained by the integrators with the exact solution by computing the norm $\|\mathbf{u}_{ex}-\mathbf{u}_{ap}\| = \sqrt{\sum_i (\mathbf{u}_{ex}-\mathbf{u}_{ap})_i^2}$, where $\mathbf{u}_{ex}$ corresponds to the exact solution (computed numerically with high accuracy) and $\mathbf{u}_{ap}$ is the numerical approximation obtained by each method. The cost is measured by the number of FFTs for different values of the time step h. In all cases the largest time step considered correspond to the stability limit of each method, i.e. the largest value of h before an overflow appears. The cost of the processed methods is measured by the cost of their kernels. All computations are carried out in fortran with double precision.

The first experiment checks the standard splitting methods appearing in the literature in order to choose those with the best performances for this particular problem. The purpose is to use them afterwards as the reference methods to which compare the new processed schemes. In figure 1 we show the results obtained in double logarithmic scale for $N = 128$. The methods considered are $\mathbf{O}_2$, $\mathrm{YS}_3 4$, $\mathrm{M}_{17}8$, $\mathrm{GM}_4 4$, $\mathrm{GM}_6 4$, $\mathrm{GM}_8 8$ and $\mathrm{GM}_{12}12$. The superiority of the integrators $\mathrm{GM}_n n$ (especially tailored for the Schrödinger equation) is manifest when accurate results are desired (the well known schemes $\mathrm{YS}_3 4$, $\mathrm{M}_{17}8$ show their asymptotic behavior only for very small time steps). Thus, in the sequel we take as reference the most stable ($\mathbf{O}_2$ and $\mathrm{GM}_6 4$) and most accurate methods (GM_{12}) (broken lines in the next figures).

It is important to remark that $\mathbf{O}_2$ has the highest stability it is possible to reach using splitting methods [32]. One must also keep in mind that the stability can also be constrained by the size of the mesh, i.e. the value of N considered in the space discretization.

The purpose of the next experiments is twofold: first, to analyze the performance of the two families of processed (low and high order) methods under different circumstances, and second, to show their superiority in comparison with the previous standard splitting methods.

Since the new processed methods require a relatively large number of stages per step, their efficiencies are frequently manifest when large time steps are considered, otherwise roundoff accuracy is usually reached, making the methods difficult to compare. To avoid the constraint involved in the mesh size, we have repeated the numerical experiments both for $N = 64$ and $N = 128$. Figure 2 shows the results obtained with $N = 128$ (left figures) and $N = 64$ (right figures) for the schemes $\mathrm{P}_{19}10$, $\mathrm{P}_{32}16$ and $\mathrm{P}_{38}20$ of order 10, 16 and 20, respectively (top figures) and the second order methods $\mathrm{P}_{19}2$, $\mathrm{P}_{32}2$ and $\mathrm{P}_{38}2$ (bottom figures).

The improvement in the performance of each family of methods when increasing the number of stages is clearly visible at the figure. This phenomenon takes place even for very large numbers of stages. Nevertheless, what it is really striking is that the second order methods reach a surprisingly high accuracy before they show their asymptotic second order behavior ($\mathrm{P}_{32}2$ nearly and $\mathrm{P}_{38}2$ always reach roundoff error before this second order slope is observed). In addition, the $\mathrm{P}_k 2$ methods show better stability properties. To clarify this fact, figure 3 shows the results obtained for $N = 128$ for the schemes $\mathbf{O}_2$, $\mathrm{GM}_6 4$, $\mathrm{GM}_{12}12$, $\mathrm{P}_{38}20$ and $\mathrm{P}_{38}2$, where the superiority of the second order method $\mathrm{P}_{38}2$ is clearly manifest. It is also noticeable that $\mathrm{P}_{38}2$ shows nearly the same stability as the scheme $\mathrm{GM}_6 4$.

IV. THEORETICAL ANALYSIS OF THE NEW PROCESSED METHODS

In this section we present the main ideas which led us to construct the methods presented in section II. The purpose is, basically, to justify the high performance shown by the new integrators and to pave the way for the more technical study carried out in [12]. Therefore, this section can be safely skipped by readers interested only as users of the methods.

The main reason which allows us to build highly efficient integrators for the Schrödinger equation resides basically in the small number of order conditions the kernel has to satisfy to attain a given order [28]. In addition, the resulting order conditions can be easily found and solved explicitly. Consequently, one is able to construct methods of any order without difficulty. Moreover, this permit us to improve the schemes in a relatively simple way just by considering additional stages in order to increase the accuracy and stability at a ratio which compensates the extra computational cost.

The nature of equation (8) allows one quite naturally to build explicit methods which only involve matrix-vector products, i.e. polynomial approximations to the solution (9). This can be efficiently achieved with a composition of basic maps as in (15) since, with a given number of products, it leads to the polynomial with the highest possible degree. At the same time symplecticity is automatically preserved. Notice that, for example, from the equality

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -bh\mathbf{H} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{I} & ah\mathbf{H} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{p} \end{Bmatrix} = \begin{pmatrix} \mathbf{I} & ah\mathbf{H} \\ -bh\mathbf{H} & \mathbf{I} - abh^2\mathbf{H}^2 \end{pmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{p} \end{Bmatrix} \tag{22}$$

a straightforward computation of the right hand side of (22) involves four products while the left hand side requires only two.

The following crucial observation greatly simplifies the analysis and construction of our methods. Since $\mathbf{H}$ is a real symmetric matrix, it is diagonalizable with real eigenvalues $\lambda_j \in \mathbb{R}$. Whence, the linear system (8) can be decoupled into N scalar harmonic oscillators of the form

$$\frac{d}{dt}q_j = \lambda_j p_j, \qquad \frac{d}{dt}p_j = -\lambda_j q_j, \quad j = 1, \ldots, N. \tag{23}$$

Moreover, any composition method (15) is invariant with respect to a linear change of variables of the form $\bar{\mathbf{q}} =$

$\mathbf{Gq}$, $\bar{\mathbf{p}} = \mathbf{Gp}$, and thus, for the theoretical analysis of (15) applied to (8), we may consider the application of the integrator (15), and more generally, the processed integrator (17), to each of the scalar harmonic oscillators (23).

Consequently, the problem of finding appropriate processed composition methods with kernel (15) for equation (8) can be reduced to the following. Find coefficients a_i, b_i in a 2×2 matrix of the form

$$\mathbf{K}(x) = \begin{pmatrix} 1 & 0 \\ -b_k x & 1 \end{pmatrix} \begin{pmatrix} 1 & a_k x \\ 0 & 1 \end{pmatrix} \cdots \cdots \begin{pmatrix} 1 & 0 \\ -b_1 x & 1 \end{pmatrix} \begin{pmatrix} 1 & a_1 x \\ 0 & 1 \end{pmatrix} \tag{24}$$

and a 2×2 matrix

$$\mathbf{P}(x) = \begin{pmatrix} P_1(x) & P_2(x) \\ P_3(x) & P_4(x) \end{pmatrix} \tag{25}$$

such that $\mathbf{S}(x) = \mathbf{P}(x)\mathbf{K}(x)\mathbf{P}(x)^{-1}$ approximates the solution matrix

$$\mathbf{O}(x) = \begin{pmatrix} \cos(x) & \sin(x) \\ -\sin(x) & \cos(x) \end{pmatrix}$$

with $x = \lambda_j t$. For our analysis, we allow the entries $P_j(x)$ of $\mathbf{P}(x)$ to be arbitrary functions, although in the practical implementation to the system (8) each $P_j(x)$ will be replaced by an appropriate polynomial approximation.

Clearly, (24) can be written in the form

$$\mathbf{K}(x) = \begin{pmatrix} K_1(x) & K_2(x) \\ K_3(x) & K_4(x) \end{pmatrix}, \tag{26}$$

where the entries $K_j(x)$ are polynomials in x. If $\mathbb{P}_{2r-1}$ and $\mathbb{P}_{2r}$ denote the set of polynomials of degrees $2r-1$ and $2r$ which only contain odd and even terms, respectively, then $K_1(x) \in \mathbb{P}_{2k-2}$, whereas $K_2(x), K_3(x) \in \mathbb{P}_{2k-1}$ and $K_4(x) \in \mathbb{P}_{2k}$. Obviously, the degree of the polynomials should be changed appropriately if, for example, $b_k = 0$, as it happens for a symmetric composition.

What makes finding good processing methods a simpler task than obtaining non-processed composition methods is the fact that, as observed in [33], both accuracy and stability essentially depend on $\operatorname{tr}\mathbf{K}(x) = K_1(x) + K_4(x)$ (notice that this is a polynomial in even powers of x). For a processed method to be stable for a given x it is required that $|\operatorname{tr}\mathbf{K}(x)| \leq 1$, and for accuracy, $\operatorname{tr}\mathbf{K}(x)$ must be a good approximation to $2\cos(x)$. In general (see [12] for details), there is an infinite number of choices for the set of coefficients $\{a_j, b_j\}$ such that, for $\mathbf{K}(x)$ in (24), $\operatorname{tr}\mathbf{K}(x)$ coincides with a prescribed even polynomial in x (appropriately chosen so that it approximates $2\cos(x)$ and has good stability properties). Obviously, for each such a choice of the set of coefficients $\{a_j, b_j\}$, a different matrix (26) will be obtained. Such different choices are seen to be essentially equivalent [12] in our context, provided that $K_2(x) = K_3(x) = 0$ whenever $|K_1(x) + K_4(x)| = 2$.

As previously mentioned and for simplicity, we restrict ourselves to symmetric methods (that is, $b_k = 0$, $a_{k-j+1} = a_j$, $b_{k-j} = b_j$). Then it is easy to check that $K_1(x) = K_4(x)$ and thus $\operatorname{tr}\mathbf{K}(x) = 2K_1(x)$. In other words, the accuracy and stability properties of the processed method essentially depend only on the polynomial $K_1(x)$. In particular, the processed integrator will have order of accuracy $n = 2q$ if

$$K_1(x) = \cos(x) + \mathcal{O}(x^{2q+1}) \quad \text{as} \quad x \to 0. \tag{27}$$

For a k-stage symmetric kernel, the polynomial $K_1(x)$ has the form

$$K_1(x) = 1 + \sum_{j=1}^{q} \alpha_j \, x^{2j} + x^{2q} \sum_{j=1}^{k-q} \alpha_j \, x^{2j},$$

where we have split the coefficients of the polynomial in two sets. The first one, $\{\alpha_1, \ldots, \alpha_q\}$ is chosen so as to ensure that the processed method has order $n = 2q$ (accuracy for *small* values of x), whereas the second set $\{\alpha_{q+1}, \ldots, \alpha_k\}$ is used to improve accuracy for large values of x *and* stability.

From these considerations it is clear that the first set is determined by conditions (27), that is,

$$\alpha_i = \frac{(-1)^i}{(2i)!}, \quad i = 1, \ldots, q.$$

With respect to the second set of coefficients, we propose to determine $\{\alpha_{q+1}, \ldots, \alpha_k\}$ by requiring that

$$K_1(j\pi) = \cos(j\pi) = (-1)^j, \quad \frac{dK_1}{dx}(j\pi) = -\sin(j\pi) = 0, \tag{28}$$

$j = 1, \ldots, l$, where $k = q + 2l$. Notice the interpolatory nature of conditions (28), which permits one, for instance, to get the coefficients by solving a linear system of equations. For every considered polynomial $K_1(x)$ of this class, we have observed that, for all $x \in [0, (l-1)\pi]$ (and in many cases for all $x \in [0, l\pi]$), the stability condition $|K_1(x)| \leq 1$ holds and $K_1(x)$ is a fairly good approximation to $\cos(x)$).

Once the polynomial $K_1(x)$ has been fixed, the next step is to determine appropriate polynomials $K_2(x)$ and $K_3(x)$ in (26). Since the determinant of each factor in (24) is 1, then $\det(\mathbf{K}(x)) \equiv 1$, i.e.,

$$K_1^2(x) - K_2(x)K_3(x) \equiv 1. \tag{29}$$

On the other hand, with conditions (28) one has

$$\mathbf{K}(j\pi) = \begin{pmatrix} (-1)^j & K_2(j\pi) \\ K_3(j\pi) & (-1)^j \end{pmatrix}$$

and since $\det(\mathbf{K}(j\pi)) = 1$, then either $K_2(j\pi) = 0$ or $K_3(j\pi) = 0$. However, unless $K_2(j\pi) = K_3(j\pi) = 0$, the

matrix $\mathbf{K}(j\pi)$ is linearly unstable (i.e., $[\mathbf{K}(j\pi)]^M$ grows linearly with M).

In consequence, we need to find a pair $(K_2(x), K_3(x))$ of polynomials in odd powers of x such that (29) holds and $K_2(j\pi) = K_3(j\pi) = 0$ for $j = 1, \ldots, l$. As shown in [12], there is a finite number of such pairs $(K_2(x), K_3(x))$ of polynomials, and in some isolated cases the corresponding $\mathbf{K}(x)$ does not admit a decomposition of the form (24). In [12] we derive a finite step-by-step constructive algorithm to obtain the coefficients a_i, b_i from the polynomials $K_i(x)$ if there exists such a decomposition for a given $\mathbf{K}(x)$ with $\det(\mathbf{K}(x)) \equiv 1$.

As for the processor, it is possible in fact to consider a block-diagonal matrix instead of the more general expression (25),

$$\mathbf{P} = \begin{pmatrix} P_1(x) & 0 \\ 0 & P_2(x) \end{pmatrix}, \ \mathbf{P}^{-1} = \begin{pmatrix} P_2(x) & 0 \\ 0 & P_1(x) \end{pmatrix},$$

with $P_1(x)P_2(x) \equiv 1$. We then choose $P_1(x)$ so that the matrix

$$\mathbf{S}(x) = \mathbf{P}\,\mathbf{K}\,\mathbf{P}^{-1} = \begin{pmatrix} K_1(x) & P_1^2(x)K_2(x) \\ P_2^2(x)K_3(x) & K_1(x) \end{pmatrix}$$

corresponding to the processed method is orthogonal. That is, $P_1^2(x)K_2(x) = -P_2^2(x)K_3(x)$, which gives at once

$$P_1(x) = \sqrt[4]{\frac{-K_3(x)}{K_2(x)}}, \quad P_2(x) = \sqrt[4]{\frac{-K_2(x)}{K_3(x)}}. \qquad (30)$$

Recall that, for the practical application to (8), $P_1(x)$ and $P_2(x)$ in (30) must be replaced by an appropriate polynomial approximation (for instance, obtained by truncating their Taylor expansion).

In our experiments we have found that in all cases the performance of the methods always improves when l increases. This is the reason why we have chosen methods with relatively large values of l ($l = 7, 12, 14$ for the kernels with 19, 32 and 38 stages, respectively). Notice that for $l = 14$ we are adding $2l = 28$ extra parameters (with respect to those required by the order of precision of the processed method).

A second family of processed methods has been also constructed, which only differs in the way $K_1(x)$ is chosen. The remaining steps are exactly the same. For this family we take $\alpha_1 = 1$ to ensure consistence (this guarantees that the methods are of second order since the kernel is symmetric) and choose $\{\alpha_2, \ldots, \alpha_q\}$ in such a way that the integral

$$\int_{-l\pi}^{l\pi} \left(1 - \left(\frac{x}{l\pi}\right)^2\right)^{-1/2} \left(\frac{K_1(x) - \cos(x)}{x^{2n+2}}\right)^2 dx$$

is minimized. Notice that this is equivalent to minimizing in the least square sense the coefficients of the Chebyshev series expansion of $(K_1(x) - \cos(x))/(x^{2n+2})$ in the interval $[-l\pi, l\pi]$.

This procedure makes sense since our aim is to construct methods which are stable in a relatively large interval, $x \in [-l\,\pi, l\,\pi]$, and are also accurate when large time steps are considered. High order methods ensure accuracy for small values of x because they approximate the Taylor series expansion of the exact solution. On the other hand, by applying Chebyshev techniques we determine kernels that, while being low order approximations for small time steps (e.g., second order), give very accurate results for values of x in the whole interval (see [12] for more details). The processed methods denoted by P_k2 in section II belong precisely to this second family of schemes.

As an illustration, in table III we show the coefficients in (18) for the second order 38-stage kernel of the method $\mathrm{P}_{38}2$. From these coefficients, and using for example *Mathematica* or *Matlab*, one can obtain, by composition, the matrix $K(x)$ in (26). Then, making use of the polynomials $K_2(x)$ and $K_3(x)$ we obtain the functions $P_1(x)$ and $P_2(x)$ for the processor as given in eq. (30). Finally, it only remains to take a Taylor series expansion up to a desired order. For the reader convenience, in table IV we collect the coefficients c_i, d_i in (20) up to $s = 21$. All these coefficients are also available from the author upon request. In spite of their surprisingly small values for the coefficients c_i, d_i we must keep in mind that we are considering $P_1(x) = \sum_{i=0}^{s} c_i x^{2i}$, where x can take relatively large values. In practice, we did not find any trouble when running our fortran codes in double precision. Obviously, we would have smaller coefficients by rewriting the polynomial as $P_1(x) = \sum_{i=0}^{s} (e_i x)^{2i}$ where, for example, $e_5 = 0.016761\ldots, e_{10} = 0.018518\ldots$ and $e_{20} = 0.019731\ldots$.

V. CONCLUSIONS

We have considered a new family of symplectic splitting methods for numerically solving the time-dependent Schrödinger equation. These methods are designed for integrating linear separable systems where the different parts in which they are split satisfy certain commutator relations. A particular case where they can be applied is precisely the Schrödinger equation when the wave function is separated in its real and imaginary parts.

The proposed methods are built using the processing technique. This procedure allows us to consider numerical schemes with as many stages as desired for optimization purposes. We have shown that this strategy leads to highly efficient methods whose relative performance depend both on the number of stages and the criterion for optimization considered.

Several algorithms for the practical implementation of the new methods are presented and their superiority in comparison with other symplectic integrators published in the literature is manifest through numerical experiments. It is especially remarkable the high performance shown by the second order processed methods.

TABLE III: Coefficients for the symmetric kernel (18) with $m = 19$, corresponding to the processed method $P_{38}2$.

$a_1 = 0.021567285179758507570535029527 8$	$b_1 = 0.043146145488108635999087625827 7$
$a_2 = 0.043172634385310163973536971499 8$	$b_2 = 0.043185323459336415208749029206 3$
$a_3 = 0.043132429779569059994991278386 02$	$b_3 = 0.042970474465098214753936388546 8$
$a_4 = 0.042785296150567532011820041940 1$	$b_4 = 0.043036430087145449924388788374 0$
$a_5 = 0.044974793077247686994863089127 5$	$b_5 = 0.053280567850892122735079878196 8$
$a_6 = 0.521477840977180737598212898081$	$b_6 = -0.000074163259065200898234960429 9511$
$a_7 = -0.460297865581209561666776462059$	$b_7 = 0.054925268504928076884600967328 2$
$a_8 = 0.047665772371778444673756470398 2$	$b_8 = 0.057292231828906343681421400831 3$
$a_9 = -0.299809415632442402707251772031$	$b_9 = -0.000216083699929765754852184048464$
$a_{10} = 0.360890555491738732398154005651$	$b_{10} = 0.042926282729985071023168967959 8$
$a_{11} = 0.035531086024797552599350571732 7$	$b_{11} = 0.050959058338225962551795708253 3$
$a_{12} = 0.045145910959192914369839685478 7$	$b_{12} = 0.012587646630311939636735292990 3$
$a_{13} = 0.151663982419594313475358779605$	$b_{13} = -0.001101436018750557512175885243 09$
$a_{14} = -0.122723981192628473398202625228$	$b_{14} = 0.058986448589350873984573566850 7$
$a_{15} = -0.034200364472280225513252392096 2$	$b_{15} = -0.003939190912103381986615777740 09$
$a_{16} = 0.051470280247056559488864327710 3$	$b_{16} = 0.090918979158864182368679156310 3$
$a_{17} = -0.003469161496833743744014917139 03$	$b_{17} = -0.107654717879545729464023522278$
$a_{18} = 0.020104643066961682381420284561 0$	$b_{18} = 0.025427811389330993619764468064 8$
$a_{19} = -0.024525127775059992631968367599 6$	$b_{19} = \frac{1}{2} - (b_1 + \cdots + b_{18})$
$a_{20} = 1 - 2(a_1 + \cdots + a_{19})$	

TABLE IV: Coefficients for the polynomials $P_1(x)$ and $P_2(x)$ in (20) with $s = 21$ for $P_{38}2$ ($c_0 = d_0 = 1$).

$c_1 = 0.000116251208684740621114 0814$	$c_2 = 3.376774894743804480444394 \cdot 10^{-8}$	$c_3 = 1.176364067599484205038903 \cdot 10^{-11}$
$c_4 = 4.437111761894176717316941 \cdot 10^{-15}$	$c_5 = 1.749973819201524252032138 \cdot 10^{-18}$	$c_6 = 7.101748878564126570715907 \cdot 10^{-22}$
$c_7 = 2.939931769324440416879823 \cdot 10^{-25}$	$c_8 = 1.235098758247133102034345 \cdot 10^{-28}$	$c_9 = 5.248386453665149303792009 \cdot 10^{-32}$
$c_{10} = 2.250866251009862206361312 \cdot 10^{-35}$	$c_{11} = 9.727578606034733795739798 \cdot 10^{-39}$	$c_{12} = 4.231641947350449068306722 \cdot 10^{-42}$
$c_{13} = 1.851409459980067426102173 \cdot 10^{-45}$	$c_{14} = 8.141553608452406208018081 \cdot 10^{-49}$	$c_{15} = 3.596667466064486029961227 \cdot 10^{-52}$
$c_{16} = 1.595498786085559337026367 \cdot 10^{-55}$	$c_{17} = 7.104576813414967870669619 \cdot 10^{-59}$	$c_{18} = 3.174598116648571190359996 \cdot 10^{-62}$
$c_{19} = 1.423077177952293495040530 \cdot 10^{-65}$	$c_{20} = 6.398117951527209690698617 \cdot 10^{-69}$	$c_{21} = 2.884478510968248948572185 \cdot 10^{-72}$
$d_1 = -0.000116251208684740621114 0814$	$d_2 = -2.025340542677493159320967 \cdot 10^{-8}$	$d_3 = -5.483616185447620695388045 \cdot 10^{-12}$
$d_4 = -1.748185395473289243875044 \cdot 10^{-15}$	$d_5 = -6.075023900031386380514259 \cdot 10^{-19}$	$d_6 = -2.227092296947007254380344 \cdot 10^{-22}$
$d_7 = -8.469091056567204221082539 \cdot 10^{-26}$	$d_8 = -3.308402509398670050765033 \cdot 10^{-29}$	$d_9 = -1.319641733480979355653975 \cdot 10^{-32}$
$d_{10} = -5.353346141747406366467657 \cdot 10^{-36}$	$d_{11} = -2.202620915392627214792992 \cdot 10^{-39}$	$d_{12} = -9.173684223172953098611281 \cdot 10^{-43}$
$d_{13} = -3.861783526343716602117122 \cdot 10^{-46}$	$d_{14} = -1.641163468907425875108297 \cdot 10^{-49}$	$d_{15} = -7.033925071359782763595843 \cdot 10^{-53}$
$d_{16} = -3.037693851132668729625454 \cdot 10^{-56}$	$d_{17} = -1.320846410906512328044568 \cdot 10^{-59}$	$d_{18} = -5.778602796374270082897366 \cdot 10^{-63}$
$d_{19} = -2.542100400250845548947583 \cdot 10^{-66}$	$d_{22} = -1.123916118043500908715140 \cdot 10^{-69}$	$d_{21} = -4.991692562368483793888509 \cdot 10^{-73}$

The coefficients a_i, b_i for the kernels in a processed method can be easily obtained with any accuracy for all cases considered in this work because they are the solution of linear equations. On the other hand, for non-processed schemes, these coefficients have to solve relatively large systems of non-linear equations, and then their numerical solution is not so straightforward. However, it could be possible that most techniques used to obtain efficient kernels could also be used to find highly efficient and stable non-processed methods, being this an interesting problem to be analyzed.

In conclusion, we claim that the processing technique leads to extraordinarily efficient symplectic split operator methods por the Schrödinger equation, and thus they deserve further analysis and study. In particular, it would be very interesting to analyze under which conditions this technique is superior to other schemes used in practice, such as the Chebyshev scheme or methods based on Krylov subspace techniques [3].

Acknowledgments

This work has been partially supported by Ministerio de Educación y Ciencia (Spain) under project MTM2004-00535 (co-financed by the ERDF of the European Union). The work of SB has also been supported by a contract in the Pogramme Ramón y Cajal 2001.

[1] M.D. Feit, J.A. Fleck, Jr, and A. Steiger, J. Comput. Phys. **47**, 412 (1982).

[2] R. Kosloff, J. Phys. Chem. **92**, 2087 (1988).

[3] C. Leforestier, R.H. Bisseling, C. Cerjam., M.D. Feit, R.

Friesner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H.-D. Meyer, N. Lipkin, O. Roncero, and R. Kosloff, J. Comput. Phys. **94**, 59 (1991).
[4] A.D. Bandrauk and Hai Shen, Chem. Phys. Lett. **176**, 428 (1991).
[5] A.D. Bandrauk and Hai Shen, J. Chem. Phys. **99**, 1185 (1993).
[6] H. Frauenkron and P. Grassberger, Int. J. Mod. Phys. C **5**, 37 (1994).
[7] S. Gray and D.E. Manolopoulos, J. Chem. Phys. **104**, 7099 (1996).
[8] D. Kosloff and R. Kosloff, J. Comp. Phys. **52**, 35 (1983).
[9] H. Tal-Ezer and R. Kosloff, J. Chem. Phys. **81**, 3967 (1984).
[10] S. Gray and J.M. Verosky, J. Chem. Phys. **100**, 5011 (1994).
[11] W. Zhu, X. Zhao, and Y. Tang, J. Chem. Phys. **104**, 2275 (1996).
[12] S. Blanes, F. Casas, and A. Murua, (*work in progress*)
[13] M. Creutz and A. Gocksch, Phys. Rev. Lett. **63**, 9 (1989).
[14] M. Suzuki, Phys. Lett. A **146**, 319 (1990).
[15] H. Yoshida, Phys. Lett. A **150**, 262 (1990).
[16] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Ser. Comput. Math. 31, Springer-Verlag, Berlin 2002.
[17] R.I. McLachlan and R.G.W. Quispel, Acta Numerica **11**, 341 (2002).
[18] M. Sophronious and G. Spaletta, Optimization Methods Software, **20**, 597 (2005).
[19] S. Blanes and P.C. Moan, J. Comp. Appl. Math. **142**, 313 (2002).
[20] S.A. Chin, Phys. Lett. A **226**, 344 (1997).
[21] I.P. Omelyan, I.M. Mryglod, and R. Folk, Phys. Rev. E **66**, 026701 (2002).
[22] X. Liu, P. Ding, J. Hong, and L. Wang, Comput. Math. Appl. **50**, 637 (2005).
[23] W. Kahan and R.C. Li, Math. Comp. **66**, 1089 (1997).
[24] R.I. McLachlan, in Integration Algorithms and Classical Mechanics, Vol. **10**, J.E. Marsden, G.W. Patrick, and W.F. Shadwick, eds., American Mathematical Society, Providence, R.I., 141 (1996)..
[25] M. Suzuki, *Phys. Lett. A* **180**, 232 (1993).
[26] J.M. Sanz-Serna and M.P. Calvo, Numerical Hamiltonian Problems, Chapman & Hall, London 1994
[27] R.I. McLachlan, Numer. Algorithms **31**, 233 (2002).
[28] S. Blanes, F. Casas, and A. Murua, SIAM J. Numer. Anal. **42**, 531 (2004).
[29] R.B. Walker and R.K. Preston, J. Chem. Phys. **67**, 2017 (1977).
[30] B. Leimkuhler and S. Reich, *Simulating Hamiltonian Dynamics*, Cambridge University Press, Cambridge 2004
[31] R.I. McLachlan, SIAM J. Sci. Comput. **16**, 151 (1995).
[32] M.M. Chawla and S.R. Sharma, BIT **21**, 455 (1981).
[33] R.I. McLachlan and S.K. Gray, Appl. Numer. Math. **25**, 275 (1997).
[34] The coefficients for $P_{38}2$ are collected in tables III and IV in section IV and for the the rest of the schemes the coefficients are available from the authors upon request.

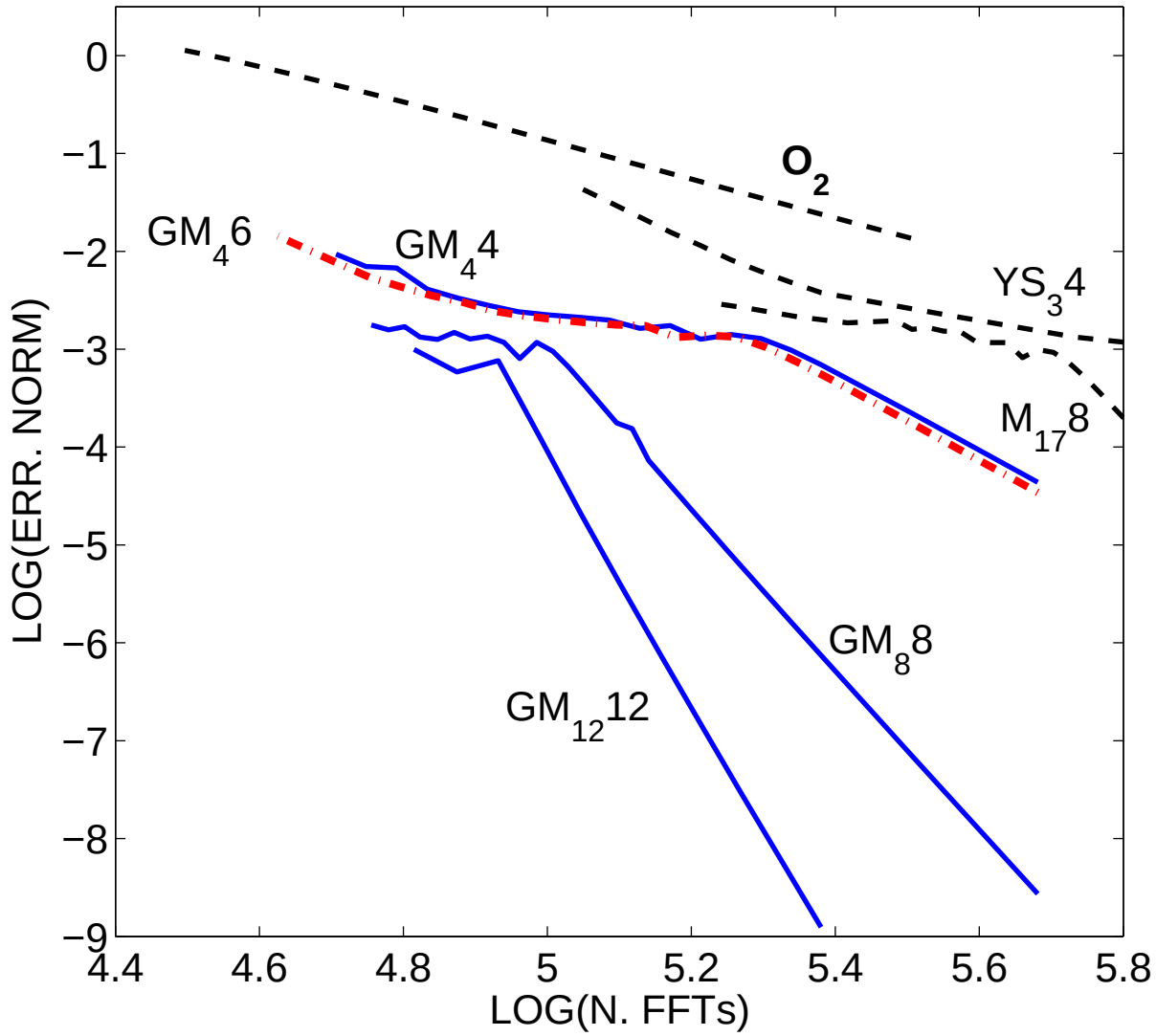

FIG. 1: Error in the wave function versus the number of FFT calls in logarithmic scale for the methods $\mathbf{O}_2$, YS_34, $M_{17}8$, GM_44, GM_64, GM_88 and $GM_{12}12$.

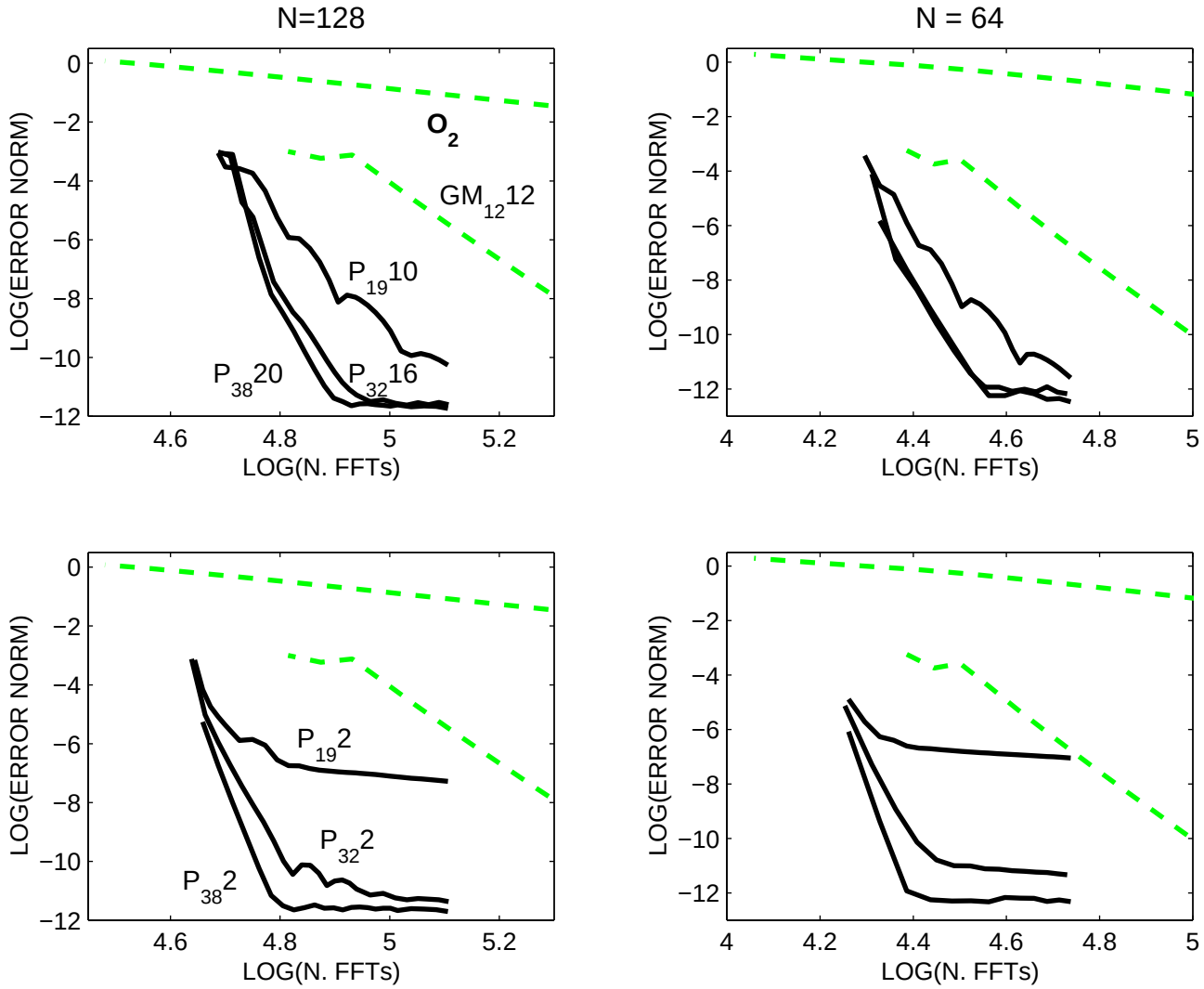

FIG. 2: Error in the wave function versus the number of FFT calls in logarithmic scale for $P_{19}10$, $P_{32}16$ and $P_{38}20$ (top figures) and $P_{19}2$, $P_{32}2$ and $P_{38}2$ (bottom figures) for $x_m = -1/10$, $N = 128$ (left figures) and $N = 64$ (right figures). The results for $\mathbf{O}_2$ and GM_{12} (broken lines) are included as reference.

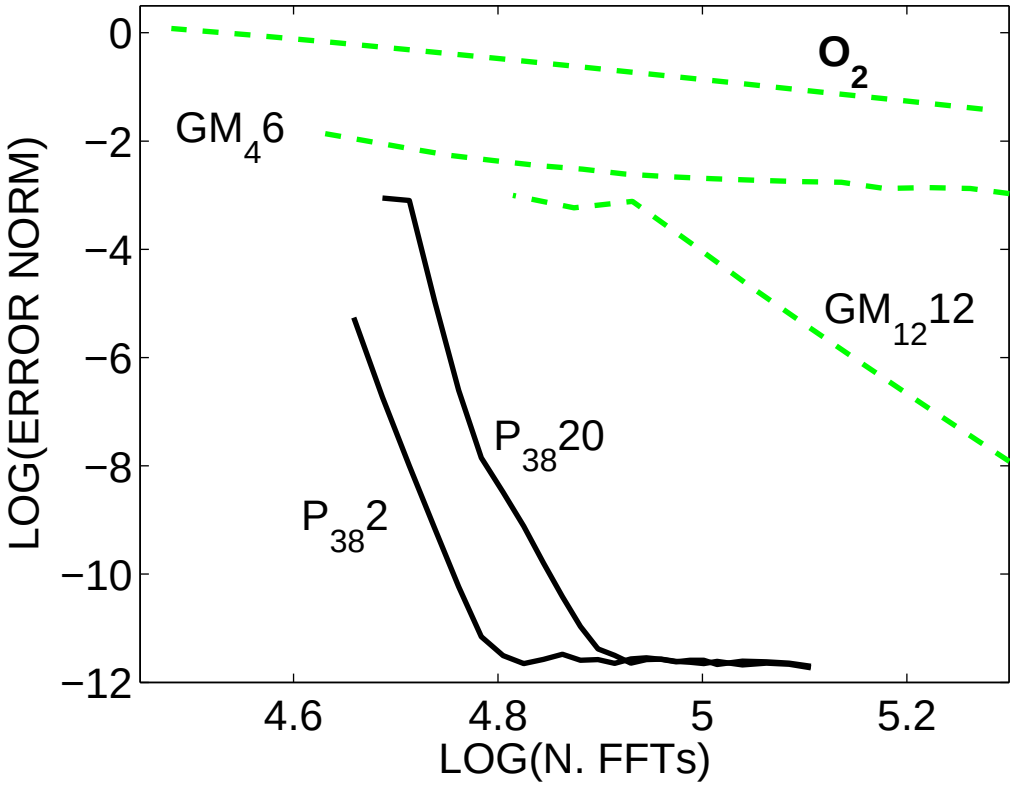

FIG. 3: Same as Fig. 2 with $N = 128$ for the processed methods with 38-stage kernels of effective order 20 and 2. The results of $GM_6 4$ are also included as a reference.