Introduction to symplectic integrator methods for solving diverse dynamical equations

Siu A. Chin

Department of Physics and Astronomy

Texas A&M University

College Station, TX, 77840, USA

Third Lecture

The algebratization of algorithms; higher order algorithms; the proof of canonicity and the proof of non-forward time step algorithms beyond second-order.

In Lecture II we have arrived at the Poisson operator solution of classical dynamics.

$$W(t) = \mathrm{e}^{t(T+V)} W(0) = \mathrm{e}^{tH} W(0).$$

For the most common, separable Hamiltonian

$$H(q_i, p_i) = \sum_{i=1}^{n} \frac{p_i^2}{2m} + U(q_i),$$

if we define

$$\mathbf{v} \equiv \frac{\mathbf{p}}{m} \qquad \mathbf{a}(\mathbf{q}) \equiv \frac{\mathbf{F}(\mathbf{q})}{m}$$

then

$$T = \mathbf{v} \cdot \frac{\partial}{\partial \mathbf{q}} \qquad V = \mathbf{a}(\mathbf{q}) \cdot \frac{\partial}{\partial \mathbf{v}}.$$

We now observe that

$$\mathrm{e}^{tT} \begin{pmatrix} \mathbf{q} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{q} + t\mathbf{v} \\ \mathbf{v} \end{pmatrix} \qquad \mathrm{e}^{tV} \begin{pmatrix} \mathbf{q} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{q} \\ \mathbf{v} + t\mathbf{a}(\mathbf{q}) \end{pmatrix}.$$

That is, although $\exp(tT + tV)$ cannot be solved exactly, the component operator $\exp(tT)$ and $\exp(tV)$ are just shift operators, and can be evaluated exactly. This, therefore suggests that we should try to approximate

$$e^{t(T+V)} = \prod_i e^{a_i tT} e^{b_i tV}$$

to take advantage of the solvability of $\exp(tT)$ and $\exp(tV)$.

The advantage of this product decomposition or splitting is that the resulting algorithm is always a canonical transformation with $\det M = 1$, and now known as symplectic integrators.

Moreover, this reduces the derivation of algorithms for solving classical dynamics from doing *analysis* (analyzing how derivatives are to be approximated) to that of just doing *algebra* (determining the coefficients $\{a_i, b_i\}$).

First-order algorithms

The Baker-Campbell-Hausforff formula

$$e^A e^B = e^{A+B+\frac{1}{2}[A,B]+\frac{1}{12}([A,[A,B]]-[B,[A,B]])+\cdots}$$

is the key to this approach, suggesting that we can approximate

$$e^{\Delta t(T+V)}$$

by

$$\mathcal{T}_{1A} = e^{\Delta t V} e^{\Delta t T} = e^{\Delta t(T+V)+\frac{1}{2}\Delta t^2[V,T]+O(\Delta t^3)} = e^{\Delta t H_{1A}}$$

where the Hamiltonian operator of the approximation is

$$H_{1A} = T + V + \frac{1}{2}\Delta t[V,T] + \cdots = H + O(\Delta t)$$

which differs from the original Hamiltonian by an error term first order in Δt. We call this a first-order operator splitting 1A

This first-order splitting produces the following first-order symplectic algorithm 1A:

$$\mathbf{q}(t) = \mathrm{e}^{\Delta t V}\mathrm{e}^{\Delta t T}\mathbf{q} = \mathrm{e}^{\Delta t V}(\mathbf{q} + \Delta t\mathbf{v}) = \mathbf{q} + \Delta t(\mathbf{v} + \Delta t\mathbf{a}(\mathbf{q})),$$

which is equivalent to first updating

$$\mathbf{v}_1 = \mathbf{v} + \Delta t\mathbf{a}(\mathbf{q})$$

then setting

$$\mathbf{q}_1 = \mathbf{q} + \Delta t\mathbf{v}_1.$$

If we identify $\mathbf{q}_1 = \mathbf{q}(\Delta t)$ and $\mathbf{v}_1 = \mathbf{v}(\Delta t)$, then this is precisely Cromer's algorithm!

Note the variables are updated in the reverse order of the operators. That is, the operators are acting from right to left, but the variables are updated according to each operator from left to right.

The first-order algorithm corresponds to

$$\mathcal{T}_{1B} = \mathrm{e}^{\Delta t T}\mathrm{e}^{\Delta t V} = \mathrm{e}^{\Delta t(T+V)+\frac{1}{2}\Delta t^2[T,V]+O(\Delta t^3)}$$

is then

$$\mathbf{q}(t) = \mathrm{e}^{\Delta t T}\mathrm{e}^{\Delta t V}\mathbf{q} = \mathrm{e}^{\Delta t T}\mathbf{q} = \mathbf{q} + \Delta t\mathbf{v},$$

$$\mathbf{v}(t) = \mathrm{e}^{\Delta t T}\mathrm{e}^{\Delta t V}\mathbf{v} = \mathrm{e}^{\Delta t T}(\mathbf{v} + \Delta t\mathbf{a}(\mathbf{q})) = \mathbf{v} + \Delta t\mathbf{a}(\mathbf{q} + \Delta t\mathbf{v}),$$

which is equivalent to first updating

$$\mathbf{q}_1 = \mathbf{q} + \Delta t\mathbf{v}$$

then

$$\mathbf{v}_1 = \mathbf{v} + \Delta t\mathbf{a}(\mathbf{q}_1)$$

which is the canonical transformation produced by F_3! We will call this algorithm 1B.

Time-symmetry

These first-order symplectic integrators are all stable with $\det M = 1$. However, they have one defect, that is not characteristic of the exact solution. Since classical mechanics is time-symmetric, running the exact solution backward in time would retrace the trajectory back to its origin. This is not case for these first order algorithms, since it is easy to see that

$$\mathcal{T}_{1A}(-\Delta t)\mathcal{T}_{1A}(\Delta t) = \mathrm{e}^{-\Delta t V}\mathrm{e}^{-\Delta t T}\mathrm{e}^{\Delta t V}\mathrm{e}^{\Delta t T} \neq 1$$

What kind of splitting

$$\mathcal{T}(\Delta t) = \prod_i \mathrm{e}^{a_i \Delta t T}\mathrm{e}^{b_i \Delta t V}$$

would produce a time-symmetric algorithm? (Hint: in order for $\mathcal{T}(-\Delta t)\mathcal{T}(\Delta t) = 1$ we must have the right-edge operator of $\mathcal{T}(-\Delta t)$ cancels the left edge operator of $\mathcal{T}(\Delta t)$.)

What kind of splitting would produce a time-symmetric algorithm? Since we must have the right-edge operator of $\mathcal{T}(-\Delta t)$ cancels the left edge operator of $\mathcal{T}(\Delta t)$ we must have

$$\mathcal{T}(\Delta t) = \mathrm{e}^{a_1 \Delta t T} \cdots \mathrm{e}^{a_1 \Delta t T} \quad \text{or} \quad \mathcal{T}(\Delta t) = \mathrm{e}^{b_1 \Delta t V} \cdots \mathrm{e}^{b_1 \Delta t V}.$$

After that cancellation, the same argument applies to the next operator

$$\mathcal{T}(\Delta t) = \mathrm{e}^{a_1 \Delta t T} \mathrm{e}^{b_1 \Delta t V} \cdots \mathrm{e}^{b_1 \Delta t V} \mathrm{e}^{a_1 \Delta t T}$$

or

$$\text{or} \quad \mathcal{T}(\Delta t) = \mathrm{e}^{b_1 \Delta t V} \mathrm{e}^{a_1 \Delta t T} \cdots \mathrm{e}^{a_1 \Delta t T} \mathrm{e}^{b_1 \Delta t V}.$$

So, the answer is clear, the splitting must be left-right symmetric, in order for

$$\mathcal{T}(-\Delta t)\mathcal{T}(\Delta t) = 1.$$

A time-symmetric splitting $\mathcal{T}(\Delta t)$, obeying

$$\mathcal{T}(-\Delta t)\mathcal{T}(\Delta t) = 1$$

must have the form

$$\mathcal{T}(\Delta t) = \exp(\Delta t(T + V) + \Delta t^3 E_3 + \Delta t^5 E_5 + \cdots),$$

with only odd powers of Δt, since any even powers of Δt would not cancel. This means that any time-symmetric, or left-right symmetric splitting, must have

$$H_{\mathcal{T}} = H + \Delta t^2 E_3 + \Delta t^4 E_5 + \cdots$$

which is at least *second-order*.

Second-order algorithms

It's simple to produce left-right symmetric splittings,

$$\mathcal{T}_{2A}(\Delta t) = \mathcal{T}_{1A}(\Delta t/2)\mathcal{T}_{1B}(\Delta t/2) = \mathrm{e}^{\frac{1}{2}\Delta t V}\mathrm{e}^{\frac{1}{2}\Delta t T}\mathrm{e}^{\frac{1}{2}\Delta t T}\mathrm{e}^{\frac{1}{2}\Delta t V}$$

$$= \mathrm{e}^{\frac{1}{2}\Delta t V}\mathrm{e}^{\Delta t T}\mathrm{e}^{\frac{1}{2}\Delta t V},$$

which translates to the symplectic algorithm

$$\mathbf{v}_1 = \mathbf{v} + \frac{1}{2}\Delta t\mathbf{a}(\mathbf{q})$$

$$\mathbf{q}_1 = \mathbf{q} + \Delta t\mathbf{v}_1$$

$$\mathbf{v}_2 = \mathbf{v}_1 + \frac{1}{2}\Delta t\mathbf{a}(\mathbf{q}_1)$$

known as the *velocity-Verlet* algorithm, empirically found long before this derivation. We should call this algorithm 2A.

Similarly, we have:

$$\mathcal{T}_{2B}(\Delta t) = \mathcal{T}_{1B}(\Delta t/2)\mathcal{T}_{1A}(\Delta t/2) = \mathrm{e}^{\frac{1}{2}\Delta t T}\mathrm{e}^{\frac{1}{2}\Delta t V}\mathrm{e}^{\frac{1}{2}\Delta t V}\mathrm{e}^{\frac{1}{2}\Delta t T}$$

$$= \mathrm{e}^{\frac{1}{2}\Delta t T}\mathrm{e}^{\Delta t V}\mathrm{e}^{\frac{1}{2}\Delta t T},$$

which translates to the symplectic algorithm 2B

$$\mathbf{q}_1 = \mathbf{q} + \frac{1}{2}\Delta t\mathbf{v}$$

$$\mathbf{v}_1 = \mathbf{v} + \Delta t\mathbf{a}(\mathbf{q}_1)$$

$$\mathbf{q}_2 = \mathbf{q}_1 + \frac{1}{2}\Delta t\mathbf{v}_1$$

which is sometime referred to as the *position-Verlet* algorithm.

From this analysis, there are only TWO canonical first and second-order symplectic algorithms.

Higher order algorithms

Once we have these two second-order algorithms, we can generate symplectic integrator of any order. Let $\mathcal{T}_2$ be either of the two second-order algorithm. Consider now

$$\mathcal{T}_4 = \mathcal{T}_2(\epsilon)\mathcal{T}_2(-s\epsilon)\mathcal{T}_2(\epsilon),$$

which is left-right symmetric but with a negative middle time step. Since

$$\mathcal{T}_2(\epsilon) = \exp(\epsilon H + \epsilon^3 E_3 + \epsilon^5 E_5 + \cdots),$$

$$\mathcal{T}_2(\epsilon)\mathcal{T}_2(-s\epsilon)\mathcal{T}_2(\epsilon) = \exp\Big[(2-s)\epsilon H + (2-s^3)\epsilon^3 E_3 + O(\epsilon^5)\Big].$$

If we now choose $s = 2^{1/3}$, then the ϵ^3 error term will vanish and we will have a *fourth-order*, known as the *Forest-Ruth* algorithm. The next thing is to renormalize the time step so that $(2-s)\epsilon = \Delta t$, giving

$$\mathcal{T}_4(\Delta t) = \mathcal{T}_2\Big(\frac{\Delta t}{2-s}\Big)\mathcal{T}_2\Big(-\frac{s\Delta t}{2-s}\Big)\mathcal{T}_2\Big(\frac{\Delta t}{2-s}\Big).$$

Now we can repeat the process to produce a *sixth-order* algorithm:

$$\mathcal{T}_6(\Delta t) = \mathcal{T}_4\Big(\frac{\Delta t}{2-s}\Big)\mathcal{T}_4\Big(-\frac{s\Delta t}{2-s}\Big)\mathcal{T}_4\Big(\frac{\Delta t}{2-s}\Big)$$

where now $s = 2^{1/5}$. Or a *eighth-order* algorithm via

$$\mathcal{T}_8(\Delta t) = \mathcal{T}_6\Big(\frac{\Delta t}{2-s}\Big)\mathcal{T}_6\Big(-\frac{s\Delta t}{2-s}\Big)\mathcal{T}_6\Big(\frac{\Delta t}{2-s}\Big)$$

where now $s = 2^{1/7}$, etc..

This is a great advance in symplectic integrators, where one can easily produce arbitrarily higher-order algorithms. But there are two disadvantages: 1) Beyond second-order, some time steps must be negative. This means that these splitting methods cannot be used to solve *time-irreversible* equations, or semi-group problems, involving the *heat/diffusion kernel*. 2) The number of operators grows exponentially with order of the algorithm. We will come back to address two points.

Proof of Canoniticity

We will now prove that any splitting of the form

$$\mathrm{e}^{t(T+V)} = \prod_i \mathrm{e}^{a_i tT} \mathrm{e}^{b_i tV}$$

produces a canonical transformation. Recall that a canonical transformation $(q_i, p_i) \to (Q_i, P_i)$ is a transformation such that Hamilton's equations are preserved for the new variables

$$\dot{Q}_i = \frac{\partial K}{\partial P_i}, \qquad \dot{P}_i = -\frac{\partial K}{\partial Q_i}, \tag{6}$$

with respect to $K(Q_i, P_i) = H(q_i, p_i)$. From Poisson's equation of motion , we must also have

$$\dot{Q}_i = \{Q_i, H\}, \qquad \dot{P}_i = \{P_i, H\}.$$

Now that Q_i and P_i are our new variables, we must therefore express (sum over repeated indices) the partial derivatives of H in the Poisson brackets in terms of Q_i, P_i and K:

$$\frac{\partial H}{\partial q_k} = \frac{\partial K}{\partial Q_j}\frac{\partial Q_j}{\partial q_k} + \frac{\partial K}{\partial P_j}\frac{\partial P_j}{\partial q_k}$$

$$\frac{\partial H}{\partial p_k} = \frac{\partial K}{\partial Q_j}\frac{\partial Q_j}{\partial p_k} + \frac{\partial K}{\partial P_j}\frac{\partial P_j}{\partial p_k}.$$

Substitute the above then gives straightforwardly,

$$\dot{Q}_i = \{Q_i, Q_j\}\frac{\partial K}{\partial Q_j} + \{Q_i, P_j\}\frac{\partial K}{\partial P_j}$$

$$\dot{P}_i = \{P_i, P_j\}\frac{\partial K}{\partial P_j} + \{P_i, Q_j\}\frac{\partial K}{\partial Q_j}.$$

This agrees with (6) if and only if

$$\{Q_i, Q_j\} = 0, \qquad \{P_i, P_j\} = 0, \qquad \{Q_i, P_j\} = \delta_{ij}. \tag{7}$$

This means that the transformed variable must also obey the fundamental Poisson brackets. Thus the fundamental Poisson brackets are an equivalent way of characterizing a canonical transformation.

Furthermore, in (7) the Poisson brackets are computed with respect to $\{q_i, p_i\}$. One clearly would have obtained the same fundamental Poisson brackets if computed with respect to $\{Q_i, P_i\}$! This examplifies that Poisson brackets are canonically invariant, are the same for all canonical transformed variables.

Lie operators and Lie Transforms

For any dyanmical function $S(q_i, p_i)$, we can define an associated *Lie* operator $\hat{S}$ via the Poisson bracket

$$\hat{S} = \{\cdot, S\} \equiv \sum_{i=1}^{n} \Big(\frac{\partial S}{\partial p_i} \frac{\partial}{\partial q_i} - \frac{\partial S}{\partial q_i} \frac{\partial}{\partial p_i} \Big),$$

that is, when $\hat{S}$ acts on any other dynamical function $W(q_i, p_i)$, its effect is

$$\hat{S}W = \{W, S\} \equiv \sum_{i=1}^{n} \Big(\frac{\partial S}{\partial p_i} \frac{\partial}{\partial q_i} - \frac{\partial S}{\partial q_i} \frac{\partial}{\partial p_i} \Big) W.$$

The importance of introducing Lie operators is that the exponential of *any* Lie operator, called a *Lie transform*, defined formally by the infinite *Lie series*,

$$T_S = \mathrm{e}^{\epsilon \hat{S}} = 1 + \epsilon \hat{S} + \frac{1}{2!} + \epsilon^2 \hat{S}^2 + \frac{1}{3!} \epsilon^3 \hat{S}^3 + \cdots,$$

produces an *explicit* canonical transformation via

$$Q_i = \mathrm{e}^{\epsilon \hat{S}} q_i, \qquad P_i = \mathrm{e}^{\epsilon \hat{S}} p_i. \tag{8}$$

The small parameter ϵ is introduced to guarantee the series' convergence to the exponential function.

To show that (8) is a canonical transformation, we verify that Q_i and P_i satisfy the fundamental Poisson bracket (7). The key steps are elementary; all we need are some preliminary lemmas. First, we note that $\hat{S}$ is just a first order differential operator, and therefore for any two functions of the dynamical variables, $f(q_i, p_i)$ and $g(q_i, p_i)$, we have the product rule:

$$\hat{S}(fg) = (\hat{S}f)g + f(\hat{S}g).$$

From here, it is not too difficult to show that

$$\mathrm{e}^{\epsilon \hat{S}}(fg) = (\mathrm{e}^{\epsilon \hat{S}} f)(\mathrm{e}^{\epsilon \hat{S}} g), \tag{9}$$

and hence

$$e^{\epsilon\hat{S}}\{f,g\} = \{e^{\epsilon\hat{S}}f, e^{\epsilon\hat{S}}g\}.$$

Take the simplest case of $\hat{S} = v\partial/\partial q$, then $e^{\epsilon\hat{S}}f(q)$ is just the Taylor expansion of $f(q+\epsilon v)$. Eq.(9) is then obviously true

$$e^{\epsilon\hat{S}}\big(f(q)g(q)\big) = f(q+\epsilon v)g(q+\epsilon v) = (e^{\epsilon\hat{S}}f)(e^{\epsilon\hat{S}}g).$$

For Q_i and P_i defined by (8) we can now use to compute

$$\{Q_i, Q_j\} = \{e^{\epsilon\hat{S}}q_i, e^{\epsilon\hat{S}}q_j\} = e^{\epsilon\hat{S}}\{q_i, q_j\} = 0$$

$$\{P_i, P_j\} = \{e^{\epsilon\hat{S}}p_i, e^{\epsilon\hat{S}}p_j\} = e^{\epsilon\hat{S}}\{p_i, p_j\} = 0$$

$$\{Q_i, P_j\} = \{e^{\epsilon\hat{S}}q_i, e^{\epsilon\hat{S}}p_j\} = e^{\epsilon\hat{S}}\{q_i, p_j\} = e^{\epsilon\hat{S}}\delta_{ij} = \delta_{ij}.$$

In this Lie transform formulation of canonical transformation, we see explicitly how the fundamental Poisson brackets are preserve and transferred to the new variables. The last equality in follows since δ_{ij} is a constant and all terms of $\mathrm{e}^{\epsilon \hat{S}}$ acting on it vanish except the first term, which is 1.

Finally we recall that,

$$T = \hat{T} = \{\cdot, \sum_k \frac{p_k^2}{2m}\} = \sum_i \frac{p_i}{m} \frac{\partial}{\partial q_i}$$

and

$$V = \hat{V} = \{\cdot, U(q_i)\} = \sum_i F_i \frac{\partial}{\partial p_i}$$

are Lie operators and $\mathrm{e}^{\Delta t T}$ and $\mathrm{e}^{\Delta t V}$ are Lie transforms.

No forward time step algorithms beyond second order

We note that we have a middle negative time step in the triplet construction of the fourth-order algorithm. This negative time-step is unavoidable and is the content of the Sheng-Suzuki theorem. We give here a simplified proof by Blanes and Casas. This is a purely mathematical result applies to the splitting of any two operators A and B.

The statement is that if one were to approximate

$$\mathrm{e}^{\epsilon(A+B)} \quad \text{by} \quad \prod_i \mathrm{e}^{\alpha_i \epsilon A} \mathrm{e}^{\beta_i \epsilon B},$$

to any order higher than the second-order, then the set of coefficients $\{\alpha_i, \beta_i\}$ cannot be all positive.

Let the two first-order splitting be

$$T_A(\epsilon) = \mathrm{e}^{\epsilon A}\mathrm{e}^{\epsilon B} \qquad \text{and} \qquad T_B(\epsilon) = \mathrm{e}^{\epsilon B}\mathrm{e}^{\epsilon A}.$$

Since

$$T_B(-\epsilon)T_A(\epsilon) = \mathrm{e}^{-\epsilon B}\mathrm{e}^{-\epsilon A}\mathrm{e}^{\epsilon A}\mathrm{e}^{\epsilon B} = 1.$$

This means that

$$T_B(-\epsilon) = T_A^{-1}(\epsilon) \qquad \text{and} \qquad T_B(\epsilon) = T_A^{-1}(-\epsilon).$$

By the Baker-Campbell-Hausdorff formula

$$T_A(\epsilon) = \mathrm{e}^{\epsilon A}\mathrm{e}^{\epsilon B} = \exp(\epsilon F_1 + \epsilon^2 F_2 + \epsilon^3 F_3 + \cdots) = \exp(F)$$

where $F_1 = A + B$, $F_2 = \frac{1}{2}[A, B]$, $F_3 = \frac{1}{12}([(A - B), [A, B]])$, etc.. Clearly then

$$T_A^{-1}(\epsilon) = \exp(-F) = \exp(-\epsilon F_1 - \epsilon^2 F_2 - \epsilon^3 F_3 + \cdots),$$

and therefore

$$T_B(\epsilon) = T_A^{-1}(-\epsilon) = \exp(\epsilon F_1 - \epsilon^2 F_2 + \epsilon^3 F_3 + \cdots).$$

Hence,

$T_A(a_1\epsilon)T_B(a_2\epsilon)$

$= \exp[(a_1 + a_2)\epsilon F_1 + (a_1^2 - a_2^2)\epsilon^2 F_2 + (a_1^3 + a_2^3)\epsilon^3 F_3 + O(\epsilon^3)[F_1, F_2]\cdots]$

$T_A(a_3\epsilon)T_B(a_4\epsilon)$

$= \exp[(a_3 + a_4)\epsilon F_1 + (a_3^2 - a_4^2)\epsilon^2 F_2 + (a_3^3 + a_4^3)\epsilon^3 F_3 + O(\epsilon^3)[F_1, F_2]\cdots]$

Now coming back to $\prod_i \mathrm{e}^{\alpha_i \epsilon A}\mathrm{e}^{\beta_i \epsilon B}$ and define $\{a_i, b_i\}$ from $\{\alpha_i, \beta_i\}$ as follow: $\alpha_1 = a_1$, $\beta_1 = a_1 + a_2$, $\alpha_2 = a_2 + a_3$, $\beta_2 = a_3 + a_4$, etc., then

$\mathrm{e}^{\alpha_1 \epsilon A}\mathrm{e}^{\beta_1 \epsilon B}\mathrm{e}^{\alpha_2 \epsilon A}\mathrm{e}^{\beta_2 \epsilon B}\ldots$

$= \mathrm{e}^{a_1 \epsilon A}\mathrm{e}^{a_1 \epsilon B}\mathrm{e}^{a_2 \epsilon B}\mathrm{e}^{a_2 \epsilon A}\mathrm{e}^{a_3 \epsilon A}\mathrm{e}^{a_3 \epsilon B}\mathrm{e}^{a_4 \epsilon B}\ldots$

$= T_A(a_1\epsilon)T_B(a_2\epsilon)T_A(a_3\epsilon)T_B(a_4\epsilon)\cdots$

$= \exp[C_1\epsilon F_1 + C_2\epsilon^2 F_2 + C_3\epsilon^3 F_3 + O(\epsilon^3)[F1, F2]\cdots]$

where

$$C_1 = a_1 + a_2 + a_3 + a_4 + \cdots$$

$$C_2 = a_1^2 - a_2^2 + a_3^2 - a_4^2 + \cdots$$

$$C_3 = a_1^3 + a_2^3 + a_3^3 + a_4^3 + \cdots$$

For a first-order algorithm, we must have $C_1 = 1$, for a second-order algorithm, we must have additionally $C_2 = 0$. For a third-order algorithm, we must have additionally, as a necessary condition, $C_3 = 0$. (This is because there is also the $O(\epsilon^3)[F_1, F2]$ term that needs to vanish.)

If C_3 where to be zero, then at some point, one must have

$$a_i^3 + a_{i+1}^3 < 0.$$

Since

$$x^3 + y^3 = (x+y)[\frac{3}{4}y^2 + (x - \frac{1}{2}y)^2]$$

$$x^3 + y^3 < 0 \quad \Rightarrow \quad x + y < 0,$$

we must have

$$a_i + a_{i+1} < 0.$$

Recall that

$$\alpha_1 = a_1, \quad \beta_1 = a_1 + a_2, \quad \alpha_2 = a_2 + a_3, \quad \beta_2 = a_3 + a_4 \quad \text{etc.}$$

then one of the α_i or β_i must be negative. The proof can be refined to show that at least one α_i AND one β_i must be negative.