NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

Construction of Runge-Kutta methods of Crouch-Grossman type of high order

by

Z. Jackiewicz, A. Marthinsen and B. Owren

PREPRINT
NUMERICS NO. 4/1999

NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
TRONDHEIM, NORWAY

This report has URL http://www.math.ntnu.no/preprint/numerics/1999/N4-1999.ps
Address: Department of Mathematical Sciences, Norwegian University of Science and Technology,
N-7491 Trondheim, Norway.

CONSTRUCTION OF RUNGE–KUTTA METHODS OF CROUCH–GROSSMAN TYPE OF HIGH ORDER

Z. JACKIEWICZ[1], A. MARTHINSEN[2] and B. OWREN[3] *

[1]*Department of Mathematics, Arizona State University, Tempe, Arizona 85287, USA.*
email: jackiewi@zjsun.la.asu.edu

[2]*Department of Mathematical Sciences, NTNU, N-7034 Trondheim, Norway.*
email: Arne.Marthinsen@math.ntnu.no

[3]*Department of Mathematical Sciences, NTNU, N-7034 Trondheim, Norway.*
email: Brynjulf.Owren@math.ntnu.no

Abstract.

An approach is described to the numerical solution of order conditions for Runge-Kutta methods whose solutions evolve on a given manifold. This approach is based on least squares minimization using the Levenberg-Marquardt algorithm. Examples of methods of order four, five and six are given and numerical experiments are presented which confirm that the derived methods have the expected order of accuracy.

AMS subject classification: 65L05.

Key words: Runge-Kutta methods, order conditions, geometric integration, rigid frames, least squares minimization.

1 Introduction

The solution of many systems of ordinary differential equations (ODEs) preserves certain constraints or invariants. For example, a Hamiltonian system preserves the symplectic structure (all Poincaré invariants) [16]; skew-Hermitian systems maintain unitariness of the solution [7]; isospectral flows retain the eigenvalues of the initial matrix [2]; conservative mechanical systems preserve energy [17]; rotating rigid body in space preserves angular momentum [17]; the solution of differential-algebraic equations (DAEs) preserves algebraic constrains [15].

Multi-purpose methods and software which can efficiently solve the wide range of possible problems usually cannot incorporate known qualitative information about the solution, and this may lead to severe shortcomings. It was demonstrated, for example, by Führer and Leimkuhler [9] that the numerical solution of DAEs for constrained mechanical motion may drift from the position constraints in an unpredictable way. Similarly, Dieci, Russel and Van Vleck [7] observed in the context of skew-Hermitian problems that the use of a nonunitary integrator is equivalent to modifying the original problem into a nonskew-Hermitian one,

*This work was sponsored in part by The Norwegian Research Council under contract no. 111038/410, through the SYNODE project. WWW: `http://www.math.ntnu.no/num/synode/`

which is generally either unstable or stiff for long time integration and presents all sorts of different difficulties.

Many ODE systems which preserve certain constraints can be formulated as differential equations on manifolds. This was demonstrated, for example, by Arnold [1] for Hamiltonian systems; Crouch and Grossman [3] (see also [7]) for skew-symmetric systems (in this case the underlying manifold is the Lie group of unitary matrices); by Calvo, Iserles and Zanna [2] for isospectral flows; and by Rheinboldt [15] for DAEs. It is important, therefore, to derive numerical methods whose solutions also evolve on the manifold. There seem to be two principal approaches to accomplish this task. The first one is to use any classical numerical scheme to compute an approximation to the solution at a given point and then to project this result to the manifold on which the solution is known to lie. To increase computational efficiency the projection after a fixed number of time-steps can be made rather than after each step. The second approach, which received recently a lot of attention, consists of construction of numerical methods whose solutions automatically evolve on a given manifold. Following the terminology of [4] such methods will be called geometrically exact.

The two important early developments in this area are due to Crouch and Grossman [3] and Munthe-Kaas [10]. The paper [3] was first overlooked in numerical analysis community and only recently received the proper recognition. The approach taken in [3] is based on the use of predefined vector fields $E_1, E_2, \ldots, E_m$ on R^d, $m \leq d$, and the differential system is expressed in the form

$$\begin{cases} y' = F(y) = \displaystyle\sum_{i=1}^{m} f_i(y)E_i(y), \quad t \geq 0, \\ y(0) = y_0, \end{cases} \tag{1.1}$$

where f_i are real analytic functions on R^d. The numerical methods (Runge-Kutta (RK), linear multistep) are then defined through various compositions of the flows of vector fields in the span of $E_1, E_2, \ldots, E_m$, i.e. by using flows of the more elementary system

$$y' = \sum_{i=1}^{m} a_i E_i(y) \tag{1.2}$$

obtained by freezing the coefficients $a_i = f_i(p)$ at some point p. This will be explained in more detail in case of RK methods in Section 2. The underlying assumption (assumption 1 in [3]) is then that one must be able to compute the flows of (1.2) for any $p \in R^d$ to arbitrary accuracy. In the Euclidean case corresponding to $E_i = \partial/\partial x_i$, $m = d$, the methods proposed in [3] reduce to traditional RK or linear multistep methods. In the general case, the methods constructed in this way will evolve on a submanifold $\mathcal{M}$ of R^d whose tangent space at the point p is the Lie algebra generated by $E_1, E_2, \ldots, E_m$ evaluated at p. Hence, this approach is quite general and may be used to tailor integration methods which will preserve any prescribed constraints. This process is illustrated in [4] for the RK method of order three which preserves unitariness of the solution to the rolling ball equations, and in [5] for RK and linear multistep methods which preserve unitariness of the solution of the system describing the evolution of the attitude of a rigid body, such as a spacecraft.

Munthe-Kaas [10] deals with the RK methods for ODEs evolving on Lie groups, i.e., manifolds which also possess the structure of a continuous group. As demonstrated in [17], such problems are of considerable interest in computational mechanics. Lie group actions constitute a set of basic flows which are usually easy to compute (this means that Assumption 1 in [3] is satisfied) and if a numerical integration scheme can be expressed in terms of these flows, the numerical solution will automatically stay on the manifold. In [10] order conditions for RK methods are obtained by comparing the Lie-Butcher series for the exact solution with the corresponding Lie-Butcher series for the numerical solution. In this theory (as well as in [3]), the elementary differentials are represented as commutators between vector fields (in classical theory they are represented as rooted trees). Although [10] deals only with the abelian (commutative) case, the theory can be extended in a natural way to non-abelian Lie groups.

The order conditions derived in [3] are necessarily quite complicated and difficult to solve. In [3] they were solved for low orders yielding a first example of an RK method of order three whose solution evolves on a given manifold. Such methods will be referred to as RK methods of Crouch-Grossman type. The theory of order conditions for these methods (including implicit methods) was further refined by Owren and Marthinsen in [13] and these authors also obtained the first example of explicit RK method of Crouch-Grossman type of order four. Munthe-Kaas, in a recent paper [11], have also found a RK method of order four whose solution evolves on a Lie group. He also proved in [12] that any classical RK method can be implemented as a RK method of the same order whose solution stays on a general homogeneous manifold.

The organization of this paper is as follows. In Section 2 we define explicit and implicit RK methods of Crouch-Grossman type and discuss briefly order conditions for these methods. In Section 3 we describe the approach to the numerical solution of a system of nonlinear equations corresponding to order conditions which is based on least squares minimization. Using this technique we were able to solve the resulting order conditions up to order five and examples of such methods of order four and five are presented in Appendix A. We also present a set of coefficients that approximately satisfy the order six conditions. Finally, in Section 4 we report the results of some numerical experiments on the spinning top problem which confirm that the constructed methods have the expected order of accuracy.

2 RK methods of Crouch-Grossman type

To define explicit RK methods for the numerical integration of (1.1) assume that an approximation $p = y_k$ to $y(t_k)$ is already computed and consider the differential system with coefficients frozen at the point p

$$\left\{ \begin{array}{l} y' = F_p(y) = \displaystyle\sum_{i=1}^{m} f_i(p)E_i(y), \quad t \in [t_k, t_{k+1}], \\ y(t_k) = p, \end{array} \right. \tag{2.1}$$

$t_{k+1} = t_k + h$. We then define a numerical approximation y_{k+1} to $y(t_{k+1})$ in terms of the flows of the vector field $F(y)$ with coefficients frozen at some points as follows

$$\left\{ \begin{array}{l} Y_1 = p, \quad F_p^1(y) = F_p(y), \\ Y_2 = e^{ha_{21}F_p^1}p, \quad F_p^2(y) = F_{Y_2}(y), \\ Y_3 = e^{ha_{32}F_p^2}e^{ha_{31}F_p^1}p, \quad F_p^3(y) = F_{Y_3}(y), \\ \qquad\qquad\vdots \\ Y_s = e^{ha_{s,s-1}F_p^{s-1}}e^{ha_{s,s-2}F_p^{s-2}}\cdots e^{ha_{s1}F_p^1}p, \quad F_p^s(y) = F_{Y_s}(y), \\ y_{k+1} = e^{hb_sF_p^s}e^{hb_{s-1}F_p^{s-1}}\cdots e^{hb_1F_p^1}p. \end{array} \right. \tag{2.2}$$

In the above formulas we have used a standard notation $e^{tF}p$ for the t-flow of the vector field F through p. Such methods were referred to in [3] as explicit RK algorithms adapted to a frame or in [13] as integration methods based on rigid frames. In this paper we will refer to (2.2) as RK methods of Crouch-Grossman type. As explained in [3] it follows that if the solution $y(t)$ of (1.1) evolves on a manifold $\mathcal{M}$ then the numerical approximations y_k computed by (2.2) will also evolve on this manifold $\mathcal{M}$ assuming that $y_0 \in \mathcal{M}$.

Owren and Marthinsen [13] defined also an implicit version of these methods by the following formulas

$$\left\{ \begin{array}{l} Y_r = e^{ha_{rs}F_p^s}e^{ha_{r,s-1}F_p^{s-1}}\cdots e^{ha_{r1}F_p^1}p, \quad F_p^r(y) = F_{Y_r}(y), \\ y_{k+1} = e^{hb_sF_p^s}e^{hb_{s-1}F_p^{s-1}}\cdots e^{hb_1F_p^1}p, \end{array} \right. \tag{2.3}$$

$r = 1, 2, \ldots, s$, which are implicit both in Y_r and F_p^r. Owren and Welfert [14] then proposed two variants of the Newton iterations to solve the system (2.3) in the case when the manifold $\mathcal{M}$ is a Lie group.

We will represent explicit methods of the form (2.2) by Butcher tableaux of coefficients

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|ccccc} c_1 = 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array},$$

where $c_i = \sum_{j=1}^{i-1} a_{ij}$, $i = 1, 2, \ldots, s$.

It was demonstrated in [13] that the method (2.3) has order q if

$$\Phi(t) = \frac{1}{\gamma(t)} \tag{2.4}$$

for all $t \in T_O^2 \cup T_O^3 \cup \cdots \cup T_O^{q+1}$. Here, T_O^r stands for the set of ordered rooted trees of order r, $\Phi(t)$ is the elementary weight corresponding to t and $\gamma(t)$ is the density of the tree t. We refer to [13] for the definitions of these notions.

3 Solution of order conditions by least squares minimization

3.1 Automatic generation of order conditions

The order conditions (2.4) are defined recursively in [13]. The only tree in T_O^1 is denoted τ and trees in T_O^{q+1} where $q \geq 1$ is written in the usual way as $t = [t_1, \ldots, t_\mu]$. The elementary weight $\Phi(t)$ as well as the elementary stage weights $\Phi_r(t)$, $1 \leq r \leq s$, are defined recursively in terms of $\Phi_r(t_k)$, $1 \leq k \leq \mu$, where $\Phi(\tau) = \Phi_r(\tau) = 1$. A set of Fortran subroutines was written to use the same type of recursion to build the $\Phi_r(t)$ for all $t \in T_O^1 \cup \cdots \cup T_O^q$. Each tree t is assigned an integer index, together with a list of integers $i_1, \ldots, i_\mu$ such that if $t = [t_1, \ldots, t_\mu]$ then i_j is the index of t_j. To generate this data structure, we wrote a Maple program whose output is a Fortran subroutine which sets up the data structure for arbitrary high orders. It was proved in Owren and Marthinsen [13] and Tracogna and Welfert [18] that the order conditions (2.4) are not all independent. The Maple program takes advantage of this reduction by creating a integer table in which all entries corresponding to trees that can be discarded are set to zero. This information is used by the subroutine which computes the $\Phi(t)$.

3.2 Numerical solution of order conditions

To solve the order conditions (2.4) we have used an approach based on least squares minimization of the objective function $f(A, b)$ defined by

$$f(A,b) = \sum_{t \in T_O^2 \cup T_O^3 \cup \cdots \cup T_O^{q+1}} \left(\Phi(t) - \frac{1}{\gamma(t)} \right)^2,$$

which depends on the coefficients A and b of the method (2.2). This function was minimized using subroutine `lmdif.f` from MINPACK starting with many randomly generated initial guesses. Then the minima of this function which are equal to zero are also solutions to order conditions (2.4).

The subroutine `lmdif.f` is based on the Levenberg-Marquardt algorithm and we refer to [6] for a detailed description of this approach. The random guesses were generated using the subroutine `zufall.f` which is uniform random number generator written by W. P. Petersen from ETH, Zürich. The iterations were continued until the relative error between two consecutive iterates was at most $xtol = 10^{-18}$ or when relative reductions in the sum of squares were at most

$ftol = 10^{-18}$ or until we reached the maximum number of functions evaluations. This resulted in many solutions for which the value of the function $f(A, b)$ was reduced to about 10^{-15} for methods of order four and five. The system of order conditions seems to be very ill-conditioned for methods of order six and we were only able to reduce $f(A, b)$ to about 10^{-6} for these methods. These solutions were then further improved by running `lmdif.f` in extended precision with $xtol = ftol = 10^{-32}$. This process usually resulted in solutions for which the value of $f(A, b)$ was reduced to about 10^{-31} for methods of order four and five and to about 10^{-8} for methods of order six.

We have performed an extensive computer search looking for methods (2.2) of order $q = 4$ with $s = 5$, order $q = 5$ with $s = 9$, and order $q = 6$ with $s = 16$, where s stands for the number of stages. We have listed in the table below the order q, the number of stages s, the number of free parameters $n = s(s + 1)/2$ for explicit methods given by a_{ij}, $i = 2, 3, \dots, s$, $j = 1, 2, \dots, i - 1$, and b_i, $i = 1, 2, \dots, s$, and the number N_{CG} of order conditions for methods (2.2). For comparison, we have also listed in the fifth column of this table the number N_{RK} of order conditions for classical RK methods.

q	s	n	N_{CG}	N_{RK}
4	5	15	13	8
5	9	45	38	17
6	16	136	113	37

We have found many methods of order four and five. We have also found coefficients that satisfy the order six conditions approximately up to $6 - 7$ significant digits. In Appendix A we present some examples of these methods with moderately sized coefficients. For the method of order four the abscissae c_i, $i = 1, 2, \dots, 5$, are in the interval $[0, 1]$.

The numerical experiments presented in Section 4 confirm that all these methods achieve the expected order of accuracy.

4 Numerical verification of order

We use the spinning top [13], [8] as a test problem to verify the order of the methods presented in Appendix A. The top is modeled as a rigid body that is suspended at one point, and has the Lie group $G = \mathrm{SO}(3) \times \mathfrak{so}(3)$ as its configuration space. The Lie algebra of G is isomorphic to $\mathfrak{so}(3) \times \mathfrak{so}(3)$.

A point in the configuration space G is described as a pair $\big(B(t), \omega(t)\big)$. The orthogonal matrix with determinant one, $B(t)$, transforms the top from local to global coordinates. The skew-symmetric matrix $\omega(t)$ is the angular momentum of the spinning top.

The governing equations are described in e.g. [13], [8], and we use as initial values

$$B(0) = B_0 = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & \cos(\phi) & \sin(\phi) \\ 0.0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

and

$$\omega(0) = \omega_0 = \begin{bmatrix} 0.0 & -1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix},$$

with $\phi = \pi/16$. We integrate from $t_0 = 0$ to $t_{\rm end} = 1.0$ with constant stepsize $h = 0.1$.

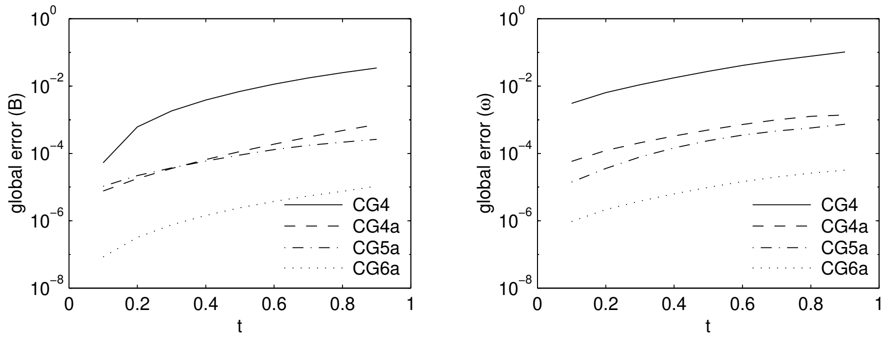

Figure 4.1: The Crouch–Grossman method applied to the spinning top problem. The graphs show the global error of the two solution components as a function of time.

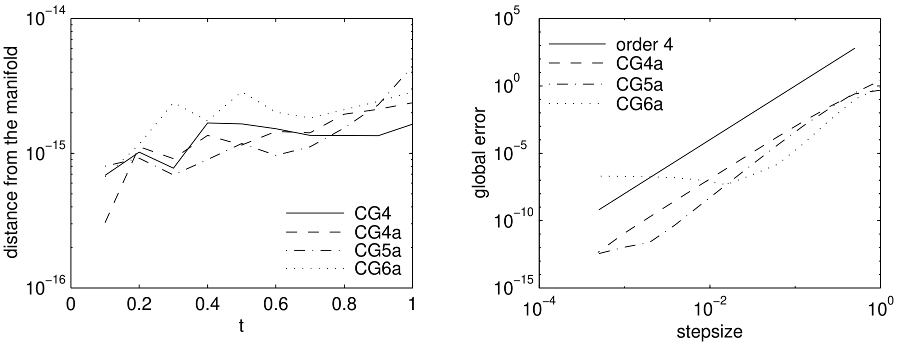

Figure 4.2: The Crouch-Grossman method applied to the spinning top problem. The left graph shows the distance from the configuration space G as a function of time, while the right graph shows the global error as a function of the stepsize. The reference line (solid) in the right graph indicates order four. We show only the results for the first component $B(t)$.

The integrators are denoted by **CG4a** (Example 1), **CG5a** (Example 2) and **CG6a** (Example 3). We compare these schemes with the fourth order scheme, denoted by **CG4**, proposed in [13]. Figure 4.1 shows the global error of the two solution components as a function of time. It is clear that the fourth order scheme proposed in Example 1 is superior to the **CG4** scheme derived in [13]. We also see that, although the **CG6a** scheme satisfies the order conditions only approximately (6-7 decimals), the results produced while integrating the above test problem is significantly better than the results from the other schemes. Figure 4.2 shows, on the left graph, the distance between the numerical solution and the configuration space G as a function of time (2-norm), while the right graph shows the global error as a function of the stepsize. The slopes of the graphs give the approximation order. The **CG4a** and **CG5a** methods are indeed of order four and five, respectively, but we see that the sixth order method only approximately satisfies the order conditions.

Acknowledgements. Arne Marthinsen and Brynjulf Owren are grateful to the Department of Mathematics, Arizona State University, for its hospitality during their stay in the Spring of 1998.

REFERENCES

1. V.I. Arnold, Mathematical Methods of Classical Mechanics, Springer Verlag, New York 1989.
2. M.P. Calvo, A. Iserles and A. Zanna, Numerical solution of isospectral flows, Math. Comp. 66(1997), 1461–1486.
3. P.E. Crouch and R. Grossman, Numerical intgration of ordinary differential equations on manifolds, J. Nonlinear Sci. 3(1993), 1–33.
4. P.E. Crouch and Y. Yan, On the numerical integration of the rolling ball equations using geometrically exact algorithms, Mech. Struct. & Mach. 23(1995), 257–272.
5. P.E. Crouch, Y. Yan and R. Grossman, On the numerical integration of the dynamic attitude equations, manuscript.
6. J.E. Dennis, Jr. and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Classics in Applied Mathematics, SIAM, Philadelphia 1996.
7. L. Dieci, R.D. Russel and E.S. Van Vleck, Unitary integrators and application to continuous orthonormalization techniques, SIAM J. Numer. Anal. 31(1994), 261–281.
8. K. Engø and A. Marthinsen, Modeling and solution of some mechanical problems on Lie groups, Multibody System Dynamics 2(1998), 71–88.
9. C. Führer and B.J. Leimkuhler, Numerical solution of differential-algebraic equations for constrained mechanical motion, Numer. Math. 59(1991), 55–69.
10. H. Munthe-Kaas, Lie-Butcher theory for Runge-Kutta methods, BIT 35(1995), 572–587.
11. H. Munthe-Kaas, Runge-Kutta methods on Lie groups, BIT 38(1998), 90–110.
12. H. Munthe-Kaas, High order Runge–Kutta methods on manifolds, Appl. Numer. Math., 29(1999), 115–127.
13. B. Owren and A. Marthinsen, Runge–Kutta methods adapted to manifolds and based on rigid frames, BIT 39(1999), 116–142.

14. B. Owren and B. Welfert, The Newton iteration on Lie groups, Preprint Numerics No. 3/1996, Norwegian University of Science and Technology, Trondheim, Norway 1996.
15. W.C. Rheinboldt, Differential-algebraic systems as differential equations on manifolds, Math. Comp. 43(1984), 473–482.
16. J.M. Sanz-Serna and M.P. Calvo, Numerical Hamiltonian Problems, Chapman & Hall, London, New York 1994.
17. J.C. Simo and K.K. Wong, Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum, Internat. J. Numer. Methods Engrg. 31(1991), 19–52.
18. S. Tracogna and B. Welfert, Notes on riffle shuffles, manuscript.

A Examples of methods

We present below examples of RK methods of Crouch-Grossman type of order $q = 4$ with $s = 5$ stages, order $q = 5$ with $s = 9$ stages, and order $q = 6$ with $s = 16$ stages. All coefficients have been truncated to 16 digits after the decimal point.

Example 1: q=4, s=5 (CG4a).

$$\begin{aligned}
a_{2,1} &= 0.8177227988124852, & a_{3,1} &= 0.3199876375476427,\\
a_{3,2} &= 0.0659864263556022, & a_{4,1} &= 0.9214417194464946,\\
a_{4,2} &= 0.4997857776773573, & a_{4,3} &= -1.0969984448371582,\\
a_{5,1} &= 0.3552358559023322, & a_{5,2} &= 0.2390958372307326,\\
a_{5,3} &= 1.3918565724203246, & a_{5,4} &= -1.1092979392113465,
\end{aligned}$$

$$\begin{aligned}
b_1 &= 0.1370831520630755, & c_1 &= 0.0,\\
b_2 &= -0.0183698531564020, & c_2 &= 0.8177227988124852,\\
b_3 &= 0.7397813985370780, & c_3 &= 0.3859740639032449,\\
b_4 &= -0.1907142565505889, & c_4 &= 0.3242290522866937,\\
b_5 &= 0.3322195591068374, & c_5 &= 0.8768903263420429.
\end{aligned}$$

Example 2: q=5, s=9 (CG5a).

$$\begin{aligned}
a_{2,1} &= 0.5119828014137712, & a_{3,1} &= 0.1256672120643583,\\
a_{3,2} &= 0.7516202380829815, & a_{4,1} &= -0.1859318985660963,\\
a_{4,2} &= 0.9805295406563367, & a_{4,3} &= -0.5333302514616872,\\
a_{5,1} &= 0.1693767277911736, & a_{5,2} &= 0.9505578305836004,\\
a_{5,3} &= -0.2601241719456298, & a_{5,4} &= -0.2298819100175240,\\
a_{6,1} &= -0.3672779810475256, & a_{6,2} &= -0.0086029950554851,\\
a_{6,3} &= -0.0010336337263253, & a_{6,4} &= 0.4747844514543335,\\
a_{6,5} &= -0.1827675787966842, & a_{7,1} &= -0.1967584902954152,\\
a_{7,2} &= -0.0841805485664162, & a_{7,3} &= -0.0269530236366960,\\
a_{7,4} &= 0.7559747536983863, & a_{7,5} &= -0.0238706198984107,\\
a_{7,6} &= 0.0482925299135956, & a_{8,1} &= -0.5309452158934434,\\
a_{8,2} &= 0.0197456651284920, & a_{8,3} &= -0.0890782597312347,\\
a_{8,4} &= 1.1112114193183167, & a_{8,5} &= -0.2549231869664070,\\
a_{8,6} &= 0.0690881269948897, & a_{8,7} &= -0.0190658555145840,\\
a_{9,1} &= 0.0, & a_{9,2} &= -0.1116041993954347,\\
a_{9,3} &= 0.0007837329078394, & a_{9,4} &= 0.1273989332794543,\\
a_{9,5} &= -0.2951798037735399, & a_{9,6} &= -0.5806798044068681,\\
a_{9,7} &= -0.3715982236980106, & a_{9,8} &= 1.1170906343009428,
\end{aligned}$$

$$
\begin{aligned}
b_1 &= 0.7927513978455029, & c_1 &= 0.0,\\
b_2 &= 0.0, & c_2 &= 0.5119828014137712,\\
b_3 &= 0.3490698579749237, & c_3 &= 0.8772874501473397,\\
b_4 &= 0.3139686995007973, & c_4 &= 0.2612673906285532,\\
b_5 &= -0.4266876632081064, & c_5 &= 0.6299284764116202,\\
b_6 &= -0.8094238190622946, & c_6 &= -0.0848977371716866,\\
b_7 &= 1.3450516465346317, & c_7 &= 0.4725046012150439,\\
b_8 &= -0.9242094915998407, & c_8 &= 0.3060326933360294,\\
b_9 &= 0.3594793720143862, & c_9 &= -0.1137887307856168.
\end{aligned}
$$

Example 3: q=6, s=16 (CG6a).

$$
\begin{aligned}
a_{2,1} &= 0.08286617, & a_{3,1} &= -0.13772068, & a_{3,2} &= 0.36716433,\\
a_{4,1} &= -0.25167910, & a_{4,2} &= 0.38141457, & a_{4,3} &= -0.00388100,\\
a_{5,1} &= -0.73989121, & a_{5,2} &= 2.02878622, & a_{5,3} &= -0.30650542,\\
a_{5,4} &= -0.90079679, & a_{6,1} &= -0.72119356, & a_{6,2} &= 0.31589164,\\
a_{6,3} &= 1.20250077, & a_{6,4} &= -1.03226953, & a_{6,5} &= 0.96314218,\\
a_{7,1} &= -0.80706244, & a_{7,2} &= 0.78434507, & a_{7,3} &= 0.71120581,\\
a_{7,4} &= -0.69119029, & a_{7,5} &= 0.65089413, & a_{7,6} &= -0.03269763,\\
a_{8,1} &= 0.18251540, & a_{8,2} &= -0.33594286, & a_{8,3} &= 0.67396446,\\
a_{8,4} &= -0.19916288, & a_{8,5} &= 0.17904428, & a_{8,6} &= -0.00559558,\\
a_{8,7} &= 0.04781843, & a_{9,1} &= 0.59829475, & a_{9,2} &= -0.91755067,\\
a_{9,3} &= 0.63091080, & a_{9,4} &= 0.13050267, & a_{9,5} &= 0.06572704,\\
a_{9,6} &= 0.53610222, & a_{9,7} &= 0.10465447, & a_{9,8} &= -0.21875824,\\
a_{10,1} &= 0.81456689, & a_{10,2} &= -1.01083047, & a_{10,3} &= 0.17884608,\\
a_{10,4} &= 0.46716208, & a_{10,5} &= -0.09397705, & a_{10,6} &= -0.13875871,\\
a_{10,7} &= 0.24366855, & a_{10,8} &= 0.16005871, & a_{10,9} &= 0.24743348,\\
a_{11,1} &= 0.06334807, & a_{11,2} &= 0.83943479, & a_{11,3} &= 0.46681376,\\
a_{11,4} &= -2.85588860, & a_{11,5} &= -0.03478793, & a_{11,6} &= -0.32005164,\\
a_{11,7} &= 0.25715271, & a_{11,8} &= -0.57798558, & a_{11,9} &= -1.69123378,\\
a_{11,10} &= 2.38817986, & a_{12,1} &= -1.68249569, & a_{12,2} &= -1.52757929,\\
a_{12,3} &= 2.66247175, & a_{12,4} &= 2.42971459, & a_{12,5} &= -2.95203730,\\
a_{12,6} &= 2.50616251, & a_{12,7} &= -2.38134644, & a_{12,8} &= 0.83864638,\\
a_{12,9} &= 1.12787276, & a_{12,10} &= -2.65439436, & a_{12,11} &= 0.09419125,\\
a_{13,1} &= -0.89047469, & a_{13,2} &= 1.79715345, & a_{13,3} &= -0.67220608,\\
a_{13,4} &= 0.00835048, & a_{13,5} &= -0.04180624, & a_{13,6} &= -0.51608521,\\
a_{13,7} &= 1.40206288, & a_{13,8} &= -0.26251475, & a_{13,9} &= 0.09339365,\\
a_{13,10} &= 0.01178571, & a_{13,11} &= 0.24955372, & a_{13,12} &= -0.23815139,\\
a_{14,1} &= 0.09422289, & a_{14,2} &= 0.77774172, & a_{14,3} &= -1.41299444,\\
a_{14,4} &= 1.34960405, & a_{14,5} &= -0.76151845, & a_{14,6} &= 1.68421875,\\
a_{14,7} &= -1.48472940, & a_{14,8} &= 0.88771458, & a_{14,9} &= 0.23803071,\\
a_{14,10} &= -0.29527691, & a_{14,11} &= 0.17556226, & a_{14,12} &= -0.15270356,\\
a_{14,13} &= -0.15445516, & a_{15,1} &= -1.59334123, & a_{15,2} &= 0.71526521,\\
a_{15,3} &= -0.52019794, & a_{15,4} &= 0.64373590, & a_{15,5} &= 1.36060458,\\
a_{15,6} &= -0.97980269, & a_{15,7} &= 2.18658056, & a_{15,8} &= -1.87240277,\\
a_{15,9} &= 2.09554059, & a_{15,10} &= 0.01098630, & a_{15,11} &= 0.46073164,\\
a_{15,12} &= -0.42435386, & a_{15,13} &= -1.16128741, & a_{15,14} &= -0.63539680,\\
a_{16,1} &= -0.01331208, & a_{16,2} &= 0.79340651, & a_{16,3} &= -0.05910596,\\
a_{16,4} &= 0.19807094, & a_{16,5} &= -0.60401546, & a_{16,6} &= -0.18637406,\\
a_{16,7} &= 0.08476021, & a_{16,8} &= 0.43807966, & a_{16,9} &= 0.44989686,
\end{aligned}
$$

$$
\begin{aligned}
a_{16,10} &= 0.07852516, & a_{16,11} &= 0.14867187, & a_{16,12} &= -0.13575501,\\
a_{16,13} &= -0.17541368, & a_{16,14} &= -0.11323535, & a_{16,15} &= -0.03842925,
\end{aligned}
$$

$$
\begin{aligned}
b_1 &= 0.02632004, & c_1 &= 0.0, & b_2 &= 0.12717426,\\
c_2 &= 0.08286617, & b_3 &= 0.24411184, & c_3 &= 0.22944365,\\
b_4 &= -0.01270790, & c_4 &= 0.12585446, & b_5 &= -0.01284543,\\
c_5 &= 0.08159279, & b_6 &= -0.00955321, & c_6 &= 0.72807150,\\
b_7 &= -0.02228604, & c_7 &= 0.61549466, & b_8 &= 0.39395605,\\
c_8 &= 0.54264125, & b_9 &= 0.28291010, & c_9 &= 0.92988304,\\
b_{10} &= -0.14261072, & c_{10} &= 0.86816956, & b_{11} &= 0.00035311,\\
c_{11} &= -1.46501832, & b_{12} &= -0.00029156, & c_{12} &= -1.53879385,\\
b_{13} &= -0.07689126, & c_{13} &= 0.94106152, & b_{14} &= -0.08578008,\\
c_{14} &= 0.94541705, & b_{15} &= -0.01345727, & c_{15} &= 0.28666209,\\
b_{16} &= 0.30159809, & c_{16} &= 0.86577035.
\end{aligned}
$$