B-series and Applications

The order of Runge–Kutta and General Linear Methods

John Butcher

University of Auckland
butcher@math.auckland.ac.nz

July 2022

1. Introduction (Contents)

Examples of initial value problems

$$y' = 0, \quad y(0) = 1, \quad \text{motionless}$$

$$y' = 1, \quad y(0) = 0, \quad \text{constant motion}$$

$$y' = y, \quad y(0) = 1, \quad \text{exponential growth}$$

$$y' = -y, \, y(0) = 1, \quad \text{exponential decay}$$

$$\begin{aligned} y' &= z, & y(0) &= 1, \\ z' &= -y, & z(0) &= 0, \end{aligned}$$

harmonic oscillator

$$\begin{aligned} y' &= z, & y(0) &= 1, \\ z' &= -\sin(y), & z(0) &= 0, \end{aligned}$$

simple pendulum

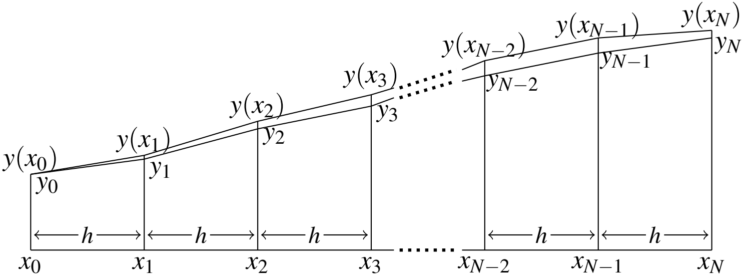
$$\begin{aligned} y'' &= \frac{-y}{(y^2+z^2)^{3/2}}, \\ z'' &= \frac{-z}{(y^2+z^2)^{3/2}}, \end{aligned}$$

inverse square force

$$y' = \frac{y-x}{y+x}.$$

Runge test problem

What we want from numerical methods



- We want high order p

$$y(x_1) - y_1 = O(h^{p+1}),$$
$$y(x_N) - y_N = O(h^p).$$

- We want low computational cost
- We want stability

What methods are available?

1. Euler method

$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1})$$

2. Runge–Kutta methods

$$\begin{aligned}
Y_1 &= y_{n-1}, & F_1 &= f(x_{n-1}, Y_1),\\
Y_2 &= y_{n-1} + \tfrac{1}{3}hF_1, & F_2 &= f(x_{n-1} + \tfrac{1}{3}h, Y_2),\\
Y_3 &= y_{n-1} \qquad\quad + \tfrac{2}{3}hF_2, & F_3 &= f(x_{n-1} + \tfrac{2}{3}h, Y_3),\\
y_n &= y_{n-1} + \tfrac{1}{4}hF_1 \qquad + \tfrac{3}{4}hF_3. & &
\end{aligned}$$

3. Linear Multistep methods

$$y_n = y_{n-1} + \tfrac{3}{2}hf(y_{n-1}) - \tfrac{1}{2}hf(y_{n-2})$$

4. General Linear methods

2. Runge–Kutta methods and B-series (Contents)

Initial value problem

Given an initial value problem

$$y'(x) = f(y(x)),$$
$$y(x_0) = y_0,$$

and a unit time distance h, many questions can be asked, such as

- What is the solution at $x = x_0 + h$?
- What is the approximate solution at $x = x_0 + h$, computed by a Runge–Kutta method?

The answers to these questions can be typically written as a series (a "B-series") of the form

$$y_0 + \sum_{\mathtt{t}} a(\mathtt{t}) \frac{h^{|\mathtt{t}|}}{\sigma(\mathtt{t})} F(\mathtt{t})(y_0),$$

where

- $\mathtt{t}$ is a rooted tree
- $a(\mathtt{t})$ is a sequence characteristic of the question being asked
- $\sigma(\mathtt{t})$ is the "symmetry" of $\mathtt{t}$
- $F(\mathtt{t})$ is an "elementary differential"

One of our aims will be to become familiar with the quantities appearing in this series. This is postponed to Sections 3 and 4

Coefficient sequences

The coefficient sequence a, for two important cases, are

- Exact solution:
  $$a(\mathtt{t}) = \mathtt{t}!^{-1},$$
- Runge–Kutta approximation:
  $$a(\mathtt{t}) = \Phi(\mathtt{t}),$$

Order conditions

By comparing these two series for terms for which $|\mathtt{t}| \le p$, we find the condition for a Runge–Kutta method to have order p.

$$\Phi(\mathtt{t}) = \frac{1}{\mathtt{t}!}, \qquad |\mathtt{t}| \le p.$$

We will recall this result and the meaning of $\Phi(\mathtt{t})$ in Section 4

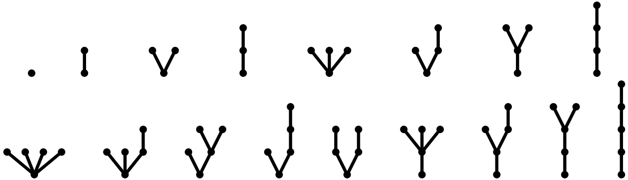
3. Introduction to trees (Contents)

As in the previous section, t will denote a tree, with T the set of all trees.

Order, symmetry and factorial

The order of a tree is the number of vertices.

Here is a list of the trees up to order 5:



For a given tree t, the order is written as $|t|$.

The symmetry of $\mathtt{t}$, denoted by $\sigma(\mathtt{t})$, is the order of its symmetry group.

In graph theory this is the order of the permutation group of the set of vertices, subject to leaving the set of edges invariant.

The factorial of a tree, denoted by $\mathtt{t}!$, is a generalisation of the factorial of a positive integer.

It will be defined by induction after we have seen some examples.

3. Introduction to trees (3)

Order, symmetry and factorial, to order 4, are shown below

$\|t\|$	t	$\sigma(t)$	$t!$
1		1	1
2		1	2
3		2	3
3		1	6
4		6	4
4		1	8
4		2	12
4		1	24

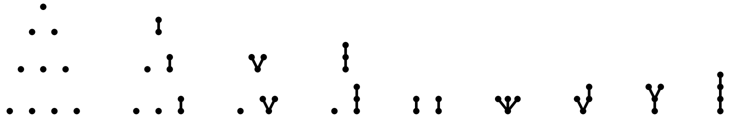
Forests

A forest is a juxtaposition of trees; that is, a number of trees written in a row.

The order of a forest is the sum of the orders of the constituent trees.

The empty forest will be denoted by 1, as the identity of the monoid of forests.

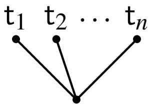
Forests of orders 1 to 4 are:



3. Introduction to trees (5)

Tree recursion

Generating trees by induction from $\tau := \bullet$ is possible: Let $\mathtt{t}_1\mathtt{t}_2\cdots\mathtt{t}_n$ be a forest; then $[\mathtt{t}_1\mathtt{t}_2\cdots\mathtt{t}_n]$ is defined to be the tree



The factorial can now be defined as

$$\tau! = 1,$$

$$\mathtt{t}! = |\mathtt{t}|\prod_{i=1}^{n}\mathtt{t}_i!, \qquad \mathtt{t} = [\mathtt{t}_1\mathtt{t}_2\cdots\mathtt{t}_n]$$

Repeat a previous table, using notations for trees based on iterated use of τ, and $[\ \cdot\ ]$

$\|t\|$	t	$\sigma(t)$	$t!$
1	τ	1	1
2	$[\tau]$	1	2
3	$[\tau^2]$	2	3
3	$[_2\tau]_2$	1	6
4	$[\tau^3]$	6	4
4	$[\tau[\tau]]$	1	8
4	$[_2\tau^2]_2$	2	12
4	$[_3\tau]_3$	1	24

Note the notation for repeated trees and brackets.

3. Introduction to trees (7)

Stump recursion

The notation $[\mathtt{t}_1\mathtt{t}_2\cdots\mathtt{t}_n]$ can also be written in Polish notation (invented by logician Łukasiewicz) using the prefix operator τ_n.

τ_n is interpreted as a root with n unfilled valencies which acts on n trees so that

$$\tau_n\mathtt{t}_1\mathtt{t}_2\cdots\mathtt{t}_n = [\mathtt{t}_1\mathtt{t}_2\cdots\mathtt{t}_n]$$

Examples are

 $= \tau$

 $= \tau_1\tau$

 $= \tau_2\tau^2$

 $= \tau_1\tau_1\tau$

 $= \tau_3\tau^3$

 $= \tau_2\tau\tau_1\tau$

 $= \tau_1\tau_2\tau^2$

 $= \tau_1\tau_1\tau_1\tau$

Subtrees and supertrees

The relationship $\mathtt{t}' \leq t$ means that $\mathtt{t}'$ can be formed from $\mathtt{t}$ by discarding ("pruning") some of the vertices or, alternatively, $\mathtt{t}$ can be formed from $\mathtt{t}'$ by splicing on additional vertices.

If $\mathtt{t}' \leq t$, then $\mathtt{t} \setminus \mathtt{t}'$ is the forest of discarded subtrees when $\mathtt{t}'$ is formed from $\mathtt{t}$.

If there is more than one way of pruning $\mathtt{t}$ to form $\mathtt{t}'$, then $\mathtt{t} \setminus \mathtt{t}'$ will be a linear combination of forests.

Up to $|\mathsf{t}| \leq 4$, the value of $\mathsf{t} \setminus \mathsf{t}'$ is as follows

t \ t′	•							
•	1							
	•	1						
	••	2 •	1					
		•		1				
	•••	3 ••	3 •		1			
	•	•• + 	•	•		1		
		••		2 •			1	
				•				1

Algebraic structures

Let B be the set of mappings $\mathsf{T} \to \mathbb{R}$. If $a, b \in \mathsf{B}$ and ab is defined by

$$(ab)(\mathsf{t}) = a(\mathsf{t}) + \sum_{\mathsf{t} \leq \mathsf{t}} a(\mathsf{t} \setminus \mathsf{t}')b(\mathsf{t}'), \qquad (*)$$

then B is a group.

By replacing T by $\mathsf{T}^{\#} = \mathsf{T} \cup \{\varnothing\}$, and defining $a(\varnothing) = b(\varnothing) = 1$ for $a, b \in \mathsf{B}$ and replacing $(*)$ by

$$(ab)(\mathsf{t}) = a(\mathsf{t})b(\varnothing) + \sum_{\mathsf{t} \leq \mathsf{t}} a(\mathsf{t} \setminus \mathsf{t}')b(\mathsf{t}'), \qquad (\dagger)$$

$$(ab)(\mathsf{t}) = a(\mathsf{t})b(\varnothing) + \sum_{\mathsf{t} \le \mathsf{t}} a(\mathsf{t} \setminus \mathsf{t}')b(\mathsf{t}'), \qquad (\dagger)$$

Thus $b(\mathsf{t}) \mapsto (ab)(\mathsf{t})$ becomes a linear operation, and B is converted to an algebra B^*.

To verify the linear nature of this operation, write ($\dagger$), in the form

$$ab = \Lambda(a)b.$$

Note that ab is group multiplication and $\Lambda(a)b$ is matrix times vector multiplication.

To form the infinite matrix $\Lambda(a)$:

1. Expand the given table of $\mathtt{t} \setminus \mathtt{t}'$ to all trees
2. Insert an initial row and an initial column corresponding to the empty tree $\varnothing$
3. Replace each tree $\mathtt{t}$ by $a(\mathtt{t})$

$$\Lambda(a) = \begin{bmatrix}
1 & & & & & & & & & \cdots \\
a(\tau) & 1 & & & & & & & & \cdots \\
a([\tau]) & a(\tau) & 1 & & & & & & & \cdots \\
a([\tau^2]) & a(\tau)^2 & 2a(\tau) & 1 & & & & & & \cdots \\
a([[\tau]]) & a([\tau]) & a(\tau) & 0 & 1 & & & & & \cdots \\
a([\tau^3]) & a(\tau)^3 & 3a(\tau)^2 & 3a(\tau) & 0 & 1 & & & & \cdots \\
a([\tau[\tau]]) & a(\tau)a([\tau]) & a(\tau)^2 + a([\tau]) & a(\tau) & a(\tau) & 0 & 1 & & & \cdots \\
a([[\tau^2]]) & a([\tau^2]) & a(\tau)^2 & 0 & 2a(\tau) & 0 & 0 & 1 & & \cdots \\
a([[[\tau]]]) & a([[\tau]]) & a([\tau]) & 0 & . & 0 & 0 & 0 & 1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix}$$

4. Introduction to B-series (Contents)

Elementary differentials

By repeated differentiation we can find the higher derivatives of y satisfying $\frac{d}{dx} = f(y(x))$. These are

$$
\begin{aligned}
y' &= f, \\
y'' &= f'f, \\
y^{(3)} &= f''ff + f'f'f, \\
y^{(4)} &= f^{(3)}fff + 3f''ff'f + f'f''ff + f'f'f'f,
\end{aligned}
$$

where the linear operator f' and the multilinear operators f'' and $f^{(3)}$ act according to the rules of Polish notation.

This motivates the definition:

Given $\mathtt{t} = [\mathtt{t}_1 \mathtt{t}_2 \cdots \mathtt{t}_n]$,

$$F(\mathtt{t}) = f^{(n)} F(\mathtt{t}_1) F(\mathtt{t}_2) \cdots F(\mathtt{t}_n)$$

with $F(\tau) = f$.

Unless otherwise indicated, $f, f', \ldots$ are evaluated at y_0.
It is interesting to compare the formulae

$$\begin{array}{ccccccc} \mathtt{t} & = \tau_n & \mathtt{t}_1 & \mathtt{t}_2 & \cdots & \mathtt{t}_n \\ F(\mathtt{t}) & = f^{(n)} & F(\mathtt{t}_1) & F(\mathtt{t}_2) & \cdots & F(\mathtt{t}_n) \end{array}$$

B-series vector

Given a trio (f, y_0, h), define the (infinite) B-series vector

$$B_h = [y_0, hF(\tau), h^2F([\tau]), \tfrac{1}{2}h^3F([\tau^2]), \ldots,],$$

with typical component

$$B_h(\mathsf{t}) = \frac{h^{|\mathsf{t}|}}{\sigma(\mathsf{t})}F(\mathsf{t})$$

If y_0 is replaced by y_1, then we can write $B_h y_1$.

1

Coefficient vector

To write down a B-series we need a sequence of coefficients to combine with the B-series basis vector

$$a^{\intercal} = [a(\varnothing), a(\tau), a([\tau]), a([\tau^2]), \ldots,].$$

4. Introduction to B-series (4)

B-series

The complete series is

$$B_h a = a(\varnothing) y_0 + \sum_{\mathsf{t}} \frac{h^{|\mathsf{t}|} a(\mathsf{t})}{\sigma(\mathsf{t})} F(\mathsf{t})$$

Compositions

Consider mappings defined by B-series as follows

$$y_0 \xrightarrow{B_h a} y_1 \xrightarrow{B_h b} y_2$$

so that

$$\begin{aligned} y_1 &= B_h a y_0, \\ y_2 &= B_h b y_1 \end{aligned}$$

We want to construct B-series coefficients which give the composed mapping.

This is found by calculating ab in the B-group.

That is

$$ab = \Lambda(a)b$$

and the composition formula is

$$B_h b B_h a = B_h ab$$

Special members and a subgroup

Define $\mathsf{E} \in \mathsf{B}$ by

$$\mathsf{E}(\mathsf{t}) = \mathsf{t}!^{-1},$$

so that $B_h\mathsf{E}$ is the B-series for th flow of f.

Define $\mathsf{D} \in \mathsf{B}^*$ by

$$\begin{aligned} \mathsf{D}(\varnothing) &= 0, \\ \mathsf{D}(\tau) &= 1, \\ \mathsf{D}(\mathsf{t}) &= 0, \qquad |\mathsf{t}| > 1, \end{aligned}$$

so that

$$B_h a\mathsf{D} = hf(B_h a).$$

Define $1+\mathsf{O}_{p+1}$ as the normal subgroup of B such that

$$a(\mathsf{t})=0, \qquad a\in 1+\mathsf{O}_{p+1}, \quad |\mathsf{t}|\le p.$$

Note that O_{p+1} is an ideal in B^*.

The quotient group $\mathsf{B}/(1+\mathsf{O}_{p+1})$ is isomorphic to the group in which mappings $\mathsf{T}\to\mathbb{R}$ are restricted to trees with $|\mathsf{t}|\le p$.

Application to Runge–Kutta methods

An application of these is in the analysis of Runge–Kutta methods.

Corresponding to

$$\begin{aligned} Y_i &= y_0 + h\sum_j a_{ij} f(Y_j), \\ y_1 &= y_0 + h\sum_i b_i f(Y_i) = y(x_0+h) + O(h^{p+1}), \end{aligned}$$

we write

$$\begin{aligned} \eta &= \mathbf{1} + A\eta\mathsf{D}, & (*) \\ \mathsf{E} &= 1 + b^{\mathsf{T}}\eta\mathsf{D} + \mathsf{O}_{p+1}. \end{aligned}$$

We will widen the applications of this type of analysis in Section 5. In the meantime, we will find a recursive formula for $\eta(\mathsf{t})$, satisfying $(*)$ and $\Phi(\mathsf{t}) := b^{\mathsf{T}}\eta\mathsf{D}$.

Elementary weights

The values of $\Phi(\mathsf{t})$ are known as “Elementary weights”. We will first find $\eta(\mathsf{t})$ and $\eta\mathsf{D}(\mathsf{t})$ using the fact that

$$\eta\mathsf{D}([\mathsf{t}_1\mathsf{t}_2\cdots\mathsf{t}_n]) = \prod_{i=1}^{n}\eta(\mathsf{t}_i), \qquad (\dagger)$$

where the product is componentwise.

Starting with $\eta(\tau) = A\mathbf{1}$, the use of $(\dagger)$ enables η and $\eta\mathsf{D}(\mathsf{t})$ to be found for all trees using

$$\eta([\mathsf{t}_1\mathsf{t}_2\cdots\mathsf{t}_n]) = A\prod_{i=1}^{n}\eta(\mathsf{t}_i).$$

Finally, we have $\Phi(\tau) = b^{\top}\mathbf{1}$ and

$$\Phi([\mathsf{t}_1\mathsf{t}_2\cdots\mathsf{t}_n]) = b^{\top}\prod_{i=1}^{n}\eta(\mathsf{t}_i).$$

5. General Linear Methods and B-series (Contents)

Formulation

We will use the GLM formulation based on a partitioned $(s+r)\times(s+r)$ matrix

$$M = \begin{bmatrix} A & U \\ B & V \end{bmatrix}$$

with stages Y_i, and output components $y_i^{[n]}$, computed by

$$Y_i = h\sum_{j=1}^{s} a_{ij} f(Y_j) + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}$$

$$y_i^{[n]} = h\sum_{j=1}^{s} b_{ij} f(Y_j) + \sum_{j=1}^{r} v_{ij} y_j^{[n-1]}$$

Order analysis

For $n=1$, write $B_h\xi$ to represent $y^{[0]}$ and $B_h\eta$ to represent Y.

We then have

$$\eta = A\eta\mathsf{D} + U\xi,$$

$$\xi\mathsf{E} = B\eta\mathsf{D} + V\xi + \mathsf{O}_{p+1}.$$

Starting and finishing methods

The asymptotic error of a GLM depends not only on the coefficient matrices (A, U, B, V) but also on

1. how $y^{[0]}$ is computed from y_0
   the "starting method" S_h and
2. how $y(x_n)$ is recovered from $y^{[n]}$
   the "finishing method" F_h

If these are related by $F_h \circ S_h = \text{id}$, then it remains to consider S_h.

However, S_h is represented as the B-series ξ in this analysis and order can be defined in a relative sense, and an absolute sense

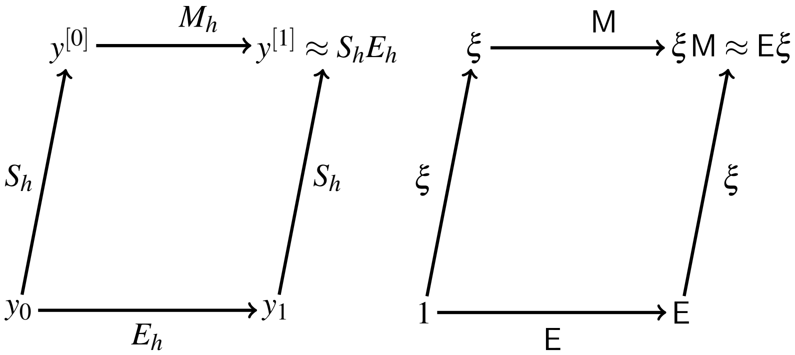
Definition

$M = (A, U, B, V)$ has order p relative to ξ if

$$\eta = A\eta \mathsf{D} + U\xi,$$
$$\xi \mathsf{E} = B\eta \mathsf{D} + V\xi + \mathsf{O}_{p+1}.$$

Definition

$M = (A, U, B, V)$ has order p if there exists ξ such that M has order p, relative to ξ.



Effective order of Runge–Kutta methods

$$(bab^{-1})(\mathsf{t}) = \mathsf{t}!^{-1}, \qquad |\mathsf{t}| \le p.$$

References

1. Butcher, J.C.: ***An algebraic theory of integration methods,*** Math. Comp. **26**, 79–106 (1972)

2. Butcher, J.C.: ***B-series: Algebraic Analysis of Numerical Methods,*** Springer-Verlag (2021)

3. Hairer, E. and Wanner, G.: ***Multistep-multistage-multiderivative methods for ordinary differential equations,*** Computing **11**, 287–303 (1973)

4. Hairer, E. and Wanner, G.: ***On the Butcher group and general multi-value methods,*** Computing **13**, 1–15 (1974)

5. Butcher, J.C.: ***B-series: Algebraic Analysis of Numerical Methods,*** Springer-Verlag (2021)