Martin–Luther–Universität
Halle–Wittenberg
Institut für Mathematik

Strong stability preserving explicit peer methods

Zoltán Horváth, Helmut Podhaisky and Rüdiger Weiner

Report No. 04 (2014)

Editors:
Professors of the Institute for Mathematics, Martin-Luther-University Halle-Wittenberg.

Electronic version: see `http://www2.mathematik.uni-halle.de/institut/reports/`

Strong stability preserving explicit peer methods

Zoltán Horváth, Helmut Podhaisky and Rüdiger Weiner

Report No. 04 (2014)

Zoltán Horváth
Széchenyi István Egyetem
Egyetem tér 1
H-9026 Győr, Hungary
Email: horvathz@sze.hu

Helmut Podhaisky
Rüdiger Weiner
Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät II
Institut für Mathematik
Theodor-Lieser-Str. 5
D-06120 Halle/Saale, Germany
Email: helmut.podhaisky@mathematik.uni-halle.de
ruediger.weiner@mathematik.uni-halle.de

Strong stability preserving explicit peer methods

Zoltán Horváth [*], Helmut Podhaisky [†], Rüdiger Weiner [‡]

December 19, 2014

Abstract

In this paper we study explicit peer methods up to order $p = 13$ which have the strong stability preserving (SSP) property. These methods have the favourable property of a high stage order. The effective SSP coefficient is maximized by solving a nonlinear constraint optimization problem numerically to high precision. The coefficient matrices of the optimized methods are sparse in a very structured way.

MSC: 65L05, 65L06
Keywords: SSP, positivity, explicit peer methods, nonstiff initial value problems

1 Introduction

The concept of strong stability preserving (SSP) methods was introduced by Shu and Osher [13] for the numerical solution of a hyperbolic conservation law. Discretizing the spatial derivatives with the method of lines (MOL) yields a system of ordinary differential equations

$$y' = f(t, y), \quad y(t_0) = y_0 \in \mathbb{R}^n, \quad t \in [t_0, t_e]. \tag{1}$$

We assume that spatial discretization is chosen such that the semidiscrete solution satisfies the strong stability property

$$\|y + hf(t, y)\| \leq \|y\| \text{ for all } y \in \mathbb{R}^n \text{ and } h \leq h_E, \tag{2}$$

where $\|\cdot\|$ represents a norm or convex functional. The significance of this condition is that $\|\cdot\|$ will be non-increasing for approximations computed with the explicit Euler method with $h \leq h_E$.
Naturally, one is interested in higher order methods satisfying an analogue of (2) for the numerical solution for step sizes $h \leq C \cdot h_E$. The positive constant C is called the SSP coefficient of the method. The order of explicit Runge-Kutta methods which preserve strong stability cannot exceed four [11], furthermore their stage-order is only one. Explicit linear multistep SSP methods have no known order bound, however they need a large number of steps for higher order [5]. The deficiencies of classical methods have created a recent interest in high order General Linear Methods (GLM, [2], [8]) which have the SSP property, e.g. [14], [9]. In [3] strong stability preserving GLM up to stage order four are considered. Bresten et al. [1] construct multistep Runge-Kutta methods up to order 10.
In this paper we consider a special class of explicit GLM, explicit peer methods introduced in [15]. These methods have been successfully applied to nonstiff ODEs with step size control in

[*]Széchenyi István Egyetem, 9026. Győr, Egyetem tér 1, Hungary (horvathz@sze.hu).
[†]Institut für Mathematik, Universität Halle, D-06099 Halle, Germany (helmut.podhaisky@mathematik.uni-halle.de).
[‡]Institut für Mathematik, Universität Halle, D-06099 Halle, Germany (weiner@mathematik.uni-halle.de).

[16]. For these methods the stage order is equal to the order of consistency. We investigate the SSP properties of explicit peer methods and prove a theorem which allows to construct such methods of high order (and consequently of high stage order).
The outline of the paper is as follows:
In Section 2 we introduce explicit peer methods and give a short overview about important properties like consistency, zero-stability and convergence.
In Section 3 we discuss SSP property for explicit peer methods. We prove a theorem which allows to determine the SSP coefficient from the parameters of the method. Furthermore a simple relation to linear stability is shown.
The construction of explicit peer methods with large SSP coefficients is presented in Section 4. The numerical optimization problems are solved with Mathematica. We have found SSP methods up to order 13.
Section 5 gives results of numerical tests. We verify numerically the order of the constructed methods, show the advantage of high stage order and illustrate the theoretical SSP properties by computing the step sizes ensuring the TVD property for the Buckley-Leverett equation.
Section 6 contains our conclusions.

2 Explicit peer methods

Explicit peer methods for problem (1) as introduced in [15] read

$$Y_{m,i} = \sum_{j=1}^{s} b_{ij} Y_{m-1,j} + h_m \sum_{j=1}^{s} a_{ij} f(t_{m-1,j}, Y_{m-1,j}) + h_m \sum_{j=1}^{i-1} r_{ij} f(t_{m,j}, Y_{m,j}), \quad i = 1, \dots, s. \tag{3}$$

Here $b_{i,j}$, $a_{i,j}$, c_i and r_{ij}, $i, j = 1, \dots, s$ are the parameters of the method. At each step s approximations $Y_{m,i}$ are computed for the exact solution $y(t_{m,i})$ with $t_{m,i} = t_m + c_i h_m$. The nodes c_i are assumed to be pairwise distinct. Defining matrices $B = (b_{i,j})_{i,j=1,\dots s}$, $A = (a_{i,j})$, $R = (r_{i,j})$ and vectors $Y_m = (Y_{m,i})_{i=1}^{s} \in \mathbb{R}^{sn}$ and $F_m = (f(t_{m,i}, Y_{m,i}))_{i=1}^{s}$ lead to the compact form.

$$Y_m = (B \otimes I) Y_{m-1} + h(A \otimes I) F_{m-1} + h(R \otimes I) F_m,$$

where R is strictly lower triangular.
The coefficients of the method (3) depend, in general, on the step size ratio $\sigma = h_m / h_{m-1}$. Like multistep methods peer methods need also s starting values $Y_{0,i}$. We collect here some results from [15]:
Conditions for the order of consistency of explicit peer methods can be derived by considering the residuals $\Delta_{m,i}$ obtained when the exact solution is put into the method

$$\Delta_{m,i} := y(t_{m,i}) - \sum_{j=1}^{s} b_{ij} y(t_{m-1,j}) - h_m \sum_{j=1}^{s} a_{ij} y'(t_{m-1,j}) - h_m \sum_{j=1}^{i-1} r_{ij} y'(t_{m,j}), \quad i = 1, \dots, s.$$

Definition 1. *The peer method* (3) *is consistent of order* p *if*

$$\Delta_{m,i} = \mathcal{O}(h_m^{p+1}), \quad i = 1, \dots, s. \qquad \square$$

In contrast to explicit Runge-Kutta methods, all stage values of peer methods are approximations of order p to the solution $y(t + c_i h_m)$, i.e., the stage order is equal to the order. This makes these methods advantageous especially for MOL problems when space and time step sizes are reduced simultaneously. By Taylor series follows that a peer method (3) has order of consistency p iff

$$c_i^l - \sum_{j=1}^{s} b_{ij} \frac{(c_j - 1)^l}{\sigma^l} - l \sum_{j=1}^{s} a_{ij} \frac{(c_j - 1)^{l-1}}{\sigma^{l-1}} - l \sum_{j=1}^{i-1} r_{ij} c_j^{l-1} = 0, \quad i = 1, \dots, s, \quad l = 0, \dots, p \tag{4}$$

is satisfied, [15]. This condition (4) can be written conveniently as

$$\exp(c\sigma z) - B\exp(z(c-\mathbb{1})) - A\sigma z\exp(z(c-\mathbb{1})) - R\sigma z\exp(\sigma z) = \mathcal{O}(z^{p+1}),$$

where $\mathbb{1} = (1, \dots, 1)^\top$. The exponentials of the vectors are defined componentwise.
The condition (4) for order 0 is referred to as *preconsistency.* It takes the form

$$B\mathbb{1} = \mathbb{1}. \tag{5}$$

Explicit peer methods are a special class of *general linear methods, GLMs.* GLMs are typically investigated for constant step sizes only. An overview can be found in [2] and [8]. For the investigation of SSP in this paper we will restrict to the case of constant step sizes $h_m = h$ (cf. [10], [3]), too.
A convergence result for peer methods can be found in [16]. For the constant step sizes considered here, the zero stability criterion reduces to the power boundeness of B.

Theorem 1. *Let B, A, R denote the coefficients of a peer method* (3). *If the method is consistent of order p and $\|B^m\| \le K$ for a fixed K for all $m \in \mathbb{N}$ and the starting values satisfy $Y_{0,i} - y(t_{0,i}) = \mathcal{O}(h^p)$ then method is convergent of order p.*

However, the order of convergence may be greater than the order of consistency. In [16] explicit peer methods of order of consistency s and order of convergence $p = s+1$ were constructed and gave excellent results in numerical comparisons with dopri5 and dop853 [6].
In general, one step with method (3) requires s function calls. However, if the matrices A, B and R contain additional zeros some of those stage values don't actually require new function evaluations because previously computed stages are just re-used in the current step. If just one stage value is re-used this property is usually referred to as *first-same-as-last* (FSAL) property of Runge–Kutta methods or peer methods [12], respectively. Ketcheson et al. [10] give a similar condition ("Type II methods") for two–step Runge–Kutta method. Interestingly, the peer methods found numerically by optimizing the SSP property are often sparse in a certain way. This motivates the following definition.

Definition 2. *A peer method* (3) *is said to have n_s* shifted stages *and $s_e = s - n_s$* effective stages *if*

$$c_i = c_{i+1} - 1, \quad e_i^\top A = e_i^\top R = [0, \dots, 0], \quad e_i^\top B = e_{i+1}^\top \quad \textit{for} \quad i = 1, \dots, n_s. \tag{6}$$

From (6) follows $Y_{m,n_s} = Y_{m+1,n_s-1} = \cdots = Y_{m+n_s,1}$. The number of function evaluations per step for these methods is s_e. A method with $n_s = s-1$ has only one effective stage and is therefore equivalent to a linear multistep method.

3 Conditions for SSP

The SSP condition shall guarantee that a convex functional $\|\cdot\|$ of the numerical approximations is non-increasing. In the precise definition we state this property relative to the same property for the explicit Euler method.

Definition 3. *A peer method* (3) *is* strong stability preserving under the explicit Euler condition with SSP coefficient $C > 0$ *if $\forall h_E > 0, \forall f$ and $\forall \|\cdot\|$ convex functionals with*

$$(\forall v \in \mathbb{R}^n\colon \|v + h_E f(v)\| \le \|v\|)$$

$\forall h\colon h \le C \cdot h_E$ the condition

$$\max_i \|Y_{m,i}\| \le \max_i \|Y_{m-1,i}\|$$

holds true.

The peer methods (3) can be written as a generic numerical process, i.e. the theory developed by Spijker [14] can be used to derive the criteron stated below. For this we would need to transform the scheme. Since we prefer to work with the original matrices, we include a derivation of the result here. To find the SSP coefficient C from the parameters of a given method, we define $g\colon \mathbb{R}^+ \to \mathbb{R}^{s\times(3s)}$ as the matrix-valued function

$$g(r) = (I + rR)^{-1}(R, A, B - rA).$$

We will show in Lemma 1 and Theorem 2 that the SSP coefficient is $C = \max\{r\colon g(r) \geq 0\}$ (where $g(r) \geq 0$ means that all components of $g(r)$ are nonnegative). The function $g(r)$ is defined for all r for explicit methods, the components of $g(r)$ are polynomials in r because R is strictly lower triangular.

Lemma 1. *Let $g(C) \geq 0$ be satisfied for $C > 0$. Then $g(r) \geq 0$ for all $r \in [0, C]$.*

Proof. We decompose $g(r) = \widehat{g}(-r) = [\widehat{g}_1(-r), \widehat{g}_2(-r), \widehat{g}_3(-r)]$ using

$$\begin{aligned}
\widehat{g}_1(\xi) &= (I - \xi R)^{-1}R,\\
\widehat{g}_2(\xi) &= (I - \xi R)^{-1}A,\\
\widehat{g}_3(\xi) &= (I - \xi R)^{-1}(B + \xi A).
\end{aligned}$$

Differentiating with respect to ξ yields

$$\begin{aligned}
\widehat{g}_1'(\xi) &= -(I - \xi R)^{-1}(-R)(I - \xi R)^{-1}R = \widehat{g}_1(\xi)^2,\\
\widehat{g}_2'(\xi) &= \widehat{g}_1(\xi) \cdot \widehat{g}_2(\xi),\\
\widehat{g}_3'(\xi) &= \widehat{g}_1(\xi) \cdot \widehat{g}_3(\xi) + \widehat{g}_2(\xi).
\end{aligned}$$

The assumption $g(C) \geq 0$ implies $\widehat{g}(-C) \geq 0$ and by induction follows that all derivatives $\widehat{g}^{(l)}(-C)$, $l \in \mathbb{N}$ are nonnegative. If $\widehat{g}(\xi)$ is analytic, we can expand $g(\xi)$ around C in a Taylor series. Thus the proof is completed by noting that all terms

$$g(r) = \widehat{g}(-r) = \sum_{l=0}^{\infty} \widehat{g}^{(l)}(-C)\frac{(C-r)^l}{l!} \geq 0$$

are nonnegative for $r \leq C$. Since $g(r)$ is polynomial in r this series is actually a finite sum and therefore convergent. □

Theorem 2. *Let (B, A, R) be the coefficients of a peer method* (3) *and assume that*

$$C = \max_{r\in\mathbb{R}^+}\{r\colon (I + rR)^{-1}(R, A, B - rA) \geq 0\} \tag{7}$$

is positive. Then this method is strong stability preserving with SSP coefficient C.

Proof. From Lemma 1 follows $g(r) \geq 0$ for all $r \in [0, C]$. We have to show that this implies

$$\max_i \|Y_{m,i}\| \leq \max_i \|Y_{m-1,i}\| \tag{8}$$

for $h = r \cdot h_E$. From $\lim\limits_{r\to+0} g(r) \geq 0$ follows $B \geq 0$, $A \geq 0$, $R \geq 0$. We will re-write the peer method using convex combinations of explicit Euler steps. By adding and subtracting Y-values we get

$$\begin{aligned}
Y_m &= BY_{m-1} \pm rAY_{m-1} + hAF_{m-1} \pm rRY_m + hRF_m\\
Y_m &= (I + rR)^{-1}\big((B - rA)Y_{m-1} + rA(Y_{m-1} + \frac{h}{r}F_{m-1}) + rR(Y_m + \frac{h}{r}F_m)\big)
\end{aligned}$$

Using the assumption (2) for $h = r \cdot h_E$ for the Euler methods

$$\|Y_{m-1,j} + \frac{h}{r} F_{m-1,j}\| \leq \|Y_{m-1,j}\|, \quad \|Y_{m,j} + \frac{h}{r} F_{m,j}\| \leq \|Y_{m,j}\|. \tag{9}$$

yields with $g(r) \geq 0$

$$\begin{aligned} |Y_m| &\leq (I + rR)^{-1} \big((B - rA)|Y_{m-1}| + rA|Y_{m-1}| + rR|Y_m| \big) \\ (I + rR)^{-1} |Y_m| &\leq (I + rR)^{-1} B |Y_{m-1}| \\ |Y_m| &\leq B|Y_{m-1}|, \end{aligned}$$

where

$$|Y_m| = \begin{pmatrix} \|Y_{m,1}\| \\ \vdots \\ \|Y_{m,s}\| \end{pmatrix}.$$

From this, we finally obtain (8) since $B \geq 0$ and $B\mathbb{1} = \mathbb{1}$. $\square$

The SSP property is closely related to linear stability. Applying the peer methods (3) to the linear test equation $y' = \lambda y$ leads to the recursion $Y_m = M(z)Y_{m-1}$ with the stability matrix [15]

$$M(z) = (I - zR)^{-1}(B + zA). \tag{10}$$

Note that $M(z) = \widehat{g}_3(z)$ and hence, by the proof of Lemma 1, follows that the derivatives of $M(z)$ are nonnegative in $[-C, 0]$.

Corollary 1. *Let $M(z)$ be the stability matrix of a peer method which has SSP coefficient C. For all $l \in \mathbb{N}$ and $\xi \in [-C, 0]$ holds $\frac{d^l}{d\xi^l} M(\xi) \geq 0$.*

Lemma 2. *Let $M(z)$ be the stability matrix of a peer method which has SSP coefficient $C > 0$. Then*

$$\{z \colon z \in \mathbb{C},\ |1 + z/C| \leq 1\} \subseteq \{z \colon z \in \mathbb{C},\ M(z) \text{ is power-bounded}\}$$

holds, i.e., the stability domain of the explicit Euler method scaled by C is contained within the stability domain of the peer method.

Proof. The explicit Euler discretization for $y' = \lambda y$ satisfies $|y_{m+1}| = |y_m + h_E \lambda y_m| \leq |y_m|$ when $h_E \lambda \in \{z \colon |1 - z| \leq 1\}$. This implies for $h \leq C \cdot h_E$

$$\max_i \|Y_{m+1,i}\| \leq \max_i \|Y_{m,i}\| \leq \cdots \leq \max_i \|Y_{0,i}\|.$$

With $Y_m = M(z)^m Y_0$ the statement follows. $\square$

To compare different methods of the same order we need to analyse the leading term of the local truncation error. Since we assume high stage order there is no difference between the linear and the general case, i.e. the error constant can be found by analysing the dominant eigenvalue of $M(z)$.

Definition 4. *Let $M(z)$ be the stability matrix of a peer method (3) of order p. We say that η_{p+1}* is the leading error constant *if the dominant eigenvalue $\lambda(z)$ of $M(z)$ at $z = 0$ satisfies*

$$\lambda(z) = e^z - \eta_{p+1} z^{p+1} + \mathcal{O}(z^{p+2}). \tag{11}$$

By Taylor series expansion of the characteristic equation one obtains from (11), after some calculation, the explicit formula

$$\eta_{p+1} = e_s^T \left(I - B + \mathbb{1} e_s^T\right)^{-1} \left(\frac{c^{p+1}}{(p+1)!} - B\frac{(c-1)^{p+1}}{(p+1)!} - A\frac{(c-1)^p}{p!} - R\frac{c^p}{p!} \right). \tag{12}$$

For a fair comparison of methods of the same order we define the *effective error constant* by

$$\eta_{\text{eff}} = s_e \cdot \eta_{p+1}^{1/p}, \tag{13}$$

where s_e is the effective number of function evaluations per step. The coefficient η_{eff} is proportional to the amount of work needed to reach a certain accuracy. This holds since the global error is, for sufficiently small step sizes, proportional to $\eta_{p+1} \cdot h^p$. We have verified in numerical experiments that η_{eff} is an accurate measure for the efficiency for peer methods (because the structure of local error is the same for different methods).

Example 1. The fourth order peer method with $c = \left(-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, 1\right)^\top$ and the coefficient matrices

$$B = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{4}{25} & \frac{5}{9} & 0 & \frac{64}{225} \\ \frac{1}{5} & \frac{1}{4} & \frac{1}{8} & \frac{17}{40} \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{16}{15} \\ \frac{97}{15360} & \frac{4717}{15360} & \frac{23}{3072} & \frac{3}{10} \end{pmatrix}, \quad R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{3}{10} & 0 & \frac{1041}{1024} & 0 \end{pmatrix} \tag{14}$$

has $s = 4$, $n_s = 2$. The SSP coefficient can be calculated from (7), $C = 4(75 - \sqrt{2849})/347 \approx 0.24927$ and $\mathcal{C}_{\text{eff}} = 0.124634$. We find $\eta_5 = 17783/1002960$. The stability domain is drawn in Figure 1. Note that we have included this method (14) to provide a simple example showing the structure. It is not recommend to use this method in real computations; the optimized method with $s = p = 4$ and $n_s = 2$ has a much larger value $\mathcal{C}_{\text{eff}} = 0.2273$, cf. Table 3.

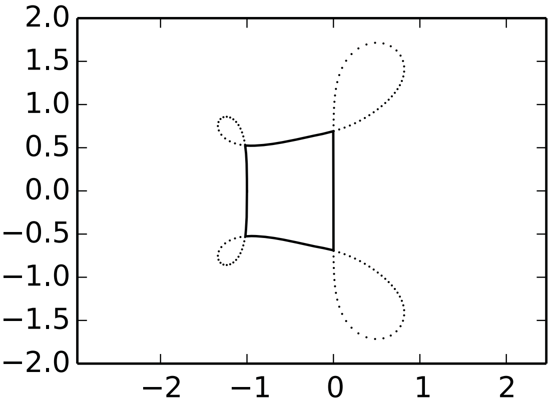


Figure 1: Domain of linear stability for the method (14). The dotted lines mark the unstable part of the root locus curve.

4 Numerical optimization

Peer methods with the largest possible SSP coefficient can be found by solving the following optimization problem: Find the maximal value $r > 0$ and the coefficients A, B, c, and R subject to the constraints of Eq. (4) for order p, Eq. (6) for the structure of the method and $g(r) \geq 0$. The optimal value of r may depend on bounds for the nodes c_i.

Example 2 (Two weakly coupled Euler methods). Consider the two stage, second order, parallel method ($R = 0$) depending on the parameter $\eta = -c_1$ for the form

$$c = \begin{pmatrix} -\xi \\ 1 \end{pmatrix}, \quad B = \begin{pmatrix} \frac{\xi^2+2\xi}{(\xi+1)^2} & \frac{1}{(\xi+1)^2} \\ \frac{1}{(\xi+1)^2} & \frac{\xi^2+2\xi}{(\xi+1)^2} \end{pmatrix}, \quad A = \begin{pmatrix} \frac{\xi}{\xi+1} & 0 \\ 0 & \frac{\xi+2}{\xi+1} \end{pmatrix}.$$

This method preserves strong stability for $\xi > 0$ with $C = \frac{\xi}{\xi+1}$. For $\xi \to \infty$ we have $C \to 1$. Note that the off-diagonal elements of B are small, thus the method resembles two explicit Euler methods. The weak coupling raises the order to two but does not reduce the SSP coefficient severely. Since $c_1 \to -\infty$ will not be useful for practical computations we need to restrict the interval for the nodes in our optimization.

We included the constraints $-s \leq c_i \leq 1$ and $c_s = 1$, cf. Example 2. For the results shown in Tables 1–11 we used Mathematica's `FindMaximum[]` with the option `WorkingPrecision -> 50`, which implies `AccuracyGoal -> 25` and `PrecisionGoal-> 25`, i.e., the absolute and relative tolerances for the numerical optimization were 10^{-25}, see Section A for algorithmic details and source code.

Methods of order 2 to 12. In the following tables we give the numerically obtained values of the SSP coefficient $\mathcal{C}_{\text{eff}} = C/s_e$ for methods with n_s shifts and s_e effective stages, i.e. with $s = n_s + s_e$ stages and with s_e functions evaluations per step.

Table 1: $\mathcal{C}_{\text{eff}}$ for methods of order 2.

$n_s \backslash s_e$	1	2	3	4	5	6
0		0.3535	0.6563	0.7681	0.8255	0.8603
1		0.7120	0.8310	0.8806	0.9077	0.9248
2	1/2	0.8172	0.8881	0.9194	0.9370	0.9562
3	2/3	0.8662	0.9163	0.9391	0.9521	0.9606
4	3/4	0.8945	0.9331	0.9510	0.9614	0.9681

Table 2: $\mathcal{C}_{\text{eff}}$ for methods of order 3.

$n_s \backslash s_e$	1	2	3	4	5	6
0			0.3345	0.5300	0.5749	0.5828
1		0.4304	0.5531	0.5443	0.5749	0.5828
2		0.5738	0.5537	0.5443	0.5749	0.5828
3	1/3	0.5738	0.5537	0.5443	0.5749	0.5828
4	1/2	0.5738	0.5537	0.5443	0.5749	0.5828

Table 3: $\mathcal{C}_{\text{eff}}$ for methods of order 4.

$n_s \backslash s_e$	1	2	3	4	5	6
1			0.1748	0.3877	0.4579	0.4880
2		0.2273	0.2536	0.4456	0.5015	0.5227
3		0.3296	0.4282	0.4710	0.5214	0.5388
4	0.0211	0.4075	0.4666	0.4880	0.5328	0.5480

Table 4: $\mathcal{C}_{\text{eff}}$ for methods of order 5.

$n_s \backslash s_e$	2	3	4	5	6
1			0.2440	0.3438	0.3921
2		0.1317	0.3535	0.3988	0.3988
3	0.1682	0.3015	0.3802	0.3981	
4	0.2631	0.3748	0.3989	0.3988	

Table 5: $\mathcal{C}_{\text{eff}}$ for methods of order 6.

$n_s \backslash s_e$	3	4	5	6
2		0.1547	0.2725	0.3038
3		0.2624	0.3118	0.3394
4	0.2115	0.3033	0.3404	0.3546

Methods of order $p \geq 13$. We have not yet systematically searched for methods of order $p > 12$ except for a few isolated example methods. In the appendix in Section A.2 we compute two peer methods of order $p = 13$ with $\mathcal{C}_{\text{eff}} = 0.097030$ ($s = 14$, $s_e = 5$) and $\mathcal{C}_{\text{eff}} = 0.112333$ ($s = 15$, $s_e = 5$). For linear multistep methods, where the optimization problem is less demanding, we were able to reach $p = 18$ with $s = 76$ steps with $\mathcal{C}_{\text{eff}} = 0.04456740$. Optimal lineare multistep methods up to order $p = 15$ and $s = 50$ can be found in Table 8.2 in [5] (where the last entry is for $p = 15$ and $s = 50$ with $\mathcal{C}_{\text{eff}} = 0.034$).

Accuracy vs. stability. Fig. 2 compares the effective error constant η_{eff} defined in (13) with the SSP coefficient $\mathcal{C}_{\text{eff}}$ for methods of order $p = 8$ which have been found by optimizing $\mathcal{C}_{\text{eff}}$. The positive slope indicates that better stability leads to less accuracy, i.e. one should not look for the largest value of $\mathcal{C}_{\text{eff}}$ but for a resonable compromise. This figure (and similar results for other orders) seems to indicate that lineare multistep methods ($s_e = 1$) are most promising. A drawback, however, is the very high number of steps because this makes the starting procedure more costly and total memory consumption higher.

Table 6: $\mathcal{C}_{\text{eff}}$ for methods of order 7.

$n_s \backslash s_e$	4	5	6
2		0.1726	0.1995
3		0.2399	0.2841
4	0.2338	0.2865	0.2991

Table 7: $\mathcal{C}_{\text{eff}}$ for methods of order 8.

$n_s \backslash s_e$	4	5	6
3		0.1010	0.1358
4	0.0824	0.1988	0.2327
5	0.1800	0.2353	0.2540

Table 8: $\mathcal{C}_{\text{eff}}$ for methods of order 9.

$n_s \backslash s_e$	4	5	6
3			0.0725
4		0.1326	0.1531
5	0.1049	0.1729	0.2156

Table 9: $\mathcal{C}_{\text{eff}}$ for methods of order 10.

$n_s \backslash s_e$	4	5	6
5		0.0781	0.1023
6		0.1399	0.1790
7	0.1105	0.1765	0.1975

Table 10: $\mathcal{C}_{\text{eff}}$ for methods of order 11.

$n_s \backslash s_e$	4	5	6
6		0.1026	0.1234
7		0.1269	0.1578
8	0.0941	0.1539	0.1792

Table 11: $\mathcal{C}_{\text{eff}}$ for methods of order 12.

$n_s \backslash s_e$	4	5
7		0.0588
8		0.0953
9	0.0592	0.0953

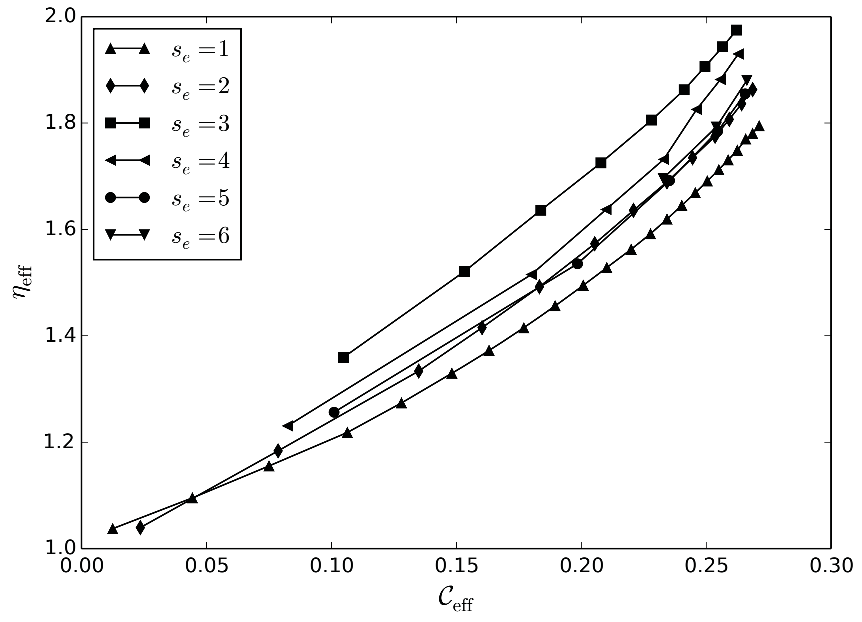


Figure 2: Strong stability vs. accuracy for peer methods of order 8. Methods with $s_e = 1$ are linear multistep methods. They appear to be slightly more accurate than methods with $s_e > 1$ with comparable SSP coefficient $\mathcal{C}_{\text{eff}}$.

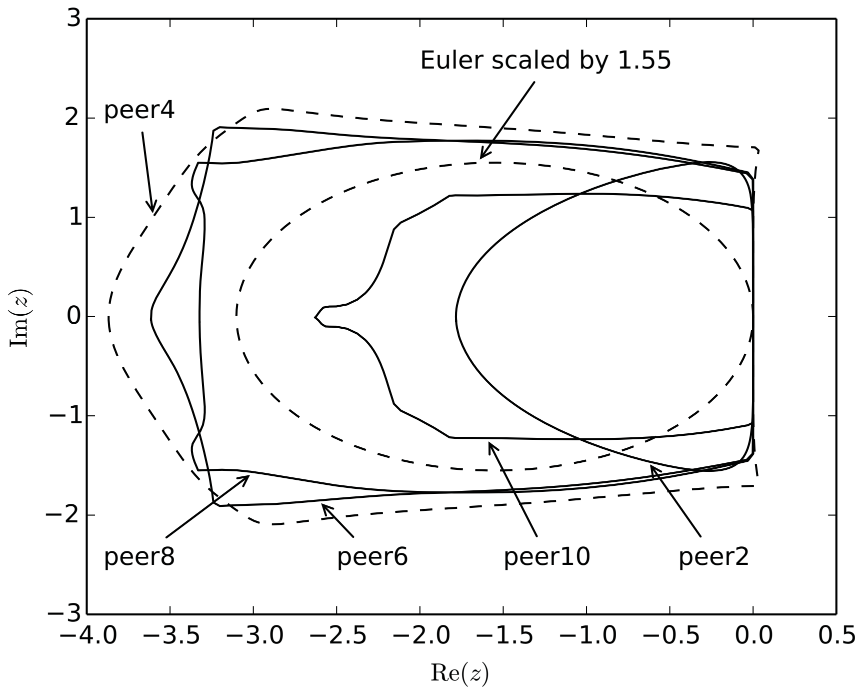


Figure 3: Stability regions of the peer methods from Table 12. The region for explicit Euler scaled by $4 \cdot \mathcal{C}_{\text{eff}} \approx 1.55$ is within the region of peer4 by Lemma 2.

5 Numerical illustration

We are going to report numerical experiments with the explicit peer methods shown in Table 16. The labels indicate s_e, n_s and the order p. For all methods the stage order is equal to the order of consistency p. Figure 3 shows the stability region of the methods.

Table 12: Peer methods used in the experiments.

method	order	stages	s_e	n_s	$\mathcal{C}_{\text{eff}}$	η_{p+1}
peer2	2	2	2	0	0.3535	0.063113
peer4	4	5	4	1	0.3877	0.003501
peer6	6	7	5	2	0.2725	0.000246
peer8	8	10	6	4	0.2327	0.000041
peer10	10	11	5	6	0.1399	0.000028

For comparison we include the explicit Euler method (euler) with $\mathcal{C}_{\text{eff}} = 1$, the Shu-Osher Runge-Kutta method (ssp3) [13] with $\mathcal{C}_{\text{eff}} = 1/3$ and with Butcher tableau

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ 1/2 & 1/4 & 1/4 & \\ \hline & 1/6 & 1/6 & 2/3 \end{array}$$

and the popular method of Dormand/Prince (dopri5) [4] of order 5, which, however, has no positive SSP coefficient.

We have implemented the methods in Python 3. In all tests we used constant step sizes. The starting values for the peer methods were computed with dop853 with rtol = atol = 10^{-12}.

Test 1: Order test.

We consider the nonstiff two-dimensional Brusselator [6] given by

$$\begin{aligned} u_t &= 1 + u^2 v - 4.4u + \alpha\,(u_{xx} + u_{yy}) \\ v_t &= 3.4u - u^2 v + \alpha\,(v_{xx} + v_{yy}) \end{aligned} \tag{15}$$

on $(x, y) \in \Omega = [0, 1]^2$, $t \in [0, 7.5]$, $\alpha = 0.002$, with Neumann boundary conditions

$$\frac{\partial u}{\partial n} = 0\,, \qquad \frac{\partial v}{\partial n} = 0 \text{ on } \partial\Omega$$

and initial conditions $u(0, x, y) = 0.5 + y$, $v(0, x, y) = 1 + 5x$. Discretization with central differences with 21 space points yields a system of ordinary differential equations of dimension $n = 882$. Figure 4 shows the accuracy obtained at $t = 7.5$. The reference solution has been computed with dop853 with rtol = atol = 10^{-12}. All methods show the expected orders.

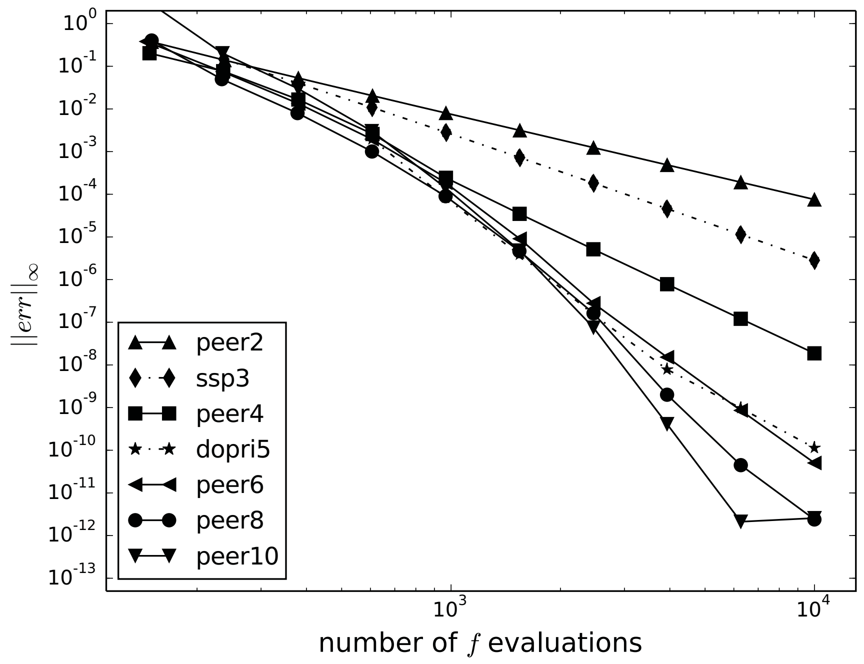


Figure 4: Work vs. precision for the Brusselator (15)

Test 2: Test of order reduction.

Here we show the advantage of the high stage order of the peer methods. We consider the following example (cf. [3]):

$$\begin{aligned} u_t &= -u_x + b(t, x), \quad 0 \le x \le 1, \quad 0 \le t \le 1 \\ b(t, x) &= (t - x)/(1 + t)^2 \end{aligned} \tag{16}$$

Initial values and left boundary conditions are taken from the exact solution $u(t, x) = (1+x)/(1+t)$. The spatial derivative is discretised with a first order scheme which doesn't introduce spatial discretization errors since the exact solution is linear in x. We decrease h and the space step size Δx simultaneously with $\Delta x = 2h$.

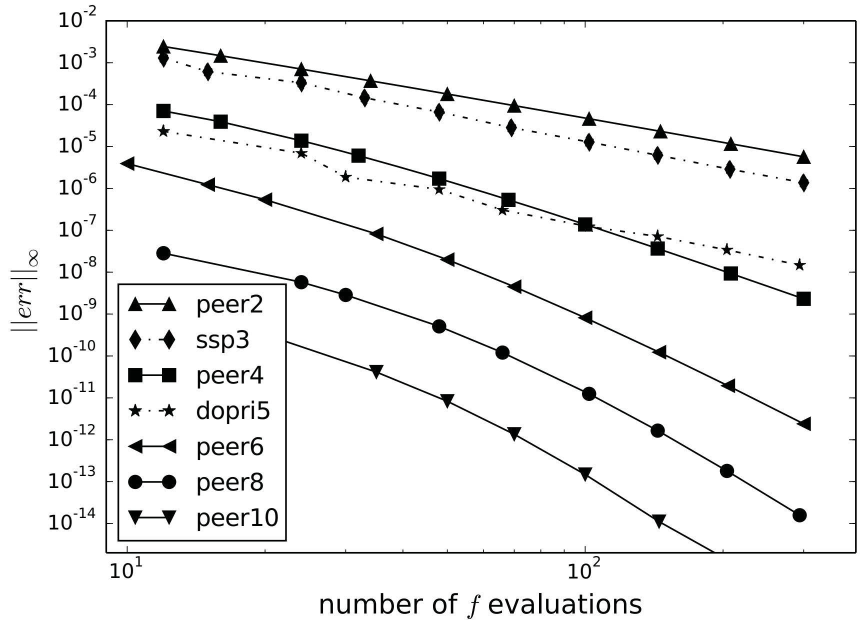


Figure 5: Work vs. precision for the convection equation (16).

The results are given in Figure 5. The Runge–Kutta methods ssp3 and dopri5 suffer from order reduction to order 2, whereas there is no order reduction for the peer methods.
Test 3: TVD time step.
We consider the Buckley–Leverett equation [7]

$$u_t + f(u)_x = 0, \quad 0 \le t \le 1/4, \quad 0 \le x \le 1, \quad f(u) = \frac{cu^2}{cu^2 + (1-u)^2} \tag{17}$$

with periodic boundary condition and $c = \frac{1}{3}$. The initial condition is

$$u(0,x) = \begin{cases} 0 & \text{for } 0 \le x < 1/2 \\ 1 & \text{otherwise .} \end{cases}$$

Discretization in space on a uniform grid is done with $n = 100$ grid points $x_j = j\,\Delta x$, $j = 0\ldots, n-1$, $\Delta x = 1/100$ with Koren limiter, see [7]. Since the problem is not sufficiently smooth with respect to time the high order methods do not perform better than ssp3 as seen in Fig. 6. It it clearly seen that dopri5 doesn't work well for larger step sizes compared with the other methods.
The Euler discretization for (17) is known to be total variation diminishing (TVD) with $h_E = \Delta x/4 = 0.0025$ [10]. We solved this problem with increasing numbers of steps and computed the total variation for the mth integration interval. For the peer methods this reads

$$TV(m) = \max_{i=1,\ldots,s} \sum_{j=1}^{n} |Y_{m,i,j} - Y_{m,i,j-1}|, \quad Y_{m,i,0} = Y_{m,i,n}.$$

Fig. 7 shows the maximal increase of the total variation in the numerical solution. The vertical lines indicate the critical step sizes for the preservation of the total variation. The scaling is chosen to see the numerically obtained values for $\mathcal{C}_{\text{eff}}$. It can be seen that dopri5 produces solutions with increased variation except for rather small step sizes.

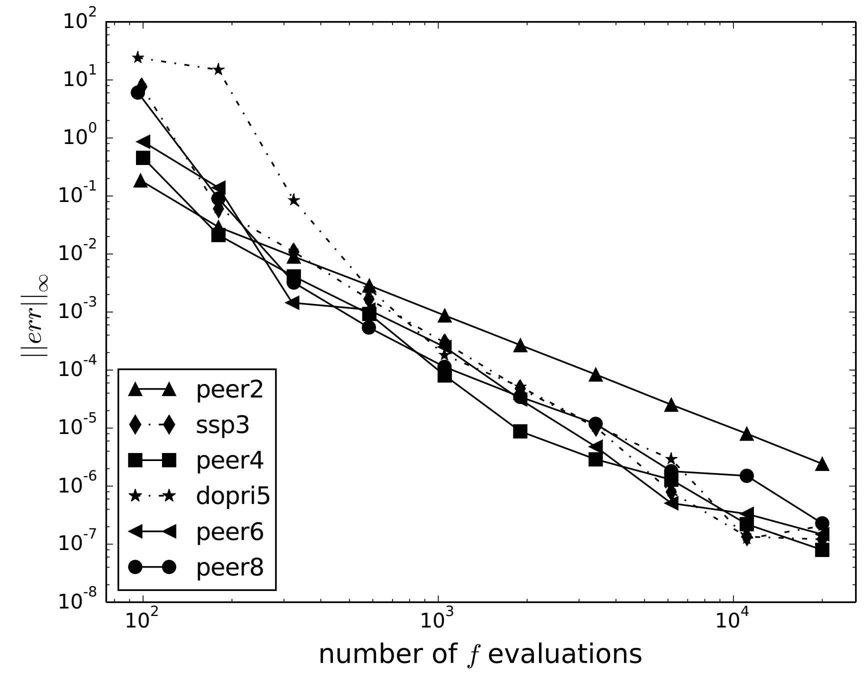


Figure 6: Work vs. precision for Buckley–Leverett equation (17).

6 Conclusions and future work

We have constructed explicit peer methods up to order 13 with large SSP coefficients $\mathcal{C}_{\text{eff}}$. The matrices (B, A, R) found by the numerical optimization process are often sparse.

The numerical tests agree with the theoretical results and show that these methods are a promising candidates for solving semidiscretized hyperbolic equations. Especially the high stage order seems to be advantageous.

Explicit peer methods have been implemented with step size control for general ODEs and successfully applied and compared with efficient RK methods. The investigation of explicit peer methods with variable step size in the SSP context and their implementation will be the topic of future work. Furthermore, we plan to consider implicit peer methods.

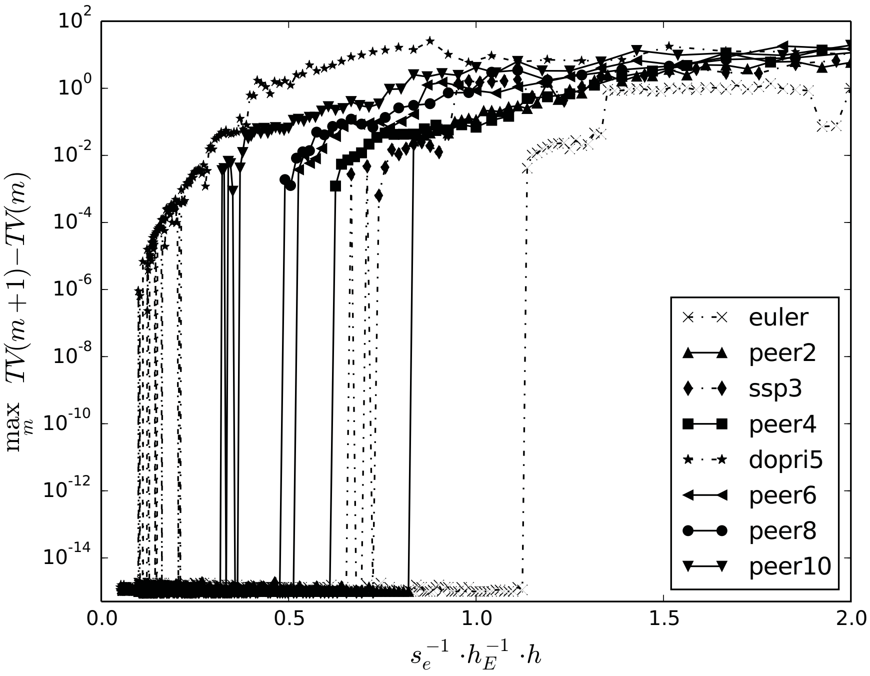


Figure 7: Maximal increase of the total variation for the Buckley–Leverett equation (17). The scaling of the abscissa is chosen to allow to simply read off the numerically obtained value corresponding to $\mathcal{C}_{\text{eff}}$, e.g. Euler's method is, for this example, total variation diminishing with $h \leq 1.1 \cdot h_E$ rather than $h \leq h_E$.

References

[1] C. Bresten, S. Gottlieb, Z. Grant, D. Higgs, D. Ketcheson, and A. Nemeth, *Explicit strong stability preserving multistep Runge-Kutta methods*, arXiv:1307.8058

[2] Butcher, J. C.: *General linear methods*, Acta Numer. 15, 157–256 (2006).

[3] E. Constantinescu and A. Sandu, *Optimal explicit strong-stability-preserving general linear methods*, SIAM Journal on Scientific Computing 32, 3130–3150 (2009).

[4] J. R. Dormand and P. J. Prince, *A family of embedded Runge-Kutta formulae*, J. Comput. Appl. Math., 6:19–26 (1980).

[5] S. Gottlieb, D. Ketcheson, and C. W. Shu, *Strong stability preserving Runge-Kutta and multistep time discretizations*, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2011.

[6] E. Hairer, S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations, I. Nonstiff Problems*, second revised edition, Springer, Berlin, 1993.

[7] W. Hundsdorfer and J. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer, 2003.

[8] Z. Jackiewicz, *General Linear Methods for Ordinary Differential Equations*, John Wiley & Sons Ltd., Chichester, 2009.

[9] D. Ketcheson, *Computation of optimal monotonicity preserving general linear methods*, Math. Comp. 78, Number 267, 1497–1513 (2009)

[10] D. I. Ketcheson, S. Gottlieb, and C. B. Macdonald, *Strong stability preserving two-step Runge-Kutta methods*, SIAM Journal on Numerical Analysis 49, 2618–2639 (2011).

[11] J. F. B. M. Kraaijevanger, *Contractivity of Runge-Kutta methods*, BIT 31, 482–528 (1991).

[12] B. A. Schmitt, R. Weiner, and St. Beck, *Two-step peer methods with continuous output*, BIT Numer. Math. 53, 717–739 (2013).

[13] C. W. Shu and S. Osher, *Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes*, Journal of Computational Physics 77, 439–471 (1988).

[14] M. N. Spijker, *Stepsize conditions for general monotonicity in numerical inital value problems*, SIAM J. Numer. Anal. 45, 1226–1245, 2007

[15] R. Weiner, K. Biermann, B.A. Schmitt, and H. Podhaisky, *Explicit two-step peer methods*, Computers & Mathematics with Applications 55, 609–619 (2008)

[16] R. Weiner, B. A. Schmitt, H. Podhaisky, and St. Jebens, *Superconvergent explicit two-step peer methods*, Journal of Computational and Applied Mathematics 223, 753–764 (2009).

A PEERLESS – a script to find explicit peer methods with large SSP coefficients

We describe briefly the functions which we have implemented using Mathematica (version 9 and 10) to optimize and analyse SSP peer methods. All files, including text files containing the coefficients, can be downloaded from http://sim.mathematik.uni-halle.de/helmut/2014/ssp. Note that this code has evolved to satisfy the needs of the authors. It should be regarded as a prototype which could be improved in many ways. We include it here mainly to make your computations reproducible.

A.1 List of functions

We use the symbol `mf` to denote a specific peer method represented as a list of rules, e.g. $\texttt{mf} = \{s \to 4, p \to 2, c \to \{\dots\}, B \to \{\{\dots\}\}, \dots\}$. The individual functions add components to this data structure or modify values. The main computation work is in the function `opx` which replaces the coefficients c, B, A and R of the method given as an input by optimized values.

- `euler[s, ns, p]` creates the peer coefficients containing explicit Euler over several steps. The order is of course 1, but the optimizer will later use the value of p specified here for the constraints.
- `bisec[mf]` computes the SSP coefficient by bisection.
- `incp[mf]` expressed the wish to increase the order by 1.
- `incns[mf]` increases n_s by 1.
- `incs[mf]` adds a trivial stage at $(c_{s-1}+1)/2$ to the scheme.
- `inc[mf]` re-optimization with order and number of steps increased by 1.
- `checkorder[mf]` evaluates the residuals in the order conditions.
- `refine[acc, ratQ][mf]` puts small components to zero and projects onto the order condition within the specified accuracy.
- `opx[fixcQ, wp, maxiter][mf]` changes (B, A, R) and if $\neg$`fixedQ` also c to optimize the SSP coefficient $\mathcal{C}$.
- `op[mf]` optimizes with MachinePrecision.
- `op50[mf]` optimizes with 50 decimal digits WorkingPrecision.
- `disp[mf]` draws a `MatrixPlot` of the coefficients in (B, A, R).
- `stabplot[mf]` draws the stability region.

A.2 Examples how to use the code

Example 1: Order 4 with nice nodes. We start the optimization with fixed nodes $c = (-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}, 1)$ and increase s and n_s successively. The resulting methods have nice rational coefficients. In a second run, we allow that also the nodes can be changed. This leads to slightly larger values of $\mathcal{C}_{\text{eff}}$, but the coefficients cannot be written in compact rational form. However, we can convert the floating point numbers to fractions and project then onto the order conditions. This procedure does not change the value obtained for $\mathcal{C}_{\text{eff}}$. Finally, we draw the sparsity pattern of the matrices (B, A, R) and the stability region. The total running time of the script is less approximately 10 seconds on a PC (2.2 GHz).

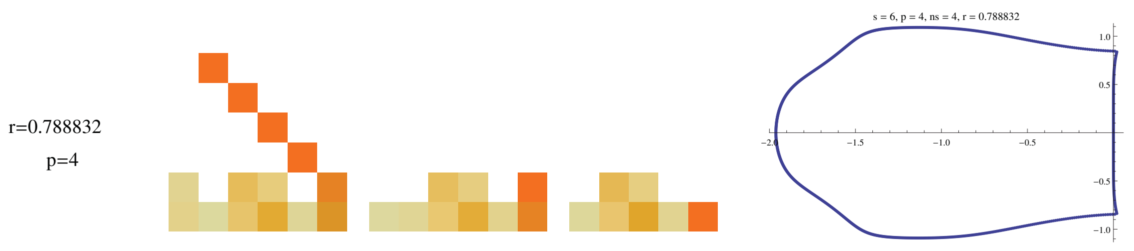
```
<< ../peerless
nicenodes = opx[True]@incns@opx[True]@incns@opx[True]@euler[4, 2, 4];
Print[c/.nicenodes]
m = op@incns@op@incns@op@euler[4, 2, 4];
Print[c/.m]
m1 = refine[7, True][m];
Print["Check with rationals, Ceff=",r/(s-ns) /. disp@bisec[m1]];
opx[m1]

Print["Are we attracted to the same optimum?"]
Print["(i) start with se=2, add a dummy stage"]
Print["optimize then gives Ceff=" , r/(s-ns) /. op@incs@op@euler[4, 2, 4]]
Print[ "(ii) start right with se = 3 gives Ceff=", r/(s-ns) /. op@euler[5, 2, 4]]
Print["Yes!"]
```

Output

```
Mathematica 10.0 for Linux x86 (64-bit)
Copyright 1988-2014 Wolfram Research, Inc.

In[1]:= PEERLESS (03-10-2014)
op: Ceff = 0.1333333374728111, s = 4, p = 4, ns = 2
op: Ceff = 0.2829330624588440, s = 5, p = 4, ns = 3
op: Ceff = 0.3944016353947241, s = 6, p = 4, ns = 4
   7     5     3     1   1
{-(-), -(-), -(-), -(-), -, 1}
   2     2     2     2   2
op: Ceff = 0.2273599676877895, s = 4, p = 4, ns = 2
op: Ceff = 0.3296922358675425, s = 5, p = 4, ns = 3
op: Ceff = 0.4075484396587121, s = 6, p = 4, ns = 4
{-3.5127715830482957, -2.5127715830482957, -1.5127715830482957,
 -0.5127715830482957, 0.4872284169517043, 1}
-Graphics-
Check with rationals, Ceff=0.407548
Are we attracted to the same optimum?
(i) start with se=2, add a dummy stage
op: Ceff = 0.2273599676877895, s = 4, p = 4, ns = 2
op: Ceff = 0.3321686485627544, s = 5, p = 4, ns = 2
optimize then gives Ceff=0.3321686485627544
op: Ceff = 0.3321686660623004, s = 5, p = 4, ns = 2
(ii) start right with se = 3 gives Ceff=0.3321686660623004
Yes!
```



Example 2: Generating the methods uses in the computations. We compute the methods shown in Table 12 with 50 digits of working precision, generate coefficient files in

Octave/Matlab format and compute the leading error constants η_{p+1}. The running time is approximately 7 minutes due to the high cost of the long arithmetic.

```
<< ../peerless
m202 = op50@euler[2, 0, 2];
m514 = op50@euler[5, 1, 4];
m726 = op50@euler[7, 2, 6];
m1068 = op50@op@euler[10, 4, 8];
m11610 = op50@op@euler[11, 6, 10];
lom = {m202, m514, m726, m1068, m11610};
Map[toml,lom]
Print[Map[eta,lom]]
Save["methods-used.txt", lom]
```

```
———————————————————————— Output ————————————————————————
Mathematica 10.0 for Linux x86 (64-bit)
Copyright 1988-2014 Wolfram Research, Inc.

In[1]:= PEERLESS (03-10-2014)
op: Ceff = 0.3535533905932737622200369155, s = 2, p = 2, ns = 0
op: Ceff = 0.3877131846025990072641154 96, s = 5, p = 4, ns = 1
op: Ceff = 0.2725344659885362389998240 15, s = 7, p = 6, ns = 2
op: Ceff = 0.2327747521601473, s = 10, p = 8, ns = 4
op: Ceff = 0.2327747521628969155360363 32, s = 10, p = 8, ns = 4
op: Ceff = 0.1399935225384896, s = 11, p = 10, ns = 6
op: Ceff = 0.1399935237955421311416467 0, s = 11, p = 10, ns = 6
{0.06311327607339290446624070808875537 3,
 0.003501374575027615854174121363268420045116818605154,
 0.000245881313566138232892727785596740748171858687643,
 0.000040750998903774448179936071226416696878182933035,
 0.000028892563949272032586564536918926883607002457 96}
```

Example 3: Order 13 with five effective stages. We start with order $p = 5$ and increase s, p and n_s simultaneously to reach methods of order $p = 13$ with five effective stages. The running time is approximately 4 minutes.

```
<< ../peerless
m6 = op@incns@op@euler[5, 0, 5];
m13 = Nest[inc, m6, 8];
m13refined = op50@m13;
Print["verify with bisection: ", r/(s-ns) /. bisec@m13refined]
op@incns[m13];
```

```
———————————————————————— Output ————————————————————————
Mathematica 10.0 for Linux x86 (64-bit)
Copyright 1988-2014 Wolfram Research, Inc.

In[1]:= PEERLESS (03-10-2014)
op: Ceff = 0.02874598585415610, s = 5, p = 5, ns = 0
op: Ceff = 0.3438977017441961, s = 6, p = 5, ns = 1
op: Ceff = 0.2725343649752956, s = 7, p = 6, ns = 2
op: Ceff = 0.2399135569544406, s = 8, p = 7, ns = 3
op: Ceff = 0.1985077222070308, s = 9, p = 8, ns = 4
```

```
op: Ceff = 0.1729225824044389, s = 10, p = 9, ns = 5
op: Ceff = 0.1399934289453578, s = 11, p = 10, ns = 6
op: Ceff = 0.1269368335474908, s = 12, p = 11, ns = 7
op: Ceff = 0.0953206662730194, s = 13, p = 12, ns = 8
op: Ceff = 0.0970302631116597, s = 14, p = 13, ns = 9
op: Ceff = 0.09703046704508972283044762174721962, s = 14, p = 13, ns = 9
verify with bisection: 0.0970305
op: Ceff = 0.1123332777157739, s = 15, p = 13, ns = 10
```

B Source code

peerless

```
(* Reference: Horvath, Podhaisky, Weiner: Strong stability
preserving explicit peer methods. 2014.  License: CC BY 3.0 *)

Print["PEERLESS (03-10-2014)"]

Off[NMaximize::nsol, Maximize::infeas, $MinPrecision::preccon, FindMaximum::precw]

euler[s_, ns_, p_] := Module[{se}, se = s - ns;
  { c -> Table[1/se + k, {k, -ns, -1}] ~Join~ Table[k/se, {k, 1, se}],
    A -> Table[If[i >= ns + 1 && j == s, 1/se, 0],  {i, 1, s}, {j, 1, s}],
    B -> Table[If[(i <= ns && j == i + 1) || (i > ns && j == s), 1, 0],  {i, 1, s}, {j, 1, s}],
    R -> Table[ If[i > ns + 1 && ns < j < i, 1/se, 0], {i, 1, s}, {j, 1, s}],
    Symbol["p"] -> p, Symbol["s"] -> s,  Symbol["ns"] -> ns }
]

bisec[mf_] :=  Module[{s, A, B, R, M, a, b, m},
  {s, A, B, R} = (Symbol /@ {"s", "A", "B", "R"}) /. mf;
  M[r_] :=    Min[Inverse[IdentityMatrix[s] + r*R].Join[A, R, B - r A, 2]];
  a = 0; b = s;
  If[Min[{A, B, R}] >= 0, While[b - a > 1*^-10, m = (b + a)/2; If[M[m] < 0, b = m, a = m]]];
  Append[DeleteCases[mf, r -> x_], r -> N[a]]
]

incp[mf_] := mf /. (p -> x_) :> (p -> x+1)

incns[mf_] := Module[{A,R,B,c,ns,p,s},
  {B,A,R,c,ns,p,s} = Symbol/@{"B","A","R","c","ns","p","s"}/.mf;
 { Symbol["A"] -> Prepend[Map[Prepend[#, 0] &, A], Table[0, {s + 1}]],
   Symbol["R"] -> Prepend[Map[Prepend[#, 0] &, R], Table[0, {s + 1}]],
   Symbol["B"] -> Prepend[Map[Prepend[#, 0] &, B], Table[If[j == 2, 1, 0], {j, 1, s + 1}]],
   Symbol["c"] -> Prepend[c, c[[1]] - 1],
   Symbol["s"] -> s+1, Symbol["ns"] -> ns + 1, Symbol["p"] -> p}
]

incs[mf_] := Module[ {A,R,B,c,cs1,ns,p,s,rep},
    {B,A,R,c,ns,p,s} = Symbol/@{"B","A","R","c","ns","p","s"}/.mf;
    cs1 = c[[-2]];
    rep[X_, row_] := Transpose[ Insert[Transpose[Insert[X, row, -2]],
                       Table[0, {Length[X] + 1}], -2]];
    {Symbol["c"] -> c[[1 ;; -3]] ~Join~ {cs1, (cs1 + 1)/2, 1},
     Symbol["A"] -> rep[A, A[[-2]]],
     Symbol["R"] -> rep[R, R[[-2]] + Table[If[j == s - 1, cs1, 0], {j, 1, s}]],
     Symbol["B"] -> rep[B, B[[-2]]],
     Symbol["s"] -> s+1, Symbol["ns"] -> ns, Symbol["p"] -> p}
]

inc[mf_] := op@incp@incns[mf]

op:=opx[]
```

```
op50[mf_] := opx[False, 50, 40000]@refine[50, False, 20][mf]

opx[fixcQ_:False, wp_:$MachinePrecision, maxiter_:5000][mf_] :=
  Module[ {s, p, c, A, B, R, cc, a, b, r,
          eq, eq1, M, ll, opt, cond, sol,vars, inits,
          ineq,ns,rs,wx, beta, j, k},
    {s, p, ns} = Symbol/@{"s","p","ns"}/.mf;
    c = Array[cc,{s}];
    cc[s]=1; For[j=ns,j>=1,j--,cc[j]=cc[j+1]-1];
    If[fixcQ, For[j=ns+1, j<=s, j++,  cc[j]=(Symbol["c"]/.mf)[[j]]]];
    A = Array[a, {s, s}];
    B = Array[b, {s, s}];
    R = Table[If[j>=i,0,r[i,j]], {i,1,s},{j,1,s}];
    For[j = 1, j <= ns, j++,
      A[[j, All]] = Table[0, {s}];
      R[[j, All]] = Table[0, {s}];
      B[[j, All]] = Table[0, {s}]; B[[j, j + 1]] = 1];
    eq = Table[Sum[b[i,j],{j,1,s}]==1, {i, ns+1, s}];
    eq1 = Table[-cc[i]^j/j! +
          Sum[b[i,k] (cc[k]-1)^j/j!,{k,1,s}] +
          Sum[a[i,k] If[j==1,1,(cc[k]-1)^(j-1)/(j-1)!],{k,1,s}] +
          Sum[r[i,k] If[j==1,1, cc[k]^(j-1)/(j-1)!],{k,1,i-1}],{i,ns+1,s}, {j,1,p}];
    ineq = {Table[a[i,j]>= 0, {i,ns+1,s}, {j,1,s}],
            Table[b[i,j]>= 0, {i,ns+1,s}, {j,1,s}],
            Table[r[i,j]>= 0, {i,ns+1,s}, {j,1,i-1}]};
     If[Not[fixcQ], AppendTo[ineq, Table[cc[i] >=-s, {i,ns+1,s-1}]]];
     M = Inverse[IdentityMatrix[s]+rs*R].Join[A, R, B-rs A,2];
     ll = Select[Flatten[M], Exponent[#,rs]>0 &];
     cond = And@@Flatten[{eq,Thread[eq1==0], ineq, Thread[ll>=0]}];
     vars =  Flatten[{rs,
               If[Not[fixcQ],Table[cc[i],{i,ns+1,s-1}],{}],
               Table[a[i,j],{i,ns+1,s},{j,1,s}],
               Table[b[i,j],{i,ns+1,s},{j,1,s}],
               Table[r[i,j],{i,ns+1,s},{j,1,i-1}]}];
     inits = Flatten[{0,
         If[Not[fixcQ], Table[(Symbol["c"]/.mf)[[i]],{i,ns+1,s-1}],{}],
                       Table[(Symbol["A"]/.mf)[[i,j]],{i,ns+1,s},{j,1,s}],
                       Table[(Symbol["B"]/.mf)[[i,j]],{i,ns+1,s},{j,1,s}],
                       Table[(Symbol["R"]/.mf)[[i,j]],{i,ns+1,s},{j,1,i-1}]}];
     {rs,sol} = FindMaximum[{rs,cond}, Thread[{vars,inits}],
        MaxIterations->maxiter, WorkingPrecision->wp];
     Print[StringForm["op: Ceff = ‘‘, s = ‘‘, p = ‘‘, ns = ‘‘ ", rs/(s-ns), s, p, ns]];
     {Symbol["s"]    -> s,      Symbol["p"]    -> p,
      Symbol["ns"]   -> ns,     Symbol["c"]    -> c/.sol,
      Symbol["B"]    -> B/.sol, Symbol["A"]    -> A/.sol,
      Symbol["R"]    -> R/.sol, Symbol["r"]    -> rs,
      Symbol["Ceff"]  -> rs/(s-ns)}
]

checkorder[mf_] := Module[{s,p,ns,c,j,ord,rord,vars,A,B,R,a,b,r,x0,t1},
    {s, p, ns} = Symbol/@{"s","p","ns"}/.mf;
    c = Symbol["c"] /. mf;
    A = Array[a,{s,s}];  B = Array[b,{s,s}];
    R = Array[r,{s,s}];
    t1 = Join@@{Thread[Flatten[A]->Flatten@(Symbol["A"]/.mf)],
       Thread[Flatten[B]->Flatten@(Symbol["B"]/.mf)],
       Thread[Flatten[R]->Flatten@(Symbol["R"]/.mf)]};
    ord = Flatten[{
      Table[Sum[b[i, j], {j, 1, s}] - 1, {i, ns + 1, s}],
      Table[- c[[i]]^j/j! + Sum[b[i, k] (c[[k]] - 1)^j/j!, {k, 1, s}]
        + Sum[a[i, k] If[j == 1, 1, (c[[k]] - 1)^(j - 1)/(j - 1)!], {k, 1, s}]
        + Sum[r[i, k] If[j == 1, 1, c[[k]]^(j - 1)/(j - 1)!],
          {k, 1, i - 1}], {i, ns + 1, s}, {j, 1, p}]}];
```

```
   ord/.t1
]

eta[mf_] := Module[{s, ns, p, c, B, A, R,rhs,ees},
  {s,ns,p,c,B,A,R} = Symbol/@{"s","ns","p","c","B","A","R"}/.mf;
    rhs = c^(p + 1)/(p + 1)! - ( B.(c - 1)^(p + 1)/(p + 1)! + A.(c - 1)^p/p! + R.c^p/p!);
    ees = Outer[Times, Table[1, {s}], Append[Table[0, {s - 1}], 1]];
    (Inverse[(IdentityMatrix[s] - B + ees)].rhs)[[s]]
]

refine[acc_:$MachinePrecision, ratQ_:False,cutoff_:12][mf_]:= Module[
     {s, p, ns, c, j, ord, rord, vars, dead, alive,
      A, B, R, a, b, r, x0, t1,x, bs ,L, sol, ref},
    ref[x_] := If[ratQ, Rationalize[x,10^-acc], SetAccuracy[x,acc]];
    {s, p, ns} = Symbol/@{"s","p","ns"}/.mf;
    c = ref[Symbol["c"] /. mf];
    c[[s]] = 1;
    For[j = ns, j >= 1, j--, c[[j]] = c[[j + 1]] - 1];
    A = Array[a,{s,s}];
    B = Array[b,{s,s}];
    R = Array[r,{s,s}];
    t1 = Join@@{
       Thread[Flatten[A]->Flatten@(Symbol["A"]/.ref[mf])],
       Thread[Flatten[B]->Flatten@(Symbol["B"]/.ref[mf])],
       Thread[Flatten[R]->Flatten@(Symbol["R"]/.ref[mf])]};
    dead = Thread[First/@Select[t1,Abs[Last[#]]<=10^-cutoff&]->0];
    alive = Select[t1,Abs[Last[#]]>10^-cutoff&];
    ord = Flatten[{
      Table[Sum[b[i, j], {j, 1, s}] - 1, {i, ns + 1, s}],
      Table[- c[[i]]^j/j!
        + Sum[b[i, k] (c[[k]] - 1)^j/j!, {k, 1, s}]
        + Sum[a[i, k] If[j == 1, 1, (c[[k]] - 1)^(j - 1)/(j - 1)!],
              {k,  1, s}]
        + Sum[r[i, k] If[j == 1, 1, c[[k]]^(j - 1)/(j - 1)!],
           {k, 1, i - 1}], {i, ns + 1, s}, {j, 1, p}]}];
    rord = ord/. dead;
    vars = Variables[rord];
    x0 = vars /. alive;
    {bs, L} = CoefficientArrays[rord, vars];
    x = x0 - PseudoInverse[L].(L.x0 + bs);
    sol = Union[dead,Thread[vars->x]];
   { Symbol["s"] -> s,  Symbol["p"] -> p,
     Symbol["ns"]-> ns, Symbol["c"] -> c,
     Symbol["B"] -> Table[If[i<=ns,If[j==i+1,1,0],b[i,j]],{i,1,s},{j,1,s}]/.sol,
     Symbol["A"] -> Table[If[i<=ns,0,a[i,j]],{i,1,s},{j,1,s}]/.sol,
     Symbol["R"] -> Table[If[(i<=ns)||(j>=i),0,r[i,j]],{i,1,s},{j,1,s}]/.sol}
]

stabplot[mf_] := Module[{cp, t, xx},
  cp = CharacteristicPolynomial[
         Inverse[IdentityMatrix[s] - z R].(B + z A) /. mf, x];
  xx[zz_] := Max[Abs[x /. Solve[(cp /. z -> zz) == 0, x]]] <= 1;
  t[phi_] :=  Select[z /. NSolve[(cp /. x -> Exp[2 I Pi phi ]) == 0, z], xx];
  ListPlot[({Re[#], Im [#]}) & /@ Flatten[t /@ Range[0, 1, 0.002]],
    PlotLabel -> StringForm["s = ‘‘, p = ‘‘, ns = ‘‘, r = ‘‘", s, p, ns, r]/.mf]]

disp[mf_] :=  With[{
    bar = Map[MatrixPlot[(ToExpression[#]/. mf), Frame -> False] &, {"B", "A", "R"}]},
    Print[GraphicsGrid[{Prepend[bar, StringForm["r=‘‘ \n  p=‘‘",
            N[Symbol["r"]/.mf,4],Symbol["p"]/.mf]]}]]; mf]

toml[mf_] :=
 Export["m" <> ToString[s /. mf] <>  ToString[ns /. mf] <>
   ToString[p /. mf] <> ".mat",
  {"s" -> ({{s}} /. mf),
   "p" -> ({{p}} /. mf),
   "ns" -> ({{ns}} /. mf),
   "c" -> Transpose@({c} /. mf),
   "B" -> (B /. mf),
   "A" -> (A /. mf),
   "R" -> (R /. mf),
   "ropt" -> ({{r}} /. mf),
   "eta" -> {{eta[mf]}}
   }, "LabeledData"]
```

Reports of the Institutes 2014

01-14. R. Weiner and J. Bruder, *Exponential Krylov peer integrators*

02-14. B. Soleimani and Chr. Tammer, *Optimality Conditions for Approximate Solutions of Vector Optimization Problems with Variable Ordering Structures*

03-14. Ch. Tammer, V. A. Tuan and C. Zălinescu, *The Lipschitzianity of convex vector and set-valued functions*

Reports are available via WWW: http://www2.mathematik.uni-halle.de/institut/reports/