AFRL-SR-AR-TR-09-0036

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)* 05-02-2009	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From - To)* April 1, 2006- Nov 30, 2008
4. TITLE AND SUBTITLE Implicit high order strong stability preserving Runge--Kutta time discretizations		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER FA9550-06-1-0255
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Sigal Gottlieb		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U of Massachusetts Dartmouth 285 Old Westport Road North Dartmouth MA 02747		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NL 875 N Randolph St Arlington, VA 22203 Dr Fariba Fahroo		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution A: Approved for Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research involved the investigation, development, and testing of diagonally split Runge-Kutta (DSRK) methods and implicit Runge—Kutta methods with reference to their strong stability preserving (SSP) properties for large time-steps.
The research found that DSRK methods which are unconditionally SSP reduce to first order for the stepsizes of interest, and the PI introduced an analysis which explains this phenomenon and shows that it is unavoidable. The PI and her students developed a methodology for finding optimal implicit SSP Runge--Kutta methods up to order six (which is the maximal possible order for these methods) and eleven stages, and found that the effective SSP coefficient can be no more than two, making these methods not competitive with explicit methods for most applications, but useful in a carefully chosen subset of problems. The results of this grant are a complete analysis of implicit SSP Runge--Kutta methods and the SSP properties, which demonstrate the need for the SSP property in solutions

15. SUBJECT TERMS

strong stability preserving, Runge—Kutta methods, time-discretizations, hyperbolic problems with shocks.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Sigal Gottlieb
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	41	**19b. TELEPHONE NUMBER** *(include area code)* 401-751-9416

Implicit high order strong stability preserving Runge–Kutta time discretizations

AFOSR grant number FA9550-06-1-0255 Sigal Gottlieb
Mathematics Department
University of Massachusetts – Dartmouth

Abstract
We investigated diagonally split Runge-Kutta (DSRK) methods to identify and test unconditionally strong stability preserving (SSP) methods, and implicit SSP time-stepping methods to find methods with a large SSP coefficient. We found that DSRK methods which are unconditionally SSP reduce to first order for the stepsizes of interest, and introduced an analysis which explains this phenomenon and shows that it is unavoidable. We found optima; implicit SSP Runge–Kutta methods up to order six (which is the maximal possible order for these methods) and eleven stages, and found that the effective SSP coefficient can be no more than two, making these methods not competitive with explicit methods for most applications, but useful in a carefully chosen subset of problems. We now have a complete analysis of implicit SSP Runge–Kutta methods and demonstrations of the need for the SSP property in solutions of hyperbolic PDEs with shocks.

1 Summary of Aims and Results

Strong stability preserving (SSP) high order time discretizations were developed to ensure nonlinear stability properties necessary in the numerical solution of hyperbolic partial differential equations with discontinuous solutions. SSP methods preserve the strong stability properties – in any norm, seminorm or convex functional – of the spatial discretization coupled with first order Euler time stepping, when the timestep is suitably restricted. Explicit strong stability preserving (SSP) Runge-Kutta methods ([17], [18],[19], [20], [4],[5], [6]) have been successfully used with a wide range of spatial discretizations, including spectral, discontinuous Galerkin, and weighted essentially non-oscillatory (WENO) methods. These high order methods preserve any nonlinear stability properties satisfied by the spatial discretization coupled with the forward Euler time-stepping. However, all general linear methods suffer from a SSP time-step restriction. This motivates the search for high order implicit time-stepping methods with SSP properties and a large allowable time-step, which is the overarching goal of this project.

The connections between the SSP property and the theory of contractivity have provided efficient tools for the study of SSP multistep and Runge-Kutta methods, which we utilized in the search for optimal implicit SSP Runge–Kutta methods. Furthermore, contractivity theory allowed us to determine order barriers on SSP methods, to establish bounds on the SSP coefficient, and to conclude that the SSP coefficient is not only sufficient but necessary for strong stability preservation in an arbitrary norm for an arbitrary semi-discretization that satisfies a strong stability condition under forward Euler integration.

This work described below was performed in collaboration with David Ketcheson, a doctoral candidate at University of Washington in Seattle (advised by Dr. R. LeVeque), and Colin Macdonald, then a doctoral candidate at Simon Fraser University (advised by Dr. S. Ruuth) and now a postdoctoral fellow at UCLA (working with Dr. S. Osher).

The aims of AFOSR grant number FA9550-06-1-0255 were to

- Use results and formulations from contractivity and monotonicity theory to find optimal class of higher order implicit SSP Runge-Kutta methods.
- Find higher order implicit diagonally split Runge-Kutta methods (DSRK) which are SSP methods with no stepsize restriction.
- Test the optimal implicit SSP Runge-Kutta methods for use with flux-implicit WENO spatial discretizations.
- Test the DSRK with no time-step restriction with spectral and WENO spatial discretizations.

In the grant period we have gone further than we proposed or anticipated. The following are the accomplishments under this grant:

1. We conducted a thorough numerical study of second and third order diagonally split Runge–Kutta methods on a variety of problems. These methods have proved disappointing, due to severe reduction of order which renders them no better than backward Euler, which is unconditionally SSP [16]. We analyzed the cause of this order reduction and found a way to avoid it, however this renders the SSP coefficient as small as for implicit Runge–Kutta methods.
2. The connections between the SSP property and the theory of contractivity optimal have provided efficient tools for the study of SSP multistep and Runge-Kutta methods. Methods of these types have been thoroughly investigated, and their development seems to be essentially complete. Furthermore, contractivity theory allowed us to determine order barriers on SSP methods, to establish bounds on the SSP coefficient, and to conclude that the SSP coefficient is not only sufficient but necessary for strong stability preservation in an arbitrary norm for an arbitrary semi-discretization that satisfies a strong stability condition under forward Euler integration.

3. We found optimal implicit SSP Runge–Kutta methods up to order six and up to eleven stages. These methods are diagonally implicit or singly diagonally implicit and have sparse, efficient representations. Furthermore, the implicit solutions at each stage of a SSP Runge-Kutta method have provable existence and uniqueness properties.

4. Our work demonstrated that implicit SSP methods are unlikely to be efficient enough to out-perform the explicit methods. We define the *effective SSP coefficient* of a method $c_{eff} = \frac{c}{m}$ to normalize the step-size coefficient c relative to the number of stages m in a method. The very restrictive bound $c_{eff} \leq 2$ has been proven for implicit multistep methods [15, 10] and conjectured for implicit Runge-Kutta methods [12]. In contrast, explicit methods have a bound $c_{eff} \leq 1$.

5. In the wider class of explicit general linear methods (which includes both Runge–Kutta and multistep methods as a subset) the bound $c_{eff} \leq 1$ was proved [7].

6. Although the focus of this grant was implicit Runge-Kutta methods, the tools developed for this grant allowed David Ketcheson to independently perform a more thorough study of explicit low-storage Runge-Kutta methods [13] as well as implicit and explicit multi-step methods [7]. We found that the SSP Runge-Kutta methods tend to have a variety of nice properties, such as small error constants and large regions of absolute stability.

7. We showed that spectral deferred correction methods can be written as Runge–Kutta method, and are thus amenable to the techniques for efficient optimization found using the connections to contractivity theory. Using these connections, we also conclude that these methods suffer from the same order barriers and bounds on the SSP coefficient.

8. David Ketcheson further studied the SSP properties of the Runge–KuttaChebyshev methods. Verwers second order methods all have negative Butcher coe?cients, so they are not SSP under any positive timestep. We have found first and second order SSP methods up to 10 stages that have the theoretically optimal timestep. These are promising for fully explicit integration of convection-di?usion equations without operator splitting. Unlike IMEX, exponential di?erencing, etc., they apply the same integration method to the sti? and non-sti? parts)

1.1 Publications:

Publications resulting from this grant are:

1. "A numerical study of diagonally split Runge-Kutta methods for PDEs with discontinuities" by C.B. Macdonald, S. Gottlieb, and S. Ruuth. Journal of Scientific Computing, 36(1):89-112, (2008).

2. "Optimal implicit strong stability preserving Runge-Kutta methods" by D. Ketcheson, C. Macdonald, and S. Gottlieb. Applied Numerical Mathematics (to appear).

3. "Highly E?cient Strong Stability Preserving Runge-Kutta Methods with Low Storage Implementations" by D. Ketcheson. SIAM Journal on Scientic Computing, 30 (4): 2113-2136 (2008). Winner of the SIAM student paper prize.

4. "Computation of optimal monotonicity preserving general linear methods" by David I. Ketcheson. Math. of Comp. (2008)

5. "High Order Strong Stability Preserving Time Discretizations" by S. Gottlieb, D.I. Ketcheson and C.-W. Shu. Journal of Scientific Computing 38:251-289 (2009).

1.2 Dissemination

Other dissemination efforts related to this grant:

1. We set up a web-site devoted to SSP methods, to collect all the latest results and most useful information about strong stability preserving time discretizations. http://www.cfm.brown.edu/people/sg/ssp.html

2. We organized a minisymposium at the 2006 annual SIAM conference which brought together Rong Wang (who presented his joint work with Ray Spiteri), Inma Higueras, Steven Ruuth and his student Colin Macdonald. This minisymposia led to productive discussions with Adrian Sandu and his student on the topic of SSP multirate time-stepping.

   (a) *Positivity and Monotonicity for IMEX Methods* by **Inmaculada Higueras**, Universidad Pblica de Navarra, Spain.

   (b) *Variable Step-Size IMplicit-EXplicit Linear Multistep Methods* by **Steve Ruuth**, Simon Fraser University, Canada; Dong Wang, University of Illinois at Urbana-Champaign.

   (c) *In Search of Implicit High-Order Strong Stability Preserving Methods with Relaxed Time-Step Restrictions* Sigal Gottlieb, University of Massachusetts; **Colin Macdonald**, Simon Fraser University; Steve Ruuth, Simon Fraser University, Canada.

   (d) *Comments on Linear Instability of Time Integration Methods with the Fifth-Order WENO Spatial Discretization* Raymond J. Spiteri and **Rong Wang**, University of Saskatchewan, Canada.

3. We have organized a minisymposium which will take place at the 2008 SIAM annual meeting, which will feature the following:

(a) *Strong Stability Preserving Time-Stepping Methods* by **Sigal Gottlieb**, University of Massachusetts, Dartmouth; David Ketcheson, University of Washington; Colin Macdonald, Simon Fraser University

(b) *Optimal Explicit and Implicit SSP Runge-Kutta Methods* by **David I. Ketcheson**, University of Washington; **Colin Macdonald**, Simon Fraser University; Sigal Gottlieb, University of Massachusetts, Dartmouth;

(c) *Practical considerations for IMEX SSP Runge-Kutta methods* by **Inmaculada Higueras**, Universidad Pblica de Navarra, Spain; Teo Roldán.

(d) *Generalizations of Positivity and Strong Stability Preservation* by **Zoltan Horvath**, Szchenyi Istvan University, Gyr, Hungary.

(e) *High Order Discretizations of Kinetic Equations* by **Lorenzo Pareschi**, University of Ferrara, Italy.

(f) *Multirate SSP Methods for Hyperbolic PDEs* by **Emil Constantinescu** and Adrian Sandu, Virginia Polytechnic Institute & State University.

(g) *Do We Know WENO?* by **Raymond J. Spiteri** and Rong Wang, University of Saskatchewan, Canada.

(h) *Stage-exceeding Order SSP Time-stepping for Runge-Kutta Discontinuous Galerkin Methods* by **Clint Dawson**, University of Texas, Austin; Ethan Kubatko, University of Texas at Austin.

4. Seminar presentation "Strong Stability Preserving time discretizations with optimal time-step restrictions" at UMass Amherst on October 30, 2007.

5. Workshop presentation "Strong Stability Preserving Time Discretizations" at the Statistical and Applied Mathematical Sciences Institute's (SAMSI) 2007-2008 Program on Random Media Interface Problems Workshop in North Carolina on November 15, 2007.

6. Seminar presentation "On Strong Stability Preserving Runge-Kutta and Multistep Time Discretizations" at the (NYU) Courant Institute's Numerical Analysis and Scientific Computing seminar on November 30, 2007.

7. Seminar presentation "Time stepping methods for numerical solution of hyperbolic PDEs with shocks" in MIT's Mathematics Department's Numerical Methods for Partial Differential Equations seminar on November 12, 2008.

8. Seminar presentation "Time stepping methods for numerical solution of hyperbolic PDEs with shocks" Mathematics Department Colloquium in The University of Connecticut – Storrs on November 13, 2008.

9. Book contract with for World Scientific Publishing for a monograph on SSP time-discretization methods (together with C.-W. Shu).

These minisymposia, seminars, workshops, and website have caused the topic of time-stepping to be more widely discussed and studied, and has inspired collaborations and other research on the topic.

2 Detailed Progress By Year:

Year 1: We studied the class of diagonally split Runge–Kutta methods to find high order, unconditionally SSP methods. Diagonally split Runge–Kutta (DSRK) ([1, 2, 8, 9]) time discretization methods are a class of implicit time-stepping schemes which offer both (formal) high-order convergence and a form of nonlinear stability known as unconditional contractivity. This combination is not possible within the classes of Runge–Kutta or linear multistep methods and therefore appears promising for the strong stability preserving (SSP) time-stepping community which is generally concerned with computing oscillation-free numerical solutions of PDEs.

We conducted a thorough numerical study of second and third order diagonally split Runge–Kutta methods on a variety of of archetypal test cases including linear advection, Burgers' equation, a diffusion equation with discontinuous initial data, and the Black-Scholes equation. The numerical tests verified the asymptotic order of the schemes as well as the unconditional contractivity property. However, in every numerical experiment, diagonally split Runge–Kutta methods suffer from order reduction at large step-sizes. Indeed, for time-steps larger than those typically chosen for explicit methods, these diagonally split Runge–Kutta methods behave like first-order implicit methods. In every numerical experiment, the unconditionally contractive diagonally split Runge–Kutta methods were out-performed by the first-order backward Euler scheme when $\Delta t > 2\Delta t_{FE}$, and by explicit Runge–Kutta methods or Crank–Nicolson when $\Delta t \leq 2\Delta t_{FE}$. At larger time-steps, the unconditionally contractive diagonally split Runge–Kutta schemes are strong stability preserving (SSP) but suffer from order reduction, making backward Euler a better choice. At small step-sizes, Crank–Nicolson and explicit SSP Runge–Kutta methods are SSP, and produce far more accurate results at a smaller computational cost. Indeed, for time-steps larger than those typically chosen for explicit methods, these DSRK methods behave like first-order implicit methods. This is unfortunate, because it is precisely to allow a large time-step that we choose to use implicit methods. We studied this order reduction phenomenon analytically, and showed that higher stage order of the underlying Runge–Kutta schemes was insufficient to avoid order reduction. We then derived DSRK stage order conditions and constructed DSRK schemes with higher stage which do not suffer from order reduction. However, because of the high stage order, these schemes cannot be unconditionally contractive, and the resulting SSP coefficient are comparable to implicit Runge–Kutta [16].

Year 2: In the second year of the project we surveyed the literature on contractive methods and extracted results which are applicable to SSP methods, identified efficient techniques to find the radius of absolute monotonicity, and found optimal implicit SSP Runge–Kutta methods of order up to six.

Using the results from contractivity theory, we were able to identify the following order barriers and bounds on the SSP coefficient of Runge–Kutta, multistep, and general linear methods:

- Runge–Kutta Methods

  1. An SSP Runge–Kutta method with can be no more than fourth order accurate if it is explicit and no more than sixth order accurate if it is implicit [14].
  2. Implicit Runge–Kutta methods that are unconditionally SSP must have order at most one. This result is in contrast with linear stability and B-stability, where some high-order implicit methods (i.e., the A-stable methods and the algebraically stable methods, respectively) are unconditionally stable.
  3. The implicit SSP Runge–Kutta of order $p > 1$ have an SSP coefficient that is not dramatically larger than those for explicit methods [15, 3, 12].
  4. Any SSP method must have stage order $\tilde{p} \leq 2$, and explicit Runge–Kutta method must have stage order $\tilde{p} \leq 1$. The stage order $\tilde{p}$ is a lower bound on the order of convergence when a method is applied to arbitrarily stiff problems. Low stage order may lead to *order reduction*, i.e. slow convergence, when computing solutions of stiff ODEs.
  5. All m-stage diagonally implicit methods have order at most $m + 1$.
  6. All SSP m-stage singly diagonally implicit methods have order at most $m + 1$.
  7. SSP singly diagonally implicit methods, which are both singly implicit and diagonally implicit, have the same order barrier ($p \leq 4$) as explicit methods.

- Multistep Methods

  1. For explicit s-step methods of order p, the SSP coefficient is bounded by $c \leq \frac{s-p}{s-1}$ for $s > 1$
  2. for implicit methods of order $p > 1$, the SSP coefficient is bounded by $c \leq 2$.
  3. While there appears to be no limit to the order of accuracy of SSP linear multistep methods, high order accurate methods of this type are subject to very small timestep restrictions and require very many steps.

- General Linear Methods
  Any explicit m-stage, s-step general linear method of order p, has SSP coefficient bounded by the number of its stages, $c \leq m$.

Although no unconditionally SSP method can have order greater than one [21], we explored the possibility that implicit methods may have SSP coefficients significantly larger than those of explicit methods with the same order and number of stages. The question we wished to answer was whether the allowable step-size can be large enough to offset the extra computational effort required in the implicit solution of the resulting system at each iteration.

Using the efficient formulation of the problem of finding the radius of contractivity of a method, it was possible to use MATLAB to perform a search for optimal implicit SSP Runge–Kutta methods. These results gave us optimal methods of order up to six, which is the maximal order for implicit SSP Runge–Kutta methods. In fact, only existence of methods of order up to five was previously established [14]. Our search successfully found methods of order six, establishing that this is indeed possible and that the order barrier is sharp.

Recently, Ferracina and Spijker investigated optimal singly diagonally implicit Runge–Kutta methods [3]. They showed that such methods have order at most four, and found optimal methods (by numerical optimization) of up to order four and up to eight stages. They also conjectured the form of optimal second and third order methods with any number of stages. Using numerical optimization techniques, we performed an extensive search among the much larger class of *fully implicit* SSP Runge-Kutta methods [12]. Remarkably, searching among the class of fully implicit methods, the optimal methods of second and third order were found to be singly diagonally implicit; in fact, they were the very methods found already in [3]. The optimal methods of fourth through sixth order were found to be diagonally implicit. Many of these implicit methods have representations that allow for very efficient implementation in terms of storage. In order to accurately measure the efficiency of these methods, we define the *effective SSP coefficient* of a method as $c_{eff} = \frac{c}{m}$; this normalization enables us to compare the cost of integration up to a given time using diagonally implicit schemes of order $p > 1$. Unfortunately, the optimal implicit SSP methods have effective SSP coefficient less than or equal to two, making them probably too inefficient for practical use. We list effective SSP coefficients of the numerically optimal methods in Table 2.1. The coefficients of the most efficient representations of SSP implicit Runge–Kutta methods are available online [11].

The SSP condition provides a guarantee of other necessary properties. When considering implicit Runge-Kutta methods, it is important to determine whether there exists a unique solution of the stage equations. The strong stability preserving timestep restriction turns out to be sufficient for this as well [14, Theorem 7.1]. Furthermore, the SSP condition serves to guarantee that the errors introduced in the

	Implicit Methods				
m / p	2	3	4	5	6
1	2	-	-	-	-
2	2	1.37	-	-	-
3	2	1.61	0.68	-	-
4	2	1.72	1.11	0.29	
5	2	1.78	1.21	0.64	
6	2	1.82	1.30	0.83	0.030
7	2	1.85	1.31	0.89	0.038
8	2	1.87	1.33	0.94	0.28
9	2	1.89	1.34	0.99	0.63
10	2	1.90	1.36	1.01	0.81
11	2	1.91	1.38	1.03	0.80

Table 2.1: Effective SSP coefficients of best known implicit methods. A dash indicates that SSP methods of this type cannot exist. A blank space indicates that no SSP methods of this type were found.

solution of the stage equations due to numerical roundoff and (for implicit methods) errors in the implicit solve are not unduly amplified [14, Theorem 7.2].

In summary, we have gone further than we proposed or anticipated possible in the study of SSP implicit Runge-Kutta methods. We have found optimal methods of order up to six and up to eleven stages, which are diagonally implicit and which have sparse representations, thus making them more efficient for implementation. We also have enough information to conjecture that the optimal effective SSP coefficient over this class of methods is bounded by $c_{eff} \leq 2$.

In addition to our results, the methodology we adopted in this search also led to work that is beyond the scope of this grant. David Ketcheson has used the ideas and techniques developed in the process of this research to find low storage optimal explicit SSP Runge–Kutta methods of order up to four and of many stages [13].

Year 3: The next step in our research involved the preliminary testing of implicit and explicit SSP methods on a variety of problems. We carried out many numerical experiments which showed the need for, and benefit of SSP methods.

Using a nonlinear example, we showed that even when the spatial discretization is total variation diminishing (TVD) when coupled with forward Euler integration, this is not sufficient to guarantee that it will be TVD when combined with a higher order time-discretization. We considered Burgers' equation with a sine wave initial condition and periodic boundary conditions. The solution is right-travelling and over time steepens into a shock. We discretize using a first order conservative upwind approximation which is TVD for $\Delta t \leq \Delta x$ when coupled with forward Euler. Using this fact we can conclude that if we integrate, instead, using backward Euler, the solution will be TVD for all values of Δt. However, when coupled with second-order A-stable implicit trapezoidal rule or the A-stable, L-stable, and B-stable implicit midpoint rule, this is not TVD for $\Delta t > 2\Delta x$.

Using a non-SSP explicit Runge–Kutta with a second order TVD flux-differencing method with the superbee slope limiter, we further demonstrated that the timestep restriction associated with the linear SSP property does not suffice to give reasonably good behavior in the nonlinear case.

We also performed experiments of SSP methods coupled with the weighted essentially non-oscillatory method. We observe advantages to the use of SSP methods for WENO methods on linear and nonlinear problems. The time-step at which the total variation begins to rise by more than 10^{-13} is much higher for the SSP methods than for the corresponding non-SSP methods. We observe that in each case the timestep restriction for L_2 linear stability is larger than that required for the TVD property, and that the non-SSP method is less efficient than the SSP methods.

For SSP Runge-Kutta methods, it is desirable that the internal stages also be strongly stable. This means requiring not only that $||u^{n+1}|| \leq ||u^n||$, but also that each stage $u^{(i)}$ for $i = 1, ..., m$ satisfy $||u^{(i)}|| \leq ||u^{(i-1)}||$. Since the SSP argument relies on convexity, which is satisfied at the intermediate stages as well, SSP Runge-Kutta methods have intermediate stage SSP properties. The SSP guarantee of provable stability even for the intermediate stages is given with no additional cost. This condition is frequently necessary in the approximate solution of hyperbolic PDEs. For example, in the numerical solution of the Euler equations of gas dynamics, it is important that negative pressure or density values be avoided even in the intermediate stages. Violations of these bounds are more than theoretically problematic, as they lead to non-physical states and typically to failure of the solution algorithm. We considered the Riemann problem for the Euler equations with fifth-order WENO used for the spatial-discretization. When we determined the largest CFL number σ for which the density and pressure values remain positive at all Runge-Kutta stages, we find that we see that the SSP methods allow a more efficient time-step than the non-SSP methods.

We examined the class of spectral deferred correction methods methods, and demonstrated that they can be written as explicit Runge-Kutta methods. Using this fact, we can immediately establish bounds on the SSP coefficient of spectral deferred correction methods and also conclude that downwind operators will be required in order for explicit spectral DC methods to be SSP if they are of order greater than four. Similarly, implicit spectral DC methods cannot be SSP without downwinding if their order exceeds six.

Finally, David Ketcheson independently studied the SSP properties of the Runge–Kutta Chebyshev methods. Verwers second order methods all have negative Butcher coe?cients, so they are not SSP under any positive timestep. He found first and second order SSP methods up to 10 stages that have the theoretically optimal time-step. These are promising for fully explicit integration of convection-di?usion equations without operator splitting. Unlike IMEX, exponential di?erencing, etc., they apply the same integration method to the sti? and non-sti? parts)

3 Transitions

Guowei Wei (Michigan State University) and Shan Zhao have implemented our SSP methods in their matched interface and boundary method to obtain high order schemes in both space and time for hyperbolic equations. They report that "Your SSP methods work great!".

Zhilin Li at North Carolina State University requested the coefficients of the second order tow-stage implicit SSP scheme to use these with free boundary/moving interface problems for which stability is always an issue. I was able to advise him on how to apply this most efficiently.

Marsha Berger (NYU) and Uri Shumlak (University of Washington) requested the SSP review paper. Additionally, Marsha Berger requested that I recommend specific SSP methods from the paper.

Francis X. Giraldo (Naval Postgraduate School in Monterey, CA) contacted me asking about the theoretical limits on the order of SSP explicit methods.

4 Acknowledgement/Disclaimer

This work was sponsored (in part) by the Air Force Office of Scientific Research, USAF, under grant/contract number FA9550-06-1-0255. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

References

[1] A. Bellen, Z. Jackiewicz, and M. Zennaro, *Contractivity of waveform relaxation Runge-Kutta iterations and related limit methods for dissipative systems in the maximum norm.* SIAM J. Num. Anal., 31(2): 499-523, 1994.

[2] A. Bellen and L. Torelli. *Unconditional contractivity in the maximum norm of diagonally split Runge-Kutta methods.* SIAM J. Num. Anal. 34(2):528-543, 1997.

[3] L. Ferracina and M.N. Spijker. Strong stability of singly-diagonally-implicit Runge-Kutta methods. *Applied Numerical Mathematics*, 2008. doi: 10.1016/j.apnum.2007.10.004

[4] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Mathematics of Computation, **67**, 1998, pp.73-85.

[5] S. Gottlieb, C.-W. Shu and E. Tadmor, *Strong Stability Preserving High-Order Time Discretization Methods*, SIAM Review, **43**, 2001, pp.89-112.

[6] S. Gottlieb, *On High Order Strong Stability Preserving Runge-Kutta and Multi Step Time Discretizations.* Journal of Scientific Computing **vol. 25** (2005), pp. 105-128.

[7] S. Gottlieb, D.I. Ketcheson and C.-W. Shu. High Order Strong Stability Preserving Time Discretizations. Journal of Scientific Computing 38:251-289 (2009).

[8] Z. Horvath. *Positivity of Runge-Kutta and diagonally split Runge-Kutta methods.* Appl. Numer. Math. 28:309-326.

[9] K.J. In'T Hout, *A note on unconditional maximum norm contractivity of diagonally split Runge-Kutta methods.* SIAM J. on Num. Anal. 33:1125-1134, 1996.

[10] W. Hundsdorfer, S.J. Ruuth and R.J. Spiteri. Monotonicity-preserving linear multistep methods. *SIAM Journal on Numerical Analysis*, 41:605–623, 2003.

[11] D.I. Ketcheson, C.B. Macdonald, S. Gottlieb. Strong Stability Preserving Methods (website). `http://www.cfm.brown.edu/people/sg/ssp.html` (2007).

[12] D.I. Ketcheson, C.B. Macdonald and S. Gottlieb. Optimal implicit strong stability preserving Runge-Kutta methods. To appear in *Applied Numerical Mathematics*, doi: 10.1016/j.apnum.2008.03.034.

[13] D.I. Ketcheson. Highly efficient strong stability preserving Runge-Kutta methods with low-storage implementations. SIAM Journal on Scientic Computing, 30 (4): 2113-2136 (2008).

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

[15] H.W.J. Lenferink. Contractivity-preserving implicit linear multistep methods. *Mathematics of Computation*, 56:177–199, 1991.

[16] C.B. Macdonald, S. Gottlieb, and S. Ruuth. A numerical study of diagonally split Runge-Kutta methods for PDEs with discontinuities. Journal of Scientific Computing, 36(1):89-112, (2008).

[17] C.-W. Shu, *Total-variation-diminishing time discretizations*, SIAM Journal on Scientific and Statistical Computing, **9**, 1988, pp.1073-1084.

[18] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, Journal of Computational Physics, **77**, 1988, pp.439-471.

[19] C.-W. Shu, *A survey of strong stability preserving high order time discretizations*, in **Collected Lectures on the Preservation of Stability under Discretization**, D. Estep and S. Tavener, editors, SIAM, 2002, pp.51-65.

[20] R.J. Spiteri and S.J. Ruuth, *A new class of optimal high-order strong-stability-preserving time discretization methods, SIAM Journal on Numerical Analysis,* **40**, *2002, pp.469-491.*

[21] M.N. Spijker. Contractivity in the numerical solution of initial value problems. Numerische Mathematik, *42:271–290, 1983.*

5 Appendix I

The optimal implicit Runge–Kutta methods found under this grant

Shu-Osher Coefficients for Second Order methods: The optimal s-stage second order implicit SSP Runge-Kutta method has SSP coefficient 2s and Shu-Osher form

$$\lambda = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & 1 & \ddots & \\ & & \ddots & 0 \\ & & & 1 \end{bmatrix}, \quad \mu = \begin{bmatrix} \frac{1}{2s} & & & \\ \frac{1}{2s} & \frac{1}{2s} & & \\ & \frac{1}{2s} & \ddots & \\ & & \ddots & \frac{1}{2s} \\ & & & \frac{1}{2s} \end{bmatrix}.$$

Shu-Osher Coefficients for Third Order methods: The optimal s-stage third order implicit SSP Runge-Kutta method has SSP coefficient $s - 1 + \sqrt{s^2 - 1}$ and Shu-Osher form

$$\mu = \begin{bmatrix} \mu_{11} & & & \\ \mu_{21} & \ddots & & \\ & \ddots & \mu_{11} & \\ & & \mu_{21} & \mu_{11} \\ & & & \mu_{s+1,s} \end{bmatrix}, \quad \lambda = \begin{bmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & 0 & \\ & & 1 & 0 \\ & & & \lambda_{s+1,s} \end{bmatrix},$$

where

$$\mu_{11} = \frac{1}{2}\left(1 - \sqrt{\frac{s-1}{s+1}}\right), \quad \mu_{21} = \frac{1}{2}\left(\sqrt{\frac{s+1}{s-1}} - 1\right),$$

$$\mu_{s+1,s} = \frac{s+1}{s(s+1+\sqrt{s^2-1})}, \quad \lambda_{s+1,s} = \frac{(s+1)(s-1+\sqrt{s^2-1})}{s(s+1+\sqrt{s^2-1})}.$$

Shu-Osher Coefficients for Fourth Order methods:

1. Optimal 3-stage, 4th-order method:

$$\begin{aligned} \mu_{11} &= 0.157330905682085 & \mu_{21} &= 0.342491639470766 \\ \mu_{22} &= 0.047573123554705 & \mu_{32} &= 0.338136048168635 \\ \mu_{33} &= 0.157021682372699 & \mu_{41} &= 0.081822264233578 \\ \mu_{42} &= 0.079106848361263 & \mu_{43} &= 0.267698531248384 \\ \lambda_{21} &= 0.703541497995214 & \lambda_{32} &= 0.694594303739345 \\ \lambda_{41} &= 0.168078141811591 & \lambda_{42} &= 0.162500172803529 \\ \lambda_{43} &= 0.549902549377947 \end{aligned}$$

2. Optimal 4-stage, 4th-order method:

$\mu_{11} = 0.119309657880174$	
$\mu_{21} = 0.226141632153728$	$\mu_{22} = 0.070605579799433$
$\mu_{32} = 0.180764254304414$	$\mu_{33} = 0.070606483961727$
$\mu_{43} = 0.212545672537219$	$\mu_{44} = 0.119309875536981$
$\mu_{51} = 0.010888081702583$	$\mu_{52} = 0.034154109552284$
$\mu_{54} = 0.181099440898861$	$\lambda_{21} = 1$
$\lambda_{32} = 0.799340893504885$	$\lambda_{43} = 0.939878564212065$
$\lambda_{51} = 0.048147179264990$	$\lambda_{52} = 0.151029729585865$
$\lambda_{54} = 0.800823091149145$	

3. Optimal 5-stage, 4th-order method:

$\mu_{11} = 0.072154507748981$	$\mu_{21} = 0.165562779595956$
$\mu_{22} = 0.071232036614272$	$\mu_{32} = 0.130035287184462$
$\mu_{33} = 0.063186062090477$	$\mu_{43} = 0.154799860761964$
$\mu_{44} = 0.077017601068238$	$\mu_{54} = 0.158089969701175$
$\mu_{55} = 0.106426690493882$	$\mu_{65} = 0.148091381629243$
$\mu_{52} = 0.007472809894781$	$\mu_{62} = 0.017471397966712$
$\lambda_{21} = 1$	$\lambda_{32} = 0.785413771753555$
$\lambda_{43} = 0.934991917505507$	$\lambda_{54} = 0.954864191619538$
$\lambda_{65} = 0.894472670673021$	$\lambda_{52} = 0.045135808380468$
$\lambda_{62} = 0.105527329326976$	

4. Optimal 6-stage, 4th-order method:

$\mu_{11} = 0.077219435861458$	$\mu_{21} = 0.128204308556198$
$\mu_{22} = 0.063842903854499$	$\mu_{32} = 0.128204308556197$
$\mu_{33} = 0.058359965096908$	$\mu_{41} = 0.008458154338733$
$\mu_{43} = 0.103230521234296$	$\mu_{44} = 0.058105933032597$
$\mu_{54} = 0.128204308556197$	$\mu_{55} = 0.064105484788524$
$\mu_{63} = 0.008043763906343$	$\mu_{65} = 0.120160544649854$
$\mu_{66} = 0.077016336936138$	$\mu_{73} = 0.013804194371285$
$\mu_{76} = 0.114400114184912$	$\lambda_{21} = 1$
$\lambda_{32} = 1$	$\lambda_{41} = 0.065974025631326$
$\lambda_{43} = 0.805203213502341$	$\lambda_{54} = 1$
$\lambda_{63} = 0.062741759593964$	$\lambda_{65} = 0.937258240406037$
$\lambda_{73} = 0.107673404480272$	$\lambda_{76} = 0.892326595519728$

5. Optimal 7-stage, 4th-order method:

$\mu_{11} = 0.081324471088377$	$\mu_{21} = 0.108801609187400$
$\mu_{22} = 0.051065224656204$	$\mu_{32} = 0.108801609187400$
$\mu_{33} = 0.036491713577701$	$\mu_{43} = 0.094185417979586$
$\mu_{44} = 0.037028821732794$	$\mu_{54} = 0.108801609187400$
$\mu_{55} = 0.040474271914787$	$\mu_{65} = 0.108801609187400$
$\mu_{66} = 0.061352000212100$	$\mu_{73} = 0.020631403945188$
$\mu_{76} = 0.088170205242212$	$\mu_{77} = 0.080145231879588$
$\mu_{83} = 0.001561606596621$	$\mu_{87} = 0.107240002590779$
$\lambda_{21} = 1$	$\lambda_{32} = 1$
$\lambda_{43} = 0.865661994183934$	$\lambda_{54} = 1$
$\lambda_{65} = 1$	$\lambda_{73} = 0.189624069894518$
$\lambda_{76} = 0.810375930105481$	$\lambda_{83} = 0.014352789524754$
$\lambda_{87} = 0.985647210475246$	

6. Optimal 8-stage, 4th-order method:

$\mu_{11} = 0.080355939553359$	$\mu_{21} = 0.093742212796061$
$\mu_{22} = 0.054617345411549$	$\mu_{32} = 0.093742212796061$
$\mu_{33} = 0.039438131644116$	$\mu_{43} = 0.093742212796061$
$\mu_{44} = 0.032427875074076$	$\mu_{51} = 0.004426522032754$
$\mu_{54} = 0.083174746150582$	$\mu_{55} = 0.030116385482588$
$\mu_{65} = 0.093742212796061$	$\mu_{66} = 0.038334326442344$
$\mu_{76} = 0.093742212796061$	$\mu_{77} = 0.058861620081910$
$\mu_{84} = 0.021977226754808$	$\mu_{87} = 0.071764986041253$
$\mu_{88} = 0.055606577879005$	$\mu_{98} = 0.093742212796061$
$\lambda_{21} = 1$	$\lambda_{32} = 1$
$\lambda_{43} = 1$	$\lambda_{51} = 0.047220157287989$
$\lambda_{54} = 0.887270992114641$	$\lambda_{65} = 1$
$\lambda_{76} = 1$	$\lambda_{84} = 0.234443225728203$
$\lambda_{87} = 0.765556774271797$	$\lambda_{98} = 1$

7. Optimal 9-stage, 4th-order method:

$\mu_{11} = 0.068605696784244$	$\mu_{21} = 0.082269487560004$
$\mu_{22} = 0.048685583036902$	$\mu_{32} = 0.077774790319743$
$\mu_{33} = 0.039925150083662$	$\mu_{43} = 0.083046524401968$
$\mu_{44} = 0.031928917146492$	$\mu_{54} = 0.083046524401968$
$\mu_{55} = 0.029618614941264$	$\mu_{61} = 0.008747971137402$
$\mu_{62} = 0.001326570052113$	$\mu_{65} = 0.072971983212453$
$\mu_{66} = 0.029699905991308$	$\mu_{76} = 0.083046524401968$
$\mu_{77} = 0.035642110881905$	$\mu_{87} = 0.083046524401969$
$\mu_{88} = 0.050978240433952$	$\mu_{95} = 0.017775897980583$
$\mu_{98} = 0.065270626421385$	$\mu_{99} = 0.057552171403649$
$\mu_{10,9} = 0.083046524401968$	$\lambda_{21} = 0.990643355064403$
$\lambda_{32} = 0.936520713898770$	$\lambda_{43} = 1$
$\lambda_{54} = 1$	$\lambda_{61} = 0.105338196876962$
$\lambda_{62} = 0.015973817828813$	$\lambda_{65} = 0.878687985294225$
$\lambda_{76} = 1$	$\lambda_{87} = 1$
$\lambda_{95} = 0.214047464461523$	$\lambda_{98} = 0.785952535538477$
$\lambda_{10,9} = 1$	

8. Optimal 10-stage, 4th-order method:

$\mu_{11} = 0.053637857412307$	$\mu_{21} = 0.073302847899924$
$\mu_{22} = 0.042472343576273$	$\mu_{32} = 0.063734820131903$
$\mu_{33} = 0.039816143518898$	$\mu_{43} = 0.072590353622503$
$\mu_{44} = 0.034233821696022$	$\mu_{54} = 0.073302847899924$
$\mu_{55} = 0.030626774272464$	$\mu_{65} = 0.073302847899924$
$\mu_{66} = 0.029485772863308$	$\mu_{72} = 0.008896701400356$
$\mu_{76} = 0.064406146499568$	$\mu_{77} = 0.033369849008191$
$\mu_{87} = 0.073302847899924$	$\mu_{88} = 0.037227578299133$
$\mu_{98} = 0.073302847899924$	$\mu_{99} = 0.046126339053885$
$\mu_{10,6} = 0.012892211367605$	$\mu_{10,9} = 0.060410636532319$
$\mu_{10,10} = 0.053275700719583$	$\mu_{11,10} = 0.073302847899924$
$\lambda_{21} = 1$	$\lambda_{32} = 0.869472632481021$
$\lambda_{43} = 0.990280128291965$	$\lambda_{54} = 1$
$\lambda_{65} = 1$	$\lambda_{72} = 0.121369109867354$
$\lambda_{76} = 0.878630890132646$	$\lambda_{87} = 1$
$\lambda_{98} = 1$	$\lambda_{10,6} = 0.175875995775857$
$\lambda_{10,9} = 0.824124004224143$	$\lambda_{11,10} = 1$

9. Optimal 11-stage, 4th-order method:

$\mu_{11} = 0.056977945207836$	$\mu_{21} = 0.065880156369595$
$\mu_{22} = 0.043484869703481$	$\mu_{32} = 0.065880156369595$
$\mu_{33} = 0.035790792116714$	$\mu_{41} = 0.000026595081404$
$\mu_{43} = 0.061212831485396$	$\mu_{44} = 0.029306212740362$
$\mu_{54} = 0.065880156369595$	$\mu_{55} = 0.028274789742965$
$\mu_{65} = 0.065880156369595$	$\mu_{66} = 0.025442782369057$
$\mu_{76} = 0.065880156369595$	$\mu_{77} = 0.029602951078198$
$\mu_{83} = 0.009935800759662$	$\mu_{87} = 0.055944355609932$
$\mu_{88} = 0.027887296332663$	$\mu_{98} = 0.065880156369595$
$\mu_{99} = 0.033340440672342$	$\mu_{10,9} = 0.065880156369595$
$\mu_{10,10} = 0.042024506703707$	$\mu_{11,7} = 0.012021727578515$
$\mu_{11,10} = 0.053858428791080$	$\mu_{11,11} = 0.045164424313434$
$\mu_{12,11} = 0.065880156369595$	$\lambda_{21} = 1$
$\lambda_{32} = 1$	$\lambda_{41} = 0.000403688802047$
$\lambda_{43} = 0.929154313811668$	$\lambda_{54} = 1$
$\lambda_{65} = 1$	$\lambda_{76} = 1$
$\lambda_{83} = 0.150816289869158$	$\lambda_{87} = 0.849183710130842$
$\lambda_{98} = 1$	$\lambda_{10,9} = 1$
$\lambda_{11,7} = 0.182478734735714$	$\lambda_{11,10} = 0.817521265264286$
$\lambda_{12,11} = 1$	

Shu-Osher Coefficients for Fifth Order methods:

1. Optimal 4-stage, 5th-order method:

$\mu_{21} = 0.125534208080981$	$\mu_{22} = 0.125534208080983$
$\mu_{32} = 0.350653119567098$	$\mu_{33} = 0.048181647388277$
$\mu_{41} = 0.097766579224131$	$\mu_{42} = 0.000000005345013$
$\mu_{43} = 0.404181556145118$	$\mu_{44} = 0.133639210602434$
$\mu_{51} = 0.022869941925234$	$\mu_{52} = 0.138100556728488$
$\mu_{53} = 0.157510964003014$	$\mu_{54} = 0.277310825799681$
$\lambda_{21} = 0.143502249669229$	$\lambda_{32} = 0.400843023432714$
$\lambda_{41} = 0.111760167014216$	$\lambda_{42} = 0.000000006110058$
$\lambda_{43} = 0.462033126016285$	$\lambda_{51} = 0.026143376902960$
$\lambda_{52} = 0.157867252871240$	$\lambda_{53} = 0.180055922824003$
$\lambda_{54} = 0.317003054133379$	

2. Optimal 5-stage, 5th-order method:

$\mu(2,1) = 0.107733237609082$	
$\mu(2,2) = 0.107733237609079$	$\mu(3,1) = 0.000009733684024$
$\mu(3,2) = 0.205965878618791$	$\mu(3,3) = 0.041505157180052$
$\mu(4,1) = 0.010993335656900$	$\mu(4,2) = 0.000000031322743$
$\mu(4,3) = 0.245761367350216$	$\mu(4,4) = 0.079032059834967$
$\mu(5,1) = 0.040294985548405$	$\mu(5,2) = 0.011356303341111$
$\mu(5,3) = 0.024232322953809$	$\mu(5,4) = 0.220980752503271$
$\mu(5,5) = 0.098999612937858$	$\mu(6,3) = 0.079788022937926$
$\mu(6,4) = 0.023678103998428$	$\mu(6,5) = 0.194911604040485$
$\lambda(2,1) = 0.344663606249694$	$\lambda(3,1) = 0.000031140312055$
$\lambda(3,2) = 0.658932601159987$	$\lambda(4,1) = 0.035170229692428$
$\lambda(4,2) = 0.000000100208717$	$\lambda(4,3) = 0.786247596634378$
$\lambda(5,1) = 0.128913001605754$	$\lambda(5,2) = 0.036331447472278$
$\lambda(5,3) = 0.077524819660326$	$\lambda(5,4) = 0.706968664080396$
$\lambda(6,3) = 0.255260385110718$	$\lambda(6,4) = 0.075751744720289$
$\lambda(6,5) = 0.623567413728619$	

3. Optimal 6-stage, 5th-order method:

$\mu(2,1) = 0.084842972180459$	$\mu(2,2) = 0.084842972180464$
$\mu(3,2) = 0.149945333907731$	$\mu(3,3) = 0.063973483119994$
$\mu(4,3) = 0.175767531234932$	$\mu(4,4) = 0.055745328618053$
$\mu(5,1) = 0.024709139041008$	$\mu(5,4) = 0.173241563951140$
$\mu(5,5) = 0.054767418942828$	$\mu(6,2) = 0.014574431645716$
$\mu(6,3) = 0.026804592504486$	$\mu(6,5) = 0.159145416202648$
$\mu(6,6) = 0.085074359110886$	$\mu(7,3) = 0.004848530454093$
$\mu(7,4) = 0.042600565019890$	$\mu(7,6) = 0.151355691945479$
$\lambda(2,1) = 0.422021261021445$	$\lambda(3,2) = 0.745849859731775$
$\lambda(4,3) = 0.874293218071360$	$\lambda(5,1) = 0.122906844831659$
$\lambda(5,4) = 0.861728690085026$	$\lambda(6,2) = 0.072495338903420$
$\lambda(6,3) = 0.133329934574294$	$\lambda(6,5) = 0.791612404723054$
$\lambda(7,3) = 0.024117294382203$	$\lambda(7,4) = 0.211901395105308$
$\lambda(7,6) = 0.752865185365536$	

4. Optimal 7-stage, 5th-order method:

$\mu_{21} = 0.077756487471956$	$\mu_{22} = 0.077756487471823$
$\mu_{32} = 0.126469010941083$	$\mu_{33} = 0.058945597921853$
$\mu_{43} = 0.143639250502198$	$\mu_{44} = 0.044443238891736$
$\mu_{51} = 0.011999093244164$	$\mu_{54} = 0.145046006148787$
$\mu_{55} = 0.047108760907057$	$\mu_{62} = 0.011454172434127$
$\mu_{63} = 0.027138257330487$	$\mu_{65} = 0.122441492758580$
$\mu_{66} = 0.037306165750735$	$\mu_{73} = 0.020177924440034$
$\mu_{76} = 0.140855998083160$	$\mu_{77} = 0.077972159279168$
$\mu_{84} = 0.009653207936821$	$\mu_{85} = 0.025430639631870$
$\mu_{86} = 0.000177781270869$	$\mu_{87} = 0.124996366168017$
$\lambda_{21} = 0.482857811904546$	$\lambda_{32} = 0.785356333370487$
$\lambda_{43} = 0.891981318293413$	$\lambda_{51} = 0.074512829695468$
$\lambda_{54} = 0.900717090387559$	$\lambda_{62} = 0.071128941372444$
$\lambda_{63} = 0.168525096484428$	$\lambda_{65} = 0.760345962143127$
$\lambda_{73} = 0.125302322168346$	$\lambda_{76} = 0.874697677831654$
$\lambda_{84} = 0.059945182887979$	$\lambda_{85} = 0.157921009644458$
$\lambda_{86} = 0.001103998884730$	$\lambda_{87} = 0.776211398253764$

5. Optimal 8-stage, 5th-order method:

$\mu_{21} = 0.068228425119547$	$\mu_{22} = 0.068228425081188$
$\mu_{32} = 0.105785458668142$	$\mu_{33} = 0.049168429086829$
$\mu_{43} = 0.119135238085849$	$\mu_{44} = 0.040919294063196$
$\mu_{51} = 0.009164078944895$	$\mu_{54} = 0.120257079939301$
$\mu_{55} = 0.039406904101415$	$\mu_{62} = 0.007428674198294$
$\mu_{63} = 0.019703233696280$	$\mu_{65} = 0.105180973170163$
$\mu_{66} = 0.045239659320409$	$\mu_{73} = 0.015335646668415$
$\mu_{76} = 0.116977452926909$	$\mu_{77} = 0.050447703819928$
$\mu_{84} = 0.011255581082016$	$\mu_{85} = 0.006541409424671$
$\mu_{87} = 0.114515518273119$	$\mu_{88} = 0.060382824328534$
$\mu_{95} = 0.002607774587593$	$\mu_{96} = 0.024666705635997$
$\mu_{98} = 0.104666894951906$	$\lambda_{21} = 0.515658560550227$
$\lambda_{32} = 0.799508082567950$	$\lambda_{43} = 0.900403391614526$
$\lambda_{51} = 0.069260513476804$	$\lambda_{54} = 0.908882077064212$
$\lambda_{62} = 0.056144626483417$	$\lambda_{63} = 0.148913610539984$
$\lambda_{65} = 0.794939486396848$	$\lambda_{73} = 0.115904148048060$
$\lambda_{76} = 0.884095226988328$	$\lambda_{84} = 0.085067722561958$
$\lambda_{85} = 0.049438833770315$	$\lambda_{87} = 0.865488353423280$
$\lambda_{95} = 0.019709106398420$	$\lambda_{96} = 0.186426667470161$
$\lambda_{98} = 0.791054172708715$	

6. Optimal 9-stage, 5th-order method:

$$\begin{aligned}
\mu_{21} &= 0.057541273792734 & \mu_{22} &= 0.057541282875429\\
\mu_{32} &= 0.089687860942851 & \mu_{33} &= 0.041684970395150\\
\mu_{43} &= 0.101622955619526 & \mu_{44} &= 0.040743690263377\\
\mu_{51} &= 0.009276188714858 & \mu_{54} &= 0.101958242208571\\
\mu_{55} &= 0.040815264589441 & \mu_{62} &= 0.011272987717036\\
\mu_{65} &= 0.101125244372555 & \mu_{66} &= 0.040395338505384\\
\mu_{73} &= 0.003606182878823 & \mu_{74} &= 0.018205434656765\\
\mu_{76} &= 0.090586614534056 & \mu_{77} &= 0.042925976445877\\
\mu_{84} &= 0.011070977346914 & \mu_{87} &= 0.101327254746568\\
\mu_{88} &= 0.046669302312152 & \mu_{95} &= 0.010281040119047\\
\mu_{98} &= 0.102117191974435 & \mu_{99} &= 0.050500143250113\\
\mu_{10,6} &= 0.000157554758807 & \mu_{10,7} &= 0.023607648002010\\
\mu_{10,9} &= 0.088454624345414 & \lambda_{21} &= 0.511941093031398\\
\lambda_{32} &= 0.797947256574797 & \lambda_{43} &= 0.904133043080300\\
\lambda_{51} &= 0.082529667434119 & \lambda_{54} &= 0.907116066770269\\
\lambda_{62} &= 0.100295062538531 & \lambda_{65} &= 0.899704937426848\\
\lambda_{73} &= 0.032083982209117 & \lambda_{74} &= 0.161972606843345\\
\lambda_{76} &= 0.805943410735452 & \lambda_{84} &= 0.098497788983963\\
\lambda_{87} &= 0.901502211016037 & \lambda_{95} &= 0.091469767162319\\
\lambda_{98} &= 0.908530232837680 & \lambda_{10,6} &= 0.001401754777391\\
\lambda_{10,7} &= 0.210035759124536 & \lambda_{10,9} &= 0.786975228149903
\end{aligned}$$

7. Optimal 10-stage, 5th-order method:

$$\begin{aligned}
\mu_{21} &= 0.052445615058994 & \mu_{22} &= 0.052445635165954 \\
\mu_{32} &= 0.079936220395519 & \mu_{33} &= 0.038724845476313 \\
\mu_{43} &= 0.089893189589075 & \mu_{44} &= 0.037676214671832 \\
\mu_{51} &= 0.007606429497294 & \mu_{54} &= 0.090180506502554 \\
\mu_{55} &= 0.035536573874530 & \mu_{62} &= 0.009295158915663 \\
\mu_{65} &= 0.089447242753894 & \mu_{66} &= 0.036490114423762 \\
\mu_{73} &= 0.003271387942850 & \mu_{74} &= 0.015255382390056 \\
\mu_{76} &= 0.080215515252923 & \mu_{77} &= 0.035768398609662 \\
\mu_{84} &= 0.009638972523544 & \mu_{87} &= 0.089103469454345 \\
\mu_{88} &= 0.040785658461768 & \mu_{95} &= 0.009201462517982 \\
\mu_{98} &= 0.089540979697808 & \mu_{99} &= 0.042414168555682 \\
\mu_{10,6} &= 0.005634796609556 & \mu_{10,7} &= 0.006560464576444 \\
\mu_{10,9} &= 0.086547180546464 & \mu_{10,10} &= 0.043749770437420 \\
\mu_{11,7} &= 0.001872759401284 & \mu_{11,8} &= 0.017616881402665 \\
\mu_{11,10} &= 0.079160150775900 & \lambda_{21} &= 0.531135486241871 \\
\lambda_{32} &= 0.809542670828687 & \lambda_{43} &= 0.910380456183399 \\
\lambda_{51} &= 0.077033029836054 & \lambda_{54} &= 0.913290217244921 \\
\lambda_{62} &= 0.094135396158718 & \lambda_{65} &= 0.905864193215084 \\
\lambda_{73} &= 0.033130514796271 & \lambda_{74} &= 0.154496709294644 \\
\lambda_{76} &= 0.812371189661489 & \lambda_{84} &= 0.097617319434729 \\
\lambda_{87} &= 0.902382678155958 & \lambda_{95} &= 0.093186499255038 \\
\lambda_{98} &= 0.906813500744962 & \lambda_{10,6} &= 0.057065598977612 \\
\lambda_{10,7} &= 0.066440169285130 & \lambda_{10,9} &= 0.876494226842443 \\
\lambda_{11,7} &= 0.018966103726616 & \lambda_{11,8} &= 0.178412453726484 \\
\lambda_{11,10} &= 0.801683136446066 & &
\end{aligned}$$

8. Optimal 11-stage, 5th-order method:

$\mu_{21} = 0.048856948431570$	$\mu_{22} = 0.048856861697775$
$\mu_{32} = 0.072383163641108$	$\mu_{33} = 0.035920513887793$
$\mu_{43} = 0.080721632683704$	$\mu_{44} = 0.034009594943671$
$\mu_{51} = 0.006438090160799$	$\mu_{54} = 0.081035022899306$
$\mu_{55} = 0.032672027896742$	$\mu_{62} = 0.007591099341932$
$\mu_{63} = 0.000719846382100$	$\mu_{65} = 0.079926841108108$
$\mu_{66} = 0.033437798720082$	$\mu_{73} = 0.003028997848550$
$\mu_{74} = 0.012192534706212$	$\mu_{76} = 0.073016254277378$
$\mu_{77} = 0.033377699686911$	$\mu_{84} = 0.008251011235053$
$\mu_{87} = 0.079986775597087$	$\mu_{88} = 0.035640440183022$
$\mu_{95} = 0.008095394925904$	$\mu_{98} = 0.080142391870059$
$\mu_{99} = 0.036372965664654$	$\mu_{10,6} = 0.005907318148947$
$\mu_{10,7} = 0.005394911565057$	$\mu_{10,9} = 0.076935557118137$
$\mu_{10,10} = 0.032282094274356$	$\mu_{11,7} = 0.003571080721480$
$\mu_{11,8} = 0.008920593887617$	$\mu_{11,10} = 0.075746112223043$
$\mu_{11,11} = 0.042478561828713$	$\mu_{12,8} = 0.004170617993886$
$\mu_{12,9} = 0.011637432775226$	$\mu_{12,11} = 0.072377330912325$
$\lambda_{21} = 0.553696439876870$	$\lambda_{32} = 0.820319346617409$
$\lambda_{43} = 0.914819326070196$	$\lambda_{51} = 0.072962960562995$
$\lambda_{54} = 0.918370981510030$	$\lambda_{62} = 0.086030028794504$
$\lambda_{63} = 0.008158028526592$	$\lambda_{65} = 0.905811942678904$
$\lambda_{73} = 0.034327672500586$	$\lambda_{74} = 0.138178156365216$
$\lambda_{76} = 0.827494171134198$	$\lambda_{84} = 0.093508818968334$
$\lambda_{87} = 0.906491181031666$	$\lambda_{95} = 0.091745217287743$
$\lambda_{98} = 0.908254782302260$	$\lambda_{10,6} = 0.066947714363965$
$\lambda_{10,7} = 0.061140603801867$	$\lambda_{10,9} = 0.871911681834169$
$\lambda_{11,7} = 0.040471104837131$	$\lambda_{11,8} = 0.101097207986272$
$\lambda_{11,10} = 0.858431687176596$	$\lambda_{12,8} = 0.047265668639449$
$\lambda_{12,9} = 0.131887178872293$	$\lambda_{12,11} = 0.820253244225314$

Shu-Osher Coefficients for sixth Order methods:

1. The optimal sixth-order, six-stage method ($c = 0.18$):

$$\begin{array}{|ll}
\mu_{21} = 0.306709397198437 & \mu_{22} = 0.306709397198281 \\
\mu_{31} = 0.100402778173265 & \mu_{32} = 0.000000014622272 \\
\mu_{33} = 0.100402700098726 & \mu_{41} = 0.000015431349319 \\
\mu_{42} = 0.000708584139276 & \mu_{43} = 0.383195003696784 \\
\mu_{44} = 0.028228318307509 & \mu_{51} = 0.101933808745384 \\
\mu_{52} = 0.000026687930165 & \mu_{53} = 0.136711477475771 \\
\mu_{54} = 0.331296656179688 & \mu_{55} = 0.107322255666019 \\
\mu_{61} = 0.000033015066992 & \mu_{62} = 0.000000017576816 \\
\mu_{63} = 0.395057247524893 & \mu_{64} = 0.014536993458566 \\
\mu_{65} = 0.421912313467517 & \mu_{66} = 0.049194928995335 \\
\mu_{71} = 0.054129307323559 & \mu_{72} = 0.002083586568620 \\
\mu_{73} = 0.233976271277479 & \mu_{74} = 0.184897163424393 \\
\mu_{75} = 0.303060566272042 & \mu_{76} = 0.135975816243004 \\
\lambda_{21} = 0.055928810359256 & \lambda_{31} = 0.018308561756789 \\
\lambda_{32} = 0.000000002666388 & \lambda_{41} = 0.000002813924247 \\
\lambda_{42} = 0.000129211130507 & \lambda_{43} = 0.069876048429340 \\
\lambda_{51} = 0.018587746937629 & \lambda_{52} = 0.000004866574675 \\
\lambda_{53} = 0.024929494718837 & \lambda_{54} = 0.060412325234826 \\
\lambda_{61} = 0.000006020335333 & \lambda_{62} = 0.000000003205153 \\
\lambda_{63} = 0.072039142196788 & \lambda_{64} = 0.002650837430364 \\
\lambda_{65} = 0.076936194272824 & \lambda_{71} = 0.009870541274021 \\
\lambda_{72} = 0.000379944400556 & \lambda_{73} = 0.042665841426363 \\
\lambda_{74} = 0.033716209818106 & \lambda_{75} = 0.055263441854804 \\
\lambda_{76} = 0.024795346049276 &
\end{array}$$

2. The optimal sixth-order, seven-stage method ($c = 0.26$):

$\mu_{21} = 0.090485932570398$	
$\mu_{22} = 0.090485932570397$	
$\mu_{32} = 0.346199513509666$	
$\mu_{33} = 0.056955495796615$	$\lambda_{21} = 0.023787133610744$
$\mu_{41} = 0.089183260058590$	$\lambda_{32} = 0.091009661390427$
$\mu_{42} = 0.122181527536711$	$\lambda_{41} = 0.023444684301672$
$\mu_{43} = 0.340520235772773$	$\lambda_{42} = 0.032119338749362$
$\mu_{44} = 0.086699362107543$	$\lambda_{43} = 0.089516680829776$
$\mu_{51} = 0.214371998459638$	$\lambda_{51} = 0.056354565012571$
$\mu_{52} = 0.046209156887254$	$\lambda_{52} = 0.012147561037311$
$\mu_{53} = 0.215162143673919$	$\lambda_{53} = 0.056562280060094$
$\mu_{54} = 0.000000362542364$	$\lambda_{54} = 0.000000095305905$
$\mu_{55} = 0.209813410800754$	$\lambda_{61} = 0.000000155574348$
$\mu_{61} = 0.000000591802702$	$\lambda_{62} = 0.102670355321862$
$\mu_{62} = 0.390556634551239$	$\lambda_{63} = 0.000000129323288$
$\mu_{63} = 0.000000491944026$	$\lambda_{64} = 0.086906235023916$
$\mu_{64} = 0.330590135449081$	$\lambda_{65} = 0.001948095974350$
$\mu_{65} = 0.007410530577593$	$\lambda_{71} = 0.000000005742021$
$\mu_{66} = 0.070407008959133$	$\lambda_{72} = 0.085547570527144$
$\mu_{71} = 0.000000021842570$	$\lambda_{73} = 0.018145676643359$
$\mu_{72} = 0.325421794191472$	$\lambda_{74} = 0.098149750494075$
$\mu_{73} = 0.069025907032937$	$\lambda_{75} = 0.001982854233713$
$\mu_{74} = 0.373360315300742$	$\lambda_{76} = 0.001436838619770$
$\mu_{75} = 0.007542750523234$	$\lambda_{81} = 0.011609230551384$
$\mu_{76} = 0.005465714557738$	$\lambda_{82} = 0.053848246287940$
$\mu_{77} = 0.063240270982556$	$\lambda_{83} = 0.050281417794762$
$\mu_{81} = 0.044161355044152$	$\lambda_{84} = 0.067254353278777$
$\mu_{82} = 0.204837996136028$	$\lambda_{85} = 0.004201954631994$
$\mu_{83} = 0.191269829083813$	$\lambda_{86} = 0.004238754905099$
$\mu_{84} = 0.255834644704751$	$\lambda_{87} = 0.039733519691061$
$\mu_{85} = 0.015984178241749$	
$\mu_{86} = 0.016124165979879$	
$\mu_{87} = 0.151145768228502$	

3. The optimal sixth-order, eight-stage method ($c = 2.25$):

$\mu_{21} = 0.078064586430339$
$\mu_{22} = 0.078064586430334$
$\mu_{31} = 0.000000000128683$
$\mu_{32} = 0.207887720440412$
$\mu_{33} = 0.051491724905522$
$\mu_{41} = 0.039407945831803$
$\mu_{43} = 0.256652317630585$
$\mu_{44} = 0.062490509654886$
$\mu_{51} = 0.009678931461971$
$\mu_{52} = 0.113739188386853$
$\mu_{54} = 0.227795405648863$
$\mu_{55} = 0.076375614721986$
$\mu_{62} = 0.010220279377975$
$\mu_{63} = 0.135083590682973$
$\mu_{65} = 0.235156310567507$
$\mu_{66} = 0.033370798931382$
$\mu_{72} = 0.000000009428737$
$\mu_{73} = 0.112827524882246$
$\mu_{74} = 0.001997541632150$
$\mu_{75} = 0.177750742549303$
$\mu_{76} = 0.099344022703332$
$\mu_{77} = 0.025183595544641$
$\mu_{81} = 0.122181071065616$
$\mu_{82} = 0.000859535946343$
$\mu_{83} = 0.008253954430873$
$\mu_{84} = 0.230190271515289$
$\mu_{85} = 0.046429529676480$
$\mu_{86} = 0.017457063072040$
$\mu_{87} = 0.017932893410781$
$\mu_{88} = 0.322331010725841$
$\mu_{91} = 0.011069087473717$
$\mu_{92} = 0.010971589676607$
$\mu_{93} = 0.068827453812950$
$\mu_{94} = 0.048864283062331$
$\mu_{95} = 0.137398274895655$
$\mu_{96} = 0.090347431612516$
$\mu_{97} = 0.029504401738350$
$\mu_{98} = 0.000167109498102$

$\lambda_{21} = 0.175964293749273$
$\lambda_{31} = 0.000000000290062$
$\lambda_{32} = 0.468596806556916$
$\lambda_{41} = 0.088828900190110$
$\lambda_{43} = 0.578516403866171$
$\lambda_{51} = 0.021817144198582$
$\lambda_{52} = 0.256377915663045$
$\lambda_{54} = 0.513470441684846$
$\lambda_{62} = 0.023037388973687$
$\lambda_{63} = 0.304490034708070$
$\lambda_{65} = 0.530062554633790$
$\lambda_{72} = 0.000000021253185$
$\lambda_{73} = 0.254322947692795$
$\lambda_{74} = 0.004502630688369$
$\lambda_{75} = 0.400665465691124$
$\lambda_{76} = 0.223929973789109$
$\lambda_{81} = 0.275406645480353$
$\lambda_{82} = 0.001937467969363$
$\lambda_{83} = 0.018605123379003$
$\lambda_{84} = 0.518868675379274$
$\lambda_{85} = 0.104656154246370$
$\lambda_{86} = 0.039349722004217$
$\lambda_{87} = 0.040422284523661$
$\lambda_{91} = 0.024950675444873$
$\lambda_{92} = 0.024730907022402$
$\lambda_{93} = 0.155143002154553$
$\lambda_{94} = 0.110144297841125$
$\lambda_{95} = 0.309707532056893$
$\lambda_{96} = 0.203650883489192$
$\lambda_{97} = 0.066505459796630$
$\lambda_{98} = 0.000376679185235$

4. The optimal sixth-order, nine-stage method ($c = 5.80$):

$\mu_{21} = 0.060383920365295$
$\mu_{22} = 0.060383920365140$
$\mu_{31} = 0.000000016362287$
$\mu_{32} = 0.119393671070984$
$\mu_{33} = 0.047601859039825$
$\mu_{42} = 0.000000124502898$
$\mu_{43} = 0.144150297305350$
$\mu_{44} = 0.016490678866732$
$\mu_{51} = 0.014942049029658$
$\mu_{52} = 0.033143125204828$
$\mu_{53} = 0.020040368468312$
$\mu_{54} = 0.095855615754989$
$\mu_{55} = 0.053193337903908$
$\mu_{61} = 0.000006536159050$
$\mu_{62} = 0.000805531139166$
$\mu_{63} = 0.015191136635430$
$\mu_{64} = 0.054834245267704$
$\mu_{65} = 0.089706774214904$
$\mu_{71} = 0.000006097150226$
$\mu_{72} = 0.018675155382709$
$\mu_{73} = 0.025989306353490$
$\mu_{74} = 0.000224116890218$
$\mu_{75} = 0.000125522781582$
$\mu_{76} = 0.125570620920810$
$\mu_{77} = 0.019840674620006$
$\mu_{81} = 0.000000149127775$
$\mu_{82} = 0.000000015972341$
$\mu_{83} = 0.034242827620807$
$\mu_{84} = 0.017165973521939$
$\mu_{85} = 0.000000000381532$
$\mu_{86} = 0.001237807078917$
$\mu_{87} = 0.119875131948576$
$\mu_{88} = 0.056749019092783$
$\mu_{91} = 0.000000072610411$
$\mu_{92} = 0.000000387168511$
$\mu_{93} = 0.000400376164405$
$\mu_{94} = 0.000109472445726$
$\mu_{95} = 0.012817181286633$
$\mu_{96} = 0.011531979169562$
$\mu_{97} = 0.000028859233948$
$\mu_{98} = 0.143963789161172$
$\mu_{99} = 0.060174596046625$
$\mu_{10,1} = 0.001577092080021$
$\mu_{10,2} = 0.000008909587678$
$\mu_{10,3} = 0.000003226074427$
$\mu_{10,4} = 0.000000062166910$
$\mu_{10,5} = 0.009112668630420$
$\mu_{10,6} = 0.008694079174358$

$\lambda_{21} = 0.350007201986739$
$\lambda_{31} = 0.000000094841777$
$\lambda_{32} = 0.692049215977999$
$\lambda_{42} = 0.000000721664155$
$\lambda_{43} = 0.835547641163090$
$\lambda_{51} = 0.086609559981880$
$\lambda_{52} = 0.192109628653810$
$\lambda_{53} = 0.116161276908552$
$\lambda_{54} = 0.555614071795216$
$\lambda_{61} = 0.000037885959162$
$\lambda_{62} = 0.004669151960107$
$\lambda_{63} = 0.088053362494510$
$\lambda_{64} = 0.317839263219390$
$\lambda_{65} = 0.519973146034093$
$\lambda_{71} = 0.000035341304071$
$\lambda_{72} = 0.108248004479122$
$\lambda_{73} = 0.150643488255346$
$\lambda_{74} = 0.001299063147749$
$\lambda_{75} = 0.000727575773504$
$\lambda_{76} = 0.727853067743022$
$\lambda_{81} = 0.000000864398917$
$\lambda_{82} = 0.000000092581509$
$\lambda_{83} = 0.198483904509141$
$\lambda_{84} = 0.099500236576982$
$\lambda_{85} = 0.000000002211499$
$\lambda_{86} = 0.007174780797111$
$\lambda_{87} = 0.694839938634174$
$\lambda_{91} = 0.000000420876394$
$\lambda_{92} = 0.000002244169749$
$\lambda_{93} = 0.002320726117116$
$\lambda_{94} = 0.000634542179300$
$\lambda_{95} = 0.074293052394615$
$\lambda_{96} = 0.066843552689032$
$\lambda_{97} = 0.000167278634186$
$\lambda_{98} = 0.834466572009306$
$\lambda_{10,1} = 0.009141400274516$
$\lambda_{10,2} = 0.000051643216195$
$\lambda_{10,3} = 0.000018699502726$
$\lambda_{10,4} = 0.000000360342058$
$\lambda_{10,5} = 0.052820347381733$
$\lambda_{10,6} = 0.050394050390558$
$\lambda_{10,7} = 0.103597678603687$
$\lambda_{10,8} = 0.159007699664781$
$\lambda_{10,9} = 0.624187175011814$

5. The optimal sixth-order, ten-stage method ($c = 8.10$):

$$
\begin{aligned}
\mu_{21} &= 0.054638144097621\\
\mu_{22} &= 0.054638144097609\\
\mu_{32} &= 0.094708145223810\\
\mu_{33} &= 0.044846931722606\\
\mu_{43} &= 0.108958403164940\\
\mu_{44} &= 0.031071352647397\\
\mu_{51} &= 0.004498251069701\\
\mu_{52} &= 0.005530448043688\\
\mu_{54} &= 0.107851443619437\\
\mu_{55} &= 0.018486380725450\\
\mu_{62} &= 0.015328210231111\\
\mu_{63} &= 0.014873940010974\\
\mu_{64} &= 0.000000013999299\\
\mu_{65} &= 0.093285690103096\\
\mu_{66} &= 0.031019852663844\\
\mu_{73} &= 0.023345108682580\\
\mu_{74} &= 0.000000462051194\\
\mu_{76} &= 0.100142283610706\\
\mu_{77} &= 0.037191650574052\\
\mu_{84} &= 0.020931607249912\\
\mu_{85} &= 0.007491225374492\\
\mu_{86} &= 0.000000004705702\\
\mu_{87} &= 0.094887152674486\\
\mu_{88} &= 0.041052752299292\\
\mu_{94} &= 0.000000000437894\\
\mu_{95} &= 0.013484714992727\\
\mu_{96} &= 0.012301077330264\\
\mu_{98} &= 0.097178530400423\\
\mu_{99} &= 0.039273658398104\\
\mu_{10,1} &= 0.000987065715240\\
\mu_{10,2} &= 0.000000347467847\\
\mu_{10,6} &= 0.004337021151393\\
\mu_{10,7} &= 0.011460261685365\\
\mu_{10,8} &= 0.002121689510807\\
\mu_{10,9} &= 0.104338127248348\\
\mu_{10,10} &= 0.042268075457472\\
\mu_{11,3} &= 0.000656941338471\\
\mu_{11,7} &= 0.015039465910057\\
\mu_{11,8} &= 0.004816543620956\\
\mu_{11,9} &= 0.031302441038151\\
\mu_{11,10} &= 0.071672462436845
\end{aligned}
$$

$$
\begin{aligned}
\lambda_{21} &= 0.442457635916190\\
\lambda_{32} &= 0.766942997969774\\
\lambda_{43} &= 0.882341050812911\\
\lambda_{51} &= 0.036426667979449\\
\lambda_{52} &= 0.044785360253007\\
\lambda_{54} &= 0.873376934047102\\
\lambda_{62} &= 0.124127269944714\\
\lambda_{63} &= 0.120448606787528\\
\lambda_{64} &= 0.000000113365798\\
\lambda_{65} &= 0.755424009901960\\
\lambda_{73} &= 0.189047812082446\\
\lambda_{74} &= 0.000003741673193\\
\lambda_{76} &= 0.810948446244362\\
\lambda_{84} &= 0.169503368254511\\
\lambda_{85} &= 0.060663661331375\\
\lambda_{86} &= 0.000000038106595\\
\lambda_{87} &= 0.768392593572726\\
\lambda_{94} &= 0.000000003546047\\
\lambda_{95} &= 0.109198714839684\\
\lambda_{96} &= 0.099613661566658\\
\lambda_{98} &= 0.786948084216732\\
\lambda_{10,1} &= 0.007993221037648\\
\lambda_{10,2} &= 0.000002813781560\\
\lambda_{10,6} &= 0.035121034164983\\
\lambda_{10,7} &= 0.092804768098049\\
\lambda_{10,8} &= 0.017181361859997\\
\lambda_{10,9} &= 0.844926230212794\\
\lambda_{11,3} &= 0.005319886250823\\
\lambda_{11,7} &= 0.121789029292733\\
\lambda_{11,8} &= 0.039004189088262\\
\lambda_{11,9} &= 0.253485990215933\\
\lambda_{11,10} &= 0.580400905152248
\end{aligned}
$$

6 Appendix 2

This appendix contains MATLAB scripts which we created as part of this project.

- *Evaluate the Radius of absolute monotonicity from the Butcher array:*

```
  function r = am_radius(A,b)
%By David Ketcheson
%Evaluates the Radius of absolute monotonicity
%of a Runge-Kutta method, given the Butcher array.
%
%For an m-stage method, A should be an m x m matrix
%and b should be a column vector of length m.
%
%Accuracy can be changed by modifying the value of eps.
%Methods with very large radii of a.m. (>50) will require
%rmax to be increased.

rmax=50; eps=1.e-12;

m=length(b); e=ones(m,1);
K=[A;b'];
rlo=0; rhi=rmax;

while rhi-rlo>eps  %use bisection
  r=0.5*(rhi+rlo);
  X=eye(m)+r*A; beta=K/X; ech=r*K*(X\e);
  if (min(beta(:))<-3.e-16 || max(ech(:))>1.+3.e-16)
    rhi=r;
  else
    rlo=r;
  end
end

if rhi==rmax % r>=rmax
  error('Error: increase value of rmax in am_radius.m');
else
  r=rlo;
end
```

- *Butcher array from the Shu Osher array:*

```
function [A,b,c]=shuosher2butcher(alpha,beta);
```

```
%function [A,b,c]=shuosher2butcher(lambda,mu);
%
%By David Ketcheson
%
%Generate Butcher form of a Runge-Kutta method,
%given its Shu-Osher or modified Shu-Osher form
%
%For an m-stage method, alpha and beta (or lambda and mu) should be
%matrices of dimension (m+1) x m
%
%Note that MATLAB indexes from 1, while the Shu-Osher coefficients
%are usually indexed from zero.

s=size(alpha,2);
X=eye(s)-alpha(1:end-1,:);
A=X\beta(1:end-1,:);
b=beta(end,:)+alpha(end,:)*A; b=b';
c=sum(A,2);
```

- *Butcher to modified Shu-Osher:* This function generates the modified Shu-Osher form of a Runge-Kutta method, given its Butcher form and radius of absolute monotonicity

```
function [lambda,mu1]=butcher2modshuosher(A,b,r);
%By David Ketcheson
%Generate modified Shu-Osher form of a Runge-Kutta method,
%given its Butcher form and radius of absolute monotonicity
%
%For an m-stage method, A should be an m x m matrix
%and b should be a column vector of length m.
%
%Note that MATLAB indexes from 1, while the Shu-Osher coefficients
%are usually indexed from zero.
Aup=triu(A);
if max(abs(Aup))>0 mclass='implicit'; else mclass='explicit'; end
if nargin<3 r = am_radius(A,b); end

s=size(A,1);
K=[A;b'];
G=eye(s)+r*A;
mu1=K/G;
lambda=r*mu1;
for i=1:s+1
  if strcmp(mclass,'implicit') %0 stage is u_n
   alpha(i)=1-sum(lambda(i,1:s));
  end
end

%Eliminate diagonal terms of lambda array (for implicit methods)
lambda=lambda; mu1=mu1;
for i=1:s
  if lambda(i,i)~=1
    fac=1/(1-lambda(i,i));
    mu1(i,:)=fac*mu1(i,:);
    lambda(i,:)=fac*lambda(i,:);
    lambda(i,i)=0.;
  end
end
```

- *Butcher to Shu-Osher:*

```
  function [alpha,beta]=butcher2shuosher(A,b,r);
%By David Ketcheson
%
%Generate Shu-Osher form of an explicit Runge-Kutta method,
%given its Butcher form and radius of absolute monotonicity
%
%For an m-stage method, A should be an m x m matrix
%and b should be a column vector of length m.
%
%Note that MATLAB indexes from 1, while the Shu-Osher coefficients
%are usually indexed from zero.

if nargin<3 r = am_radius(A,b); end

s=size(A,1);
K=[A;b'];
G=eye(s)+r*A;
beta=K/G;
alpha=r*beta;
for i=2:s+1
  alpha(i,1)=1-sum(alpha(i,2:s));
end
```

- *Radius of circle contractivity of a Runge-Kutta method, given the Butcher array:*

```
function [r] = cc_radius(A,b)
%By David Ketcheson
%
```

```
%Evaluates the Radius of circle contractivity
%of a Runge-Kutta method, given the Butcher array
%
%For an m-stage method, A should be an m x m matrix
%and b should be a column vector of length m.
%
%Accuracy can be changed by modifying the value of eps.
%Methods with very large radii of a.m. (>1000) will require
%rmax to be increased.

rmax=1000; eps=1.e-13;

if min(b)<=0
  r=0;
else
  m=length(b);
  B=diag(b);
  M=B*A+A'*B-b*b';
  rlo=0; rhi=rmax;
  while rhi-rlo>eps
    r=0.5*(rhi+rlo);
    X=M+B/r;
    if min(eig(X))<-3.e-16
      rhi=r;
    else
      rlo=r;
    end
  end
end
if rhi==rmax % r>=rmax
  error('Error: increase value of rmax in cc_radius.m');
else
  r=rlo;
end
```

- *Make the Butcher arrays for different methods:*

```
function [A,b,c,r]=makebutcher(name,s)
%By David Ketcheson
%
%Set up Butcher arrays A,b,c for various methods
%Also returns SSP coefficient r
%For families of methods, optional input s is the number of stages

if nargin<2 s=1; end
```

```
switch name
%================SSP Methods=======================

  %================Explicit Methods=======================
  case 'FE11'
    %Forward Euler
    s=1; r=1;
    A=[0];
    b=[1]'; c=[0]';

  case 'SSP22'
    s=2; r=1;
    A=[0 0; 1 0];
    b=[1/2 1/2]'; c=sum(A,2);

  case 'SSP42'
    s=4; r=3;
    A=[0 0 0 0; 1/3 0 0 0; 1/3 1/3 0 0; 1/3 1/3 1/3 0];
    b=1/4*ones(m,1); c=sum(A,2);

  case 'SSP33'
    s=3; r=1;
    A=[0 0 0; 1 0 0; 1/4 1/4 0];
    b=[1/6 1/6 2/3]'; c=sum(A,2);

  case 'SSP43'
    s=4; r=2;
    A=[0 0 0 0; 1/2 0 0 0; 1/2 1/2 0 0; 1/6 1/6 1/6 0];
    b=[1/6 1/6 1/6 1/2]'; c=sum(A,2);

  case 'SSP104'
    s=10; r=6;
    alpha0=diag(ones(1,s-1),-1);
    alpha0(6,5)=2/5; alpha0(6,1)=3/5;
    beta0 =1/6*diag(ones(1,s-1),-1);
    beta0(6,5)=1/15;
    A=(eye(s)-alpha0)\beta0;
    b=1/10*ones(s,1); c=sum(A,2);

  case 'rSSPs2'
    %Rational (optimal, low-storage) s-stage 2nd order SSP
    if s<2 error('Explicit second order SSP family requires s>=2'); end
    r=s-1;
    alpha=[zeros(1,s);eye(s);];
  alpha(s+1,s)=(s-1)/s;
  beta=alpha/r;
  alpha(s+1,1)=1/s;
  A=(eye(s)-alpha(1:s,:))\beta(1:s,:);
  b=beta(s+1,:)+alpha(s+1,:)*A; b=b';
  c=sum(A,2);

case 'rSSPs3'
  %Rational (optimal, low-storage) s^2-stage 3rd order SSP
  if round(sqrt(s))~=sqrt(s) || s<4
     error('Explicit third order SSP family requires s=n^2, n>1');
  end
  n=s^2; r=n-s;
  alpha=[zeros(1,n);eye(n);];
  alpha(s*(s+1)/2+1,s*(s+1)/2)=(s-1)/(2*s-1);
  beta=alpha/r;
  alpha(s*(s+1)/2+1,(s-1)*(s-2)/2+1)=s/(2*s-1);
  A=(eye(n)-alpha(1:n,:))\beta(1:n,:);
  b=beta(n+1,:)+alpha(n+1,:)*A; b=b';
  c=sum(A,2);


%=================Implicit Methods========================
case 'BE11'
  %Backward Euler
  s=1; r=1.e10;
  A=[1];
  b=[1]'; c=[1]';

case 'SDIRK34' %3-stage, 4th order singly diagonally implicit (SSP)
  s=3; r=1.7588;
  g=0.5*(1-cos(pi/18)/sqrt(3)-sin(pi/18));
  q=(0.5-g)^2;
  A=[g     0    0
     0.5-g g    0
     2*g  1-4*g g];
  b=[1/(24*q) 1-1/(12*q) 1/(24*q)]';
  c=sum(A,2);

 case 'ISSPm2'
  %Optimal DIRK SSP schemes of order 2
  r=2*s;
  i=repmat((1:s)',1,s); j=repmat(1:s,s,1);
  A=1/s*(j<i) + 1/(2*s)*(i==j);
  b=1/s*ones(s,1);
```

```
    c=sum(A,2);

  case 'ISSPs3'
    %Optimal DIRK SSP schemes of order 3
    if s<2 error('Implicit third order SSP schemes require s>=2'); end
    r=s-1+sqrt(s^2-1);
    i=repmat((1:s)',1,s); j=repmat(1:s,s,1);
    A=1/sqrt(s^2-1)*(j<i) + 0.5*(1-sqrt((s-1)/(s+1)))*(i==j);
    b=1/s*ones(s,1);
    c=sum(A,2);


%================Classical Methods========================

  %Gauss-Legendre methods -- order 2s
  case 'GL1'
    r=2; A=1/2; b=1; c=1/2;
  case 'GL2'
    r=0;
    A=[1/4 1/4-sqrt(3)/6
       1/4+sqrt(3)/6 1/4];
    b=[1/2 1/2]';
    c=[1/2-sqrt(3)/6 1/2+sqrt(3)/6]';
  case 'GL3'
    r=0;
    A=[5/36 (80-24*sqrt(15))/360 (50-12*sqrt(15))/360
       (50+15*sqrt(15))/360 2/9  (50-15*sqrt(15))/360
       (50+12*sqrt(15))/360 (80+24*sqrt(15))/360 5/36];
    b=[5/18 4/9 5/18]';
    c=[(5-sqrt(15))/10 1/2 (5+sqrt(15))/10]';

  %Radau IA methods -- order 2s-1
  case 'RIA1'
    r=1;
    A=1; b=1; c=0;
  case 'RIA2'
    r=0;
    A=[1/4 -1/4
       1/4 5/12];
    b=[1/4 3/4]';
    c=[0 2/3]';
  case 'RIA3'
    r=0;
    A=[1/9 (-1-sqrt(6))/18 (-1+sqrt(6))/18
       1/9 (88+7*sqrt(6))/360 (88-43*sqrt(6))/360
```

```
      1/9 (88+43*sqrt(6))/360 (88-7*sqrt(6))/360];
    b=[1/9 (16+sqrt(6))/36 (16-sqrt(6))/36]';
    c=[0 (6-sqrt(6))/10 (6+sqrt(6))/10]';

  %Radau IIA methods -- order 2s-1
  case 'RIIA1'
    r=1;
    A=1; b=1; c=1;
  case 'RIIA2'
    r=0;
    A=[5/12 -1/12
      3/4 1/4];
    b=[3/4 1/4]';
    c=[1/3 1]';
  case 'RIIA3'
    r=0;
    A=[(88-7*sqrt(6))/360 (296-169*sqrt(6))/1800 (-2+3*sqrt(6))/225
      (296+169*sqrt(6))/1800 (88+7*sqrt(6))/360 (-2-3*sqrt(6))/225
      (16-sqrt(6))/36 (16+sqrt(6))/36 1/9];
    b=[(16-sqrt(6))/36 (16+sqrt(6))/36 1/9 ]';
    c=[(4-sqrt(6))/10 (4+sqrt(6))/10 1];

  %Lobatto IIIA methods -- order 2s-2
  case 'LIIIA2'
    r=0;
    A=[0 0
      1/2 1/2];
    b=[1/2 1/2]';
    c=[0 1]';
  case 'LIIIA3'
    r=0;
    A=[0 0 0
      5/24 1/3 -1/24
      1/6 2/3 1/6];
    b=[1/6 2/3 1/6]';
    c=[0 12 1];

%==================Miscellaneous Methods================

  case 'Mid22'
    %Midpoint 22 method
    s=2; r=0.5;
    A=[0  0
      1/2 0];
    b=[0 1]'; c=[0 1/2]';
  case 'MTE22'
    %Minimal truncation error 22 method (Heun)
    s=2; r=0.5;
    A=[0  0
       2/3 0];
    b=[1/4 3/4]'; c=[0 2/3]';

  case 'CN22'
    %Crank-Nicholson
    s=2; r=2;
    A=[0  0
       1/2 1/2];
    b=[1/2 1/2]'; c=[0 1]';

  case 'Heun33'
    s=3; r=0;
    A=[0 0 0; 1/3 0 0; 0 2/3 0];
    b=[1/4 0 3/4]'; c=sum(A,2);

  case 'RK44'  %Classical fourth order
    s=4; r=0;
    A=[0 0 0 0; 1/2 0 0 0; 0 1/2 0 0; 0 0 1 0];
    b=[1/6 1/3 1/3 1/6]'; c=sum(A,2);

%====================DSRK Methods=======================

  case 'DSso2'
    %CBM's DSRKso2
    s=2; isdsrk=1;
    A=[3/4 -1/4
        1    0];
    W=[1/2   0
       1     0];
    b=[1 0]'; c=[1/2 1]';

  case 'DSRK2'
    %CBM's DSRK2
    s=2; isdsrk=1;
    A=[1/2 -1/2
       1/2  1/2];
    W=[ 0    0
       1/2  1/2];
    b=[1/2 1/2]'; c=[0 1]';
  case 'DSRK3'
    %Zennaro's DSRK3
    s=3; isdsrk=1;
    A=[5/2 -2 -1/2
       -1  2  -1/2
       1/6 2/3 1/6];
    W=[ 0  0  0
       7/24 1/6 1/24
       1/6 2/3 1/6];
    b=[1/6 2/3 1/6]'; c=[0 1/2 1]';
 %===================="Non-SSP" Methods of Wong & Spiteri=======================
  case 'NSSP21'
    m=2; r=0;
    A=[0   0
       3/4 0];
    b=[0 1]'; c=[0 3/4]';

  case 'NSSP32'
    m=3; r=0;
    A=[0   0 0
       1/3 0 0
       0   1 0];
    b=[1/2 0 1/2]'; c=[0 1/3 1]';

  case 'NSSP33'
    m=3; r=0;
    A=[0    0    0
       -4/9 0    0
       7/6  -1/2 0];
    b=[1/4 0 3/4]'; c=[0 -4/9 2/3]';

  case 'NSSP53'
    m=5; r=0;
    A=[0 0 0 0 0
       1/7 0 0 0 0
       0 3/16 0 0 0
       0 0 1/3 0 0
       0 0 0 2/3 0];
    b=[1/4 0 0 0 3/4]'; c=[0 1/7 3/16 1/3 2/3]';
end
```

- *Order of a Runge–Kutta method:*

```
function p=rk_order(A,b,c)
%By David Ketcheson
```

```
%
%Determine order of a RK method, up to sixth order
%Order conditions from text of Hairer, Norsett, & Wanner
%
%For an m-stage method, input A should be a m x m matrix;
%b and c should be column vectors of length m

eps=1.e-14;
m=length(b); % # of stages
em=ones(m,1);
p=0;

if sum(b)-1<eps
  p=1;
end

z(1)=sum(A'*b)-1/2;
if (p==1 && abs(z(1))<eps) p=2; end

z(1)=c'.^2*b-1/3;
z(2)=b'*A^2*em-1/6;
if(max(abs(z))<eps && p==2) p=3; end

z(1)=b'*c.^3-1/4;
z(2)=(b.*c)'*A^2*ones(m,1)-1/8;
z(3)=b'*A*c.^2-1/12;
z(4)=b'*A^2*c-1/24;
if(max(abs(z))<eps && p==3) p=4; end

z(1)=c'.^4*b-1/5;
z(2)=(b.*c.^2)'*A*c-1/10;
z(3)=b'*(A*c).^2-1/20;
z(4)=(b.*c)'*A*c.^2-1/15;
z(5)=b'*A*c.^3-1/20;
z(6)=(b.*c)'*A^2*c-1/30;
z(7)=b'*A*diag(c)*A*c-1/40;
z(8)=b'*A^2*c.^2-1/60;
z(9)=b'*A^3*c-1/120;
if(max(abs(z))<eps && p==4) p=5; end

if p==5
  z(1)=c'.^5*b-1/6;
  z(2)=b'*diag(c).^3*A*c-1/12;
  z(3)=b'*diag(c)*(A*c).^2-1/24;
  z(4)=b'*diag(c).^2*A*c.^2-1/18;
  z(5)=b'*((A*c.^2).*(A*c))-1/36;
  z(6)=b'*diag(c)*A*c.^3-1/24;
  z(7)=b'*A*c.^4-1/30;
  z(8)=b'*diag(c).^2*A^2*c-1/36;
  z(9)=b'*((A^2*c).*(A*c))-1/72;
  z(10)=b'*diag(c)*A*diag(c)*A*c-1/48;
  z(11)=b'*A*diag(c).^2*A*c-1/60;
  z(12)=b'*A*(A*c).^2-1/120;
  z(13)=b'*diag(c)*A^2*c.^2-1/72;
  z(14)=b'*A*diag(c)*A*c.^2-1/90;
  z(15)=b'*A^2*c.^3-1/120;
  z(16)=b'*diag(c)*A^3*c-1/144;
  z(17)=b'*A*diag(c)*A^2*c-1/180;
  z(18)=b'*A^2*diag(c)*A*c-1/240;
  z(19)=b'*A^3*c.^2-1/360;
  z(20)=b'*A^4*c-1/720;
  if(max(abs(z))<eps) p=6; print('This method has order at least six'); end
end
```

- Absolute monotonic polynomials:

```
function [gamma,R]=Rsp(s,p)
%By David Ketcheson
%
%Returns the optimal absolutely monotonic polynomial of degree s
%and order of accuracy p
%gamma contains the coefficients of the Taylor series about z=-r
%To construct the polynomial, use:
%>  syms z phi
%>  phi=simplify(sum((1.+z/R).^(0:s).*gamma));
%
%Uses the MATLAB optimization toolbox

%=========================================================
%Set options for linprog
opts=optimset('TolX',1.e-15,'TolFun',1.e-15,'MaxIter',10000000,...
              'LargeScale','on','Simplex','off','Display','off');
acc=1.e-15; %Accuracy of bisection search
%=========================================================

if p==s %In this case, the optimal polynomial is just the Taylor polynomial
  R=1;
  for i=0:p
    d(i+1)=R^i;
    for j=0:s
```

```
      B(i+1,j+1)=prod(j-(0:i-1));
    end
  end
  gamma=(B\ones(s+1,1))';

else
  M=s+1;
  rmax=s-p+1.0001;
  rmin=0;
  r=rmax;  %Initial guess
  c=zeros(M,1);
  clear B d;

  while (rmax-rmin>acc) %Find R by bisection
    %Set up and improve conditioning of equality constraints
    for i=0:p
      rescale=r^i; d(i+1)=r^i/rescale;
      for j=0:s
        B(i+1,j+1)=prod(j-(0:i-1))/rescale;
      end
    end
    %Test feasibility for this value of r
    [x,lambda,exitflag]=linprog(c,[],[],B,d,zeros(M,1),zeros(M,1)+1.e6,c,opts);
    if exitflag==1;
      rmin=r; r=(r+rmax)/2;
    else
      rmax=r; r=(rmin+r)/2;
    end
  end

  %Now get a feasible solution so we have the coefficients of the method
  R=rmin;
  for i=0:p
    rescale=R^i;
    d(i+1)=R^i/rescale;
    for j=0:s
      B(i+1,j+1)=prod(j-(0:i-1))/rescale;
    end
  end
  [gamma,lambda,exitflag]=linprog(c,[],[],B,d,zeros(M,1),zeros(M,1)+1.e6,c,opts)
end
```