What to do with Stiff Systems? Notes on Deferred Correction Methods for Solving Temporal Differential Equation Systems

Namdi Brandon, Ph.D

July 11, 2018

Abstract

This contains information on a novel way to solve time-dependent differential equation systems using deferred correction methods. Specifically, there is information about the a recently developed method that is designed to perform well for stiff systems. This document is intended to be notes for the purpose of helping give a preliminary understanding on the mathematics behind the Jacobian-Free Newton-Krylov method (see Qu et al. [2016]). I suggest reading these notes first before diving into the full descriptions of the presented material in the References section. On the other hand, if you feel like diving head-first (figuratively speaking, of course) into the full, unadulterated description of the mathematics of the presented material without the aid of these notes, by all means do so. After all, fortune favors the bold. But I digress. I hope you find this document helpful.

Namdi

1 Introduction

We are tying to solve the following ordinary differential equation (ODE) system

$$\begin{cases} \frac{dy(t)}{dt} = f(t, y(t)) \\ y(0) = y_0 \end{cases} \tag{1}$$

which has the solution

$$y(t) = y_0 + \int_0^t f(\tau, y(\tau))\, \mathrm{d}\tau$$

1.1 Overview

This document procedes as follows. First, we discuss a few methods to numerically approximate solutions to ODEs: the spectral (Gauss quadrature) solution,

Euler's method, the Spectral Deferred Corrections (SDC) method and a novel Jacobian-Free Newton-Krylov (JFNK) method designed for stiff systems. Afterwards, we give some examples comapring the performance of the previously discussed methods. And lastly, I give some pseudocode underlying how some of these algorithms work.

1.2 Definitions, abbreviations, and mathematical symbols

Table 1: Table of Abbreviations

Abbreviation	Meaning
JFNK	Jacobian-Free Newton-Krylov
ODE	ordinary differential equation
SDC	Spectral Deferred Corrections

Table 2: Mathematical Symbols

Symbol	Meaning
Δt	the temporal step size
p	the number of spectral (Gaussian) nodes
S	the spectral integration (Gaussian quadrature) matrix
$\tilde{S}$	the backward Euler integration matrix
C	the correction matrix
$\rho(A)$	the spectral radius (i.e., the largest magnitude of the eigenvalues) of a matrix A
$J_H(x_0)$	the Jacobian matrix of a function $H(x)$ evaluated at the value $x = x_0$
t	time

The backward Euler integration matrix applied to the time step Δt

$$\Delta t\tilde{S} = \begin{bmatrix} \Delta t_0 & 0 & \cdots & 0 & 0 \\ \Delta t_0 & \Delta t_1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdots & 0 & 0 \\ \Delta t_0 & \Delta t_1 & \cdots & \Delta t_{p-2} & 0 \\ \Delta t_0 & \Delta t_1 & \cdots & \Delta t_{p-2} & \Delta t_{p-1} \end{bmatrix}$$

where $\Delta t_i = t_{i+1} - t_i$

The spectral integration (Gaussian quadrature) matrix applied to the time step Δt where each entry is of S is given by

$$\Delta t S_{ij} = \int_0^{t_i} \left(\prod_{k \neq j} \frac{x - x_k}{x_j - x_k} \right) \mathrm{d}x$$

2 Spectral Solution

First, a time step of size $t \in [0, \Delta t]$ is broken into p spectral (Gaussian) nodes. The spectral (Gaussian quadrature) solution is defined as the following

$$y = y_0 + \Delta t S F(y) \tag{2}$$

where

$$y = \begin{bmatrix} y(t_0) \\ y(t_1) \\ y(t_2) \\ \vdots \\ y(t_{p-1}) \end{bmatrix}, y_0 = \begin{bmatrix} y_0 \\ y_0 \\ y_0 \\ \vdots \\ y_0 \end{bmatrix}, F(y) = \begin{bmatrix} f(t_0, y(t_0)) \\ f(t_1, y(t_1)) \\ f(t_2, y(t_2)) \\ \vdots \\ f(t_{p-1}, y(t_{p-1})) \end{bmatrix}$$

and S is a spectral integration matrix (Gaussian quadrature matrix) that's normalized for $t \in [0, 1]$ and is a dense matrix. The spectral solution has the property that it is $O(\Delta t^{2p})$.

- **Pros**: A high order solution $O(\Delta t^{2p})$. There are additional great properties. It's A-stable, symplectic (energy conserving), and the error decays exponsentionally when p increases.
- **Cons**: The solution is difficult (expensive) to invert.

3 Backward Euler

The spectral solution is hard to invert due to S being dense. Therefore, use an Euler method to solve a preconditioned system

$$y = y_0 + \Delta t \tilde{S} F(y) \tag{3}$$

where $\tilde{S}$ is the Euler's method integration matrix (easy to invert).

- **Pros**: Euler's method is simple
- **Cons**: Euler's method is simple. It's $O(\Delta t)$, which limits the size of the time step Δt and may cause simulations to take substantially more steps, increasing run time.

4 Spectral Deferred Corrections (SDC)

SDC is an iterative way to obtain high order solutions, using first order methods. Done by using an initial approximation to the solution (e.g. via Euler's method), and imporving the solution by iteratively approximating the error (the deferred correction). That is, SDC works by doing the following

1. obtain a low-order initial solution $\tilde{y}$ that approximates the exact solution y

2. approximate the error $\delta = y - \tilde{y}$ with the approximation $\tilde{\delta}$
3. update the latest approximation for the solution $\tilde{y} \leftarrow \tilde{y} + \tilde{\delta}$
4. go to step to 2 and repeat until convergence $||\tilde{\delta}|| \approx 0$

- **Pros**: The converged solution is the spectral solution (2). That is to say, SDC is able to obtain high order solutions $O(\Delta t^{2p})$, which implies the ability to take larger time steps than Euler's method. SDC iteratively uses Euler's method (a simple method) to do so.
- **Cons**: One does not know *a priori* how many iterations that SDC will take to converge. SDC could take many iterations to convege, increasing run time. This is especially, true for stiff systems. To avoid this, one may decrease the step size Δt (accelerating convergence of SDC), at the cost of taking more time steps (incresing run time). **SDC may not be practical for stiff systems**.

4.1 Theory

We are trying to obtain the spectral solution

$$y = y_0 + \Delta t S F(y).$$

However, we cannot feasibly calculate the true solution $y(t)$. However, if we have an approximation to the solution $\tilde{y}(t)$, we define the error $\delta(t) = y(t) - \tilde{y}(t)$. Therefore, we can solve for δ and have

$$\tilde{y} + \delta = y_0 + \Delta t S F(\tilde{y} + \delta).$$

Now we solve a preconditioned (approximate) system for the approximation of the exact error δ, $\tilde{\delta}$

$$\tilde{y} + \tilde{\delta} = y_0 + \Delta t \tilde{S} F(\tilde{y} + \tilde{\delta}) + \Delta t (S - \tilde{S}) F(\tilde{y})$$

Now let $\tilde{v} = \tilde{y} + \tilde{\delta}$ and rewrite the system as

$$\tilde{v} = y_0 + \Delta t (S - \tilde{S}) F(\tilde{y}) + \Delta t \tilde{S} F(\tilde{v}) \tag{4}$$

We now do the following

1. solve (4) for $\tilde{v}$ using Euler's method. Note that (4) resembles (3) but with an extra $\Delta t (S - \tilde{S}) F(\tilde{y})$ on the right hand side
2. set $\tilde{\delta} = \tilde{v} - \tilde{y}$
3. update the solution $\tilde{y} \leftarrow \tilde{y} + \tilde{\delta}$

The SDC method continues iteratively until convergence. Given that SDC undergoes k iterations, SDC is an $O(\Delta t^k)$ method. The approximation $\tilde{y}$ approximately satisfies (up to $O(\tilde{\delta})$ error)

$$\tilde{y} = y_0 + \Delta t S F(\tilde{y})$$

which is the spectral solution (2).

4.2 Properties

To understand the properties of SDC, let's examine the model system

$$\begin{cases} \frac{dy(t)}{dt} = \lambda y(t) \\ y(0) = y_0 \end{cases}$$

where $\lambda < 0$. The spectral solution is now

$$\begin{aligned} y &= y_0 + \lambda \Delta t S y \\ \implies (I - \lambda \Delta t S) y &= y_0 \end{aligned}$$

We now use Euler's method as a preconditioner

$$\begin{aligned} (I - \lambda \Delta t S) y &= y_0 && \text{the spectral system to solve} \\ \implies (I - \lambda \Delta t \tilde{S})^{-1} (I - \lambda \Delta t S) y &= (I - \lambda \Delta t \tilde{S})^{-1} y_0 && \text{apply Euler's method} \\ \implies (I - C) y &= \tilde{b} && \text{rewrite} \\ \implies y &= (I - C)^{-1} \tilde{b} && \text{the solution} \end{aligned}$$

where $\tilde{b} = (I - \lambda \Delta t \tilde{S})^{-1} y_0$ is the Euler's method solution and $C = I - (I - \lambda \Delta t \tilde{S})^{-1}(I - \lambda \Delta t S)$ is the **correction matrix**. If $\rho(C) < 1$, then we can write the SDC solution as

$$\begin{aligned} y &= (I - C)^{-1} \tilde{b} \\ &= (I + C + C^2 + \dots) \tilde{b} \end{aligned}$$

where $\tilde{\delta}^{[k]} = C^{[k+1]} \tilde{b}$, $k = 0, 1, \dots$

4.2.1 Convergence behavior

SDC has different convergence behaviors, which can be seen by understanding the correction matrix C. Note, that the correction matrix can be written as

$$\begin{aligned} C &= I - (I - \lambda \Delta t \tilde{S})^{-1} (I - \lambda \Delta t S) \\ &= \lambda \Delta t (I - \lambda \Delta t \tilde{S})^{-1} (S - \tilde{S}) \end{aligned}$$

When the system is non-stiff (i.e, $|\lambda \Delta t| \ll 1$), we have

$$C_{ns} = \lambda \Delta t (I + \lambda \Delta t \tilde{S} + (\lambda \Delta t \tilde{S})^2 + \cdots)(S - \tilde{S})$$

Note that to leading order $C_{ns} \approx \lambda \Delta t$ and $C_{ns}^k \approx (\lambda \Delta t)^k$. Hence, for the non-stiff case, we should expect that after k SDC iterations, the integration accuracy of SDC should be $O(\Delta t^k)$.

When the system is extremely stiff (i.e, $|\lambda \Delta t| \approx 1$), we have

$$C_s = I - \tilde{S}^{-1} S$$

which is order reduction. Order reduction, in this case, is a state when the error decays no longer as a function of the step size Δt but instead by a fixed constant

(independent of Δt. Order reduction implies a much slower rate of convergence. For stiff systems, SDC converges to the spectral solution with $O(\rho(C_s)^k)$. Table 3 showcases the spectral radius $\rho(C_s)$ for various numbers of Gauss-Lobatto nodes (a set of Gauss nodes where both end points are used) .

Table 3: $\rho(C_s)$ for different numbers p of Gauss-Lobatto nodes, extremely stiff case

p	3	4	5	6	7	8	9
$\rho(C_s)$	0.5000	0.5922	0.6837	0.7576	0.8150	0.8600	0.8957
p	10	11	12	13	**14**	15	16
$\rho(C_s)$	0.9247	0.9485	0.9685	0.9853	**0.9998**	1.0123	1.0233
p	17	18	19	20	21	25	50
$\rho(C_s)$	1.0330	1.0415	1.0492	1.0560	1.0622	1.0820	1.1333

For example, Table 3 shows that even though using SDC nodes with $p = 14$ Gauss-Lobatto nodes leads to the spectral solution after 1 time step of size Δt as being $O(\Delta t^{2\times 14-1})$, in a stiff system, the solution would take a large amount of iterations to converge because after k iterations the error would have decayed by 0.9998^k. On the other hand, choosing p nodes such that $p > 14$ would cause SDC to diverge.

5 Modified Jacobian-Free Newton-Krylov Method

The JFNK method is an iterative method that takes advantage of the properties behind SDC iterations to greatly accelerate convergence for SDC for stiff systems (where SDC experiences order reduction). This method converges to the same solution as SDC but with substantially fewer calculations.

- **Pros**: The converged solution is the spectral solution (2).This method performs well for stiff systems while taking large step size Δt.
- **Cons**: Like SDC, one does not know how many iterations are needed for this JFNK method to converge.

5.1 Theory

5.1.1 Spectral solution

Let's assume that the ODE system that we are trying to solve (1) is stiff. We are trying to obtain the spectral solution (2) so that we have

$$\begin{aligned} y &= y_0 + \Delta t S F(y) \\ \implies y - \Delta t S F(y) - y_0 &= 0 \end{aligned} \tag{5}$$

We can now view (5) as a non-linear **root fining problem** for y. That is, we need to find a y such that

$$A(y) = 0 \tag{6}$$

where $A(y) = y - \Delta t SF(y) - y_0$. Given that we have a non-linear root fining problem, one can try to use Newton's method. However, using Newton's method to directly approximate the spectral solution would be inefficient and not practical.

5.1.2 SDC solution

Solving for the spectral solution directly is difficult. So let's solve an approximate (preconditioned) system instead.

Given an initial approximation $y^{[0]}$, SDC obtains an approximate solution

$$\begin{cases} \delta^{[k]} & = H(y^{[k]}) \quad \text{calculate an SDC correction} \\ y^{[k+1]} & = y^{[k]} + \delta^{[k]} \quad \text{update the SDC solution} \end{cases} \tag{7}$$

where $H(\tilde{y})$ is the procedure that completes 1 SDC iteration. Note that SDC converges when we find a value $\tilde{y}$ such that

$$H(\tilde{y}) = 0 \tag{8}$$

Let's use Newton's method to solve for the solution of the preconditioned (SDC) system (8) as apposed to (6). Let $x^{[0]}$ be an initial approximation (e.g. a solution obtained via Euler's method) to the solution of (8). We can write Newton's method as solving for the successive approximation $x^{[1]}$

$$\begin{aligned} x^{[1]} &= x^{[0]} - J_H^{-1}(x^{[0]})H(x^{[0]}) \quad \text{Newton's method} \\ \implies J_H(x^{[0]})\left(x^{[1]} - x^{[0]}\right) &= -H(x^{[0]}) \\ \implies J_H(x^{[0]})\Delta x &= -H(x^{[0]}) \quad \text{a Newton iteration} \end{aligned}$$

where $\Delta x = x^{[1]} - x^{[0]}$.

Using Newton's method to find the solution to $H(x) = 0$ comes down to solving a series of linear systems

$$\begin{cases} J_H(x^{[n]})\Delta x & = -H(x^{[n]}) \\ x^{[n+1]} & = x^{[n]} + \Delta x \quad n = 0, \dots \end{cases} \tag{9}$$

As written, the Newton iteration (9) is difficult to solve because of the Jacobian matrix $J_H(x^{[n]})$ has the following properties:

1. $J_H(x^{[n]})$ usually does not have a known analytical representation
2. $J_H(x^{[n]})$ could be numerically approximated, which is computationally expensive
3. assuming we have a numerical approximation of $J_H(x^{[n]})$, solving the system (9) can be computationally expensive

For these issues concerning the Jacobian matrix, Newton's method may not be feasible. This has led to research to find ways to solve Newton's method while avoiding using the Jacobian in (9) explcitly (i.e., being free of the Jacobian).

5.1.3 A modified Jacobian-Free Newton-Krylov method

We need to find a way to solve (9) **without using the Jacobian matrix explicitly**. Keeping both this and the SDC procedure (7) in mind, we can look at properties of $H(\tilde{y})$

$$\begin{aligned}
H\left(y^{[k+1]}\right) &= H\left(y^{[k]} + \delta^{[k]}\right) && \text{identity from (7)} \\
\implies \delta^{[k+1]} &= H\left(y^{[k]} + \delta^{[k]}\right) && \text{by definition from (7)} \\
\implies \delta^{[k+1]} &= H\left(y^{[k]}\right) + J_H(y^{[k]})\delta^{[k]} + O(||\delta^{[k]}||^2) && \text{Taylor expansion} \\
\implies \delta^{[k+1]} &= \delta^{[k]} + J_H(y^{[k]})\delta^{[k]} + O(||\delta^{[k]}||^2) && \text{by definition from (7)} \\
\implies \delta^{[k+1]} - \delta^{[k]} &= J_H(y^{[k]})\delta^{[k]} + O(||\delta^{[k]}||^2) && \text{rewrite}
\end{aligned}$$

We have

$$\delta^{[k+1]} - \delta^{[k]} \approx J_H(y^{[k]})\delta^{[k]} \tag{10}$$

That is to say, that the Jacobian matrix J_H applied at the k^{th} SDC solution $y^{[k]}$ times the k^{th} deferred correction $\delta^{[k]}$ is equal to the difference between successive SDC iterations $\delta^{[k+1]} - \delta^{[k]}$. This is a big deal!

He can solve Newton iteration (9) for the SDC procudure **without the direct use of the Jacobian matirx!** In (9), if we express Δx as a linear combination of deferred correction vectors from p consecutive iterations of SDC

$$\Delta x = c_0\delta^{[0]} + c_1\delta^{[1]} + \dots c_{p-1}\delta^{[p-1]} \tag{11}$$

and use $y^{[p]}$ as the approximate solution in the n^{th} Newton iteration (i.e., we set $x^{[n]} = y^{[p]}$ in (9), we obtain

$$\begin{aligned}
J_H(y^{[p]})\Delta x &= -H(y^{[p]}) && \text{using (9)} \\
\implies J_H(y^{[p]})\Delta x &= -\delta^{[p]} && \text{by definition from (7)} \\
\implies J_H(y^{[p]})\sum_{j=0}^{p-1} c_j\delta^{[j]} &= -\delta^{[p]} && \text{using from (11)} \\
\implies \sum_{j=0}^{p-1} c_j J_H(y^{[p]})\delta^{[j]} &= -\delta^{[p]} && \\
\implies \sum_{j=0}^{p-1} c_j(\delta^{[k+1]} - \delta^{[k]}) &\approx -\delta^{[p]} && \text{using (10)}
\end{aligned} \tag{12}$$

Now we just solve (12) for the unknowns $c = [c_0 \dots c_{p-1}]^T$. This a system of equations

$$Ac = -\delta^{[p]} \tag{13}$$

where the columns of A are the following

$$A = \left[\delta^{[1]} - \delta^{[0]} \middle| \quad \delta^{[2]} - \delta^{[1]} \middle| \quad \cdots \middle| \quad \delta^{[p]} - \delta^{[p-1]}\right]$$

That is, instead of dealing with the Jacobian matrix in the Newton iteration by solving

$$J_H(x^{[n]})\Delta x = -H(x^{[n]}),$$

we instead use SDC approximation for $x^{[n]}$ by setting $x^{[n]} = y^{[p]}$ to get

$$J_H(y^{[p]})\Delta x = -\delta^{[p]}$$

and use the the properties of SDC and solve a Jacobinan-Free system of equations for c

$$Ac = -\delta^{[p]}$$

to get the next solution in the Newton's iteration $x^{[n+1]}$

$$x^{[n+1]} = y^{[p]} + \sum_{j=0}^{p-1} c_j \delta^{[j]}.$$

In this way, we can use Newton's iteration to solve for $H(\tilde{y}) = 0$ using Newton's method without ever having to explicitly deal with the Jacobian matrix!

6 Applications

6.1 Stiff system: cosine problem

Let's see the performance for SDC and the modified JFNK method for the following stiff, linear ODE system.

$$\begin{aligned}
\frac{dy_1(t)}{dt} &= \lambda_1(y_1(t) - \cos(t)) - \sin(t) \\
\frac{dy_2(t)}{dt} &= \lambda_2(y_2(t) - \cos(t)) - \sin(t) \\
\frac{dy_3(t)}{dt} &= \lambda_3(y_3(t) - \cos(t)) - \sin(t)
\end{aligned}$$

where $\lambda_1 = -\frac{10^{-3}}{\pi}, \lambda_2 = -\frac{10^2}{\pi}, \lambda_3 = -\frac{10^5}{\pi}$ and the initial condition is $y_1(0) = y_2(0) = y_3(0) = 1$. Note that the exact solution for this system is

$$y_1(t) = y_2(t) = y_3(t) = \cos(t)$$

Figure 1 shows the results of using SDC and the modified JFNK method for solving this system in the interval $t \in [0, \Delta t]$ using $p = 5$ Gauss-Lobatto node points and $\Delta t = 1$ using SDC and the modified JFNK methods.

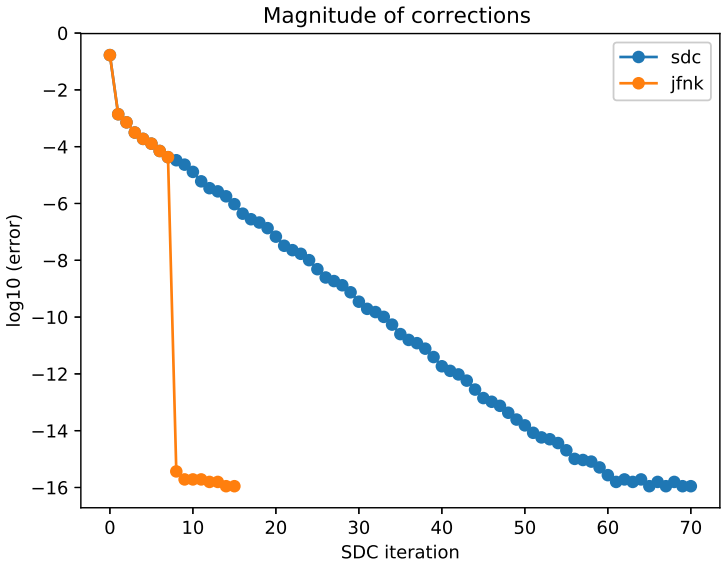

Figure 1: Overall performance of the methods. Shown are the relative magnitude of the corrections $\frac{||\delta^{[k]}||_\infty}{||y^{[k]}||_\infty}$ over all components for the k^{th} iteration

The following gives some analysis of Figure 1.

Backward Euler Note that value of $\approx 10^{-1}$ at iteration 0 is an approximation of the error from using backward Euler method, since it is the first step in both the SDC and JFNK method.

SDC SDC converges to the spectral solution (2) in about 60 iterations. Because the problem is stiff, an iteration 1, SDC experiences order convergence and the corrections decreases by about 0.6837 (from Table 3) for each iteration.

JFNK JFNK converges to the spectral solution (2) in about 10 iterations. Once the order convergence is detected, around iteration 1, the method does $p+1$ SDC iterations and to solves the Jacobian system in Newton's method using (12). The *jump* in accuracy from the SDC iterations is due to solving (7). This jump in convergence is the strength of the JFNK. The reason why only 1 Newton iteration is needed for JFNK to converge to the solution is because the ODE is a lineary system and Newton's method converges quadraticly.

Figure 2 shows the convergence capability for each component for both the SDC method and the JFNK method, respectively. Note that in Figure 2 the approximations for y_1, the non-stiff component $\left(|\lambda_1| = \frac{10^{-3}}{\pi}\right)$, converges

to the solution much faster than y_2 and y_3, which are substantially more stiff $\left(|\lambda_2| = \frac{10^2}{\pi} \text{ and } |\lambda_3| = \frac{10^5}{\pi}, \text{ respectively}\right)$.

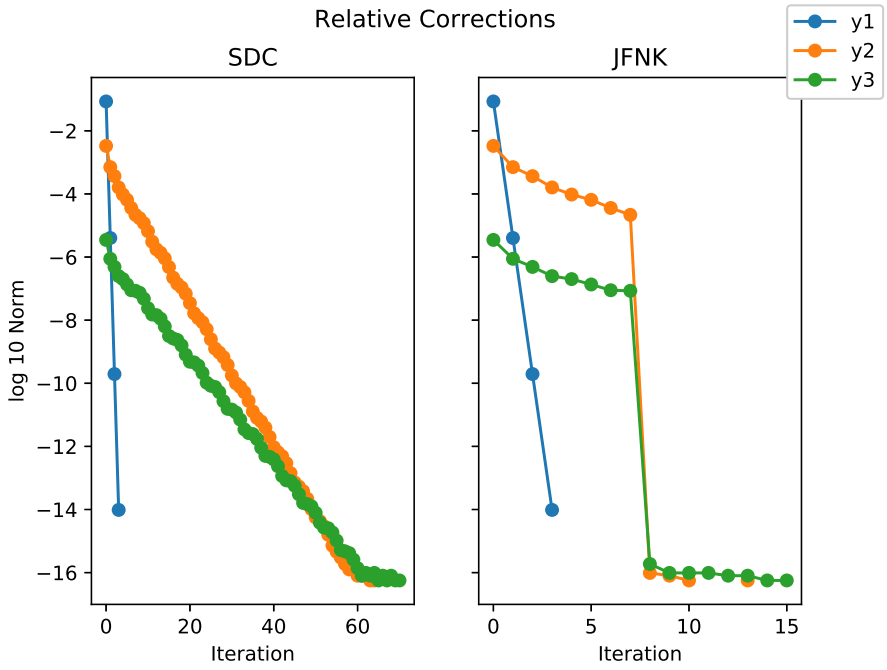

Figure 2: Performance of the methods for each component. Shown are the relative magnitude of the corrections $\frac{||\delta^{[k]}||_\infty}{||y^{[k]}||_\infty}$ for each component for the k^{th} iteration

6.2 Stiff system: Van der Pol's Oscillator

Let's now examine the performance of SDC and the modified JFNK method for the following stiff, non-linear ODE system, Van der Pol's oscillator.

$$\begin{aligned} \frac{dy_1(t)}{dt} &= y_2(t) \\ \frac{dy_2(t)}{dt} &= \lambda\left[1 - y_1^2(t)\right] y_2(t) - y_1(t) \end{aligned}$$

where $\lambda = 20$ and $[y_1(0), y_2(0)]^T = [2, 1]^T$.

Figure 3 shows the results of using SDC and the modified JFNK method for solving this system in the interval $t \in [0, \Delta t]$ using $p = 10$ Gauss-Lobatto node points and $\Delta t = 0.25$ using SDC and the modified JFNK methods.

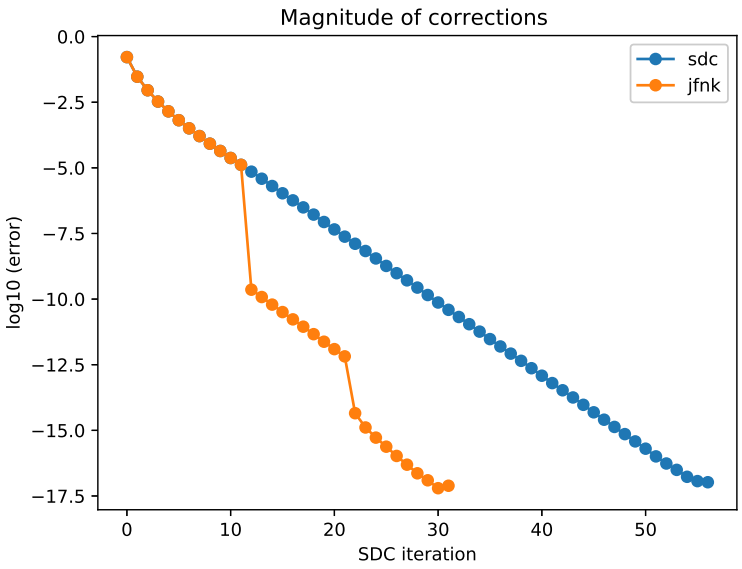

Figure 3: Overall performance of the methods. Shown are the relative magnitude of the corrections $\frac{||\delta^{[k]}||_\infty}{||y^{[k]}||_\infty}$ over all components for the k^{th} iteration

The following gives some analysis of Figure 3.

Backward Euler Note that value of $\approx 10^{-1}$ at iteration 0 is an approximation of the error from using backward Euler method, since it is the first step in both the SDC and JFNK method.

SDC SDC converges to the spectral solution (2) in about 58 iterations. Because the problem is stiff, SDC experiences order convergence and the corrections decreases by about 0.52 for each iteration, indicating that the problem is stiff but not stiff enough to have maximum order reduction of 0.9247 (from Table 3) .

JFNK JFNK converges to the spectral solution (2) in about 31 iterations. Once the order convergence is detected, the method does $p + 1$ SDC iterations and to solves the Jacobian system in Newton's method using (12). In this example, the JFNK solves the Newton's iterations (9) 3 times. These jumps greatly accelerate convergence

Figure 4 shows the convergence capability for each component for both the SDC method and the JFNK method, respectively.

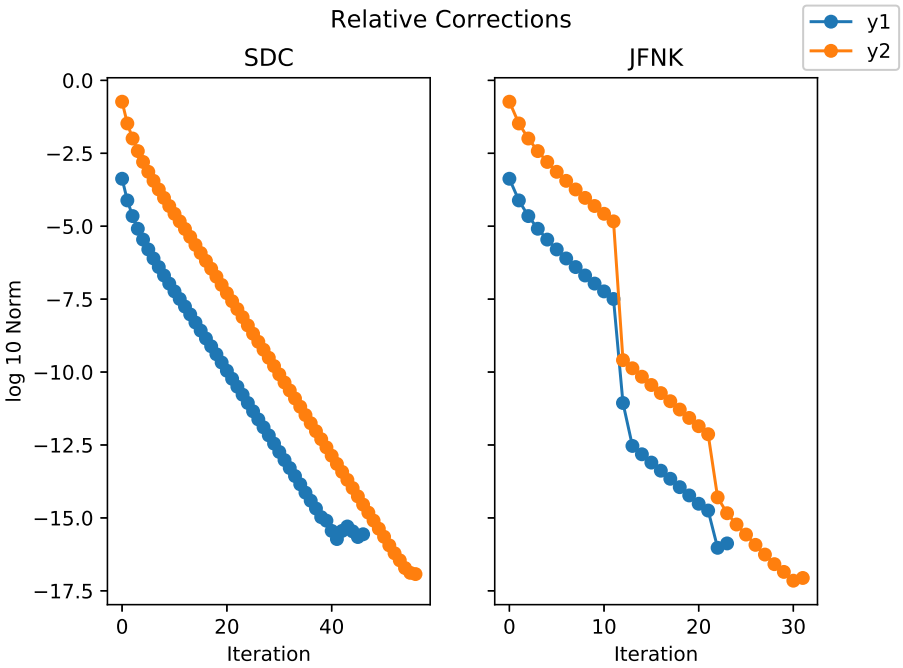

Figure 4: Performance of the methods for each component. Shown are the relative magnitude of the corrections $\frac{||\delta^{[k]}||_\infty}{||y^{[k]}||_\infty}$ for each component for the k^{th} iteration

7 Pseudocode

7.1 SDC algorithm

Algorithm 1 Use SDC to approximate the solution at each node within the time step

procedure SDC SWEEP
  Given an initial approximation y of the exact solution $y(t)$

$$y = [y_0\, y_1\, \cdots\, y_{p-1}]^T$$

Calculate the correction δ:
  calculate $f = F(y)$
  for $i = 0, \ldots, p-1$ **do**
    set $h_i = t_i - t_{i-1}$
    set $b_i = y_{i-1} + \Delta t S^i_{i-1} \cdot f - h_i f_i$
    use backward Euler method to solve for v_i as seen in (4)

$$v_i - h_i F(v_i) = b_i$$

    Calculate the correction at the i^{th} node

$$\delta_i = v_i - y_i$$

Summary of the method:
  $\delta = H(y)$.

Algorithm 2 Run SDC iteratively until the solution converges

procedure SDC ITERATIONS
  Given an initial approximation $\tilde{y}$
  set $y^{[0]} = \tilde{y}$
  set $k = 0$
  while (not converged) **do**
    $\delta^{[k]} = H(y^{[k]})$
    $y^{[k+1]} \leftarrow y^{[k]} + \delta^{[k]}$
    if $\frac{||\delta^{[k]}||}{||y^{[k]}||} < \epsilon$ **then**
      the solution has converged.
    update $k \leftarrow k + 1$
  return $y^{[k]}$ as the solution

7.2 JFNK

Algorithm 3 Run intitial SDC iterations within the JFNK method until order convergence is observed

procedure JFNK INITIAL ITERATIONSL
    run backward Euler sweep to obtain $y^{[0]}$
Initial iterations:
    run 2 SDC iterations

$$\begin{cases} \delta^{[k]} & \leftarrow H(y^{[k]}) \\ y^{[k+1]} & \leftarrow y^{[k]} + \delta^{[k]}, k = 0, 1 \end{cases}$$

    $k \leftarrow 2$
    while (not is stiff) & (not converged) **do**
        calculate the ratio of corrections

$$r = \frac{||\delta^{[k-1]}||}{||\delta^{[k-2]}||}$$

        if $\frac{r}{\rho(C_s)} > 0.1$ **then**
            Order reduction has been observed. The problem is stiff; hence, stop initial itearations.
        else
            run SDC for another iteration

$$\begin{cases} \delta^{[k]} & \leftarrow H(y^{[k]}) \\ y^{[k+1]} & \leftarrow y^{[k]} + \delta^{[k]} \end{cases}$$

            $k \leftarrow k + 1$
        Stop the procedure if SDC has converged

Algorithm 4 Solve the Newton's method iterations using the modified Jacobian-Free Newton-Krylov method

procedure JFNK ITERATIONS

When SDC has met order reduction, use Newton's method. Assume an initial approximation $\tilde{y}$.

JFNK iterations:

set $y^{[0]} \leftarrow \tilde{y}$

run SDC for $p+1$ iterations to obtain

$$\begin{cases} \delta^{[i]} & \leftarrow H(y^{[i]}) \\ y^{[i+1]} & \leftarrow y^{[i]} + \delta^{[i]}, i = 0 \dots p \end{cases}$$

solve $H(y) = 0$ using Newton's method with $y^{[p]}$ as the approximate solution

$$J_H(y^{[p]})\Delta y = -H(y^{[p]}) = -\delta^{[p]} \tag{14}$$

and the improved solution in the Newton's method becomes

$$\tilde{y} \leftarrow y^{[p]} + \Delta y$$

If we rewrite Δy in (14) as

$$\Delta y = \sum_{j=0}^{p-1} c_j \delta^{[j]} \tag{15}$$

Inserting (15) into (14), we have

$$\begin{aligned} J_H(y^{[p]}) \sum_{j=0}^{p-1} c_j \delta^{[j]} &= -\delta^{[p]} \\ \implies \sum_{j=0}^{p-1} c_j J_H(y^{[p]}) \delta^{[j]} &= -\delta^{[p]} \\ \implies \sum_{j=0}^{p-1} c_j (\delta^{[j+1]} - \delta^{[j]}) &= -\delta^{[p]} \end{aligned} \tag{16}$$

We can rewrite (16) and solve for the unknowns $c = [c_0 \dots c_{p-1}]^T$

$$Ac = -\delta^{[p]}$$

where the columns of A are the following

$$A = \left[\delta^{[1]} - \delta^{[0]} \middle| \quad \delta^{[2]} - \delta^{[1]} \middle| \quad \cdots \middle| \quad \delta^{[p]} - \delta^{[p-1]}\right]$$

update the solution using (15)

$$\tilde{y} \leftarrow y^{[p]} + \sum_{j=0}^{p-1} c_j \delta^{[j]}$$

go to step 4

About me

Namdi Brandon received his Ph.D. in mathematics from the University of North Carolina at Chapel Hill in 2015.

References

Namdi Brandon. *Novel Integration in Time Methods via Deferred Correction Formulations and Space-Time Parallelization*. PhD thesis, The University of North Carolina at Chapel Hill, University of North Carolina at Chapel Hill Graduate School, Chapel Hill, NC, 8 2015.

Python Software Foundation. Python software foundation website, 2018. URL `https://www.python.org`. Accessed: 2018-06-01.

Jingfang Huang, Jun Jia, and Michael Minion. Accelerating the convergence of spectral deferred correction methods. *Journal of Computational Physics*, 214(2):633 – 656, 2006. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2005.10.004. URL `http://www.sciencedirect.com/science/article/pii/S0021999105004663`.

Wenzhen Qu, Namdi Brandon, Dangxing Chen, Jingfang Huang, and Tyler Kress. A numerical framework for integrating deferred correction methods to solve high order collocation formulations of odes. *Journal of Scientific Computing*, 68(2):484–520, Aug 2016. ISSN 1573-7691. doi: 10.1007/s10915-015-0146-9. URL `https://doi.org/10.1007/s10915-015-0146-9`.

Eric W. Weisstein. Lobatto quadrature. from mathworld–a wolfram web resource, 2018. URL `http://mathworld.wolfram.com/LobattoQuadrature.html`. Accessed: 2018-06-01.