SPECTRAL DEFERRED CORRECTION (SDC)

A. DONEV, COURANT

These notes are based on extensive prior work by Mike Minion. Using SDC efficiently in practice, especially for PDEs, requires a lot of additional tricks & machinery not described here (good for final project!)

Original SDC method comes from Rokhlin & Leslie Greengard (Courant)

SDC is a way to get spectral accuracy for ODEs. One use is to take larger time steps. Or, to solve PDEs like the KdV equation to spectral accuracy in space & time (final project!).

$$\varphi'(t) = F(t, \varphi(t))$$

$$\varphi(0) = \varphi_0$$

solve on $t \in (0, T]$

Key idea: Turn ODE into an integral equation & use spectral quadrature (orthogonal polynomials) such as:

1) Clenshaw-Curtis quadrature (Chebyshev polynomials) on type 1 Chebyshev grid (does not include endpoints) or type 2 Cheb. grid (includes endpoints)
2) Gauss-Legendre quadrature (no end points) or Gauss-Lobato quadrature (includes endpoints)

$$\varphi(t) = \varphi_0 + \int_0^t F(\tau, \varphi(\tau))\, d\tau$$

If we have $F_k = F(\tau_k, \varphi(\tau_k))$, $k = 1, .., M$, where $\{\tau_k\}$ are roots of some orthogonal polynomial / quadrature nodes, then we could find an orthogonal polynomial interpolant $\tilde{F}(t)$ s.t. $\tilde{F}(t_k) = F_k$ and integrate the interpolant

4

This gives us a spectral integration matrix

$$(S\vec{F})_k = \int_0^{t_k} \tilde{F}(\tau)\, d\tau$$

There are easily available public routines to construct S for different choices of orthogonal polynomial, e.g., function <u>intmat</u> in Chebfun library.

This gives us a system of $M \cdot N$ nonlinear equations (here N is the dimension of $\varphi(t)$):

$$\varphi_k = \varphi_0 + S\{F(t_k, \varphi_k)\}_{k=1,..M} \quad \dots (*)$$

If we solve this very large system (especially if M is large), we will get $\{\varphi_k\}_{k=1,\dots,M}$ and thus also an interpolating solution $\tilde{\varphi}(t)$!

This has the same order of accuracy as the underlying quadrature rule, but it is very expensive. How do we actually solve (*) efficiently?

This is where Greengard / Minion come in.

Assume we are given an approximation to the solution $\varphi^{(m)}(t)$ (usually an interpolant), where m is an iteration count

The integral equation for the <u>error</u> $\delta(t)=\varphi(t)-\varphi^{(m)}(t)$ is

$$\delta(t)=\int_0^t\left[F(\tau,\varphi^{(m)}(\tau)+\delta(\tau))-F(\tau,\varphi^{(m)}(\tau))\right]d\tau+\mathcal{E}(t)$$

where the residual

$$\mathcal{E}(t)=\varphi_0-\varphi^{(m)}(t)+\int_0^t F(\tau,\varphi^{(m)}(\tau))d\tau$$

measures the error in the integral equation (if $\mathcal{E}=0 \Leftrightarrow \delta=0$)

8

Matrix form of residual

$$\underset{\sim}{\tilde{\varepsilon}}(\underset{\sim}{\vec{\varphi}}) = \left(\vec{\varphi}_0 + \underset{=}{\Delta t}\,\underset{\sim}{\overleftrightarrow{S}}\,\underline{\underline{\vec{F}(\underset{\sim}{\vec{\varphi}})}}\right) - \underset{\sim}{\vec{\varphi}}$$

Gauss quadrature

Here Δt is a "large" time step size chosen based on memory requirements & stability

current guess for $\varphi(\{t_k\})$ e.g. $\{\varphi^{(m)}_k\}$

We want $\tilde{\varepsilon}(\vec{\varphi}^{(m)}) = 0 \Rightarrow$ ODE has been solved

SDC iteration:

1) Use some predictor method to solve ODE, for example, use Forward / Backward Euler depending on whether the ODE is non-stiff / stiff & get predicted solution:

$$\varphi^{(0)} = \varphi(\{t_k\})$$

Note: Whether the grid of quadrature nodes includes or not the left/right endpoint affects stability [for FE include left, but for BE include right].

There are spectral quadrature rules that include only one of the endpoints, called Gauss-Radau nodes, see course webpage for link.

Start iterating $m = 1, 2, \ldots$

2) Iterate until "convergence":

a) Compute $\tilde{\mathcal{E}}^{(m)} = \tilde{\mathcal{E}}(\vec{\varphi}^{(m)})$

using spectral quadrature

b) Compute correction $\vec{\delta}^{(k)}$

with FE/BE using (**)

(this is like solving ODE

using FE/BE on a non-uniform

grid of time points):

12

From

$$\delta(t) = \int_0^t \left[F(\tau, \varphi^{(m)}(\tau) + \delta(\tau)) - F(\tau, \varphi^{(m)}(\tau)) \right] d\tau + \varepsilon(t)$$

approximate as piecewise constant over (t_j, t_{j+1})

$$\Downarrow$$

$$\tilde{\delta}(t_{j+1}) - \tilde{\delta}(t_j) = (\tilde{\varepsilon}_{j+1} - \tilde{\varepsilon}_j)$$

$$(**) \quad (t_{j+1} - t_j) \left[F(\tau, (\varphi^{(m)} + \delta)(t_{j/j+1})) - F(\tau, \varphi^{(m)}) \right] d\tau$$

Forward Euler (j) / Backward Euler ($j+1$)

$j = 1, \dots, M$

with initial condition:

$$\tilde{\delta}_0 = \delta(t=0) = \varepsilon(t=0) = \tilde{\varepsilon}_0 = 0$$

FE

c) Correct solution

$$\varphi^{(m+1)} = \varphi^{(m)} + \delta^{(m)}$$

This iteration is found to, with suitable choice of quadrature nodes, inherit the stability of Forward / Backward Euler!

A useful way to think of SDC iteration is as a preconditioned iterative method to solve

$$\vec{\varphi} = \vec{\varphi_0} + \Delta t \overleftrightarrow{S} \vec{F}(\vec{\varphi})$$

where FE/BE is the preconditioner. The basic SDC method is based on fixed-point iteration

Properties for linear problems for sufficiently small Δt:

a) the SDC iteration converges

b) Each iteration increases the order of accuracy (in Δt) by 1, up to the maximal order of the quadrature rule.

So SDC is a way to get any order of accuracy we desire, without having to derive complicated RK formulas, for example. But it may be expensive (especially in memory for PDEs). The practical question is how much bigger of a time step we can take for the same accuracy as say RK.