P. Grohs
S. Hosseini
Ž. Kereta

Spring Term 2015

Numerical Analysis II

ETH Zürich
D-MATH

Problem Sheet 9

Problem 9.1 Stability Function of s-Stage Order s Explicit RK Method

Show that applying any *explicit* s-stage Runge-Kutta method of order s, with a stepsize h, to the model problem $\dot{y} = \lambda y$, where $\lambda \in \mathbb{C}$, yields

$$y_1 = \left(\sum_{j=0}^{s} \frac{z^j}{j!}\right) y_0, \quad z = h\lambda.$$

HINT: First show that y_1/y_0 is polynomial of degreee s in z and then use the Taylor expansion of the exact solution.

Solution: Equations for the stages of an explicit s-stage Runge-Kutta method applied to $\dot{y} = \lambda y$ are given by

$$k_i = \lambda\left(y_0 + h\sum_{j=1}^{i-1} a_{ij}k_j\right), \quad i = 1, \dots, s,$$

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i k_i,$$

where the coefficients a_{ij}, b_i, $i = 1, \dots, s$ and $1 \le j \le i-1$ are defined by the method.

Let use induction to show that we have $hk_i \in \mathcal{P}_i(z)$, $z := h\lambda$ for every $i = 1, \dots, s$.

For $s = 1$, we have

$$hk_1 = h\lambda y_0 = zy_0,$$

thus hk_1 is a polynomial of degree 1.

Assume now that $hk_j \in \mathcal{P}_j(z)$ for $1 \le j \le n$. It follows

$$\begin{aligned} hk_{n+1} &= h\lambda\left(y_0 + h\sum_{j=1}^{n} a_{n+1,j}k_j\right) \\ &= z\left(y_0 + h\sum_{j=1}^{n-1} a_{n+1,j}k_j + ha_{n+1,n}k_n\right). \end{aligned}$$

Hence, $hk_{n+1} \in \mathcal{P}_{n+1}(z)$ since the expression inside the parentheses is polynomial of degree n by induction. Furthermore, we see that

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i k_i$$

is another polynomial of degree s.

The exact solution of the differential equation can be written as

$$y(h) = y_0 \mathrm{e}^{h\lambda} = y_0 \sum_{i=0}^{\infty} \frac{z^i}{i!},$$

and, since the method is of order s, we have

$$\boldsymbol{\Phi}^h y - \boldsymbol{\Psi}^h y = \mathcal{O}(h^{s+1}).$$

This means that the first s coefficients of the polynomials coincide and taken together, we have thus shown that

$$y_1 = \left(\sum_{j=0}^{s} \frac{z^j}{j!} \right) y_0, \quad z = h\lambda.$$

Problem 9.2 Stability of Extrapolation Methods

Extrapolation is a valuable technique for the construction of single-step methods of higher order. Of particular interest is the extrapolation of the explicit Euler method whose stability properties we will now study.

(9.2a) Consider the autonomous initial value problem

$$y' = \lambda y \qquad y(0) = 1, \qquad \lambda \in \mathbb{C}. \tag{9.2.1}$$

Let h be the base stepsize. Show that after $n \in \mathbb{N}$ steps of the explicit Euler with step size $h_n = \frac{h}{n}$ the approximate solution is of the form

$$y(h) \approx y_n = \left(1 + \frac{h\lambda}{n} \right)^n.$$

Solution: We shall use induction. A single step of the explicit Euler with step size $\frac{h}{n}$ yields

$$y_1 = y_0 + h_n f(y_0) = y_0 + \frac{h}{n}\lambda y_0 = 1 + \frac{h\lambda}{n}.$$

Let $m \leq n - 1$. From the induction hypothesis

$$y_m = \left(1 + \frac{h\lambda}{n} \right)^m$$

it follows that

$$y_{m+1} = y_m + \frac{h}{n} f(y_m) = y_m + \frac{h}{n}\lambda y_m = \left(1 + \frac{h\lambda}{n} \right) y_m = \left(1 + \frac{h\lambda}{n} \right)^{m+1}.$$

Therefore, $y_n = \left(1 + \frac{h\lambda}{n}\right)^n$.

(9.2b) Show that the stability function $S(z)$ of the extrapolated Euler method after k steps with extrapolation sequence $\{n_i\}_{i=1}^k$ is given by the recursive formula

$$\begin{aligned} S_{i,1}(z) &= \left(1+\frac{z}{n_i}\right)^{n_i} && \text{for } i = 1,\dots,k, \\ S_{i,\ell}(z) &= S_{i,\ell-1}(z) + \frac{S_{i,\ell-1}(z) - S_{i-1,\ell-1}(z)}{\frac{n_i}{n_{i-\ell+1}} - 1} && \text{for } 2 \le \ell \le i, \\ S(z) &:= S_{k,k}(z). \end{aligned}$$

HINT: You can use the Aitken-Neville scheme [NUMODE, Eq. (2.4.5)], [NUMODE, Eq. (2.4.6)]. Why?

Solution: Extrapolation via the Aitken-Neville scheme is admissible because the explicit Euler has consistency order 1. Let us now recall that by Aitken-Neville we have

$$\begin{aligned} T_{i,1} &:= y_{h_i}(x_0 + h) \\ T_{i,\ell} &:= T_{i,\ell-1} + \frac{T_{i,\ell-1} - T_{i-1,\ell-1}}{\frac{n_i}{n_{i-\ell+1}} - 1} && \text{for } 2 \le \ell \le i. \end{aligned}$$

Notice that n_i steps of a method that uses stepsize $\frac{h}{n_i}$ yields the approximation at $t_0 + h$. Hence, $S_{i,1}(z) = \left(1+\frac{z}{n_i}\right)^{n_i}$ holds by subproblem (9.2a).

We will finish the proof by induction. Clearly, since $T_{i,l}$ is a linear combination of two methods, then its stability function is a linear combination of the stability functions of those two methods. In other words

$$\begin{aligned} T_{i,\ell} &= \left(1 + \frac{n_{i-\ell+1}}{n_i - n_{i-\ell+1}}\right) T_{i,\ell-1} + \frac{n_{i-\ell+1}}{n_i - n_{i-\ell+1}} T_{i-1,\ell-1} \\ &= \left(1 + \frac{n_{i-\ell+1}}{n_i - n_{i-\ell+1}}\right) S_{i,\ell-1}(z) y_0 + \frac{n_{i-\ell+1}}{n_i - n_{i-\ell+1}} S_{i-1,\ell-1}(z) y_0 \\ &= \left(S_{i,\ell-1}(z) + \frac{S_{i,\ell-1}(z) - S_{i-1,\ell-1}(z)}{\frac{n_i}{n_{i-\ell+1}} - 1}\right) y_0 \end{aligned}$$

Therefore, $S_{i,\ell}(z) = S_{i,\ell-1} + \frac{S_{i,\ell-1} - S_{i-1,\ell-1}}{\frac{n_i}{n_{i-\ell+1}} - 1}$, as was required.

(9.2c) Write the MATLAB function

```
function s = StabilityEval(z, n)
```

in which given `z` (a complex number) and a vector of integers `n` we compute `s`, the value of the stability function S, of the extrapolated Euler as defined in (9.2b), at the point `z`.

Solution:

Listing 9.1: Implementation of subproblem (9.2c)

```
1 function s = StabilityEval(z, n)
2
```

```
k=length(n);
S=(1+z./n).^n;

% In this implementation we will use the structure of the recursion,
% namely, the fact that recursion only uses the values in the current and
% previous row. This specific structure allows us to keep all of the
% important values in only a vector (instead of a matrix).
for l=2:k % Go over columns
    for i=2:k-l+2 % Go over rows
        % The recursion. Write over the preexisting value as we don't need
        % it any longer
         S(i-1)=S(i)+(S(i)-S(i-1))/(n(i+l-2)/n(i-1)-1);
    end
end

s=S(1);

% A straightforward implementation could look something like this
% Stemp=zeros(k,k);
% Stemp(:,1)=((1+z./n).^n)';
% for ll=2:k %for all columns
%     for ii=ll:k %for all lines
%
    Stemp(ii,ll)=Stemp(ii,ll-1)+(Stemp(ii,ll-1)-Stemp(ii-1,ll-1))...
%             /(n(ii)/n(ii-ll+1)-1);
%     end
% end
% Stemp(end,end);

end
```

(9.2d) Complete the MATLAB function `StabilityDomain.m`, which plots the stability domain of the extrapolated Euler method using 5 steps extrapolation steps (i.e. `n=1:5`). Comment on the stability properties of this method.

Solution: The method has a bounded stability domain, thus can't be A-stable.

Listing 9.2: Implementation of subproblem (9.2d)

```
function StabilityDomain

% Set the extrapolation sequence
n = 1:5;
```

```
5
6  % Build the grid using meshgrid
7  [X, Y] = meshgrid(-5:0.1:5);
8
9  % Generate complex mesh
10 Z = X + 1i*Y;
11
12 % Evaluate the stability function S on each point of the
       complex mesh
13 Sz = zeros(size(Z));
14 for ii = 1:length(Z)
15     for jj = 1:length(Z)
16         Sz(ii, jj) = StabilityEval(Z(ii, jj), n);
17     end
18 end
19
20 % Plotting
21 figure;
22 % call contourf() with contour lines abs(S) = [0 1 2 5] here:
23 [~, h] = contourf(X, Y, abs(Sz), [0 1 2 5]);
24 set(h,'ShowText','on','TextStep',get(h,'LevelStep'));
25 colormap(autumn);
26 colorbar
27
28 title('Stability domain of the extrapolated explicit Euler');
29 xlabel('Re');
30 ylabel('Im');
31 axis square;
32 grid on;
33
34 print -depsc StabilityDomain
35 end
```

(9.2e) Prove that the extrapolated Euler method is not A-stable for any extrapolation sequence.

Solution: We have previously seen (Lemma 2.3.19 of the lecture notes) that extrapolation methods that are based on explicit methods are also explicit Runge-Kutta methods. Since explicit Runge-Kutta methods have a bounded stability domain, they cannot be A-stable and the conclusion follows.

Published on 21 April 2015.

To be submitted by 28 April 2015.

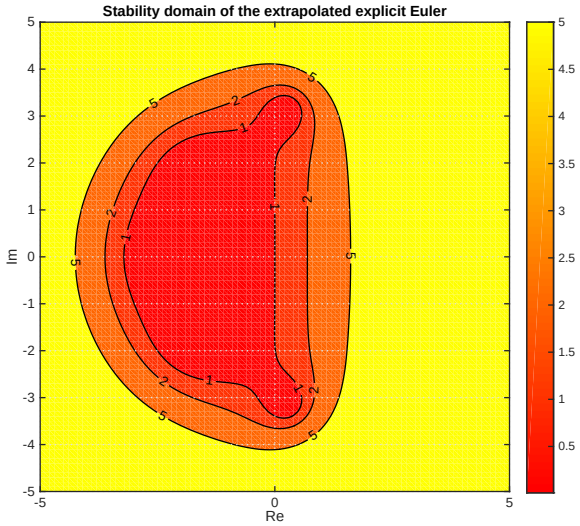

Figure 9.1: The stability domain as given by subproblem subproblem (9.2d)

References

[NUMODE] Lecture Slides for the course “Numerical Methods for Ordinary Differential Equations”, SVN revision # 63606.

Last modified on April 20, 2015