TANA09 Lecture 9

Extrapolation

- Estimates of error terms.
- Differentiation and Integration.

Adaptive methods

- An adaptive integration code

Extrapolation

Example Find eigen values λ such that

$$\begin{cases} u_{xx} - \lambda u = 0, \quad 0 < x < 1, \\ u(0) = 0 \text{ och } u'(1) + u(1) = 0, \end{cases}$$

has a solution $u \neq 0$.

Method Approximate the problem by a matrix $A \in \mathbb{R}^{n \times n}$ and compute the eigen values using `eig` in Matlab.

The dimension n of the matrix controls the accuracy. Larger n should give a better eigen value.

Solution Introduce a grid $0 = x_0 < x_1 < \cdots < x_n = 1$, with *step size* $h = x_i - x_{i-1}$.

Approximate the differential equation by

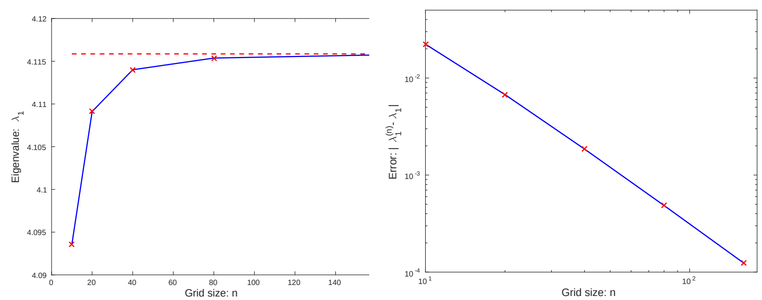
$$\frac{1}{h^2}\left(u(x_{i-1}) - 2u(x_i) + u(x_{i+1})\right) - \lambda u(x_i) = 0, \quad 1 \leq i \leq n-1.$$

and the boundary condition by

$$\frac{1}{2h}\left(3u(x_n) - 4u(x_{n-1}) + u(x_{n-2})\right) + u(x_n) = 0.$$

This gives us an eigen value problem $Au = \lambda u$, where A is $n \times n$.

Plot the first eigen value $\lambda_1(h)$ and the error $|\lambda_1(h) - \lambda_1|$.



Observation The error is a straight line if plotted in a log-log scale. What does this mean? How to exploit?

Example Collect the computed eigenvalues in a table

h	0.1	0.05	0.025	0.0125
$\lambda_1(h)$	4.0936	4.1091	4.1140	4.1154

A theoretical analysis shows that the truncation error can be estimated

$$R_T \approx Ch^2$$

where C is a constant. Use the table to estimate C and the error in the approximation $\lambda_1(h)$, for $h = 0.0125$.

> **Definition** A *numerical method* computes a value $F(h)$ where h is a *discretization parameter* that controls the accuracy in the result.

> **Definition** The numerical method is *convergent* if $F(h) \to F_0$, as $h \to 0$, where F_0 is the exact value.

This means that the method has a *truncation error* $R_T \to 0$ as $h \to 0$.

Example We want to compute the integral

$$I = \int_a^b f(x)dx$$

using the *Trapezoidal method*. We introduce a grid $\{x_i\}$, $h = x_{i+1} - x_i$, and use the formula

$$T(h) = h\left(\frac{f(x_1)}{2} + f(x_2) + \cdots + f(x_{n-1}) + \frac{f(x_n)}{2}\right).$$

> **Lemma** If $f(x)$ is continuous on $[a, b]$ then the Trapezoidal method is convergent.

Taylor expansions and extrapolation

> **Theorem** Suppose $F(h)$ has $(k+1)$ continuous derivatives. Then
>
> $$F(h) = F_0 + c_1 h + c_2 h^2 + \cdots + c_k h^k + \mathcal{O}(h^{k+1}).$$

Observation This means that the method has a *truncation error* that can be written as

$$R_T = c_1 h + c_2 h^2 + c_3 h^3 + \ldots$$

provided that a sufficient number of continuous derivatives exists.

Theorem If the function $f(x)$ has $(k+1)$ continuous derivatives then the *Trapezoidal method* computes an approximnation satisfying

$$T(h) = I + c_1h^2 + c_2h^4 + \cdots + c_kh^k + \mathcal{O}(h^{k+1}).$$

We only have even terms in the error expansion. The specific error expansion is known for many numerical methods.

Example We use the trapezoidal method and compute approximations of an integral using various step sizes.

h	0.2	0.1	0.05	0.025
$T(h)$	1.589339	1.577520	1.574243	1.573402

Use the table and estimate the error in the approximations for $h = 0.1, 0.05$ and 0.025.

Also eliminate the first error term and estimate the error in the new approximations.

In Matlab

```
>> Rt=( T(1:3)-T(2:4) )/3
 Rt =
  0.003940   0.001092   0.000281

>> T2 = T(2:4)-Rt
 T2 =
   1.573580   1.573151   1.573119

>> Rt2 = ( T2(1:2)-T2(2:3) )/15
 Rt2 = 1.0e-4 *
    0.2864   0.02111
```

Observation We see that the error in the approximation $T_2(h)$ decreaes by a factor $13.5 \approx 16$ when h decrease by a factor of 2.

Example Compute the integral

$$I = \int_0^{0.1} \sqrt{x}e^{-x}dx,$$

using the trapezoidal method. We obtain

h	0.05	0.025	0.0125	0.00625
$T(h)$	0.017788	0.019101	0.019586	0.019762

Estimating the truncation error using the formula $R_T(h) = (T(h) - T(h/2))/3$ gives

h	0.025	0.0125	0.00625
$R_T(h)$	-0.000438	-0.000162	-0.0000587

Here the error **can not** be written as ch^2. Extrapolation does not work.

Richardsson Extrapolation

Theorem Suppose

$$F(h) = F_0 + c_1 h^p + c_2 h^{p+1} + \mathcal{O}(h^{p+2}).$$

Then

$$F_1(h) = F(h) - \frac{1}{q^p - 1}(F(qh) - F(h))$$

$$= F_0 + \tilde{c}_2 h^{p+1} + \mathcal{O}(h^{p+2}).$$

Remark We have eliminated one term from the error expansion. This can drastically reduce the error for a numerical method. Important to know the error expansions.

Example Comppute a derivative $f'(1)$ using a centered differens formula

$$D_0(h) = \frac{1}{2h}(f(x+h) - f(x-h)).$$

We use two different step sizes and obtain

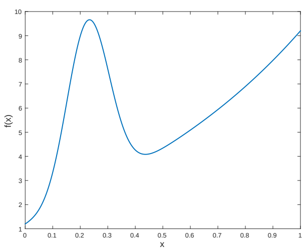
$$D_0(0.3) = 1.25657, \qquad D_0(0.1) = 1.28420$$

Estimate the error in the value $D_0(0.1)$. Improve the accuracy using extrapolation.

Observation Lower order methods and extrapolation is often simpler to use than deriving higher order methods.

Adaptive methods

Example We want to compute the integral of a function $f(x)$ with a given maximumn error ϵ. First plot the function



Observation The second derivative is of different order of magnitude on different subintervals. Adapt the step size h according to this fact.

Method Let $h = b - a$, and $c = (a+b)/2$. Compute

$$I_h = \frac{h}{2}(f(a) + f(b)), \text{ and } I_{h/2} = \frac{h}{2}\left(\frac{f(a)}{2} + f(c) + \frac{f(b)}{2}\right).$$

Then

$$I = \int_a^b f(x)dx = I_{h/2} + R_T, \quad R_T \approx (I_h - I_{h/2})/3.$$

If $|R_T| < \epsilon$ we are done. Otherwise apply the method recursively and compute

$$I_1 = \int_a^c f(x)dx \quad \text{and} \quad I_2 = \int_c^b f(x)dx,$$

with an error at most $\epsilon/2$. Gives $I = I_1 + I_2$ with error at most ϵ.

```
function [I,Rt]=AdaptInt( fun,a,b,tol,fa,fb )
  if nargin<5,
     fa=feval(fun,a);fb=feval(fun,b);
  end
  h=b-a;c=(a+b)/2;fc=feval( fun , c);
  I1=h*(fa+fb)/2;I2=h*(fa/2+fc+fb/2)/2;
  Rt=abs(I1-I2)/3;
  if Rt<tol
     I=I2;
  else
     [I1,Rt1]=AdaptInt( fun,a,c,tol/2,fa,fc );
     [I2,Rt2]=AdaptInt( fun,c,b,tol/2,fc,fb );
     I=I1+I2;Rt=Rt1+Rt2;
  end
end
```

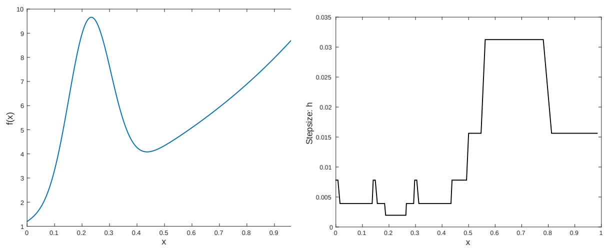
Observation One function evaluation in each recursion step.

In Matlab we write

```
>> f=@(x) <expression for f(x)>;
>> [I,Rt]=AdaptInt( f , 0 , 1 , 10^-3 )
 I =
   5.960159817775207
 Rt =
   5.342169304992216e-04
```

Number of function evaluations is 159.

If we instead require `tol`$= 10^{-5}$ we need 1569 function evaluations.



The function $f(x)$ and the step size h for the case `tol`$= 10^{-3}$.

In Matlab `integral` computes integrals with adaptive step size choice.

Summary

- Extrapolation is a very powerful technique for estimating errors and improving approximations.
- We need to know the error expansion for different numerical methods.
- Offers a way to check if a computer program is implemented correctly.
- Adaptive methods adjust to the problem at hand. The step size is chosen so a specific tolerance is achived.