Exercises

$$\int_0^1 \cos(x^2)\exp(-x)dx$$

$$\int_0^1 \sqrt{x}\cos(x^2)\exp(-x)dx$$

$$\int_0^1 \frac{1}{\sqrt{x}}\cos(x^2)\exp(-x)dx$$

$$\int_0^1 1000\exp(-1/x)\exp(1/(1-x)dx$$

DE (double exponential) rule:

$$x = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)\tanh(c\sinh t), \qquad x \in [a,b] \to t \in [-\infty, \infty]$$

$$\frac{dx}{dt} = \frac{1}{2}(b-a)\,\mathrm{sech}^2(c\sinh t)c\cosh t \sim \exp(-c\exp|t|) \quad \text{as} \quad |t| \to \infty$$

Similar handling of trimming error and discretization error:

$$h \sim \frac{\log(2\pi N w/c)}{N}, \qquad \epsilon \sim e^{-kN/\log N}$$

Today

- Some practicalities with using DE rule
- Wrap up of the course until now
    - System of linear equations
    - System of non-linear equations
    - Numerical integration
- Discussion of the remaining subjects
    - Ordinary differential equations
    - Partial differential equations
- Half time course evaluation
- Exercises using DE Rule with Ole

Practicalities with using DE rule

$$x = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)\tanh(c\sinh t), \qquad x \in [a,b] \to t \in [-\infty, \infty]$$

$$\frac{dx}{dt} = \frac{1}{2}(b-a)\,\mathrm{sech}^2(c\sinh t)\,c\cosh t \sim \exp(-c\exp|t|) \quad \text{as} \quad |t| \to \infty$$

Avoiding numerical overflow of $\mathrm{sech}^2(c\sinh t)$

$$q = e^{-2\sinh t} \tag{4.5.10}$$

(we take $c = 1$ for simplicity) so that

$$\frac{dx}{dt} = 2(b-a)\frac{q}{(1+q)^2}\cosh t \tag{4.5.11}$$

For large positive t, q just underflows harmlessly to zero. Negative t is handled by using the symmetry of the trapezoidal rule about the midpoint of the interval. We write

$$\begin{aligned} I &\simeq h \sum_{j=-N}^{N} f(x_j) \left.\frac{dx}{dt}\right|_j \\ &= h\left\{ f[(a+b)/2] \left.\frac{dx}{dt}\right|_0 + \sum_{j=1}^{N}[f(a+\delta_j) + f(b-\delta_j)] \left.\frac{dx}{dt}\right|_j \right\} \end{aligned} \tag{4.5.12}$$

where

$$\delta = b - x = (b-a)\frac{q}{1+q} \tag{4.5.13}$$

$$
\begin{aligned}
I &\simeq h \sum_{j=-N}^{N} f(x_j) \left.\frac{dx}{dt}\right|_j \\
&= h\left\{ f[(a+b)/2] \left.\frac{dx}{dt}\right|_0 + \sum_{j=1}^{N} [f(a+\delta_j) + f(b-\delta_j)] \left.\frac{dx}{dt}\right|_j \right\}
\end{aligned}
\qquad (4.5.12)
$$

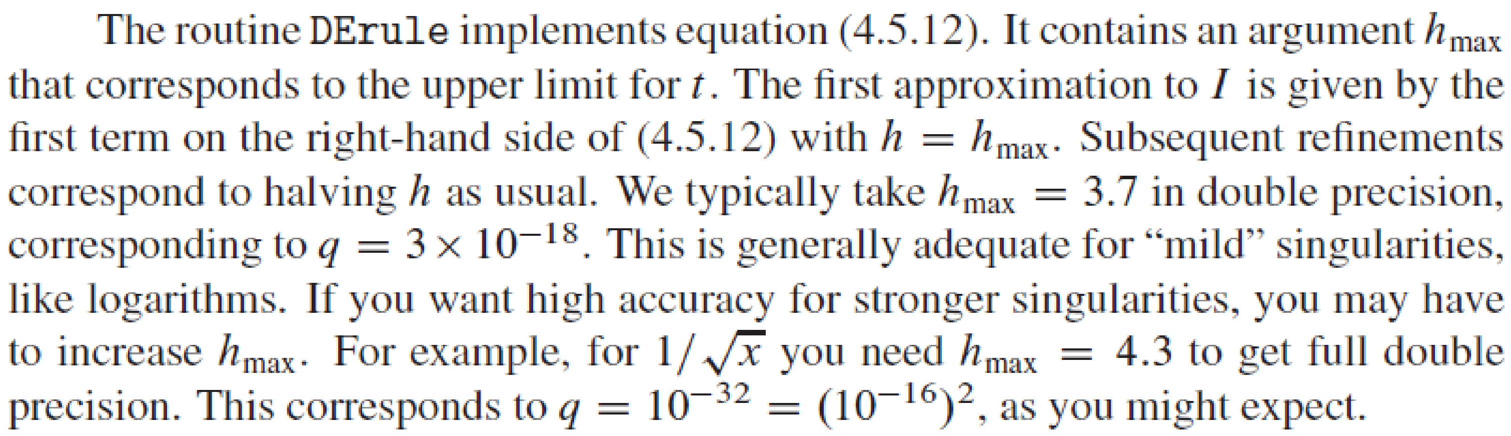
where

$$
\delta = b - x = (b-a)\frac{q}{1+q} \qquad (4.5.13)
$$

A second possible problem is that cancellation errors in computing $a+\delta$ or $b-\delta$ can cause the computed value of $f(x)$ to blow up near the endpoint singularities. To handle this, you should code the function $f(x)$ as a function of two arguments, $f(x,\delta)$. Then compute the singular part using δ directly. For example, code the function $x^{-\alpha}(1-x)^{-\beta}$ as $\delta^{-\alpha}(1-x)^{-\beta}$ near $x=0$ and $x^{-\alpha}\delta^{-\beta}$ near $x=1$. (See §6.10 for another example of a $f(x,\delta)$.) Accordingly, the routine DErule below expects the function f to have two arguments. If your function has no singularities, or the singularities are "mild" (e.g., no worse than logarithmic), you can ignore δ when coding $f(x,\delta)$ and code it as if it were just $f(x)$.

You need to implement this

```
q=exp(-2.0*sinh(t));
del=(b-a)*q/(1.0+q);
fact=q/SQR(1.0+q)*cosh(t);
sum += fact*(func(a+del,del)+func(b-del,del));
```



The routine `DErule` implements equation (4.5.12). It contains an argument $h_{\max}$ that corresponds to the upper limit for t. The first approximation to I is given by the first term on the right-hand side of (4.5.12) with $h = h_{\max}$. Subsequent refinements correspond to halving h as usual. We typically take $h_{\max} = 3.7$ in double precision, corresponding to $q = 3 \times 10^{-18}$. This is generally adequate for "mild" singularities, like logarithms. If you want high accuracy for stronger singularities, you may have to increase $h_{\max}$. For example, for $1/\sqrt{x}$ you need $h_{\max} = 4.3$ to get full double precision. This corresponds to $q = 10^{-32} = (10^{-16})^2$, as you might expect.

You need to select hmax

Solutions of systems of linear equations (NR Chapter 2):

Gaussian-elimination with back substitution
I expect that you can outline the algorithm to generate an upper triangular matrix and that you know how to perform pivoting. Furthermore, you should know the computational complexity of the algorithm as function of the matrix size.

LU-decomposition
You must be able to explain the relation to Gaussian elimination and know the structure of L and U and how this structure is exploited when solving systems of linear equations. You must also be able to show how to use the software routine from NR.

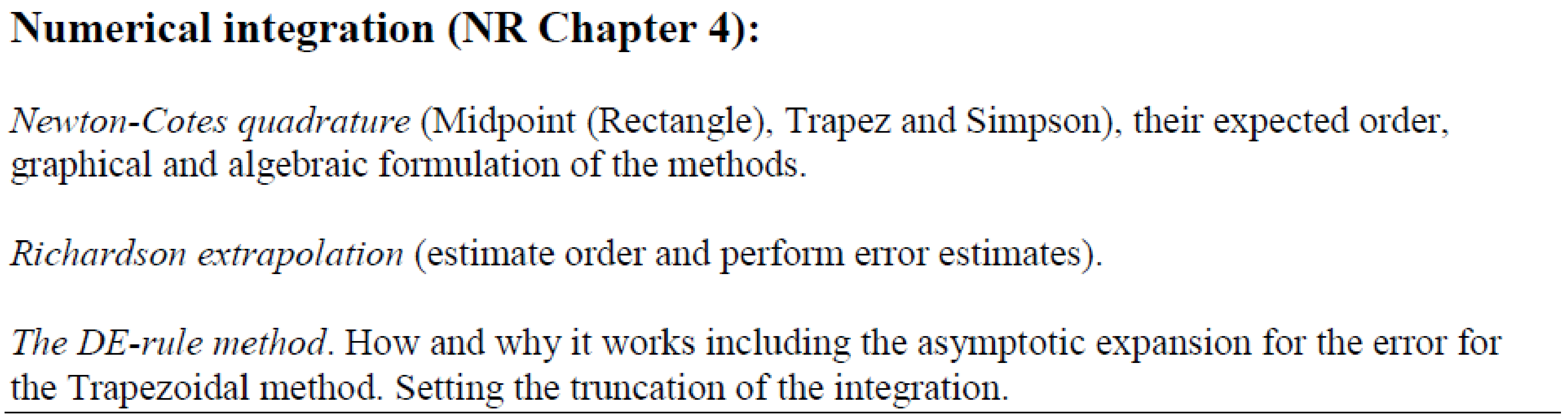
Cholesky decomposition
You must know what a symmetric and positive definite matrix is as these are a condition for using the method. Furthermore, you must be able to explain why to use Cholesky instead of LU decomposition whenever the conditions are satisfied. You must also be able to show how to use the software routine from NR.

Singular Value Decomposition (SVD)
You must know the structure and size of U,W and V for a MxN matrix A. You must know the advantages and disadvantages of applying SVD for solving an MxN systems of linear equations where M>=N. In particular you must be able to explain why SVD are more robust than using the Normal equations (A^TAx=A^Tb) near singularities. You must also be able to show how to use the software routine from NR.

Solutions of systems of non-linear equations (NR Chapter 9):

One equation in one unknown. You must be able to outline Bisection, Secant, Regula Falsi (False Position), Newton and Ridder and discuss the advantages and disadvantages in connection with robustness, convergence speed and convergence order. You must also be able to show how to use the software routine from NR.

Systems of non-linear equations : The Jacobian matrix computed analytically and with finite differences. Newtons method. The “globally convergent” Newton’s method including the line search and backtracking method for ensuring convergence by controlling that F(x).F(x) is decreasing. You must also be able to show how to use the software routine from NR.



Numerical integration (NR Chapter 4):

Newton-Cotes quadrature (Midpoint (Rectangle), Trapez and Simpson), their expected order, graphical and algebraic formulation of the methods.

Richardson extrapolation (estimate order and perform error estimates).

The DE-rule method. How and why it works including the asymptotic expansion for the error for the Trapezoidal method. Setting the truncation of the integration.

Remaining part (next two slides) subject to minor changes…

Ordinary differential equations:

Initial value problems

Fixed stepsize methods: Euler, Reverse Euler, Trapez, Midtpoint, Leap-frog. You must be able to make a graphical and/or algebraic outline of how each of the methods works. You must know the expected order of each of these methods. In addition you should know that the 4th order Runge-Kutta method (rk4) exists. Richardson extrapolation for these methods. Stability regions for the methods.

Adaptive stepsize methods: StepperDopr5, StepperDopr853, StepperRoss. You should be able to very qualitatively outline how the stepsize is controlled. You should be able to explain what the content of the needed input to the NR routines is.

Stability and convergence: Local vs. Global order, stability and stability regions.

Boundary value problems

Model problem (y''(x)=F(y'(x),y(x),x); y(a)=alpha; y(b)=beta), shooting method, method based on solving a set of nonlinear equations. Solving tridiagonal systems of linear equations

Elliptical partial differential equations

Model problem (Poisson’s equation on the unit square), numerical solution to the model problem, Dirichlet (fixed position) and Neumann (free) boundary conditions. You should know the band-structure of the related system of linear equations and how it should be solved efficiently. Richardson extrapolation.