Practice Sessions
Astrophysical Simulations

Part 4b: Differential equation
(Solution)

Master of Science in Physics and Astronomy
2018-2019
Peter Camps
peter.camps@ugent.be
S9, 1st floor, office 110.014

GHENT UNIVERSITY

Assignment: solve differential equation

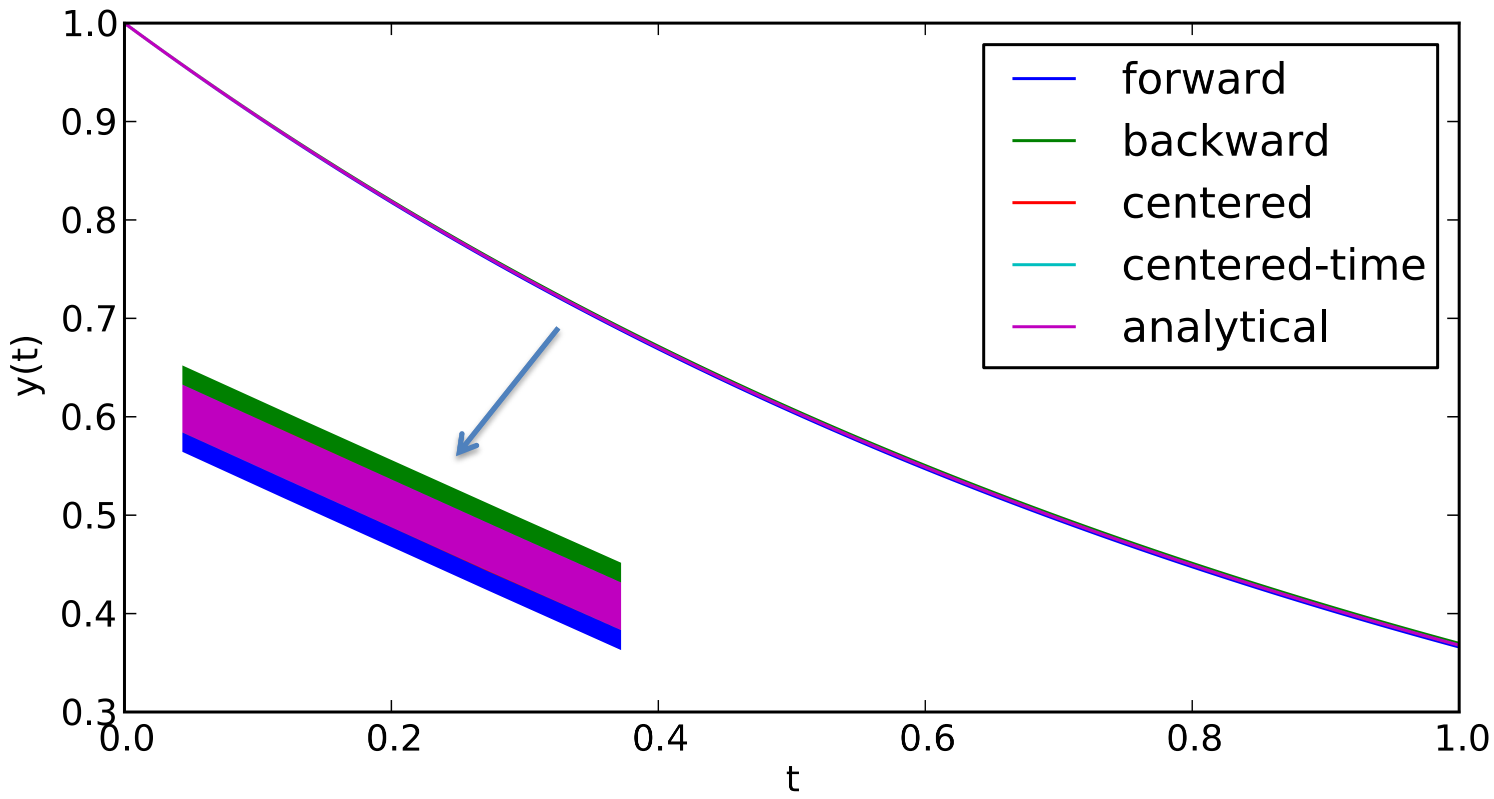
Part 1

- Implement the forward, backward, centered and centered-time finite difference methods to numerically solve the ODE
  $$\dot{y} = -y \qquad \text{with } y(0)=1$$
  for y(t) in the interval $t \in [\,0,1\,]$ using time step h=0.01
- Plot the solutions and the errors relative to the analytical solution, in function of t

Part 2

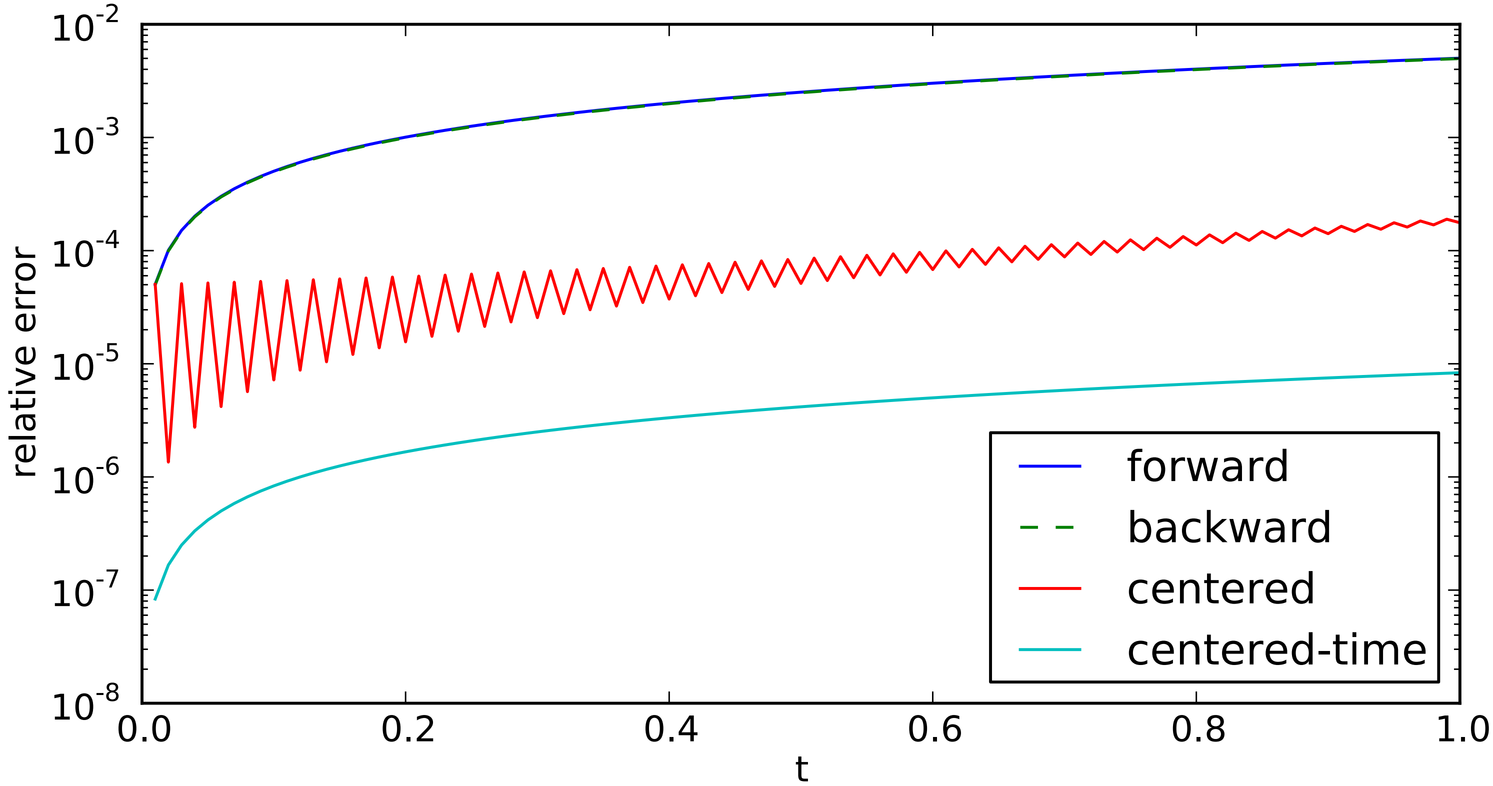
- Adjust (a copy of) your program to use a sequence of time steps $h = 1, 0.1, 0.01, \dots 10^{-9}$ (do not output all the computed points!)
- Plot the relative error at the end of the interval (t_{end} =1) in function of h, and interpret the results

Time evolution of y(t) for h=0.01



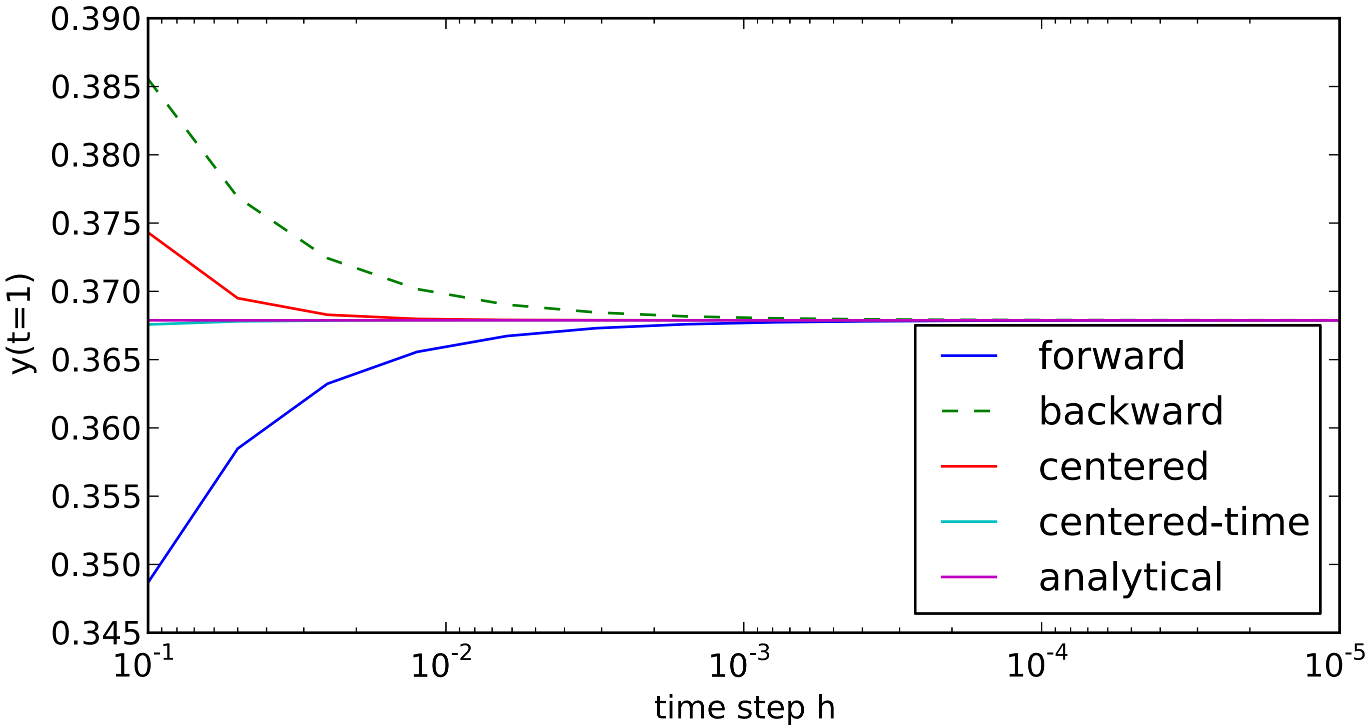
- All implemented methods produce results near the analytical solution
- The forward and backward methods have “opposite” errors
- This plot does not allow proper evaluation of the results

Error on time evolution of y(t) for h=0.01



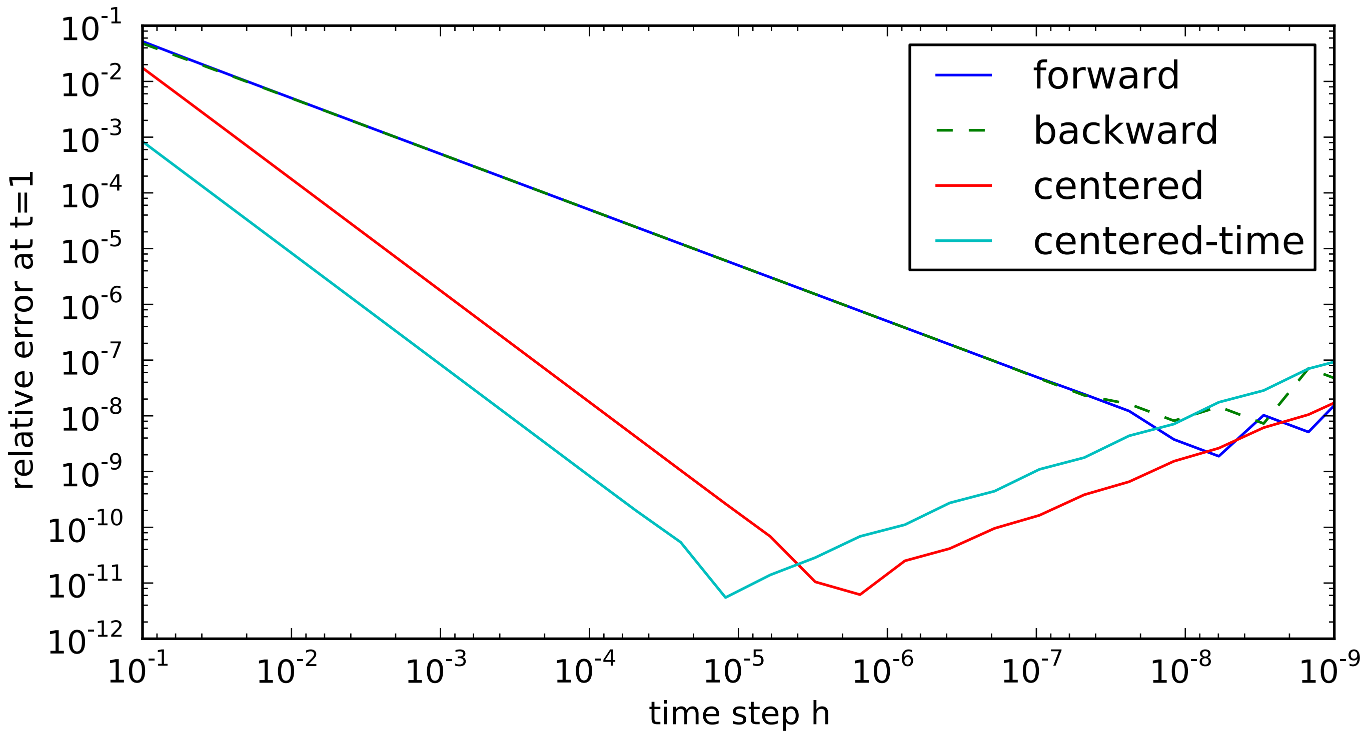
- At the end of the interval (i.e. after 100 steps) the relative error for the forward and backward difference methods is substantial (0.5%)
- The centered-time difference method far outperforms the other methods
  - » Almost three orders of magnitudes better than forward and backward
  - » Even a lot better than centered method which has the same order

Error on y(t=1) in function of h



- The forward and backward methods produce errors with opposite sign
- The centered and centered-time difference methods produce better results for the same time step
- The linear scale of the plot does not allow evaluating the results for small h

Error on y(t=1) in function of h



- The centered and centered-time difference methods produce far better results for the same time step
- Choosing the time step too small has an adverse effect on the result (in addition to increasing the runtime); for the centered-time method, the optimal time step value is $h \approx 10^{-5}$ achieving an accuracy of about 10^{-11}

Finite difference functions

Scheme	Order	Equation
Forward difference	$O(h^2)$	$y_{n+1} = y_n + hg_n$
Backward difference	$O(h^2)$	$y_{n+1} = y_n + hg_{n+1}$
Centered difference	$O(h^3)$	$y_{n+1} = y_{n-1} + 2hg_n$
Centered time difference	$O(h^3)$	$y_{n+1} = y_n + h(g_n + g_{n+1})/2$

Substitute $g_n = - y_n$ and rearrange if needed

```
double forward(double y_n, double h) {
    return (1. - h) * y_n;
}
double backward(double y_n, double h) {
    return y_n / (1. + h);
}
double centered(double y_nm1, double y_n, double h) {
    return y_nm1 - 2.*h * y_n;
}
double centered_time(double y_n, double h) {
    return y_n * (1. - 0.5*h) / (1. + 0.5*h);
}
```

Initial conditions

```
const double t0 = 0.;
const double y0 = 1.;
const double t_end = 1.;
const double h = 0.01;

ofstream outfile("ode1.txt");
outfile << setprecision(15);

double t = t0;
double forw = y0;
double back = y0;
double cent = y0;
double cntm = y0;

outfile << t << ' ' << y_anal(t) << ' '
        << forw << ' ' << back << ' '
        << cent << ' ' << cntm << '\n';
```

Setup constants defining the problem

Open the output file and set high precision because we will be evaluating small differences between numbers

Initialize variables for the time and for each method to the initial conditions

Output the initial conditions as the first line in the file

The first time step

The first time step is performed outside of the loop because of the special needs of the centered difference method; we need to make an initial guess of y_1 because the method calculates y_{n+1} from y_{n-1} and y_n

```
t += h;
forw = forward(forw, h);
back = backward(back, h);
double cent_prev = cent;
cent = forward(cent, h);
cntm = centered_time(cntm, h);

outfile << t << ' ' << y_anal(t) << ' '
        << forw << ' ' << back << ' '
        << cent << ' ' << cntm << '\n';
```

Remember the y_{n-1} value for the centered difference method

Use forward to make an initial step for the centered difference method

Output the results for the first time step

The time loop

```
for (t += h; t < t_end+h/2.; t += h) {

    forw = forward(forw, h);
    back = backward(back, h);
    double temp = cent;
    cent = centered(cent_prev, cent, h);
    cent_prev = temp;
    cntm = centered_time(cntm, h);

    outfile << t << ' ' << y_anal(t) << ' '
            << forw << ' ' << back << ' '
            << cent << ' ' << cntm << '\n';
}

outfile.close();
```

Loop over time with steps h; ensure that we go just beyond t_{end}

Remember the y_{n-1} value for the centered difference method

Output the results for each time step

Close the output file after the loop has completed

Adjusting the code for looping over h

```
double h = 0.1;
while (h >= 5e-10) {
    cout << "Starting time step " << h << endl;

    . . .

    outfile << h << ' ' << y_anal(t-h) << ' '
            << forw << ' ' << back << ' '
            << cent << ' ' << cntm << '\n';

    h /= 2;
}
```

Loop over a sequence of time steps h

Here goes the previous code, with all output removed

Output the results for each value of h

Use t-h because the for loop has incremented the loop variable by one extra step

Calculate the next time step h (spaced evenly in logarithmic space)

Questions?