This lab explores the use of multistep methods of the form

$$\sum_{m=0}^{s} a_m y_{n+m} = h \sum_{m=0}^{s} b_m f(t_{n+m}, y_{n+m}) \quad \text{where} \quad n = 0, 1, \ldots.$$

Such an s-step method uses information about the previous s time steps to obtain an approximation for y_{n+s}. From a practical point of view, one difficulty is getting started. Given the differential equation

$$y' = f(t, y) \qquad \text{such that} \qquad y(t_0) = y_0$$

how does one obtain the values for y_1, $y_2, \ldots,$ y_{s-1} so that y_s can subsequently be approximated using a multistep method?

One might consider applying Euler's method to approximate y_1, a 2-step method involving y_0 and y_1 to approximate y_2, a 3-step method involving y_0, y_1 and y_2 to approximate y_3 and so forth until enough time history is built up to use the s-step method for the rest of the computation. The difficulty with such an idea is the low order of the initial Euler step limits the order of convergence for the entire method.

Consider an interval $[t_0, T]$ on which one wants to approximate a solution using N time steps of size $h = (T - t_0)/N$. Let $e_n = y_n - y(t_n)$ where y_n approximates the exact solution $y(t_n)$. If Euler's method is used for the first step, then

$$\|e_1\| \leq (1 + h\lambda)\|e_0\| + ch^2 = ch^2.$$

Here we have assumed there is no rounding error in the representation of the initial condition $y(t_0) = y_0$. Unfortunately, no matter how accurate the remaining steps are, the error e_1 will propagate such that the total error is at best

$$E_N = \max\{\, \|e_n\| : n = 0, \ldots, N \,\} = \mathcal{O}(h^2).$$

Since one of the main uses of multistep methods is to obtain highly accurate approximations in cases where evaluating f is computationally expensive, such a restriction on the order of convergence is a problem.

The usual way to overcome this problem is to employ a less efficient method of sufficiently high order to compute the needed time history and then finish the computation using the multistep method. One such method has already been considered: the Taylor method. More commonly Runge–Kutta methods are used to start a multistep method.

In this lab we gain experience with multistep methods without having to worry about how to generate the time history needed to get them started by considering differential equations for which the exact solution is known. In this case the values $y_1, y_2, \ldots, y_{s-1}$ needed to begin the multistep method may be obtained from the exact solution by setting

$$y_1 = y(t_1), \qquad y_2 = y(t_2), \qquad \ldots, \qquad y_{s-1} = y(t_{s-1}).$$

We emphasize that the above way of starting a multistep method is of little practical use and intended only for the purpose of directly studying the method on its own. In particular, if the exact solution were known, none of these numerical methods would be needed in the first place.

Algebraic Techniques

Please create a working directory called `lab02` to hold your work for this lab. This section derives the s-step Adams–Bashforth and Nystrom methods using the polynomials

$$\rho(w) = \sum_{m=0}^{s} a_m w^m \qquad \text{and} \qquad \sigma(w) = \sum_{m=0}^{s} b_m w^m$$

along with the theorem that corresponding multistep method is of order p provided $\rho(w) - \sigma(w) \log w = \mathcal{O}(|w-1|^{p+1})$.

Since the Adams–Bashforth and Nystrom methods are both explicit, the Dahlquist first barrier implies $p = s$. For the Adams–Bashforth take $\rho(w) = w^{s-1}(w-1)$ and then find the polynomial $\sigma(w)$ such that

$$\sigma(w) = \frac{\rho(w)}{\log w} + \mathcal{O}\big(|w-1|^p\big).$$

This may be done with the Julia `TaylorSeries` package for $s = 3$ as follows:

```
julia> using TaylorSeries

julia> s=3
3
```

```
julia> rho(w)=w^(s-1)*(w-1)
rho (generic function with 1 method)

julia> t=Taylor1(s)
 1.0 t + O(t⁴)

julia> sigma=rho(t+1)/log(t+1)
 1.0 + 2.5 t + 1.9166666666666667 t² + O(t³)

julia> z=evaluate(sigma,[t-1])[1]
 0.41666666666666674 - 1.3333333333333335 t
 + 1.9166666666666667 t² + O(t³)

julia> b=z.coeffs
3-element Vector{Float64}:
  0.41666666666666674
 -1.3333333333333335
  1.9166666666666667
```

Observing that

$$\tfrac{5}{12} \approx 0.41666666666666674, \qquad -\tfrac{4}{3} \approx -1.3333333333333335$$

$$\text{and} \qquad \tfrac{23}{12} \approx 1.9166666666666667$$

verifies the third-order method

$$y_{n+1} = y_n + h\big\{\tfrac{5}{12} f(t_{n-2}, y_{n-2}) - \tfrac{4}{3} f(t_{n-1}, y_{n-1}) + \tfrac{23}{12} f(t_n, y_n)\big\}.$$

We remark that the above expression has been written in an explicit form that makes it easier to program.

The three-step Nystrom method can be obtained in a similar way by setting $\rho(w) = w^{s-2}(w^2-1)$ and then solving for σ. In this case one obtains

$$y_{n+1} = y_{n-1} + h\big\{\texttt{b[1]} f(t_{n-2}, y_{n-2}) + \texttt{b[2]} f(t_{n-1}, y_{n-1}) + \texttt{b[3]} f(t_n, y_n)\big\}.$$

where `b=z.coeffs` are the computed coefficients for the polynomial σ. Note $b_m =$ `b[m+1]` since array indices in Julia start at 1 and have to be shifted.

The first thing to turn in for this lab is a program called `algebra.jl` which computes the coefficients for σ corresponding to the 3-step Nystrom scheme along with the output of that program.

Numerical Experiments

The second part of this lab is a numerical test of the 3-step Nystrom method. What needs to be done shall be described in terms of an example which performs a similar set of steps involving the Adams–Bashforth method.

Theoretically, the 3-step Adams–Bashforth and Nystrom methods both approximate the exact solution to order $\mathcal{O}(h^3)$. This means there is a constant c such that the total error

$$E_N = \max\left\{\|y_n - y(t_n)\| : n = 0, \ldots, N\right\} \leq ch^3 \quad \text{as} \quad h \to 0.$$

We now compute the approximations y_n for different values of N and verify the rate of convergence.

Recall the linear differential equation

$$y' + 3y = \sin t \qquad \text{and} \qquad y(0) = 7$$

with exact solution

$$y(t) = 7.1e^{-3t} + 0.3\sin t - 0.1\cos t.$$

Note that this is the same differential equation considered in the previous lab on Euler's method.

Test the 3-step methods derived above using eight steps to approximate the solution $y(t)$ on the interval $[0, 1]$. This description in this lab provides a walk though for the Adams–Bashforth method. Please modify it for the Nystrom method before turning your work in.

Create a file called `numerics.jl`. After defining `f`, `y`, `T`, `N` and `h`, create two arrays `tn` and `yn` to hold the results of the calculation along with a third array `fn` to remember the values of $f(t_n, y_n)$ so previous values of f don't have to be recomputed. The reuse of previously computed values is one of the reasons multistep methods are useful when f is expensive.

Since this is a 3-step method, initialize the first three values of `tn`, `yn` and `fn` using the exact solution as

$$\begin{array}{llll} \texttt{tn[1]} = t_0, & \texttt{yn[1]} = y(t_0) & \text{and} & \texttt{fn[1]} = f(t_0, y(t_0)), \\ \texttt{tn[2]} = t_1, & \texttt{yn[2]} = y(t_1) & \text{and} & \texttt{fn[2]} = f(t_1, y(t_1)), \\ \texttt{tn[3]} = t_2, & \texttt{yn[3]} = y(t_2) & \text{and} & \texttt{fn[3]} = f(t_2, y(t_2)). \end{array}$$

Again note the offset by one for the index used in Julia compared to the mathematics. Finally make a loop to compute `tn` and `yn`.

The Julia code to do all of the above for the 3-step Adams–Bashforth method should look like

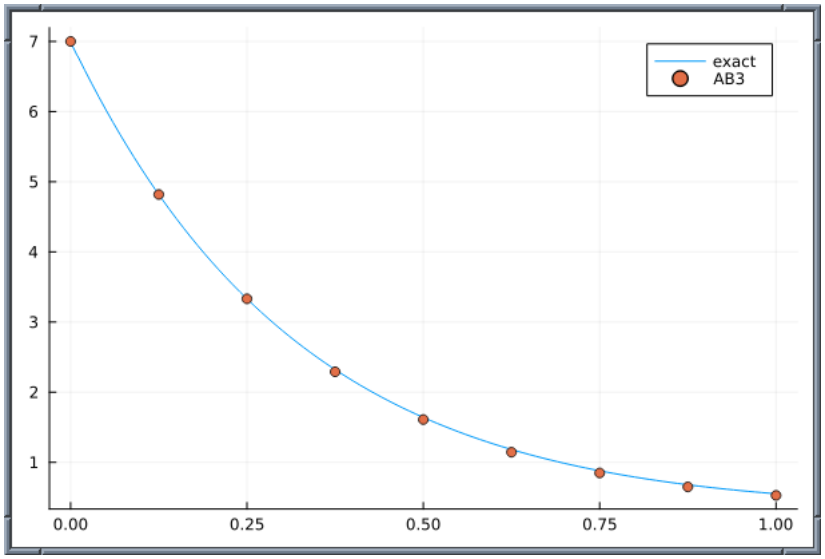
```
f(t,y)=sin(t)-3*y
y(s)=7.1*exp(-3*s)+0.3*sin(s)-0.1*cos(s)
t0=0
y0=7

T=1
N=8
h=(T-t0)/N
tn=zeros(N+1)
yn=zeros(N+1)
fn=zeros(N+1)

for n=0:2
    tn[n+1]=t0+h*n
    yn[n+1]=y(tn[n+1])
    fn[n+1]=f(tn[n+1],yn[n+1])
end
for n=3:N
    tn[n+1]=t0+h*n
    yn[n+1]=yn[n]+h*(5/12*fn[n-2]-4/3*fn[n-1]+23/12*fn[n])
    fn[n+1]=f(tn[n+1],yn[n+1])
end
```

Run the code by typing `include("numerics.jl")` in the REPL.

At this point it is useful to visualize the solution and its approximation. This can be done interactively using the `Plots` package. To obtain



enter the following into the REPL:

```
julia> using Plots

julia> ts=0:0.01:1
0.0:0.01:1.0

julia> plot(ts,y.(ts),label="exact")

julia> scatter!(tn,yn,label="AB3")
```

Note that the approximation is visually indistinguishable from the exact solution. We will check how the error depends on the step size shortly. Before that, however, modify the program to use the Nystrom method instead of Adams–Bashforth. Change the legend on the graph so the approximation given by the scatter plot is labeled `Nystrom3`. Then type `savefig("nystrom.pdf")` to save the final graph to a PDF as part of the work to turn in for this lab.

Study of Convergence

So far we've derived the Nystrom method using algebraic techniques and used eight time steps of that method to compute an approximation of the differential equation. As with the Euler method, next add an additional loop to compute the error E_N for different values of N.

After some changes we arrive at

```
f(t,y)=sin(t)-3*y
y(s)=7.1*exp(-3*s)+0.3*sin(s)-0.1*cos(s)
t0=0
y0=7

T=1
J=10
EN=zeros(J)
HN=zeros(J)
for j=1:J
    N=2^(j+2)
    h=(T-t0)/N
    tn=zeros(N+1)
    yn=zeros(N+1)
    fn=zeros(N+1)

    for n=0:2
        tn[n+1]=t0+h*n
        yn[n+1]=y(tn[n+1])
        fn[n+1]=f(tn[n+1],yn[n+1])
    end
    for n=3:N
        tn[n+1]=t0+h*n
        yn[n+1]=yn[n]+h*(5/12*fn[n-2]-4/3*fn[n-1]+23/12*fn[n])
        fn[n+1]=f(tn[n+1],yn[n+1])
    end
    HN[j]=h
    EN[j]=maximum(abs.(yn-y.(tn)))
end

using Plots
scatter(HN,EN,scale=:log10,
    legend=:topleft,label="EN")
plot!(HN,25*HN.^3,label="25*h^3")
```

Note how the error E_N parallels the line ch^3. This verifies the method is order $\mathcal{O}(h^3)$. The value of c was found to be 25 for the Adams–Bashforth method using guess and check. Now, change the code so it uses the Nystrom method, adjust c if necessary and type `savefig("error.pdf")` in the REPL to save the error graph.

Although both methods are third order, it is interesting to compare the value of c between the Adams–Bashforth and Nystrom methods. This may be done for extra credit by performing a least squares fit to find c for both the Adams–Bashforth and Nystrom methods. Is c very much different in either case? Please upload your extra credit work under `lab02ec` on the course management system.

Submitting Your Work

Three things should be uploaded for grading:

- A PDF file containing `algebra.jl` and its output.
- The graph `nystrom.pdf` of the Nystrom approximation.
- The graph `error.pdf` for the error analysis.

The files `nystrom.pdf` and `error.pdf` have already been created and should be in the `lab02` subdirectory. The only thing left is to convert `algebra.jl` and its output into a PDF file for upload. In the lab the commands

```
$ julia algebra.jl >algebra.out
$ j2pdf -o algebra.pdf algebra.jl algebra.out
```

may be used to produce a file `algebra.pdf` suitable for uploading. You may check your submission using `evince` to view the PDF files.

Before leaving don't forget to close the applications open on your desktop and logout. Exit the Julia REPL by typing ⟨ctrl⟩-`d` and then ⟨ctrl⟩-`d` again to close the terminal. The editor has a menu at the top. If using one of the lab computers, please reboot it into Microsoft Windows.