Math 467/667: Homework 1 Solutions

1. [Iserles 1.1] Apply the method of proof of Theorems 1.1 and 1.2 to prove the implicit midpoint rule (1.12) is second order and convergent.

The implicit midpoint rule is

$$y_{n+1} = y_n + hf\big(t_n + \tfrac{1}{2}h, \tfrac{1}{2}(y_n + y_{n+1})\big).$$

This is the special case of the method discussed in the next problem. Please look at the work there when $\theta = 1/2$ for a proof that the method is second order.

Define the error $e_n = y(t_n) - y_n$. By Taylor's theorem

$$y(t_{n+1}) = y(t_n + \tfrac{1}{2}h + \tfrac{1}{2}h) = y(t_n + \tfrac{1}{2}h) + \tfrac{1}{2}hy'(t_n + \tfrac{1}{2}h) + \mathcal{O}(h^2)$$

and

$$y(t_n) = y(t_n + \tfrac{1}{2}h - \tfrac{1}{2}h) = y(t_n + \tfrac{1}{2}h) - \tfrac{1}{2}hy'(t_n + \tfrac{1}{2}h) + \mathcal{O}(h^2).$$

Therefore

$$y(t_{n+1}) = y(t_n) + hy'(t_n + \tfrac{1}{2}h) + \mathcal{O}(h^2)$$

and

$$y(t_n + \tfrac{1}{2}h) = \tfrac{1}{2}\big(y(t_n) + y(t_{n+1})\big) + \mathcal{O}(h^2).$$

It follows that

$$\begin{aligned}
|e_{n+1}| &= \big|y(t_{n+1}) - y_{n+1}\big| \\
&= \big|y(t_n) + hy'(t_n + \tfrac{1}{2}h) - y_n - hf\big(t_n + \tfrac{1}{2}h, \tfrac{1}{2}(y_n + y_{n+1})\big)\big| + \mathcal{O}(h^2) \\
&\le |e_n| + h\big|f\big(t_n + \tfrac{1}{2}h, y(t_n + \tfrac{1}{2}h)\big) - f\big(t_n + \tfrac{1}{2}h, \tfrac{1}{2}(y_n + y_{n+1})\big)\big| + \mathcal{O}(h^2).
\end{aligned}$$

Now use the Lipschitz condition to conclude

$$\begin{aligned}
&\big|f\big(t_n + \tfrac{1}{2}h, y(t_n + \tfrac{1}{2}h)\big) - f\big(t_n + \tfrac{1}{2}h, \tfrac{1}{2}(y_n + y_{n+1})\big)\big| \le \lambda\big|y(t_n + \tfrac{1}{2}h) - \tfrac{1}{2}(y_n + y_{n+1})\big| \\
&\qquad \le \tfrac{1}{2}\lambda\big|y(t_n) + y(t_{n+1}) - \tfrac{1}{2}(y_n + y_{n+1})\big| + \mathcal{O}(h^2) \le \tfrac{1}{2}\lambda\big(|e_n| + |e_{n+1}|\big).
\end{aligned}$$

Consequently,

$$|e_{n+1}| \le |e_n| + \tfrac{1}{2}h\lambda\big(|e_n| + |e_{n+1}|\big) + \mathcal{O}(h^2)$$

and so

$$|e_{n+1}| \le \gamma|e_n| + \mathcal{O}(h^2) \le \gamma|e_n| + Ah^2 \qquad \text{where} \qquad \gamma = \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}$$

for some A large enough.

Since $|e_0| = |y(t_0) - y_0| = 0$, then by induction we have

$$\begin{aligned}
&|e_1| \le Ah^2, \qquad |e_2| \le \gamma Ah^2 + Ah^2 \\
&|e_3| \le \big(\gamma^2 + \gamma + 1\big)Ah^2 \\
&\quad \cdots \\
&|e_n| \le \big(\gamma^{n-1} + \cdots + \gamma + 1\big)Ah^2 = \frac{\gamma^n - 1}{\gamma - 1}Ah^2.
\end{aligned}$$

We now claim that

$$\frac{\gamma^n - 1}{\gamma - 1} A h^2 \to 0 \qquad \text{as} \qquad n \to \infty \qquad \text{where} \qquad h = \frac{T - t_0}{n}.$$

To see this, we recall for $x > 0$ that

$$1 + x \le 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots = e^x$$

and for $0 < x < 1/2$ that

$$\frac{1}{1-x} = 1 + \frac{x}{1-x} \le 1 + 2x \le e^{2x}.$$

Since $h \to 0$ we may assume $h\lambda \le 1$ and conclude $\gamma^n \le \left(e^{h\lambda/2} e^{h\lambda}\right)^2 = e^{3(T-t_0)\lambda/2}$. Now

$$\gamma - 1 = \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} - 1 = \frac{h\lambda}{1 - \frac{1}{2}h\lambda} \qquad \text{implies} \qquad \frac{1}{\gamma - 1} = \frac{1 - \frac{1}{2}h\lambda}{h\lambda}.$$

Therefore,

$$|e_n| \le \frac{\gamma^n - 1}{\gamma - 1} A h^2 \le \left(e^{3(T-t_0)\lambda/2} - 1\right) \frac{1 - \frac{1}{2}h\lambda}{h\lambda} A h^2 \le \left(e^{3(T-t_0)\lambda/2} - 1\right) \frac{1}{\lambda} A h \to 0$$

as $h \to 0$ and $n \to \infty$. This shows the implicit midpoint rule is convergent.

2. [Iserles 1.4] Given $\theta \in [0,1]$, find the order of the method

$$y_{n+1} = y_n + hf\big(t_n + (1-\theta)h, \theta y_n + (1-\theta)y_{n+1}\big).$$

Expanding the truncation error

$$\tau(h) = y(t+h) - y(t) - hf\big(t + (1-\theta)h, \theta y(t) + (1-\theta)y(t+h)\big)$$

in powers of h by using the Mathematica script

```
eq=y'->Function[t,f[t,y[t]]]
r=y[t+h]-y[t]-h*(f[t+(1-theta)*h,theta*y[t]+(1-theta)*y[t+h]])
Dr=r
For[i=0,i<4,i++,
    t0=Simplify[Dr/.h->0];
    Print["Tau^",i," = ",t0];
    If[t0==0,Null,Break[],Break[]];
    Dr=Simplify[D[Dr,h]/.eq]
]
```

to compute the derivatives $\tau^{(k)}(0) =$ `Tau^`k yields

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= eq=y'->Function[t,f[t,y[t]]]

Out[1]= y' -> Function[t, f[t, y[t]]]

In[2]:= r=y[t+h]-y[t]-h*(f[t+(1-theta)*h,theta*y[t]+(1-theta)*y[t+h]])

Out[2]= -(h f[t + h (1 - theta), theta y[t] + (1 - theta) y[h + t]]) - y[t] +

>    y[h + t]

In[3]:= Dr=r

Out[3]= -(h f[t + h (1 - theta), theta y[t] + (1 - theta) y[h + t]]) - y[t] +

>    y[h + t]

In[4]:= For[i=0,i<4,i++,
t0=Simplify[Dr/.h->0];
Print["Tau^",i,"=",t0];
If[t0==0,Null,Break[],Break[]];
Dr=Simplify[D[Dr,h]/.eq]
```

```
]
Tau^0 = 0
Tau^1 = 0
                                             (0,1)                (1,0)
Tau^2 = (-1 + 2 theta) (f[t, y[t]] f      [t, y[t]] + f      [t, y[t]])

In[5]:=
```

This indicates for values of $\theta \neq 1/2$ that $\tau(h) = \mathcal{O}(h^2)$ and consequently that the method is first order. If $\theta = 1/2$ then the modified script

```
eq=y'->Function[t,f[t,y[t]]]
theta=1/2
r=y[t+h]-y[t]-h*(f[t+(1-theta)*h,theta*y[t]+(1-theta)*y[t+h]])
Dr=r
For[i=0,i<4,i++,
    t0=Simplify[Dr/.h->0];
    Print["Tau^",i," = ",t0];
    If[t0==0,Null,Break[],Break[]];
    Dr=Simplify[D[Dr,h]/.eq]
]
```

yields

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= eq=y'->Function[t,f[t,y[t]]]

Out[1]= y' -> Function[t, f[t, y[t]]]

In[2]:= theta=1/2

        1
Out[2]= -
        2

In[3]:= r=y[t+h]-y[t]-h*(f[t+(1-theta)*h,theta*y[t]+(1-theta)*y[t+h]])

              h      y[t]   y[h + t]
Out[3]= -(h f[- + t, ---- + --------]) - y[t] + y[h + t]
              2       2        2

In[4]:= Dr=r

              h      y[t]   y[h + t]
```

```
Out[4]= -(h f[- + t, ---- + --------]) - y[t] + y[h + t]
              2       2        2

In[5]:= For[i=0,i<4,i++,
t0=Simplify[Dr/.h->0];
Print["Tau^",i,"=",t0];
If[t0==0,Null,Break[],Break[]];
Dr=Simplify[D[Dr,h]/.eq]
]
Tau^0 = 0
Tau^1 = 0
Tau^2 = 0
                   2  (0,2)               (0,1)           (1,0)
Tau^3 = (f[t, y[t]]  f     [t, y[t]] - 2 f     [t, y[t]] f     [t, y[t]] -

                      (0,1)         2    (1,1)               (2,0)
>      2 f[t, y[t]] (f     [t, y[t]]  - f     [t, y[t]]) + f     [t, y[t]]) / 4

In[6]:=
```

This shows when $\theta = 1/2$ that $\tau(h) = \mathcal{O}(h^3)$ and the method is second order.

3. [Iserles 1.5] Provided that f is analytic, it is possible to obtain from $y' = f(t,y)$ an expression for the second derivative of y, namely $y'' = g(t,y)$, where

$$g(t,y) = \frac{\partial f(t,y)}{\partial t} + \frac{\partial f(t,y)}{\partial y} f(t,y).$$

Find the orders of the methods

$$y_{n+1} = y_n + hf(t_n, y_n) + \tfrac{1}{2}h^2 g(t_n, y_n)$$

and

$$y_{n+1} = y_n + \tfrac{1}{2}h\big[f(t_n, y_n) + f(t_{n+1}, y_{n+1})\big] + \tfrac{1}{12}h^2\big[g(t_n, y_n) - g(t_{n+1}, y_{n+1})\big].$$

For the first method use the script

```
1  eq=y'->Function[t,f[t,y[t]]]
2  r=y[t+h]-y[t]-h*f[t,y[t]]-h^2/2*g[t,y[t]]
3  g=Function[{t,y},Evaluate[D[f[t,y],t]+D[f[t,y],y]*f[t,y]]]
4  Dr=r
5  For[i=0,i<6,i++,
6      t0=Simplify[Dr/.h->0];
7      Print["Tau^",i," = ",t0];
8      If[t0==0,Null,Break[],Break[]];
9      Dr=Simplify[D[Dr,h]/.eq]
10 ]
```

Note that the use of `Evaluate` was necessary in line 3 for the definition of g to prevent lazy evaluation so the derivatives in line 9 work properly. When run, we obtain

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= eq=y'->Function[t,f[t,y[t]]]

Out[1]= y' -> Function[t, f[t, y[t]]]

In[2]:= r=y[t+h]-y[t]-h*f[t,y[t]]-h^2/2*g[t,y[t]]

                                 2
                                h  g[t, y[t]]
Out[2]= -(h f[t, y[t]]) - ------------- - y[t] + y[h + t]
                                      2

In[3]:= g=Function[{t,y},Evaluate[D[f[t,y],t]+D[f[t,y],y]*f[t,y]]]

                                    (0,1)          (1,0)
```

```
Out[3]= Function[{t, y}, f[t, y] f     [t, y] + f     [t, y]]

In[4]:= Dr=r

Out[4]= -(h f[t, y[t]]) - y[t] + y[h + t] -

      2              (0,1)             (1,0)
     h  (f[t, y[t]] f     [t, y[t]] + f     [t, y[t]])
>    -----------------------------------------------
                             2

In[5]:= For[i=0,i<6,i++,
t0=Simplify[Dr/.h->0];
Print["Tau^",i,"=",t0];
If[t0==0,Null,Break[],Break[]];
Dr=Simplify[D[Dr,h]/.eq]
]
Tau^0 = 0
Tau^1 = 0
Tau^2 = 0
                  2  (0,2)             (0,1)           (1,0)
Tau^3 = f[t, y[t]]  f     [t, y[t]] + f     [t, y[t]] f     [t, y[t]] +

                  (0,1)         2      (1,1)              (2,0)
>    f[t, y[t]] (f     [t, y[t]]  + 2 f     [t, y[t]]) + f     [t, y[t]]

In[6]:=
```

This shows that $\tau(h) = \mathcal{O}(h^3)$ for the first method which we then infer is order 2.

For the second method, the script

```
eq=y'->Function[t,f[t,y[t]]]
r=y[t+h]-y[t]-h/2*(f[t,y[t]]+f[t+h,y[t+h]])-
    h^2/12*(g[t,y[t]]-g[t+h,y[t+h]])
g=Function[{t,y},Evaluate[D[f[t,y],t]+D[f[t,y],y]*f[t,y]]]
Dr=r
For[i=0,i<6,i++,
    t0=Simplify[Dr/.h->0];
    Print["Tau^",i," = ",t0];
    If[t0==0,Null,Break[],Break[]];
    Dr=Simplify[D[Dr,h]/.eq]
]
```

yields

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
```

Math 467/667: Homework 1 Solutions


```
In[1]:= eq=y'->Function[t,f[t,y[t]]]

Out[1]= y' -> Function[t, f[t, y[t]]]

In[2]:= r=y[t+h]-y[t]-h/2*(f[t,y[t]]+f[t+h,y[t+h]])-
h^2/12*(g[t,y[t]]-g[t+h,y[t+h]])

        -(h (f[t, y[t]] + f[h + t, y[h + t]]))
Out[2]= -------------------------------------- -
                          2

      2
     h  (g[t, y[t]] - g[h + t, y[h + t]])
>    ------------------------------------ - y[t] + y[h + t]
                      12

In[3]:= g=Function[{t,y},Evaluate[D[f[t,y],t]+D[f[t,y],y]*f[t,y]]]

                                  (0,1)          (1,0)
Out[3]= Function[{t, y}, f[t, y] f     [t, y] + f     [t, y]]

In[4]:= Dr=r

        -(h (f[t, y[t]] + f[h + t, y[h + t]]))
Out[4]= -------------------------------------- - y[t] + y[h + t] -
                          2

       2              (0,1)
>    (h  (f[t, y[t]] f     [t, y[t]] -

                              (0,1)                       (1,0)
>         f[h + t, y[h + t]] f     [h + t, y[h + t]] + f     [t, y[t]] -

           (1,0)
>         f     [h + t, y[h + t]])) / 12

In[5]:= For[i=0,i<6,i++,
t0=Simplify[Dr/.h->0];
Print["Tau^",i,"=",t0];
If[t0==0,Null,Break[],Break[]];
Dr=Simplify[D[Dr,h]/.eq]
]
```

Math 467/667: Homework 1 Solutions

```
Tau^0 = 0
Tau^1 = 0
Tau^2 = 0
Tau^3 = 0
Tau^4 = 0
                   4  (0,4)             (0,1)         3  (1,0)
Tau^5 = (f[t, y[t]]  f     [t, y[t]] + f     [t, y[t]]  f     [t, y[t]] +

          (0,2)           (1,0)         2
>      3 f     [t, y[t]] f     [t, y[t]]  +

                 3     (0,2)         2      (0,1)           (0,3)
>      f[t, y[t]]  (4 f     [t, y[t]]  + 7 f     [t, y[t]] f     [t, y[t]] +

             (1,3)              (0,1)         2  (2,0)
>         4 f     [t, y[t]]) + f     [t, y[t]]  f     [t, y[t]] +

          (1,1)           (2,0)               (1,0)           (2,1)
>      4 f     [t, y[t]] f     [t, y[t]] + 6 f     [t, y[t]] f     [t, y[t]] +

                 2      (0,1)         2  (0,2)
>      f[t, y[t]]  (11 f     [t, y[t]]  f     [t, y[t]] +

              (0,1)           (1,2)
>         15 f     [t, y[t]] f     [t, y[t]] +

              (0,3)           (1,0)
>         6 (f     [t, y[t]] f     [t, y[t]] +

                (0,2)           (1,1)             (2,2)
>            2 f     [t, y[t]] f     [t, y[t]] + f     [t, y[t]])) +

        (0,1)              (1,0)           (1,1)             (3,0)
>      f     [t, y[t]] (7 f     [t, y[t]] f     [t, y[t]] + f     [t, y[t]]) +

                    (0,1)         4      (0,1)         2  (1,1)
>      f[t, y[t]] (f     [t, y[t]]  + 9 f     [t, y[t]]  f     [t, y[t]] +

           (0,1)               (0,2)           (1,0)
>         f     [t, y[t]] (13 f     [t, y[t]] f     [t, y[t]] +

                (2,1)
>            9 f     [t, y[t]]) +
```

```
                (1,1)          2      (1,0)            (1,2)
>         4 (2 f      [t, y[t]]  + 3 f      [t, y[t]] f      [t, y[t]] +

              (0,2)           (2,0)             (3,1)
>            f     [t, y[t]] f     [t, y[t]] + f     [t, y[t]])) +

        (4,0)
>      f     [t, y[t]]) / 6

In[6]:=
```

This shows that $\tau(h) = \mathcal{O}(h^5)$ for the second method which we then infer is order 4.

4. [Iserles 1.8] Let f be analytic. Prove that, for sufficiently small $h > 0$ and an analytic function x, the function

$$x(t+h) - x(t-h) - hf\big(\tfrac{1}{2}\big(x(t-h) + x(t+h)\big)\big)$$

can be expanded into power series in odd powers of h. Deduce that the error in the implicit midpoint rule (1.13) when applied to autonomous ODEs $y' = f(y)$ also admits an expansion in odd powers of h. Hint: First try to prove the statement for a scalar function f. Once you have solved this problem, a generalization should present no difficulties.

Let

$$\xi(h) = x(t+h) - x(t-h) - hf\big(\tfrac{1}{2}\big(x(t-h) + x(t+h)\big)\big)$$

and note that

$$\xi(-h) = x(t-h) - x(t+h) + hf\big(\tfrac{1}{2}\big(x(t+h) + x(t-h)\big)\big) = -\xi(h).$$

Therefore $\xi(h)$ is an odd function in h. It follows that that the derivatives

$$\xi^{(k)}(h) = \begin{cases} \text{is an even function for } k \text{ odd} \\ \text{is an odd function for } k \text{ even.} \end{cases}$$

Since any odd function must be zero at the origin, we conclude $\xi^{(k)}(0) = 0$ for k even. Consequently the power series expansion

$$\xi(h) = \sum_{k=0}^{\infty} \frac{1}{k!} h^k \xi^{(k)}(0) = \sum_{k \text{ odd}} \frac{1}{k!} h^k \xi^{(k)}(0)$$

contains only odd powers of h. Upon noting that all of the above holds verbatim when $\xi(h)$ is a vector-valued function, we conclude that this result holds equally well when f is vector valued.

Finally, to draw a connection to the implicit midpoint rule first note the f in that rule is different than the one in the definition of $\xi(h)$. To avoid confusion, we therefore call the force in the implicit midpoint rule by g and write

$$y_{n+1} = y_n + hg\big(t_n + \tfrac{1}{2}h, \tfrac{1}{2}(y_n + y_{n+1})\big),$$

Note when g is autonomous that this rule becomes

$$y_{n+1} = y_n + hg\big(\tfrac{1}{2}(y_n + y_{n+1})\big)$$

with the resulting truncation error

$$\tau(h) = y(t_{n+1}) - y(t_n) - hg\big(\tfrac{1}{2}(y(t_n) + y(t_{n+1}))\big).$$

Now, taking $g(y) = 2f(x)$ and identifying $y(t) = x(t - h/2)$ yields

$$\begin{aligned} \tau(h) &= y(t+h) - y(t) - hg\big(\tfrac{1}{2}(y(t) + y(t+h))\big) \\ &= x(t+h/2) - x(t-h/2) - \tfrac{1}{2}hf\big(\tfrac{1}{2}(x(t-h/2) + x(t+h/2))\big) = \xi(h/2). \end{aligned}$$

Therefore $\tau(h) = \xi(h/2)$. This shows that, provided a shift of $h/2$ is made in time, that the truncation error in the implicit midpoint rule admits an expansion in odd powers of h. Frankly, I found such shifting in time when computing the truncation error a bit confusing if not dubious.

5. [Iserles 2.3] Instead of (2.3), consider the identity

$$y(t_{n+s}) = y(t_{n+s-2}) + \int_{t_{n+s-2}}^{t_{n+s}} f(\tau, y(\tau))d\tau.$$

(i) Replace $f(\tau, y(\tau))$ by the interpolating polynomial p from Section 2.1 and substitute y_{n+s-2} in place of $y(t_{n+s-2})$. Prove that the resultant explicit Nystrom method is of order $p = s$.

As in Section 2.1 consider the interpolating polynomial p of order $s-1$ such that

$$p(t_m) = f(t_m, y(t_m)) \qquad \text{for} \qquad m = 0, \ldots, s-1.$$

As before, interpolation theory and the smoothness of y implies

$$p(t) = y'(t) + \mathcal{O}(h^s) \qquad \text{for} \qquad t \in [t_{n+s-1}, t_{n+s}].$$

Since τ is already used as the variable of integration let the truncation error in the resulting method by written ψ_n. Thus,

$$y(t_{n+s}) = y(t_{n+s-2}) + \int_{t_{n+s-2}}^{t_{n+s}} p(\tau)d\tau + \psi_n(h).$$

It follows setting $t = t_{n+s-1}$, dropping the subscripts and using the fact that $\mathcal{O}(h^s)$ stands for a uniform bound on the error in p that

$$\begin{aligned}
\psi(h) &= y(t+h) - y(t-h) - \int_{t-h}^{t+h} p(\tau)d\tau \\
&= y(t+h) - y(t-h) - \int_{t-h}^{t+h} \Big(y'(t) + \mathcal{O}(h^s)\Big)d\tau \\
&= y(t+h) - y(t-h) - \int_{t-h}^{t+h} y'(t)d\tau - 2h\mathcal{O}(h^s) \\
&= y(t+h) - y(t-h) - \big(y(t+h) - y(t-h)\big) + \mathcal{O}(h^{s+1}) = \mathcal{O}(h^{s+1}).
\end{aligned}$$

Therefore, the explicit Nystrom method is of order $p = s$.

(ii) Derive the two-step Nystrom method in a closed form by using the above approach.

To derive the two-step method we set $s = 2$ and explicitly write down the polynomial p as

$$\begin{aligned}
p(t) &= p_0(t)f(t_n, y_n) + p_1(t)f(t_{n+1}, y_{n+1}) \\
&= \frac{t - t_{n+1}}{t_n - t_{n+1}} f(t_n, y_n) + \frac{t - t_n}{t_{t+1} - t_n} f(t_{n+1}, y_{n+1}) \\
&= \frac{1}{h}\big((t_{n+1} - t)f(t_n, y_n) + (t - t_n)f(t_{n+1}, y_{n+1})\big) \\
&= \frac{1}{h}\big(t_{n+1}f(t_n, y_n) - t_n f(t_{n+1}, y_{n+1}) + t(f(t_{n+1}, y_{n+1}) - f(t_n, y_n))\big).
\end{aligned}$$

Consequently,

$$\begin{aligned}\int_{t_n}^{t_{n+2}} p(\tau)d\tau &= 2\big(t_{n+1}f(t_n,y_n) - t_n f(t_{n+1},y_{n+1})\big)\\ &\quad + \tfrac{1}{2h}\big(t_{n+2}^2 - t_n^2\big)\big(f(t_{n+1},y_{n+1}) - f(t_n,y_n)\big)\\ &= 2\big(t_{n+1}f(t_n,y_n) - (t_{n+1}-h)f(t_{n+1},y_{n+1})\big)\\ &\quad + \tfrac{1}{2h}\big((t_{n+1}+h)^2 - (t_{n+1}-h)^2\big)\big(f(t_{n+1},y_{n+1}) - f(t_n,y_n)\big)\\ &= 2t_{n+1}\big(f(t_n,y_n) - f(t_{n+1},y_{n+1})\big) + 2hf(t_{n+1},y_{n+1})\\ &\quad + \tfrac{1}{2h}\big(2ht_{n+1}\big)\big(f(t_{n+1},y_{n+1}) - f(t_n,y_n)\big)\\ &= 2hf(t_{n+1},y_{n+1}).\end{aligned}$$

Therefore the two-step Nystrom method is $y_{n+2} = y_n + 2hf(t_{n+1},y_{n+1})$.

(iii) Find the coefficients of the two-step and three-step Nystrom methods by noticing that $\rho(w) = w^{s-2}(w^2-1)$ and evaluating σ from (2.13).

For $s=2$ we obtain

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= s=2

Out[1]= 2

In[2]:= rho=w^(s-2)*(w^2-1)

                2
Out[2]= -1 + w

In[3]:= T1=Series[rho/Log[w],{w,1,s-1}]

                                    2
Out[3]= 2 + 2 (-1 + w) + O[-1 + w]

In[4]:= P1=Normal[T1]

Out[4]= 2 + 2 (-1 + w)

In[5]:= sigma=Expand[P1]

Out[5]= 2 w

In[6]:=
```

which is consistent with the previous problem.

For $s = 3$ we obtain

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= s=3

Out[1]= 3

In[2]:= rho=w^(s-2)*(w^2-1)

                  2
Out[2]= w (-1 + w )

In[3]:= T1=Series[rho/Log[w],{w,1,s-1}]

                                    2
                         7 (-1 + w)            3
Out[3]= 2 + 4 (-1 + w) + ----------- + O[-1 + w]
                              3

In[4]:= P1=Normal[T1]

                                    2
                         7 (-1 + w)
Out[4]= 2 + 4 (-1 + w) + -----------
                              3

In[5]:= sigma=Expand[P1]

                      2
        1   2 w   7 w
Out[5]= - - --- + ----
        3    3     3

In[6]:=
```

This shows the three-step Nystrom method is

$$y_{n+3} = y_{n+1} + h\big[\tfrac{1}{3}f(t_n, y_n) - \tfrac{2}{3}f(t_{n+1}, y_{n+1}) + \tfrac{7}{3}f(t_{n+2}, y_{n+2})\big].$$

(iv) Derive the two-step third-order implicit Milne method, again letting $\rho(w) = w^{s-2}(w^2-1)$ but allowing σ to be of degree s.

The Mathematica computation

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= s=2

Out[1]= 2

In[2]:= rho=w^(s-2)*(w^2-1)

                2
Out[2]= -1 + w

In[3]:= T1=Series[rho/Log[w],{w,1,s}]

                                    2
                            (-1 + w)             3
Out[3]= 2 + 2 (-1 + w) + --------- + O[-1 + w]
                                3

In[4]:= P1=Normal[T1]

                                    2
                            (-1 + w)
Out[4]= 2 + 2 (-1 + w) + ---------
                                3

In[5]:= sigma=Expand[P1]

                     2
        1   4 w   w
Out[5]= - + --- + --
        3    3    3

In[6]:=
```

implies the two-step third-order implicit Milne method is

$$y_{n+2} = y_n + h\big[\tfrac{1}{3}f(t_n, y_n) + \tfrac{4}{3}f(t_{n+1}, y_{n+1}) + \tfrac{1}{3}f(t_{n+2}, y_{n+2})\big].$$

6. [Iserles 2.4] Determine the order of the three-step method

$$y_{n+3} - y_n = h\left[\tfrac{3}{8} f(t_{n+3}, y_{n+3}) + \tfrac{9}{8} f(t_{n+2}, y_{n+2}) + \tfrac{9}{8} f(t_{n+1}, y_{n+1}) + \tfrac{3}{8} f(t_n, y_n)\right],$$

the three-eighths scheme. Is it convergent?

Compute the order of the method in two different ways. First, using a similar script as used for problem Iserles 1.4. Namely,

```
eq=y'->Function[t,f[t,y[t]]]
r=y[t+h]-y[t-2*h]-h*(3/8*f[t+h,y[t+h]]+9/8*f[t,y[t]]+
    9/8*f[t-h,y[t-h]]+3/8*f[t-2*h,y[t-2*h]])
Dr=r
For[i=0,i<8,i++,
    t0=Simplify[Dr/.h->0];
    Print["Tau^",i," = ",t0];
    If[t0==0,Null,Break[],Break[]];
    Dr=Simplify[D[Dr,h]/.eq]
]
```

The output

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.

In[1]:= eq=y'->Function[t,f[t,y[t]]]

Out[1]= y' -> Function[t, f[t, y[t]]]

In[2]:= r=y[t+h]-y[t-2*h]-h*(3/8*f[t+h,y[t+h]]+9/8*f[t,y[t]]+
9/8*f[t-h,y[t-h]]+3/8*f[t-2*h,y[t-2*h]])

             9 f[t, y[t]]   3 f[-2 h + t, y[-2 h + t]]
Out[2]= -(h (------------ + -------------------------- +
                  8                     8

          9 f[-h + t, y[-h + t]]   3 f[h + t, y[h + t]]
>         ---------------------- + --------------------)) - y[-2 h + t] +
                    8                        8

>    y[h + t]

In[3]:= Dr=r

             9 f[t, y[t]]   3 f[-2 h + t, y[-2 h + t]]
Out[3]= -(h (------------ + -------------------------- +
                  8                     8
```

```
          9 f[-h + t, y[-h + t]]   3 f[h + t, y[h + t]]
>         --------------------- + -------------------)) - y[-2 h + t] +
                    8                        8

>    y[h + t]

In[4]:= For[i=0,i<8,i++,
t0=Simplify[Dr/.h->0];
Print["Tau^",i,"=",t0];
If[t0==0,Null,Break[],Break[]];
Dr=Simplify[D[Dr,h]/.eq]
]
Tau^0 = 0
Tau^1 = 0
Tau^2 = 0
Tau^3 = 0
Tau^4 = 0
                       4  (0,4)              (0,1)         3  (1,0)
Tau^5 = (-9 (f[t, y[t]]  f     [t, y[t]] + f     [t, y[t]]  f     [t, y[t]] +

            (0,2)           (1,0)         2
>        3 f     [t, y[t]] f     [t, y[t]]  +

                   3     (0,2)         2      (0,1)           (0,3)
>        f[t, y[t]]  (4 f     [t, y[t]]  + 7 f     [t, y[t]] f     [t, y[t]] +

               (1,3)              (0,1)         2  (2,0)
>           4 f     [t, y[t]]) + f     [t, y[t]]  f     [t, y[t]] +

            (1,1)           (2,0)
>        4 f     [t, y[t]] f     [t, y[t]] +

            (1,0)           (2,1)
>        6 f     [t, y[t]] f     [t, y[t]] +

                   2      (0,1)         2  (0,2)
>        f[t, y[t]]  (11 f     [t, y[t]]  f     [t, y[t]] +

                (0,1)           (1,2)
>           15 f     [t, y[t]] f     [t, y[t]] +

                (0,3)           (1,0)
>           6 (f     [t, y[t]] f     [t, y[t]] +
```

```
                     (0,2)          (1,1)            (2,2)
>               2 f     [t, y[t]] f     [t, y[t]] + f     [t, y[t]])) +

             (0,1)              (1,0)           (1,1)
>         f     [t, y[t]] (7 f     [t, y[t]] f     [t, y[t]] +

                (3,0)
>           f     [t, y[t]]) + f[t, y[t]]

               (0,1)         4      (0,1)         2  (1,1)
>         (f     [t, y[t]]  + 9 f     [t, y[t]]  f     [t, y[t]] +

                (0,1)              (0,2)           (1,0)
>           f     [t, y[t]] (13 f     [t, y[t]] f     [t, y[t]] +

                     (2,1)
>               9 f     [t, y[t]]) +

                     (1,1)         2      (1,0)            (1,2)
>           4 (2 f     [t, y[t]]  + 3 f     [t, y[t]] f     [t, y[t]] +

                   (0,2)           (2,0)             (3,1)
>               f     [t, y[t]] f     [t, y[t]] + f     [t, y[t]])) +

             (4,0)
>         f     [t, y[t]])) / 2

In[5]:=
```

indicates that $\tau(h) = \mathcal{O}(h^5)$ and so the three-eights scheme is fourth order.

The same result can be achieved using Theorem 2.1 from the text. In this case we take

$$\rho(w) = w^3 - 1 \qquad \text{and} \qquad \sigma(w) = \tfrac{3}{8}w^3 + \tfrac{9}{8}w^2 + \tfrac{9}{8}w + \tfrac{3}{8}.$$

The script

```
rho=w^3-1
sigma=3/8*w^3+9/8*w^2+9/8*w+3/8
r=rho-Log[w]*sigma
Series[r,{w,1,5}]
```

produces the output

```
Wolfram Language 12.1.1 Engine for Linux ARM (32-bit)
Copyright 1988-2020 Wolfram Research, Inc.
```

Math 467/667: Homework 1 Solutions

```
In[1]:= rho=w^3-1

              3
Out[1]= -1 + w

In[2]:= sigma=3/8*w^3+9/8*w^2+9/8*w+3/8

                     2      3
        3   9 w   9 w    3 w
Out[2]= - + --- + ---- + ----
        8    8     8      8

In[3]:= r=rho-Log[w]*sigma

                                2      3
              3    3   9 w   9 w    3 w
Out[3]= -1 + w  - (- + --- + ---- + ----) Log[w]
                   8    8     8      8

In[4]:= Series[r,{w,1,5}]

                   5
        -3 (-1 + w)             6
Out[4]= ------------ + O[-1 + w]
             80

In[5]:=
```

which again indicates that $\tau(h) = \mathcal{O}(h^5)$.

Convergence of the method can be checked using the Dahlquist root condition. In this case the roots of $\rho(w) = 0$ all lie on the boundary of the unit disk and are simple. This implies the scheme is convergent.