Math/CS 467/667 Final Review

1. What is the difference between implicit and explicit Runge–Kutta schemes? When would you use one over the other?

Runge–Kutta schemes are usually described using tableaux of the form

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

An explicit scheme results when A is lower triangular; whereas implicit schemes result when A is not lower triangular. Explicit schemes are easy to program for non-linear problems; however, solving a non-linear problem using an implicit scheme requires inversion of a non-linear function at every time step. On the other hand, implicit schemes have greater stability properties which allow for much larger time steps when solving stiff problems. In particular, the domain of stability for a Runge–Kutta method is given by $|r(z)| < 1$ where

$$r(z) = 1 + zb^T(1 - zA)^{-1}\mathbf{1} \quad \text{and} \quad \mathbf{1} = (1, 1, \ldots, 1).$$

If A is lower triangular, then $r(z)$ is a polynomial and the domain of stability will always be a bounded region. Thus, explicit Runge–Kutta methods are never unconditionally stable.

2. What is a divide-and-conquer method? What is their significance and what is the benefit of using such a strategy? Explain why the fast Fourier transform is fast.

A divide-and-conquer method takes a large problem and divides it into smaller problems which are easier to solve. These smaller problems may, in turn, be further divided into even smaller problems until the resulting problems are trivial to solve. Suppose a problem of size N takes N^k steps to solve but may be broken into two smaller problems of size $N/2$. Assuming each of the smaller problems takes $(N/2)^k$ steps to solve, the total number of steps now needed to solve the two smaller problems is

$$(N/2)^k + (N/2)^k = 2^{1-k}N^k$$

which results in computational savings when $k > 1$. Moreover, as the smaller problems are independent of one another, they can be computed in parallel. So divide-and-conquer methods can not only reduce the total number of steps to solve a problem, but allow those steps to be performed in parallel.

In the case of the Fourier transform, the computation of size N may be split into two smaller problems of size $M = N/2$ as

$$\begin{aligned} Ny_k &= \sum_{j=0}^{N-1} x_j e^{-2\pi ijk/N} = \sum_{\text{even}} x_j e^{-2\pi ijk/N} + \sum_{\text{odd}} x_j e^{-2\pi ijk/N} \\ &= \sum_{j=0}^{N/2-1} x_{2j} e^{-2\pi i(2j)k/N} + \sum_{j=0}^{N/2-1} x_{2j+1} e^{-2\pi i(2j+1)k/N} \\ &= \sum_{j=0}^{M-1} x_{2j} e^{-2\pi ijk/M} + e^{-2\pi ik/N} \sum_{j=0}^{N/2-1} x_{2j+1} e^{-2\pi ijk/M}. \end{aligned}$$

Since each Fourier transform takes N^2 number of operations, the number of operations required to perform the two smaller problems is $2^{-1}N^2$. As one also needs to add the results of the two smaller transforms together to find the original transform the total number of operations is $2^{-1}N^2 + N$. Now assuming $N = 2^n$ for some n, after m subdivisions, the total number of operations is given by

$$T_m = 2^{-m}2^{2n} + m2^n = 2^n(2^{n-m} + m)$$

Successively divided until reaching Fourier transforms of size one yields that

$$T_n = 2^n(1 + n) \approx N \log_2(N).$$

Thus, divide-and-conquer allows one to calculate a Fourier transform with a complexity that depends almost linearly on the size of the problem rather than as N^2.

3. Find the linear stability domain for the first-order backwards difference formula given by $y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$.

Consider the linear differential equation

$$y' = \lambda y \qquad \text{with} \qquad y(t_0) = y_0.$$

The backwards difference formula yields

$$y_{n+1} = y_n + h\lambda y_{n+1} \qquad \text{or} \qquad y_{n+1} = \frac{1}{1 - h\lambda} y_n$$

Solving this recurrence relation for y_n obtains

$$y_n = \Big(\frac{1}{1 - h\lambda}\Big)^n y_0.$$

Since

$$y_n \to 0 \qquad \text{if and only if} \qquad \Big|\frac{1}{1 - h\lambda}\Big| < 1$$

we conclude, upon identifying $z = h\lambda$, that $\mathcal{D} = \{\, z : |z - 1| > 1 \,\}$.

4. State the the Runge–Kutta fourth-order scheme and the Taylor second-order scheme. Prove the Runge–Kutta scheme is actually higher order than the Taylor scheme.

Consider the initial value problem

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

The Runge–Kutta scheme is given by

$$y_{n+1} = y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6$$

where

$$\begin{aligned} k_1 &= hf(y_n, t_n) \\ k_2 &= hf(y_n + k_1/2, t_n + h/2) \\ k_3 &= hf(y_n + k_2/2, t_n + h/2) \\ k_4 &= hf(y_n + k_3, t_n + h) \end{aligned}$$

and the Taylor scheme is given by

$$y_{n+1} = y_n + hf(y_n, t_n) + \frac{h^2}{2}\big(f_y(y_n, t_n) f(y_n, t_n) + f_t(y_n, t_n)\big).$$

To show the Taylor is no more than second-order accurate consider the differential equation

$$y' = 3t^2 \quad \text{where} \quad y(0) = 0.$$

The exact solution to this differential equation is $y(t) = t^3$. Since $f_y(y, t) = 0$ and $f_t(y, t) = 6t$, plugging this solutions into the right side of the Taylor scheme yields

$$(t_n)^3 + h3t_n^2 + \frac{h^2}{2} 6t_n = n^3h^3 + 3n^2h^3 + 3nh^3 = (n^3 + 3n^2 + 3n)h^3.$$

However, plugging the solution into the left side yields

$$(t_{n+1})^3 = (n + 1)^3h^3 = (n^3 + 3n^2 + 3n + 1)h^3.$$

The truncation error for this specific solution is

$$\psi = (n^3 + 3n^2 + 3n + 1)h^3 - (n^3 + 3n^2 + 3n)h^3 = h^3.$$

Therefore at time $T = Nh$ the total error is

$$|y(T) - y_N| = Nh^3 = Th^2,$$

which shows that the method is no better than second order.

To show that the Runge–Kutta method is higher order, we need only show it is at least order 3 even though it is really order 4. The truncation error may be computed plugging the solution $y(t)$ into the Runge–Kutta numeric scheme as

$$\psi(h) = y(t + h) - (y(t) + (\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4)/6$$

where

$$\begin{aligned} \kappa_1 &= hf\big(y(t), t\big) \\ \kappa_2 &= hf\big(y(t) + \kappa_1/2, t + h/2\big) \\ \kappa_3 &= hf\big(y(t) + \kappa_2/2, t + h/2\big) \\ \kappa_4 &= hf\big(y(t) + \kappa_3, t + h\big) \end{aligned}$$

and then expanding ψ using Taylor's theorem about $h = 0$ as

$$\psi(h) = \sum_{k=0}^{4} \frac{\psi^{(k)}(0)}{k!} h^k + R_4.$$

Now, to show the method is at least order 3 one needs to show that $\psi^{(k)}(0) = 0$ for $k = 0, \ldots, 3$. The Maple script

```
1  restart;
2  kernelopts(printbytes=false):
3  g[0]:=y(t+h)-(y(t)+(k1+2*k2+2*k3+k4)/6);
4
5  k4:=h*f(y(t)+k3,t+h);
6  k3:=h*f(y(t)+k2/2,t+h/2);
7  k2:=h*f(y(t)+k1/2,t+h/2);
8  k1:=h*f(y(t),t);
9
10 S:={D(y)(t)=f(y(t),t),D(y)(t+h)=f(y(t+h),t+h)};
11
12 G[0]:=simplify(subs(h=0,g[0])):
13 print(psi(0)=G[0]);
14 for k from 1 to 3
15 do
16     g[k]:=subs(S,diff(g[k-1],h));
17     G[k]:=simplify(subs(h=0,g[k]));
18     print((D@@k)(psi)(0)=G[k]);
19 od:
```

computes $\psi^{(k)}(0)$ as `G[k]` in line 17 and produces the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> g[0]:=y(t+h)-(y(t)+(k1+2*k2+2*k3+k4)/6);
                                          k1     k2     k3     k4
                 g[0] := y(t + h) - y(t) - ---- - ---- - ---- - ----
                                          6      3      3      6

>
> k4:=h*f(y(t)+k3,t+h);
                         k4 := h f(y(t) + k3, t + h)

> k3:=h*f(y(t)+k2/2,t+h/2);
                                            k2
                       k3 := h f(y(t) + ----, t + h/2)
```

```
                                                  2

> k2:=h*f(y(t)+k1/2,t+h/2);
                                                  k1
                              k2 := h f(y(t) + ----, t + h/2)
                                                  2

> k1:=h*f(y(t),t);
                                   k1 := h f(y(t), t)

>
> S:={D(y)(t)=f(y(t),t),D(y)(t+h)=f(y(t+h),t+h)};
       S := {D(y)(t) = f(y(t), t), D(y)(t + h) = f(y(t + h), t + h)}

>
> G[0]:=simplify(subs(h=0,g[0])):
> print(psi(0)=G[0]);
                                      psi(0) = 0

> for k from 1 to 3
> do
>     g[k]:=subs(S,diff(g[k-1],h));
>     G[k]:=simplify(subs(h=0,g[k]));
>     print((D@@k)(psi)(0)=G[k]);
> od:
                                     D(psi)(0) = 0

                                      (2)
                                    (D   )(psi)(0) = 0

                                      (3)
                                    (D   )(psi)(0) = 0

> quit
bytes used=2950664, alloc=2555436, time=0.11
```

The results indicate that $\psi^{(k)}(0) = 0$ for $k = 0, \ldots, 3$. Note that the loop in line 14 can be increased to check $k = 0, \ldots, 4$ which further shows the method is fourth order.

5. Consider solving the heat equation $u_t = u_{xx}$ on the interval $[0, 1]$ with boundary conditions $u(0, t) = u(1, t) = 0$ using forward differences in time and central differences in space. Why does $\Delta t \leq \frac{1}{2}(\Delta x)^2$ ensure stability?

Let $u_i^n \approx u(x_i, t_n)$ where $x_i = i\Delta x$ and $t_n = n\Delta t$. Here $\Delta x = 1/m$ and $i = 0, \ldots, m$. Plugging in the finite difference approximations into the heat equation yields

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}.$$

Consequently, writing $u^n = (u_1^n, \dots u_{m-1}^n)$ yields $u^{n+1} = Au^n$ where

$$A = \begin{bmatrix} 1-2\mu & \mu & 0 & \cdots & 0 \\ \mu & 1-2\mu & \mu & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \mu & 1-2\mu & \mu \\ 0 & \cdots & 0 & \mu & 1-2\mu \end{bmatrix} \qquad \text{and} \qquad \mu = \frac{\Delta t}{(\Delta x)^2}.$$

It follows that $u^n = A^n u^0$ and the system is stable if and only if $\rho(A) \le 1$. We now turn our attention to computing the eigenvalues and eigenvectors of the matrix A.

Consider a vector of the form $\xi = (\alpha, \alpha^2, \dots, \alpha^{m-1})$. Then

$$(A - \lambda I)\xi = \begin{bmatrix} \mu + (1-2\mu-\lambda)\alpha + \mu\alpha^2 \\ \mu\alpha + (1-2\mu-\lambda)\alpha^2 + \mu\alpha^3 \\ \vdots \\ \mu\alpha^{k-1} + (1-2\mu-\lambda)\alpha^k + \mu\alpha^{k+1} \\ \vdots \\ \mu\alpha^{m-3} + (1-2\mu-\lambda)\alpha^{m-2} + \mu\alpha^{m-1} \\ \mu\alpha^{m-2} + (1-2\mu-\lambda)\alpha^{m-1} + \mu\alpha^m \end{bmatrix} - \begin{bmatrix} \mu \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \mu\alpha^m \end{bmatrix}$$

Solving for α so that

$$\mu + (1-2\mu-\lambda)\alpha + \mu a^2 = 0$$

yields

$$\alpha_\pm = \frac{-(1-2\mu-\lambda) \pm \sqrt{(1-2\mu-\lambda)^2 - 4\mu^2}}{2\mu}.$$

Let $\xi_\pm = (\alpha_\pm, \alpha_\pm^2, \dots, \alpha_\pm)$. Then

$$A(c_1\xi_+ + c_2\xi_-) = -\big((c_1+c_2)\mu, 0, \dots, 0, \mu(c_1\alpha_+^m + c_2\alpha_-^m)\big).$$

It follows that solving for λ such that there exists c_1 and c_2 where

$$c_1 + c_2 = 0 \qquad \text{and} \qquad c_1\alpha_+^m + c_2\alpha_-^m = 0$$

yields the eigenvalues. The first equation implies $c_2 = -c_1$ from which the second equation implies $a_+^m = a_-^m$. The only way this could happen is if $|a_+| = |a_-|$ which, in turn, implies that the square root is imaginary. Therefore, a_+ and a_- are complex conjugates of each other and there exists ρ and θ such that

$$a_+ = \rho e^{i\theta} \qquad \text{and} \qquad a_- = \rho e^{-i\theta}.$$

Therefore $e^{i\theta m} = e^{-i\theta m}$ or $e^{2i\theta m} = 1$. Taking $\theta = k\pi/m$ for $k = 1, \dots, m-1$ yields $m-1$ linearly independent eigenvectors of the form $c_1\xi_+ + c_2\xi_-$. As $A \in \mathbf{R}^{(m-1)\times(m-1)}$ there are at most $m-1$ linearly independent eigenvectors. Hence, we have found them all.

Now, the square root is imaginary only when λ is real and $|1-2\mu-\lambda|<2\mu$. Thus,

$$0<1-\lambda<4\mu \qquad \text{or} \qquad 1-4\mu<\lambda<1.$$

To ensure that $|\lambda|\le 1$ it is sufficient that

$$1-4\mu\ge -1 \qquad \text{or} \qquad \Delta t\le \frac{1}{2}(\Delta x)^2.$$

At this point it is possible to directly calculate the eigenvalues from the equation

$$\mu+(1-2\mu-\lambda)e^{\pi ik/m}+\mu e^{2\pi ik/m}=0$$

as given on page 482 in Chapter 12 of Faires and Burden; however, in the interest of brevity this is omitted here.

6. Find the order of the following quadrature formula:

(i) $\displaystyle\int_0^1 f(\tau)d\tau \approx \frac{1}{8}f(0)+\frac{3}{8}f\Big(\frac{1}{3}\Big)+\frac{3}{8}f\Big(\frac{2}{3}\Big)+\frac{1}{8}f(1)$

To determine the order, we set $f(x)=x^k$ and find the largest K such that the formula is exact for all $k\le K$. The Maple script

```
restart;
kernelopts(printbytes=false):
T:=f->int(f(tau),tau=0..1)-(1/8*f(0)+3/8*f(1/3)+3/8*f(2/3)+1/8*f(1));
T(x->1);
T(x->x);
T(x->x^2);
T(x->x^3);
T(x->x^4);
```

produces the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> T:=f->int(f(tau),tau=0..1)-(1/8*f(0)+3/8*f(1/3)+3/8*f(2/3)+1/8*f(1));
                     1
                    /
                   |
   T := f ->       |   f(tau) dtau - 1/8 f(0) - 3/8 f(1/3) - 3/8 f(2/3) - 1/8 f(1)
                   |
                  /
                    0
```

```
> T(x->1);
                                           0

> T(x->x);
                                           0

> T(x->x^2);
                                           0

> T(x->x^3);
                                           0

> T(x->x^4);
                                          -1
                                          ---
                                          270

> quit
bytes used=526672, alloc=458668, time=0.02
```

which shows the method is fourth order.

$$\textbf{(ii)}\quad \int_0^1 f(\tau)d\tau \approx \frac{2}{3}f\Big(\frac{1}{4}\Big) - \frac{1}{3}f\Big(\frac{1}{2}\Big) + \frac{2}{3}f\Big(\frac{3}{4}\Big)$$

Following the same pattern as above, the Maple script

```
restart;
kernelopts(printbytes=false):
T:=f->int(f(tau),tau=0..1)-(2/3*f(1/4)-1/3*f(1/2)+2/3*f(3/4));
T(x->1);
T(x->x);
T(x->x^2);
T(x->x^3);
T(x->x^4);
```

produces the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> T:=f->int(f(tau),tau=0..1)-(2/3*f(1/4)-1/3*f(1/2)+2/3*f(3/4));
                     1
                    /
                   |
       T := f ->   |   f(tau) dtau - 2/3 f(1/4) + 1/3 f(1/2) - 2/3 f(3/4)
                   |
                  /
                    0
```

```
> T(x->1);
                                       0

> T(x->x);
                                       0

> T(x->x^2);
                                       0

> T(x->x^3);
                                       0

> T(x->x^4);
                                     7/960

> quit
bytes used=525784, alloc=458668, time=0.01
```

which shows this method is also fourth order.

7. Let $f_n = f(x_n)$ where $x_n = nh$ and f is a smooth function. Find values for the constants A, B, C and D such that the approximation

$$f''(x_n) \approx \frac{1}{h^2}\big(Af_{n+3} + Bf_{n+2} + Cf_{n+1} + Df_n\big)$$

is as accurate as possible. What is the order of the approximation?

Define

$$\tau(h) = h^2 f''(x) - (Af(x+3h) + Bf(x+2h) + Cf(x+h) + Df(x))$$

and then expand τ in Taylor series about $h = 0$. We then solve for A, B, C and D so that as many of the low-order terms of the power series vanish as possible. Since there are four unknowns, it is generally possible to make the four coefficients of 1, h, h^2 and h^3 vanish. The order of the method would then be h^4/h^2 or order 2. The Maple script

```
restart;
kernelopts(printbytes=false):
T:=h^2*diff(f(x),x$2)-(A*f(x+3*h)+B*f(x+2*h)+C*f(x+h)+E*f(x));
S:=series(T,h,5);
eq1:=simplify(coeff(S,h,0)/f(x))=0;
eq2:=simplify(coeff(S,h,1)/diff(f(x),x))=0;
eq3:=simplify(coeff(S,h,2)/diff(f(x),x$2))=0;
eq4:=simplify(coeff(S,h,3)/diff(f(x),x$3))=0;
eq5:=simplify(coeff(S,h,4)/diff(f(x),x$4))=0;
S2:=solve({eq1,eq2,eq3,eq4},{A,B,C,E});
subs(S2,eq5);
```

Computes the series expansion of τ and then solves for the constants. Note that the constant D has been replaced by E in the script because Maple reserves the letter D for derivative. The output of the script is

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> T:=h^2*diff(f(x),x$2)-(A*f(x+3*h)+B*f(x+2*h)+C*f(x+h)+E*f(x));
              / 2      \
            2 |d       |
   T := h  |--- f(x)| - A f(x + 3 h) - B f(x + 2 h) - C f(x + h) - E f(x)
              |  2     |
              \dx      /

> S:=series(T,h,5);
S := (-C f(x) - A f(x) - E f(x) - B f(x)) +

    (-2 B D(f)(x) - 3 A D(f)(x) - C D(f)(x)) h +

    /                                       / 2      \                     \
    |       (2)                   (2)          |d       |           (2)       |
    |-2 B (D   )(f)(x) - 9/2 A (D   )(f)(x) + |--- f(x)| - 1/2 C (D   )(f)(x)|
    |                                       |  2     |                     |
    \                                       \dx      /                     /

     2              (3)                    (3)                    (3)          3
    h  + (-9/2 A (D   )(f)(x) - 1/6 C (D   )(f)(x) - 4/3 B (D   )(f)(x)) h  +

              (4)                   (4)                     (4)          4
    (-27/8 A (D   )(f)(x) - 2/3 B (D   )(f)(x) - 1/24 C (D   )(f)(x)) h  +

       5
    O(h )

> eq1:=simplify(coeff(S,h,0)/f(x))=0;
                          eq1 := -C - A - E - B = 0

> eq2:=simplify(coeff(S,h,1)/diff(f(x),x))=0;
                          eq2 := -2 B - 3 A - C = 0

> eq3:=simplify(coeff(S,h,2)/diff(f(x),x$2))=0;
                                       9 A
                       eq3 := -2 B - --- + 1 - C/2 = 0
                                        2

> eq4:=simplify(coeff(S,h,3)/diff(f(x),x$3))=0;
                                   9 A         4 B
                         eq4 := - --- - C/6 - --- = 0
                                    2           3
```

```
> eq5:=simplify(coeff(S,h,4)/diff(f(x),x$4))=0;
                                 27 A   2 B    C
                     eq5 := - ---- - --- - ---- = 0
                                  8      3     24

> S2:=solve({eq1,eq2,eq3,eq4},{A,B,C,E});
                  S2 := {A = -1, E = 2, B = 4, C = -5}

> subs(S2,eq5);
                                     11
                                     -- = 0
                                     12

> quit
bytes used=1401116, alloc=1179432, time=0.04
```

which shows the choice

$$A = -1, \qquad B = 4, \qquad C = -5 \qquad \text{and} \qquad D = 2$$

obtains a method of order 2. The script ends by checking that the above choice of constants yields the non-zero value of 11/12 for the coefficient on h^4, which verifies the method is no greater than second order.

8. Find weights w_1, w_2 and w_3 such that the approximation formula

$$\int_0^3 f(x)dx \approx w_1 f(0) + w_2 f(2) + w_3 f(4)$$

is exact for all polynomials of degree 2. Comment on the fact that $f(4)$ is evaluated outside the interval of integration.

The Maple script

```
restart;
kernelopts(printbytes=false):
T:=f->int(f(tau),tau=0..3)-(w1*f(0)+w2*f(2)+w3*f(4));
eq1:=T(x->1)=0;
eq2:=T(x->x)=0;
eq3:=T(x->x^2)=0;
solve({eq1,eq2,eq3},{w1,w2,w3});
```

produces the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
```

```
> T:=f->int(f(tau),tau=0..3)-(w1*f(0)+w2*f(2)+w3*f(4));
                            3
                           /
                          |
              T := f ->   |   f(tau) dtau - w1 f(0) - w2 f(2) - w3 f(4)
                          |
                         /
                          0

> eq1:=T(x->1)=0;
                          eq1 := 3 - w1 - w2 - w3 = 0

> eq2:=T(x->x)=0;
                          eq2 := 9/2 - 2 w2 - 4 w3 = 0

> eq3:=T(x->x^2)=0;
                          eq3 := 9 - 4 w2 - 16 w3 = 0

> solve({eq1,eq2,eq3},{w1,w2,w3});
                          {w3 = 0, w1 = 3/4, w2 = 9/4}

> quit
bytes used=769468, alloc=655240, time=0.04
```

which shows that

$$w_1 = \frac{3}{4}, \qquad w_2 = \frac{9}{4} \qquad \text{and} \qquad w_3 = 0$$

is the unique choice of w_i's such that the approximation formula is exact for all polynomial of degree 2. It is interesting that the $w_3 = 0$. This is a coincidence and not because 4 was outside the interval. In general, we would expect $w_3 \neq 0$. For example, finding the weights η_i such that

$$\int_0^3 f(x)dx \approx \eta_1 f(0) + \eta_2 f(3) + \eta_3 f(4)$$

is exact for polynomials of degree two gives that

$$\eta_1 = \frac{9}{8}, \qquad \eta_2 = 3 \qquad \text{and} \qquad \eta_3 = -9/8.$$

Note that quadrature formula whose weights have minus signs in them are generally less stable numerically than formula consisting of all positive weights. In this case the weight η_3 for the function evaluated outside in interval of integration is negative. For reference the above calculations were performed by the Maple script

```
restart;
kernelopts(printbytes=false):
T:=f->int(f(tau),tau=0..3)-(eta1*f(0)+eta2*f(3)+eta3*f(4));
eq1:=T(x->1)=0;
eq2:=T(x->x)=0;
eq3:=T(x->x^2)=0;
solve({eq1,eq2,eq3},{eta1,eta2,eta3});
```

which produced the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> T:=f->int(f(tau),tau=0..3)-(eta1*f(0)+eta2*f(3)+eta3*f(4));
                          3
                         /
                        |
         T := f ->      |   f(tau) dtau - eta1 f(0) - eta2 f(3) - eta3 f(4)
                        |
                       /
                         0

> eq1:=T(x->1)=0;
                        eq1 := 3 - eta1 - eta2 - eta3 = 0

> eq2:=T(x->x)=0;
                         eq2 := 9/2 - 3 eta2 - 4 eta3 = 0

> eq3:=T(x->x^2)=0;
                         eq3 := 9 - 9 eta2 - 16 eta3 = 0

> solve({eq1,eq2,eq3},{eta1,eta2,eta3});
                       {eta1 = 9/8, eta3 = -9/8, eta2 = 3}

> quit
bytes used=770212, alloc=655240, time=0.02
```

9. Suppose $y' = f(y,t)$ and define $g(s) = f\big(y(s),s\big)$. Write the 2nd order Taylor polynomial given by

$$p_2(s) = g(t_n) + g'(t_n)(s-t_n) + \frac{1}{2}g''(t_n)(s-t_n)^2$$

in terms of the function f.

Calculating yields

$$\begin{aligned}
g(t_n) &= f\big(y(t_n),t_n\big)\\
g'(t_n) &= f_y\big(y(t_n),t_n\big)y'(t_n) + f_t\big(y(t_n),t_n\big)\\
&= f_y\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big) + f_t\big(y(t_n),t_n\big)\\
g''(t_n) &= f_{yy}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big)^2 + f_{yt}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big)\\
&\quad + f_y\big(y(t_n),t_n\big)^2 f\big(y(t_n),t_n\big) + f_y\big(y(t_n),t_n\big)f_t\big(y(t_n),t_n\big)
\end{aligned}$$

$$\begin{aligned}
&\quad + f_{yt}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big) + f_{tt}\big(y(t_n),t_n\big)\\
&= f_{yy}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big)^2 + 2f_{yt}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big)\\
&\quad + f_y\big(y(t_n),t_n\big)^2 f\big(y(t_n),t_n\big) + f_y\big(y(t_n),t_n\big)f_t\big(y(t_n),t_n\big)\\
&\quad + f_{tt}\big(y(t_n),t_n\big)
\end{aligned}$$

Therefore

$$\begin{aligned}
p_2(s) = f\big(y(t_n),t_n\big) &+ \Big\{f_y\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big) + f_t\big(y(t_n),t_n\big)\Big\}(s-t_n)\\
&+ \frac{1}{2}\Big\{f_{yy}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big)^2 + 2f_{yt}\big(y(t_n),t_n\big)f\big(y(t_n),t_n\big)\\
&\quad + f_y\big(y(t_n),t_n\big)^2 f\big(y(t_n),t_n\big) + f_y\big(y(t_n),t_n\big)f_t\big(y(t_n),t_n\big)\\
&\quad + f_{tt}\big(y(t_n),t_n\big)\Big\}(s-t_n)^2
\end{aligned}$$

assuming I didn't make a typo typing it in. For reference, the calculation can also be done with Maple

```
restart;
kernelopts(printbytes=false):
g:=s->f(y(s),s);
p2:=g(t_n)+D(g)(t_n)(s-t_n)+(1/2)*(D@@2)(g)(t_n)(s-t_n)^2;
```

with output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> g:=s->f(y(s),s);
                              g := s -> f(y(s), s)

> p2:=g(t_n)+D(g)(t_n)(s-t_n)+(1/2)*(D@@2)(g)(t_n)(s-t_n)^2;
p2 := f(y(t_n), t_n) + D[1](f)(y(t_n), t_n)(s - t_n) %1

     + D[2](f)(y(t_n), t_n)(s - t_n) + 1/2 (

    (D[1, 1](f)(y(t_n), t_n)(s - t_n) %1 + D[1, 2](f)(y(t_n), t_n)(s - t_n)) %1

                                             (2)
     + D[1](f)(y(t_n), t_n)(s - t_n) (D    )(y)(t_n)(s - t_n)

                                                                                2
     + D[1, 2](f)(y(t_n), t_n)(s - t_n) %1 + D[2, 2](f)(y(t_n), t_n)(s - t_n))

%1 := D(y)(t_n)(s - t_n)
```

```
> quit
bytes used=1175304, alloc=982860, time=0.05
```

10. Suppose $y' = f(y)$ where

$$f(y) = \begin{bmatrix} -y_2 - y_3 \\ y_1 + ay_2 \\ b + y_3(y_1 - c) \end{bmatrix}$$

with a, b and c constant. Find $\frac{d}{dt} f(y(t))$.

The Maple script

```
restart;
kernelopts(printbytes=false):
f:=[-y2(t)-y3(t),y1(t)+a*y2(t),b+y3(t)*(y1(t)-c)];
S:=[seq(diff(y||k(t),t)=f[k],k=1..3)];
r1:=diff(f,t);
r2:=subs(S,r1);
r3:=subs({y1(t)=y1,y2(t)=y2,y3(t)=y3},r2);
Vector(r3);
```

produced the output

```
    |\^/|     Maple 9.5 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2004
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> restart;
> kernelopts(printbytes=false):
> f:=[-y2(t)-y3(t),y1(t)+a*y2(t),b+y3(t)*(y1(t)-c)];
       f := [-y2(t) - y3(t), y1(t) + a y2(t), b + y3(t) (y1(t) - c)]

> S:=[seq(diff(y||k(t),t)=f[k],k=1..3)];
      d                          d
S := [-- y1(t) = -y2(t) - y3(t), -- y2(t) = y1(t) + a y2(t),
      dt                         dt

    d
    -- y3(t) = b + y3(t) (y1(t) - c)]
    dt

> r1:=diff(f,t);
       /d       \   /d       \  /d       \     /d       \
r1 := [-|-- y2(t)| - |-- y3(t)|, |-- y1(t)| + a |-- y2(t)|,
       \dt      /   \dt      /  \dt      /     \dt      /

    /d       \                      /d       \
    |-- y3(t)| (y1(t) - c) + y3(t) |-- y1(t)|]
    \dt      /                      \dt      /
```

```
> r2:=subs(S,r1);
r2 := [-y1(t) - a y2(t) - b - y3(t) (y1(t) - c),

    -y2(t) - y3(t) + a (y1(t) + a y2(t)),

    (b + y3(t) (y1(t) - c)) (y1(t) - c) + y3(t) (-y2(t) - y3(t))]

> r3:=subs({y1(t)=y1,y2(t)=y2,y3(t)=y3},r2);
r3 := [-y1 - a y2 - b - y3 (y1 - c), -y2 - y3 + a (y1 + a y2),

    (b + y3 (y1 - c)) (y1 - c) + y3 (-y2 - y3)]

> Vector(r3);
                   [       -y1 - a y2 - b - y3 (y1 - c)       ]
                   [                                          ]
                   [         -y2 - y3 + a (y1 + a y2)         ]
                   [                                          ]
                   [(b + y3 (y1 - c)) (y1 - c) + y3 (-y2 - y3)]

> quit
bytes used=526628, alloc=458668, time=0.02
```

Therefore the answer is

$$\frac{d}{dt}f(y(t)) = \begin{bmatrix} -y_1 - ay_2 - b - y_3(y_1 - c) \\ -y_2 - y_3 + a(y_1 + ay_2) \\ \big(b + y_3(y_1 - c)\big)(y_1 - c) - y_3(y_2 + y_3) \end{bmatrix}.$$

11. Consider the two-point boundary value problem

$$y'' = p(x)y' + q(x)y + r(x) \qquad \text{where} \qquad y(a) = \alpha \quad \text{and} \quad y(b) = \beta.$$

Use the finite differences

$$y'(x) \approx \frac{y(x+h) - y(x-h)}{2h} \qquad \text{and} \qquad y''(x) \approx \frac{y(x+h) - 2y(x) + y(x-h)}{h^2}.$$

to construct matrix $\mathbf{A}$ and vector $\mathbf{b}$ such that the vector $\mathbf{y} = (y_1, \ldots, y_{n-1})$ which solves $\mathbf{Ay} = \mathbf{b}$ approximates the solution to the differential equation as $y_j \approx y(x_j)$ where $x_j = a + jh$ and $h = (b-a)/n$. What is the order of the approximation?

Plugging the finite difference approximations into the differential equation and writing $p_j = p(x_j)$, $q_j = q(x_j)$ and $r_j = r(x_j)$ gives

$$\frac{y_{j+1} - 2y_j + y_{j-1}}{h^2} = p_j \frac{y_{j+1} - y_{j-1}}{2h} + q_j y_j + r_j.$$

Further clearing the denominator

$$\big(2 - hp_j\big)y_{j+1} - \big(4 + 2h^2 q_j\big)y_j + \big(2 + hp_j\big)y_{j-1} = 2h^2 r_j.$$

Since $y_0 = \alpha$ and $y_n = \beta$, then in matrix form we have

$$\mathbf{A} = \begin{bmatrix} -4-2h^2q_1 & 2-hp_1 & 0 & \cdots & 0 \\ 2+hp_2 & -4-2h^2q_2 & 2-hp_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 2+hp_{m-2} & -4-2h^2q_{m-2} & 2-hp_{m-2} \\ 0 & \cdots & 0 & 2+hp_{m-1} & -4-2h^2q_{m-1} \end{bmatrix}$$

and

$$\mathbf{b} = \begin{bmatrix} 2h^2r_1 - (2+hp_1)\alpha \\ 2h^2r_2 \\ \vdots \\ 2h^2r_{m-2} \\ 2h^2r_{m-1} - (2-hp_{m-1})\beta \end{bmatrix}.$$

Please observe that that the boundary conditions have been placed in the first and last components of the vector $\mathbf{b}$. To compute the order note by Taylor's theorem that

$$y(x \pm h) = y(x) \pm hy'(x) + \frac{h^2}{2}y''(x) \pm \frac{h^3}{6}y'''(\xi_\pm)$$

for some $\xi_+ \in (x, x+h)$ and $\xi_- \in (x-h, x)$. Therefore

$$\begin{aligned} y'(x) - \frac{y(x+h)-y(x-h)}{2h} &= y'(x) - \frac{2hy'(x) + (h^3/6)(y'''(\xi_+)+y'''(\xi_-))}{2h} \\ &= \frac{h^2}{12}\big((y'''(\xi_+)+y'''(\xi_-)\big) = \mathcal{O}(h^2) \end{aligned}$$

Expanding Taylor's series one term more yields

$$y(x \pm h) = y(x) \pm hy'(x) + \frac{h^2}{2}y''(x) \pm \frac{h^3}{6}y'''(x) + \frac{h^4}{24}y^{(4)}(\eta_\pm)$$

for some $\eta_+ \in (x, x+h)$ and $\eta_- \in (x-h, x)$. Therefore

$$\begin{aligned} y''(x) - \frac{y(x+h)-2y(x)+y(x-h)}{h^2} &= y''(x) - \frac{h^2y''(x) + (h^4/24)(y^{(4)}(\eta_+)+y^{(4)}(\eta_-))}{h^2} \\ &= \frac{h^2}{24}\big((y^{(4)}(\eta_+)+y^{(4)}(\eta_-)\big) = \mathcal{O}(h^2). \end{aligned}$$

As the finite differences for $y'(x)$ and $y''(x)$ are both second order, the approximation of the two-point boundary value problem is also second order.