ADVANCED NUMERICAL METHODS
BACHELOR'S DEGREE IN INDUSTRIAL TECHNOLOGY ENGINEERING
JUNE 19, 2017

TIME: 3 hours **30 points total**

EXAM ANSWER

1a) A Taylor polynomial is a truncated Taylor series expansion without its error term. It is also an osculating polynomial with interpolation data at one only multiple node. For the degree of the polynomial to be ≤ 3 we need 4 conditions (4 interpolation data). At $x_0 = 0$ these conditions will be that the values of p_3 and its 1st, 2nd and 3rd derivatives coincide with those of f. These are:

$$f(x) = \cos(2x) \quad \rightarrow \quad f(0) = 1$$
$$f'(x) = -2\sin(2x) \quad \rightarrow \quad f'(0) = 0$$
$$f''(x) = -4\cos(2x) \quad \rightarrow \quad f''(0) = -4$$
$$f^{3)}(x) = 8\sin(2x) \quad \rightarrow \quad f^{3)}(0) = 0$$

So the interpolation data are $p_3(0) = 1,\ \ p_3'(0) = 0,\ \ p_3''(0) = -4,\ \ p_3^{3)}(0) = 0$. **(1p)**

The corresponding table of *divided differences with repetitions* (using Octave) is:

```
[fii, zi, table] = anm_tableddr(0, [1 0 -4 0]); disp(table)
    i    zi    fi    fi1   fi2   fi3
    -    --    --    ---   ---   ---
    0    0     1
    1    0     1     0
    2    0     1     0     -2
    3    0     1     0     -2    0
```

This is also very easy to do "by hand"—both "−2" of the column of second-order differences are $f_{2,2} = f_{2,3} = f''(0)/2! = -4/2 = -2$. **(1p)**

The corresponding osculating polynomial (Taylor polynomial or Taylor truncated series) is:

```
[pp, coefs, chars] = anm_newton([], 0, [1 0 -4 0]); chars{1:2}
    p3(x) = 1 + 0*(x-0) + -2*(x-0)*(x-0) + 0*(x-0)*(x-0)*(x-0)
    p3(x) = ((0*(x-0) + -2)*(x-0) + 0)*(x-0) + 1
```

or $$\boxed{p_3(x) = 1 - 2x^2}$$ **(0.5p)**

1b) The error term of osculating polynomials is, in general:

$$\boxed{e(x) = \frac{f^{k)}(\xi)}{k!}(x-x_0)^{k_0}(x-x_1)^{k_1}\cdots(x-x_n)^{k_n}}$$ **(1p)**

for some ξ between the nodes and x (or in an interval $[a, b]$ where the nodes and ξ are and where $f \in C^k$; ξ depends on x). k_i is the number of conditions at node x_i, and k is the total number of conditions $= \Sigma\, k_i$. In our case $k = k_0 = 4$ at the only node $x_0 = 0$, so:

$$e(x) = \frac{f^{4)}(\xi)}{4!}(x-0)^4 = \frac{16\cos(2\xi)}{4!}x^4 = \boxed{\frac{2}{3}\cos(2\xi)x^4} \quad \text{for some } \xi \text{ between 0 and } x$$

which, as expected, is precisely the remainder of order 4 of the Taylor series expansion (in its Lagrange representation, i.e. $f^{4)}(\xi)\,x^4/4!$) —remember that $e(x) = f(x) - p_3(x)$. **(0.5p)**

1c) If I notice that $p_3(x) = 1 - 2x^2$ satisfies $p_3(-1) = p_3(1) = -1$, I will immediately know that $p_5(x) = p_3(x)$ (because of the uniqueness of osculating polynomials). However, I am explicitly asked to complete the table of section a) and don't have much time for noticing things. I will place the info from the two new nodes at the bottom (two final rows, corresponding to $i = 4, 5$). This is, again, very easy to do "by hand". With Octave:

```
[fii,zi,table] = anm_tableddr([0 -1 1], [1 0 -4 0; ...
                 -1 NaN NaN NaN; -1 NaN NaN NaN]); disp(table)
    warning: Nodes are not sorted. But I'm sure you know what you're doing.
    warning: called from anm_tableddr at line 58 column 5
    i    zi    fi    fi1    fi2    fi3    fi4    fi5
    -    --    --    ---    ---    ---    ---    ---
    0     0     1
    1     0     1     0
    2     0     1     0     -2
    3     0     1     0     -2     0
    4    -1    -1     2     -2     0      0
    5     1    -1     0     -2     0      0      0
```

To reuse the whole table of section a) unchanged I could also have added the two new nodes −1 and 1 at the beginning of the original table, or one node at the beginning and the other one at the end, because the table is of *divided* differences, and permutations of the nodes do not alter divided differences[1].

So now, regardless of whether I had noticed it before or not, I see that the new principal divided differences are 0 and 0, so no new term has to be added, and:

$$\boxed{p_5(x) \equiv p_3(x)} \qquad \textbf{(1p)}$$

Since the new polynomial is the same as the old one, one would be tempted to think that its error term should also be the same. But it is not, because f is different now, and $e(x)$ is the error of $p_3(x)$ *with respect to a specific function* $f(x)$. Indeed, the two new interpolation points are not satisfied by $f(x)=\cos(2x)$: $f(-1)=-0.41615\neq-1$ and $f(1)=-0.41615\neq-1$. Hence,

$$\underline{p_5 \text{ does not interpolate } f(x)=\cos(2x)} \qquad \textbf{(0.5p)}$$

(although it will be very close at values of x close to $x_0=0$). To obtain the new error term we just particularize the general one of section c) with $x_0=0$, $k_0=4$, $x_1=-1$, $x_2=1$, $k_1=k_2=1$, $k=6$, getting:

$$\boxed{e(x)=\frac{f^{6)}(\xi)}{6!}x^4(x+1)(x-1)=\frac{f^{6)}(\xi)}{720}x^4(x^2-1)}$$

for some ξ between the nodes and x. **(0.5p)**

2) Under quite general smoothness conditions ($f\in C^1$), the maximum signal will occur at a stationary point x, i.e. where $f'(x)=0$. The second table shows that $f'(x)$ goes from positive to negative in an apparently monotonic fashion. We must estimate the intermediate point x where $f'(x)=0$ (which should lie between 3.75 and 6.25; remember to check this at the end).

One could calculate the polynomial $p_3(x)$ interpolating $f'(x)$ using the four nodes of that 2nd table, but then we would have to find its root. In this case the polynomial's degree is ≤ 3, so there exists an exact, closed formula for that[2]; however, finding a root of a polynomial is typically more "difficult" than just evaluating it (and, in any case, we are asked to find x by *evaluating* a polynomial).

So to estimate x by evaluating a polynomial we will use *inverse interpolation*. We will interchange the abscissas and the ordinates of the second table (checking that the new abscissas are monotonic—this is important), calculate the corresponding interpolation polynomial q_3, and evaluate it at 0. **(1p)**

The new nodes are not equally spaced anymore, so we will use *divided* differences to calculate the inverse-interpolation polynomial. With Octave, calling X_i / Y_i the new abscissas / ordinates:

[1] Had it been a table of *finite* differences, the sorting order would have had to be respected; however, osculating polynomials are not constructed with finite differences, because multiple nodes are like nodes infinitely close to one another, and hence not equally-spaced.

[2] For the quadratic equation there is the well-known $x=[-b\pm\text{sqrt}(b^2-4ac)]/(2a)$. For the cubic and quartic equations there are similar expressions—albeit larger, and forking into 3 and 4 values, respectively. But the general quintic equation does not have a closed-form solution (although many particular cases do). The roots of higher-degree polynomials can be found using iterative methods (secant, Newton-Raphson, etc.).

```
format long g
Xi = [0.56640 0.07029 -0.27079 -0.41701];  % new abscissas = old ordinates
Yi = [1.25    3.75     6.25     8.75];     % new ordinates = old abscissas
[fii, Xi, table] = anm_tableddr(Xi', Yi'); disp(table)
    warning: Nodes are not sorted. But I'm sure you know what you're doing.
    warning: called from anm_tableddr at line 58 column 5
    i         xi    fi1              fi2             fi3              fi4
    -         --    ---              ---             ---              ---
    0    0.56640   1.25
    1    0.07029   3.75    -5.0392050150
    2   -0.27079   6.25    -7.3296587311    2.7358827937
    3   -0.41701   8.75   -17.0975242785   20.0448708135   -17.6009884176
```

The inverse-interpolation polynomial $q_3(X)$ and its value at $X = 0$ are: **(1p)**

```
[x, coefs, chars] = anm_newton(0, Xi', Yi', 9); chars, x
    warning: Nodes are not sorted. But I'm sure you know what you're doing.
    p3(x) = 1.25 + -5.03920502*(x-0.5664) + 2.73588279*(x-0.5664)*
        (x-0.07029) + -17.6009884*(x-0.5664)*(x-0.07029)*(x--0.27079)
    p3(x) = ((-17.6009884*(x--0.27079) + 2.73588279)*(x-0.07029) +
        -5.03920502)*(x-0.5664) + 1.25
    p3(0) = ((-17.6009884*(0--0.27079) + 2.73588279)*(0-0.07029) +
        -5.03920502)*(0-0.5664) + 1.25
    p3(0) = ((-17.6009884*0.27079 + 2.73588279)*-0.07029 + -5.03920502)*
        -0.5664 + 1.25
    x =     4.02337534066208
```

Hence the polynomial is: $q_3(X) = 1.25 - 5.03920502\,(X - 0.5664) +$ **(1p)**

$$+\,2.73588279\,(X-0.5664)(X-0.07029) - 17.6009884\,(X-0.5664)(X-0.07029)(X+0.27079)$$

and its value at $X = 0$, evaluated optimally (i.e. using the Hörner-like algorithm) is:

$$q_3(X=0) = [(-17.6009884\cdot 0.27079 + 2.73588279)\,(-0.07029) - 5.03920502]\,(-0.5664) + 1.25 = \underline{x = 4.02337534}$$

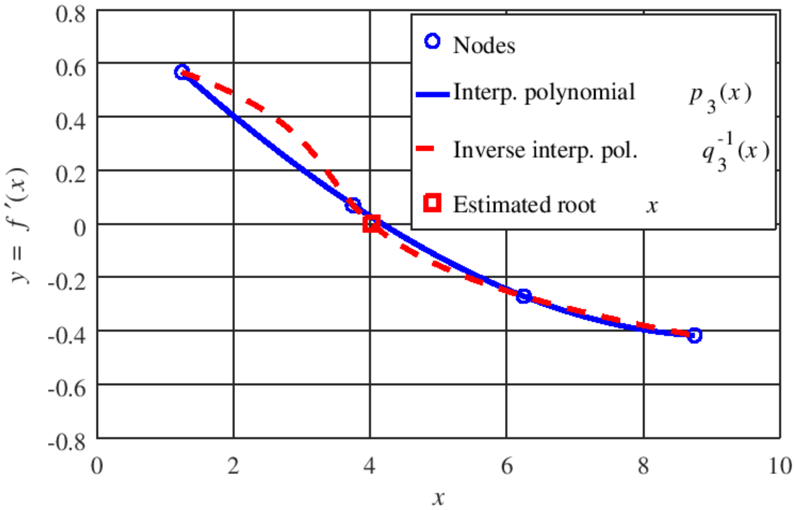
This lies indeed between 3.75 and 6.25, as expected. **(0.5p)**

Of course it is also possible (but unnecessary) to sort the new nodes X_i in ascending order.

Finally let us plot $p_3(x)$ and $q_3(X)$. In order to plot them together, I will interchange abscissas and ordinates for q_3. The best way to understand the figure precisely is to read the Octave code that generates it:

```
close all, figure, hold on, format short g
xi = Yi'; yi = Xi;                          % keeping previous variables
plot(xi, yi, 'ob', 'LineWidth', 1)
p3 = @(xx) anm_newton(xx, xi, yi);
q3 = @(yy) anm_newton(yy, yi, xi);
xx = linspace(xi(1), xi(end));
yy = linspace(yi(1), yi(end));
plot(xx, p3(xx), 'b',   'LineWidth', 2)
plot(q3(yy), yy, 'r--', 'LineWidth', 2)
plot(x, 0,       'sr',  'LineWidth', 2)
xlabel('\itx')
ylabel('{\ity} = {\itf \prime}({\itx})')
legend('Nodes', ...
    'Interp. polynomial {\itp}_3({\itx})', ...
    'Inverse interp. pol. {\itq}_3^{-1}({\itx})', ...
    'Estimated root \itx')
```

The output is:



where, of course, $q_3^{-1}(x)$ does not represent $1/q_3(x)$, but the inverse function of $q_3(y)$.

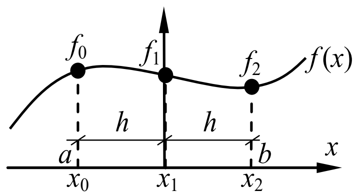
One could argue that the inverse interpolation seems to introduce artificial oscillations, or that $f'(x)$ is probably more like the solid blue line than like the red dashed one. We don't really know. If true, the maximum value of $f(x)$ is probably attained at a value of x slightly more to the right (root of p_3). Finally, to plot with interchanged abscissas/ordinates you can execute this code:

```
figure, hold on
plot(yi, xi,       'ob',  'LineWidth', 1)
plot(p3(xx), xx, 'b',   'LineWidth', 2)
plot(yy, q3(yy), 'r--', 'LineWidth', 2)
plot(0, x,         'sr',  'LineWidth', 2)
```

3a) We will integrate the interpolation polynomial both in its Lagrange and in its Newton representation with equally-spaced nodes. We are only asked to do it "by integrating an interpolation polynomial", so both answers comply.

The basic interpolatory idea is: $$I = \int_a^b f(x)dx \simeq \int_a^b p(x)dx = Q$$

The Simpson formula is, by definition, the closed Newton-Cotes one of three nodes, so $x_0 = a$, $x_1 = a + h = b - h$, and $x_2 = b$:



In this case: $$Q = \int_{x_0}^{x_2} p_2(x)dx$$

Let us first integrate the Lagrange representation of $p_2(x)$—a more "theoretically modest" option:

$$p_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f_2$$

Integrating: $$Q = \int_{x_0}^{x_2} \left(\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f_2 \right) dx =$$

$$= \underbrace{\left(\int_{x_0}^{x_2} \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} dx \right)}_{w_0} f_0 + \underbrace{\left(\int_{x_0}^{x_2} \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} dx \right)}_{w_1} f_1 + \underbrace{\left(\int_{x_0}^{x_2} \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} dx \right)}_{w_2} f_2$$

To take advantage of some symmetries, these 3 integrals will be easier to calculate with this change of variable: $x = x_1 + ht; \quad dx = h\,dt$

The 1st weight w_0 is:

$$w_0 = \int_{x_0}^{x_2} \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}dx = \int_{-1}^{1}\frac{ht\,h(t-1)}{(-h)(-2h)}hdt = \frac{h}{2}\int_{-1}^{1}(t^2-t)dt = \frac{h}{2}\int_{-1}^{1}t^2dt = \frac{h}{2}2\int_0^1 t^2dt = h\frac{1^3}{3} = \frac{h}{3}$$

Similarly, for the 3rd weight, $w_2 = h/3$ too. As for the 2nd weight w_1:

$$w_1 = \int_{-1}^{1}\frac{h(t+1)\,h(t-1)}{h\,(-h)}hdt = -h\int_{-1}^{1}(t^2-1)dt = -h2\int_0^1(t^2-1)dt = -2h\left(\frac{1^3}{3}-1\right) = -2h\frac{-2}{3} = \frac{4h}{3}$$

Substituting:
$$Q = w_0f_0 + w_1f_1 + w_2f_2 = \boxed{\frac{h}{3}\left(f_0+4f_1+f_2\right)} \qquad \textbf{(1.75p)}$$

Now let's write and integrate the Newton representation of $p_2(x)$ with equally-spaced nodes. Here the customary change of variable is $x = x_0 + ht$
and then $p_2(x) = q(t(x))$

where
$$q(t) = \binom{t}{0}f_0 + \binom{t}{1}\Delta f_0 + \binom{t}{2}\Delta^2 f_0 = f_0 + t(f_1-f_0) + \frac{t(t-1)}{2!}(\Delta f_1 - \Delta f_0) =$$
$$= (1-t)f_0 + tf_1 + \frac{t^2-t}{2}[(f_2-f_1)-(f_1-f_0)] =$$
$$= (1-t)f_0 + tf_1 + \frac{t^2-t}{2}(f_2-2f_1+f_0) = \left(1-\frac{3t}{2}+\frac{t^2}{2}\right)f_0 + \left(2t-t^2\right)f_1 + \left(\frac{t^2}{2}-\frac{t}{2}\right)f_2$$

Integrating:
$$Q = \int_{x_0}^{x_2} q(t(x))dx = \int_0^2 q(t)\,hdt = h\int_0^2\left[\left(1-\frac{3t}{2}+\frac{t^2}{2}\right)f_0 + \left(2t-t^2\right)f_1 + \left(\frac{t^2}{2}-\frac{t}{2}\right)f_2\right]dt =$$
$$= \left[h\int_0^2\left(1-\frac{3t}{2}+\frac{t^2}{2}\right)dt\right]f_0 + \left[h\int_0^2\left(2t-t^2\right)dt\right]f_1 + \left[h\int_0^2\left(\frac{t^2}{2}-\frac{t}{2}\right)dt\right]f_2$$

Hence:
$$w_0 = h\left[t-\frac{3t^2}{4}+\frac{t^3}{6}\right]_0^2 = h\left(2-3+\frac{8}{6}\right) = \frac{h}{3}$$
$$w_1 = h\left[t^2-\frac{t^3}{3}\right]_0^2 = h\left(4-\frac{8}{3}\right) = \frac{4h}{3}$$
$$w_2 = h\left[\frac{t^3}{6}-\frac{t^2}{4}\right]_0^2 = h\left(\frac{8}{6}-1\right) = \frac{h}{3}$$

with the same result as before.

As for the truncation error term E, it is easily obtained by integrating the first monomial (of $1, x, x^2, x^3\ldots$) that is not integrated exactly by the rule, i.e., by integrating x^{N+1} where N is the rule's polynomial degree, and then isolating K from its known expression $E = Kf^{N+1)}(\xi)$ f. s. $\xi \in [a,b]$.

With three nodes, $N \geq 2$ by construction (remember we were integrating $p_2(x)$ before); but since the Simpson rule is a Newton-Cotes one with an odd number of nodes, we gain one extra unit over that minimum, resulting in $N = 3$. If we didn't remember this fact, integrating x^3 will remind us of it. Without loss of generality, the calculations will be simpler if we place the origin of abscissas at the central node, so $x_0 = -h$, $x_1 = 0$, $x_2 = h$. With $f(x) = x^3$:

$$\int_{-h}^{h} x^3dx = 0 = Q+E = \frac{h}{3}(f_0+4f_1+f_2)+E = \frac{h}{3}\left((-h)^3+4\cdot0^3+h^3\right) = 0+E \quad\Rightarrow\quad E=0$$

and the error E being 0 means that x^3 is integrated exactly and $N \geq 3$. Let's now try with $f(x) = x^4$:

$$\int_{-h}^{h} x^4dx = 2\frac{h^5}{5} = Q+E = \frac{h}{3}(f_0+4f_1+f_2)+E = \frac{h}{3}\left((-h)^4+4\cdot0^4+h^4\right) = \frac{2h^5}{3}+E \quad\Rightarrow\quad E\neq 0$$

The error E being non-zero means that x^4 is not integrated exactly, so $N < 4$, and therefore $N = 3$. Hence the form of the error term is $E = Kf^{N+1)}(\xi) = Kf^{4)}(\xi)$ for some $\xi \in [a,b]$. For $f(x) = x^4$,

$f^{4)}(x) = 4! = 24$. Substituting:

$$2\frac{h^5}{5} = \frac{2h^5}{3} + K\cdot 24 \quad\Rightarrow\quad K = \frac{h^5}{24}\left(\frac{2}{5}-\frac{2}{3}\right) = \frac{h^5}{24}\,\frac{6-10}{15} = \frac{-h^5}{90}$$

Because $f^{4)}(x)$ is constant for $f(x) = x^4$, ξ does not appear in it and K could be isolated in terms of h alone. But the expression $E = Kf^{4)}(\xi)$ is valid not only for $f(x) = x^4$, so substituting $K = -h^5/90$ into E we finally obtain the error term of the simple Simpson rule:

$$\boxed{E = \frac{-h^5 f^{4)}(\xi)}{90}} \qquad \underline{\text{f. s. } \xi \in (a,b)} \qquad \textbf{(1.75p)}$$

The subject's function **anm_nc** confirms that these results are correct:

```
[Q, h, xi, wi, Es, Ks, Ec, Kc, N] = anm_nc(3, 1); Q, Es, Ec, N
    Q = Simpson's rule: Q = h * (1/3 f(x0) + 4/3 f(x1) + 1/3 f(x2))
    Es = -1/90 * h^5 * f^{4)}(\xi)
    Ec = -1/180 * h^4 * (b - a) * f^{4)}(\xi)
    N =         3
```

3b) The compound rule Q_C is obtained by subdividing the interval $[a,b]$ into M equally-wide subintervals, applying the simple rule to each one of them, and summing. The 1st subinterval will use nodes $x_1 = a$, $x_2 = a+h$, $x_3 = a+2h$. The 2nd subinterval will use x_3, x_4, x_5. The 3rd one x_5, x_6, x_7, etc. The last, M-th subinterval will use nodes x_{2M-1}, x_{2M}, $x_{2M+1} = b$. (We could also have called the nodes from $x_0 = a$ to $x_{2M} = b$, but we'll not use this notation here[3].) The distance between adjacent nodes is h. Each subinterval is of width $2h$, so $b-a = M\cdot 2h \Rightarrow h = (b-a)/(2M)$ (to also be used later). The i-th node is then $x_i = a+(i-1)h$ $(i = 1, 2, \ldots, 2M+1)$. The corresponding nodal ordinates can be noted $f_i = f(x_i)$. Then the compound quadrature rule reads:

$$\boxed{Q_C = \frac{h}{3}(f_1+4f_2+f_3)+\frac{h}{3}(f_3+4f_4+f_5)+\frac{h}{3}(f_5+4f_6+f_7)+\ldots+\frac{h}{3}(f_{2M-1}+4f_{2M}+f_{2M+1})}$$

or $$\boxed{Q_C = \frac{h}{3}(f_1+4f_2+2f_3+4f_4+2f_5+4f_6+2f_7+\ldots+2f_{2M-1}+4f_{2M}+f_{2M+1})}$$

or $$\boxed{Q_C = \frac{h}{3}(f_1+f_{2M+1})+\frac{4h}{3}(f_2+f_4+f_6+\ldots+f_{2M})+\frac{2h}{3}(f_3+f_5+f_7+\ldots+f_{2M-1})}$$

Q_C can be rearranged and expressed in other ways too, some with Greek Σ summation symbols, etc. All are good if they are good. The rule must involve the first and last nodal ordinates, four times the sum at all the even interior nodes, and two times the sum at all the odd interior nodes. **(1.25p)**

As for the truncation error term E_C, it is the algebraic sum of the errors made at all the subintervals. If in the 1st subinterval the simple rule makes an error 0.1 in defect ($E_1 = 0.1$), in the 2nd subinterval an error 0.2 in excess ($E_2 = -0.2$), in the 3rd subinterval 0.3 in defect ($E_3 = 0.3$), etc., then the error of the compound rule, after summing all the simple rules, will be the algebraic sum of their errors:

$$E_C = E_1 + E_2 + E_3 + \ldots = 0.1 - 0.2 + 0.3\ldots$$

In general, since E_i is given by the error of the simple rule we derived above:

$$E_C = \sum_{i=1}^{M} E_i = \sum_{i=1}^{M} \frac{-h^5 f^{4)}(\xi_i)}{90} = \frac{-h^5}{90}\sum_{i=1}^{M} f^{4)}(\xi_i)$$

where ξ_i is some point in the i-th subinterval. Multiplying and dividing by their number M:

$$E_C = \frac{-h^5}{90} M \frac{\sum_{i=1}^{M} f^{4)}(\xi_i)}{M} = \frac{-h^5}{90} M\,\overline{f^{4)}} = \frac{-h^5}{90}\,\frac{b-a}{2h}\,\overline{f^{4)}} = \frac{-h^4(b-a)}{180}\,\overline{f^{4)}}$$

[3] It makes more sense to start with x_0 when we will use an interpolation polynomial by all the nodes, because if the last one is x_n, the interpolation polynomial will be of degree $\le n$. That is not the case with compound rules. Starting with x_1, the index of the last node is the total number of nodes.

where $\overline{f^{4)}}$ is the arithmetic mean of the M values $f^{4)}(\xi_i)$ ($i = 1,\ldots,M$). As such, it must be somewhere between the maximum and the minimum value $f^{4)}(\xi_i)$. If $f^{4)}$ is continuous, i.e. if $f \in C^4([a,b])$, then, by Weierstrass's Intermediate Value Theorem, there must exist at least one $\xi \in [a,b]$ where $f^{4)}(\xi) = \overline{f^{4)}}$.

Substituting:
$$\boxed{E_C = \frac{-h^4(b-a)f^{4)}(\xi)}{180}}$$
<u>for some ξ in $[a,b]$</u>. This agrees with the output of **anm_nc** above. **(1.25p)**

3c) After checking that the number of nodes is odd as needed (indeed, there are 9):

```
format long g; h = 0.1;
f = [0 2.1220 3.0244 3.2568 3.1399 2.8579 2.5140 2.1639 1.8358];
Qc = h/3 * (f(1) + f(end)) + 4 * h/3 * sum(f(2:2:(end - 1))) + ...
                             2 * h/3 * sum(f(3:2:(end - 2)))
    Qc =      2.02649333333333
```

So the answer is: <u>$Q_C = 2.02649333333333$</u> **(1p)**

3d) We will use the expression of E_C above with $f(x) = \exp(x^2)$. Let's calculate its 4th derivative so we can substitute into E_C:

$$f(x) = e^{x^2}$$
$$f'(x) = 2xe^{x^2}$$
$$f''(x) = (4x^2 + 2)e^{x^2}$$
$$f^{3)}(x) = (8x^3 + 4x + 8x)e^{x^2} = (8x^3 + 12x)e^{x^2}$$
$$f^{4)}(x) = (16x^4 + 24x^2 + 24x^2 + 12)e^{x^2} = (16x^4 + 48x^2 + 12)e^{x^2}$$

This last expression is strictly positive for every x in the interval of integration $[0.5, 1]$. So substituting into E_C above with $f^{4)}(\xi) > 0$, $b - a > 0$ and $h^4 > 0$ necessarily gives $E_C < 0$. Since any negative number is less than 10^{-8}, any number of subintervals guarantees that $E_C < 10^{-8}$ as requested. For example, applying the Law of Least Effort:

<u>one subinterval ($M = 1$, or simple rule) is enough.</u> **(1.5p)**

Actually, any natural number M is a correct answer to this exercise as long as you justify the answer.

If we wanted to make sure that the *absolute value* of the error is less than 10^{-8}, we would proceed differently, namely, by finding an upper bound B of $|f^{4)}(x)|$ in $[0.5, 1]$ and substituting into E_C. We see that $f^{4)}$ is monotonic in that interval by calculating its derivative:

$$f^{5)}(x) = (32x^5 + 96x^3 + 24x + 64x^3 + 96x)e^{x^2} = (32x^5 + 160x^3 + 120x)e^{x^2}$$

which is strictly positive in the whole interval. Therefore $f^{4)}$ is strictly increasing[4] and, being positive, its maximum absolute value B, which is a tight upper bound of $|f^{4)}(x)|$ in $[0.5, 1]$, takes place at $x = 1$:

```
format long g
B = (16 + 48 + 12) * exp(1)
    B =      206.589418962887
```

Substituting into E_C (with abuse of notation in that the number of subintervals M must be an integer):

$$|E_C| = \left|\frac{-h^4(b-a)f^{4)}(\xi)}{180}\right| \le \frac{h^4(1-0.5)B}{180} = 10^{-8} \quad \Rightarrow \quad h = \sqrt[4]{\frac{180 \times 10^{-8}}{0.5B}}$$

```
h = (180e-8 / 0.5 / B)^(1/4)
    h =   0.0114894332573683
```

We now calculate the number of subintervals M that would result in this value of h:

[4] It is easy to see that $f(x) = \exp(x^2)$ and all of its derivatives are strictly positive in $[0.5, 1]$ (and therefore strictly increasing) by looking at its McLaurin series expansion (which is that of $\exp(x)$ with x^2 instead of x) and realizing that all its termwise derivatives can only have positive coefficients and positive powers of x.

$$b-a=M\cdot 2h \quad\Rightarrow\quad M=0.5/2/h=21.75912$$

Now the abuse of notation is apparent; but we also know that, in order to guarantee that $|E_C|<10^{-8}$, we need to take the next integer *larger* than 21.759, so

$M=22$ subintervals guarantee that $|E_C|<10^{-8}$

Finally, as usual, this is the number of subintervals we find a priori, but a posteriori a few fewer typically suffice. Octave's function **guadgk** gives a very precise value of our integral, which can be considered as exact for the purposes of this exercise; and our subject's function **anm_nc** implements all Newton-Cotes simple and compound rules, Simpson's included. Observe carefully:

```
f = @(x) exp(x.^2); I = quadgk(f, 0.5, 1)  % "exact" value of the integral
   I =    0.917664641723559
for M = 1:22, disp([M, I - anm_nc(f, 0.5, 1, 3, 1, M)]), end
                          1  -0.000879181025439157
                          2  -5.97334649216075e-005
                          3  -1.19951835295673e-005
                          4  -3.81761815382298e-006
                          5  -1.56795988726088e-006
                          6  -7.57275820162384e-007
                          7  -4.09124913391956e-007
                          8  -2.39961106762721e-007
                          9  -1.49866353660322e-007
                         10  -9.83554089284411e-008
                         11  -6.71922735229202e-008
                         12  -4.74498617064611e-008
                         13  -3.44540995733666e-008
                         14  -2.56180060498323e-008
                         15  -1.94413870557852e-008
                         16  -1.50189943814993e-008
                         17  -1.17855194492478e-008
                         18  -9.37722177685174e-009
                         19  -7.55381890371609e-009
                         20  -6.15283535232436e-009
                         21  -5.06209629769216e-009
                         22  -4.20269186118816e-009
```

So a posteriori we see that $M\geq 18$ subintervals also make $|E_C|<10^{-8}$

4) First the ODE of order 2 must be transformed into a system of 2 ODEs of order 1.

Calling $y=y_1$, $y'=y_2$: $\begin{cases} y_1'=y_2 \\ y_2'=2y_1+\cos(t)-e^t \end{cases}$ **(1.75p)**

the initial conditions being: $t_0=0\,;\quad \mathbf{y_0}=\begin{pmatrix}-1\\2\end{pmatrix}$ **(1p)**

We can now use Octave for the tedious work:

```
format long g
f = @(t, y) [y(2); 2 * y(1) + cos(t) - exp(t)];
t0 = 0; y0 = [-1; 2];
h = 0.1; t1 = t0 + h; t2 = t0 + 2 * h;
% First step:
k1 = f(t0,       y0) * h
    k1 =                 0.2
                        -0.2                                 % (-3p)
k2 = f(t0 + h/2, y0 + k1/2) * h
    k2 =                0.19
        -0.185252083598106                                   % (-3.25p)
k3 = f(t0 + h/2, y0 + k2/2) * h
    k3 =  0.190737395820095
        -0.186252083598106                                   % (-3.5p)
```

```
k4 = f(t0 + h,   y0 + k3) * h
    k4 =  0.181374791640189
         -0.172869196115743                                   % (-3.75p)
y1 = y0 + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    y1 =  -0.80952506945327
           1.81402041158197                                   % (-4p)
% Second step:
k1 = f(t1,       y1) * h
    k1 =  0.181402041158197
         -0.172921689170416                                   % (-4.25p)
k2 = f(t1 + h/2, y1 + k1/2) * h
    k2 =  0.172755956699676
         -0.161071126254058                                   % (-4.5p)
k3 = f(t1 + h/2, y1 + k2/2) * h
    k3 =  0.173348484845494
         -0.161935734699911                                   % (-4.75p)
k4 = f(t1 + h,   y1 + k3) * h
    k4 =  0.165208467688206
         -0.151368934953448                                   % (-5p)
y2 = y1 + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    y2 = -0.636388504130479
           1.65230302057667                                   % (-5.25p)
% Check with anm_ode:
[tout, yout] = anm_ode(f, [0 0.2], y0, 2, 'RK4')
    tout =              0                 0.1                   0.2
    yout =             -1    -0.80952506945327    -0.636388504130479
                        2     1.81402041158197      1.65230302057667
```

Looks good.

Since the problem is originally stated in terms of a single ODE (of order 2), its solution is $y(t) = y_1(t)$, so we are only interested in the first component of **y**. Hence the solution is:

$$\underline{y(0.2) \approx -0.6363885} \qquad \textbf{(-5.5p)}$$

5) The absolute stability condition is that all the eigenvalues $\mathrm{eigs}(Jh) = h\,\mathrm{eigs}(J)$ are in the method's absolute stability region. If the eigenvalues of J are real, that means that they must be in the absolute stability interval $(-2.78, 0)$ (which is the intersection of the absolute stability region with the real axis of the complex plane). In our case, with only one ODE, the Jacobian matrix J is $1{\times}1$, or a single element $\partial f/\partial y = f_y$ where $f(t,y) = t^2 - ty$, so: $\quad J = f_y = -t$

Absolute stability condition:

$$J h \in (-2.78, 0)$$
$$-2.78 < J h < 0$$
$$-2.78 < -t h < 0$$

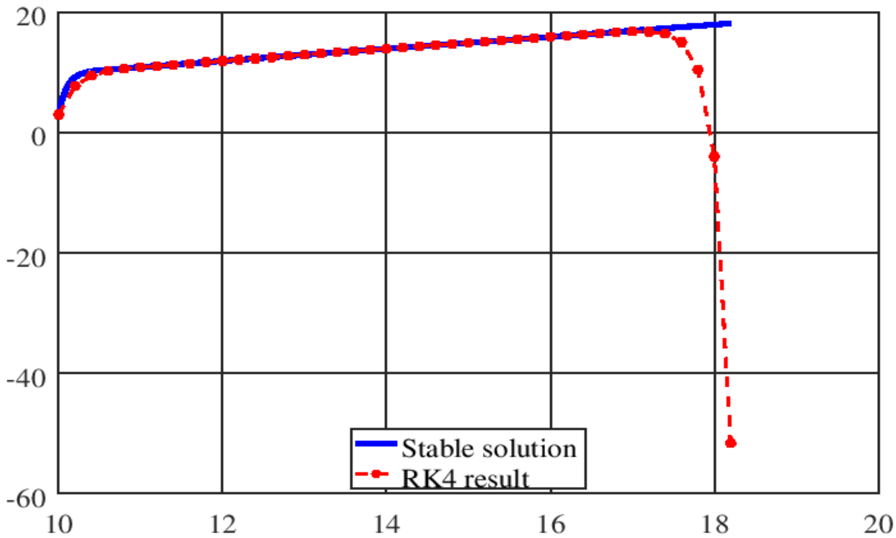
The last inequation always holds for $t \geq 10$ and $h = 0.2$ (as the exercise establishes).

The second-last inequation is satisfied iff $2.78 > t h$, or $t < 2.78/h = 2.78/0.2 = 13.9 = b$. Hence

$$\underline{\text{the method will be stable for } t \in [10, 13.9]} \qquad \textbf{(1.5p)}$$

It will be interesting to solve the problem with the RK4 method and $h = 0.2$ in an interval $[10, b]$ with $b > 13.9$ and see if we observe the effects of the method's numerical instability:

```
figure, hold on
b = 18.2;                              % greater than 13.9
f = @(t, y) t^2 - t * y;  interv = [10 b];  y0 = 3;  h = 0.2;
[tout, yout] = anm_ode(f, interv, y0, 0.01, 'RK4');  % stable
plot(tout, yout, 'b', 'LineWidth', 2)
[tout, yout] = anm_ode(f, interv, y0, h, 'RK4');
plot(tout, yout, '.--r', 'LineWidth', 1, 'MarkerSize', 12)
legend('Stable solution', 'RK4 result', 'Location', 'South')
```



The unstable RK4 solution visibly separates from the exact one a bit later than at $t=13.9$ —it's more like around $t=17$; but after it does, it goes astray really fast!

6) Going by the "theory":

$$E = e'(z) = \left(f[x_0,x_1,\ldots,x_n,z]\,\Pi(z)\right)' = f[x_0,x_1,\ldots,x_n,z]'\,\Pi(z) + f[x_0,x_1,\ldots,x_n,z]\,\Pi'(z) =$$

$$= 1!\,f[x_0,x_1,\ldots,x_n,z,z]\,\Pi(z) + f[x_0,x_1,\ldots,x_n,z]\,\Pi'(z) = \frac{f^{n+2)}(\xi)}{(n+2)!}\Pi(z) + \frac{f^{n+1)}(\eta)}{(n+1)!}\Pi'(z)$$

for some ξ, η in an interval containing the nodes.

Now compare with: $E_1 = \dfrac{f^{3)}(\xi)}{3!}\Pi(z)$; $E_2 = \dfrac{f^{4)}(\xi)}{4!}\Pi(z)$; $E_3 = \dfrac{f^{3)}(\xi)}{3!}\Pi'(z)$

$$E_4 = \frac{f^{3)}(\xi)}{3!}\Pi(z) + \frac{f^{4)}(\eta)}{4!}\Pi'(z); \quad E_5 = \frac{f^{3)}(\xi)}{3!}\Pi'(z) + \frac{f^{4)}(\eta)}{4!}\Pi(z)$$

The term E_1 can be a particular case of the general one if $n=1$ (two nodes x_0, x_1, which suffice to estimate $f'(z)$) and if $\Pi'(z)=0$. The only way for this to happen —which is geometrically obvious for a second-degree parabola $\Pi(x)$— is that z is the midpoint of both nodes, i.e., that both nodes lie symmetrically on both sides of z:

if $x_0 = z-kh,\ x_1 = z+kh$; e.g. if $x_0 = z-h,\ x_1 = z+h$, IT CAN BE. **(0.5p)**

The term E_2 could be another particular case if $n=2$ (three nodes x_0, x_1, x_2, which are more than enough to estimate $f'(z)$) if their positions relative to z make $\Pi'(z)=0$. There are ways to achieve this result. For instance, one can choose three distinct nodes just anywhere and then choose z at either the relative minimum or the relative maximum of $\Pi(x)$. Finally write the positions of the nodes as $x_i = z \pm k_i h$ where h is any distance of your choice (for instance, the one from z to the nearest node).

If nodes chosen as described above, IT CAN BE. **(0.5p)**

The term E_3 could be another particular case if $n=2$ (three nodes x_0, x_1, x_2, which are more than enough to estimate $f'(z)$) if $\Pi(z)=0$ (and the only way for this to happen is that z is one of the nodes). You can choose any 3 distinct nodes with z being one of them, and the error term will have the form E_3. E.g.: if $x_0 = z$; $x_1 = z + k_1 h$; $x_2 = z + k_2 h$ $\forall\, k_1, k_2, h \in \mathbb{R}$, IT CAN BE. **(0.5p)**

The term E_4 cannot be such error term, because n should be equal to 1 in the term with $\Pi(z)$ and equal to 3 in the term with $\Pi'(z)$, and no number can be 1 and 3 at the same time. IT CANNOT BE. **(0.5p)**

Finally the term E_5 could be another particular case if $n=2$ (three nodes x_0, x_1, x_2, which are more than enough to estimate $f'(z)$) with both $\Pi(z)\neq 0$ and $\Pi'(z)\neq 0$. This is what will happen most of the times if you choose three distinct nodes at random positions with respect to z. Observe that E_5 is like E_3 plus the added term we made the "effort" to eliminate in E_3 by choosing the nodes and z in very specific relative positions. For instance, we can be sure that $\Pi(z)\neq 0$ and $\Pi'(z)\neq 0$ with the following configuration —just imagine the plot of $\Pi(x)$ to convince yourself—:

if $x_0=z+h$; $x_1=z+2h$; $x_2=z+3h$, IT CAN BE. **(0.5p)**

7) There are several ways to do this. Since the error term is not asked, this time I will go with an ad-hoc manipulation of Taylor series that shows how to use $O(h^n)$ remainders conveniently. First I write the two expansions I'm obviously interested in:

$$f(z+2h)=f(z)+f'(z)2h+\frac{f''(z)}{2!}4h^2+O(h^3)$$
$$f(z+3h)=f(z)+f'(z)3h+\frac{f''(z)}{2!}9h^2+O(h^3)$$

I want to get rid of $f'(z)$ so I can isolate $f''(z)$ in terms of $f(z), f(z+2h), f(z+3h)$. I can easily do that by multiplying the first equation by 3, the second by 2, and subtracting:

$$3f(z+2h)=3f(z)+6f'(z)h+6f''(z)h^2+O(h^3)$$
$$2f(z+3h)=2f(z)+6f'(z)h+9f''(z)h^2+O(h^3)$$

Observe that $O(h^3)$ (pronounced "big O" of h^3) remains a remainder of order 3 of h even after multiplied by 2 or by 3. Now subtracting:

$$3f(z+2h)-2f(z+3h)=f(z)+0-3f''(z)h^2+O(h^3)$$

Observe that $O(h^3)-O(h^3)=O(h^3)$ even if it could be a remainder of higher order, because by definition they are also remainders of order 3. In fact It could be 0 (if the first $O(h^3)$ were exactly the same as the second one—which is not the case) and still be a $O(h^3)$.

Isolating $f''(z)$: $$\boxed{f''(z)=\frac{f(z)-3f(z+2h)+2f(z+3h)}{3h^2}+O(h)}$$ **(2.5p)**

A systematic application of Taylor series expansions yields the result given by **`anm_difftaylor`**:

```
anm_difftaylor(2, [0 2 3]);
    warning: division by zero [...]
    f"(z) = D + E   where
    D = (1/3 * f(z) - f(z + 2*h) + 2/3 * f(z + 3*h)) / h^2;
        Polynomial degree N = 2;  Order of convergence O = 1.  Error term:
    E = 1/factorial(3) * (8 * f3(\xi_1) - 18 * f3(\xi_2)) * h
        for some \xi_i (i = 0,...,n), each between z and node xi.
    |Etot| <= |E| + |Er|   where   |E| <= g1(h),  |Er| <= g2(h),   where:
        g1(h) = 5/3*M*h^1   where M >= |f3(x)| for all x bt the nodes and z.
        g2(h) = AF*ep = 2*h^-2*ep   where ep >= |fi - fibar| for i = 0:n.
        |Etot| <= g1(h) + g2(h) = g(h) =  5/3*M*h^1 + 2*h^-2*ep
        h opt => g min => g'(h) = 1 * 5/3 * M * 1 - 2 * 2 * h^-3 * ep = 0 =>
        h_opt  =  (12/5*ep/M)^(1/3)  =  1.338865900164 * ep^(1/3) * M^(-1/3)
        g_min = g(h_opt)  =  3.347164750410847 * M^(2/3) * ep^(1/3)
```

The differentiation of interpolation polynomials and errors yields the results by **`anm_diffinterpol`**:

```
anm_diffinterpol(2, [0 2 3])
      f"(z) = D + E,  where:
    D = A0 f(z) + A1 f(z + 2*h) + A2 f(z + 3*h)
    E = 1/6 f3(\xi_1) PI"(z) + 1/12 f4(\xi_2) PI'(z) + 1/60 f5(\xi_3) PI(z)
      Lagrange base functions Li(x), coefficients Ai in D, polynomial
    PI(x), and coefficients Kj in E (principal term first), all executable
    if symbolic package/toolbox installed:
syms x z h
L0 = (x - (z + 2*h)) * (x - (z + 3*h)) / (6 * h^2);
L1 = (x - z) * (x - (z + 3*h)) / (-2 * h^2);
L2 = (x - z) * (x - (z + 2*h)) / (3 * h^2);
A0 = subs(diff(L0, x, 2), x, z);
A1 = subs(diff(L1, x, 2), x, z);
```

```
A2 = subs(diff(L2, x, 2), x, z);
PI = (x - z) * (x - (z + 2*h)) * (x - (z + 3*h));
PI2x = simplify(diff(PI, x, 2)); PI2z = subs(PI2x, x, z), K3 = 1/6 * PI2z;
PI1x = simplify(diff(PI, x, 1)); PI1z = subs(PI1x, x, z), K4 = 1/12 * PI1z;
PI0x = simplify(diff(PI, x, 0)); PI0z = subs(PI0x, x, z), K5 = 1/60 * PI0z;
Ai = [A0 A1 A2]
Kj = [K3 K4 K5]
syms f3xi1 f4xi2 f5xi3
E = K3 * f3xi1 + K4 * f4xi2 + K5 * f5xi3
    PI2z =  -10*h
    PI1z =  6*h^2
    PI0z =      0
    Ai =    [ 1/3/h^2,  -1/h^2, 2/3/h^2]
    Kj =    [  -5/3*h, 1/2*h^2,       0]
    E =      -5/3*h*f3xi1+1/2*h^2*f4xi2
```

It can also be done by indeterminate coefficients.

All of which is consistent with what we found manipulating Taylor series ad-hoc.