ADVANCED NUMERICAL METHODS ORDINARY EXAM CALL, 5/27/2016. FULL ANSWER

Disclaimer: The answers here are far more complete than could possibly be expected from the student in a real exam situation. Their main purpose is a teaching one, not an assessment-related one.

Exercise 1

The conditions given are not those of an osculating interpolation polynomial, because the ones on $p(1)$, $p(2)$ and $p'(2)$ are missing. Therefore we have to study this problem concretely, without help from the existence and uniqueness theorem of interpolation polynomials.

We have 4 conditions, so we can have reasonable expectations (to be confirmed) that a polynomial with 4 degrees of freedom can satisfy them and do it uniquely. A complete polynomial of degree ≤ 3 has 4 degrees of freedom (since we are free to choose its 4 coefficients):

$$p(x) = a + bx + cx^2 + dx^3$$

Differentiating it: $\quad p'(x) = b + 2cx + 3dx^2\ ; \qquad p''(x) = 2c + 6dx$

So the 4 conditions are:

$$\begin{cases} a + b(-1) + c(-1)^2 + d(-1)^3 = f(-1) \\ b + 2c(-1) + 3d(-1)^2 = f'(-1) \\ b + 2c(1) + 3d(1)^2 = f'(1) \\ 2c + 6d(2) = f''(2) \end{cases}$$

which is a linear system in a, b, c, d and the independent terms are all data of the problem. Iff the system is compatible (has a solution) there will exist a polynomial of degree ≤ 3 satisfying the conditions given. And iff it is compatible determined (unique solution) the polynomial will be unique in $\mathbb{P}_3$.

To see how many solutions the system has, the method of choice (at least manually) is to apply Gauss's method by performing elementary row operations on its extended matrix :

$$\begin{pmatrix} 1 & -1 & 1 & -1 & f(-1) \\ 0 & 1 & -2 & 3 & f'(-1) \\ 0 & 1 & 2 & 3 & f'(1) \\ 0 & 0 & 2 & 12 & f''(2) \end{pmatrix} \left\langle R_3 - R_2 \right\rangle \begin{pmatrix} 1 & -1 & 1 & -1 & f(-1) \\ 0 & 1 & -2 & 3 & f'(-1) \\ 0 & 0 & 4 & 0 & f'(1) - f'(-1) \\ 0 & 0 & 2 & 12 & f''(2) \end{pmatrix} \left\langle R_4 - \frac{R_3}{2} \right\rangle$$

$$\begin{pmatrix} \underline{1} & -1 & 1 & -1 & f(-1) \\ 0 & \underline{1} & -2 & 3 & f'(-1) \\ 0 & 0 & \underline{4} & 0 & f'(1) - f'(-1) \\ 0 & 0 & 0 & \underline{12} & f''(2) - \dfrac{f'(1) - f'(-1)}{2} \end{pmatrix}$$

Here the backward substitution would not only give us the solution (which we are not asked about), but just seeing that it can always be done uniquely proves that the solution exists and is unique. (The same conclusion can be drawn, among other possibilities, by applying the Rouché-Capelli theorem, here known as Rouché-Frobenius: rg(coefficient matrix) = rg(extended matrix) = no. of unknowns = 4, where the ranges can be known just by counting principal elements, i.e. first non-zero elements of their row in the last matrix.)

There also exist infinitely many other polynomials of degrees > 3 satisfying those 4 conditions. Indeed, starting from the polynomial p obtained as above, we could add the conditions $p(1)$, $p(2)$, $p'(2)$ it satisfies (so p "becomes" an osculating interpolation polynomial), and then any number of additional interpolation points p does not satisfy. The existence and uniqueness of these new osculating polynomials is granted by theorem; but they must necessarily be of degree > 3 because $p(x)$ was unique in $\mathbb{P}_3$.

Summarizing: <u>There always exists a unique polynomial of degree ≤ 3 satisfying those 4 conditions, and infinitely many of degrees > 3</u>. (1.5p)

Exercise 2

A) The nodes look "evenly spaced" (only –1 and 1, so distance $h=2$), but they are not, because we have conditions on f' and f''. This is like having three nodes infinitely close to one another (triple node) for the three conditions on $x_0=-1$, or two (double node) for the two conditions on $x_1=1$. Hence like having 5 unevenly-spaced nodes, so we cannot use finite differences and we must use *divided differences with repetitions* to obtain the *osculating polynomial*. The table, with the help of Octave, is:

```
xi = [-1; 1]; fi = [-1 4 -6; 3 4 NaN];
[fii, zi, tdr] = anm_tableddr(xi, fi);
prettyprintt(tdr(:,2:end), {'x_i', 'f_i', 'f_1i', 'f_2i', 'f_3i', 'f_4i'})
     ans =
     x_i   f_i   f_1i   f_2i   f_3i   f_4i
     ---   ---   ----   ----   ----   ----
      -1    -1    NaN    NaN    NaN    NaN
      -1    -1      4    NaN    NaN    NaN
      -1    -1      4     -3    NaN    NaN
       1     3      2     -1      1    NaN
       1     3      4      1      1      0
```

All the values here are placed and calculated as usual, including the value –3 at column $f_{2,i}$, which is the datum $f''=-6$ divided by 2! (because it is a second derivative).

The last divided difference is 0, so any one of the five conditions is redundant[1]. Using the first four, for simplicity: $\underline{p_3(x)=-1+4(x+1)-3(x+1)^2+(x+1)^3}$

"Optimal evaluation" for us usually means applying the Hörner-like algorithm, i.e. taking as many common factors as possible, resulting in less operations and more stable error propagation than other methods like the direct evaluation of the expression above. Rearranging:

$$p_3(x)=-1+(x+1)\,\{4+(x+1)\,[-3+(x+1)]\}$$

Evaluating at $x=0.5$ with the operations performed in the order indicated:

$$p_3(0.5)=-1+\underbrace{(x+1)\underbrace{\{4+\underbrace{(x+1)\underbrace{[-3+\underbrace{(x+1)}_{1.5}]}_{-1.5}}_{-2.25}\}}_{1.75}}_{2.625}=\underline{1.625}$$

Hence $\qquad p_3(0.5)=1.625$

It does not get much "more optimal" than this. The redundancy "most optimal"[2] in the exercise statement might refer[3] to evaluating $x+1=1.5$ only once instead of three times above. Or it might refer to using the data closest to $x=0.5$, as is usually (when you do not have such round and exact numbers as here) best in order to reduce errors. In that case we could use the last four data, instead of the first four. From the last four rows of the same table:

$$p_3(x)=-1+4(x+1)-(x+1)^2+(x+1)^2(x-1)$$

Hörner-like rearrangement: $\qquad p_3(x)=-1+(x+1)\,\{4+(x+1)\,[-1+(x-1)]\}$

[1] Because, counting the multiplicity of the nodes, it is like having five nodes determining a Lagrange polynomial of degree 3. Any four of the five nodes will determine the same polynomial of degree 3.

[2] Per dictionary, optimal = best or most favorable ⇒ most optimal = "most best" or "most most favorable".

[3] It actually referred to not expanding $p(x)$ in powers of x. (After consultation with the exercise's author.)

Evaluation at $x = 0.5$: $$p_3(0.5) = -1 + \underbrace{(x+1)\{\underbrace{4+\underbrace{(x+1)\underbrace{[-1+\underbrace{(x-1)}_{-0.5}]}_{-1.5}}_{-2.25}}_{1.75}\}}_{2.625} = \underline{1.625}$$

with the same result (and dubious gain in optimality). (3p)

B) The best estimation of the error made by $f(0.5) \approx 1.625$ would arguably be to calculate the osculating polynomial p_5 obtained with all the data available (including the new one, $f''(1) = 8$), evaluate p_5 at 0.5, and subtract 1.625. Because that would be {our arguably best estimation of the exact value} minus {the approximate value whose error we want to estimate}.

The new table would add a final row (and a mostly empty final column) and change no other number in the previous one, so we could just add it if working by hand; but here for clarity (and because Octave does it for us) we will redo the whole table again:

```
xi = [-1; 1]; fi = [-1 4 -6; 3 4 8];
[fii, zi, t2] = anm_tableddr(xi, fi);
prettyprintt(t2(:,2:end), {'x_i','f_i','f_1i','f_2i','f_3i','f_4i','f_5i'})
```

```
      ans =
      x_i    f_i    f_1i    f_2i    f_3i    f_4i     f_5i
      ---    ---    ----    ----    ----    ----    -----
       -1     -1     NaN     NaN     NaN     NaN      NaN
       -1     -1       4     NaN     NaN     NaN      NaN
       -1     -1       4      -3     NaN     NaN      NaN
        1      3       2      -1       1     NaN      NaN
        1      3       4       1       1       0      NaN
        1      3       4       4     1.5    0.25    0.125
```

Here the final 4 in the column $f_{2,i}$ is the new datum $f'' = 8$ divided by 2! (because it is a second derivative), and the rest of the values are also placed and calculated as usual.

Since the second-last coefficient is still 0 (we are just adding one final row), the only additional term that must be added to $p_3(x)$ to obtain $p_5(x)$ is, using the last divided difference in the new table:

$$p_5(x) - p_3(x) = p_5(x) - p_4(x) = h_5(x) = 0.125\,(x+1)^3\,(x-1)^2$$

Hence: $$e(0.5) \approx p_5(0.5) - p_3(0.5) = 0.125 \times 1.5^3 \times (-0.5)^2 = 0.10546875$$

A similar justification for essentially the same thing goes as follows. We know that in this case $p_3 \equiv p_4$, where p_4 is the osculating polynomial satisfying the initial five data. So an exact expression of the error made by $p_3(x)$ is the well-known one for the error made by $p_4(x)$, namely:

$$e(x) = f[x_0, x_0, x_0, x_1, x_1, x]\,\Pi(x)$$

This equation is exact, but we do not know the value of the divided difference in it[4]. Yet we know it is of order 5 (6 nodes), so related with $f^{5)}$, which hopefully does not change too wildly. Hence we can estimate it as the divided difference of order 5 that we do have, namely, 0.125 at the end of the new table:

$$e(x) \approx 0.125\,\Pi(x) \quad \Rightarrow \quad e(0.5) \approx 0.125\,\Pi(0.5) = 0.125\,(0.5 - x_0)^3\,(0.5 - x_1)^2$$

This, of course, is the same thing: $\underline{e(0.5) \approx 0.105469}$ (1.5p)

[4] To calculate it we would need the value of $f(x)$; but if we knew $f(x)$ the error $e(x)$ would be known directly as $f(x) - p_3(x)$.

Exercise 3

The intermediate unknowns to calculate cubic splines are the moments, i.e. the nodal second derivatives. For a natural cubic spline, by definition, the first and last moments are zero, so not unknown. With five nodes that leaves three unknowns (the three interior moments). This discards the last matrix because it is 5×5, not 3×3.

As for the other two matrices, even if we have not memorized their exact expressions, we should remember they are very well conditioned, namely, strictly diagonally dominant: the absolute value of each element in the principal diagonal must be strictly greater than the sum of those of the other elements in its row. This discards the first one.

So the second matrix.

(Additionally you might remember that each interior element in the principal diagonal is twice the sum of the elements immediately to its right and left, or above and below it, which does happen in the second matrix. Or that the super- and subdiagonal are both constant, which is also enough to discriminate.)

(1p)

Exercise 4

$$I=\int_1^2\left[(x-1)^2+\frac{x^3}{\sqrt{(x-1)(2-x)}}\right]dx=\underbrace{\int_1^2(x-1)^2\,dx}_{I_1}+\underbrace{\int_1^2\frac{x^3}{\sqrt{(x-1)(2-x)}}dx}_{I_2}$$

The first integral is very easy to calculate exactly with Simpson's rule, for example, which is of polynomial degree 3 (the subintegral function being a polynomial of degree 2 ≤ 3):

$$I_1=\int_1^2(x-1)^2\,dx=\frac{h}{3}\left(f_0+4f_1+f_2\right)=\frac{1/2}{3}\left(0^2+4\left(\frac{1}{2}\right)^2+1^2\right)=\frac{1/2}{3}2=\frac{1}{3}$$

We can check the result with the Fundamental Theorem of Calculus, or Barrow's Law:

$$I_1=\int_1^2(x-1)^2\,dx=\int_0^1x^2dx=\left.\frac{x^3}{3}\right]_0^1=\frac{1}{3}=0.\hat{3}\quad\text{ok}$$

The other integral looks very much like a linear change of variable will transform it into a Gauss-Chebyshev one. (If it doesn't, we will still have transformed the interval of integration into [−1, 1] to try Gauss-Legendre rules.)

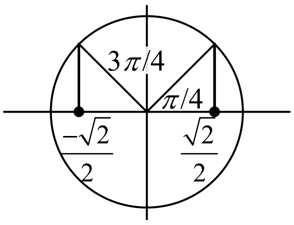
Applying the linear change of variable $x=1.5+0.5\,t$:

$$I_2=\int_1^2\frac{x^3}{\sqrt{(x-1)(2-x)}}dx\underset{x=\frac{3}{2}+\frac{1}{2}t}{=}\int_{-1}^1\frac{\left(\frac{3}{2}+\frac{1}{2}t\right)^3}{\sqrt{\left(\frac{1}{2}+\frac{1}{2}t\right)\left(\frac{1}{2}-\frac{1}{2}t\right)}}\frac{dt}{2}=\int_{-1}^1\frac{(3+t)^3/8}{\sqrt{(1+t)(1-t)}}dt$$

$$I_2=\int_{-1}^1\frac{(3+t)^3/8}{\sqrt{1-t^2}}dt=\int_{-1}^1\frac{g(t)}{\sqrt{1-t^2}}dt\quad\text{where } g(t)=(3+t)^3/8$$

The form of the integral is indeed that of a Gauss-Chebyshev one (interval [−1, 1] with the right weight function). The numerator is a polynomial of degree 3, so any Gauss-Chebyshev rule with 2 or more nodes will solve it exactly (the polynomial degree of the rule is $2n_n-1$ where n_n is the number of nodes). We choose 2 nodes to get the simplest rule.

The nodes are the roots of the Chebyshev polynomial of degree 2:



$$T_2(t) = \cos(2(\operatorname{acos} t)) = 0 \quad \Rightarrow \quad \operatorname{acos} t = \{\pi/4, 3\pi/4\} \quad \Rightarrow$$

$$t = \{\cos(\pi/4), \cos(3\pi/4)\} \quad \rightarrow \quad t_0 = -\sqrt{2}/2, \quad t_1 = \sqrt{2}/2$$

The Gauss-Chebyshev weights are all equal and their sum is π. (If you don't remember this bit, apply the formula with $g(x) \equiv 1$.) Hence: $w_0 = \pi/2, \quad w_1 = \pi/2$

Applying the rule: $I_2 = w_0 g(t_0) + w_1 g(t_1) = \dfrac{\pi}{2}\dfrac{\left(3-\sqrt{2}/2\right)^3}{8} + \dfrac{\pi}{2}\dfrac{\left(3+\sqrt{2}/2\right)^3}{8} =$

$$= \frac{\pi}{16}\left[\left(3^3 - 3\cdot 3^2\sqrt{2}/2 + 3\cdot 3\cdot 1/2 - \sqrt{2}/4\right) + \left(3^3 + 3\cdot 3^2\sqrt{2}/2 + 3\cdot 3\cdot 1/2 + \sqrt{2}/4\right)\right] =$$

$$= \frac{\pi}{16} 2\left(3^3 + 3\cdot 3\cdot 1/2\right) = \frac{\pi}{8}\frac{54+9}{2} = \frac{63\pi}{16} \approx 12.3700210735098$$

Adding I_1 and I_2: $I = \boxed{\dfrac{1}{3} + \dfrac{63\pi}{16}} \approx 0.\hat{3} + 12.3700210735 = 12.7033544068$ (4p)

Exercise 5

Theory starting on page 63 of the Classroom Notes in Spanish. (2p)

Exercise 6

A) First we must transform the ODE of order 2 into a system of order 1. Calling $y_1 = y, \quad y_2 = y'$:

$$y'' + t\,y' + y = 0 \quad \rightarrow \quad y_2' + ty_2 + y_1 = 0 \quad \rightarrow \quad \begin{cases} y_1' = y_2 \\ y_2' = -y_1 - ty_2 \end{cases}$$

The differential problem with matrix notation:

$$\mathbf{y}' = \begin{pmatrix} f_1(t,\mathbf{y}) \\ f_2(t,\mathbf{y}) \end{pmatrix} = \begin{pmatrix} y_2 \\ -y_1 - ty_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ -1 & -t \end{pmatrix}}_{J} \mathbf{y} = \mathbf{f}(t,\mathbf{y}); \quad t_0 = 0; \quad \mathbf{y_0} = \begin{pmatrix} y_1(t_0) \\ y_2(t_0) \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

One step must take us from $t_0 = 0$ to $t_1 = 0.2$, so the step size is $h = 0.2$. The calculations will be shown with Octave. You try to reproduce these numbers with your calculator:

```
format long g                          % see more than 6 siginificant digits
h = 0.2;                               % step size
t0 = 0; y0 = [1; 2];                   % initial conditions
f = @(t,y) [0 1; -1 -t] * y;           % right-hand side of system of ODEs
f = @(t,y) [y(2); -y(1)-t*y(2)];       % alternative way to define f(t,y)
k1 = f(t0, y0) * h                     % 1st column constant of the RK4 step
     k1 =                    0.4
                            -0.2
k2 = f(t0 + h/2, y0 + k1/2) * h        % 2nd column constant of the RK4 step
     k2 =                   0.38
                          -0.278
k3 = f(t0 + h/2, y0 + k2/2) * h        % 3rd column constant of the RK4 step
     k3 =                 0.3722
                        -0.27522
k4 = f(t0 + h, y0 + k3) * h            % 4th column constant of the RK4 step
```

```
     k4 =                0.344956
                        -0.3434312
y1 = y0 + (k1 + 2*k2 + 2*k3 + k4) / 6 % end of the step
     y1 =      1.37489266666667
               1.72502146666667
t1 = h;                              % in case I need this value later
[tout, yout] = anm_ode(f, [t0 h], y0, 1, 'RK4')       % confirm result
     tout =                 0                      0.2
     yout =                 1         1.37489266666667
                            2         1.72502146666667
```

Rounding to 6 significant digits (doing it after every intermediate result would not have changed much), the result is: $\underline{y(0.2) \approx 1.37489}$

$\underline{y'(0.2) \approx 1.72502}$ (4p)

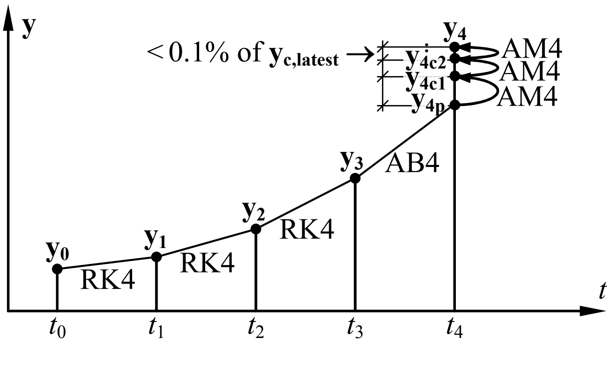
B) The Adams predictor-corrector method predicts with the explicit Adams-Bashforth method of order 4 (AB4) and corrects with the implicit Adams-Moulton method of order 4 (AM4). And if you don't remember this, there you have the advance formulas you have to use: one of an explicit method (the first one provided, expressing y_{n+1} in terms of known things, i.e. with indices $\leq n$, which is AB4) and one of an implicit method (the second one, expressing y_{n+1} as a function of itself, which is AM4).

AB4 cannot be applied unless one already has four calculated points—look at its advance formula—, and that is why we are given another three points (vector values) besides the initial conditions. The one at $t_1 = 0.2$ coincides with what we calculated in section A), which is reassuring. Just in case let us see if the other two also correspond to applying the RK4 method:

```
[tout, yout] = anm_ode(f, [t0 3*h], y0, 3, 'RK4')            % confirm data
     tout =    0              0.2                0.4                0.6
     yout =    1  1.37489266666667    1.6817602636138   1.90110065378217
               2  1.72502146666667  1.32729589455448  0.859339607730699
anm_signif(yout(:,2:end)', 6)
     ans =              1.37489                     1.72502
                        1.68176                      1.3273
                         1.9011                     0.85934
```

They do coincide with the numbers provided.

Let us now draw a little scheme to always know what we are about to do:



Let's show the calculations with Octave. Make sure you can reproduce them with your calculator:

```
t0 = 0; h = 0.2; t1 = h; t2 = 2*h; t3 = 3*h; t4 = 4*h;
% To use the data provided without any extra significant digits:
y0=[1;2]; y1=[1.37489;1.72502]; y2=[1.68176;1.3273]; y3=[1.9011;0.85934];
% Prediction with AB4:
f = @(t,y) [0 1; -1 -t] * y;    % right-hand side of the system of ODEs
f0 = f(t0, y0), f1 = f(t1, y1), f2 = f(t2, y2), f3 = f(t3, y3)
```

```
     f0 =          2
                  -1
     f1 =    1.72502
           -1.719894
     f2 =     1.3273
            -2.21268
     f3 =    0.85934
           -2.416704
y4p = y3 + h/24 * (55*f3 - 59*f2 + 37*f1 - 9*f0)     % y4 "predicted"
     y4p =     2.02425616666667
                     0.38428435
% First correction with AM4:
f4p = f(t4, y4p)
     f4p =           0.38428435
              -2.33168364666667
y4c1 = y3 + h/24 * (9*f4p + 19*f3 - 5*f2 + f1)       %  y4 corrected once
     y4c1 =    2.02505449291667
                   0.3796814765
% Termination criterion (using the customary infinity norm):
abs( (y4c1 - y4p) ./ y4c1 ) * 100
     ans =   0.0394224576569374
               1.21229867267432
% The second value exceeds 0.1, so we must correct again:
f4c1 = f(t4, y4c1)
     f4c1 =        0.3796814765
              -2.32879967411667
y4c2 = y3 + h/24 * (9*f4c1 + 19*f3 - 5*f2 + f1)       %  y4 corrected twice
     y4c2 =    2.02470927740417
               0.37989777444125
% Termination criterion:
abs( (y4c2 - y4c1) ./ y4c2 ) * 100
     ans =   0.0170501274603856
               0.0569358274256995
% Both < 0.1, so we take this as definitive:
y4 = y4c2;
% Confirm results (except that RK4 were rounded in exercise statement):
[tout, yout, ci] = anm_ode(f, [t0 t4], y0, 4, 'PC4', 2); % a posteriori
[tout' yout']                                             % more legible
     ans =         0                    1                  2
                 0.2     1.37489266666667   1.72502146666667
                 0.4      1.6817602636138   1.32729589455448
                 0.6     1.90110065378217  0.859339607730699
                 0.8     2.02471000503372  0.379897225785726
ci{:, end}                                                % convergence info
     ans =                    2
     ans =     2.02425911139167
              0.384282175493761
     ans =     2.02505510478401
              0.379680845489921
     ans =     2.02471000503372
              0.379897225785726
```

The termination criterion, imprecisely expressed in the exercise statement as “precision 0.1%”, was applied elementwise rather than using the infinity norm of the vectors **y**. The reason for this is that the scales and magnitudes of the different elements of **y** are different. If, for instance, the first element y_1 of **y** is a position in m, and the second element $y_2=y_1'$ is a velocity in ms^{-1}, it makes little sense to check if the correction (change) in the velocity exceeds 0.1% of the last position, or if the correction in the position exceeds 0.1% of the latest velocity. It makes much more sense to check that the correction in the position does not exceed 0.1% of the latest position *and* the correction in the velocity does not exceed 0.1% of the latest velocity.

Rounding to 6 significant digits—doing it after every intermediate result would not have changed much—, the result is:

$$y(0.8) \approx 2.02471$$
$$y'(0.8) \approx 0.379898$$

(3p)

Exercise 7

A) Theory starting on page 153 of the Classroom Notes in Spanish. (1p)

B) Theory starting on page 156 of the Classroom Notes in Spanish. (1p)

Exercise 8

A) Interpolatory idea: $A_i = L_i''(z)$

In our case: $x_0 = z-2h; \quad x_1 = z-h; \quad x_2 = z$

Differentiating the base functions (a.k.a. Lagrange base polynomials):

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-z+h)(x-z)}{(-h)(-2h)} = \frac{x^2+\ldots}{2h^2}; \quad A_0 = L_0''(z) = \frac{2}{2h^2} = \frac{1}{h^2}$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-z+2h)(x-z)}{(h)(-h)} = \frac{x^2+\ldots}{-h^2}; \quad A_1 = L_1''(z) = \frac{-2}{h^2}$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-z+2h)(x-z+h)}{(2h)(h)} = \frac{x^2+\ldots}{2h^2}; \quad A_2 = L_2''(z) = \frac{2}{2h^2} = \frac{1}{h^2}$$

Hence:
$$f''(z) = \frac{f(z-2h) - 2f(z-h) + f(z)}{h^2} + E$$

This reminds very much of the *centered* divided difference of three nodes to estimate $f''(z)$. The weights are the same. But here z is the rightmost node instead of the central one. However, since $p_2(x)$ has a constant second derivative, it does not matter where z is. (2p)

B) Theory starting at section 3.2.2.2 of the Classroom Notes in English. (2p)

C) According to B):

$$= \frac{f^{n+1)}(\xi_1)}{(n+1)!}\Pi''(z) + 2\frac{f^{n+2)}(\xi_2)}{(n+2)!}\Pi'(z) + 2\frac{f^{n+3)}(\xi_3)}{(n+3)!}\Pi(z)$$

for some ξ_1, ξ_2, ξ_3, between the nodes and z, with $n=2$ (3 nodes). Since z is one of the nodes, $\Pi(z)=0$, so only the first and second terms remain. We have to calculate $\Pi'(z)$ and $\Pi''(z)$ for the case at hand:

$$\Pi(x) = (x-x_0)(x-x_1)(x-x_2) = (x-z+2h)(x-z+h)(x-z)$$

$$\Pi'(x) = 1(x-z+h)(x-z) + (x-z+2h)1(x-z) + (x-z+2h)(x-z+h)1 = 3x^2 + (-6z+6h)x + \ldots$$

$$\Pi'(z) = 0 + 0 + 2h^2; \quad \Pi''(x) = 6x + (-6z+6h); \quad \Pi''(z) = 6z + (-6z+6h) = 6h$$

Substituting: $E = \frac{f^{3)}(\xi_1)}{3!}6h + 2\frac{f^{4)}(\xi_2)}{4!}2h^2 = \boxed{f^{3)}(\xi_1)h + \frac{f^{4)}(\xi_2)}{6}h^2 \quad \text{f.s. } \xi_1, \xi_2 \in (z-2h, z)}$ (1p)