ADVANCED NUMERICAL METHODS

DEGREE IN INDUSTRIAL TECHNOLOGY ENGINEERING

JULY 5, 2014 – RESOLUTION

1.-

a) Since the nodes are not evenly-spaced, we must use *divided* differences:

i	xi	fi	fi1	fi2	fi3	fi4	fi5
0	0	0					
1	1	1	1				
2	1.5	0	-2	-2			
3	11	0	0	0.2	0.2		
4	11.5	1	2	0.2	0	-0.0174	
5	12.5	0	-1	-2	-0.2	-0.0174	0

All numbers have been calculated with the usual formula, $f_{i,k} = (f_{i,k-1} - f_{i-1,k-1}) / (x_i - x_{i-k})$, with 3-significant-digit arithmetic throughout. The operations are trivial, but here they are anyway:

$f_{1,1} = (1-0) / (1-0) = 1$
$f_{2,1} = (0-1) / (1.5-1) = -2$
$f_{3,1} = (0-0) / (11-1.5) = 0$
$f_{4,1} = (1-0) / (11.5-11) = 2$
$f_{5,1} = (0-1) / (12.5-11.5) = -1$
$f_{2,2} = (-2-1) / (1.5-0) = -2$
$f_{2,3} = (0--2) / (11-1) = 0.2$
$f_{2,4} = (2-0) / (11.5-1.5) = 0.2$
$f_{2,5} = (-1-2) / (12.5-11) = -2$
$f_{3,3} = (0.2--2) / (11-0) = 0.2$
$f_{3,4} = (0.2-0.2) / (11.5-1) = 0$
$f_{3,5} = (-2-0.2) / (12.5-1.5) = -0.2$
$f_{4,4} = (0-0.2) / (11.5-0) = -0.0173913 \approx -0.0174$
$f_{4,5} = (-0.2-0) / (12.5-1) = -0.0173913 \approx -0.0174$
$f_{5,5} = (-0.0174--0.0174) / (12.5-0) = 0$

b) We can readily write the Newton representation of the interpolation polynomial $p_5(x)$ from the numbers in the table of divided differences:

$$p_5(x) = \underline{x - 2x(x-1) + 0.2x(x-1)(x-1.5) - 0.0174x(x-1)(x-1.5)(x-11)}$$

The interpolation polynomial $p_5(x)$ <u>is of degree 4</u> (because the last divided difference turns out to be exactly 0). It is called $p_5(x)$ because it is the only polynomial of degree ≤ 5 that satisfies the interpolation data (which, by the way, means that no polynomial of degree 5 does).

Of course there exist infinitely many other polynomials that also satisfy the interpolation data. You can simply add any more points arbitrarily and calculate the corresponding unique interpolation polynomial. However, as said above, all of them must be of degree 6 or higher.

c) The Hörner algorithm (and its variant to evaluate polynomials written in its Newton representation) is optimal in the sense of its computational cost (you have to calculate fewer products than with any other method, like direct evaluation for instance) and its numerical stability (the roundoff errors propagate with less amplification than with other methods).

d) Using the Hörner variant for Newton polynomials, and with 3-significant-digit arithmetic throughout:

$$p_5(x) = \left\{\left[\left(-0.0174(x-11)+0.2\right)(x-1.5)-2\right](x-1)+1\right\}x$$

$$p_5(6.8) = \underbrace{\left\{\underbrace{\left[\underbrace{\underbrace{\left(\underbrace{\underbrace{-0.0174(-4.2)}_{0.0731}+0.2}_{0.273}\right)(5.3)}_{1.45}-2}_{-0.550}\right](5.8)}_{-3.19}+1\right\}}_{-2.19}6.8 = \underline{-14.9}$$

e) Since $f_0 = f_2 = f_3 = f_5 = 0$, only the second and the second-last base functions, $L_1(x)$ and $L_4(x)$, appear in $p_5(x)$:

$$p_5(x) = f_1L_1(x) + f_4L_4(x) = 1\frac{(x-0)(x-1.5)(x-11)(x-11.5)(x-12.5)}{(1-0)(1-1.5)(1-11)(1-11.5)(1-12.5)} +$$
$$+1\frac{(x-0)(x-1)(x-1.5)(x-11)(x-12.5)}{(11.5-0)(11.5-1)(11.5-1.5)(11.5-11)(11.5-12.5)}$$

$$p_5(x) = \frac{x(x-1.5)(x-11)(x-11.5)(x-12.5)}{(-0.5)\times(-10)\times(-10.5)\times(-11.5)} + \frac{x(x-1)(x-1.5)(x-11)(x-12.5)}{11.5\times 10.5\times 10\times 0.5\times(-1)}$$

Let us now evaluate it at $x = 6.8$ with 3-significant-digit arithmetic throughout. We will perform the operations in the order in which they appear:

$$p_5(6.8) = \frac{\overbrace{6.8\times 5.3\times(-4.2)\times(-4.7)\times(-5.7)}^{-4050}}{\underbrace{(-0.5)\times(-10)}_{5.00}\times\underbrace{(-10.5)\times(-11.5)}_{-52.5}}^{604} + \frac{\overbrace{6.8\times 5.8\times 5.3\times(-4.2)(-5.7)}^{5005}}{\underbrace{11.5\times 10.5}_{121}\times\underbrace{10\times 0.5\times(-1)}_{-10}}^{-605} =$$

$$= \frac{-4050}{604} + \frac{5005}{-605} = -6.71 - 8.27 = \underline{-15.0}$$

This last value is much closer to the exact one, −15.0052..., than the −14.9 that we obtained applying the Hörner-like algorithm to the Newton representation of $p_5(x)$. Of course with exact arithmetic both ways would result in the same, exact value, because $p_5(x)$ is unique (you can just write it in different forms), so this difference is necessarily the result of the propagation of roundoff errors when using 3-significant-digit arithmetic.

It is a little surprising that the final roundoff error turned out larger with the Hörner variant (which has good numerical stability) than when using the expression of $p_5(x)$ in terms of Lagrange base functions (which is known to be numerically unstable, as we have seen in the subject's theory). However, in specific cases, this can happen because of positive and negative roundoff error cancellation. It will probably turn out that with higher-precision arithmetic in both cases the roundoff error eventually becomes smaller using the Hörner variant –you can try it if you want–.

In fact, the final roundoff error of the Hörner variant is on the order of the precision of the arithmetic used (namely, the last one of the three significant digits of the result is wrong by one unit), which is actually not bad at all (many numerical methods show an annoying tendency to really amplify errors as they propagate). The anomalous result here is the practically 4-significant-digit precision obtained with the Lagrange base functions. This is *necessarily* a coincidence if one is using 3-significant-digit arithmetic!

f) With the two new nodes I would add two new rows to the table of divided differences, calculate the unique interpolation polynomial $p_7(x)$, evaluate it at $x = 6.8$, and subtract $p_5(6.8)$ from it (so we get simply $e_5(6.8) \approx h_6(6.8) + h_7(6.8)$, i.e., the two new terms evaluated at $x = 6.8$).

This is a straightforward generalization of the theory we studied on the estimation of the error of an interpolation polynomial at z, in which just one additional node was used for the estimation. If we now have information from two new nodes and our best estimation of $f(z)$ is, reasonably, $p_7(z)$, our best estimation of the error made with $p_5(z)$ must necessarily be

$e_5(z) = f(z) - p_5(z) \approx p_7(z) - p_5(z) = h_6(z) + h_7(z)$ (according to the definition of $h_i(x)$ at the beginning of the theory on Newton polynomials).

I would use the Newton representation (table of divided differences), rather than the Lagrange base functions, so I can use the work done before and do not have to start all over from square one; but regardless of the representation used, it will still be valid that $e_5(z) \approx p_7(z) - p_5(z)$.

The assumption we are making is, of course, that $p_7(x)$ follows $f(x)$ more closely than $p_5(x)$. That is not always true.

Another reasonable way to answer to this question, even if not so precise, is to use the expression in the theory just as we studied it, i.e. $e(z) = f[x_0,x_1,\ldots,x_n,z]\,\Pi(z)$, and estimate the divided difference in it, for instance, by the average of the two new divided differences of order $n+1$ that appear in the table after adding the two new nodes.

g) If $f(6.7) = 0$ and $f(6.9) = 0$, $f(6.8)$ is going to be very close to 0, so the error made with $p_5(6.8)$ is going to be very close to $-p_5(6.8)$, namely, error about 15 (i.e., in defect).

2.-

a) Cubic splines are optimal in that they are least-oscillating in the following precise sense.

Let there be $n+1$ nodes $x_0 < x_1 < \cdots < x_n$ and the corresponding nodal ordinates $y_0, y_1, \cdots, y_n$.

Consider the set G of all functions $g \in C^2([x_0,x_n])$ satisfying the interpolation data, i.e., such that $g(x_i) = y_i$ $(i = 0,...,n)$, and also such that $g''(x_0) = g''(x_n) = 0$.

By definition, the natural cubic spline $s_3(x)$ by those nodes belongs to G (since it is sufficiently smooth, i.e. of class C^2; it satisfies the interpolation data; and, being a natural spline, $s''(x_0) = s''(x_n) = 0$): $s_3 \in G$

Now, if one measures oscillations in a specific, reasonable way, it can be proved that there is no other function in G that oscillates less than s_3!

The way to measure oscillations is the integral between x_0 and x_n of the square of the second derivative of the function. This is reasonable, because the second derivative of a function is directly related with its curvature (albeit not via direct proportionality), and a straight line, which has clearly no oscillations, has identically zero second derivative. The derivative is squared so that there is no cancellation of its positive and negative values in $[x_0,x_n]$.

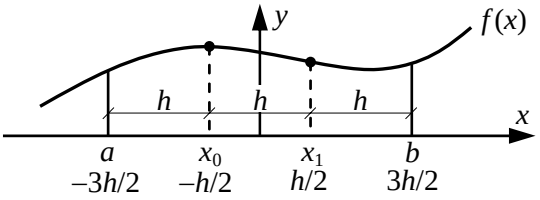
The result reads: $$\int_{x_0}^{x_n} \left[g''(x)\right]^2 dx \geq \int_{x_0}^{x_n} \left[s_3''(x)\right]^2 dx \quad \forall g \in G$$

b) The figure shows $s_3(x)$ oscillating less than $p_5(x)$. Near the middle of the interval, for instance, we can see that $p_5(x)$ is close to −16, while $s_3(x)$ is close to −6.

The question arises as to why, if $s_3(x)$ is the least-oscillating function in G, its absolute value near the center of the interval cannot be even less than about 6. The answer is that then $s_3''(x)$ must increase somewhere else in [0,12.5], and the integral figure of merit measuring oscillations increases.

3.-

a) The open Newton-Cotes rule of two nodes has distance between nodes $h=(b-a)/3$ and nodes $x_0=a+h$, $x_1=a+2h$, where a and b are the limits of integration. The weights or coefficients do not depend on a and b alone, but only on their difference (or, alternatively, on h). This allows one to set the origin of abscissas on the midpoint of $[a,b]$ so as to calculate the coefficients more easily but without loss of generality. The following figure shows a, b and the nodes x_0, x_1 in terms of h:



We will determine the two coefficients by imposing the exact integration of the monomials 1, x. Like this the rule will also integrate exactly any linear combination of them, i.e. the rule will be exact in $\mathbb{P}_1$. Calling the coefficients A_0, A_1:

$$\left.\begin{array}{ll} f(1)=1 & \rightarrow \quad \int_{-3h/2}^{3h/2} 1\,dx = 3h = A_0 1 + A_1 1 \\ f(1)=x & \rightarrow \quad \int_{-3h/2}^{3h/2} x\,dx = 0 = A_0(-3h/2) + A_1(-3h/2) \end{array}\right\} \Rightarrow \quad \boxed{A_0 = A_1 = \frac{3h}{2}}$$

b) The polynomial degree is at least 1 (and actually 1), so the error term is of the form $E = Kf''(\xi)$ for some $\xi \in (a,b)$:

$$\int_{-3h/2}^{3h/2} f(x)dx = \frac{3h}{2} f(x_0) + \frac{3h}{2} f(x_1) + K f''(\xi)$$

If we apply this to the function $f(x)=x^2$ (the first monomial that is not integrated exactly), which is convenient because its second derivative is constant, which will allow us to determine K:

$$f(x)=x^2 \quad \rightarrow \quad \int_{-3h/2}^{3h/2} x^2 dx = 2\frac{(3h/2)^3}{3} = \frac{3h}{2}\left(\frac{-h}{2}\right)^2 + \frac{3h}{2}\left(\frac{h}{2}\right)^2 + K\cdot 2$$

$$\frac{9h^3}{4} = \frac{3h^3}{4} + 2K \quad \Rightarrow \quad K = \frac{6h^3}{8} = \frac{3h^3}{4} \quad \Rightarrow \quad \boxed{E_S = \frac{3h^3 f''(\xi)}{4}}$$

for some ξ between *a* and *b*, where E_S stands for the error of the simple rule.

c) The compound rule is consists in subdividing [*a*,*b*] into *N* subintervals of equal width and applying the simple rule to each one of them. Therefore, the error of the compound rule E_C is the algebraic sum of the errors made with the *N* simple rules:

$$E_C = \sum_{i=1}^{N} E_S(i) = \sum_{i=1}^{N} \frac{3h^3 f''(\xi_i)}{4} = \frac{3h^3}{4} \sum_{i=1}^{N} f''(\xi_i)$$

where ξ_i is some point belonging to the *i*-th subinterval $[a_i,b_i]$. Multiplying and dividing by *N*:

$$E_C = \frac{3h^3}{4} N \frac{\sum_{i=1}^{N} f''(\xi_i)}{N} = \frac{3h^3}{4} N \overline{f''}$$

where $\overline{f''}$ is the arithmetic mean of the *N* values $f''(\xi_i)$. As such, it must be some intermediate value between the greatest and the smaller of them. Assuming that $f'' \in C([a,b])$, by virtue of Weierstrass's Intermediate Value Theorem, there must be at least one $\xi \in C[a,b]$ such that $f''(\xi) = \overline{f''}$, and substituting:

$$E_C = \frac{3h^3}{4} N f''(\xi) = \frac{3h^3}{4} \frac{(b-a)}{3h} f''(\xi) \quad \Rightarrow \quad \boxed{E_C = \frac{h^2 (b-a) f''(\xi)}{4}} \qquad (1)$$

for some ξ between *a* and *b*.

d) To guarantee that the absolute error does not exceed $5 \cdot 10^{-4}$ we express the limiting case writing, with some abuse of notation (which there's nothing wrong with if you know what you are doing):

$$|E_C| = \left| \frac{h^2 (b-a) f''(\xi)}{4} \right| = 5 \cdot 10^{-4}$$

For the worst-case scenario, we will substitute $f''(\xi)$ by its maximum absolute value in the interval [0,0.3]:

$$f(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}$$

$$f'(x) = \frac{2}{\sqrt{\pi}} (-2x) e^{-x^2}$$

$$f''(x) = \frac{2}{\sqrt{\pi}} \left[(-2x)(-2x) e^{-x^2} - 2e^{-x^2} \right] = \frac{2}{\sqrt{\pi}} (4x^2 - 2) e^{-x^2}$$

We have been told that this is an increasing function in [0,0.3]. Functions that are monotonic in an interval (i.e., functions that are either increasing or decreasing in that interval) take their extreme values in the interval at its endpoints, and are therefore bounded by them:

$$f''(0) = \frac{2}{\sqrt{\pi}}(4\times 0^2 - 2)e^{-0^2} = \frac{-4}{\sqrt{\pi}} = -2.257$$

$$f''(0.3) = \frac{2}{\sqrt{\pi}}(4\times 0.3^2 - 2)e^{-0.3^2} = -1.691 \qquad (2)$$

The larger absolute value is the first one. Hence we write:

$$\frac{h^2\times(0.3-0)\times 2.257}{4} = 5\cdot 10^{-4} \quad \rightarrow \quad h = \sqrt{\frac{20\times 10^{-4}}{0.3\times 2.257}} = \sqrt{0.00295} = 0.0543$$

$$\rightarrow \quad N = \frac{b-a}{3h} = \frac{0.3-0}{3\times 0.0543} = 1.84 \quad \Rightarrow \quad \boxed{N = 2 \text{ subintervals suffice}}$$

Now it is obvious that we have been abusing notation for a while, because N must be an integer, so it could not be equal to 1.84; however, since we know in what sense precision improves (the more subintervals, the better), we take $N = 2 > 1.84$ to "err on the side of caution", and in this way the absolute error is guaranteed to be less than $5\cdot 10^{-4}$.

e) The first subinterval is [0,0.15] with nodes 0.05 and 0.10; and the second one is [0.15,0.30] with nodes 0.20 and 0.25. The distance between the nodes in each subinterval is $h = 0.05$ weights are all $3h/2 = 0.075$:

$$Q_2 = 0.075\times\big(f(0.05) + f(0.10) + f(0.20) + f(0.25)\big) =$$

$$= 0.075\times(1.12556 + 1.11715 + 1.08413 + 1.06001) = 0.075\times 4.38685 = \boxed{Q_2 = 0.32901}$$

f) From (1) and (2): $\dfrac{h^2(b-a)(-2.257)}{4} \le E_C \le \dfrac{h^2(b-a)(-1.691)}{4}$

$$\frac{0.05^2\times 0.3\times(-2.257)}{4} \le E_C \le \frac{0.05^2\times 0.3\times(-1.691)}{4}$$

$$\boxed{E_C \in [-0.00042, -0.00032]}$$

The error actually made is (with 5 decimals):

$$E_C = \text{erf}(0.3) - Q_2 = 0.32863 - 0.32901 = -0.00038$$

which clearly belongs to the interval as expected.

g) The order of convergence of the compound rule is $p = 2$ because that is the exponent of h in (1). If we pass from $N = 2$ to $N = 4$ subintervals, h is divided by 2, and therefore the error

will be divided by $2^p = 4$ approximately:

$$E_4 \approx \frac{E_2}{4} = \frac{-0.00038}{4} = -0.000095 \quad \Rightarrow$$

$$Q_4 \approx \text{erf}(0.3) - E_4 = 0.32863 - -0.000095 = \boxed{0.328725}$$

N.B. It is not asked, but applying the compound rule with $N = 4$ subintervals, the result is 0.328724, which is very much as expected.

h) For the absolute error to be less than 10^{-16} we proceed like in section d). With some abuse of notation:

$$\frac{h^2 \times (0.3-0) \times 2.257}{4} = 10^{-16} \quad \rightarrow \quad h = \sqrt{\frac{4 \times 10^{-16}}{0.3 \times 2.257}} \quad \rightarrow$$

$$N = \frac{b-a}{3h} = \frac{0.3-0}{3} \sqrt{\frac{0.3 \times 2.257}{4 \times 10^{-16}}} \approx 4\,114\,304 \quad \rightarrow \quad \boxed{N > 4 \times 10^6 \text{ subintervals}}$$

We would need more than 4 million subintervals to guarantee double precision in erf(0.3) (and that, assuming that there were no roundoff error propagation). The obvious practical conclusion is that this rule is too bad for this purpose. Newton-Cotes rules are actually all too bad for many serious applications. In general, whenever you want to get "value for money" (precision for computational cost), Gauss rules are the way to go. A compound Gauss-Legendre rule with a good number of nodes and many subintervals (but not on the order of the millions!) might get the job done. In practice, there exist very optimized series expansions that allow to calculate erf and many other functions with very high precision and much, much less computational cost.

4.-

a) This is the THEORY about the "instability of numerical differentiation". See your classroom notes (either the ones you took from the blackboard or the Spanish version, p. 112).

b) This is obviously going to be the first centered difference formula studied in the subject for $f''(z)$ (with order of convergence 2) but with $2h$ instead of h. I will obtain it using ad-hoc Taylor series expansions. (To see how to obtain it by differentiating the error term of the interpolation polynomial, see the classroom notes and replace h with $2h$ throughout. And you can also follow the more general theory to obtain difference formulas to approximate $f^{k)}(z)$ and particularize it for $k = 2$, $h_0 = -2h$, $h_1 = 0$, $h_2 = 2h$; see for instance the exam of the ordinary 2014 call.)

I will write the two Taylor expansions of interest as infinite series, and later decide where it is convenient for me to truncate them (i.e., what Taylor remainder to write and what smoothness

conditions to demand from $f(x)$):

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

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

Now it is clear to me that I want to add them term by term, and that I want to use the Taylor remainders of order 4:

$$f(z-2h) + f(z+2h) = 2f(z) + 2\frac{f''(z)}{2!}(2h)^2 + \frac{f^{4)}(\xi_1)}{4!}(2h)^4 + \frac{f^{4)}(\xi_2)}{4!}(2h)^4$$

for some $\xi_1 \in (z-2h,z)$ and $\xi_2 \in (z,z+2h)$, and assuming that $f \in C^4([z-2h,z+2h])$ (or at least that $f^{4)}$ exists and is bounded in that interval).

Isolating $f''(z)$: $$f''(z) = \frac{f(z-2h) + f(z+2h) - 2f(z)}{4h^2} - \frac{f^{4)}(\xi_1) + f^{4)}(\xi_2)}{6}h^2$$

As h tends to 0^+, ξ_1 tends to z from the left and ξ_2 tends to z from the right. If $f^{4)}$ is continuous in a neighborhood of z, both terms in the last numerator will tend to $f^{4)}(z)$, which is finite (possibly 0), so the last term is a remainder $O(h^2)$, i.e., it tends to 0 "at least as fast as (proportionally to) h^2" –maybe faster, if $f^{4)}(z)$ turns out to be 0–. If $f^{4)}$ is continuous, the last numerator will be equal to $2f^{4)}(\xi)$ for some ξ near z.

To give mathematical rigor to these essentially correct reasoning, consider the function $g(x) = 2f^{4)}(x)$. Its value on ξ_1 is $2f^{4)}(\xi_1)$, and its value on ξ_2 is $2f^{4)}(\xi_2)$. The numerator $f^{4)}(\xi_1) + f^{4)}(\xi_2)$ is an intermediate value one between both of them –just check the cases $f^{4)}(\xi_1) < f^{4)}(\xi_2)$, $f^{4)}(\xi_1) > f^{4)}(\xi_2)$, and $f^{4)}(\xi_1) = f^{4)}(\xi_2)$–. Therefore, if $g \in C([\xi_1,\xi_2])$ (which we are assuming, since $f \in C^4([z-2h,z+2h])$, and $\xi_1 \in (z-2h,z)$, $\xi_2 \in (z,z+2h)$), by virtue of Weierstrass's Intermediate Value Theorem, there must exist at least one $\xi \in (\xi_1,\xi_2)$ such that $g(\xi) = f^{4)}(\xi_1) + f^{4)}(\xi_2)$. Therefore, $f^{4)}(\xi_1) + f^{4)}(\xi_2) = 2f^{4)}(\xi)$ for some $\xi \in (z-2h,z+2h)$.

Substituting: $$f''(z) = \boxed{\frac{f(z-2h) - 2f(z) + f(z+2h)}{4h^2}} - \frac{f^{4)}(\xi)}{3}h^2$$

c) The error term is: $$\boxed{E = -\frac{f^{4)}(\xi)}{3}h^2 \text{ for some } \xi \in (z-2h, z+2h)} \qquad (3)$$

Since $E = O(h^2)$, the formula is of order of convergence 2, which means that the truncation error tends to 0 proportionally to h^2 (or faster).

And the formula is of order 3 because it is exact in $\mathbb{P}_3$ (since every polynomial of degree ≤ 3 has identically-zero fourth derivative, hence $E = 0$) but not in $\mathbb{P}_4$ (because x^4 has identically non-zero fourth derivative, hence $E \neq 0$).

A note about the terminology: The *order of convergence* is sometimes also referred to as the *order of precision* (in our case, 2) because it describes how fast the error decreases (i.e., the precision increases) as h decreases. The word *order* (without further specification) of the formula (in our case 3, because the formula is exact in $\mathbb{P}_3$ but not in $\mathbb{P}_4$) can lead to confusion with the *order of convergence or of precision*, so it is sometimes also referred to as the *polynomial degree* of the formula, which is a much more descriptive term. In general, the word "order" often refers to the number of times a function is differentiated (like in the order of a differential equation, not its degree), while the word "degree" often refers to the value of an exponent or power (like in the degree of a polynomial, not its order).

d) The total error E_T is equal to the truncation error E (the one we would make with exact data and exact arithmetic) plus the roundoff error E_r (by definition, the one attributable exclusively to inexact data and inexact arithmetic):

$$E_T = E + E_r$$

Therefore: $$|E_T| \le |E| + |E_r|$$

We will try to minimize the maximum absolute error that can be made, so we will substitute both right-hand terms by respective upper bounds of them, and then minimize the sum.

An upper bound of $|E|$ can be obtained from (3):

$$|E| \le \frac{M}{3} h^2$$

where M is any upper bound of $|f^{4)}(\xi)|$. We will later calculate M.

As for $|E_r|$, the roundoff error E_r was modeled and approximated in section a) (the theory about "instability of numerical differentiation"). We did not get an exact value, because we only considered errors in the data ordinates (not abscissas) and we didn't analyze error propagations either; but that would be out of the scope of this subject, and the theory studied is enough to get a good idea about the value of the roundoff error.

So from section a), we know that the amplification factor AF is equal to the sum of the absolutes of the coefficients; in our case:

$$AF = \frac{1}{4h^2} + \frac{2}{4h^2} + \frac{1}{4h^2} = \frac{1}{h^2} \quad \Rightarrow \quad |E_r| \le AF \cdot \varepsilon = \frac{\varepsilon}{h^2}$$

where ε is the maximum absolute error in the nodal ordinates, i.e. in the values $f(z-2h)$, $f(z)$, $f(z+2h)$.

Substituting: $$|E_T| \le \frac{M}{3} h^2 + \frac{\varepsilon}{h^2} = g(h)$$

To minimize $g(h)$: $$g'(h) = \frac{2M}{3}h - 2\frac{\varepsilon}{h^3} = 0 \quad \Rightarrow \quad h = \sqrt[4]{\frac{3\varepsilon}{M}}$$

(We see that it's a minimum rather than a maximum because $g''(h) = 2M/3 + 6\varepsilon/h^4 > 0$; and we expected that to be the case because truncation errors increase as h increases and roundoff errors increase as h decreases, so there must be an optimum, i.e. a minimum, somewhere in between.)

Regarding the value of M, since we are not told for what z we must apply the formula, we will take M to be the maximum of $|f^{4)}(x)|$ for $x \in (-\infty,\infty)$ –if it exists–. Differentiating four times, $f^{4)} = f$, and its maximum absolute value takes place at $x = \pi/4 + k\pi$ (since $f^{5)}(x) = 0 \Rightarrow \cos(x) - \sin(x) = 0 \quad \Rightarrow \quad x = \pi/4 + k\pi \quad \Rightarrow \quad |f^{4)}(x)| \le |\sin(\pi/4) + \cos(\pi/4)| = 2\cdot 2^{1/2}/2 = 2^{1/2}$). Substituting we obtain the optimal value of h, h_{opt}, in terms of ε:

$$\boxed{h_{opt} = \sqrt[4]{3\varepsilon/\sqrt{2}}}$$

As for the value of ε, it depends on the precision of the data. If the data are not experimental, that depends mostly on the precision of the arithmetic used (rather than the precision of the measuring device used). With double-precision arithmetic, and a function like $f(x) = \sin(x) + \cos(x)$, whose values are in the range between $-2^{1/2}$ and $2^{1/2}$, the maximum value of ε will be on the order of 10^{-16}. Substituting[1]:

$$h_{opt} \approx \sqrt[4]{3\cdot 10^{-16}/\sqrt{2}} = 0.00012$$

which gives us a very reasonable value of h to use. That's interesting, because with about 16 significant digits in the arithmetic, one would probably expect to have a few more orders of magnitude to safely decrease h… isn't that what you would've expected?

With simple-precision arithmetic, i.e. with ε on the order of 10^{-8}, as used by many programming languages to speed up calculations over double-precision ones, we get:

$$h_{opt} \approx \sqrt[4]{3\cdot 10^{-8}/\sqrt{2}} = 0.012$$

which is surprisingly large too, but actually quite close to the optimal value, as can be checked by actually carrying out the calculations.

5.-

a) Picard's Theorem provides sufficient (although not necessary) conditions for the

[1] A bit more precisely, MATLAB, which uses double-precision arithmetic by default, gives 2.22e-16 as the result of `eps(sqrt(2))`, i.e., as the distance to the closest number to $2^{1/2}$ that it can represent.

existence and uniqueness of the solution to initial value problems. We will check if the problem at hand meets the conditions of the theorem, which states:

Let D be the region of the ty plane $D = \{(t,y) / t_0 \le t \le t_f, -\infty < y < \infty\}$. If $f \in C(D)$ and it verifies the Lipschitz condition for the variable y, namely, if $\exists\, L > 0$ such that

$$|f(t,y_2) - f(t,y_1)| \le L\,|y_2 - y_1| \quad \forall\,(t,y_2),(t,y_1) \in D$$

then the problem $y' = f(t,y)$ with $y(t_0) = y_0$ has a unique solution $y(t)$ for $t \in [t_0,t_f]$.
It is both intuitive and easy to prove (see Classroom Notes) that a sufficient condition for f to be Lipschitzian in this sense is that $|\partial f/\partial y| \le L$, i.e., that f_y is bounded in D (and f, $f_y \in C(D)$).

In our case, both $f(t,y) = (1+t)y$ and $f_y(t,y) = 1+t \in C(\mathbb{R}^2)$, and the absolute value of the latter is upper-bounded in D by the Lipschitz constant $L = 1.5$ (since in our problem D is the region between $t_0 = 0$ and $t_f = 0.5$). Therefore, the solution $y(t)$ to the problem exists and it is unique.

b) The four points provided are the ones needed to start the multistep method of order 4 we are requested to apply. This is typically done using the Runge-Kutta method of the same order of convergence as the multistep method we will use, in our case RK4. Even if the points are provided, just out of curiosity, let's see if they are indeed the ones obtained with the RK4 method. Using MATLAB, the numbers are:

```
>> f = @(t,y) (1+t).*y;
>> t0 = 0;
>> y0 = 1;
>> [tt,yy] = anm_ode(f,[t0 t0+3*h],y0,3,'RK4')
   tt =   0   0.1          0.2          0.3
   yy =   1   1.1107105   1.2460764   1.4119892
>> ff = f(tt,yy)
   ff =   1   1.2217815   1.4952917   1.8355859
```

Therefore RK4 seems indeed to be the method used, although f_2 has apparently been wrongly rounded to 1.49530 instead of to 1.49529. In any case, we will use the numbers provided.

We now have to take the next two steps (in the sense of calculating two more points, y_4 and y_5). We will first define the functions of the advance formulas of the AB4 and AM4 methods (noted like this because they are both of order of convergence 4). The following is an almost-literal transcription of the advance formulas provided in the exercise statement:

```
>> ab4=@(yn,h,fn,fnm1,fnm2,fnm3) yn+h/24*(55*fn-59*fnm1+37*fnm2-9*fnm3);
>> am4=@(yn,h,fnM1,fn,fnm1,fnm2) yn+h/24*(9*fnM1+19*fn-5*fnm1+fnm2);
```

For the first prediction, using AB4, these are the numbers we will need in order to apply the formula for $n = 3$ (from the exercise statement):

```
>> f0 = 1;
   f1 = 1.22178;
```

```
f2 = 1.49530;   # prolly meant to be 1.49529
f3 = 1.83559;
y3 = 1.41199;
```

Applying the formula:

```
>> y40 = ab4(y3,h,f3,f2,f1,f0)
   y40 = 1.6159092
```

The result has been called **y40** to stand for "$y_4^{(0)}$", or 0-th estimate (the only role played by the "prediction" provided by AB4 is to be the starter of the fixed-point AM4 iterations).

Now the method requires us to carry out fixed-point corrections with the AM4 method until the "precision" in the calculation of the solution (in the sense of the distance between the last two values y_n calculated) does not exceed 10^{-4}. For the first correction:

```
>> t4 = 0.4; f40 = f(t4,y40)
   f40 = 2.2622729
>> y41 = am4(y3,h,f40,f3,f2,f1)
   y41 = 1.6160814
```

where **y41** stands for "$y_4^{(1)}$". Let's see if the distance between **y40** and **y41** exceeds 10^{-4}:

```
>> abs(y41-y40)
   ans = 0.00017
```

It does, so we need to make a second correction and check for the termination criterion:

```
>> f41 = f(t4,y41)
   f41 = 2.2625140
>> y42 = am4(y3,h,f41,f3,f2,f1)
   y42 = 1.6160905
>> abs(y42-y41)
   ans = 9.042e-006
```

Since the distance does not exceed 10^{-4}, we can take this value as definitive in $t_4 = 0.4$:

```
>> y4 = y42
   y4 = 1.6160905
```

Now comes a prediction of y_5 with AB4 and fixed-point corrections with AM4 until the termination criterion is met. The process is very similar:

```
>> f4 = f(t4,y4)
   f4 = 2.2625267
>> y50 = ab4(y4,h,f4,f3,f2,f1)
   y50 = 1.8680456
>> t5 = 0.5; f50 = f(t5,y50)
   f50 = 2.8020685
>> y51 = am4(y4,h,f50,f4,f3,f2)
   y51 = 1.8682737
>> abs(y51-y50)
```

```
   ans = 0.0002281
>> f51 = f(t5,y51)
   f51 = 2.8024106
>> y52 = am4(y4,h,f51,f4,f3,f2)
   y52 = 1.8682865
>> abs(y52-y51)
   ans = 1.28e-005
>> y5 = y52
   y5 = 1.8682865
```

All the operations were internally performed with double-precision arithmetic. You can round off all the results above to 5 decimals so as to comply with the exercise requirement.

Using the function **anm_ode** yields the same result (maybe with slightly better precision due to the fact that it does apply RK4 to start the method, instead of using the truncated values provided in the exercise statement):

```
>> [tt,yy,ci] = anm_ode(f,[t0 0.5],y0,5,'PC4',1e-4)
   tt =  0  0.1         0.2         0.3         0.4         0.5
   yy =  1  1.11071049  1.24607639  1.41198919  1.61608949  1.86828541
   ci =
   []     []     []     []    [               2]    [               2]
   []     []     []     []    [1.61590975632967]    [1.86804397513036]
   []     []     []     []    [1.61608052092319]    [1.86827255239958]
   []     []     []     []    [1.61608948606435]    [1.86828540987097]
```