ADVANCED NUMERICAL METHODS

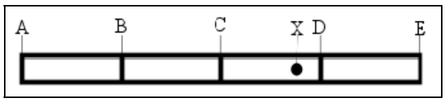
Degree in Industrial Technology Engineering

MAY 29, 2017

TIME: 3 hours **29 points total**

N.B. Exercises needing a calculator should be solved with rounding to 6 significant digits.

1.- A body moving along a 10 m-long underground pipe emits a signal whose intensity could be measured at 5 equally-spaced points according to the scheme below; the intensities obtained are the ones shown in the table.



A	B	C	D	E
1.61534	2.18174	2.25203	1.98124	1.56423

One wants to estimate the intensity of the signal at point *X*, located at a distance of 2 m from the pipe's midpoint, via interpolation.

a) From the estimation of truncation errors, decide whether the quadratic or the cubic interpolation is more advisable. (3 points)

Some thoughts before calculating anything[1].

The truncation error is related with $f^{3)}(\xi)$ for $p_2(x)$ (quadratic interpolation) or with $f^{4)}(\xi)$ for $p_3(x)$ (cubic); and these are related with third- and fourth-order differences respectively, which can be seen in a table of differences, so I will build one of those (which are also good to estimate truncation errors, if we need that in order to answer the question).

The nodes are equally spaced, so we can use divided or finite differences. The latter *initially* entail less manual calculations[2]. They also keep rounding errors neatly confined to 5 *decimal* digits in the table[3]. Moreover, by experience, exercises with equally-spaced nodes are usually expected to be solved using finite differences, so I'll do that (but divided differences would also do just fine).

As a general rule, remember to be alert to detect the possible appearance of spurious changes of sign at some column, with increasing absolute values towards its right. If that happens, we should not use those columns, and that would limit the order of the interpolation used. But here we have so few data points and so many significant digits[4] that we may not reach the columns when that can happen. If that is the case, we should reason as follows.

[1] You don't need to write any of the following—just build the table of differences as soon as you realize it's a good way to go, and later reason out according to what you see. This resolution is for teaching purposes more than for exam-revision ones.

[2] Although computers evaluate polynomials based on divided differences noticeably faster, because the quotients in them are computed once and for all, while the ones in $q(t)$ are not.

[3] This may be nice when solving exercises by hand, but mostly irrelevant when using computers.

[4] In reality you don't even sense something in an underground pipe remotely with precision on the order of 6 significant digits, as our data come with, all too often. This exercise seems quite academic.

If the differences of order 2 are constant, all 5 data points can be generated by a polynomial p_2 of degree ≤ 2, and any 3, 4 or 5 points out of those 5 will give the same polynomial $p_2 \equiv p_3 \equiv p_4$. Then the quadratic and the cubic interpolations would be the same thing. If the differences of order 2 are only approximately constant, the quadratic interpolation using 3 points (C, D, E, closest to X) and the cubic one using 4 points (B, C, D, E) would give very similar results. We could even choose the quadratic one for simplicity. And if the differences of order 2 are not even close to constant, in principle a cubic interpolation would be more advisable in order to capture more of the behavior of f. Furthermore, if the differences of order 3 are not close to constant, the quartic interpolation using all 5 points could be considered—but the exercise statement disregards it. Also note that the point X where we have to estimate f is not particularly close to the interval ends, so the Runge effect is not to be feared so much, especially when considering such low-degree polynomials.

Let's finally calculate the table of finite differences and see what we see. I'll do it with Octave, but this is also easy to do by hand and/or with a calculator:

```
xi = linspace(0, 10, 5)                   % equally-spaced nodes
   xi =     0         2.5           5          7.5          10
yi = [1.61534  2.18174  2.25203  1.98124  1.56423];
[Diy0, table] = anm_tablefd(xi, yi);    % table of finite differences
tchar = anm_prettyprintable(table, 6)
   i       xi    Δ^0f(xi)         Δ^1          Δ^2         Δ^3         Δ^3
   -       --    -------          ---          ---         ---         ---
   0        0    1.61534
   1      2.5    2.18174      0.56640
   2        5    2.25203      0.07029     -0.49611
   3      7.5    1.98124     -0.27079     -0.34108     0.15503
   4       10    1.56423     -0.41701     -0.14622     0.19486     0.03983
```

We see no *spurious* changes of sign, etc., and the three differences of order 2 are not even close to constant, so in principle p_3 should capture the behavior of f better than p_2 and a cubic interpolation is probably more advisable than a quadratic one (but maybe less than the quartic one).

This should be enough to justify the answer, without numerically estimating truncation errors. However, since we are explicitly asked about them, let's estimate them at X for p_2 and for p_3 as usual. This time let's first do the calculations and then reason out according to what we see.

An estimation $e_{2,3}$ of the error of p_2 at X can be obtained by adding the information at node B to the one at the last three nodes, C, D, E, already used by p_2. The following equation is exact:

$$e(X) = f[C,D,E,X]\,\Pi(X) = f[C,D,E,X]\,(X-C)\,(X-D)\,(X-E)$$

but would need the value of $f(X)$ to compute the divided difference in it (and then $e(X) = f(X) - p_2(X)$ is also exact and simpler). However we can estimate $f[C,D,E,X]$ as $f[C,D,E,B] = f[B,C,D,E]$ (because it is a difference of the same order 3 as the one being estimated, and hence related with the same derivative $f^{3)}$; and because permutations of the nodes do not affect divided differences). Hence:

$$e(X) \approx e_{2,3} = f[B,C,D,E]\,(X-C)\,(X-D)\,(X-E)$$

Using the relationship between divided and finite differences:

$$e_{2,3} = \frac{\Delta^3 f(B)}{3!h^3}(X-C)(X-D)(X-E) = \frac{0.19486}{3!h^3}(X-C)(X-D)(X-E)$$

where 0.19486 is found in the table. With the change of variable $x = C + ht = ht \quad (h = 2.5)$:

$$e_{2,3} = \frac{0.19486}{3!h^3} ht\, h(t-1)\, h(t-2) = \frac{0.19486}{3!} t(t-1)(t-2)$$

At $x = X = 7, \quad t = (X - C)/h = (7-5)/2.5 = 0.8$ so:

$$e_{2,3} = \frac{0.19486}{3!} 0.8(-0.2)(-1.2) = 0.00623552$$

Similarly our estimation for the error of p_3 at X would be, by incorporating the info on node A:

$$e_{3,4} = \frac{0.03983}{4!} 1.8 \cdot 0.8 \cdot (-0.2) \cdot (-1.2) = 0.000573552$$

where 0.03983 is also found in the table.

We see that our estimation of the error of p_3 is less than 10% of that of p_2:

$$e_{3,4} / e_{2,3} = 0.000573552 / 0.00623552 = 0.0919814225597864$$

and conclude that p_3 is more advisable than p_2.

Right?

Not so fast: there's much of a circular reference here. I.e., the conclusion we have drawn is partly based on the assumption that it is true. Note that $e_{2,3}$, as calculated above, is precisely the term $h_3(X)$ that has to be added to $p_2(X)$ to obtain $p_3(X)$, so $e_{2,3} = p_3(X) - p_2(X)$. Whenever we estimate the error of p_2 like that, we are implicitly assuming that $p_3(X)$ is a better estimation of $f(X)$ than $p_2(X)$, which is what we wanted to decide on! By that token, the cubic interpolation would *always* be better than the quadratic one, and we would not need to calculate any error estimation.

Similarly $e_{3,4}$, as calculated above, is precisely the term $h_4(X)$ that has to be added to $p_3(X)$ to obtain $p_4(X)$, so $e_{3,4} = p_4(X) - p_3(X)$, and whenever we estimate the error of p_3 like that we are implicitly assuming that $p_4(X)$ is a better estimation of $f(X)$ than $p_3(X)$.

Therefore, "from the estimation of truncation errors" (which is what the exercise asks), p_3 would *always* be better than p_2 (and p_4 better than p_3). Instead of calculating truncation error estimations, we should probably just look at the table of differences searching for spurious changes of sign and then reason out in the way in which we started.

One might argue that the following makes a little bit more sense: assuming that $p_4(X)$, using all 5 nodes, is a better estimation of $f(X)$ than both $p_2(X)$ and $p_3(X)$, we can estimate the errors of $p_2(X)$ and $p_3(X)$ as their differences to $p_4(X)$. Of course this assumption is not granted either, but at least the circularity is not so clear because we are not assuming that p_3 is "better" than p_2 in order to decide precisely that[5]. Let's follow this path and see what gives:

We already have $e_{3,4}$: $\quad e_{3,4} = p_4(X) - p_3(X) = 0.000573552$

As for $e_{2,4}$: $\quad e_{2,4} = p_4(X) - p_2(X) = [p_4(X) - p_3(X)] + [p_3(X) - p_2(X)] = e_{3,4} + e_{2,3} =$

$$= 0.000573552 + 0.00623552 = 0.006809072$$

So our estimation of the error of $p_3(X)$ is still less than 10% of that of $p_2(X)$:

$$e_{3,4} / e_{2,4} = 0.000573552 / 0.006809072 = 0.0842335049475171$$

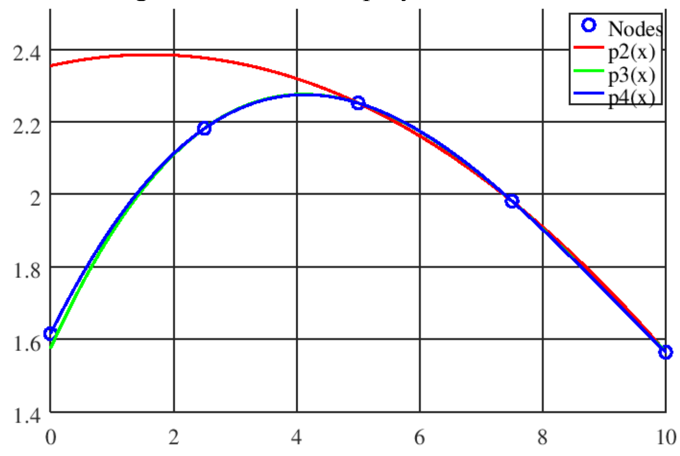
In this case there is no apparent contradiction: $p_4(X)$ can indeed be a much better approximation to $f(X)$ than both $p_2(X)$ and $p_3(X)$, and still $p_3(X)$ be better than $p_2(X)$. However –and it is not

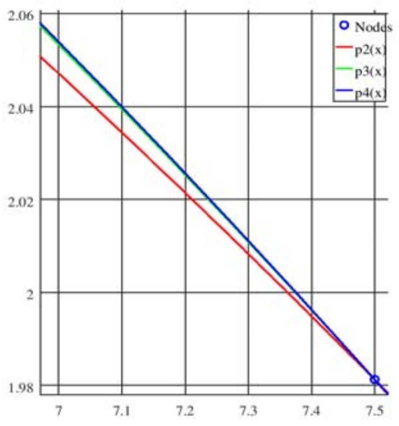
[5] But we are assuming that p_4 is better than p_3, which amounts to the same…

hard to find such cases– imagine that $p_2(X)=1$, $p_3(X)=1.1$, $p_4(X)=0.9$ (or 0.99). Is p_3 better than p_2? Not according to p_4, which is supposedly the best of the three!

The truth is that one cannot automatically assume that adding a new node makes the new interpolation polynomial any better, and that is exactly the principle applied when we estimate truncation errors in the usual way.

Here is a figure with the three polynomials.





Finally, something intriguing for you. The inverse error estimation ratios are:

$$e_{2,4}/e_{3,4}=0.006809072/0.000573552=11.8717605378414$$
$$e_{2,3}/e_{3,4}=0.00623552/0.000573552=10.8717605378414$$

I ignore why all the decimal digits coincide.

b) Using a numerically optimal method, and according to the answer to the previous section, estimate the intensity of the signal at point X. (2 points)

"Numerically optimal" means we have to evaluate $p_3(X)$ using the Hörner-like algorithm. In this case we will be using nodes B, C, D, E, so:

$$x=B+th=2.5+2.5t \quad\Rightarrow\quad t=(x-2.5)/2.5$$
$$\Rightarrow\quad t(x=X=7)=(7-2.5)/2.5=1.8$$

and the finite differences to use are the last elements of last 4 rows. We must evaluate $q(1.8)$:

$$q(t)=2.18174+0.07029\binom{t}{1}-0.34108\binom{t}{2}+0.19486\binom{t}{3}=$$
$$=2.18174+0.07029t-0.34108\frac{t(t-1)}{2!}+0.19486\frac{t(t-1)(t-2)}{3!}=$$
$$=2.18174+t\left\{0.07029+\frac{t-1}{2}\left[-0.34108+\frac{t-2}{3}0.19486\right]\right\}=$$
$$\underbrace{=2.18174+\underbrace{1.8\underbrace{\left\{0.07029+\underbrace{\frac{0.8}{2}\underbrace{\left[-0.34108+\underbrace{\frac{-0.2}{3}0.19486}_{-0.012990\hat{6}}\right]}_{-0.354070\hat{6}}}_{-0.1416282\hat{6}}\right\}}_{-0.0713382\hat{6}}}_{-0.12840888}}_{2.05333112}=\boxed{2.05333112}$$

2.- a) We know the cubic spline with boundary conditions determined by the nodes $x_0 < x_1 < \ldots < x_n$ and the values $f(x_0), f(x_1), \ldots, f(x_n), f'(x_0), f'(x_n)$ is optimal. Explain in what sense, and state precisely the corresponding theoretical result. (1.5 points)

Theory (straight from the Classroom Notes).

b) One wants to build a quadratic spline of class C^1 with the nodes $x_0 < x_1 < \ldots < x_n$ and the corresponding ordinates $y_0, y_1, \ldots, y_n$. Can it be done ensuring that $f'(x_0) = f'(x_n) = 0$? Justify the answer. (1.5 points)

No. The first polynomial piece $p_0(x)$ of degree ≤ 2 is uniquely determined by the three conditions $p_0(x_0)=y_0$, $p_0'(x_0)=0$, $p_0(x_1)=y_1$ (because of the uniqueness of osculating polynomials). Call $p_0'(x_1)=y'_1$. Then, for the same reason, the second polynomial piece $p_1(x)$ of degree ≤ 2 is uniquely determined by the three conditions $p_1(x_1)=y_1$, $p_1'(x_1)=y'_1$, $p_1(x_2)=y_2$ (the second one because $s \in C^1$). Etc., until the last piece $p_{n-1}(x)$, which will also be uniquely determined. It would be a big coincidence that its derivative at x_n is precisely 0 (x_n can be anywhere).

An even simpler way to prove this is with only two nodes (because a counterexample suffices to prove the general statement is false). If $y_0 \neq y_1$, we would need a 2nd-degree parabola whose first derivative vanishes on two different points and is not constant, which is not possible because it must be linear.

3.- Calculate $\int_0^{\pi} e^{\cos x} dx$ with 0.5% 'precision' using Gauss quadrature. (3.5 points)

If I had the Gauss-Legendre quadrature table I would probably use it; but since I don't, I will apply the most obvious change of variable. My suspicion is that the integral will become one of Gauss-Chebyshev, for which we don't need any table:

$$I = \int_0^{\pi} e^{\cos x} dx \underset{\substack{\cos x = t \\ -\sin x\, dx = dt}}{=} \int_1^{-1} e^t \frac{dt}{-\sqrt{1-t^2}} = \int_{-1}^{1} \frac{e^t}{\sqrt{1-t^2}} dt$$

And the integral does become a Gauss-Chebyshev one. The nodes are the Chebyshev ones, or the roots of $T_n(t) = \cos(n \operatorname{acos} t)$, while the weights are all equal and their sum is π (see the theory for a quick justification of this). The following calculations with Octave are also easy to do with a calculator:

```
f = @exp;                    % f(x) is just the numerator
ti = 0; wi = pi;             % 1 node = Midpoint
format long g
Q1 = wi * sum(f(ti))         % quadrature rule with 1 node
    Q1 =      3.14159265358979
ti = [-cos(pi/4) cos(pi/4)], wi = pi / 2
    ti =    -0.707106781186548      0.707106781186548
    wi =       1.5707963267949
Q2 = wi * sum(f(ti))         % with 2 nodes
    Q2 =      3.96026605279076
```

```
abs((Q2 - Q1) / Q2) * 100  % termination criterion: if <= 0.5
    ans =     20.6721818253613
ti = [-cos(pi/2 / 3) 0 cos(pi/2 / 3)], wi = pi / 3
    ti =    -0.866025403784439      0     0.866025403784439
    wi =       1.0471975511966
Q3 = wi * sum(f(ti))
    Q3 =     3.97732196008232
abs((Q3 - Q2) / Q3) * 100
    ans =    0.428828932199507
```

Since this is less than 0.5, we'll stay with $\underline{I \approx 3.97732196008232}$

Additionally we can check that the 0.5% is indeed a precision and not just a termination criterion:

```
Q = quadgk(@(x) exp(cos(x)), 0, pi)  % adaptive Gauss-Konrod, precise!
    Q =     3.97746326050642
abs((Q3 - Q) / Q) * 100
    ans =  0.00355252619201071
```

So yes, actually much better than 0.5%.

Finally, if you don't find the change of variable $\cos x = t$, you can still use Gauss-Chebyshev rules by adding the needed weight function "artificially". First we apply the linear change of variable from $[0, \pi]$ onto $[-1, 1]$:

$$I = \int_0^\pi e^{\cos x} dx \underset{\substack{x=\frac{\pi}{2}+\frac{\pi}{2}t \\ dx=\frac{\pi}{2}dt}}{=} \int_{-1}^1 e^{\cos\left(\frac{\pi}{2}+\frac{\pi}{2}t\right)} \frac{\pi}{2} dt$$

and now we multiply and divide by Chebyshev's weight function:

$$I = \int_{-1}^1 \frac{e^{\cos\left(\frac{\pi}{2}+\frac{\pi}{2}t\right)} \pi/2\sqrt{1-t^2}}{\sqrt{1-t^2}} dt = \int_{-1}^1 \frac{f(t)}{\sqrt{1-t^2}} dt$$

where $f(t)$ is the whole numerator. Now, with Octave:

```
f = @(t) exp(cos(pi/2 + pi/2 * t)) * pi/2 .* sqrt(1 - t.^2);
ti = 0; wi = pi;            % 1 node = Midpoint
Q1 = wi * sum(f(ti))        % quadrature rule with 1 node
    Q1 =     4.93480220054468
ti = [-cos(pi/4) cos(pi/4)]; wi = pi / 2;
Q2 = wi * sum(f(ti))        % with 2 nodes
    Q2 =     4.98643721105870
abs((Q2 - Q1) / Q2) * 100  % termination criterion: if <= 0.5
    ans =    1.03550908852323
```

which is greater than 0.5, so we would have to continue. Below is what one would so obtain. Here, **`i`** is the number of nodes, **`Q_i`** is the result of the corresponding quadrature rule, and **`I`** is the exact value of the integral. The third column shows errors as percentages of **`I`**, and the last one stopping criterions as percentages of the latest values:

```
 i                 Q_i     (I - Q_i) / I     (Q_i - Q_{i-1}) / Q_i
--                 ---     -------------     ---------------------
 1    4.93480220054468          -24.0690%                       NaN
 2    4.98643721105870          -25.3672%                   1.0355%
 3    4.14116497563213           -4.1157%                 -20.4114%
 4    4.10312333621605           -3.1593%                  -0.9271%
 5    4.05847491494595           -2.0367%                  -1.1001%
 6    4.03312526227304           -1.3994%                  -0.6285%
 7    4.01835295370568           -1.0280%                  -0.3676%
 8    4.00873655716945           -0.7862%                  -0.2398%
 9    4.00215449295299           -0.6207%                  -0.1644%
10    3.99745195462337           -0.5025%                  -0.1176%
11    3.99397571790357           -0.4151%                  -0.0870%
```

Observe that we would have to continue until $i = 7$ nodes to meet the stopping criterion ($0.3676\% < 0.5\%$), but the relative error would still be -1.0280%, so worse than 0.5%. Until reaching 11 nodes we would not obtain an error less than 0.5% with respect to the exact value of the integral. Gauss integration has lost much of its power in this way. Even Newton-Cotes rules do much better.

This also shows that a given stopping criterion does not always guarantee the corresponding real precision.

4.- Calculate $\int_1^3\left(\int_2^4 (7y^3+y^2x)dy\right)dx$ exactly with Newton-Cotes formulas and the least possible computational cost. Justify the choice of the formulas used. (3 points)

It's a polynomial of degree 1 in *x*, so the Midpoint rule will be exact in the *x* direction (polynomial degree of exactitude 1); and of degree 3 in *y*, so in that direction we need 3 nodes (polynomial degree 3—remember we gain one unit from the minimum 2 guaranteed by construction when it's a N-C rule with an odd number of nodes). In this *y* direction we can use the open or the closed Newton-Cotes rule; I will use the closed one (Simpson) because it is more common[6]. Hence: $\int_1^3 dx\int_2^4(7y^3+y^2x)dy = \int_1^3 dx\left[\frac{h_y}{3}(f_0+4f_1+f_2)\right] =$

$$= \int_1^3 dx\left[\frac{1}{3}\Big((7\cdot 2^3+2^2x)+4(7\cdot 3^3+3^2x)+(7\cdot 4^3+4^2x)\Big)\right] =$$

$$= \int_1^3 \frac{1}{3}\big(1260+56x\big)dx = 2h_x f_0 = 2\cdot 1\cdot\frac{1}{3}(1260+56\cdot 2) = \boxed{\frac{2744}{3} = 914.\hat{6}}$$

We can also do it in the other order of integration:

$$I = \int_1^3 dx\int_2^4(7y^3+y^2x)dy = \int_2^4 dy\int_1^3(7y^3+y^2x)dx = \int_2^4 dy\big[(3-1)(7y^3+y^2 2)\big] =$$

$$= \int_2^4(14y^3+4y^2)dy = \frac{1}{3}\big[(14\cdot 2^3+4\cdot 2^2)+4(14\cdot 3^3+4\cdot 3^2)+(14\cdot 4^3+4\cdot 4^2)\big] = 2744/3$$

With Octave:

[6] So I know its formula by heart: $h/3\,(f_0+4f_1+f_2)$. If you don't, you can easily derive it via integration of Lagrange base functions, or of Newton polynomials with equally-spaced nodes, or via indeterminate coefficients.

```
format long g
f = @(y) 14 * y^3 + 4 * y^2;
1/3 * (f(2) + 4 * f(3) + f(4))
    ans =      914.666666666667
```

And, to be sure, we can also calculate the exact value of the integral using Barrow's Law (Fundamental Theorem of Calculus) and go home with peace of mind:

$$\int_1^3 dx \int_2^4 (7y^3 + y^2 x)dy = \int_1^3 dx \left[7\frac{y^4}{4} + \frac{y^3}{3}x \right]_{y=2}^{y=4} = \int_1^3 dx \left[\left(7\frac{4^4}{4} + \frac{4^3}{3}x \right) - \left(7\frac{2^4}{4} + \frac{2^3}{3}x \right) \right] =$$

$$= \int_1^3 dx \left(420 + \frac{56}{3}x \right) = \left[420x + \frac{56}{3}\frac{x^2}{2} \right]_1^3 = 420(3-1) + \frac{56}{6}(9-1) = 914.\hat{6}$$

just like before.

5.- The position in space of a moving body is described by the following system of differential equations:

$$\begin{cases} x'(t) = x(t) - y(t) + t\,z(t) \\ y'(t) = -x(t) - y(t) + z(t) \\ z'(t) = t\,x(t) + y(t) - z(t) \end{cases}$$

At the initial instant the body is at point $(1, 0, -1)$. Use the **Enhanced Euler (Heun) method** to estimate its position at instants 0.1 and 0.2 (step size $h = 0.1$). (5.5 points)

We can obtain the advance formula of the Enhanced Euler or Heun method from the one of the Trapezoidal method: $$y_{n+1} = y_n + \frac{f(t_n, y_n) + f(t_{n+1}, y_{n+1})}{2} h_n$$

and substituting $y_{n+1} = y_n + f(t_n, y_n)\,h_n$ (Euler) on its right-hand side to obtain the (explicit) Heun method: $$y_{n+1} = y_n + \frac{f(t_n, y_n) + f(t_{n+1}, y_n + f(t_n, y_n)h_n)}{2} h_n$$

Now, if we want, we can rewrite this with the typical Runge-Kutta notation in terms of k_i, with $t_{n+1} = t_n + h_n$, and, for systems, with vector instead of scalar notation:

$$\mathbf{k_1} = \mathbf{f}(t_n, \mathbf{y_n})h_n$$
$$\mathbf{k_2} = \mathbf{f}(t_n + h_n, \mathbf{y_n} + \mathbf{k_1})h_n$$
$$\mathbf{y_{n+1}} = \mathbf{y_n} + \frac{\mathbf{k_1} + \mathbf{k_2}}{2}$$

Now calling $$\mathbf{y} = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

the system to be solved becomes $$\mathbf{y}' = \begin{pmatrix} y_1 - y_2 + ty_3 \\ -y_1 - y_2 + y_3 \\ ty_1 + y_2 - y_3 \end{pmatrix} = \mathbf{f}(t, \mathbf{y})$$

and the initial conditions: $$t_0 = 0\,; \quad \mathbf{y_0} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

We are asked to take two steps of size $h = 0.1$. Everything is ready for an almost-literal transcription to Octave, showing both the calculations to be done and their numerical results:

```
format long g
f = @(t, y) [y(1)-y(2)+t*y(3); -y(1)-y(2)+y(3); t*y(1)+y(2)-y(3)];
h = 0.1; t0 = 0; t1 = h; t2 = 2 * h;
y0 = [1; 0; -1];
k1 = f(t0, y0) * h                % k1 for 1st step
   k1 =              0.1
                    -0.2
                     0.1
k2 = f(t0 + h, y0 + k1) * h       % k2 for 1st step
   k2 =            0.121
                   -0.18
                   0.081
y1 = y0 + (k1 + k2) / 2           % 1st step completed
   y1 =           1.1105
                   -0.19
                 -0.9095
k1 = f(t1, y1) * h                % k1 for 2nd step (overwritten)
   k1 =         0.120955
                  -0.183
                0.083055
k2 = f(t1 + h, y1 + k1) * h       % k2 for 2st step (overwritten)
   k2 =        0.1439166
                -0.16849
               0.0699736
y2 = y1 + (k1 + k2) / 2           % 2nd step completed
   y2 =        1.2429358
               -0.365745
              -0.8329857
```

We can also check with our function **`anm_ode`**:

```
[tout, yout] = anm_ode(f, [0 0.2], y0, 2, 'Heun')
   tout =              0                   0.1                 0.2
   yout =              1                1.1105           1.2429358
                       0                 -0.19           -0.365745
                      -1               -0.9095          -0.8329857
```

The numbers coincide exactly, so
  the position at $t = 0.2$ is estimated to be $(x_2, y_2, z_2) = (1.2429358, -0.365745, -0.8329857)$

6.- Find a and b for the method $y_n = y_{n-2} + h\,[a\,f_n + b\,f_{n-3}]$ to be convergent with the maximum possible order. Is the method obtained explicit or implicit? Of how many steps? Justify the answers. (3 points)

The method is implicit (unless a turns out to be zero) because the unknown y_n appears also on the right-hand side of the advance formula used to obtain it, where $f_n = f(t_n, y_n)$. In other words, the advance formula is an *implicit* equation in the unknown y_n.

The number of steps is $k = 3$ (unless b turns out to be zero) because the maximum index is n and the minimum one is $n-3$; $k = n-(n-3) = 3$ (the 1st step being from t_{n-3} to t_{n-2}, the 2nd one from t_{n-2} to t_{n-1}, and the 3rd one from t_{n-1} to t_n).

To find a, b we apply the steps described in "the theory". First we identify coefficients with the general linear multistep method's advance formula:

$$y_{n+k} = -\sum_{i=0}^{k-1} \alpha_i y_{n+i} + h\sum_{j=0}^{k} \beta_i f_{n+i} \qquad (\alpha_k = 1)$$

Since $k=3$: $\quad y_{n+3} = -\alpha_0 y_n - \alpha_1 y_{n+1} - \alpha_2 y_{n+2} + h\left(\beta_0 f_n + \beta_1 f_{n+1} + \beta_2 f_{n+2} + \beta_3 f_{n+3}\right)$

To identify with the formula given it is easier to add 3 to all its indices:

$$y_{n+3} = y_{n+1} + h\left(a f_{n+3} + b f_n\right)$$

Hence: $\quad \alpha_0 = 0;\quad \alpha_1 = -1;\quad \alpha_2 = 0;\quad (\alpha_3 = 1)$
$\beta_0 = b;\quad \beta_1 = 0;\quad \beta_2 = 0;\quad \beta_3 = a$

First characteristic polynomial: $\quad \rho(z) = -z + z^3$
Second characteristic polynomial: $\quad \sigma(z) = b + a z^3$

Consistency: $\quad \rho(1) = -1 + 1^3 = 0$ ok
$\rho'(z) = -1 + 3z^2;\quad \rho'(1) = -1 + 3\cdot 1^2 = 2 = \sigma(1) = b + a\,1^3 \quad \Rightarrow \quad \underline{a + b = 2}$

Stability: $\quad \rho(z) = -z + z^3 = 0 \quad \Rightarrow \quad z = 0, \quad z = \pm 1$
so all the roots have complex modulus ≤ 1 and, the two with modulus 1, are simple: ok.

Convergence: we will make sure that the method is consistent (by making $a+b=2$) and have checked that it will stable, so it will be convergent, with order of convergence p, iff:

$$\frac{1}{m}\sum_{j=0}^{k} \alpha_j j^m = \sum_{j=0}^{k} \beta_j j^{m-1} \qquad \text{for } m = 1, 2, \ldots, p, \text{ but not } p{+}1 \quad (\text{and with } 0^0 = 1)$$

$m=1$: $\quad \frac{1}{1}\left(-1\cdot 1^1 + 1\cdot 3^1\right) = b0^0 + a3^0 = b + a \quad \Rightarrow \quad a + b = 2 \quad$ like before

$m=2$: $\quad \frac{1}{2}\left(-1\cdot 1^2 + 1\cdot 3^2\right) = b0^1 + a3^1 \quad \Rightarrow \quad 3a = 4 \quad \Rightarrow \quad \boxed{a = \frac{4}{3}, \quad b = \frac{2}{3}}$

with order at least 2. Let's see if it is greater than 2:

$m=3$: $\quad \frac{1}{3}\left(-1\cdot 1^3 + 1\cdot 3^3\right) = b0^2 + a3^2 \quad \Rightarrow \quad \frac{26}{3} = 9a = 9\frac{4}{3} = 12$

and since $26/3 \neq 12$ (abusing notation a bit too much above) the order is not greater than 2 so
<u>the order of convergence is $p = 2$</u>

7.- a) Obtain a numerical differentiation formula, as well as its error term in its simplest form, to estimate $f'(z)$ from the values of f at the nodes x_0, $x_1 = x_0 + h$, $x_2 = x_1 + 2h$, with $z = x_0 + 0.5h$. Do it using Taylor series. (4 points)

I'm not sure I will immediately know how to manipulate Taylor series ad-hoc and I don't have time to waste, so I will apply the general theory with remainder of the least possible order. In this case $k = 1$ (estimating 1st derivative), $n = 2$ (3 nodes—more than enough to estimate f'), $h_0 = -h/2$, $h_1 = h/2$, $h_2 = 5h/2$ (nodal locations), and $m = 2$ (equal to n, for the remainder of the least possible order). As usual:

$$f'(z) = D + E = A_0 f\left(z - \frac{h}{2}\right) + A_1 f\left(z + \frac{h}{2}\right) + A_2 f\left(z + \frac{5h}{2}\right) + E$$

Then, if $f \in C^3([x_0, x_2])$, for some ξ_i between z and each node x_i:

$$f'(z) = A_0\left[f(z) + f'(z)\frac{-h}{2} + \frac{f''(z)}{2!}\left(\frac{-h}{2}\right)^2 + \frac{f^{3)}(\xi_0)}{3!}\left(\frac{-h}{2}\right)^3 \right] +$$
$$+A_1\left[f(z) + f'(z)\frac{h}{2} + \frac{f''(z)}{2!}\left(\frac{h}{2}\right)^2 + \frac{f^{3)}(\xi_1)}{3!}\left(\frac{h}{2}\right)^3 \right] +$$
$$+A_2\left[f(z) + f'(z)\frac{5h}{2} + \frac{f''(z)}{2!}\left(\frac{5h}{2}\right)^2 + \frac{f^{3)}(\xi_2)}{3!}\left(\frac{5h}{2}\right)^3 \right] + E =$$
$$= f(z)\underbrace{\left(A_0 + A_1 + A_2\right)}_{0} + f'(z)\underbrace{\left(\frac{-h}{2}A_0 + \frac{h}{2}A_1 + \frac{5h}{2}A_2\right)}_{1} + \frac{f''(z)}{2!}\underbrace{\left(\frac{h^2}{4}A_0 + \frac{h^2}{4}A_1 + \frac{25h^2}{4}A_2\right)}_{0} +$$
$$\underbrace{+A_0\frac{f^{3)}(\xi_0)}{3!}\left(\frac{-h}{2}\right)^3 + A_1\frac{f^{3)}(\xi_1)}{3!}\left(\frac{h}{2}\right)^3 + A_2\frac{f^{3)}(\xi_2)}{3!}\left(\frac{5h}{2}\right)^3 + E}_{0}$$

The first three equations are the same we would get by the method of indeterminate coefficients. They form a 3×3 linear system which we solve for A_0, A_1 and A_2 (or, rather, for $a_i = hA_i$ so that we can manipulate a purely numeric augmented matrix):

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ -1/2 & 1/2 & 5/2 & 1 \\ 1/4 & 1/4 & 25/4 & 0 \end{pmatrix} <> \begin{pmatrix} 1 & 1 & 1 & 0 \\ -1 & 1 & 5 & 2 \\ 1 & 1 & 25 & 0 \end{pmatrix} \begin{matrix} < R_2 + R_1 > \\ < R_3 - R_1 > \end{matrix} \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 2 & 6 & 2 \\ 0 & 0 & 24 & 0 \end{pmatrix}$$

From the third equation, $A_2 = 0$ (so x_2 has nothing to offer to estimate $f'(z)$! We know that can happen…) Substituting into the second equation, $a_1 = 1 \Rightarrow A_1 = 1/h$. And into the first, $A_0 = -1/h$. Therefore the formula is:

$$f'(z) = \frac{-1}{h}f\left(z - \frac{h}{2}\right) + \frac{1}{h}f\left(z + \frac{h}{2}\right) + E = \boxed{\frac{f(z + h/2) - f(z - h/2)}{h} + E}$$

Now we can find the error term E by substituting A_i into the fourth equation above:

$$A_0\frac{f^{3)}(\xi_0)}{3!}\left(\frac{-h}{2}\right)^3 + A_1\frac{f^{3)}(\xi_1)}{3!}\left(\frac{h}{2}\right)^3 + A_2\frac{f^{3)}(\xi_2)}{3!}\left(\frac{5h}{2}\right)^3 + E = 0 \Rightarrow$$
$$E = -A_0\frac{f^{3)}(\xi_0)}{3!}\left(\frac{-h}{2}\right)^3 - A_1\frac{f^{3)}(\xi_1)}{3!}\left(\frac{h}{2}\right)^3 + 0 = -\frac{-1}{h}\frac{f^{3)}(\xi_0)}{6}\left(\frac{-h}{2}\right)^3 - \frac{1}{h}\frac{f^{3)}(\xi_1)}{6}\left(\frac{h}{2}\right)^3 =$$
$$= \frac{-h^3 f^{3)}(\xi_0) - h^3 f^{3)}(\xi_1)}{48h} = \frac{-h^2}{24}\frac{f^{3)}(\xi_0) + f^{3)}(\xi_1)}{2} = \boxed{E = \frac{-h^2}{24}f^{3)}(\xi)}$$

for some ξ between ξ_0 and ξ_1 and, therefore, also between $x_0 = z - h/2$ and $x_1 = z + h/2$. (Here we have applied Weierstrass's (sometimes called Bolzano's) Intermediate Value Theorem: if $f \in C^3([\xi_0, \xi_1])$, there must exist some intermediate point ξ where $f^{3)}$ takes as value the arithmetic mean of $f^{3)}(\xi_0)$ and $f^{3)}(\xi_1)$, because a mean is always an intermediate value between the maximum and the minimum).

Of course by now you have already noticed that the formula obtained is exactly like one of the first ones studied in the chapter, only substituting h for $h/2$, so the error term can also be obtained as we did there and then change h for $h/2$. Or manipulating Taylor series ad-hoc, which, in this case, is very easy. I'm leaving that exercise for you.

Finally, let's check that the info given by our function **`anm_difftaylor`**, which implements this theory systematically, is compatible with what we found:

```
k = 1; hi = [-1/2 1/2]; m = 2;
anm_difftaylor(k, hi, m);
    f'(z) = D + E   where   D = sum(Ai .* f(xi))   where:
        xi = z + h*[-1/2 1/2]   and   Ai = [-1 1]/h   so:
    D = (-f(z - 0.5*h) + f(z + 0.5*h)) / h;
        Polynomial degree N = 2;  Order of convergence O = 2.
        Error term:
    E = 1/factorial(3) * (-1/8 * f3(\xi_0) - 1/8 * f3(\xi_1)) * h^2
        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) = 1/24*M*h^2  where M >= |f3(x)| for all x bt nodes and z.
    g2(h) = AF*ep = 2*h^-1*ep where ep >= |fi - fibar| for all i = 0:n.
    |Etot| <= g1(h) + g2(h) = g(h) =  1/24*M*h^2 + 2*h^-1*ep
    h opt => g min => g'(h) = 2 * 1/24 * M * h - 1 * 2 * h^-2 * ep = 0
    => h_opt = (24*ep/M)^(1/3) = 2.8844991406148 * ep^(1/3) * M^(-1/3)
    g_min = g(h_opt)  =  1.040041911525952 * M^(1/3) * ep^(2/3)
```

Both D and E coincide, except that `anm_difftaylor` returns the error term E prior to applying Weierstrass's Intermediate Value Theorem (because this is something that can't always be done).

b) Justify if the formula obtained in section a) would give the exact value of $f'(x_0+0.5h)$ for this function:

$$f(x)=\begin{cases} q(x) & x\le x_1 \\ r(x) & x>x_1 \end{cases}$$

where q and r are polynomials of degrees 2 and 3, respectively. (0.5 points)

A posteriori we know that our formula uses only two nodes (because multiplying the third nodal ordinate by zero is not really using it[7]). Therefore the situation is exactly as if, from the beginning, we look for the formula of two nodes. To find the two coefficients we can impose two conditions, namely, the exact differentiation of the monomials 1, x, and therefore also of any linear combination of them. Hence the polynomial degree will be $N\ge 1$. However, from the theory, we know that when we are estimating $f'(z)$ with an even number of nodes symmetrically located on both sides of z, the polynomial degree increases exactly by 1. In our case, $N=2$.

On the other hand, since the nodes used by the formula are x_0 and x_1, both $\le x_1$, the only polynomial piece of $f(x)$ that is actually evaluated is $p(x)$, which is of degree 2, leaving $r(x)$ (of degree 3) unused. This means that the polynomial degree $N=2$ is enough and, save rounding errors the derivative said is indeed calculated exactly.

c) For $f(x)=\mathrm{Ln}(3x)$, $x_0=2$, and precision $\varepsilon=10^{-4}$, calculate the optimal step size h_{opt} to apply the formula obtained in section a). (1.5 points)

Calling E the truncation error, $E_r=D-\bar{D}$ the rounding error, and E_{tot} the total error:

$$E_{tot}=f'(x)-\bar{D}=(f'(x)-D)+(D-\bar{D})=E+E_r$$

so $$|E_{tot}|\le|E|+|E_r|$$

[7] Simpson would agree (not only Thomas, even Bart).

Now $$|E| = \left| \frac{-h^2}{24} f^{3)}(\xi) \right| \le \frac{h^2}{24} M$$

where M is an upper bound of $|f^{3)}(\xi)|$ for ξ between the nodes.

Regarding E_r: $$|E_r| \le \varepsilon \cdot AF$$

where the amplification factor AF is: $$AF = |A_0| + |A_1| = \frac{1}{h} + \frac{1}{h} = \frac{2}{h}$$

Hence $$|E_{tot}| \le \frac{h^2}{24} M + \varepsilon \frac{2}{h} = g(h)$$

We minimize $g(h)$ at the optimal step size h_{opt} by making $g'(h_{opt}) = 0$:

$$\frac{2h_{opt}}{24} M - 2\varepsilon h_{opt}^{-2} = 0 \quad \Rightarrow \quad h_{opt} = \sqrt[3]{\frac{24\varepsilon}{M}}$$

which also agrees with the output of **`anm_difftaylor`** above.

We're told that $\varepsilon = 10^{-4}$. As for M:

$$f(x) = \log(3x) \quad \to \quad f'(x) = \frac{1}{3x} 3 = x^{-1} \quad \to \quad f''(x) = -x^{-2} \quad \to \quad f^{3)}(x) = \frac{2}{x^3}$$

This is a strictly monotonically decreasing function from $x_0 = 2$ to the right (where z and the other node lie), so we can take $M = 2/2^3 = 0.25$. Substituting values:

```
format long g
ep = 1e-4; M = 0.25;
h_opt = (24 * ep / M)^(1/3)
    h_opt =     0.212531713836522
```

Hence the optimal step size is $\boxed{h_{opt} = 0.2125317138365}$

It is so large because the precision $\varepsilon = 10^{-4}$ is too bad. A more realistic value, for double-precision arithmetic, would be:

```
f = @(x) log(3 * x);
ep = eps(f(2)) / 2       % see "the theory" for explanation on this
    ep = 1.11022302462516e-016
h_opt = (24 * ep / M)^(1/3)
    h_opt = 2.2006981968802e-005
```