ADVANCED NUMERICAL METHODS
Ordinary exam call, 21/5/2018.
FULL ANSWERS / RESOLUTION

Exercise 1

Minimalistic answer (worth of full points):

An osculating polynomial $p(x)$ of $f(x)$ is one that must coincide with $f(x)$ on its values and a number of *successive* derivatives at the nodes:

$$p(x_i) = f(x_i)\,,\quad p'(x_i) = f'(x_i)\,,\quad p''(x_i) = f''(x_i)\,,\quad \ldots,\quad p^{m_i)}(x_i) = f^{m_i)}(x_i)\quad (i = 0,1,\ldots,n)$$

(1p)

Calling $k_i = m_i + 1$ the number of conditions imposed on node x_i, if a total of $k = \Sigma\, k_i$ conditions are imposed, the error term can be proved to satisfy (assuming $f \in C^k$):

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

for some ξ between the nodes and x. **(1p)**

More detailed explanations:

Osculating polynomials[1], like Lagrange interpolation polynomials, are also polynomials that must coincide with a given function $f(x)$ on a set of $n+1$ nodes $x_0, x_1, \ldots, x_n$; but, at least on one of them, there must be coincidence not only on the values of the functions, but also on a number of *successive* derivatives:

$$p(x_i) = f(x_i)\,,\quad p'(x_i) = f'(x_i)\,,\quad p''(x_i) = f''(x_i)\,,\quad \ldots,\quad p^{m_i)}(x_i) = f^{m_i)}(x_i)\quad (i = 0,1,\ldots,n)$$

The order of differentiation m_i does not need to coincide for all the nodes.

Call k_i the number of interpolation data used on node x_i. That number is $k_i = m_i + 1$, since we use the values of f and of its first m_i derivatives. The total number of interpolation data used will be noted k, so $k = \Sigma\, k_i$ $(i = 0, \ldots, n)$. In that case, it can be proved that the osculating polynomial of degree $k-1$ or less exists and is unique.

For example, Hermite osculating polynomials have $m_i = 1$ at all $n+1$ nodes, where the osculating polynomial's value and first derivative must coincide with those of $f(x)$. The total number of conditions is $k = 2(n+1) = 2n+2$, and there will be one and only one polynomial of degree $\leq 2n+1$ satisfying those conditions. Similarly, Taylor polynomials (which are truncated Taylor series expansions) have $n = 0$ (only one node) where $p(x)$ and its derivatives until the m-th one must coincide with those of f. In this case the number of conditions used is $k = m+1$ and the degree of the unique Taylor polynomial will be $\leq m$.

Osculating interpolation polynomials can be considered as a generalization of Lagrange polynomials with nodes of multiplicity greater than 1 ("multiple nodes"), i.e., nodes that can be "infinitely close to one another" in such a way that they would define the value of a number of derivatives of f or of p on them. The corresponding tables of differences are called tables of divided differences *with repetitions*. If, for example, three conditions are used on node x_0, it can be renamed (with repetitions) as z_0, z_1, z_2; then if two conditions are used on node x_1, it

[1] The term "osculating" comes from Latin and means "kissing", which is quite appropriate given the kind of close contact both functions, p and f, exhibit at the osculating points.

can be renamed as z_3, z_4, etc. If there are k conditions overall, the set of all nodes can be renamed as $z_0, z_1, \ldots, z_{k-1}$ (with some values being repeated).

Divided differences can also be generalized to multiple or repeated nodes, and in that case the error term of the osculating polynomial $p(x)$ can be written as:

$$e(x) = f(x) - p(x) = f[z_0, z_1, \ldots, z_{k-1}, x]\,\Pi(x)$$

The error term can also be proved to be of this general form:

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

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.

Exercise 2

a) Not all the nodes are equally-spaced, so I will use divided differences. The ones of order 3 must be (for some ξ between the nodes, but this is irrelevant here because $f^{3)}$ is constant):

$$f_{i,3} = \frac{f^{3)}(\xi)}{3!} = \frac{12}{6} = 2$$

Hence I will start by building a table of divided differences and see if removing one of the nodes produces a table with a constant column $f_{i,3} = 2$. With the help of Octave (you should know how to calculate these numbers):

```
clear
format long
xi = [0  1  3   4   5   6];
yi = [1  6 70 153  86 481];
[fii, zi, table] = anm_tableddr(xi, yi);
table                       % (some of the output suppressed)
    i    x_i    f_i,0    f_i,1    f_i,2    f_i,3    f_i,4    f_i,5
    0     0        1
    1     1        6        5
    2     3       70       32        9
    3     4      153       83       17        2
    4     5       86      -67      -75      -23        .
    5     6      481        .        .        .        .        .
```

When doing this by hand, as soon as I find $f_{3,3} = 2$ I accept the first four interpolation points (from which that number *2* stems) as correct; and as soon as I find $f_{4,3} = -23 \neq 2$ I discard the fifth ordinate, 86, as invalid (because that *−23* stems from the first five interpolation points, and I accepted the first four as correct). So I don't actually calculate any more numbers. I leave them blank, discard the 86, add two more 2's to column $f_{i,3}$, and try to fill in the gaps in this table:

```
    i    x_i    f_i,0    f_i,1    f_i,2    f_i,3    f_i,4    f_i,5
    0     0        1
    1     1        6        5
    2     3       70       32        9
    3     4      153       83       17        2
    4     5        .        .        .        2        0
    5     6      481        .        .        2        0        0
```

It is possible to advance locally from the $f_{i,3}$ constant column to the left, looking for triangles shaped **:.** where two of the three elements are known. If it is true that only one datum is incorrect, it should be possible to continue in this way and find again the value $f_{5,0} = 481$. This can be easily done by hand, but there is a function called **`anm_fillinddr`**, available from the usual repository, that can do it for us. Here is how to use it:

```
T = table(:, 2:end);           % `table` from previous output
T(5:6, 2:end) = NaN;
T(5:end, 5) = 2
    T =
        0     1   NaN   NaN   NaN   NaN   NaN
        1     6     5   NaN   NaN   NaN   NaN
        3    70    32     9   NaN   NaN   NaN
        4   153    83    17     2   NaN   NaN
        5   NaN   NaN   NaN     2   NaN   NaN
        6   NaN   NaN   NaN     2   NaN   NaN
anm_fillinddr(T)               % execute in Octave to see step-by-step
        0     1   NaN   NaN   NaN   NaN   NaN
        1     6     5   NaN   NaN   NaN   NaN
        3    70    32     9   NaN   NaN   NaN
        4   153    83    17     2   NaN   NaN
        5   286   133    25     2     0   NaN
        6   481   195    31     2     0     0
```

Indeed the value 481 appears again, and in the table it can be seen that

$f(5)=86$ should be replaced by $f(5)=286$ **(4p)**

b) The only functions with a constant non-zero third derivative are polynomials of degree 3, so $f(x)$ must be one—this is basic Differential Equations—. With six distinct nodes, as we have, the Lagrange interpolation polynomial of degree ≤ 5 exists and is unique; in this case, we know it is of degree $3 < 5$, so any 4 of the 6 nodes provided define the exact same polynomial $f(x)$ of degree 3.

Using only 3 nodes instead of 4 (with nodal ordinates) there are still infinitely many polynomials of degree ≤ 3 that pass by the corresponding 3 interpolation points (because the 4th interpolation point could be chosen arbitrarily). Hence, if I am not allowed to use any other interpolation point, I will have to use some other information about $f(x)$. It seems reasonable to use that $f^{3)}(x) \equiv 12$ (or $f_{i,3} = 2$). My plan is to define $f(x) = p_3(x)$ as an osculating polynomial in terms of the unknown $f'(3)$ as a parameter (3 being a double node), build the corresponding table of differences with repetitions, impose the condition that $f_{i,3} = 2$ and solve for $f'(3)$:

z_i	$f_{i,0}$	$f_{i,1}$	$f_{i,2}$	$f_{i,3}$
1	6	—	—	—
3	70	$(70-6)/(3-1)=32$	—	—
3	70	$f'(3)/1!$	$(f'(3)-32)/(3-1)$	—
4	153	$(153-70)/(4-3)=83$	$(83-f'(3))/(4-3)$	2

So the final element (2) should satisfy this:

$$2 = \frac{\dfrac{83-f'(3)}{1} - \dfrac{f'(3)-32}{2}}{4-1} \underset{\times 6}{\Rightarrow} 12 = 166 - 2f'(3) - f'(3) + 32$$

$$\Rightarrow \quad f'(3) = \frac{166+32-12}{3} = \frac{186}{3} = \boxed{f'(3) = 62} \qquad \textbf{(1.5p)}$$

c) Using *finite* (not divided) differences forces me to choose equally-spaced nodes. As we know, any 4 of the 6 will define $f(x)$ uniquely; but the only four equally-spaced nodes that we have are x_i = 3, 4, 5, 6. So I will build the corresponding table of finite differences:

```
xi = 3:6;
yi = [70 153 286 481];
[Diy0, table] = anm_tablefd(xi, yi);
table
```

```
%    i     xi       fi      Dfi     D2fi     D3fi
     0      3       70        -        -        -
     1      4      153       83        -        -
     2      5      286      133       50        -
     3      6      481      195       62       12
```

To work in terms of *t* instead of *x*, the following change of variable is applied:

$$x = x_0 + ht = 3 + t \quad \Rightarrow \quad t = -3 + x = t(x)$$

The Newton polynomial in terms of *t*, *q*(*t*), is:

$$q(t) = f(x_0)\cdot\binom{t}{0} + \Delta f(x_0)\cdot\binom{t}{1} + \Delta^2 f(x_0)\cdot\binom{t}{2} + \Delta^3 f(x_0)\cdot\binom{t}{3} =$$

$$= 70\cdot 1 + 83\cdot\frac{t}{1} + 50\cdot\frac{t(t-1)}{2\cdot 1} + 12\cdot\frac{t(t-1)(t-2)}{3\cdot 2\cdot 1} =$$

$$= 70 + \frac{t}{1}\left\{83 + \frac{t-1}{2}\left[50 + 12\frac{t-2}{3}\right]\right\} = 70 + t\left\{83 + \frac{t-1}{2}\left[50 + 4(t-2)\right]\right\} = q(t)$$

where we have taken as many common factors (nested products) as possible, including the ones in the denominators' factorials, because we will soon have to evaluate this polynomial at $x = 3.5$ *optimally* (from the point of view of computational cost and with good error propagation, i.e. with the Hörner-like algorithm).

First we are asked to obtain $f(x)$. Now $f(x) = q(t(x))$ where both $q(t)$ and $t(x)$ have been explicitly written above. Substituting:

$$\boxed{f(x) = 70 + (x-3)\left\{83 + \frac{x-4}{2}\left[50 + 4(x-5)\right]\right\}} \qquad \textbf{(1.5p)}$$

although it's also perfectly fine to give the answer in terms of $q(t)$ and $t(x)$, i.e. as

$$f(x) = q(t(x)) \quad \text{where} \quad q(t) = 70 + t\left\{83 + \frac{t-1}{2}\left[50 + 4(t-2)\right]\right\} \quad \text{and} \quad t(x) = x - 3\,.$$

To evaluate optimally at $x = 3.5$ we can use the expression explicitly written in terms of x above, but it is usually more convenient to calculate $t = t(3.5)$ and then evaluate $q(t)$:

$$t(3.5) = 3.5 - 3 = 0.5$$

$$f(3.5) = q(0.5) = 70 + 0.5\left\{83 + \frac{0.5-1}{2}\left[50 + 4(0.5-2)\right]\right\} =$$

$$= \underbrace{70 + \underbrace{0.5\{\underbrace{83 \underbrace{-0.25[\underbrace{50 + \underbrace{4(-1.5)}_{-6}}_{44}]}_{-11}\}}_{72}}_{36}}_{106} = \boxed{106}$$

The function **`anm_newtoneven`** corroborates the result: **(1.5p)**

```
[pp, coefs, chars] = anm_newtoneven(3.5, xi, yi); chars, pp
    chars =
      [1,1] = t(x) = (x-x0)/h = (x-3)/1 = 1*x + -3
      [2,1] = q(t) = 70 + 83*t/1! + 50*t*(t-1)/2! + 12*t*(t-1)*(t-2)/3!
      [3,1] = q(t) = 70 + 83*t + 25*t*(t-1) + 2*t*(t-1)*(t-2)
      [4,1] = q(t) = ((2*(t-2) + 25)*(t-1) + 83)*(t-0) + 70
      [5,1] = q(0.5) = ((2*(0.5-2) + 25)*(0.5-1) + 83)*(0.5-0) + 70
      [6,1] = q(0.5) = ((2*-1.5 + 25)*-0.5 + 83)*0.5 + 70
    pp =  106
```

Exercise 3

The form of the integral immediately suggests the change of variable $4x^2 = t^2$, or $t = 2x$, hoping to obtain some integral of the Gauss-Chebyshev type. (Another practical hint that this may be the way to go is the absence of an Appendix with Gauss nodes and weights.)

$$I = \int_0^{1/2} \frac{e^{-4x^2}}{\sqrt{1-4x^2}}dx \underset{t=2x}{=} \int_0^1 \frac{e^{-t^2}}{\sqrt{1-t^2}}\frac{dt}{2} \underset{\text{even}}{=} \int_{-1}^1 \frac{e^{-t^2}/4}{\sqrt{1-t^2}}dt$$

I will now apply Gauss-Chebyshev rules of 1, 2, 3… nodes until the distance between the last two values obtained does not exceed 1% of the last one. **(2p)**

One node ($n=0$): $x_0 = 0;\quad w_0 = \pi$

I will continue with Octave, which should tell you very precisely the operations to perform with your calculator. You can also copy-paste to the Octave command window and check the results for yourself:

```
clear
format long g
f = @(t) exp(-t.^2) / 4;
ti = 0; wi = pi;
Q1 = wi * f(ti)                        % rule of 1 node
    Q1 = 0.7853981633974483
ti = [cos(pi / 4), cos(3 * pi / 4)], wi = pi / 2;
    ti =  0.7071067811865476  -0.7071067811865475
Q2 = wi * sum(f(ti))                   % rule of 2 nodes...
    Q2 = 0.476368066182545
relative_distance_percent = abs((Q2 - Q1) / Q2) * 100
    relative_distance_percent =  64.87212707001282
ti = [cos(pi / 6), 0, cos(5 * pi / 6)], wi = pi / 3;
    ti =  0.8660254037844387  0  -0.8660254037844387
Q3 = wi * sum(f(ti))
    Q3 = 0.5091299364479339
relative_distance_percent = abs((Q3 - Q2) / Q3) * 100
    relative_distance_percent = 6.434874070450451
ti = cos([1 3 5 7] * pi / 8), wi = pi / 4;
    0.923879532511  0.382683432365  -0.382683432365  -0.923879532511
Q4 = wi * sum(f(ti))
    Q4 = 0.506452500898245
relative_distance_percent = abs((Q4 - Q3) / Q4) * 100
    relative_distance_percent = 0.5286646911487666
```

This is, for the first time, less than 1%, so

$$I \approx \boxed{Q_4 = 0.5064525} \quad \text{(stopping criterion 1\%)}$$ **(2p)**

There is a function in the usual repository that automates this:

```
[Q, info] = anm_quadgcheb(f, -100, 1)
    Q = 0.506452500898245
    info =
      1  0.7853981633974483   NaN                    NaN
      2  0.476368066182545   -0.3090300972149033    -0.6487212707001282
      3  0.5091299364479339   0.03276187026538896    0.06434874070450451
      4  0.506452500898245   -0.002677435549688911  -0.005286646911487666
```

To know more: the exact value of this integral cannot be expressed in terms of *elementary functions*, but it can in terms of a *special function* called a *modified Bessel function of the first kind*. Its more precise value is $I = 0.5066095167373\ldots$. When compared with this value, the error made by Q_4 with respect to it is about 0.031%, which is much better than the 1% "precision" of the stopping criterion. This is what happens most of the times (but not always): a given stopping criterion "precision" results in a value that is typically more precise than that.

Exercise 4

a) The first node $x_0 = a$ and the last one $x_8 = b$ are not used, so this is the *open Newton-Cotes rule of 7 nodes*. Since the number of nodes is odd, the polynomial degree is 1 unit higher than the minimum one, 6, guaranteed by construction with 7 nodes, so $N = 6 + 1 = 7$ and $m = N + 1 = 7 + 1 = 8$: $\underline{m = 8}$ **(0.75p)**

Indeed, for a polynomial of degree 7, the eighth derivative is zero, so $E = 0$; while for a polynomial of degree 8, the eighth derivative is a non-zero constant, and $E \neq 0$, in accordance with the polynomial degree being $N = 7$.

Therefore $\underline{r = 9}$ **(0.25p)**
because the power of h in the error term of the simple rule is always one unit higher than m (r is, by definition, the order of precision or order of convergence of the simple rule: $r = OCS = 9$).

The other Newton-Cotes formulas with the same order of precision and polynomial degree of exactitude are:

- The closed rule of 7 nodes;
- The closed rule of 8 nodes (with no "extra unit" of N or OC);
- The open rule of 8 nodes (with no "extra unit" of N or OC).

The rule being open or closed has no influence on the polynomial degree N and on the order of convergence of the simple rule (with $OCS = N + 2$ always); but closed rules are more stable because they tend to have fewer negative weights. And, for the same polynomial degree / order of convergence, the rules with an odd number of nodes are more "efficient" because the computational cost is less (one evaluation of f less). Hence, out of the four rules mentioned, <u>I would use the Newton-Cotes closed rule of 7 nodes</u>. **(1p)**

b) Divide $[a,b]$ into M equally-wide subintervals $[a_i,b_i]$ ($i = 1,\ldots,M$) and apply the simple rule to each one of them. The error E_C of the compound rule will be the algebraic sum of the errors of the simple rules $E_{S,i}$ applied in all the subintervals (i.e. sum with the errors' signs, because errors in excess and in defect may cancel out):

$$E_C = \sum_{i=1}^{M} E_{S,i} = \sum_{i=1}^{M} K\,h^9 f^{8)}(\xi_i) = K\,h^9 \sum_{i=1}^{M} f^{8)}(\xi_i)$$

for some values $\xi_i \in [a_i,b_i]$. Multiplying and dividing by the number M of subintervals:

$$E_C = K\,h^9 M \frac{\sum_{i=1}^{M} f^{8)}(\xi_i)}{M} = K\,h^9 M\,\overline{f^{8)}}$$

where $\overline{f^{8)}}$ is the arithmetic mean of the M values $f^{8)}(\xi_i)$, each one in one subinterval. As such, this arithmetic mean must be some intermediate value between the minimum and the maximum values $f^{8)}(\xi_i)$ (which always exist because their number M is finite). Assuming that $f^{8)}$ is continuous in $[a,b]$, by virtue of Bolzano's (or Weierstrass's) Intermediate Value Theorem, there must be at least one ξ between the nodes where $f^{8)}(\xi)=\overline{f^{8)}}$. Substituting:

$$E_C = K\,h^9 M\,\frac{\sum_{i=1}^{M} f^{8)}(\xi_i)}{M} = K\,h^9 M\, f^{8)}(\xi) \quad \text{for some } \xi\in[a,b]$$

Now, each subinterval is of width $8h$, where h is the distance between nodes in each simple rule, so $b-a=M\cdot 8h \;\Rightarrow\; M=(b-a)/(8h)$, which we also substitute:

$$E_C = K\,h^9\,\frac{b-a}{8h} f^{8)}(\xi) = \boxed{E_C = \frac{K(b-a)h^8}{8} f^{8)}(\xi) \quad \text{for some } \xi\in[a,b]} \qquad \textbf{(2.5p)}$$

which is the expression sought. It is customary to write it in terms of $(b-a)$, instead of M, because the former is fixed for a given integral, which often allows one to know a priori what distance between nodes h, or number of subintervals M, suffice to guarantee that the total error E_C does not exceed a predefined tolerance. (This can be done when one can find bounds of $f^{8)}(x)$ in $[a,b]$.)

In the case of the open 7-node Newton-Cotes rule, the constant in the error term of the simple rule turns out to be $K=3956/14175$, so the one in the one of the compound one is $K/8=989/28350$.

Exercise 5

I will first check that the solution provided is correct (although, in a real exam situation, one would be forgiven to trust):

$$y(t)=t^{1/2}\;;\quad y'(t)=\frac{1}{2}t^{-1/2}\;;\quad y''(t)=\frac{-1}{4}t^{-3/2}$$

$$f(t,y)=-\frac{1}{4ty}=-\frac{1}{4\,t\,t^{1/2}}=\frac{-1}{4}t^{-3/2}=y''(t) \qquad \text{ODE ok}$$

$$y(1)=1^{1/2}=1\,;\quad y'(1)=\frac{1}{2}1^{-1/2}=0.5 \qquad \text{Initial Conditions ok}$$

Good.

To go from $t_0=1$ to $t_f=2$ (where $y(t)=\sqrt{2}$ is the value we are seeking) we need to take two steps of size $h=0.5$. To write the advance formula of Heun's method one can first write that of the Trapezoidal's and then substitute the y_{k+1} on the right-hand side (which gives the Trapezoidal Method its implicit character) by the value of y_{k+1} given by Euler's method (thus obtaining an explicit method that can be proved to maintain the Trapezoidal's second order of convergence):

$$\text{Trapezoidal:}\qquad y_{k+1}=y_k+\frac{f(t_k,y_k)+f(t_{k+1},y_{k+1})}{2}h_k$$

$$\text{Heun's:}\qquad y_{k+1}=y_k+\frac{f(t_k,y_k)+f(t_{k+1},y_k+f(t_k,y_k)h_k)}{2}h_k$$

We can write this in terms of the k_i constants typical of the Runge-Kutta methods — remember that Heun's method is a Runge-Kutta method of order 2— as follows. I will also use bold typeface for vectors, for the case of a system of ODEs:

$$\begin{matrix} \mathbf{k}_1 = \mathbf{f}(t_k, \mathbf{y}_k)h_k \\ \mathbf{k}_2 = \mathbf{f}(t_k + h_k, \mathbf{y}_k + \mathbf{k}_1)h_k \end{matrix} \; ; \quad \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{\mathbf{k}_1 + \mathbf{k}_2}{2} \qquad \textbf{(1.5p)}$$

(It is also possible to define $\mathbf{k}_1$, $\mathbf{k}_2$ without h_k in them and multiply later.)

Finally, I will transform the second-order ODE into a system of two ODEs of order 1:

$$y_1 = y \; ; \quad y_2 = y' \; ; \quad \begin{cases} y_1' = y_2 \\ y_2' = -\dfrac{1}{4ty_1} \end{cases} \equiv \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} y_2 \\ -\dfrac{1}{4ty_1} \end{pmatrix} = \mathbf{y}' = \mathbf{f}(t, \mathbf{y})$$

$$\mathbf{y}_0 = \begin{pmatrix} y_1(t_0) \\ y_2(t_0) \end{pmatrix} = \begin{pmatrix} y(1) \\ y'(1) \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} \qquad \textbf{(1.5p)}$$

Note that the system of ODEs is not linear (y_1 is dividing, not multiplying a function of t), so it cannot be written in matrix form as $\mathbf{y}' = J(t)\,\mathbf{y} + \mathbf{g}(t)$.

The rest is operations. I will indicate them, and give results, with Octave:

```
clear
format long g
f = @(t, y)  [y(2);  -1 / (4 * t * y(1))];
y0 = [1; 0.5];  h = 0.5;
t0 = 1;  t1 = 1.5;  t2 = 2;
% First step:
k1 = f(t0,     y0     ) * h
    k1 =  0.25
         -0.125
k2 = f(t0 + h, y0 + k1) * h
    k2 =  0.1875
         -0.0666666666666667
y1 = y0 + (k1 + k2) / 2
    y1 =  1.21875
          0.404166666666667
% Scond step:
k1 = f(t1,     y1     ) * h
    k1 =  0.202083333333333
         -0.0683760683760684
k2 = f(t1 + h, y1 + k1) * h
    k2 =  0.167895299145299
         -0.0439882697947214
y2 = y1 + (k1 + k2) / 2
    y2 =  1.403739316239316
          0.347984497581272
```

Let me also check this with the function **`anm_ode`**, available from the usual repository:

```
[tt, yy] = anm_ode(f, [1 2], y0, 2, 'Heun')
    tt =  1     1.5                2
    yy =  1     1.21875            1.403739316239316
          0.5   0.404166666666667  0.347984497581272
```

Looks good.

The only value we are interested in is (rounded to 6 significant digits):

$$y(2) = y_1(2) = \boxed{1.40374 \simeq \sqrt{2}} \qquad \textbf{(1.5p)}$$

A more precise value of $\sqrt{2}$ is:

```
sqrt(2)
    ans = 1.414213562373095
```

so the value 1.40374 is not so bad considering the large step size used ($h = 0.5$).

Heun's method is of order 2, so the error with respect to the exact value $\sqrt{2}$ should tend to zero as a $O(h^2)$. For curiosity, if we execute it with $h = 0.001$:

```
[tt, yy] = anm_ode(f, [1 2], y0, 0.001, 'Heun');
yy(1, end)
    ans = 1.414213520210385
```

which is better (error on the order of 4e-8).

Exercise 6

This is the general process to follow in order to check for consistency, stability, convergence and order of convergence of linear multistep methods:

1.- Identify the coefficients α_j, β_j of the linear multistep method's advance formula with those of the general form $y_{n+k} = -\sum_{j=0}^{k-1} \alpha_j y_{n+j} + h \sum_{j=0}^{k} \beta_j f_{n+j}$ (and $\alpha_k = 1$).

2.- Write its first and second characteristic polynomials:

$$\rho(z) = \sum_{j=0}^{k} \alpha_j z^j \;; \qquad \sigma(z) = \sum_{j=0}^{k} \beta_j z^j \;.$$

3.- The method is *consistent* ($L_i / h \to 0$) iff $\underline{\rho(1) = 0, \;\; \rho'(1) = \sigma(1)}$.

4.- The method is *stable* ($|AF| < 1$) iff the roots z_i of $\rho(z)$ verify $\underline{\|z_i\| \leq 1}$ on the complex plane, and the ones with modulus 1 are simple roots.

5.- The method is *convergent* iff it is *consistent* and *stable*, with order of convergence p iff:

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

In our case:

1.- The method is of 2 steps, so $\underline{k = 2}$; and explicit, so $f_{n+k} = f_{n+2}$ does not appear on the right-hand side, so $\underline{\beta_2 = 0}$. Therefore the form of the advance formula is:

$$y_{n+2} = -\alpha_0 y_n - \alpha_1 y_{n+1} + h\left(\beta_0 f_n + \beta_1 f_{n+1} + 0\right) \quad (\text{and } \underline{\alpha_2 = 1}).$$

2.- Characteristic polynomials: $\rho(z) = \alpha_0 + \alpha_1 z + z^2$; $\sigma(z) = \beta_0 + \beta_1 z$.

In this case the sum of the roots of ρ must be zero, and so of σ:

$$\rho(z) = \alpha_0 + \alpha_1 z + z^2 = 0 \;\Rightarrow\; z_{1,2} = \frac{-\alpha_1 \pm \sqrt{\alpha_1^2 - 4\alpha_0}}{2}\;;$$

$$z_1 + z_2 = \frac{-\alpha_1 + \sqrt{\alpha_1^2 - 4\alpha_0}}{2} + \frac{-\alpha_1 - \sqrt{\alpha_1^2 - 4\alpha_0}}{2} = -\alpha_1 = 0 \;\Rightarrow\; \underline{\alpha_1 = 0}$$

$$\sigma(z) = \beta_0 + \beta_1 z = 0 \;\Rightarrow\; z = -\beta_0/\beta_1\;;\;\; -\beta_0/\beta_1 = 0 \;\Rightarrow\; \underline{\beta_0 = 0}$$

Hence $\rho(z) = \alpha_0 + z^2$; $\sigma(z) = \beta_1 z$

3.- Consistency: $\rho(1) = 0$: $\alpha_0 + 1^2 = 0 \;\Rightarrow\; \underline{\alpha_0 = -1}$

$$\rho'(1) = \sigma(1):\;\; [2z]_{z=1} = 2 = \beta_1 \cdot 1 \;\Rightarrow\; \underline{\beta_1 = 2}$$

We now have all the parameters:

$$(\alpha_0,\alpha_1,\alpha_2)=(-1,0,1)\,; \qquad \rho(z)=-1+z^2\,; \qquad (\beta_0,\beta_1,\beta_2)=(0,2,0)\,; \qquad \sigma(z)=2z$$

4.- Stability: the roots of $\rho(z)=-1+z^2$ are $z_{1,2}=\pm 1$, both with modulus $1 \le 1$ and simple (two different roots of a polynomial of degree 2).

5.- Order of conv.:

$$m=1:\ \frac{1}{1}\left(-1\cdot 0^1+1\cdot 2^1\right)=2\cdot 1^0\,;\quad 2=2\quad \text{ok}$$

$$m=2:\ \frac{1}{2}\left(-1\cdot 0^2+1\cdot 2^2\right)=2\cdot 1^1\,;\quad 2=2\quad \text{ok}$$

$$m=3:\ \frac{1}{3}\left(-1\cdot 0^2+1\cdot 2^3\right)=2\cdot 1^2\,;\quad \frac{8}{3}\neq 2\quad \text{ko}$$

Therefore <u>the order of convergence is $p = 2$</u>.

Substituting parameters, the advance formula reads:

$$\boxed{y_{n+2}=y_n+2hf_{n+1}}$$ **(4p)**

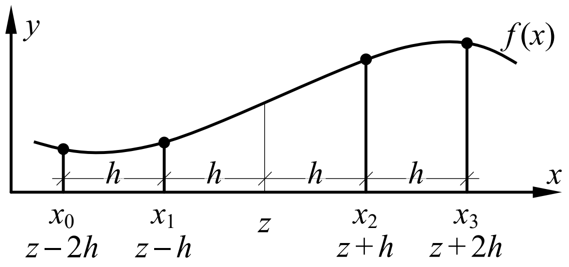
or alternatively $y_{n+1}=y_{n-1}+2hf_n$. This makes sense: the slope for the whole two-steps, of size $2h$, is taken to be the value of $f(t,y)$ at its midpoint (t_n,y_n), which is the latest calculated point at any given moment. This method, sometimes called *BF (Backward-Forward) Midpoint method*, can also be called *2-step Midpoint method.*

Exercise 7

a) For the formula to be of the highest possible order, we will locate the four nodes symmetrically on both sides of a central point z (case $\Pi'(z)=0$ in the theory—z cannot be one of the nodes (case $\Pi(z)=0$) because then the order would be one unit less). This also lets me define a formula applicable in section b), where all 5 data points in the table are equally-spaced, by distributing the nodes as follows:

$$x_0=z-2h,\quad x_1=z-h,\quad x_2=z+h,\quad x_3=z+2h$$

The figure represents the nodal positions:



Calculation of A_i by differentiating Lagrange base functions ($A_i = L_i'(z)$):

$$L_0(x)=\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}$$

$$L_0'(x)=\frac{(x-x_1)(x-x_2)\cdot 1+(x-x_1)\cdot 1\cdot(x-x_3)+1\cdot(x-x_2)(x-x_3)}{(-h)(-3h)(-4h)}$$

$$L_0'(z)=\frac{(h)(-h)+(h)(-2h)+(-h)(-2h)}{-12h^3}=\frac{1}{h}\,\frac{-1-2+2}{-12}=\underline{\frac{1}{12h}=A_0}$$

$$L_1(x)=\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}$$

$$L_1'(x)=\frac{(x-x_0)(x-x_2)\cdot 1+(x-x_0)\cdot 1\cdot(x-x_3)+1\cdot(x-x_2)(x-x_3)}{(h)(-2h)(-3h)}$$

$$L_1'(z)=\frac{(2h)(-h)+(2h)(-2h)+(-h)(-2h)}{6h^3}=\frac{1}{h}\frac{-2-4+2}{6}=\underline{\frac{-2}{3h}=A_1}$$

$$L_2(x)=\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}$$

$$L_2'(x)=\frac{(x-x_0)(x-x_1)\cdot 1+(x-x_0)\cdot 1\cdot(x-x_3)+1\cdot(x-x_1)(x-x_3)}{(3h)(2h)(-h)}$$

$$L_2'(z)=\frac{(2h)(h)+(2h)(-2h)+(h)(-2h)}{-6h^3}=\frac{1}{h}\frac{2-4-2}{-6}=\underline{\frac{2}{3h}=A_2}$$

$$L_3(x)=\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

$$L_3'(x)=\frac{(x-x_0)(x-x_1)\cdot 1+(x-x_0)\cdot 1\cdot(x-x_2)+1\cdot(x-x_1)(x-x_2)}{(4h)(3h)(h)}$$

$$L_3'(z)=\frac{(2h)(h)+(2h)(-h)+(h)(-h)}{12h^3}=\frac{1}{h}\frac{2-2-1}{12}=\underline{\frac{-1}{12h}=A_3}$$

Substituting: $f'(z)=D+E=A_0f(x_0)+A_1f(x_1)+A_2f(x_2)+A_3f(x_3)+E=$

$$=\frac{1}{12h}f(z-2h)+\frac{-2}{3h}f(z-h)+\frac{2}{3h}f(z+h)+\frac{-1}{12h}f(z+2h)+E=$$

$$=\boxed{f'(z)=\frac{f(z-2h)-8f(z-h)+8f(z+h)-f(z+2h)}{12h}+E}\qquad\textbf{(2.5p)}$$

The truncation error *E* is the derivative of the error of the interpolation polynomial at *z*:

$$E=f'(z)-D=f'(z)-p_3'(z)=\left(f(z)-p_3(z)\right)'=e'(z)=\left(f[x_0,x_1,x_2,x_3,z]\Pi(z)\right)'=$$
$$=f[x_0,x_1,x_2,x_3,z]'\Pi(z)+f[x_0,x_1,x_2,x_3,z]\Pi'(z)$$

The last term is 0 by symmetry (four nodes symmetrically located on both sides of *z* make $\Pi'(z)=0$); hence:

$$E=f[x_0,x_1,x_2,x_3,z]'\Pi(z)\underset{\text{Theorem}}{=}1!f[x_0,x_1,x_2,x_3,z,z]\Pi(z)\underset{\text{Chapter 1}}{=}\frac{f^{5)}(\xi)}{5!}\Pi(z)=$$

$$=\frac{f^{5)}(\xi)}{5!}(z-x_0)(z-x_1)(z-x_2)(z-x_3)=\frac{f^{5)}(\xi)}{5!}(2h)(h)(-h)(-2h)=\boxed{E=\frac{f^{5)}(\xi)}{30}h^4}\qquad\textbf{(2p)}$$

b) This is mostly operations using the formula obtained in section a). With Octave:

```
clear
format long g
f0 = 3.10;   f1 = 3.12;   fz = 3.14;  f2 = 3.18;  f3 = 3.24;   h = 0.01;
A0 = 1/12/h; A1 = -2/3/h;             A2 = 2/3/h; A3 = -1/12/h;
ipz = A0 * f0 + A1 * f1 + A2 * f2 + A3 * f3       % i'(z) (approx.)
    ipz =      2.83333333333335
vz  = 0.98 * ipz + 0.142 * fz                     % v(z)  (approx.)
    vz =     3.22254666666668
```

So, rounded to 6 significant digits: $\boxed{v(1.02) \approx 3.22255}$ **(1p)**