APPENDIX

Solutions

Section 1.5

Section 2.3

Section 3.1

Section 3.2

Exercise 7

(a) In general we have that $\beta = 10_\beta$ for any β. In particular we have that $7 = 10_7$, $37 = 10_{37}$, $4 = 10_4$.

(b) The equation $13 = 10_\beta$ gives that $\beta = 13$ from (a). The equation $100 = 10_\beta$ gives in the same way that $\beta = 100$. For all $a \in \mathbb{N}$ we can find a β which solves $a = 10_\beta$: It is enough to set $\beta = a$.

Section 3.3

Exercise 5

(a) $0.b_\beta$ is always a number in $[0,1)$, so that such a β exists only for $a < 1$. Since $0.b_\beta = \frac{b}{\beta} = a = \frac{b}{c}$ it follows that $\beta = c$, so that we can find a unique β for all rational a on the form $a = 0.b_\beta$.

(b) Since $0.01_\beta = \frac{1}{\beta^2}$, if $a = \frac{b}{c} = 0.01_\beta$ we must have that $c = b\beta^2$. In other words, a rational number $\frac{b}{c}$ can be written on the form 0.01_β if and only if $c = b\beta^2$.

(c) Since $0.0b_\beta = \frac{b}{\beta^2}$, if $a = \frac{c}{d} = 0.0b_\beta$ we must have that $c = \frac{bd}{\beta^2}$. In other words, a rational number $\frac{c}{d}$ can be written on the form $0.0b_\beta$ if and only if $c = \frac{bd}{\beta^2}$.

Exercise 6 when we in algorithm 3.20 obtain a b which has been seen before, we will perform the same computations again, so that the sequence will repeat. There are c possibilities for this value since we compute the remainder with c. The longest possible repeating sequence is thus one where all values of b are observed. However, the value 0 will result in the rest of the digits being 0, so the maximum length repeating sequence is obtained when the values $1, ..., c-1$ are observed for b in succession. This results in a repeating sequence of length $c-1$.

Section 3.4

Exercise 1

(a) In the equation $7_\beta + 8_\beta = 13_\beta$, the left hand side is $7+8=15$. The right hand side is $13_\beta = \beta + 3$. Solving $\beta + 3 = 15$ we get that $\beta = 12$, so that the third alternative is correct.

(c) We can write $40.125 = 5 \cdot 8 + 1/8 = 5 \cdot 8^1 + 0 \cdot 8^0 + 8^- 1$. This can also be written as 50.1_8, so that the last alternative is the correct one.

Section 4.1

Exercise 1

(c) The machine enters an infinite loop, since Python increases the precision used for numbers when the sum of two numbers is beyond the current limit. The machine will eventually run out of memory when too much recourses are required to represent the number, but this may take some billions of years. This means that the first alternative is correct.

Exercise 3 There is no way we can constuct a unique "threes's complement" in. One possibility is, in fact 4.3, to replace the representation of a negative number with the n first digits in $3^n - |x|$, where n still denotes the number of digits in x. With addition defined by neglecting digit $n+1$ as in two's complement, this representation of numbers will have the same properties as two's compliment. The only difference is that the numbers now represented only will cover two thirds of the numbers with $n+1$ digits: The lower third (which represents positive numbers), and the upper third (which represents negative numbers).

Section 4.2

Section 4.3

Exercise 3

(a) Since $5a_{16} \leq 7f_{16}$, this is encoded with one byte, so that the UTF-8 encoding is the number itself, i.e. $5a_{16}$.

(b) We have that $80_{16} \leq f5_{16} \leq 7ff_{16}$, so that two bytes are used in the UTF8-encoding. Since $f5_{16} = 1111\ 0101_2$, we have that the UTF8-encoding is $1100\ 0011\ 1011\ 0101_2 = c3b5_{16}$.

(c) We have that $80_{16} \leq 3f8_{16} \leq 7ff_{16}$, so that two bytes are used in the UTF8-encoding also here. Since $3f8_{16} = 11\ 1111\ 1000_2$, we have that the UTF8-encoding is $1100\ 1111\ 1011\ 1000_2 = cfb8_{16}$.

(d) We have that $800_{16} \leq 8f37_{16} \leq ffff_{16}$, so that three bytes are used in the UTF8-encoding also here. Since $8f37_{16} = 1000\ 1111\ 0011\ 0111_2$, we have that the UTF8-encoding is $1110\ 1000\ 1011\ 1100\ 1011\ 0111_2 = e8bcb7_{16}$.

Exercise 5

(a) These characters have code points $e6_{16} = 1110\ 0110_2$, $f8_{16} = 1111\ 1000_2$, and $e5_{16} = 1110\ 0101_2$. All of them are stored with 2 bytes in UTF-8 The UTF8-encoding of 'æ' is $1100\ 0011\ 1010\ 0110_2 = c3a6_{16}$, which corresponds to the two characters Ã¦ in the ISO Latin1 character set. The UTF8-encoding of 'ø' is $1100\ 0011\ 1011\ 1000_2 = c3b8_{16}$, which corresponds to the two characters Ã¸ in the ISO Latin1 character set. The UTF8-encoding of 'æ' is $1100\ 0011\ 1010\ 0101_2 = c3a5_{16}$, which corresponds to the two characters Ã¥ in the ISO Latin1 character set.

(b) None of the three codepoints for 'æ', 'ø', 'å', are seen to be valid UTF8-codes.

(c) All three characters are also stored with 2 bytes in UTF-16, and as $00e6_{16}$, $00f8_{16}$, and $00e5_{16}$, respectively. These are shown as æ, ø, and å, respectively. The other way, the ISO Latin1 encoding of each of the three characters is too short to be accepted as a UTF-16 encoding.

(d) Assume that the characters are stored with UTF-8. As shown in (a), the UTF8-characters are $c3a6_{16}$, $c3b8_{16}$, and $= c3a5_{16}$, which are the valid two-byte Unicode characters Ãa6, Ãb8, and Ãa5.

Conversely, assume that the characters are stored with UTF-16. The first byte in the code $1110\ 0110_2$ for 'æ', indicates that it shoould be stored with 3 bytes, but then the second byte should start with 10, which it does not. The same applies for 'å'. Finally, for the code $1111\ 1000_2$ for ø, there are no bytes in UTF-8 which start with 11111, so that all characters are invalid UTF-8 characters.

Exercise 8 We have that

$$41\ 42\ 43\ 44\ 45_{16} = 0100\ 0001\ 0100\ 0010\ 0100\ 0011\ 0100\ 0100\ 0100\ 0101_2$$

This is clearly a valid UTF-8 encoding of 5 ASCII characters. Since 5 bytes are used, it can not be a UTF-16 code, since that would require an even number of bytes being used.

Exercise 10 We have that

$$41\ C3\ 98\ 41\ C3\ 41\ 41\ C3\ 98\ 98\ 41_{16} =$$

$$0100\ 0001\ 1100\ 0011\ 1001\ 1000\ 0100\ 0001\ 1100\ 0011\ 0100\ 0001\ 0100\ 0001\ 1100\ 0011\ 1001\ 1000\ 1001\ 1000\ 0100\ 0001_2.$$ The first byte, $0100\ 0001_2$, is a valid UTF-8 character. The next two bytes, $1100\ 0011\ 1001\ 1000_2$, is a valid two-byte UTF-8 character. The next byte, 0100 0001, is again a valid UTF-8 character. the next two bytes, $1100\ 0011\ 0100\ 0001_2$ are not valid, since the second byte should start with 10 when the first starts with 110.

Section 4.5

Section 5.2

Exercise 2 The last expression may give large relative error when calculated on a machine using floating point arithmetic, since it is possible for $\sin(-x^2)$ to be close to 1/2 (cancellation). The other expressions can not give cancellation.

Exercise 6

(b) The biggest number of 9.834 and 2.45 is 9.834, and this can be written on normalised form as 0.9834×10^1. The other number can be written as 0.2450×10^1 when we use the same exponent (we added a 0 to get 4 digits). We add the significands and get $0.9834 + 0.2450 = 1.2284$. At the end we convert 1.2284×10^1 to normalized form and get 0.1228×10^2, where we at the end had to do a rounding since the significand should be represented by 4 digits only.

Exercise 7

(a) A normalised number in base β is represented as $\alpha \times \beta^n$ where n is a one digit number, and α is a number between β^{-1} and 1 represented with a four digit number in base β ($0.1 = 10^{-1}$ was exchanged with β^{-1}).

(b) In any numeral system we have three cases to consider when defining rounding rules. Note also that it is sufficient to define rounding for two-digit fractional numbers.

In the octal numeral system the three rules are:

1. A number $(0.d_1 d_2)_8$ is rounded to $0.d_1$ if the digit d_2 is 0, 1, 2 or 3.
2. If $d_1 < 7$ and d_2 is 4, 5, 6, or 7, then $(0.d_1 d_2)_8$ is rounded to $0.\tilde{d}_1$ where $\tilde{d}_1 = d_1 + 1$.
3. A number $(0.7d_2)_8$ is rounded to 1.0 if d_2 is 4, 5, 6, or 7.

In the hexadecimal numeral system the rules are similar: If the last digit is one of 8, 9, a,b,c,d,e, or f, round up, otherwise round down.

Exercise 9 Code in Python is

```
x=0.0
while x<= 2.0:
  print x
  x=x+0.1
```

In my program the last value 2.0 is not written. The explanation is that 0.1 can not be represented exactly by the computer. What actually is the case here is that the machine represents 0.1 with a number slightly bigger than 0.1. When this number is added with itself 20 times, we have a number which is bigger than 2.0, so the number is not printed. 2.0 can, however, be represented exactly by the computer.

Section 5.3

Exercise 2

(a) The absolute error is $|a-\tilde{a}| = |1-0.9994| = 0.0006$. The relative error is $\frac{|a-\tilde{a}|}{|a|} = \frac{0.0006}{1} = 0.0006$. The relative error can also be written as $0.6 \times 10^{-3} \approx 10^{-3}$, and observation 5.20 says that about the 3 most significant digits in a and $\tilde{a}$ should agree. This is not so far from the truth in this case, since 0.999 (where we have included the three most significant digits) will be rounded to 1.000.

(b) The absolute error is $|a-\tilde{a}| = |24-23.56| = 0.44$. The relative error is $\frac{|a-\tilde{a}|}{|a|} = \frac{0.44}{24} = 0.01833$. The relative error can also be written as $1.8 \times 10^{-2} \approx 10^{-2}$, and observation 5.20 says that about the 2 most significant digits in a and $\tilde{a}$ should agree. This is in fact the case, since 23.56 with the two most significant digits gives 24.

(c) The absolute error is $|a-\tilde{a}| = |-1267+1267.345| = 0.345$. The relative error is $\frac{|a-\tilde{a}|}{|a|} = \frac{0.345}{1267} = 0.000272$. The relative error can also be written as $2.7\times 10^{-4} \approx 10^{-4}$, and observation 5.20 says that about the 4 most significant digits in a and $\tilde{a}$ should agree. This is in fact the case, since -1267.345 with the two most significant digits gives -1267.

(d) The absolute error is $|a-\tilde{a}| = |124-7| = 117$. The relative error is $\frac{|a-\tilde{a}|}{|a|} = \frac{117}{124} = 0.9435$. The relative error can also be written as $0.94\times 10^{0} \approx 10^{0}$, and observation 5.20 says that no significant digits in a and $\tilde{a}$ should agree. This is the case here.

Exercise 3 (a). The absolute error the other way is $\frac{|\tilde{a}-a|}{|\tilde{a}|} = \frac{0.0006}{0.9994} = 0.0006$, so that we have approximately the same relative error. This agrees with the sentence in section 5.3.3, which says that the two relative errors should be quite close, in cases where the two relative errors are quite small..

(b). The absolute error the other way is $\frac{|\tilde{a}-a|}{|\tilde{a}|} = \frac{0.44}{23.56} = 0.0187$, so that we have approximately the same relative error. This agrees with the sentence in section 5.3.3, as for a).

(c). The absolute error the other way is $\frac{|\tilde{a}-a|}{|\tilde{a}|} = \frac{0.345}{1267.345} = 0.000272$, so that we have approximately the same relative error. This agrees with the sentence in section 5.3.3.

(d). The absolute error the other way is $\frac{|\tilde{a}-a|}{|\tilde{a}|} = \frac{117}{7} \approx 16.7$, so that we here have relative errors quite far apart. This has to do with that the relative errors are quite big,so that the observation is not valid.

Exercise 4 In example 5.9 we found the approximation $\tilde{a} = 0.1247\times 10^{2} = 13.47$ to the sum $5.645+7.821 = 13.466$. The relative error is

$$\frac{|a-\tilde{a}|}{|a|} = \frac{0.004}{13.466} \approx 0.000297 = 2.97\times 10^{-4} \approx 10^{-4}.$$

From the observation the numbers should agree in the four most significant digits. This is the case since the four most significant digits in 13.466 give 13.47.

In example 5.10 we found the approximation $0.4234\times 10^{2} = 42.34$ to the sum $42.34+0.0033 = 42.3433$. The relative error is

$$\frac{|a-\tilde{a}|}{|a|} = \frac{0.0033}{42.3433} \approx 0.000078 = 0.78\times 10^{-4} \approx 10^{-4}.$$

From the observation the numbers should agree in the four most significant digits. This is the case since the four most significant digits in 42.3433 give 42.34.

In example 5.11 we found the approximation $0.7 \times 10^{-1} = 0.07$ to the difference $10.34 - 10.27 = 0.07$. The relative error here is 0, and all significant digits should agree, which they do since the numbers are equal.

In example 5.12 we found the approximation $0.9000 \times 10^{-3} = 0.0009$ to the difference $10/7 - 1.42 \approx 0.8571 \times 10^{-3} = 0.0008571$. The relative error is

$$\frac{|a - \tilde{a}|}{|a|} = \frac{0.0000429}{0.0008571} \approx 0.05 = 0.5 \times 10^{-1} \approx 10^{-1}.$$

From the observation the numbers should agree in just the most significant digit. This is the case since 0.0008571 with only the most significant digit is 0.0009.

Section 5.4

Exercise 1

(a) We can write

$$\frac{5-\sqrt{5}}{5+\sqrt{5}} + \frac{\sqrt{5}}{2} = \frac{(5-\sqrt{5})(5-\sqrt{5})}{5^2-5} + \frac{\sqrt{5}}{2} = \frac{25-10\sqrt{5}+5}{20} + \frac{\sqrt{5}}{2} = \frac{3}{2}.$$

Therefore, the last alternative is the correct one.

(b) The last alternative is the correct one. The reason is that several bits are allocated for the exponent, see Fact 4.8. The first alternative is wrong, positive numbers can also give roundoff errors. The second alternative is wrong because we typically have limitations on the computer in the significand and the exponent, and even if the types can expand the bits used for these as in Python, the computer has at the end a limited amount of memory. The third alternative is wrong, since 64-bit integers represents numbers in the interval $[-2^{63}, 2^{63}-1]$ (Fact 4.2).

Exercise 2

(a) If x is very large we will in the expression $\sqrt{x+1} - \sqrt{x}$ subtract two large numbers very close to each other. According to observation 5.13 we then can loose precision with many digits (i.e. cancellation). If we multiply with $\sqrt{x+1} + \sqrt{x}$ up and down we get $\frac{1}{\sqrt{x+1}+\sqrt{x}}$, where we have avoided the problem of subtracting two numbers close to each other.

(b) The fomula $\ln x^2 - \ln(x^2 + x)$ is problematic for large values of x since then the two logarithms will become almost equal and we get cancellation. Using properties of the logarithm, the expression can be rewritten as

$$\ln x^2 - \ln(x^2 + x) = \ln\left(\frac{x^2}{x^2 + x}\right) = \ln\left(\frac{x}{x+1}\right)$$

which will not cause problems with cancellation.

(c) We can write $\cos^2 x - \sin^2 x = \cos(2x)$,so that we avoid subtraction of two almost equal numbers near $x = \frac{\pi}{4}$.

Section 6.1

Section 6.2

Exercise 2

(a) We have that $x_{n+2} = f(n, x_n, x_{n+1}) = 3x_{n+1} - x_n$. We compute

$$\begin{aligned}
x_2 &= 3x_1 - x_0 = 3 - 2 = 1 \\
x_3 &= 3x_2 - x_1 = 3 - 1 = 2 \\
x_4 &= 3x_3 - x_2 = 3 \times 2 - 1 = 5 \\
x_5 &= 3x_4 - x_3 = 3 \times 5 - 2 = 13.
\end{aligned}$$

(d) We have that $x_{n+1} = f(n, x_n) = -\sqrt{4 - x_n}$. We compute

$$\begin{aligned}
x_1 &= -\sqrt{4 - x_0} = -\sqrt{4 - 0} = -2 \\
x_2 &= -\sqrt{4 - x_1} = -\sqrt{4 + 2} = -\sqrt{6} \approx -2.4495 \\
x_3 &= -\sqrt{4 - x_2} = -\sqrt{4 + 2.4495} \approx -2.5396 \\
x_4 &= -\sqrt{4 - x_3} = -\sqrt{4 + 2.5396} \approx -2.5573 \\
x_5 &= -\sqrt{4 - x_4} = -\sqrt{4 + 2.5573} \approx -2.5607.
\end{aligned}$$

(e) We have that $x_{n+2} = f(n, x_n, x_{n+1}) = \frac{1}{5}(3x_{n+1} - x_n + n)$. We compute

$$\begin{aligned}
x_2 &= \frac{1}{5}(3 - 0 + 0) = \frac{3}{5} \\
x_3 &= \frac{1}{5}(\frac{9}{5} - 1 + 1) = \frac{9}{25} \\
x_4 &= \frac{1}{5}(\frac{27}{25} - \frac{3}{5} + 2) = \frac{62}{125} \\
x_5 &= \frac{1}{5}(\frac{186}{125} - \frac{9}{25} + 3) = \frac{816}{625}.
\end{aligned}$$

(f) If we insert $x_0 = 3$ we get that $x_1^2 = -15 + 1 = -14$, which clearly has no solution. Furthermore, x_{n+1} is not uniquely determined for other initial conditions, since $x_{n+1} = \pm\sqrt{1 - 5x_n}$.

Section 6.3

Exercise 2 The code can look as follows:

```
N=10
xpp=0
xp=1
for i in range(N-1):
  x=xpp+xp
  print x
  xpp=xp;
  xp=x
```

Exercise 3 The code can look as follows:

```
N=10
xppp=0
xpp=1
xp=1
for i in range(N-2):
  x=xppp+xpp+xp
  print x
  xppp=xpp
  xpp=xp;
  xp=x
```

Section 6.4

Section 6.5

Exercise 2 The code can look as follows:

```
#a
xpp=1.0
xp=2.0/3.0
for n in range(100):
  x=(-4.0*xp+4.0*xpp)/3.0
  xpp=xp
  xp=x
```

```
  print xp

#b
x=1.0
for n in range(100):
  x=(1.0+x/3.0)/3.0
  print x
```

(a) The characteristic equation is $3r^2+4r-4=0$, which has roots $r=\frac{-4\pm\sqrt{16+48}}{6}=\frac{-2\pm4}{3}$. The roots are thus -2 and $2/3$, so that the solution to the difference equation is $x_n=C(-2)^n+D(2/3)^n$. The initial values give

$$\begin{aligned} C+D&=1\\ -2C+\frac{2}{3}D&=2/3, \end{aligned}$$

which gives that $\frac{8}{3}D=\frac{8}{3}$, so that $D=1$ and $C=0$. The solution is thus $x_n=(2/3)^n$. This will go to zero, but due to roundoff in the second initial condition there will be a term on the form $\hat{\epsilon}(-2)^n$ also contributing in the simulation, so that we will will eventually get overflow. We will then get NAN, since we at the end substract $-\infty$ from ∞ or vice versa). The last alternative is thus correct.

(b) It is straightforward to check that the exact solution is $x_n=\frac{3}{8}+\frac{45}{8}\left(\frac{1}{9}\right)^n$. It is clear that simulations will converge to $3/8$.

Exercise 4

(c) The code can look as follows:

```
N=40
x=-5.0/14
print x
for n in range(N):
  x=3.0*x+5**(-n)
  print x
```

Exercise 5

(a) The characteristic equation is $r^2-r-1=0$, which has roots $r=\frac{1\pm\sqrt{1+4}}{2}=\frac{1\pm\sqrt{5}}{2}$. This means that we have two different real roots, so that the general solution is

$$x_n=C\left(\frac{1+\sqrt{5}}{2}\right)^n+D\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

The two initial values give

$$C + D = 1$$
$$C\frac{1+\sqrt{5}}{2} + D\frac{1-\sqrt{5}}{2} = \frac{1-\sqrt{5}}{2}.$$

Substituting the first in the second gives

$$C\sqrt{5} + \frac{1-\sqrt{5}}{2} = (1-\sqrt{5})/2,$$

so that $C = 0$ and $D = 1$. This gives the solution $x_n = \left(\frac{1-\sqrt{5}}{2}\right)^n$.

(b) Due to rounding in the second initial condition, the computer will simulate values of the form

$$x_n = \hat{\epsilon}\left(\frac{1+\sqrt{5}}{2}\right)^n + (1-\hat{\epsilon})\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

This means that the values of x_n eventually will overflow. In the beginning of the calculations, the values seem to converge to zero, since the term $(1-\hat{\epsilon})\left(\frac{1-\sqrt{5}}{2}\right)^n$ dominates in the beginning, and this term converges to zero.

(c) The code can look as follows:

```
from math import sqrt

xpp=1.0
xp=(1-sqrt(5.0))/2.0
for n in range(100):
  x=xpp+xp
  xpp=xp
  xp=x
  print xp
```

Exercise 6

(a) The characteristic equation is $r^2 - \frac{2}{5}r + \frac{1}{45} = 0$, which has roots

$$r = \frac{\frac{2}{5} \pm \sqrt{\frac{4}{25} - \frac{4}{45}}}{2} = \frac{\frac{2}{5} \pm \sqrt{\frac{36-20}{225}}}{2} = \frac{\frac{2}{5} \pm \frac{4}{15}}{2} = \frac{1}{5} \pm \frac{2}{15},$$

so that $r = \frac{1}{3}$ eller $r = \frac{1}{15}$. Therefore the general solution to the difference equation is $x_n = A\left(\frac{1}{15}\right)^n + B\left(\frac{1}{3}\right)^n$. The initial values $x_0 = 1$, $x_1 = \frac{1}{15}$ give

$$\begin{aligned} A + B &= 1 \\ \frac{1}{15}A + \frac{1}{3}B &= \frac{1}{15}. \end{aligned}$$

These equations can also be written as

$$\begin{aligned} A + B &= 1 \\ A + 5B &= 1. \end{aligned}$$

We quickly see that the dolution to this is $A = 1$, $B = 0$, so that the solution to the difference equation is $x_n = \left(\frac{1}{15}\right)^n = 15^{-n}$.

(b) The other initial condition cannot be represented exactly on the computer, so that the computer instead will find a solution on the form

$$\hat{x_n} = (1 - \hat{\epsilon})\left(\frac{1}{15}\right)^n + \hat{\epsilon}\left(\frac{1}{3}\right)^n,$$

where $\hat{\epsilon}$ is a small number representing the roundoff error committet by the computer. When n becomes large, the "error" $\hat{\epsilon}\left(\frac{1}{3}\right)^n$ dominates in this expression, which explains why we must expect numerical inaccuracies for large n. Note that the absolute error is not large since $\hat{\epsilon}\left(\frac{1}{3}\right)^n$ is a small number, but that the relative error is very large since, since $\hat{\epsilon}\left(\frac{1}{3}\right)^n$ is relatively much larger than $\left(\frac{1}{15}\right)^n$ for large n.

(c) $\hat{\epsilon}$ represents approximately the smallest number the machine can represent. If we use 64 bits this corresponds to $\approx 2^{-63} \approx 10^{-17}$. We have lost all significant digits when the "error" $\hat{\epsilon}\left(\frac{1}{3}\right)^n$ becomes larger that the actual solution $\left(\frac{1}{15}\right)^n$, i.e. $10^{-17}\left(\frac{1}{3}\right)^n > \left(\frac{1}{15}\right)^n$. This corresponds to $5^n > 10^{17}$, which gives $n > \frac{17\ln 10}{\ln 5} \approx 24$. The arguments given here are not exact. For example, it can be that the estimate for $\hat{\epsilon}$ is not very exact.

(d) The code can look as follows:

```
N=100
xpp=1.0
xp=1.0/15
for i in range(2,N):
  x=2.0*xp/5+xpp/45
```

```
  print i,x
  xpp=xp
  xp=x
```

Exercise 7

(c) The code can look as follows:

```
N=100
xpp=1.0
xp=0.5
for k in range(2,N):
  x=5.0*xp/2-xpp
  print k,x
  xpp=xp
  xp=x
```

Section 7.1

Section 7.2

Exercise 3

(a)

$$\begin{array}{lll} f(t)=2, & f(a)=2, & f(o)=2, \\ f(h)=2, & f(m)=1, & f(l)=2, \\ f(e)=6, & f(n)=2, & f(i)=1, \\ f(r)=3, & f(y)=1, & f(w)=1, \\ f(\sqcup)=6, & f(p)=2, & f(d)=1. \end{array}$$

(b) An example of a Huffman tree for this text can be seen in figure 16:

(c) The Huffman coding for the text "there are many people in the world" is then:

0100 0101 11 0010 11 000 0110 0010 11 000
00111 0110 0111 10110 000 1000 11 1001 1000
1010 11 000 10111 0111 000 0100 0101 11 000
001100 1001 0010 1010 001101

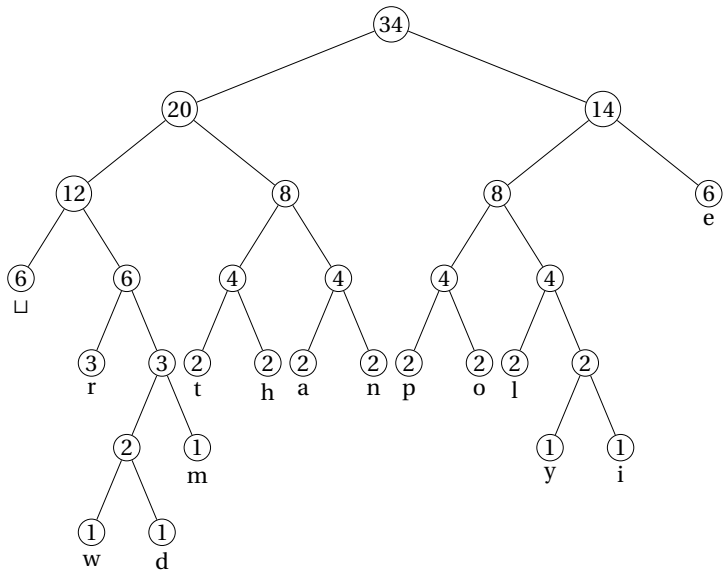

Figure 16. The Huffman tree for the text 'there are many people in the world'.

The entropy is:

$$H = 3.6325 \tag{.7}$$

which means an optimal coding of the text would use 3.6325 bits per symbol. There are 34 symbols so the minimum coding would consist of 15 bytes and 4 bits. The Huffman coding above gave 15 bytes and 5 bits of information, so this coding is very good.

Exercise 5

(a)

$$f(A) = 4,$$
$$f(B) = 2,$$
$$f(C) = 2,$$

One of the four possible Huffman codings are:

0 10 0 11 0 10 11 0

The entropy is

$$H = 1.5 \tag{.8}$$

This gives an optimal coding with 12 bits for 8 symbols, which is just what the Huffman coding gave.

(b) Dividing all the frequencies by 2 and interchanging A with C in the four trees in *a*) gives the four trees for this problem.The four sets of codes are the same (with A interchanged by C) and so is the entropy so the situation is still optimal.

Exercise 6 If we assume all letters have equal frequency, we can in the 64 first steps of constructing the Huffman tree simply combine adjacent letters into new nodes. We then have 64 nodes, each with frequency 2. We can repeat this procedure and obtain 32 new nodes with frequency 4 each, then 16 new nodes with frequency 8 each, and so on. We end up with a binary tree with 8 levels, where all nodes have two branches, and where the letters all are leaf nodes at the lowest level. The letters are then represented by all possible combination of 7 bits, since any path from the root node can be followed down to the leaf nodes.

Section 7.3

Exercise 1

(a) The first text has probabilities $p(A) = 0.5$, $p(B) = 0.5$, so that the entropy is $-0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$. The second text has probabilities $p(A) = 5/9$, $p(B) = 4/9$, so that the entropy is $-5/9\log_2(5/9) - 4/9\log_2(4/9) = 0.99108$. The statement is therefore false.

(b) We have that the probability of the symbol is $p(A) = 1$. Since $\log_2(1) = 0$, the informtaion entropy is 0, so that the statement is true.

(c) The statement is false. As an example, if we repeat a short text many times, the information entropy is unchanged.

(d) The answer to this question consists of a value of true or false. If p is the probability of true, the information entropy is $-p\log_2 p - (1-p)\log_2(1-p)$. If $p \geq 1-p$, we have that

$$-p\log_2 p-(1-p)\log_2(1-p) < -p\log_2(1-p)-(1-p)\log_2(1-p) = -\log_2(1-p) < 1.$$

Similarly when $p < 1-p$. The statement is therefore false.

Exercise 2 The information entropy for the first text is $-0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$. Clearly Huffman coding here uses 1 bit per symbol also, so this achieves a minimum number of bits per symbol.

The information entropy for the second text is $2(-0.25\log_2 0.25) - 0.5\log_2 0.5 = 1.5$. Huffman coding here uses codes of length $l(A) = l(B) = 2$, and $l(C) = 1$, so that it uses

$$p(A)l(A) + p(B)l(B) + p(C)l(C) = 0.25 \times 2 + 0.25 \times 2 + 0.5 \times 1 = 1.5$$

bits per symbol, so this achieves a minimum number of bits per symbol as well.

The information entropy for the fourth text is $-0.25\log_2 0.25 - 0.75\log_2 0.75 \approx 0.81128$. Clearly Huffman coding here uses 1 bit per symbol, so this does not achieve a minimum number of bits per symbol.

The information entropy for the fourth text is $4(-0.25\log_2 0.25) = 2$. Clearly Huffman coding here uses 2 bits per symbol also, so this achieves a minimum number of bits per symbol.

Exercise 4

(a) We have that $f(t) = 2$, $f(o) = 3$, $f(b) = 1$, $f(e) = 1$, $f(i) = 1$, $f(s) = 1$, $f(d) = 1$, and $f(\sqcup) = 4$, so that the probabilities are $p(t) = 2/14$, $p(o) = 3/14$, $p(b) = 1/14$, $p(e) = 1/14$, $p(i) = 1/14$, $p(s) = 1/14$, $p(d) = 1/14$, and $p(\sqcup) = 4/14$. The information entropy is

$$-5 \times \frac{1}{14}\log_2(1/14) - \frac{2}{14}\log_2(2/14) - \frac{3}{14}\log_2(3/14) - \frac{4}{14}\log_2(4/14) \approx 2.7534.$$

Section 7.4

Exercise 2

(a) Clearly we have that $f(A) = 9$, $f(B) = 1$, so that $p(A) = 9/10$, $p(B) = 1/10$.

(b) We have that $\lceil -\log_2(0.9^9 0.1)\rceil + 1 = 6$,

(c) The first seven A's restrict us to the interval $[0, 0.9^7]$. The next B restricts us to the interval $[0.9^8, 0.9^7]$, and the two last A's restrict us further to the interval $[0.9^8, 0.9^8 + 0.9^2(0.9^7 - 0.9^8)]$. The midpoint in this interval is $0.9^8 + 0.9^2(0.9^7 - 0.9^8)/2 \approx 0.44983823445$.

- The first bit in this is clearly 0.
- To compute the second bit we compute $2 \times 0.44983823445 \approx 0.8996764689$, so that the second bit is 1.

- We then compute $2 \times 0.8996764689 - 1 \approx 0.7999353$, so that the third bit is 1.
- We then compute $2 \times 0.7999353 - 1 \approx 0.5987058756$, so that the fourth bit is 1.
- We then compute $2 \times 0.5987058756 - 1 \approx 0.1974117512$, so that the fifth bit is 0.
- Finally we compute $2 \times 0.1974117512 \approx 0.3948235024$, so that the sixth and final bit is 0.

The arithmetic code is thus 011100 (we could here have omitted the two trailing zeros in the arithmetic code as well, since 0.0111 and 0.011100 correspond to the same number).

Exercise 3

(a) The information entropy is $-4 \times 0.25 \log_2(0.25) = 2$.

(c) Arithmetic coding requires $\lceil -\log_2(0.25^m) \rceil + 1 = 2m + 1$ bits, so that we require $\frac{2m+1}{m}$ bits per symbol. When m is large this is very close to 2.

(e) We map A to the interval $[0, 0.25]$, B to the interval $[0.25, 0.5]$, C to the interval $[0.5, 0.75]$, D to the interval $[0.75, 1]$. This means that the four intervals correspond to numbers where the first bits are 00, 01, 10, and 11, respectively. Note that these are exactly the Huffman codewords for the four letters. Clearly this also means that, when we refine the arithmetic code with a new letter, this means that we simply add the corresponding bits to the code This means that the arithmetic code is the same as the Huffman code, with the exception that we add a bit at the end. This bit represents that we restrict to the midpoint on the interval, which is achieved if we set the bit to 1 (i.e. $0.b1$ is the midpoint in $[0.b, 0.(b+1)]$).

Exercise 4 The letters A,B, and C correspond to the intervals $[0, 0.1]$, $[0.1, 0.7]$, and $[0.7, 1]$, respectively. The number 1001101 corresponds to $2^{-1} + 2^{-4} + 2^{-5} + 2^{-7} = 0.6015625$. This lies in the second interval, so that the first letter is B.

- We now apply the mapping $h_2(0.6015625) = (0.6015625 - 0.1)/(0.7 - 0.1) = 0.8359375$, and this value lies in the third interval, so that the second letter is C.

- We now apply the mapping $h_3(0.8359375) = (0.6015625 - 0.7)/(1 - 0.7) = 0.453125$, and this value lies in the second interval, so that the third letter is B.
- We now apply the mapping $h_2(0.453125) = (0.453125 - 0.1)/(0.7 - 0.1) = 0.58854166666666$, and this value lies in the second interval, so that the forth letter is B.
- We now apply the mapping $h_2(0.58854166666666) = (0.58854166666666 - 0.1)/(0.7 - 0.1) = 0.81423611111111$, and this value lies in the third interval, so that the fifth letter is C.
- We now apply the mapping $h_3(0.81423611111111) = (0.81423611111111 - 0.7)/(1 - 0.7) = 0.380787037037043$, and this value lies in the second interval, so that the sixth letter is B.
- We now apply the mapping $h_2(0.380787037037043) = (0.380787037037043 - 0.1)/(0.7 - 0.1) = 0.467978395061738$, and this value lies in the second interval, so that the seventh letter is B.
- We now apply the mapping $h_2(0.467978395061738) = (0.467978395061738 - 0.1)/(0.7 - 0.1) = 0.613297325102897$, and this value lies in the second interval, so that the eigth letter is B.
- We now apply the mapping $h_2(0.613297325102897) = (0.613297325102897 - 0.1)/(0.7 - 0.1) = 0.855495541838162$, and this value lies in the third interval, so that the nineth letter is C.
- We now apply the mapping $h_3(0.855495541838162) = (0.855495541838162 - 0.7)/(1 - 0.7) = 0.518318472793872$, and this value lies in the second interval, so that the tenth letter is B.

In summary, the text is BCBBCBBBCB.

Exercise 5 The first 99 A's restrict us to the interval $[0, 0.99^{99}]$. The last B restricts us to the interval $[0.99^{100}, 0.99^{99}]$. The arithmetic code is thus $0.99^{99}(1+0.99)/2 = 0.995 \times 0.99^{99} \approx 0.367880989461478$. We need $\lceil -\log_2(0.99^{99}0.01\rceil + 1 = 10$ bits for the arithmetic code.

- Clearly the first bit is 0.
- We compute $2 \times 0.367880989461478 \approx 0.735761978922956$, so that the second bit is 1.

- We then compute $2 \times 0.735761978922956 - 1 \approx 0.471523957845911$ so that the third bit is 0.
- We then compute $2 \times 0.471523957845911 = 0.943047915691822$ so that the fourth bit is 1.
- We then compute $2 \times 0.943047915691822 - 1 \approx 0.886095831383645$, so that the fifth bit is 1. We then compute $2 \times 0.886095831383645 - 1 \approx 0.772191662767289$, so that the sixth bit is 1.
- We then compute $2 \times 0.772191662767289 - 1 \approx 0.544383325534579$, so that the seventh bit is 1.
- We then compute $2 \times 0.544383325534579 - 1 \approx 0.088766651069157$, so that the eighth bit is 0.
- We then compute $2 \times 0.088766651069157 \approx 0.177533302138315$, so that the nineth bit is 0.
- Finally we compute $2 \times 0.177533302138315 \approx 0.355066604276630$, so that the final bit is 0.

The arithmetic code is thus 0101111000.

Section 7.6

Section 8.1

Section 8.2

Section 9.1

Exercise 3

(a) $p''(a) = f''(a)$ gives that $2b_2 = f''(a)$, so that

$$b_2 = f''(a)/2.$$

$p'(a) = f'(a)$ then gives that $b_1 + 2b_2 a = f'(a)$, so that

$$b_1 = -2b_2 a + f'(a) = -f''(a)a + f'(a).$$

$p(a) = f(a)$ then gives that $b_0 + b_1 a + b_2 a^2 = f(a)$, so that

$$\begin{aligned} b_0 &= -b_1 a - b_2 a^2 + f(a) = -(-f''(a)a + f'(a))a - f''(a)a^2/2 + f(a) \\ &= f(a) - f'(a)a + \frac{f''(a)}{2}a^2. \end{aligned}$$

(b) $p''(a) = f''(a)$ gives as before that $2b_2 = f''(a)$, so that $b_2 = f''(a)/2$. $p'(a) = f'(a)$ now gives that $b_1 = f'(a)$, while $p(a) = f(a)$ gives that $b_0 = f(a)$.

Section 9.2

Exercise 4

(a) Since $p_1(x_0) = p_2(x_0) = f(x_0)$, $p_1(x_1) = p_2(x_1) = f(x_1)$, $p_1(x_2) = p_2(x_2) = f(x_2)$, we get that

$$\begin{aligned} p(x_0) &= p_2(x_0) - p_1(x_0) = f(x_0) - f(x_0) = 0 \\ p(x_1) &= p_2(x_1) - p_1(x_1) = f(x_1) - f(x_1) = 0 \\ p(x_2) &= p_2(x_2) - p_1(x_2) = f(x_2) - f(x_2) = 0. \end{aligned}$$

Therefore, the values at the interpolation points are 0.

(b) If we write $p(x) = ax^2 + bx + c$, $p(x_0) = p(x_1) = p(x_2) = 0$ is the same as

$$\begin{aligned} a(x_0)^2 + bx_0 + c &= 0 \\ a(x_1)^2 + bx_1 + c &= 0 \\ a(x_2)^2 + bx_2 + c &= 0. \end{aligned}$$

Subtracting equations 1 and 2, and 1 and 3, we get the equations

$$\begin{aligned} a((x_0)^2 - (x_1)^2) + b(x_0 - x_1) &= 0 \\ a((x_0)^2 - (x_2)^2) + b(x_0 - x_2) &= 0. \end{aligned}$$

Dividing by $x_0 - x_1 \neq 0$ in the first equation, and by $x_0 - x_2 \neq 0$ in the second equation, we get

$$\begin{aligned} a(x_0 + x_1) + b &= 0 \\ a(x_0 + x_2) + b &= 0. \end{aligned}$$

Subtracting these we get that $a(x_1 - x_2) = 0$, so that $a = 0$ since $x_1 - x_2 \neq 0$. Now clearly $b = 0$ also, and $c = 0$. This means that $p(x) = 0$, so that $p_1(x) = p_2(x)$. In other words, the interpolating polynomial is unique.

(c) We now write $p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0$, and we get $n+1$ equations of the form

$$\begin{aligned} a_n(x_0)^n + a_{n-1}(x_0)^{n-1} + \ldots + a_0 &= 0 \\ a_n(x_1)^n + a_{n-1}(x_1)^{n-1} + \ldots + a_0 &= 0 \\ &\ldots\ldots \\ a_n(x_n)^n + a_{n-1}(x_n)^{n-1} + \ldots + a_0 &= 0, \end{aligned}$$

since also here the polynomial $p(x_i) = p_2(x_i) - p_1(x_i) = 0$ for all i. The proof that all the a_i are 0 goes in the same way, but requires much more work. This is easier proved with the aid of more linear algebra than we have learnt at this time.

Section 9.3

Exercise 3

(a) The Newton form of the quadratic, interpolating polynomial is

$$p_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1).$$

We have that

$$\begin{aligned} f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1-2}{1-0} = -1 \\ f[x_1, x_2] &= \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{0-1}{2-1} = -1 \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{-1-(-1)}{2-0} = 0. \end{aligned}$$

Inserting this in $p_2(x)$ we get that $p_2(x) = 2 - (x - x_0) = 2 - x$.

(b) Both f and p_n are interpolating polynomials of degree n, and we know then from Exercise 4 that they must be equal. In (a) the function was already a second degree polynomial, so that it must be equal to its interpolant too. This is why we obtained that p_s equaled the function itself.

Section 9.4

Section 10.2

Section 10.3

Exercise 5 For the function $f(x) = (x-1)^3$ the secant method gives

$$\begin{aligned} x_n &= x_{n-1} - \frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})} f(x_{n-1}) \\ &= x_{n-1} - \frac{x_{n-1} - x_{n-2}}{(x_{n-1}-1)^3 - (x_{n-2}-1)^3}(x_{n-1}-1)^3 \\ &= x_{n-1} - \frac{x_{n-1} - x_{n-2}}{(x_{n-1}-1)^3 - (x_{n-2}-1)^3}(x_{n-1}-1)^3 \end{aligned}$$

(a) The first 7 iterations give

1.15789473684

1.11710677382

1.08899801740

1.06700831862

1.05063360476

1.03820748989

1.02884624587

(b) We see that we still have a deviation bigger than 10^{-2} after all these iterations, so that we don't seem to get 62% new correct digits per iterations (we should at least obtain at least one new correct digit for every second iteration). This examples therefore does not agree with Observation 10.15. But notice that $f'(1) = 0$, so that we cannot find a γ as demanded by Theorem 10.14. This is the reason why we do not observe the convergence speed noted in Observation 10.15.

Section 10.4

Exercise 6

(a) We have that $f'(x) = 2x$, so that the Newton iteration takes the form

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{(x_n)^2 - 2}{2x_n} = \frac{x_n^2 + 2}{2x_n}.$$

if we subtract $\sqrt{2}$ and substitute $e_n = x_n - \sqrt{2}$ on both sides we obtain

$$e_{n+1} = \frac{x_n^2 + 2}{2x_n} - \sqrt{2} = \frac{x_n^2 + 2 - 2\sqrt{2}x_n}{2x_n} = \frac{(x_n - \sqrt{2})^2}{2x_n} = \frac{e_n^2}{2x_n}$$

(b) The secant method is

$$\begin{aligned} x_n &= x_{n-1} - \frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})} f(x_{n-1}) \\ &= x_{n-1} - \frac{x_{n-1} - x_{n-2}}{(x_{n-1})^2 - (x_{n-2})^2}((x_{n-1})^2 - 2) = x_{n-1} - \frac{(x_{n-1})^2 - 2}{x_{n-1} + x_{n-2}}. \end{aligned}$$

If we subtract $\sqrt{2}$ on both sides we can write this as

$$\begin{aligned} e_n &= e_{n-1} - \frac{(x_{n-1} + \sqrt{2})(x_{n-1} - \sqrt{2})}{x_{n-1} + x_{n-2}} \\ &= e_{n-1} - e_{n-1}\frac{x_{n-1} + \sqrt{2}}{x_{n-1} + x_{n-2}} = e_{n-1}\left(1 - \frac{x_{n-1} + \sqrt{2}}{x_{n-1} + x_{n-2})}\right) \\ &= e_{n-1}\frac{x_{n-1} + x_{n-2} - x_{n-1} - \sqrt{2}}{x_{n-1} + x_{n-2}} = e_{n-1}\frac{x_{n-2} - \sqrt{2}}{x_{n-1} + x_{n-2}} \\ &= \frac{e_{n-1}e_{n-2}}{x_{n-1} + x_{n-2}}. \end{aligned}$$

Exercise 7

(a) We have that $f'(x) = -1/x^2$, so that the Newton iteration is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{1/x_n - R}{-1/(x_n)^2} \\ &= x_n + x_n - R(x_n)^2 = x_n(2 - Rx_n). \end{aligned}$$

In this formula we do not need to perform division, only multiplication. By iterating this formula we can therefore approximate $1/R$ by only applying multiplications.

Section 11.1

Exercise 5

(a) On my computer 10^{-8} is the power of 10 which gives the least error in the approximation. This can be btained by running the following program:

```
from math import *

for p in range(15):
  h=10.0**(-p)
  print p, abs((exp(1+h)-exp(1))/h-exp(1))
```

(b) If we use the values $\epsilon^* = 7 \times 10^{-17}$ from Example 11.15, then Lemma 11.14 gives the optimal h $h^* = 2\sqrt{\epsilon^*} \approx 1.6733 \times 10^{-8}$ (terms cancel since $f(a) = f''(a)$).

Exercise 6

(a) We now write

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2} f''(a) + \frac{h^3}{6} f'''(\xi_h)$$

for some ξ_h in the interval between a and $a+h$. We can now write

$$f'(a) - \frac{f(a+h) - f(a)}{h} = -\frac{h}{2} f''(a) - \frac{h^2}{6} f'''(\xi_h).$$

After taking absolute values, the error estimate becomes

$$\frac{h}{2}|f''(a)| + \frac{h^2}{6} \max_{x \in [a,a+h]} |f'''(x)|.$$

This error estimate is similar to the one we obtained when we used a first degree Taylor expansion, but we also need a general bound on the third derivative. we still need to bound the second derivative, but only at the point a.

(b) We now write $f(a+h) = f(a) + f'(\xi_h)h$ for some ξ_h in the interval between a and $a+h$. In this expression $f'(a)$ is not present, so it is not possible to obtain an estimate of the truncation error using this Taylor expansion.

(c) The linear Taylor polynomial is the best because it is is the shortest possible Taylor expansion which can give an estimate of $f'(a)$ (as we showed in (b)), and also the one which gives the simplest expression for the truncation error in that it does not depend on any derivatives higher than the second order (as we showed in (a)).

Section 11.2

Exercise 1 The Newton form of the interpolating polynomial is $p_2(x) = f(a) + f[a, a+h](x-a) + f[a, a+h, a+2h](x-a)(x-(a+h))$. We compute that $p_2'(a) = f[a, a+h] - f[a, a+h, a+2h]h$. The divided differences can be computed as

$$\begin{aligned} f[a, a+h] &= \frac{f(a+h) - f(a)}{h} \\ f[a+h, a+2h] &= \frac{f(a+2h) - f(a+h)}{h} \\ f[a, a+h, a+2h] &= \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}. \end{aligned}$$

The approximation is thus

$$\begin{aligned} f'(a) \approx p_2'(a) &= f[a, a+h] - f[a, a+h, a+2h]h \\ &= \frac{f(a+h) - f(a)}{h} - \frac{f(a+2h) - 2f(a+h) + f(a)}{2h} \\ &= -\frac{f(a+2h) - 4f(a+h) + f(a)}{2h} \end{aligned}$$

Section 11.3

Exercise 5

(b) We can plot the cuve together with the secants as follows with Python:

```
from numpy import *
from scitools.easyviz import *

x=arange(0,6,0.05,float)
plot(x,(-x**2+10*x-5)/4)
hold('on')
plot([1,3],[(-1**2+10*1-5)/4,(-3**2+10*3-5)/4])
plot([1,5],[(-1**2+10*1-5)/4,(-5**2+10*5-5)/4])
plot([3,5],[(-3**2+10*3-5)/4,(-5**2+10*5-5)/4])
```

Section 11.4

Exercise 2

(a) On my computer 10^{-3} is the power of 10 which gives the least error in the approximation. This can be tested by running the following program:

```
from math import *

for p in range(15):
  h=10.0**(-p)
  print p, abs((exp(1-2*h)-8*exp(1-h)+8*exp(1+h)-exp(1+2*h))/(12*h)-exp(1))
```

(b) If we use the value $\epsilon^* = 7 \times 10^{-17}$ from Example 11.15 then (11.31) gives the optimal h $h^* = \sqrt[5]{\frac{27\epsilon^*}{2}} \approx 9.8875 \times 10^{-4}$ (terms cancel since $f(a) = f^{(5)}(a)$)

Section 11.5

Exercise 2

(a) On my computer 10^{-4} is the power of 10 which gives the least error in the approximation. This can be tested by running the following program:

```
from math import *

for p in range(15):
  h=10.0**(-p)
  print p, abs((exp(1-h)-2*exp(1)+exp(1+h))/h**2-exp(1))
```

(b) If we use the value $\epsilon^* = 7 \times 10^{-17}$ from Example 11.15 then Observation 11.24 gives the optimal choice of h $h^* = \sqrt[4]{36\epsilon^*} \approx 2.2405 \times 10^{-4}$ (terms cancel since $f(a) = f^{(4)}(a)$).

Exercise 3

(a) We compute the Taylor polynomial of f about a (with degree 3), and evaluate in $a+h$ and $a-h$:

$$f(a-h) = f(a) - f'(a)h + f''(a)\frac{h^2}{2} - f^{(3)}(a)\frac{h^3}{6} + f^{(4)}(\xi_1)\frac{h^4}{24}$$
$$f(a+h) = f(a) + f'(a)h + f''(a)\frac{h^2}{2} + f^{(3)}(a)\frac{h^3}{6} + f^{(4)}(\xi_2)\frac{h^4}{24},$$

where $\xi_1 \in [a-h, a]$, $\xi_2 \in [a, a+h]$. If we add these equations together we obtain that

$$f(a-h) + f(a+h) = 2f(a) + f''(a)h^2 + f^{(4)}(\xi_1)\frac{h^4}{24} + f^{(4)}(\xi_2)\frac{h^4}{24},$$

which also can be written

$$f''(a) - \frac{f(a+h) - 2f(a) + f(a-h)}{h^2} = -\frac{h^4}{24}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2)),$$

which is (11.33).

(b) We set $\overline{f(a-h)} = f(a-h)(1+\epsilon_1)$, $\overline{f(a)} = f(a)(1+\epsilon_2)$, $\overline{f(a+h)} = f(a+h)(1+\epsilon_3)$, and obtain

$$\begin{aligned}
&f''(a) - \frac{\overline{f(a+h)} - 2\overline{f(a)} + \overline{f(a-h)}}{h^2} \\
&= f''(a) - \frac{f(a+h) - 2f(a) + f(a-h)}{h^2} - \frac{\epsilon_1 f(a-h) - 2\epsilon_2 f(a) + \epsilon_3 f(a+h)}{h^2} \\
&= -\frac{h^4}{24}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2)) - \frac{\epsilon_1 f(a-h) - 2\epsilon_2 f(a) + \epsilon_3 f(a+h)}{h^2}.
\end{aligned}$$

(c) We set $M_1 = \max_{x\in[a-h,a+h]} |f^{(4)}(x)|$ and $M_2 = \max_{x\in[a-h,a+h]} |f(x)|$, insert $|\epsilon_i| \le \epsilon^*$, take absolute values in the equation above and obtain:

$$\begin{aligned}
&f''(a) - \frac{\overline{f(a+h)} - 2\overline{f(a)} + \overline{f(a-h)}}{h^2} \\
&= \left| -\frac{h^4}{24}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2)) - \frac{\epsilon_1 f(a-h) - 2\epsilon_2 f(a) + \epsilon_3 f(a+h)}{h^2} \right| \\
&\le \frac{h^4}{24}M_1 + \frac{h^4}{24}M_1 + \frac{M_2\epsilon^*}{h^2} + \frac{2M_2\epsilon^*}{h^2} + \frac{M_2\epsilon^*}{h^2} \\
&= \frac{h^4}{12}M_1 + \frac{4\epsilon^*}{h^2}M_2.
\end{aligned}$$

Exercise 4

(a) If the approximation method $f'(a) \approx c_1 f(a-h) + c_2 f(a+h)$ is to be exact for $f(x) = 1$, we must have that $0 = c_1 + c_2$, since $f(a-h) = f(a+h) = 1$, and since $f'(x) = 0$. Therefore we must have that $c_2 = -c_1$.

If the method is to be exact for $f(x) = x$ we must in the same way have that

$$1 = c_1(a-h) + c_2(a+h) = c_1(a-h) - c_1(a+h) = -2c_1 h,$$

so that $c_1 = -\frac{1}{2h}$,so that also $c_2 = \frac{1}{2h}$. The method therefore becomes $-\frac{1}{2h}f(a-h) + \frac{1}{2h}f(a+h) = \frac{f(a+h)-f(a-h)}{2h}$

(b) If $f(x) = cx + d$ we have that $f'(x) = c$, and the method takes the form

$$\frac{f(a+h) - f(a-h)}{2h} = \frac{c(a+h) + d - (c(a-h) + d)}{2h} = \frac{2ch}{2h} = c,$$

so that the method is exact for all polynomials of degree ≤ 1.We see that the method coincides with the symmetric Newton-method for differentiation, and it therefore has an error of order $\frac{1}{h^2}$, which is better than the Newton's quotient (which has an error of order $\frac{1}{h}$). It is worse than the four point method for numerical differentiation,which has order $\frac{1}{h^4}$.

Here it also should have been mentioned that the method also is exact for polynomials of degree ≤ 2 (also see Exercise 5). There are several ways to see this. First, the error estimate from Section 11.3 uses $f^{(3)}(x)$, and since all second degree polynomials have a third derivative equal to 0, the error must be zero. One could also as above substitute $f(x) = x^2$ into the formula:

$$\frac{f(a+h) - f(a-h)}{2h} = \frac{(a+h)^2 - (a-h)^2}{2h} = \frac{4ah}{2h} = 2a,$$

which also is $f'(a)$. Finally, the symmetric Newton quotient was defined as the derivativee at a of the unique parabola interpolating f at $a-h, a$, and $a+h$. If f itself is a parabola it is equal to this interpolant since it is unique, so that the symmetric Newton quotient must return the derivative.

(c) If the approximation method $f''(a) \approx c_1 f(a-h) + c_2 f(a) + c_3 f(a+h)$ is exact for $f(x) = 1$, we must have that $0 = c_1 + c_2 + c_3$. If it is exact for $f(x) = x$ we must have that

$$0 = c_1(a-h) + c_2 a + c_3(a+h) = a(c_1 + c_2 + c_3) + h(-c_1 + c_3) = h(-c_1 + c_3),$$

which gives that $c_1 = c_3$. If it is exact for $f(x) = x^2$ we must have that

$$\begin{aligned} 2 &= c_1(a-h)^2 + c_2 a^2 + c_3(a+h)^2 \\ &= a^2(c_1 + c_2 + c_3) - 2ahc_1 + 2ahc_3 + h^2(c_1 + c_3) = 2c_1 h^2, \end{aligned}$$

which gives that $c_1 = \frac{1}{h^2}$. We therefore also get that $c_3 = \frac{1}{h^2}$, and that $c_2 = -c_1 - c_2 = -\frac{2}{h^2}$, so that the method becomes

$$\frac{1}{2h} f(a-h) - \frac{1}{h} f(a) + \frac{1}{2h} f(a+h) = \frac{f(a-h) - 2f(a) + f(a+h)}{h^2}.$$

We see that this coincides with the already seen three point method to compute the second derivative in this section.

(d) All third degree polynomials have a fourth derivative equal to 0, and therefore the truncation error becomes 0 ($M_1 = 0$ in Theorem 11.23). Alternatively we can substitute $f(x) = x^3$ into the formula:

$$\begin{aligned}
&\frac{f(a-h) - 2f(a) + f(a+h)}{2h} \\
&= \frac{(a-h)^3 - 2a^3 + (a+h)^3}{h^2} \\
&= \frac{a^3 - 3a^2h + 3ah^2 - h^3 - 2a^3 + a^3 + 3a^2h + 3ah^2 + h^3}{h^2} \\
&= \frac{6ah^2}{h^2} = 6a,
\end{aligned}$$

which coincides with the second derivative of f in a.

Section 12.1

Section 12.2

Exercise 2 We get that

$$I_{mid} = \frac{1}{2}f(1/4) + \frac{1}{2}f(3/4) = \frac{1}{2}\frac{1}{16} + \frac{1}{2}\frac{9}{16} = \frac{1}{2}\frac{10}{16} = \frac{5}{16},$$

so that the first alternative is correct.

Section 12.3

Exercise 2 We have that

$$I_{trap} = \frac{1}{2}\left(\frac{f(0) + f(1)}{2} + f\left(\frac{1}{2}\right)\right) = \frac{1}{2}\left(\frac{0+1}{2} + \frac{1}{4}\right) = \frac{3}{8},$$

so that the second alternative is correct.

Exercise 5 We insert from (12.16) and (12.17) and get

$$
\begin{aligned}
&\left|\int_a^b f(x)dx - \frac{f(a)+f(b)}{2}(b-a)\right| \\
&= \left|f(a_{1/2})(b-a) + \frac{1}{2}\int_a^b (x-a_{1/2})^2 f''(\xi_1)dx - (f(a_{1/2}) + \frac{(b-a)^2}{16}f''(\xi_2) + \frac{(b-a)^2}{16}f''(\xi_3))(b-a)\right| \\
&= \left|\frac{1}{2}\int_a^b (x-a_{1/2})^2 f''(\xi_1)dx - (\frac{(b-a)^2}{16}f''(\xi_2) + \frac{(b-a)^2}{16}f''(\xi_3))(b-a)\right| \\
&\le \left|\frac{1}{2}\int_a^b (x-a_{1/2})^2 f''(\xi_1)dx\right| + \left|\frac{(b-a)^2}{16}f''(\xi_2)\right|(b-a) + \left|\frac{(b-a)^2}{16}f''(\xi_3)\right|(b-a) \\
&\le \frac{1}{2}\int_a^b (x-a_{1/2})^2 |f''(\xi_1)|dx + \frac{(b-a)^3}{16}M + \frac{(b-a)^3}{16}M \\
&\le \frac{M}{2}\int_a^b (x-a_{1/2})^2 dx + \frac{(b-a)^3}{8}M \\
&= \frac{M}{2}[\frac{1}{3}(x-a_{1/2})^3]_a^b + \frac{(b-a)^3}{8}M = \frac{M}{6}\left(\frac{1}{8}(b-a)^3 + \frac{1}{8}(b-a)^3\right) + \frac{(b-a)^3}{8}M \\
&= \left(\frac{1}{24}+\frac{1}{8}\right)M(b-a)^3 = \frac{1}{6}M(b-a)^3.
\end{aligned}
$$

Section 12.4

Exercise 4

(a) Since the error in the trapezoidal rule is bounded by $(b-a)\frac{h^2}{6}\max_{x\in[a,b]}|f''(x)|$, and since $f''(x) = \frac{8}{(1+2x)^3}$ (which attains maximum absolute value for $x = 0$), we need to choose h so that

$$(b-a)\frac{h^2}{6}\max_{x\in[a,b]}|f''(x)| = 8\frac{h^2}{6} \le 10^{-10},$$

which gives that $h \le 10^{-5}\sqrt{3}/2$. The number of function evaluations is then $\approx 1/h = 2\cdot 10^5/\sqrt{3} \approx 115470.05$, which means that at least 115471 evaluations are needed.

(b) Since the error in the midpoint rule is bounded by $(b-a)\frac{h^2}{24}\max_{x\in[a,b]}|f''(x)|$, we obtain as in (a) that

$$(b-a)\frac{h^2}{24}\max_{x\in[a,b]}|f''(x)| = 2\frac{h^2}{6} \le 10^{-10},$$

which gives that $h \le 10^{-5}\sqrt{3}$. The number of function evaluations is then $\approx 1/h = 10^5/\sqrt{3} \approx 57735.03$, which means that at least 57736 evaluations are needed.

(c) Since the error in Simpson's rule is bounded by $(b-a)\frac{h^4}{180}\max_{x\in[a,b]}|f^{(iv)}(x)|$, and since $f^{(iv)}(x) = \frac{384}{(1+2x)^5}$ (which attains maximum absolute value for $x = 0$), we need to choose h so that

$$(b-a)\frac{h^4}{180}\max_{x\in[a,b]}|f^{(iv)}(x)| = 384\frac{h^4}{180} \le 10^{-10},$$

which gives that $h \le (180\cdot 10^{-10}/384)^{1/4}$. The number of function evaluations is then $\approx 1/h = (180\cdot 10^{-10}/384)^{-1/4} \approx 382.17$, which means that at least 383 evaluations are needed.

Exercise 6

(a) It is enough to verify Simpson's rule on each interval. The integrals become

$$\begin{aligned}
\int_{a-h}^{a+h} x^3\,dx &= \frac{1}{4}((a+h)^4-(a-h)^4) = \frac{1}{4}(8a^3h+8ah^3) = 2a^3h+2ah^3\\
\int_{a-h}^{a+h} x^2\,dx &= \frac{1}{3}((a+h)^3-(a-h)^3) = \frac{1}{3}(6a^2h+2h^3) = \frac{h}{3}(6a^2+2h^2)\\
\int_{a-h}^{a+h} x\,dx &= \frac{1}{2}((a+h)^2-(a-h)^2) = \frac{1}{2}4ah = 2ah\\
\int_{a-h}^{a+h} dx &= 2h.
\end{aligned}$$

We also get that the rule itself becomes

$$\begin{aligned}
\frac{h}{3}((a-h)^3+4a^3+(a+h)^3) &= \frac{h}{3}(6a^3+6ah^2) = 2a^3h+2ah^3\\
\frac{h}{3}((a-h)^2+4a^2+(a+h)^2) &= \frac{h}{3}(6a^2+2h^2)\\
\frac{h}{3}((a-h)+4a+(a+h)) &= 2ah\\
\frac{h}{3}(1+4+1) &= 2h.
\end{aligned}$$

This shows that Simpsons' rule is exact for the given polynomials.

(b) For $f(x) = bx^3+cx^2+dx+e$ the integral is

$$\int_{a-h}^{a+h} f(x)\,dx = b\int_{a-h}^{a+h} x^3\,dx + c\int_{a-h}^{a+h} x^2\,dx + d\int_{a-h}^{a+h} x\,dx + e\int_{a-h}^{a+h} 1\,dx.$$

The rule itself now gives

$$\begin{aligned}
&\frac{h}{3}(f(a-h)+4f(a)+f(a+h)) \\
&= b\frac{h}{3}((a-h)^3+4a^3+(a+h)^3)+c\frac{h}{3}((a-h)^2+4a^2+(a+h)^2) \\
&+d\frac{h}{3}((a-h)+4a+(a+h))+e\frac{h}{3}(1+4+1)
\end{aligned}$$

We see that equality follows from that we have equality for $x^3, x^2, x, 1$.

(c) Since $f^{(4)}(x) = 0$ for all x for every third degree polynomial, it follows directly from the error estimate that the method is exact for such functions.

Section 13.1

Section 13.2

Exercise 2 The differential equation is separable since it can be written as $\frac{x'}{1-x} = \sin t$. This gives that $-\ln|1-x| = -\cos t + C$, so that $1-x = De^{\cos t}$, so that $x(t) = 1 - De^{\cos t}$.

(a) The solution in this case is $x(t) = 1 - e^{\cos t}$.

(c) The solution in this case is $x(t) = 1 + e^{\cos t}$.

(d) The solution in this case is $x(t) = 1 + 2e^{\cos t}$.

Section 13.3

Exercise 3

(a) We get that

$$\begin{aligned}
x_1 &= x_0 + hf(t_0, x_0) = 1 + 0.1f(0,1) = 1 + 0.1(0+1) = 1.1 \\
x_2 &= x_1 + hf(t_1, x_1) = 1.1 + 0.1f(0.1, 1.1) = 1.1 + 0.1(0.1+1.1) = 1.22 \\
x_3 &= x_2 + hf(t_2, x_2) = 1.22 + 0.1f(0.2, 1.22) = 1.22 + 0.1(0.2+1.22) = 1.362.
\end{aligned}$$

Exercise 5 The code can be changed as follows:

```
h = (b - a)/n;
t_0 = b;
for k = 0, 1, ..., n - 1
    x_{k+1} = x_k - hf(t_k, x_k);
    t_{k+1} = b - (k+1)h;
```

Here we simply have used Euler's method with negative step size.

Exercise 6 If we insert $x'(t) = f(t, x(t))$ in the approximation we get that $f(t,x) \approx \frac{x(t+h)-x(t)}{h}$, which can be written $x(t+h) - x(t) \approx hf(t, x(t))$, which means that $x(t+h) \approx x(t) + hf(t, x(t))$. The right hand side here is equivalent to one step with Euler's method.

Section 13.4

Exercise 2 If the step length is h, we obtain the approximation

$$x(h) \approx x(0) + hf(t,x) = 1 + h\sin h.$$

The error is given by

$$R_1(h) = \frac{h^2}{2}x''(\xi)$$

where $\xi \in (0,h)$. Since $x'(t) = \sin x(t)$, we have

$$x''(t) = x'(t)\cos x(t) = \sin x(t)\cos x(t) = \frac{\sin\big(2x(t)\big)}{2}$$

We therefore have $|x''(t)| \le 1/2$, so

$$|R_1(h)| \le \frac{h^2}{4}.$$

Section 13.5

Exercise 1

(c) We get first that $x'(1) = 1\cdot 0 - \sin 0 = 0$. We differentiate and get

$$x''(t) = x + tx' - \cos x x' = x + x'(t - \cos x),$$

so that $x''(1) = 0 + 0 = 0$. We differentiate again and get

$$x'''(t) = x' + x' + tx'' + \sin x(x')^2 - \cos x x'' = 2x' + (t - \cos x)x'' + \sin x(x')^2,$$

so that $x'''(1) = 0$ also.

(d) We get first that $x'(1) = 1/1 = 1$. We differentiate and get $x''(t) = 1/x - tx'/x^2$, so that $x''(1) = 1 - 1 = 0$. We differentiate again and get

$$x'''(t) = -\frac{x'}{x^2} - \frac{(x' + tx'')x^2 - 2xt(x')^2}{x^4} = -\frac{x'}{x^2} - \frac{(x' + tx'')x - 2t(x')^2}{x^3},$$

so that $x'''(1) = -1 - (1-2) = 0$.

Section 13.6

Exercise 5

(a) We get that

$$x''(t) = 2t + 3x^2 x'(t) - x'(t) = 2t + (3x^2 - 1)x'(t),$$

where we substitute $t^2 + x^3 - x$ for x'.

(b) One step with the quadratic Taylor method here becomes

$$x_{k+1} = x_k + hx'_k + \frac{h^2}{2}(2t_k + (3x_k^2 - 1)x'_k)$$

where $x'_k = t_k^2 + x_k^3 - x_k$. Since $x'_0 = 0 + 1 - 1 = 0$, one step with the quadratic Taylor method therefore gives $x_1 = 1$. The first step with the quadratic Taylor method clearly gives $x_1 = 1$, no matter what h is. If we use more steps the next step becomes

$$\begin{aligned} x'_1 &= t_1^2 + x_1^3 - x_1 = t_1^2 \\ x_2 &= x_1 + ht_1^2 + \frac{h^2}{2}(2t_1 + (3x_1^2 - 1)t_1^2) = 1 + ht_1^2 + \frac{h^2}{2}(2t_1 + 2t_1^2) \end{aligned}$$

If $h = 0.5$ we get that $x_2 = 1 + \frac{1}{8} + \frac{1}{8}(1 + \frac{1}{2}) = 1 + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} = \frac{21}{16} \approx 1.3125$. If $h = 0.2$ we get in the same way that the next steps give $x_2 = 1.0176$, $x_3 = 1.08109021746$, $x_4 = 1.24076835064$, $x_5 = 1.62941067817$.

(c) The code can look as follows:

```
for N in [10,100,1000]:
  xk=1
  tk=0
  h=1.0/N
  for k in range(N):
    xder=tk**2+xk**3-xk
    xk=xk+h*xder + (h**2/2)*(2*tk+(3*xk**2-1)*xder)
    tk=tk+h
  print xk
```

If we run the code we get the approximation $x(1) \approx 1.787456775$. If we substitute N with 100 or 1000 we get instead the approximations 1.90739098078, and 1.9095983769.

Section 13.7

Exercise 2

(a) One step with Euler's method gives

$$x_1 = x_0 + x_0 = 2.$$

(b) One step with quadratic Taylor gives

$$x_1 = x_0 + x_0 + \frac{1}{2}x_0 = 2.5$$

(c) One step with Euler's midpoint method gives

$$\begin{aligned} x_{1/2} &= x_0 + \frac{1}{2}x_0 = \frac{3}{2} \\ x_1 &= x_0 + x_{1/2} = \frac{5}{2} = 2.5. \end{aligned}$$

(d) With Runge-Kuttas method we get

$$\begin{aligned} &k_0 = 1 \\ &k_1 = 1 + \frac{k_0}{2} = 1.5 \\ &k_2 = 1 + \frac{k_1}{2} = 1.75 \\ &k_3 = 1 + 1.75 = 2.75 \\ &x_1 = x_0 + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3) = 1 + \frac{1}{6}(1 + 3 + 3.5 + 2.75) \approx 2.7083. \end{aligned}$$

(e) With two steps in Runge-Kuttas method we get

$$
\begin{aligned}
k_0 &= 1 \\
k_1 &= 1 + \frac{k_0}{4} = 1.25 \\
k_2 &= 1 + \frac{k_1}{4} = 1.3125 \\
k_3 &= 1 + 0.60625 = 1.6025 \\
x_1 &= x_0 + \frac{1}{12}(k_0 + 2k_1 + 2k_2 + k_3) \approx 1.6440 \\
k_0 &= x_1 = 1.6440 \\
k_1 &= x_1 + \frac{k_0}{4} \\
k_2 &= x_1 + \frac{k_1}{4} \\
k_3 &= x_1 + \frac{k_2}{2} \\
x_2 &= x_1 + \frac{1}{12}(k_0 + 2k_1 + 2k_2 + k_3) \approx 2.7100
\end{aligned}
$$

(f) The code looks as follows if we use the fourth order Runge Kutta method:

```
for N in [10,100,1000,10000]:
  xk=1
  h=1.0/N
  for k in range(N):
    k0=xk
    k1=xk+h*k0/2
    k2=xk+h*k1/2
    k3=xk+h*k2
    xk=xk+h*(k0+2.0*k1+2.0*k2+k3)/6
  print xk
```

If we run the code we get 2.71827974414. If we change N to $100, 1000, 10000$ we get 2.71828182823, 2.71828182846, and 2.71828182846.

(g) It looks like the values converge to e.

(h) The equation is of first order with linear coefficients, and we see that the solution is $x(t) = e^t$, so that $x(1) = e$.

Exercise 3 The code can look as follows:

```
from math import *

xk=2.0+exp(1)
tk=0.0
N=10
h=2.0*pi/N
for k in range(N):
  xk=xk+h*(-xk*sin(tk)+sin(tk))
  tk=tk+h
print xk

xk=2.0+exp(1)
tk=0.0
for k in range(N):
  xkhalf=xk+h*(-xk*sin(tk)+sin(tk))/2.0
  xk=xk+h*(-xkhalf*sin(tk+h/2.0)+sin(tk+h/2.0))
print xk
```

(a) Euler's metode here takes the form $x_{k+1} = x_k + h(-x_k \sin t_k + \sin t_k)$.

(b) Euler's midpoint method here takes the form

$$x_{k+1/2} = x_k + \frac{h}{2}(-x_k \sin t_k + \sin t_k)$$
$$x_1 = x_k + h(-x_{k+1/2} \sin(t_k + h/2) + \sin(t_k + h/2))$$

(c) This differential equation is separable, and one can show that the solution is on the form $1 + De^{\cos(t)}$. In particular, the solution is periodic with period 2π, so that the solution should satisfy $x(2\pi) = 2 + e$. If we run the code for different N you will see that it is first for N larger than 1000 that we begin to get close, so that we can not say anything for N as small as those given in the exercise.

Section 13.8

Exercise 1 We define $x_1 = x$, $x_2 = x_1'$. The equation can then be written as $x_2' + \sin(tx_2) - x_1^2 = e^t$, so that $x_2' = e^t - \sin(tx_2) + x_1^2$. The third alternative is therefore correct.

Section 13.9

Exercise 2 If we differentiate the first equation we get that $x''' = 1 + x' + y''$. This inserted in the second equation gives that $y''' = 1 + x' + y'' + y'' = 1 + x' + 2y''$. We

define $x_1 = x$, $x_2 = x'$, $y_1 = y$, $y_2 = y'$, $y_3 = y''$, we get the following equations:

$$\begin{aligned}
x_1' &= x_2 \\
x_2' &= t + x_1 + y_2 \\
y_1' &= y_2 \\
y_2' &= y_3 \\
y_3' &= 1 + x_2 + 2y_3.
\end{aligned}$$

where the second equation corresponds to the first original equation, and the fifth equation corresponds to the rewritten equation $y''' = 1 + x' + y'' + y'' = 1 + x' + 2y''$.

Exercise 4 In order to solve this exercise, it is most convenient to implement a function which returns the values for $\boldsymbol{f}(t, \boldsymbol{x})$. The two equations can be written as a first order system as

$$\begin{aligned}
x_1' &= x_2 \\
x_2' &= 2y_1 - \sin\left(4t^2 x_1\right) \\
y_1' &= y_2 \\
y_2' &= -2x_1 - \frac{1}{2t^2(x_2)^2 + 3},
\end{aligned}$$

with the initial condition $(x_1(0), x_2(0), y_1(0), y_2(0)) = \boldsymbol{x}_0 = (1, 2, 1, 0)$. We get that $\boldsymbol{f}(0, \boldsymbol{x}_0) = (2, 2 - \sin 0, 0, -2 - 1/(0+3)) = (2, 2, 0, -7/3)$, so that the first step with Euler's method with $h = 1$ gives

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{f}(0, \boldsymbol{x}_0) = (1, 2, 1, 0) + (2, 2, 0, -7/3) = (3, 4, 1, -7/3).$$

We now get that

$$f(1, \boldsymbol{x}_1) = (4, 2 - \sin(12), -7/3, -6 - 1/(32+3)) \approx (4, 2.5366, -7/3, -6.0286)$$

so that the second step with Euler's method gives

$$\begin{aligned}
\boldsymbol{x}_2 = \boldsymbol{x}_1 + \boldsymbol{f}(1, \boldsymbol{x}_1) &\approx (3, 4, 1, -7/3) + (4, 2.5366, -7/3, -6.0286) \\
&\approx (7, 6.5366, -1.3333, -8.3619).
\end{aligned}$$

This means that our approximations become

$$x(2) \approx 7, \quad x'(2) \approx 6.5366, \quad y(2) \approx -1.3333, \quad y'(2) \approx -8.3619.$$

For Euler's midpoint method the first step becomes

$$\begin{aligned}
\boldsymbol{x}_{1/2} &= \boldsymbol{x}_0 + f(0,\boldsymbol{x}_0)/2 = (1,2,1,0) + (2,2,0,-7/3)/2 = (2,3,1,-7/6)\\
f(1/2,\boldsymbol{x}_{1/2}) &= (3,2-\sin 2,-7/6,-4-1/(9/2+3)) \approx (3,1.0907,-1.1667,-4.1333)\\
\boldsymbol{x}_1 &= \boldsymbol{x}_0 + f(1/2,\boldsymbol{x}_{1/2}) \approx (1,2,1,0) + (3,1.0907,-1.1667,-4.1333)\\
&= (4,3.0907,-0.1667,-4.1333).
\end{aligned}$$

The second step becomes

$$\begin{aligned}
f(1,\boldsymbol{x}_1) &\approx (3.0907,-0.0454,-4.1333,-8.0452)\\
\boldsymbol{x}_{3/2} &= \boldsymbol{x}_1 + f(1,\boldsymbol{x}_1)/2\\
&\approx (4,3.0907,-0.1667,-4.1333) + (3.0907,-0.0454,-4.1333,-8.0452)/2\\
&= (5.5454,3.0680,-2.2333,-8.1560)\\
f(3/2,\boldsymbol{x}_{3/2}) &\approx (3.0680,-4.1169,-8.1560,-11.1128)\\
\boldsymbol{x}_2 &= \boldsymbol{x}_1 + f(3/2,\boldsymbol{x}_{3/2}) \approx (7.0680,-1.0262,-8.3226,-15.2461).
\end{aligned}$$

This means that our approximations become

$$x(2) \approx 7.0680, \quad x'(2) \approx -1.0262, \quad y(2) \approx -8.3226, \quad y'(2) \approx -15.2461.$$

Section 14.1

Section 14.2

Section 15.1

Section 15.2

Section 15.3