Answers to exercises

2.1 (a) $l = 31.3 \pm 0.2$ m (unless the precision is really 20 ± 1 cm; in that case $l = 31,30 \pm 0,20$ m); (b) $c = 15.3 \pm 0.1$ mM; (c) $\kappa = 252$ S/m; (d) $k/\mathrm{L\,mol^{-1}\,s^{-1}} = (35.7 \pm 0.7) \times 10^2$ or $k = (35.7 \pm 0.7) \times 10^2$ $\mathrm{L\,mol^{-1}\,s^{-1}}$; (e) $= 2.00 \pm 0.03$.

2.2 (a) 173 Pa; (b) 2.31×10^5 Pa = 2.31 bar; (c) 2.3 mmol/L; (d) 0.145 nm or 145 pm; (e) 24.0 kJ/mol; (f) 8400 kJ (note that often cal or Cal is written while kcal is meant); (g) 556 N; (h) 2.0×10^{-4} Gy; (i) 0.080 L/km or 8.0 L/100 km; (j) 6.17×10^{-30} Cm; (k) 1.602×10^{-40} $\mathrm{F\,m^2}$.

3.1 (a) 3.00 ± 0.06 (relative uncertainty 2%); (b) 6.0 ± 0.3 (relative uncertainty $\sqrt{3^2+4^2}\%$); (c) 3.000 ± 0.001. Note that $\log_{10}(1 \pm \delta) = \pm 0.434 \ln(1+\delta) \approx \pm 0.434\delta = 0.00087$. Sometimes it is easier to evaluate both boundaries: $\log_{10} 998 = 2.99913$ and $\log_{10} 1002 = 3.00087$; (d) 2.71 ± 0.06 (relative uncertainty $\sqrt{1.5^2+1^2}\%$).

3.2 $k = \ln 2/\tau_{1/2}$. The relative uncertainty in k equals the relative uncertainty in $\tau_{1/2}$. The absolute uncertainty in $\ln k$ equals the relative uncertainty in k : $\sigma(\ln k) = \sigma(k)/k$. The following values are obtained:

$\frac{1000}{T/\mathrm{K}}$	k/s^{-1}	$\ln(k/\mathrm{s}^{-1})$
1.2771	$(0.347 \pm 0.017) \times 10^{-3}$	-7.97 ± 0.05
1.2300	$(1.155 \pm 0.077) \times 10^{-3}$	-6.76 ± 0.07
1.1862	$(2.89 \pm 0.24) \times 10^{-3}$	-5.85 ± 0.08
1.1455	$(7.70 \pm 0.86) \times 10^{-3}$	-4.87 ± 0.11

Python code for logarithmic plot:
`autoplotp([Tinv,k],yscale='log',ybars=sigk)`, with `Tinv, k, sigk` from table.

3.3 9.80 ± 0.03 (Relative uncertainty is $\sqrt{0.2^2 + (2 \times 0.1^2)} = 0.28\%$)

3.4 Because $\Delta G = RT\ln(kh/k_BT)$, the derivative with respect to T equals $(\Delta G/T) + R$. That is $(30\,000/300) + 8.3 = 108.3$. This implies that a deviation in T of ± 5 yields a deviation in ΔG of $108.3 \times 5 = 540$ J/mol.

3.5 The volume from $r = 1$ equals 4.19 $\mathrm{mm^3}$; the mean of 1000 samples was found to be 4.30 $\mathrm{mm^3}$ and the standard deviation was found to be 1.27. The systematic error in the "naive" volume is -0.11, much less than the standard deviation.

4.1 $f(0) = 0.598\,74$; $f(1) = 0.315\,12$; $f(2) = 0.074\,635$; $f(3) = 0.010\,475$; $f(4) = 0.000\,965$.

4.2 You are looking for $1 - f(0) = 1 - 0.99^{20} = 0.182$.

4.3 With a sample size of n and probability p of voting candidate no. 1, the average number of votes for no. 1 will be pn with variance $p(1-p)n$ (binomial distribution). To obtain a relative standard deviation of 0.01, $n \geq 10\,000$ is required.

4.4 This distribution is binomial. (a) $\hat{p}_1 = k_0/n$; (b) $\sigma_0 = \sqrt{(k_0 k_1/n)}$; (c) same as (b); (d) Note that deviations in k_0 and k_1 are fully anticorrelated. Therefore $(k_1 \pm \sigma)/(k_0 \mp \sigma) = r(1 \pm \sigma k_1^{-1})/(1 \mp \sigma k_0^{-1}) = r[1 \pm \sigma(k_1^{-1} + k_0^{-1})]$. Standard deviation of r equals $[1 + (k_1/k_0)]/\sqrt{n})$.

4.5 Sum $\mu^k/k!$ over $k = 0$ to $k = \infty$, yielding e^{μ}.

4.6 Generate Poisson probabilities $f(k, \mu)$ and cumulative probabilities $F(k, \mu)$ from

```
from scipy import stats
f=stats.poisson.pmf
F=stats.poisson.cdf
```

(a) 2.98; (b) $(k \geq 8) : 1 - F(7, 3) = 0.012$; (c) 4 beds; 0.185 patients transported. The optimization can best be done by defining a function `cost(n)`, which computes the costs with n beds, and finding a whole number n for which cost(n) is minimal. For example:

```
def cost(n):
    krange=arange(1,n,1)
    avbeds=(f(krange,3)*krange).sum()+n*
      (1-F((n-1),3))
    return (1-F(n,3))*1500.+(n-avbeds)*300.
```

4.7 This is a Poisson process: s.d. equals the square root of the number of observed impulses. The light measurement gives 900 ± 30 impulses and the dark measurement gives 100 ± 10 impulses. The light intensity is proportional to $(900 - 100) \pm \sqrt{30^2 + 10^2} = 800 \pm 32$. Hence the relative s.d. is 4%. After repeating the measurement 100 times (or after a hundredfold increase of measuring time), the measured numbers become 100× larger, but the (absolute) errors become only 10× larger. The relative uncertainty becomes 10× smaller (0.4%).

4.8 $F(0.1) - F(-0.1) = 2 \times (0.5 - 0.4602) = 0.0796$. Note that this is almost equal to $f(0) \times 0.2 = 0.0798$.

4.9 $f(6) = 6.076 \times 10^{-9}$; $F(-6) = 1.0126 \times 10^{-9}(37/38 + \ldots) = 9.8600 \times 10^{-10}$. Compare to the exact value `stats.norm.cdf(-6.)` $= 9.8659 \times 10^{-10}$.

4.10 (a) The uniform distribution $f(x) = 1$, $0 \leq x < 1$, has average 0.5 and variance $\sigma^2 = \int_0^1 (x - 0.5)^2\, dx = 1/12$; adding 12 numbers yields a 12 times larger variance. (b) and (c) with Python code:

```
x=randn(100)
autoplotc(x,yscale='prob')
```

4.11 mean: $\langle t \rangle = 1/k$; variance $\langle (t - k^{-1})^2 \rangle = 1/k^2$.
Use $\int_0^{\infty} t^n \exp(-kt)\, dt = n!/k^{n+1}$ for evaluating integrals.

4.12 SSR = 115.6; SSE = 154.0; F = 6.005; cdf(F, 1, 8) = 0.96; treatment is significant at 5% confidence level.

5.1 Yes, Fig. 2.1 gives a straight line; $\mu = 8.68$; $\sigma = 1.10$. Accuracy ca 0.05.

5.2 Just work out the square in $\frac{1}{n} \sum (x_i - \langle x \rangle)^2$.

5.3 No: apply the equation to $y = x - c$; all terms with c cancel.

5.4 Usually things go wrong for c exceeding 10^7. Suggestion: use Python function:

```
def rmsd(c):
      n=1000
      x=randn(n)+c
      xav=x.sum()/n
      rmsd1=((x-xav)**2).sum()/n
      rmsd2=(x**2).sum()/n - xav**2
      return [rmsd1,rmsd2]
```

The first value is correct; the second may be in error.

5.5 The estimated s.d. equals $\hat{\sigma} = \sqrt{\langle(\Delta x)^2\rangle n/(n-1)}$, where $\langle(\Delta x)^2\rangle$ is the mean squared deviation. For $n = 15$ the s.d. in σ is 19%; this gives $\hat{\sigma} = 5 \pm 1$. For $n = 200$ the s.d. in σ is 5%; this gives $\hat{\sigma} = 5.1 \pm 0.3$. In the first case the mean is 75 ± 5; in the second case the mean is 75.3 ± 5.1.

5.6
1. (a) average: 29.172 s; (b) msd: 0.0315 s^2; (c) rmsd: 0.1775 s; (d) range: 28.89–29.43 s; median: 29.24 s; first quartile: 29.02 s; third quartile: 29.33 s
2. (a) mean: 29.172 s; (b) variance: 0.0354 s^2; (c) s.d.: 0.188 s; (d): 0.063 s; (e) 0.0177 s; 0.047 s; 0.016 s.
3. 29.16 ± 0.06 km/hr; deviation: $+6.6 \pm 4\%$ km/hr.
4. No. The inaccuracy of keeping the right speed is incorporated into the measurements.
5. 80%: 29.10–29.25; 90%: 29.07–29.27; 95%: 29.06–29.28 s.
6. 80%: 123.06–123.74; 90%: 123.00–123.82; 95%: 122.91–123.92 km/hr.
7. 80%: 123.06–123.76; 90%: 122.97–123.85; 95%: 122.88–123.94 km/hr.
8. 80%: 123.03–123.76; 90%: 122.91–123.90; 95%: 122.79–124.02 km/hr.

5.7 Use weighted averaging: $N_A = 6.022\,141\,89(20)$.

5.8 The plot can be made by first constructing a list z of all 27 possible values:

```
z=[-1.]+[-2./3.]*3+[-1./3.]*6+[0.]*7+[1./3.]*6
   +[2./3.]*3+[1.]
autoplotc(z,yscale='prob')
```

This plot perfectly fits a straight line through (0, 50%); $\sigma = 0.47$ (exact: 0.471).

5.9 Note that the characteristic function of $\delta(x - a)$ equals $\exp(iat)$. The probability density function of a variable x, randomly chosen from -1, 0 and $+1$, consists of three delta functions $\Phi(t) = \frac{1}{3}\delta(x+1) + \frac{1}{3}\delta(x) + \frac{1}{3}\delta(x-1)$. Its characteristic function is $\frac{1}{3}[1 + \exp(-it) + \exp(it)]$. The pdf of the sum of three such variables x_1, x_2, x_3 is the convolution of

$f(x_1), f(x_2)$ and $f(x_3)$; its characteristic function equals $\Phi(t)^3$. Working out the third power yields

$[\exp(3it) + 3\exp(2it) + 6\exp(-it) + 7 + 6\exp(-it) + 3\exp(-2it) + \exp(-3it)]/27$.

Its Fourier transform contains seven delta functions at $x = -3, -2, -1, 0, 1, 2, 3$. If not the sum but the average of three values is taken, the x values reduce by a factor 3.

The variance can be obtained from the second derivative of the characteristic function at $t = 0$, or directly from the pdf, and equals 2 for the sum, or 2/9 for the average.

6.1 Line goes through points (9, 100) and (188, 1) (precision ca 1%). Gives $k = \ln 100/(188 - 9) = 0.0257$ and $c_0 = 126$.

6.2 (In too many decimals:) Lineweaver–Burk: $K_m = 1/0.0094 = 106.383$; $v_{max} = K_m(0.04 + 0.0094)/0.35 = 15.015$; Eadie–Hofstee: $K_m = (15 - 2)/(0.120 - 0.007) = 115.04$; $v_{max} = 0.120K_m + 2 = 15.805$; Hanes: $v_{max} = 500/(39 - 7.5) = 15.873$; $K_m = 7.5v_{max} = 119.05$.

6.3 Plot the data $1000/T, k$ on a horizontal scale from 1.14 to 1.30. Draw the best line through the points; this line goes through $(1.14, 9.5\text{e}-3)$ and $(1.30, 2.0\text{e}-4)$. Hence $E/1000R = [\ln(9.5\text{e}-3/2.\text{e}-4)]/[1.30 - 1.14] = 24.13$ and $E = 200.63$ kJ/mol. Varying the slope yields E between 191.69 and 208.24. Result: $E = 201 \pm 8$ kJ/mol. Your values may differ (insignificantly) from these numbers.

6.4 68.8 ± 0.6 mmol/L (note that the unit molar (M, mol/L) is obsolete).

7.1 Use the Python program `fit` (code **7.7**). With the function $y = ax + b$, the best fit gives $a = 7.23 \pm 0.31$ and $b = 0.0636 \pm 0.0017$, with correlation coefficient $\rho_{ab} = -0.816$. From this follows $v_{\rm max} = 1/b = 15.7 \pm 0.4$ and $K_{\rm m} = a/b = 114 \pm 8$. The *relative* uncertainty δ in a/b is found from

$$\delta^2 = \left(\frac{\sigma_a}{a}\right)^2 + \left(\frac{\sigma_b}{b}\right)^2 - 2\rho_{ab}\frac{\sigma_a\sigma_b}{ab}.$$

A direct nonlinear fit to the data `[S,v]` yields $v_{\rm max} = 15.7 \pm 0.4$ and $K_{\rm m} = a/b = 115 \pm 8$.

7.2 Use the Python program `fit` (code **7.7**). With the function $y = -aT + b$ you find $\Delta S = a = 0.259 \pm 0.013$, $b = 110.3 \pm 3.9$ and $\rho_{ab} = 0.99778516$. Extrapolation to $T = 350$ gives $\Delta G(350) = 19.81 \pm 0.71$, where the s.d. has been calculated from

$$\sigma^2_{\Delta G} = 350^2\sigma_a^2 + \sigma_b^2 - 2.350.\rho_{ab}\sigma_a\sigma_b.$$

With the function $y = -a(T - 300) + b$ you find $\Delta S = a = 0.259 \pm 0.013$, $b = 32.74 \pm 0.26$ and $\rho_{ab} = 0$. Extrapolation to $T = 350$ now gives $\Delta G(350) = 19.81 \pm 0.71$, where the s.d. has been calculated from

$$\sigma_{\Delta G}^2 = 50^2 \sigma_a^2 + \sigma_b^2.$$

The results are exactly the same, but the extrapolation is much simpler in the second case where $\rho = 0$.

7.3

$$\sigma_y^2 = \left(\frac{dy}{dt}\right)^2 \sigma_t^2 = \frac{\sigma_t^2}{t^2}.$$

Hence $w_i = \sigma_y^{-2} = t_i^2/\sigma_t^2 \propto t_i^2$.

7.4 $a = 71.5 \pm 3.8$; $b = 19.1 \pm 3.9$; $p = 0.0981 \pm 0.0061$; $q = 0.0183 \pm 0.0034$. Note that these values deviate from the graphical estimate. Fitting to multiple exponentials is quite difficult; the parameters have a strong mutual correlation (e.g. $\rho_{ab} = 0.98$) and sometimes a minimum cannot be found.

7.5 For $c =$ position lens, $yf(x,[f,c]) = c + f * (c - x)/(c - x - f)$. Least-squares fitting yields $f = 55.15$; $c = 187.20$. The $S_0 = 3.13$; 4 degrees of freedom. Covariance matrix ($S_0/4*$ leastsq output yields $\sigma_1 = 0.2$; $\sigma_2 = 0.3$; $\rho = 0.91$. Result: $f = 55.1 \pm 0.2$ mm.

7.6 Find out by yourself.

7.7 The output of the program `report` gives sufficient comment. Try

```
x=arange(100.); sig=ones(100)
y1=randn(100); y2=y1+0.01*x
```

`report([x,y1,sig])` may produce an insignificant drift, while y2 may imply a significant drift.