Numerical Methods in Physics

Exercise 1, 2016/17

Numerical evaluation of integrals concerning the thermodynamics of crystals by means of numerical integration and cubic spline interpolation

by Heinrich Sormann

1 Objective

In this exercise we will calculate the *electronic specific heat* of different metals. For the evaluation of this property, we will need to compute integrals $\int_{-\infty}^{\infty} f(x)dx$ numerically. Part of the function $f(x)$ is only available in the form of tabulated values. This means that, in order to perform the integrals, we need to interpolate (spline) this data, because the integration methods we use require us to provide the values of the integrand $f(x)$ at arbitrary values x.

2 Preliminaries

We will start with a revision of numerical integration and spline interpolation in Matlab, and with some short exercises on both.

2.1 Numerical integration

In order to numerically integrate data in Matlab, we can use the function `integral(fun,xmin,xmax)`. This built-in routine needs at least 3 parameters, where `fun` is a *function handle* to the integrand. `xmin` and `xmax` are the x-limits of the integral, which can also be $\pm$`inf`.

As a quick reminder: function handles in Matlab are created by an `@`, followed by the arguments of the function in parentheses. For normal use with the routine `integral`, the function handle can only take 1 argument (which can be a vector or matrix) and must return a vector or matrix of the same size.

Listing 1: Example of how to use the function `integral`. Notice the use of vector operators in `F`.

```
F = @(x) exp(-x.^2) ./ x;
result = integral(F, 1, inf);
% or even shorter:
result = integral(@(x) exp(-x.^2) ./ x, 1 , inf);
```

`integral` can also handle multiple integrals at once. To achieve that, we must pass the additional parameters `(...,'ArrayValued',true,...)` to the integration routine. Then, of course, the function handle `fun` must only take a scalar input and produce a vector or matrix output containing all the functions we want to integrate.

Listing 2: Example use of parallel evaluation of `integral`

```
a = [0.2, 0.5, 1, 2];
F = @(x) exp(-a * x.^2) ./ x;
result = integral(F, 1, inf, 'ArrayValued', true);
```

We can specify a required precision for the integral, calling the routine `integral` the two additional parameters `'AbsTol'` and `'RelTol'` followed by the upper bound for the absolute and relative error, respectively (e. g. `integral(@(x) x.^2, 0, 2, 'AbsTol', 1E-12)`).

For further information, you can take a look at `http://de.mathworks.com/help/matlab/ref/integral.html`.

2.1.1 First preliminary task

Consider the integral

$$I(a) = \int_0^1 \frac{\ln(1+x)}{x(1+x)}\, e^{-ax}\, dx \tag{1}$$

where a is a real, positive parameter. For $a = 0$ this integral can be solved analytically.

$$I(0) = \frac{\pi^2}{12} - \frac{1}{2}(\ln 2)^2 \approx 0.5822405265\ldots$$

Calculate $I(a)$ for the values $a = (0, 0.2, 0.5, 1, 2)$ numerically, with a tolerance on the absolute error of 10^{-7}. For $a = 0$, compare the result with the analytical value and calculate its absolute error $\Delta I(0) = |I(0)_{\text{num}} - I(0)_{\text{an}}|$.

Your results should be

a	0.0	0.2	0.5	1.0	2.0
$I(a)$	0.5822405	0.5369832	0.4784139	0.4005981	0.2954009

2.2 Splines

In order to interpolate a set of data that is only known at discrete sampling points, we will use cubic splines. With this method, the unknown function between the given data points is represented by piecewise cubic polynomials, where the polynomial coefficients differ from interval to interval. At the given points, the polynomials and their derivatives are required to be continuous in order to obtain a smooth function.

In MATLAB, the function `yy = spline(x,Y,xx)` can perform this task. Given a set of data points `Y` at positions `x`, the splines are evaluated at the new positions `xx` and returned as `yy`.

If the same set of data points must be splined multiple times, `spline` can also be called without specifying a set of new x-values: `pp = spline(x,Y)`. The returned variable `pp` is a `struct` containing definitions of the intervals and the spline coefficients for each interval (piecewise polynomial). To evaluate the spline at arbitrary data points, we then need to call `yy = ppval(pp,xx)`. If we don't know a priori on which set of points xx we need to interpolate the function, this second

method is way more efficient than the one described above, because the spline coefficients are only calculated once.

When using splines, in the first and last interval additional conditions must be specified, since there are not enough data points to determine all polynomial coefficients. `spline` uses the so-called `not-a-knot` end condition. Different conditions for the end intervals can be specified when using `pp = csape(x,y,conds)`, where `conds` is a string specifying the requested type of end condition. For our purposes, this is irrelevant, since it mainly influences the first and last interval, which are small and also contribute very little to our result.

More details about `spline` are at `http://de.mathworks.com/help/matlab/ref/spline.html`, `csape` is described at `http://de.mathworks.com/help/curvefit/csape.html`.

2.2.1 Second preliminary task

Given the following data points.

x	1.00	1.25	1.50	1.75	2.00
y	0.20	-0.10	-0.60	0.00	0.50

Calculate the spline coefficients and save the coefficients of the piecewise polynomial. Then, plot the data points and the spline (at small increments) together in a graph.

If you chose to use `not-a-knot` as the end condition, your spline should look something like this.

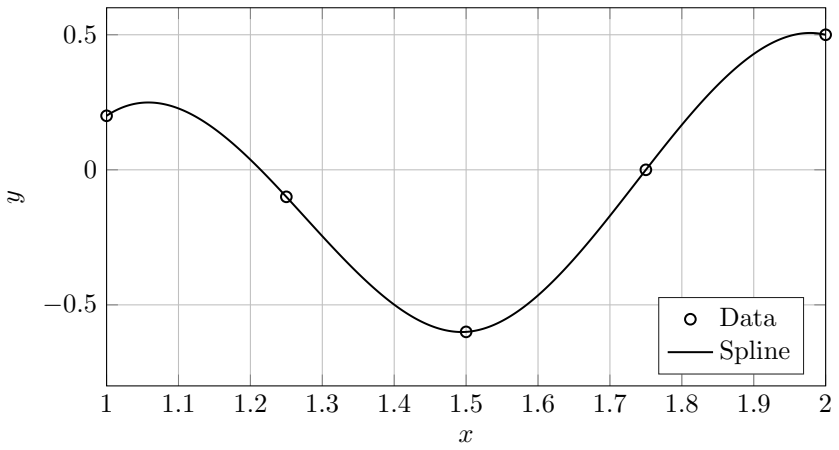

Figure 1: Cubic spline of the sample data points with not-a-knot end condition.

3 First task: Electronic specific heat

Now we can move on to our main task, which involves a practical physical problem: the numerical calculation of the electronic specific heat in metals. We will now provide some theoretical background information on how adding energy to a system changes the distribution of its electronic states, and how the heat capacity can be derived from that.

3.1 Fundamentals

This example uses concepts from statistical physics, quantum mechanics, and solid state physics, of which you probably don't have an advanced knowledge. The information we provide should be sufficient for the scope of the exercise.

In order to raise the temperature of a solid, energy must be added to the system. This means that the internal energy of a solid is a function of its absolute temperature.

$$E = E(T)$$

This internal energy is the sum of the energies of all particles within the solid. A solid metal consists of ions sitting on a regular lattice (assuming a perfect crystal) and valence electrons which can move about relatively free within the crystal (we consider core electrons as part of the ions). Therefore, the thermodynamic properties of the system are determined by the internal energy of the ionic lattice (vibrations) and the energy of the free electrons.

$$E(T) = E_{\text{ion}}(T) + E_e(T)$$

In this exercise we are only interested in the *electronic* contribution to the internal energy $E_e(T)$.

The crystal can be thought of as a periodic repetition (in all 3 spatial directions) of a small *unit cell* containing one or more atoms. Because of this periodicity, it is sufficient to consider the unit cell instead of the whole crystal. We define the *energy density*

$$u_e(T) = \frac{E_e(T)}{N_{\text{uc}}} = \int_{-\infty}^{\infty} \epsilon\, N(\epsilon) f(\epsilon, T) d\epsilon \tag{2}$$

where N_{uc} is the number of unit cells in the crystal. This formula can be interpreted as follows. ϵ is the energy of a specific electronic state, $N(\epsilon)$ is the number of available states per unit cell at that specific energy, and $f(\epsilon, T)$ is the probability that a state at energy ϵ is occupied at temperature T. Integrating this expression over all energies, we get the energy density of our crystal. We will now consider each term in the integrand separately.

Fermi function Since electrons are fermions, which follow *Fermi–Dirac statistics*, the probability $f(\epsilon, T)$ is given by the *Fermi-Dirac distribution*

$$f(\epsilon, T) = \left(\exp\left(\frac{\epsilon - \mu(T)}{k_{\text{B}} T} \right) + 1 \right)^{-1} \tag{3}$$

where $\mu(T)$ is the so-called *chemical potential* of the electrons. $f(\epsilon, T)$ is also called the *Fermi function*.

In order to understand this formula a little better, it is easier to start at very low temperatures.

$$\lim_{T\to 0} f(\epsilon, T) = \lim_{T\to 0} \left(\exp\left(\frac{\epsilon - \mu(T)}{k_\mathrm{B} T} \right) + 1 \right)^{-1} = \Theta(\mu(0) - \epsilon)$$

where $\Theta(x)$ indicates the *Heaviside step function*, which is defined as

$$\Theta(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{2} & x = 0 \\ 1 & x > 0 \end{cases}$$

So at $T = 0$, the Fermi function looks like the blue line in fig. 2.

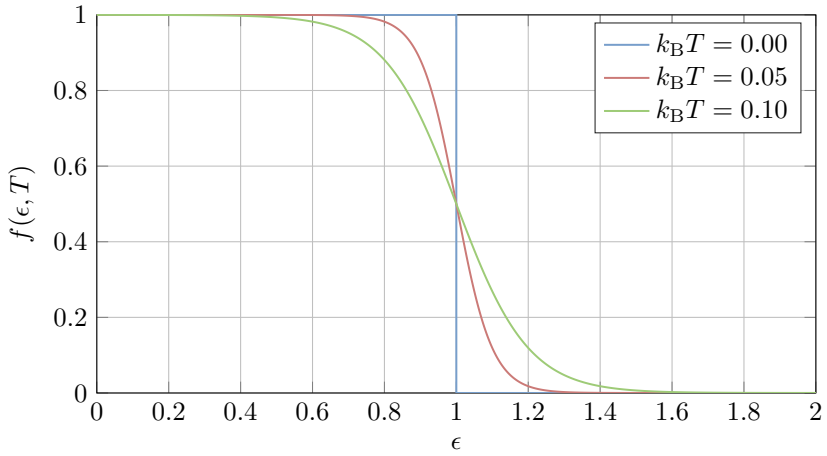

Figure 2: Fermi function for $\mu = 1$ at different temperatures. The energy ϵ is in arbitrary units, but must of course be the same as for $k_\mathrm{B}T$.

This means that at absolute zero, all quantum mechanical states with an energy below the chemical potential $\mu(0)$ are occupied (probability 1), and all states with an energy higher than $\mu(0)$ are empty (probability 0). The chemical potential at $T = 0$ is called the *Fermi energy* ϵ_F.

If we now increase the temperature, the step becomes smoother, as can be seen at the red and green line in fig. 2.

Chemical potential As indicated by the notation of $\mu(T)$, the position of the edge in the Fermi function may be temperature-dependent. $f(\epsilon, T)$ is always 0.5 at $\epsilon = \mu(T)$, which we refer to as the position of the step.

The temperature dependence of the chemical potential $\mu(T)$ follows from the conservation of electrons. The number of electrons at some energy ϵ is given by the number of states per unit cell at that energy multiplied by the probability that such a state is occupied. If we then integrate over all energies, we must reobtain the total number of electrons per unit cell. This is a known constant Z_0 for each material.

$$Z_0 \stackrel{!}{=} \int_{-\infty}^{\infty} N(\epsilon) f(\epsilon, T) d\epsilon = \int_{-\infty}^{\infty} \frac{N(\epsilon)}{\exp\left(\frac{\epsilon - \mu(T)}{k_\mathrm{B} T} \right) + 1} \, d\epsilon \tag{4}$$

The temperature dependence is now implicitly defined in eq. 4. For any given temperature T, the chemical potential $\mu(T)$ must be chosen so that eq. 4 is satisfied.

Density of states As mentioned before, we also need the number of quantum mechanical states per unit cell $N(\epsilon)$ at certain energies ϵ. More precisely, $N(\epsilon)d\epsilon$ is the number of states between energies ϵ and $\epsilon + d\epsilon$. This quantity is called the *density of states (DOS)* and has units of inverse energy. It is a material dependent property and can be calculated using sophisticated band structure methods. The DOS for the materials in this exercise are available as tabulated data.

Heat capacity Now we have all functions needed to evaluate eq. 2. After substituting the Fermi function in we have

$$u_e(T) = \int_{-\infty}^{\infty} \frac{\epsilon\, N(\epsilon)}{\exp\left(\frac{\epsilon-\mu(T)}{k_\mathrm{B}T}\right)+1}\, d\epsilon \tag{5}$$

for the electronic internal energy density.

Now we are interested in how this energy density changes with temperature. To this end we define the so-called *heat capacity*[1]

$$c_{v,\mathrm{uc}}(T) = \frac{\partial u_e(T)}{\partial T} = \int_{-\infty}^{\infty} \epsilon\, N(\epsilon)\, \frac{\partial}{\partial T}\left(\exp\left(\frac{\epsilon-\mu(T)}{k_\mathrm{B}T}\right)+1\right)^{-1} d\epsilon \tag{6}$$

We can do the derivation analytically by substituting $x = \frac{\epsilon-\mu(T)}{k_\mathrm{B}T}$

$$\frac{\partial}{\partial T}(e^x+1)^{-1} = -\frac{e^x}{(e^x+1)^2}\,\frac{\partial x}{\partial T} = -\frac{e^x}{(e^x+1)^2}\,\left(-\frac{\partial\mu(T)}{\partial T}\frac{1}{k_\mathrm{B}T} - \frac{\epsilon-\mu(T)}{k_\mathrm{B}T^2}\right)$$

So finally we get

$$c_{v,\mathrm{uc}}(T) = \int_{-\infty}^{\infty} \frac{\epsilon\, N(\epsilon)}{k_\mathrm{B}\,T^2} \exp\left(\frac{\epsilon-\mu(T)}{k_\mathrm{B}T}\right)\left(\exp\left(\frac{\epsilon-\mu(T)}{k_\mathrm{B}T}\right)+1\right)^{-2}\left(\mu'(T)\,T+\epsilon-\mu(T)\right) d\epsilon \tag{7}$$

which we can now use to compute the electronic contribution to the heat capacity in different metals.

Free electron model If the temperature is not too high and we assume that the electrons in the metal are completely free and only constrained by the outer edge of the crystal, we can approximate eq. 7 analytically (this is called the Sommerfeld expansion) to get a very simple formula for the heat capacity

$$c_{v,\mathrm{uc}}(T) \approx \frac{\pi^2}{3}\, N(\epsilon_\mathrm{F})\, k_\mathrm{B}^2\, T \tag{8}$$

which is linear in temperature. Because of the assumption of completely free electrons, this will likely only hold for those metals, in which the electrons do not feel a very strong potential from the ions (alkali metals, some noble metals, ...).

[1] An important side note: in this derivation, by heat capacity $c_{v,\mathrm{uc}}$ we refer to the *heat capacity per unit cell at constant volume*. Usually, the term "heat capacity" refers to the change of internal energy with respect to temperature of some specific object. We will later calculate the heat capacity per unit mass at constant volume, which is also called the *specific heat* c_v.

3.2 Calculation details

The materials we will calculate the specific heat c_v for are sodium, copper, and platinum. We still need to specify some details on how to do the calculation in practice.

Chemical potential For the temperature regime that we are interested in, we can make one significant simplification. Calculations show, that the temperature dependence of $\mu(T)$ in our T-range is neglectable. We also shift the energy scale, such that $\epsilon_\mathrm{F} = 0$, which we can always do without changing the physics. Now eq. 7 can be simplified by using

$$\mu(T) \approx \mu(0) = 0 \qquad \text{and} \qquad \mu'(T) \approx 0$$

and we get

$$c_{v,\mathrm{uc}}(T) = \int_{-\infty}^{\infty} \frac{\epsilon^2\, N(\epsilon)}{k_\mathrm{B}\, T^2} \exp\left(\frac{\epsilon}{k_\mathrm{B}T}\right) \left(\exp\left(\frac{\epsilon}{k_\mathrm{B}T}\right) + 1\right)^{-2} d\epsilon \tag{9}$$

Material properties The quantity we actually want to calculate is the specific heat c_v, which is the heat capacity per unit mass. So far, our formula gives the heat capacity per unit cell. To adjust for that, you need to calculate the number of unit cells per kg of material. The following quantities can be helpful.

Table 1: Physical properties of the materials in our calculation. Ω_0 is the volume of a unit cell, ρ is the density of the bulk material and T_melt is the melting temperature. a_0 denotes the *Bohr radius*.

	Ω_0 $[{a_0}^3]$	ρ $[\mathrm{kg\,m^{-3}}]$	T_melt [K]
Na	256.2881	971	371
Cu	79.6810	8960	1357
Pt	101.9794	21450	2045

When evaluating c_v, be careful to use correct units for the data and the natural constants. Some useful numbers are

$$k_\mathrm{B} = 8.6173 \cdot 10^{-5}\,\mathrm{eV/K}$$
$$1\,a_0 = 5.291772 \cdot 10^{-11}\,\mathrm{m}$$
$$1\,\mathrm{eV} = 1.602177 \cdot 10^{-19}\,\mathrm{J}$$

Density of states The function $N(\epsilon)$ is provided as data points in the files `dos_Na`, `dos_Cu`, and `dos_Pt` for sodium, copper, and platinum, respectively. They were calculated using the band structure program WIEN2k. The first line always contains the number of data points, while the following lines contain the energy in eV in the first and the density of states in eV^{-1} in the second column. They use the energy scale mentioned before, so that the Fermi energy is at $\epsilon_\mathrm{F} = 0$.

As discussed before, since we need the integrand of eq. 9 at arbitrary values ϵ, the data must be interpolated first. Compute the spline coefficients *once*, and then re-use the results to calculate the integral numerically, as described in section 2.2.

Because the data for the DOS is only available at finite energies, the integral, which actually ranges from $-\infty$ to $+\infty$, can only be evaluated between the minimum and maximum energy in the files. Because the DOS is always 0 for energies smaller than the minimum in the file, this part would not contribute anyway. For higher energies, the Fermi function is essentially 0, so this does also not introduce a noticeable error.

Numerical problems If we look at eq. 9, we see that the numerator and denominator can become very large, even though the fraction might not be. If we tell MATLAB to evaluate the expression, it will first calculate both values separately and only divide them afterwards. This can cause the inaccurate or, at worst, totally wrong results due to an overflow in the denominator. It is therefore advisable to re-write the expression $e^x(e^x+1)^{-2}$ into a different form (take a look at the definition of the hyperbolic cosine).

3.3 Expected results

In fig. 3 we can see the numerical WIEN2k as well as the free-electron-approximation results for the density of states of the different materials.

For sodium, both curves match each other quite well up to just below the Fermi energy. But even at ϵ_F and slightly above, the deviations are not too dramatic, so we expect – especially at low temperatures – that the exact and free-electron results for the specific heat should also match each other quite well.

In copper, at first sight the blue curve looks like a terrible approximation for the red one. However, it is important to keep in mind that the biggest contribution to c_v comes from the region around the Fermi energy, where the agreement between the two curves is not too bad. So also in this case we expect a relatively good agreement between the two specific heat behaviors, at least at low temperatures.

However, for platinum, we can clearly see – especially at the Fermi energy – that the two curves differ significantly. This suggests that we will see large differences also in the specific heat.

3.4 Tasks

Re-write the exponentials in eq. 9 into a numerically more stable form, as described above. Also, adjust eqs. 8 and 9 such that they will give c_v in $\mathrm{J/(kg\,K)}$.

Calculate the specific heat $c_v(T)$ of the 3 materials (Na, Cu, Pt) in $\mathrm{J/(kg\,K)}$ in the temperature range $100 \leq T \leq 4000\,\mathrm{K}$, with an increment of $\Delta T = 100\,\mathrm{K}$, using the exact formula (eq. 9), as well as the free electron approximation (eq. 8). For the numerical integration it is best to use `integral` with the `ArrayValued` option to evaluate all temperatures at once.

For each metal, plot the exact and approximate result together in a graph. Use one plot per material. Compare the two curves for each element. Pay special attention to the temperature range $T \leq 500\,\mathrm{K}$, in which the approximation should work better than at high temperatures. The materials we are studying are not solid for all calculated temperatures, but they melt at some temperature. Indicate this melting point in your diagrams.

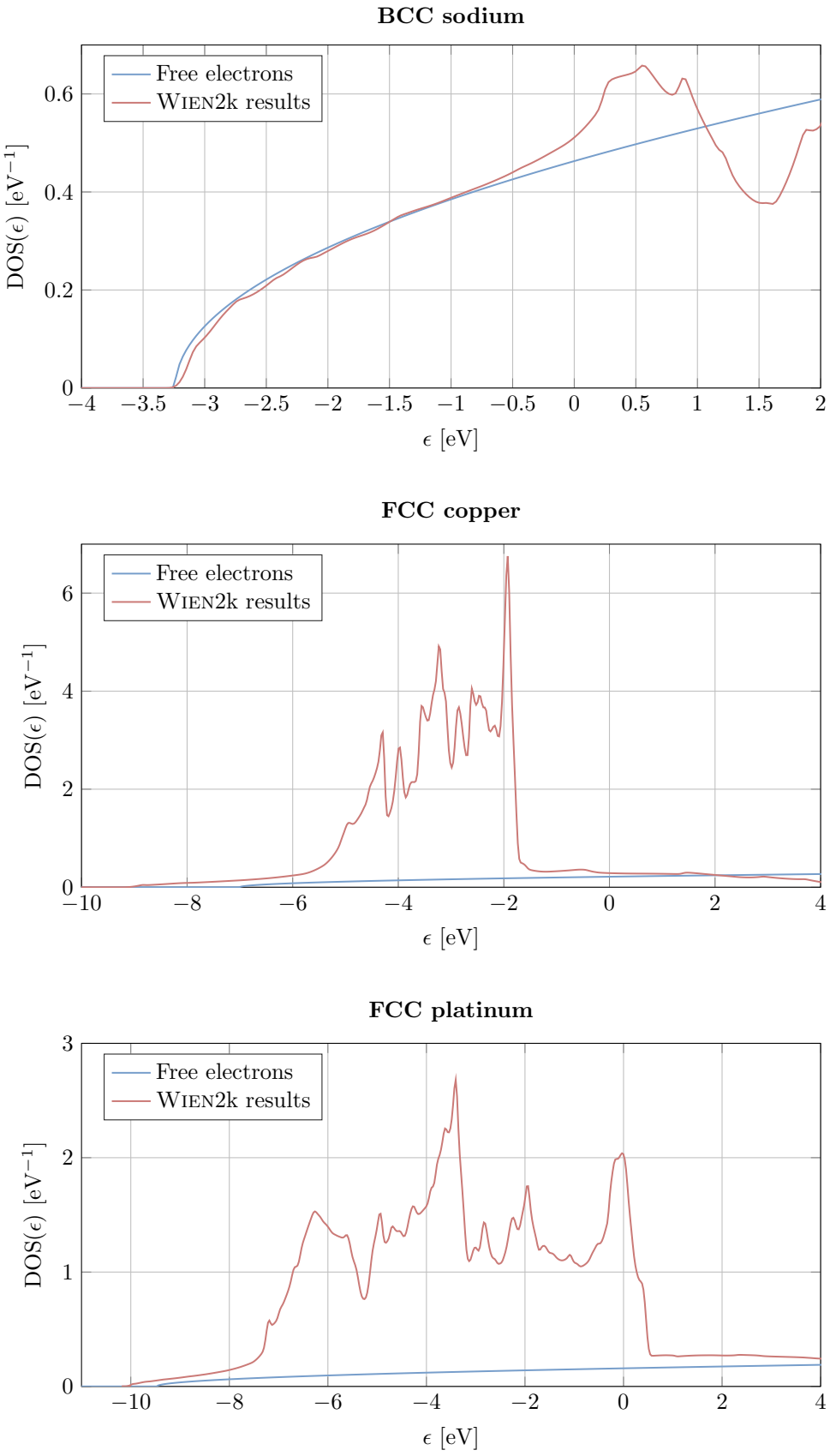

Figure 3: Exact and free-electron-model density of states of the investigated metals. ϵ is shifted so that $\epsilon_F = 0$.

4 Second task: Numerical integration

Now that we have seen a practical application of numerical integration, we will try to implement our own integrators and compare them to the results given by the internal MATLAB function `integral`. We will start with some basics on how numerical integration works. More details can be found in the script, chapter 6.

4.1 Fundamentals

A definite integral can be approximated numerically summing up weighted values of the integrand.

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{j=1}^{m} w_j f(x_j)$$

Here, m is the order of method, and w_j are weighting factors, which have to be specified. The values x_j, at which the integrand is evaluated, are often chosen to be the limits a and b of the integration interval, and equally spaced points in between. While a uniform distribution of the sampling points x_j is not strictly necessary, for simplicity we will consider only equally spaced points in this exercise.

The weights w_j can be calculated from the requirement, that if the integrand $f(x)$ is a polynomial of order $m-1$ or lower, the result of a method of order m should be exact. So if we use a fourth-order method to numerically integrate a polynomial of order 3, we must reobtain the exact analytical result. We write this as

$$\int_a^b P_{m-k}(x)dx \stackrel{!}{=} \frac{b-a}{2} \sum_{j=1}^{m} w_j P_{m-k}(x_j) \tag{10}$$

where $1 \leq k \leq m$ and $P_n(x)$ means a polynomial of order n.

4.2 Trapezoidal formula

The first method we will use is the so-called trapezoidal formula. It is an integration method of order 2, where $x_1 = a$ and $x_2 = b$. From eq. 10 we get $w_1 = w_2 = 1$.

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \left(f(a) + f(b)\right) \tag{11}$$

The name of this formula comes from the fact that the integral is approximated by a trapezoid formed by the function values $f(a)$, $f(b)$ and the corresponding points on the x-axis a and b.

4.3 Simpson formula

The second method to implement is the so-called Simpson formula, which is a method of order 3. As mentioned before, the uniformly distributed sampling points are $x_1 = a$, $x_2 = (a+b)/2$, and $x_3 = b$. Again using eq. 10, we find $w_1 = w_3 = 1/3$ and $w_2 = 4/3$.

$$\int_a^b f(x)dx \approx \frac{b-a}{6} \left(f(a) + 4\, f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{12}$$

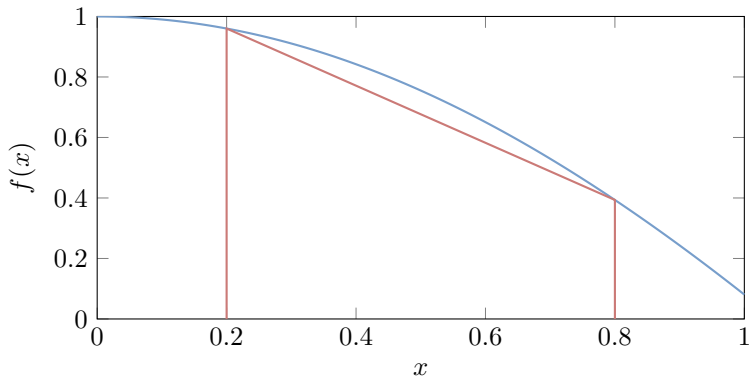

Figure 4: Visualization of the trapezoidal formula in red.

4.4 Error estimation

The methodological error made by these integration methods can be calculated analytically. Without derivation, the absolute errors for the two methods are

$$E_{\text{trap}} = -\frac{(b-a)^3}{12} f''(\xi) \qquad \text{and} \qquad E_{\text{Simps}} = -\frac{(b-a)^5}{2880} f^{(4)}(\xi)$$

with $\xi \in [a,b]$ in both cases.

4.5 Practical use

The given approximations will only produce good results, when the interval width $b - a$ is sufficiently small. In order to achieve better accuracy, we will always split the integration regions into sub-intervals. To obtain the total integral, we only need to sum up all the piecewise results. This type of integration formulas are called *composite* integration formulas.

The width of these intervals could in principal vary – and there are good adaptive methods that will choose them so that the total integral converges quickly – but we will only consider evenly spaced fixed-width intervals in this exercise.

4.6 Tasks

Implement the formulas 11 and 12 in two separate MATLAB functions with the following declaration lines.

```
function I = trapezoid(fun, a, b, n_intervals)
```

```
function I = simpson(fun, a, b, n_intervals)
```

where `fun` is a function handle that takes 1 argument `x` (which might be a matrix) and returns the function values at these x-points as a matrix of the same size. `a` and `b` are the integration boundaries and `n_intervals` is the number of sub-intervals the region from a to b should be divided into for integration. For each sub-interval, evaluate the formulas 11 and 12, respectively. The return value `I` then contains the sum of all the sub-integrals.

Test your functions by calculating the integral

$$I = \int_{-10}^{10} \frac{1}{1+x^2}\, dx$$

using both methods and varying the number of intervals logarithmically from 10 to 1000.

The analytical solution is $I = 2\arctan(10)$. Plot the difference of the numerical to the analytical solution $|I - I_{\text{num}}|$ for both methods as a function of the number of intervals in a double-logarithmic plot. Check if the results match your expectations from the formulas given for the error estimate.

Investigate the convergence of the two discussed methods further by calculating the specific heat of copper at $T = 100\,\text{K}$ and $T = 2000\,\text{K}$ with different numbers of sub-intervals. The number of intervals should be logarithmically distributed from 10 to about 10000.

★ Plot $|c_v - c_{v,\text{exact}}|$ as a function of the number of sub-intervals in a double-logarithmic graph, where $c_{v,\text{exact}}$ is the result from `integral`. Be careful to call `integral` with a reasonable accuracy argument. Use one graph per temperature.

Interpret your results and try to explain any differences you notice between methods and temperatures. When comparing the accuracy of the methods to the error estimation, you will notice something unusual. You are not required to explain this, but give it your best try.

A ★ task means that the task is optional and you can earn extra credit for it.