Integration

Numerical integration strongly depends on the particular problem at hand. Most ofen one uses adaptive mesh, sometimes the lowest order trapezoid rule and sometimes high order quadratures like Simpson rule.

There are few, sometimes competing factors one needs to consider

- speed - number of function evaluations or grid points
- precision - sometimes we need 10^{-6}, sometimes 1% is enough, but robustness is required
- robustness - adaptive or ”smart” meshes with lower order routines

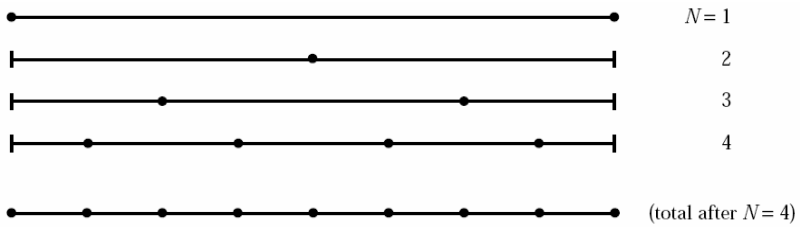
It is important to carefully examin the function to be integrated. Here are the points to be considered:

- How many dimensional integral? (more than 4 and low precision $\longrightarrow$ Monte Carlo)
- Is function very smooth or is not? ([high—low] order routine)
- Is there any divergency or cusp?
  - If divergency, can one change the variable of integration such that the divergency dissapears?
  - Can one locate the divergent point? (put additional points there - log mesh)
  - Can one locate possible cusp? (divide integral into two parts)
  - Can one subtract the divergency and calculate analytically the divergent part? (usually very efficient)
- Is function given on grid points or is given analytically? (easier to examine function if known analytically)

- Trapezoid rule:
  a) abcisas are non-equidistant
  b) function is not very smooth
  c) only moderate precision is required

$$\int_{x_1}^{x_2} f(x)dx = \frac{1}{2}(f_1 + f_2)(x_2 - x_1) + O(h^3) \tag{1}$$

$$\int_a^b f(x)dx = f_0\frac{1}{2}(x_1 - x_0) + \sum_{i=1}^{N-2} f_i\frac{1}{2}(x_{i+1} - x_{i-1}) + f_{N-1}\frac{1}{2}(x_{N-1} - x_{N-2}) + O(1/N^2) \tag{2}$$



If the input parameter is precision rather than number of points N, one can evaluate the above rule for N points and than split each interval into 2 to get $2N$ points, ...

$$S_1 = (b-a)\frac{1}{2}(f(a) + f(b)) \tag{3}$$

$$S_2 = \frac{1}{4}f(a) + \frac{1}{2}(b-a)f(\frac{a+b}{2}) + \frac{1}{4}(b-a)f(b) = \frac{1}{2}S_1 + \frac{1}{2}(b-a)f(\frac{a+b}{2}) \tag{4}$$

$$S_3 = \frac{1}{8}f(a) + \frac{1}{4}(b-a)f(a+h) + \cdots + \frac{1}{4}f(b-h) + \frac{1}{8}f(b) = \frac{1}{2}S_2 + \frac{h}{2}(f(a+\frac{h}{2}) + f(b-\frac{h}{2}))$$

$$h = \frac{1}{2}(b-a) \tag{5}$$

$$S_i = \frac{1}{2}S_{i-1} + \frac{1}{2}h_i \sum_{j=0}^{m-1} f(a + (\frac{1}{2}+j)h_i) + O(\frac{b-a}{(2m)^2}); \qquad m = 2^{i-2}; \qquad h_i = \frac{b-a}{m} \tag{6}$$

- Simpson's rule: $\int_0^{2h} f(x)dx = h(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{1}{3}f_2) + O(h^5)$

$$\int_a^b f(x)dx = h(\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \cdots + \frac{2}{3}f_{N-3} + \frac{4}{3}f_{N-2} + \frac{1}{3}f_{N-1}) + O(1/N^4)$$

$$S_i^{simpson} = \frac{4}{3}S_i - \frac{1}{3}S_{i-1} \tag{7}$$

$$S = S_1 + O(b-a)$$
$$S = S_2 + O(\frac{b-a}{2^2})$$
$$S = S_3 + O(\frac{b-a}{2^4})$$

- Another fourth order routine:

$$\int_a^b f(x)dx = h(\frac{17}{48}f_0 + \frac{59}{48}f_1 + \frac{43}{48}f_2 + \frac{49}{48}f_3 + f_4 + \cdots + f_{N-5} \tag{8}$$
$$+\frac{49}{48}f_{N-4} + \frac{43}{48}f_{N-3} + \frac{59}{48}f_{N-2} + \frac{17}{48}f_{N-1}) + O(1/N^4)$$

Implementation:

- Trapezoid for fixed number of points

```
template<class T, class functor>
T integrate_trapez_simple(functor& F, double a, double b, int N)
{// Trapezoid rule for fixed number of function evaluations
 // N at points a, a+(b-a)/(N-1),...
  double dh = (b-a)/(N-1);
  T sum = 0.5*F(a);
  double x = a+dh;
  for (int i=1; i<N-1; i++, x += dh) sum += F(x);
  sum += 0.5*F(x);
  return sum*dh;
}
```

- Trapezoid for given precision

```
template<class T, class functor>
T integrate_trapez(functor& F, double a, double b, double precision=1e-5, int max_recursion=22)
{
  T olds = 0, s = 0.5*(F(a)+F(b))*(b-a); // lowest order approximation
  int m=1;                          // number of subdivisions
  for (int i=0; i<max_recursion; i++, m*=2) { // m==2^(i-2)
    double dh = (b-a)/m;          // This is the distance between points to be added at this stage
    double x = a + 0.5*dh;        // first point added at
    T sum = 0;
    for (int j=0; j<m; j++, x+=dh) sum += F(x);
    s = 0.5*s + 0.5*dh*sum;     // The best approximation so far
    if (i > 4  &&  fabs(s-olds) <= precision*fabs(olds)){ //Avoid spurious early convergence.
      Logg("The integral step used =" << 0.5*dh << " with " << m << " divisions");
      return s;
    }
    olds=s;
  }
  std::cerr<<"Too many steps in routine integrate_trapez"<<std::endl;
  return s; // The best approximation
}
```

- Simpson for given precision

```
template<class T, class functor>
T integrate_simps(functor& F, double a, double b, double precision=1e-5, int max_recursion=22)
{
  T oldsm = 0, sm = 0; // simpson approximation
  T olds = 0, s = 0.5*(F(a)+F(b))*(b-a); // trapezoid approximation, lowest order approximation
  int m=1;                          // number of subdivisions
  for (int i=0; i<max_recursion; i++, m*=2) { // m==2^(i-2)
    double dh = (b-a)/m;          // This is the distance between points to be added at this stage
    double x = a + 0.5*dh;        // first point added at
    T sum = 0;
    for (int j=0; j<m; j++, x+=dh) sum += F(x);
    s = 0.5*s + 0.5*dh*sum;     // The best approximation so far
    sm = (4*s-olds)/3.;         // This is the first fifference with trapezoid
    if (i > 4  &&  fabs(sm-oldsm) <= precision*fabs(oldsm)){ //Avoid spurious early convergence.
      Logg("The integral step used ="<<0.5*dh<<" with "<<m<<" divisions");
      return sm;
    }
    olds=s;
    oldsm=sm;
  }
  std::cerr<<"Too many steps in routine integrate_simps"<<std::endl;
  return sm; // The best approximation
}
```

- Another fourth order routine for fixed number of points

```
template <class T, class functor>
T integrate4(functor& F, double a, double b, int Np)
{ // Forth order integration routine with equidistant mesh
  static double coeff[4] = {17./48., 59./48.,43./48.,49./48.};
  double dh = (b-a)/(Np-1), x = a;
  T sum = 0;
  for (int i=0; i<4; i++, x+=dh) sum += coeff[i]*F(x);
  for (int i=4; i<Np-4; i++, x+=dh) sum += F(x);
  for (int i=Np-4; i<Np; i++, x+=dh) sum += coeff[Np-1-i]*F(x);
  return sum*dh;
}
```

- Fully adaptive mesh for integration

  This part of the code first puts $Nmin$ number of equidistant points into list x. Then it recursively subdivides each interval as long as the difference between function values is larger than predefined precision.

```
template <class functor>
void mesh(list<double>& x, list<double>& f, const functor& fun, double a, double b, int N
{
  // This function constructs an adaptive mesh for function f.
  // The mesh is recursively subdivided until the difference between the function values
  // or there were more than Max_level subdivisions - recursion steps.
  // This function creates a list for mesh and a list of function values.
  for (int i=0; i<Nmin; i++) {
    double x0 = a + (b-a)*i/(Nmin-1.);
    x.push_back(x0);
    f.push_back(fun(x0));
  }

  list<double>::iterator xit = x.begin();
  list<double>::iterator fit = f.begin();

  vector<list<double>::iterator> xiter(x.size());
  vector<list<double>::iterator> fiter(x.size());
  for (int i=0; i<x.size(); i++){
    xiter[i] = xit;
    fiter[i] = fit;
    xit++;
    fit++;
```

```
  }

  for (int i=0; i<xiter.size()-1; i++){
    int level=0;
    recurs(x, f, fun, xiter[i], fiter[i], precision, Max_level, level);
  }
}
```

This part of the code recursively subdivides interval as long as the precision is not met, or, the interval is divided more than Maxlevel times.

```
template <class functor>
void recurs(list<double>& x, list<double>& f, const functor& fun, list<double>::iterator
  // Recursive algorithm inserts points into the mesh if the function values
  // differ more than precision
  // Recursion can not go deeper than Max_level
  list<double>::iterator fip = fit; fip++;
  list<double>::iterator xip = xit; xip++;
  if (fabs(*(fit)-*(fip))<precision || level>Max_level){
    return;
  } else{
    double x_new = 0.5*((*xit)+ (*xip));
    list<double>::iterator xin = x.insert(xip, x_new);
    list<double>::iterator fin = f.insert(fip, fun(x_new));

    recurs(x, f, fun, xit, fit, precision, Max_level, level+1);
    recurs(x, f, fun, xin, fin, precision, Max_level, level+1);
  }
}
```

Example: Calculation of the two dimensional tight-binding density of states with dispersion $\varepsilon_{\mathbf{k}} = -2t(\cos k_x + \cos k_y)$.

Analytic solution is

$$D(\omega) = \frac{1}{2t\pi^2|x|} K(1 - 1/x^2)$$

where K is eliptic integral of the first kind and $x = \omega/(4t)$ and logarithmically diverges at zero frequency.

Definition for the density of states is

$$D(\omega) = \int \frac{d^2k}{(2\pi)^2} \delta(\omega - \varepsilon_{\mathbf{k}}) = \int_{\omega=\varepsilon_{\mathbf{k}}} \frac{1}{(2\pi)^2} \frac{dl_{\mathbf{k}}}{|\nabla \varepsilon_{\mathbf{k}}|} \tag{9}$$

where the integral in this definition should be taken over the first Brillouin zone, i.e., $k_x \in [-\pi, \pi]$, $k_y \in [-\pi, \pi]$

The length of the segment of constant energy is

$$dl_{\mathbf{k}} = \sqrt{dk_x^2 + dk_y^2} = dk_x \sqrt{1 + \left(\frac{dk_y}{dk_x}\right)^2} \tag{10}$$

and taking derivative along the line of constant energy

$$\omega = -2t(\cos k_x + \cos k_y) \rightarrow 0 = \sin k_x dk_x + \sin k_y dk_y \tag{11}$$

we get

$$dl_{\mathbf{k}} = dk_x \sqrt{1 + \left(\frac{\sin k_x}{\sin k_y}\right)^2} \tag{12}$$

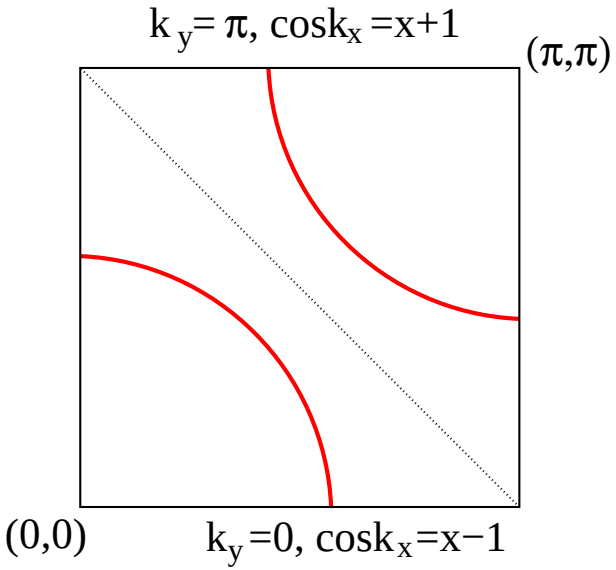
$$|\nabla \varepsilon_{\mathbf{k}}| = 2t\sqrt{\sin^2 k_x + \sin^2 k_y} \tag{13}$$

Finally

$$D(\omega) = \frac{1}{2t(2\pi)^2} \int_{\omega=\varepsilon_{\mathbf{k}}} \frac{dk_x}{|\sin k_y|} \tag{14}$$

Let's take new variable

$$x \equiv -\frac{\omega}{2t} = \cos k_x + \cos k_y$$



As shown in Figure above, we need to distinguish two cases

- $0 < x < 2$

$$D(x) = \frac{4}{2t(2\pi)^2} \int_0^{\arccos(x-1)} \frac{dk_x}{\sqrt{1-(x-\cos k_x)^2}}$$

- $-2 < x < 0$

$$D(x) = \frac{4}{2t(2\pi)^2} \int_{\arccos(x+1)}^{\pi} \frac{dk_x}{\sqrt{1-(x-\cos k_x)^2}}$$

Standart change of variables $\cos k_x = u$ gives

- $0 < x < 2$

$$D(x) = \frac{4}{2t(2\pi)^2} \int_{x-1}^{1} \frac{du}{\sqrt{1-u^2}\sqrt{1-(x-u)^2}}$$

- $-2 < x < 0$

$$D(x) = \frac{4}{2t(2\pi)^2} \int_{-1}^{x+1} \frac{du}{\sqrt{1-u^2}\sqrt{1-(x-u)^2}}$$

There are two poles in each integral: at lower and upper limit. How to remove poles

- Use non-equidistant mesh putting more points close to the divergency: tedious and not really necessary since we have analytic expression
- Subtract divergent term (see below)
- Change variables in a way to remove divergencies (see below)

If possible, it is much simpler to remove divergency by change of variables than by subtracting divergency. In this case, it is indeed possible. First we will divide integral such that each integral contains only one pole

$$\int_{x-1}^{1} \frac{du}{\sqrt{1-u^2}\sqrt{1-(x-u)^2}} = \int_{x-1}^{x/2} \cdots + \int_{x/2}^{1} \cdots \tag{15}$$

In the first integral we change variable to $v = \sqrt{1-(x-u)^2}$ and in the second $v = \sqrt{1-u^2}$. We see that both terms give equal contribution and the result is

$$2\int_{0}^{\sqrt{1-x^2/4}} \frac{dv}{\sqrt{1-v^2}\sqrt{1-(x-\sqrt{1-v^2})^2}} \tag{16}$$

Finally, density of states becomes

$$D(x) = \frac{1}{\pi^2 t}\int_{0}^{\sqrt{1-x^2/4}} \frac{dv}{\sqrt{1-v^2}\sqrt{1-(|x|-\sqrt{1-v^2})^2}} \tag{17}$$

The integral looks more complicated, however, it does not contain any pole except at $\omega = 0$ where the integral does not converge and the results is infinite (Density of states logarithmically diverges close to $\omega = 0$).

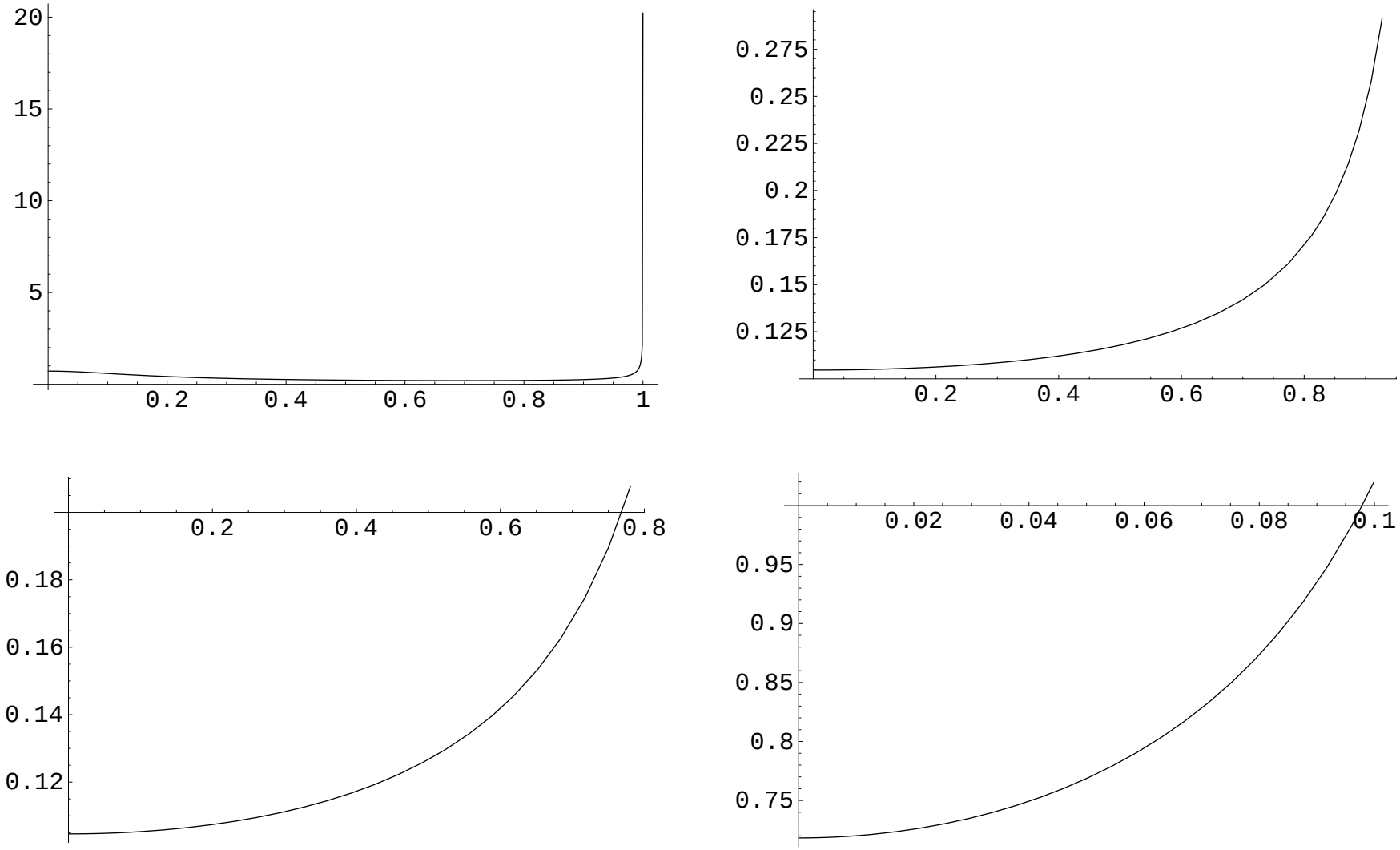


Figure 1: Change of variable - The function to be integrated for $x = 0.01$, $x = 0.75$, $x = 1.25$ and $x = 1.99$.

If the pole can not be removed by other means, we can always subtract the divergent part and treat it analytically.

Again, we will first divide the integral into two parts such that each part contains only one pole. By a change of variable $v = x - u$ in the first part, we again recognize that both parts give the same contribution to the density of states which becomes

$$D(x) = \frac{1}{\pi^2 t} \int_{x/2}^{1} \frac{du}{\sqrt{1-u^2}\sqrt{1-(x-u)^2}} \tag{18}$$

The pole is at $u = 1$ therefore we write $u = 1 - \epsilon$ to get

$$\int_0^{1-x/2} \frac{d\epsilon}{\sqrt{2\epsilon(1-\epsilon/2)2(x+\epsilon)(1-(x+\epsilon)/2)}} \tag{19}$$

and the expansion for small ϵ gives

$$\int_0^{1-x/2} \frac{d\epsilon}{2\sqrt{\epsilon(x+\epsilon)(1-x/2)}} = \frac{\log(\sqrt{1-x/2}+\sqrt{1+x/2}) - \log\sqrt{x}}{\sqrt{1-x/2}} \tag{20}$$

We kept ϵ in the second term. This term is second order in ϵ, but the expression is not exact to ϵ^2. The exact ϵ^2 expression contains spurious pole therefore we did not use it.

Finally, density of states can be calculated by

$$D(x) = \frac{1}{\pi^2 t}\int_{x/2}^{1} du \left[\frac{1}{\sqrt{1-u^2}\sqrt{1-(x-u)^2}} - \frac{1}{2\sqrt{(1-u)(1-u+x)(1-x/2)}}\right] + \frac{1}{\pi^2 t}\frac{\log(\sqrt{1-x/2}+\sqrt{1+x/2}) - \log\sqrt{x}}{\sqrt{1-x/2}} \quad (21)$$

The important point is that the function in brackets being integrated is very smooth and **not divergent**. It turns out that the slope of the integrating functions at $x = 0$ is still divergent (one could choose better function wich is more complicated to integrate for any x), but its integral is small compared to the divergent analytic term which logarithmically diverges just like the exact answer.

Density of states is even function, therefore the absolute value of x should be used above in general.

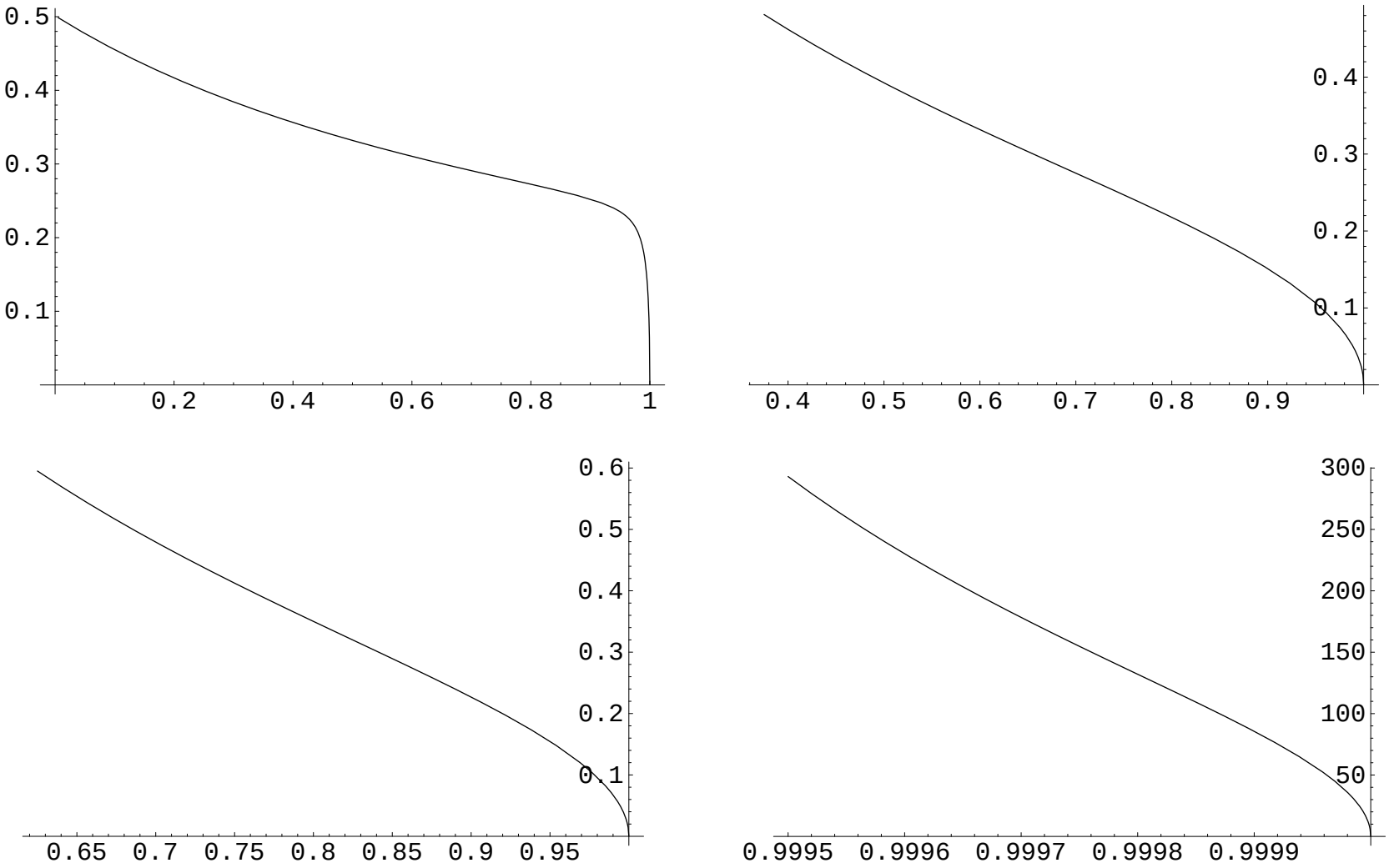


Figure 2: Subtracting divergency - The function to be integrated for $x = 0.01$, $x = 0.75$, $x = 1.25$ and $x = 1.99$.

Kramers Kronig

If the function is analytic in the upper complex plane as well as in the lower complex plane (as all response functions of physical system are) it must obey Kramars-Kronig relation

$$g'(y) = -P \int_{-\infty}^{\infty} \frac{dx}{\pi} \frac{g''(x)}{y-x}. \tag{22}$$

This is just the real-axis analog of more general Cauchy integral formula

$$g(z) = \frac{1}{2\pi i} \oint \frac{g(u)du}{u-z} \tag{23}$$

where path in the integral can be taken over the whole complex plane. Response functions are not analytic on real axis therefore we need to exclude real axis in the integral (The only function which is analytic everywhere is constant). If $g(z)$ falls off in infinity, we have

$$g(z) = \frac{1}{2\pi i} \int_{-\infty}^{\infty} \frac{[g(x+i\delta) - g(x-i\delta)]dx}{x-z} \tag{24}$$

which can be written as

$$g(z) = -\frac{1}{\pi} \int_{\infty}^{\infty} \frac{g''(x)dx}{z-x} \tag{25}$$

Calculation of the Kramars-Kronig transformation is pretty straightforward ones the subtraction of the divergent term is performed. Usually one implements Kramars-Kronig on finite grid between limits a and b such that imaginary part of g is negligable at a or b. In this case we have

$$g'(y) = \int_a^b \frac{dx}{\pi} \frac{g''(x) - g''(y)}{x - y} + \frac{g''(y)}{\pi} \mathrm{Re}[\log(b - y) - \log(a - y)] \quad (26)$$

$$= \int_a^b \frac{dx}{\pi} \frac{g''(x) - g''(y)}{x - y} + \frac{g''(y)}{\pi} \log\left(\frac{b - y}{y - a}\right) \quad (27)$$

The function being integrated is thus smooth everywhere. At point $x = y$ it should be calculated by estimating derivative $\frac{dg''(y)}{dy}$.

Multidimensional integrals

Not easy, task. Most important: Can one reduce dimensionality by analytic means?

Two methods are available for multidimensional integrals

- Use of one dimensional integration routines (best if one figures out good enough optimized non-equidistant mesh)
- Monte Carlo integration (possible only if high accuracy is not required and function is not strongly peaked)

When can one use one-dimensional routines for multidimensional integrals

- The shape of the boundary of the region of integration must be simple.
- Function can be strongly peaked, but one needs to know where peaks are.
- Time needed is simply given by N^M where N is number of points for one dimensional integral and M is dimension. For example, 3-dimensional integral with 100 point in each dimension needs 10^6 function evaluations.

Homeworks

1 Write the computer program which will compute the density of states $D(\omega)$ discussed above. Compute it by the change of variable and by subtracting the divergent term. Which technique is simpler and which is more precise? Which is faster?

2 Compute the Kramers-Kronig relation for this Density of states

$$g'(x) = \int d\omega \frac{D(\omega)}{x - \omega} \tag{28}$$

3 (optional): Code Hilbert transform of this density of states. Hilbert transform is defined by

$$w(z) = \int \frac{d\omega D(\omega)}{z - \omega} \tag{29}$$

for any complex z in the upper (or lower) half of the complex plane.
Plot hilbert transform for $w(x + 0.01i)$, $w(x + 0.2i)$ and $w(x + 0.5i)$.

4 (optional) You could improve performance of the Hilbert transform by calculating $D(\omega)$ on certain optimized (non-equidistant) mesh which is more dense at the divergencies or cusps. You can than implement Hilbert transform using the pre-calculated D. It is usually enough to calculate Hilbert transform up to 0.1% accuracy and therefore 400 points should be enough to resolve $D(\omega)$.