Computational Continuum Physics
Problem set 2

Spectral methods

Chalmers University of Technology

Viktor Lilja and Nico Guth

May 17, 2023

Problem 1: Galerkin method

See solution in appendix

Problem 2: Split-step Fourier solvers for TDSE

See solution in appendix

Problem 3: Spectral split-step solver for 2D TDSE

In this problem we study the evolution of the time dependent 2D Schrödinger equation

$$i\frac{\partial}{\partial t}\psi(x,y,t) = \left(-\nabla^2 + V(x,y)\right)\psi(x,y,t)$$

using a split-step spectral solver. The wave function is represented by samples on an M by M grid in real (x, y) and reciprocal $(k_x,\ k_y)$ space. The propagator for the problem is approximated by

$$P(t+\tau,t) = \exp\left(-iV(x,y)\frac{\tau}{2}\right)\boldsymbol{\mathcal{F}}^{-1}\exp\left(-ik^2\tau\right)\boldsymbol{\mathcal{F}}\exp\left(-iV(x,y)\frac{\tau}{2}\right) + \boldsymbol{\mathcal{O}}(\tau^3),$$

where $\boldsymbol{\mathcal{F}}$ denotes a (discrete) Fourier transform from real to reciprocal space. Because a finite representation in reciprocal space implicitly assumes real space periodicity, the solution obtained by this propagator will correspond to periodic boundary conditions.

We solved the problem for the potential

$$V(x,y) = \frac{-5}{(1+(x/5)^2+(y/4)^2)^4}$$

with initial state

$$\psi(x,y,0) = \frac{1}{\sqrt{\pi}}e^{-\left((x-1)^2+(y-1)^2\right)}.$$

in the domain $-10 \leq x, y \leq 10$ for $0 \leq t \leq 100$ and $0 \leq t \leq 1000$. Figure 1 shows the computed evolution in a short time interval.

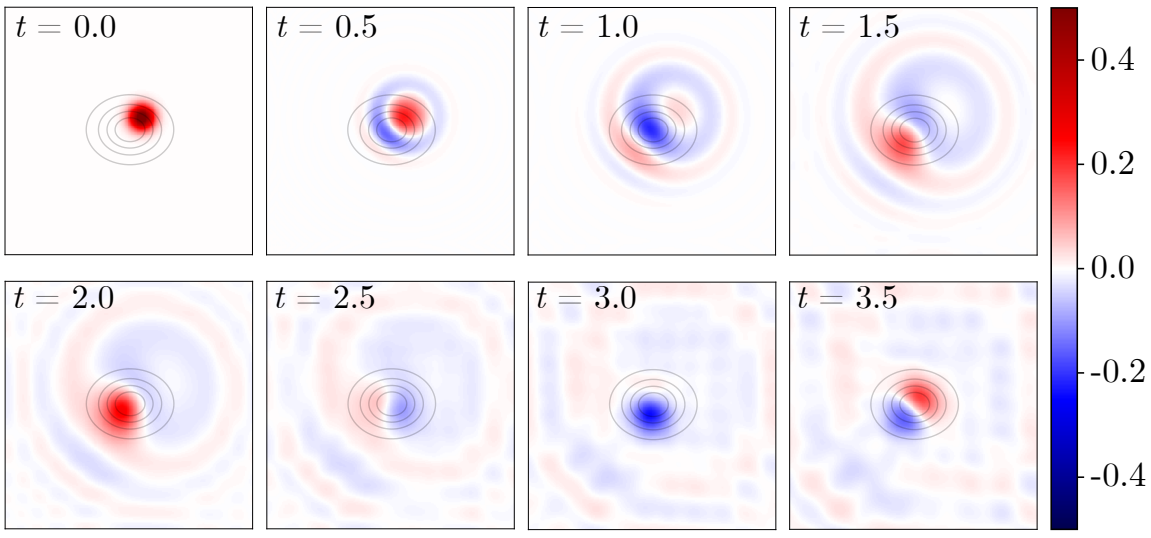


Figure 1: Real part of wave function in real space plotted at discrete time intervals. The red/blue colormap represents the value of the wave function. Thin gray lines are level curves of the potential well. As the initially Gaussian wave packet spreads out, part of it gets trapped in the potential well and part of it escapes. After $t = 2.0$ interference due to periodic boundary conditions becomes apparent.

Some simple physical considerations were used to select parameters for the simulation. The condition for the phase shift of one of the propagators to be small can be used to pick a time step. In our case $|V| \leq 5$ so we used $\tau = 0.1$ to ensure that $|V(x, y)\tau| < 1$. The grid spacing h has to be made small enough that the maximum wave number π/h in the reciprocal space captures the highest frequency component of the wave function. Classically, given that the initial state has zero kinetic energy, the maximum kinetic energy $k^2/2$ is the depth of the potential well V_0. We therefore estimate the largest wave number to be $\sqrt{2V_0}$ and hence the largest possible grid spacing $\pi/\sqrt{V_0} \approx 1$. We used $h = 0.1$.

The wave function at $(x, y) = (0.1, 0)$ was saved at each time step so that the temporal spectrum $\psi(0.1, 0, \omega)$ could be computed. The spectrum is presented in Figure 2. The physical interpretation of this spectrum can be understood by considering that the evolution of an eigenstate ψ_n of the Schrödinger equation is governed by

$$i\frac{\partial}{\partial t}\psi_n = E_n\psi_n,$$

which in the frequency domain is equivalent to

$$-\omega_n\psi_n = E_n\psi_n \quad \Leftrightarrow \quad \omega_n = -E_n.$$

Therefore, a peak in the spectrum at frequency ω corresponds to an energy level with energy $E = -\omega$. In our case, since the potential well has minimum at -5, there are no energy levels below -5 and hence no frequency peaks above 5. The lowest energy level (ground state) corresponds to the large peak at $\omega = 3.24$. The width of the peaks is an artefact of the finite time interval of the simulation; a longer simulation time results in thinner peaks.

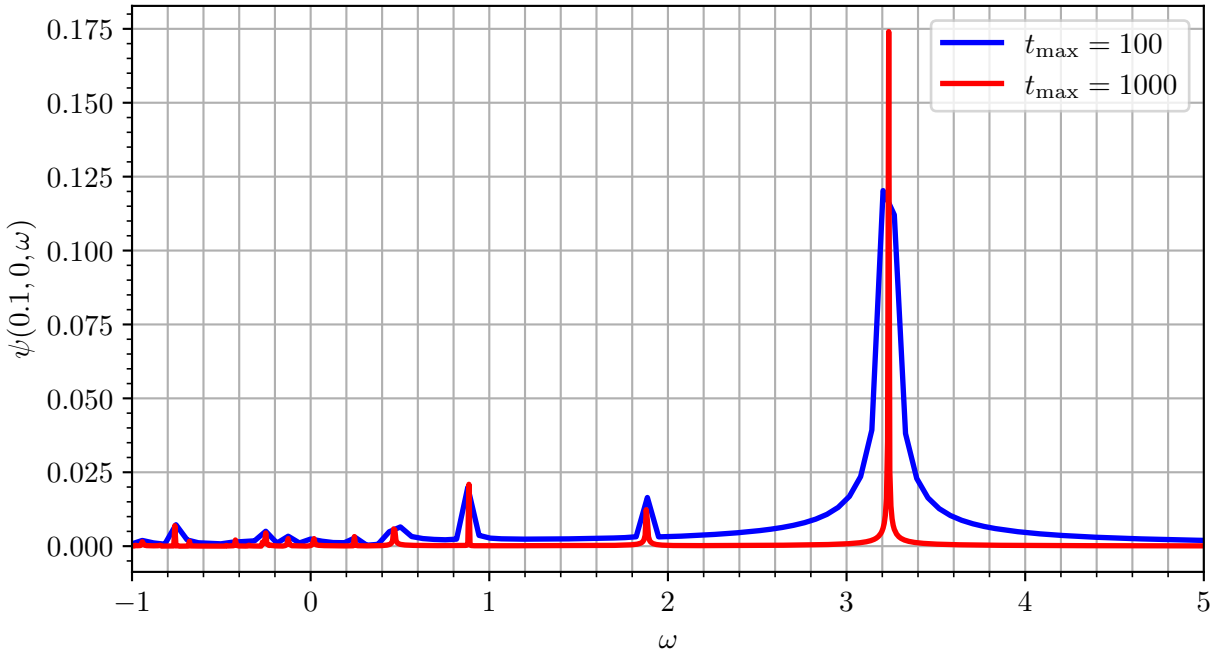


Figure 2: Temporal spectrum of the wave function evaluated at $(x, y) = (0.1, 0)$ using two different simulation lengths $t_{\max}$. The rightmost peak corresponds to the ground state, which has an energy of -3.24 units.

The accuracy of the solution was verified by also calculating the energy levels of a 2D harmonic oscillator

$$V(x, y) = \frac{x^2 + y^2}{4},$$

for which the energy levels are known analytically to be $E_n = n + 1$ for $n = 0, 1, 2, ...$ (in units where $m = 1/2, \hbar = 1, \omega = 1$). The resulting spectrum shown in Figure 3 shows that our numerical procedure can recreate the analytical result.

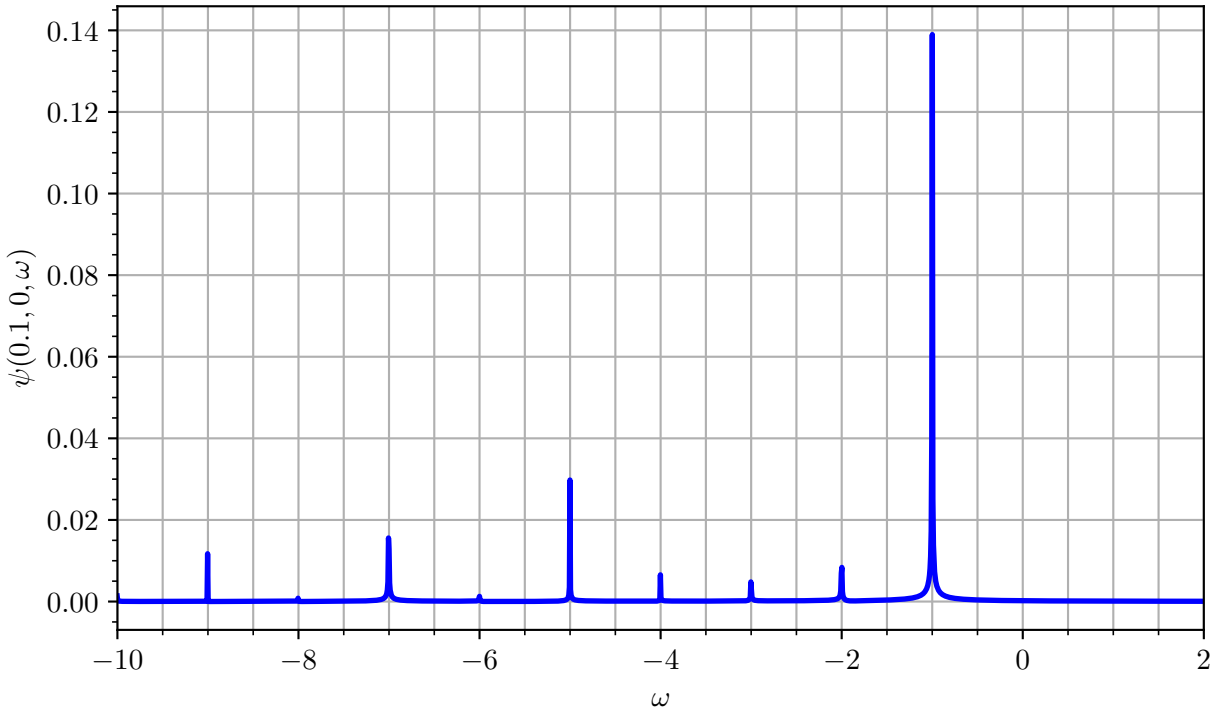


Figure 3: Temporal spectrum of the wave function evaluated at $(x, y) = (0.1, 0)$ using a harmonic oscillator test potential. The evenly spaced peaks correspond to energy levels $E = 1, 2, 3, ...$

Appendix: C++ code for problem 3

```cpp
#include <complex.h>
#include "fftw3.h"
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;

// complex number i = sqrt(-1)
const complex<double> i(0.0, 1.0);

// Given potential and wavefunction for task
double V(double x, double y)    {
    return - 5. / pow(1. + (x/5.)*(x/5.) + (y/4.)*(y/4.), 4.);
}

// Test with harmonic oscillator
//double V(double x, double y) {
//    return (x*x + y*y) / 4;
//}

complex<double> psi0(double x, double y) {
    return exp(-(x-1.)*(x-1.) - (y-1.)*(y-1.)) / sqrt(M_PI);
}


int main()
{
    // Grid resolution
    const int M = 200;

    // Real space grid
    double    L  = 10.0;          // Domain is -L < x < L,  -L < y < L
    double    dx = 2.0 * L / M;

    // Fourier space grid
    double    dk    = 2 * M_PI     / (2.0 * L);
    double    k_max = 2 * M_PI * M / (2.0 * L);

    // Time grid
    double t_min = 0;
    double t_max = 100;
    const int N  = 1000;
    double tau   = (t_max - t_min) / N;

    // For saving snapshots of wavefunction to plotting
    int frames = 10; // Number of frames excluding initial
    int frame_stride = N/frames;

    // Real space coordinates
    double x[M], y[M];
    for (int m=0; m<M; m++) {
        x[m] = -L + dx * m;
        y[m] = -L + dx * m;
```

```
    }

    // Fourier space coordinates
    double k_y[M], k_x[M];
    for (int m=0; m<M; m++) {
        if (m<M/2) {
            k_x[m] = dk * m;
            k_y[m] = dk * m;
        } else {
            k_x[m] = dk * (m-M);
            k_y[m] = dk * (m-M);
        }
    }

    // For saving value of wave function over time
    int m_x_0p1 = round((L+0.1)/(2*L) * M)-1;
    int m_y_0p0 = round((L+0.0)/(2*L) * M)-1;
    printf("Storing wavefunction at (x,y) = (%.5f,%.5f)\n",
            x[m_x_0p1], y[m_y_0p0]);
    complex<double> *psi_over_time     = new complex<double>[N];
    complex<double> *psi_time_spectrum = new complex<double>[N];

    // Create linspace for time and frequency
    double time[N], omega[N];
    for (int n=0; n<N; n++) {
        time[n] = t_min + tau * (n+1);
        omega[n] = (n<N/2) ? n     * (2. * M_PI) / (t_max - t_min) :
                             (n-N) * (2. * M_PI) / (t_max - t_min);
    }

    // Grids in real and fourier space with plans to transform between them
    auto psi_r = new complex<double>[M][M];
    auto psi_k = new complex<double>[M][M];
    fftw_plan transform_r_to_k = fftw_plan_dft_2d(M, M,
        reinterpret_cast<fftw_complex*>(&psi_r[0][0]),
        reinterpret_cast<fftw_complex*>(&psi_k[0][0]),
        FFTW_FORWARD, FFTW_ESTIMATE);
    fftw_plan transform_k_to_r = fftw_plan_dft_2d(M, M,
        reinterpret_cast<fftw_complex*>(&psi_k[0][0]),
        reinterpret_cast<fftw_complex*>(&psi_r[0][0]),
        FFTW_BACKWARD, FFTW_ESTIMATE);

    // initial wave function
    for (int m_x=0; m_x<M; m_x++)
    for (int m_y=0; m_y<M; m_y++)
        psi_r[m_x][m_y] = psi0(x[m_x], y[m_y]);

    // Save initial wavefunction
    char filename[100];
    sprintf(filename, "data/psi_r_real_n=%05d_t=%.2f.csv", 0, 0.0);
    FILE* f = fopen(filename, "w");
    for (int m_x = 0; m_x < M; m_x++)
    for (int m_y = 0; m_y < M; m_y++) {
        if (m_y != M-1) fprintf(f, "%.5e,",  real(psi_r[m_x][m_y]));
        else            fprintf(f, "%.5e\n", real(psi_r[m_x][m_y]));
    }
```

```
    fclose(f);

    // stepping through the simulation
    for (int n=1; n<N+1; n++) {

        // Propagate in space by tau/2
        for (int m_x=0; m_x<M; m_x++)
        for (int m_y=0; m_y<M; m_y++)
            psi_r[m_x][m_y] *= exp(-i * V(x[m_x], y[m_y]) * tau/2.);

        // Compute fourier transform
        fftw_execute(transform_r_to_k);

        // Propagate in fourier domain by tau
        for (int m_x=0; m_x<M; m_x++)
        for (int m_y=0; m_y<M; m_y++)
            psi_k[m_x][m_y] *= exp(-i * (k_x[m_x]*k_x[m_x] + k_y[m_y]*k_y[m_y])
    * tau);


        // Compute inverse Fourier transform
        fftw_execute(transform_k_to_r);

        // Propagate in space by tau/2
        for (int m_x=0; m_x<M; m_x++)
        for (int m_y=0; m_y<M; m_y++)
            psi_r[m_x][m_y] *= exp(-i * V(x[m_x], y[m_y]) * tau/2.);

        // Normalize (fftw forward then backward will scale by M*M)
        for (int m_x=0; m_x<M; m_x++)
        for (int m_y=0; m_y<M; m_y++)
            psi_r[m_x][m_y] /= (M*M);

        // Save updated wavefunction for plotting
        if (n % frame_stride == 0) {
            cout << "iteration " << n << "/" << N << endl;
            char filename[100];
            sprintf(filename, "data/psi_r_real_n=%05d_t=%.2f.csv", n, n * tau);
            FILE* f = fopen(filename, "w");
            for (int m_x = 0; m_x < M; m_x++)
            for (int m_y = 0; m_y < M; m_y++) {
                if (m_y != M-1) fprintf(f, "%.5e,",  real(psi_r[m_x][m_y]));
                else            fprintf(f, "%.5e\n", real(psi_r[m_x][m_y]));
            }
            fclose(f);
        }

        // Save value of wavefuntion at (x,y) = (0.1, 0)
        psi_over_time[n-1] = psi_r[m_x_0p1][m_y_0p0];
    }

    // Compute fourier transform in time
    printf("Fourier transforming\n");
    fftw_plan transform_t_to_omega = fftw_plan_dft_1d(N,
        reinterpret_cast<fftw_complex*>(&psi_over_time[0]),
        reinterpret_cast<fftw_complex*>(&psi_time_spectrum[0]),
```

```
        FFTW_FORWARD, FFTW_ESTIMATE);
    fftw_execute(transform_t_to_omega);
    fftw_destroy_plan(transform_t_to_omega);

    // Save wavefunction at a point and spectrum
    printf("Saving to file\n");
    sprintf(filename, "data/psi_time_and_frequency.csv");
    f = fopen(filename, "w");
    fprintf(f, "time, omega, abs(psi(t)), arg(psi(t)), abs(psi(omega)), arg(psi
    (omega))\n");
    for (int n = 0; n < N; n++) {
        fprintf(f, "%.5e,%.5e,%.5e,%.5e,%.5e,%.5e\n",
            time[n],
            omega[n],
            abs(psi_over_time[n]),
            arg(psi_over_time[n]),
            abs(psi_time_spectrum[n]),
            arg(psi_time_spectrum[n])
        );
    }
    fclose(f);

    fftw_destroy_plan(transform_r_to_k);
    fftw_destroy_plan(transform_k_to_r);

    return 0;
}
```

Appendix: Python code for problem 3

```
import numpy as np
import matplotlib.pyplot as plt

# For latex interpretation of the figures
plt.rcParams.update({
    "text.usetex": True,
    "font.family": "Computer Modern",
    "font.size": 11.0, # 11pt is fontsize of captions in the report
})

# Figure setup
fig = plt.figure(figsize=(6.6,4))  # textwidth in report is 6.6 inches
ax = fig.add_subplot(1,1,1)

# Iterate over data files
files = ["saved_data/psi_time_and_frequency_tmax=100_N=1000.csv",
        "saved_data/psi_time_and_frequency_tmax=1000_N=10000.csv"]
for i, filename in enumerate(files):

    # Extract data
    data = np.genfromtxt(filename, unpack=True, delimiter=",", skip_header=1)
    time, omega, abs_psi_t, arg_psi_t, abs_psi_omega, arg_psi_omega = data
    i_sorted = np.argsort(omega)

    # find the peak:
```

```
    idx_peak = np.argmax(abs_psi_omega)
    omega_peak = omega[idx_peak]
    print(f"Omega of the highest peak: {omega_peak:.5f}")

    # Add spectrum to plot
    ax.plot(omega[i_sorted], abs_psi_omega[i_sorted]/len(omega),
            ["b", "r"][i], lw=2,
            label="$t_\mathrm{max}=" + f"{round(time[-1])}$")

# Final figure formatting
ax.set_xlabel("$\omega$")
ax.set_ylabel("$\psi(0.1, 0, \omega)$")
ax.set_xlim(-1, 5)
ax.legend()
ax.grid(which="both", axis="x")
ax.grid(axis="y")
ax.minorticks_on()
fig.tight_layout()
fig.savefig("figures/spectrum.png")
fig.savefig("figures/spectrum.pdf")
```

Problem 2 of Problem set 2

20 April 2023 16:46

Problem 1. Galerkin method (5 points)

Apply Galerkin method to obtain function $f(x)$ in $0 \le x \le 1$ that satisfies the equation

$$\frac{\partial^2}{\partial x^2} f(x) = x + 1,$$

and the boundary conditions $f(0) = f(1) = 0$, using the basis functions

$$\chi_j(x) = x(1-x)^j.$$

Perform analytical computations and demonstrate that for $N = 2$ basis functions the resultant approximate solution converges to the exact solution.

Comment: Use scalar product in the form:

$$(v,u) = \int_0^1 v(x)u(x)dx.$$

problem: $\partial_x^2 f(x) = x+1$ with boundary conditions $f(0)=f(1)=0$ (I)

Find the exact solution to (I) analytically

$\partial_x^2 f = x+1 \Leftrightarrow \partial_x f = \frac{1}{2}x^2 + x + c_1 \Leftrightarrow f(x) = \frac{1}{6}x^3 + \frac{1}{2}x^2 + c_1 x + c_2$

$f(0) = 0 \Rightarrow c_2 = 0$, $f(1) = 0 \Leftrightarrow 0 = \frac{1}{6} + \frac{1}{2} + c_1 \Leftrightarrow c_1 = -\frac{2}{3}$

$\Rightarrow$ exact solution: $f(x) = \frac{1}{6}x^3 + \frac{1}{2}x^2 - \frac{2}{3}x$ (II)

---

approximate $\tilde{f}(x) = \sum_j w_j \chi_j(x)$ with $\chi_j(x) = x(1-x)^j$, $j \in \{1,2,\dots,N\}$

introduce the scalar product $(a(x), b(x)) = \int_0^1 a(x)\,b(x)\,dx$

$(\chi_k(x), (I)) \Rightarrow \sum_{j=1}^{N} w_j \underbrace{(\chi_k(x), \partial_x^2 \chi_j(x))}_{(*)} = \underbrace{(\chi_k(x), (x+1))}_{(**)}$ (III) $\Leftrightarrow M\vec{w} = \vec{s}$

evaluate (*) and (**) seperately

(*) $\partial_x^2 \chi_j(x) = \partial_x\left(\partial_x\left(x(1-x)^j\right)\right) = \partial_x\left((1-x)^j - jx(1-x)^{j-1}\right)$

$= -j(1-x)^{j-1} - j(1-x)^{j-1} + j(j-1)x(1-x)^{j-2}$

$= -2j(1-x)^{j-1} + j(j-1)x(1-x)^{j-2}$

$\Rightarrow (\chi_k, \partial_x^2 \chi_j) = \int_0^1 x(1-x)^k\left[j(j-1)x(1-x)^{j-2} - 2j(1-x)^{j-1}\right]dx$

$= \int_0^1 \left[j(j-1)x^2(1-x)^{k+j-2} - 2jx(1-x)^{k+j-1}\right]dx$ (f, g' / f, g')

use integration by parts $\int fg' = [fg] - \int f'g$ and $\int (1-x)^n dx = \left[-\frac{1}{n+1}(1-x)^{n+1}\right]$

$= j(j-1)\underbrace{\left[x^2\left(\frac{-1}{k+j-1}\right)(1-x)^{k+j-1}\right]_0^1}_{=0} - j(j-1)\int_0^1 2x\left(\frac{-1}{k+j-1}\right)(1-x)^{k+j-1}dx$ (f, g')

$-2j\underbrace{\left[x\left(\frac{-1}{k+j}\right)(1-x)^{k+j}\right]_0^1}_{=0} + 2j\int_0^1 \left(\frac{-1}{k+j}\right)(1-x)^{k+j}dx$ use integration by parts again

$= 2j(j-1)\left(\frac{1}{k+j-1}\right)\underbrace{\left[x\left(\frac{-1}{k+j}\right)(1-x)^{k+j}\right]_0^1}_{=0} - 2j(j-1)\left(\frac{1}{k+j-1}\right)\int_0^1 \left(\frac{-1}{k+j}\right)(1-x)^{k+j}dx - 2j\left(\frac{1}{k+j}\right)\int (1-x)^{k+j}dx$

$$= 2j(j-1)\left(\frac{\cdot}{k+j-1}\right)\underbrace{\left[x\left(\frac{-1}{k+j}\right)(1-x)^{k+j}\right]_0}_{=0} - 2j(j-1)\left(\frac{\cdot}{k+j-1}\right)\int\left(\frac{\cdot}{k+j}\right)(1-x)^{k+j}dx - 2j\left(\frac{\cdot}{k+j}\right)\int(1-x)^{k+j}dx$$

$$= 2j\left(\frac{1}{k+j}\right)\left(\frac{j-1}{k+j-1} - 1\right)\left[\left(\frac{-1}{k+j+1}\right)\underbrace{(1-x)^{k+j+1}}_{=\begin{cases}0 \text{ for } x=1\\ 1 \text{ for } x=0\end{cases}}\right]_0^1$$

$$\boxed{(*) = \left(\chi_k, \partial_x^2 \chi_j\right) = \left(\frac{2j}{k+j}\right)\left(\frac{j-1}{k+j-1} - 1\right)\left(\frac{1}{k+j+1}\right)} \quad (\text{IV})$$

(**)

$$\left(\chi_k, (x+1)\right) = \int_0^1 x(1-x)^k(1+x)\,dx = \int_0^1 \underbrace{(x^2+x)}_{f}\underbrace{(1-x)^k}_{g'}dx \quad \text{use integration by parts}$$

$$= \underbrace{\left[(x^2+x)\left(\frac{-1}{k+1}\right)(1-x)^{k+1}\right]_0^1}_{=0} - \int_0^1 \underbrace{(2x+1)}_{f}\underbrace{\left(\frac{-1}{k+1}\right)(1-x)^{k+1}}_{g'}dx$$

$$= \frac{1}{k+1}\left[\underbrace{(2x+1)}_{=0}\left(\frac{-1}{k+2}\right)(1-x)^{k+2}\right]_0^1 - \frac{1}{k+1}\int_0^1 2\left(\frac{-1}{k+2}\right)(1-x)^{k+2}dx$$

$$= \frac{1}{k+1}\left(\frac{-1}{k+2}\right)\underbrace{\left[(1-x)^{k+2}\right]_1^1}_{=-1} + \frac{2}{k+1}\left(\frac{1}{k+2}\right)\left(\frac{-1}{k+3}\right)\underbrace{\left[(1-x)^{k+3}\right]_0^1}_{=-1}$$

$$\boxed{(**) = \left(\chi_k, (x+1)\right) = \frac{1}{(k+1)(k+2)} + \frac{2}{(k+1)(k+2)(k+3)}} \quad (\text{V})$$

Now check which solution the Galerkin method with $N=2$ basis functions yields.

(IV) $\Rightarrow$ $\left(\chi_1, \partial_x^2\chi_1\right) = -\frac{1}{3}$, $\left(\chi_1, \partial_x^2\chi_2\right) = -\frac{1}{6}$, $\left(\chi_2, \partial_x^2\chi_1\right) = -\frac{1}{6}$, $\left(\chi_2, \partial_x^2\chi_2\right) = -\frac{2}{15}$

(V) $\Rightarrow$ $\left(\chi_1, (x+1)\right) = \frac{1}{4}$, $\left(\chi_2, (x+1)\right) = \frac{7}{60}$

(III) $\Leftrightarrow$ $M\vec{w} = \vec{s}$ with $\vec{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$, $M = \begin{pmatrix} -\frac{1}{3} & -\frac{1}{6} \\ -\frac{1}{6} & -\frac{2}{15} \end{pmatrix}$, $\vec{s} = \begin{pmatrix} \frac{1}{4} \\ \frac{7}{60} \end{pmatrix}$

inverse of 2D matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow A^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

$$\Rightarrow \det(M) = \left(-\frac{1}{3}\right)\left(-\frac{2}{15}\right) - \left(-\frac{1}{6}\right)\left(-\frac{1}{6}\right) = \frac{1}{60} \quad \Rightarrow M^{-1} = 60\begin{pmatrix} -\frac{2}{15} & \frac{1}{6} \\ \frac{1}{6} & -\frac{1}{3} \end{pmatrix} = \begin{pmatrix} -8 & 10 \\ 10 & -20 \end{pmatrix}$$

$$\Rightarrow \vec{w} = M^{-1}\vec{s} = \begin{pmatrix} -8 & 10 \\ 10 & -20 \end{pmatrix}\begin{pmatrix} \frac{1}{4} \\ \frac{7}{60} \end{pmatrix} = \begin{pmatrix} -\frac{5}{6} \\ \frac{1}{6} \end{pmatrix}$$

$$\Rightarrow \tilde{f}(x) = w_1\chi_1 + w_2\chi_2 = -\frac{5}{6}x(1-x) + \frac{1}{6}x(1-x)^2$$

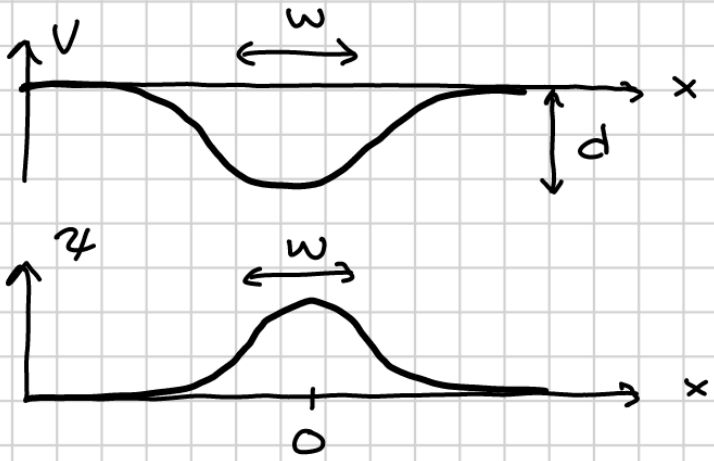
$$= -\frac{5}{6}(x-x^2) + \frac{1}{6}(x-2x^2+x^3)$$

$$\boxed{\tilde{f}(x) = \frac{1}{6}x^3 + \frac{1}{2}x^2 - \frac{1}{3}x \overset{(II)}{=} f(x)}$$

$\Rightarrow$ for $N=2$ this Galerkin method yields the exact solution of (I).

PROBLEM 2

a) In this case the main part of the wavefunction stays localized in the potential, so its width is rougly w.



The magnitudes of the potentials at the wavefunction are rougly

$$|U_i| = \frac{d}{\sqrt{x^2+y^2+z^2+w^2}} \lesssim \frac{d}{w}$$

$$|U_f| \lesssim a_{max}\, x \lesssim \{x \lesssim w\} \simeq a_{max}\, w$$

Thus, the condition $e^{i(U_i+U_f)\tau} \simeq 1$ restricts the standard method to

$$\tau \lesssim \frac{1}{d/w + a_{max}\, w} = \frac{w}{d + a_{max} w^2} \underset{\uparrow}{\simeq} \frac{1}{a_{max}\, w}$$

$d \ll a_{max} w^2$

Optimization in case a)

Use transformation from hint:

$$\psi(\vec{r},t) = e^{i\alpha(t)x}\phi(\vec{r},t)$$

Insert into Schrödinger equation:

$$\frac{\partial}{\partial t}\left(e^{i\alpha(t)x}\phi\right) = \left[i\nabla^2 - i(U_i+U_f)\right]e^{i\alpha(t)x}\phi$$

$$\frac{\partial}{\partial t}(e^{i\alpha x}\phi) = e^{i\alpha x}\left(ix\dot{\alpha}\phi + \dot{\phi}\right)$$

$$\nabla^2(e^{i\alpha x}\phi) = \frac{\partial^2}{\partial x^2}(e^{i\alpha x}\phi) + e^{i\alpha x}\left(\frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}\right) =$$

$$= \left\{\frac{\partial^2}{\partial x^2}(e^{i\alpha x}\phi) = \frac{\partial}{\partial x}\left(e^{i\alpha x} i\alpha\phi + e^{i\alpha x}\phi_x'\right) =\right.$$

$$\left. = \left(-\alpha^2 e^{i\alpha x}\phi + 2i\alpha e^{i\alpha x}\phi_x' + e^{i\alpha x}\phi_{xx}''\right)\right\} =$$

$$= e^{i\alpha x}\left(-\alpha^2\phi + 2i\alpha\frac{\partial\phi}{\partial x} + \nabla^2\phi\right)$$

$$\Rightarrow ix\dot{\alpha}\phi + \dot{\phi} = \left[i\left(-\alpha^2 + 2i\alpha\frac{\partial}{\partial x} + \nabla^2\right) - i(U_i+U_f)\right]\phi$$

This gives us a transformed equation

$$\dot{\phi} = \Big[\underbrace{\left(-i\alpha^2 - 2\alpha\frac{\partial}{\partial x} + i\nabla^2\right)}_{B(t)} - i\underbrace{\left(U_i + U_f + \dot{\alpha}x\right)}_{A(t)}\Big]\phi$$

Find propagator for A:

$$\dot{\phi} = -i(U_i + U_1 + \dot{\alpha}x)\phi$$

$$\Rightarrow \phi(t+\tau) = \underbrace{e^{-i(U_i+U_1+\dot{\alpha}x)\tau}}_{e^{\tau A(t)}}\phi(t)$$

Find propagator for new B:

$$\dot{\phi} = \left(-i\alpha^2 - 2\alpha\frac{\partial}{\partial x} + i\nabla^2\right)\phi$$

$$\Rightarrow \quad \dot{\tilde{\phi}} = \left(-i\alpha^2 - 2i\alpha k_x - ik^2\right)\tilde{\phi}$$

$$\tilde{\phi}(t+\tau) = e^{\left(-i\alpha^2 - 2i\alpha k_x - ik^2\right)\tau}\tilde{\phi}(t)$$

$$\phi(t+\tau) = \underbrace{\mathcal{F}^{-1}e^{-i(\alpha^2+2\alpha k_x+k^2)\tau}\mathcal{F}}_{e^{\tau B(t)}}\phi(t)$$

The full propagator for ϕ, is

$$P(t+\tau, t) = e^{\tau A(t)}e^{\tau B(t)} + \mathcal{O}(\tau^2) =$$

$$\boxed{= e^{-iU_i\tau}\mathcal{F}^{-1}e^{-i(\alpha(t)^2+2\alpha(t)k_x+k^2)\tau}\mathcal{F} + \mathcal{O}(\tau^2)}$$

Once we have ϕ we can compute ψ from $\psi = e^{i\alpha x}\phi$. Symmetric form of P can be used for $\mathcal{O}(\tau^3)$.

Using this propagator, the new timestep restriction is

$$\tau \leq \frac{1}{v_i + v_t + \dot{\alpha} x}$$

If we now pick

$$\dot{\alpha}(t) = -a(t) = -a_{max} e^{-\frac{(t-3D)^2}{D^2}} \sin(\omega t)$$

$$\Rightarrow \boxed{\alpha(t) = -a_{max} \int^t e^{-\frac{(t'-3D)^2}{D^2}} \sin(\omega t') dt'}$$

$\alpha(t)$ can be evaluated numerically analytically (see integrals at the end of the solution).

then $\dot{\alpha} x = -a(t) x = -v_t$, so

$$\tau \leq \frac{1}{v_i} \approx \frac{w}{d}$$

This is much better because

old τ → $\frac{1}{a_{max} w}$ $\ll$ $\frac{w}{d}$ ← new τ,

$$d \ll a_{max} w^2 \Leftrightarrow \frac{1}{a_{max} w} \ll \frac{w}{d},$$

so we can take larger steps!

Summary for case a)

1. Compute $\phi_0 = e^{-i\alpha(0)x}\psi_0$

2. Propagate using

$$\phi(t+\tau) = \underbrace{e^{-iV_i\tau}}_{\text{diagonal in } r\text{-space}} \mathcal{F}^{-1} \underbrace{e^{-i(\alpha(t)^2 + 2\alpha(t)k_x + k^2)\tau}}_{\text{diagonal in } k\text{-space}} \mathcal{F}\,\phi(t)$$

with $\tau \leq$ w/d until desired t.

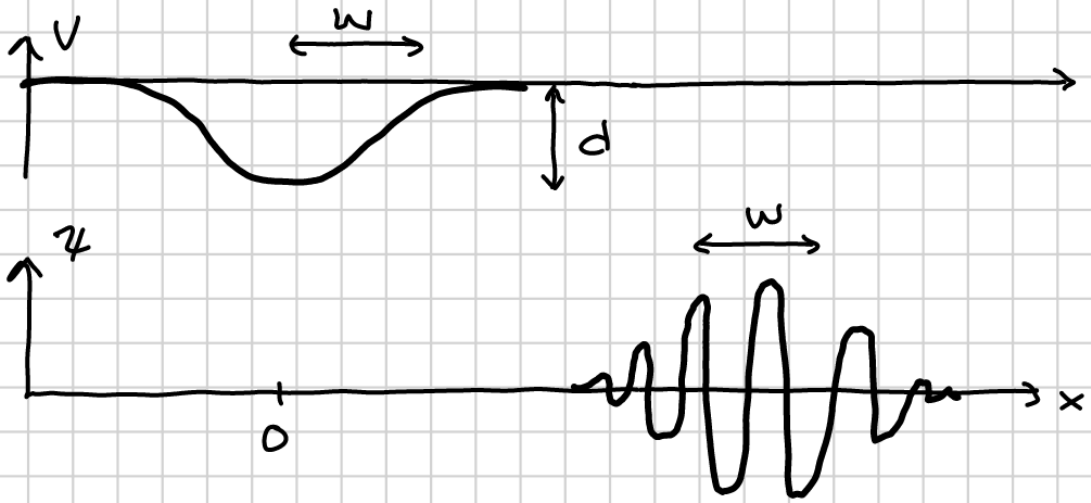
3. Compute $\psi(t) = e^{i\alpha(t)x}\phi(t)$.

$\alpha(t)$ is given analytically by

$$\alpha(t) = -\frac{\sqrt{\pi}\, a_{max} D}{4i} \sum_{\sigma} \sigma e^{A_\sigma^2 - 9} \operatorname{erf}\left(\frac{t}{D} - A_\sigma\right)$$

where $\sigma = \{1, -1\}$, $A_\sigma = 3 + \frac{\sigma i D\omega}{2}$.

b) In this case the wavefunction no longer stays localized in the potential.



This is problematic because the simulation domain has to be very large to contain the wavefunction.

Optimization for case b)

Transform with

$$\psi(t) = e^{\alpha(t)\frac{\partial}{\partial x}}\phi(t)$$

Insert in TDSE:

$$\frac{\partial}{\partial t}\left(e^{\alpha\frac{\partial}{\partial x}}\phi\right) = \left[i\nabla^2 - i(U_i+U_\ell)\right]e^{\alpha\frac{\partial}{\partial x}}\phi$$

$$e^{\alpha\frac{\partial}{\partial x}}\left(\dot{\alpha}\frac{\partial}{\partial x}\phi + \dot{\phi}\right) = \left[i\nabla^2 - i(U_i+U_\ell)\right]e^{\alpha\frac{\partial}{\partial x}}\phi$$

Note that $e^{\alpha\frac{\partial}{\partial x}}$ does not commute U_i and U_ℓ which depend on x:

$$\dot{\phi} = \underbrace{\left[i\nabla^2 - \dot{\alpha}\frac{\partial}{\partial x}\right]}_{A(t)} \underbrace{- ie^{-\alpha\frac{\partial}{\partial x}}(U_i+U_\ell)e^{\alpha\frac{\partial}{\partial x}}}_{B(t)}\phi$$

Find propagator for A:

$$\dot{\phi} = \left[i\nabla^2 - \dot{\alpha}\frac{\partial}{\partial x}\right]\phi \qquad \left(ik_x \leftrightarrow \frac{\partial}{\partial x}\right)$$

$$\dot{\tilde{\phi}} = \left[-ik^2 - i\dot{\alpha}k_x\right]\tilde{\phi}$$

$$\tilde{\phi}(t+\tau) = e^{-i(k^2+\dot{\alpha}k_x)t}\tilde{\phi}(t)$$

$$\phi(t+\tau) = \underbrace{\mathcal{F}^{-1}e^{-i(k^2+\dot{\alpha}k_x)t}\mathcal{F}}_{P_A(t,0)}\phi(t)$$

Find propagator for B:

$$\dot{\phi} = e^{-\alpha \frac{\partial}{\partial x}} (U_i + U_t) e^{\alpha \frac{\partial}{\partial x}} \phi$$

$$e^{\alpha \frac{\partial}{\partial x}} \dot{\phi} = (U_i + U_t) e^{\alpha \frac{\partial}{\partial x}} \phi \qquad (*)$$

What is the effect of $e^{\alpha \frac{\partial}{\partial x}}$ in the position basis?

$$\left(e^{\alpha \frac{\partial}{\partial x}} \psi\right)(\vec{r}) = \langle \vec{r} | e^{\alpha \frac{\partial}{\partial x}} | \psi \rangle =$$

$\frac{\partial}{\partial x} \longleftrightarrow i\hat{k}_x$

$$= \int d\vec{k} \, \langle \vec{r} | e^{i\alpha \hat{k}_x} | \vec{k} \rangle \langle \vec{k} | \psi \rangle =$$

$\langle \vec{r} | \vec{k} \rangle = e^{i\vec{k} \cdot \vec{r}}$

$$= \int d\vec{k} \, e^{i\alpha k_x} e^{i\vec{k} \cdot \vec{r}} \langle \vec{k} | \psi \rangle =$$

$$= \int d\vec{k} \, e^{i(k_x(x+\alpha) + k_y y + k_z z)} \langle \vec{k} | \psi \rangle =$$

$$= \int d\vec{k} \, e^{i\vec{k} \cdot (x+\alpha, y, z)} \langle \vec{k} | \psi \rangle =$$

$$= \int d\vec{k} \, \langle \vec{r} + \alpha \vec{1}_x | \vec{k} \rangle \langle \vec{k} | \psi \rangle =$$

$$= \langle \vec{r} + \alpha \vec{1}_x | \psi \rangle = \psi(\vec{r} + \alpha \vec{1}_x)$$

Conclusion:

$e^{\alpha \frac{\partial}{\partial x}}$ is a translation by $-\alpha$ in x.

Using this we can evaluate P_B in the position basis:

$$(*): \quad e^{\alpha \frac{\partial}{\partial x}} \dot{\phi} = -i(U_i + U_l)\, e^{\alpha \frac{\partial}{\partial x}} \phi$$

$$\Leftrightarrow \quad \dot{\phi}(\vec{r} + \alpha \vec{1}_x) = -i[U(\vec{r}) + U_l(\vec{r})]\phi(\vec{r} + \alpha \vec{1}_x)$$

$$\Leftrightarrow \quad \dot{\phi}(\vec{r}) = -i[U(\vec{r} - \alpha \vec{1}_x) + U_l(\vec{r} - \alpha \vec{1}_x)]\phi(\vec{r})$$

$$\Rightarrow \phi(\vec{r}, t+\tau) = \underbrace{e^{-i[U_i(\vec{r} - \alpha \vec{1}_x) + U_l(\vec{r} - \alpha \vec{1}_x)]t}}_{P_B(t+\tau, t)} \phi(\vec{r}, t)$$

The wavefunction at x is affected by the potential at $x - \alpha$.

The full propagator (for ϕ) is

$$\boxed{P(t+\tau, t) = e^{-i(U_i + U_l)|_{x-\alpha}\, \tau} \mathcal{F}^{-1} e^{-i(k^2 + \dot{\alpha} k_x)\tau} \mathcal{F} + \mathcal{O}(\tau^2)}$$

and ψ is given by

$$\psi(\vec{r}, t) = e^{\alpha \frac{\partial}{\partial x}} \phi(r, t) = \phi(\vec{r} + \alpha \vec{1}_x, t)$$

$$\Leftrightarrow \quad \boxed{\psi(\vec{r} - \alpha(t) \vec{1}_x) = \phi(\vec{r}, t)}$$

The idea is now to pick $\alpha(t)$ so that ϕ stays around $x = 0$ to avoid the need of a large simulation domain.

Approximating the motion classically, the center $\vec{r}_c$ of ψ should move according to Newton's second law:

$$\ddot{\vec{r}}_c = -\vec{\nabla} U \approx -\vec{\nabla} U_\ell = -a(t)\,\vec{1}_x$$

$$\Rightarrow \ddot{x}_c(t) = -a(t)$$

Since $\psi(\vec{r} - \alpha(t)\vec{1}_x) = \phi(\vec{r}, t)$, we should pick $\alpha(t) = -x_c(t)$ for $\psi(\vec{r}_c, t) = \phi(0, t)$, e.g. ϕ stays centered. Thus,

$$\boxed{\ddot{\alpha}(t) = a(t) = a_{max} e^{-\frac{(t-3D)^2}{D^2}} \sin(\omega t)}$$

$\alpha(t)$ can be computed numerically or analytically (see integrals at the end of the solution)

Summary for case b)

1. Compute $\phi_0(\vec{r}) = \psi_0(\vec{r} - \alpha(0)\vec{1}_x)$

2. Propagate using

$$\phi(t+\tau) = \underbrace{e^{-i(U_i+U_l)|_{x-\alpha}\tau}}_{\text{diagonal in } r\text{-space}} \mathcal{F}^{-1} \underbrace{e^{-i(k^2+\dot{\alpha}k_x)\tau}}_{\text{diagonal in } k\text{-space}} \mathcal{F}\phi(t)$$

on domain of width $\geq w$ until desired t.

3. Compute $\psi(\vec{r}, t) = \phi(\vec{r} + \alpha(t)\vec{1}_x, t)$

$\alpha(t)$ is given analytically by

$$\alpha(t) = \frac{\sqrt{\pi}\, a_{max} D^2}{4i} \sum_{\sigma} \sigma e^{A_\sigma^2 - 9} \times$$

$$\times \left(\frac{t}{D} - A_\sigma\right) \operatorname{erf}\left(\frac{t}{D} - A_\sigma\right) + \frac{1}{\sqrt{\pi}} e^{-\left(\frac{t}{D} - A_\sigma\right)^2}$$

where $\sigma = \{1, -1\}$, $A_\sigma = 3 + \frac{\sigma i D \omega}{2}$.

Evaluating integrals

We have

$$a(t) = a_{max} e^{-\frac{(t-3D)^2}{D^2}} \sin(\omega t)$$

$$= a_{max} e^{-\left(\frac{t^2}{D^2} - \frac{6t}{D} + 9\right)} \frac{e^{i\omega t} - e^{-i\omega t}}{2i} =$$

$$= \frac{a_{max}}{2i} \sum_{\sigma} \sigma e^{-\left(\frac{t^2}{D^2} - \left(\frac{6}{D} + \sigma i\omega\right)t + 9\right)} =$$

Where $\sigma = 1, -1$. Complete the square:

$$\frac{t^2}{D^2} - \left(\frac{6}{D} + \sigma i\omega\right)t + 9 =$$

$$= \left[\frac{t}{D} - \frac{D}{2}\left(\frac{6}{D} + \sigma i\omega\right)\right]^2 - \left[\frac{D}{2}\left(\frac{6}{D} + \sigma i\omega\right)\right]^2 + 9 =$$

$$= \left[\frac{t}{D} - \left(3 + \frac{\sigma i D\omega}{2}\right)\right]^2 - \left(3 + \frac{\sigma i D\omega}{2}\right)^2 + 9 =$$

$$\Rightarrow a(t) = \frac{a_{max}}{2i} \sum_{\sigma} e^{A_\sigma^2 - 9} \sigma e^{-\left(\frac{t}{D} - A_\sigma\right)^2}$$

where $A_\sigma = 3 + \frac{\sigma i D\omega}{2}$.

For case a) we need

$$\dot{\alpha}_a(t) = -a(t) \Rightarrow \alpha_a(t) = -\int^t a(t')\,dt'$$

Note that

$$\int^t e^{-\left(\frac{t'}{D} - A_\sigma\right)^2} dt' = \left\{ \begin{array}{l} u = t'/D - A_\sigma \\ dt' = D\,du \end{array} \right\} =$$

$$= D\int^{\frac{t}{D} - A_\sigma} e^{-u^2} du = D\,\frac{\sqrt{\pi}}{2}\,\mathrm{erf}\left(\frac{t}{D} - A_\sigma\right) (+C)$$

So

$$\alpha_a(t) = -\int^t a(t')\,dt' =$$

$$= -\frac{a_{max}}{2i} \sum_\sigma \sigma\, e^{A_\sigma^2 - 9} \left(D\,\frac{\sqrt{\pi}}{2}\,\mathrm{erf}\left(\frac{t}{D} - A_\sigma\right)\right)$$

$$\boxed{\alpha_a(t) = -\frac{\sqrt{\pi}\, a_{max} D}{4i} \sum_\sigma \sigma\, e^{A_\sigma^2 - 9}\, \mathrm{erf}\left(\frac{t}{D} - A_\sigma\right)}$$

where $\sigma = \{1, -1\}$, $A_\sigma = 3 + \frac{\sigma i D \omega}{2}$

For case b) we need

$$\ddot{\alpha}_b(t) = a(t) = -\ddot{\alpha}_a(t)$$

$$\Rightarrow \quad \alpha_b(t) = -\int^t \alpha_a(t')dt'$$

Note that

$$\int^t \mathrm{erf}\left(\frac{t'}{D} - A_\sigma\right)dt' = \left\{\begin{array}{l} u = t'/D - A_\sigma \\ dt' = D\,du \end{array}\right\} =$$

$$= D\int^{\frac{t}{D}-A_\sigma} \mathrm{erf}(u)\,du = D\left[u\,\mathrm{erf}(u) + \frac{e^{-u^2}}{\sqrt{\pi}}\right]^{\frac{t}{D}-A_\sigma}$$

$$= D\left[\left(\frac{t}{D} - A_\sigma\right)\mathrm{erf}\left(\frac{t}{D} - A_\sigma\right) + \frac{1}{\sqrt{\pi}}\, e^{-\left(\frac{t}{D}-A_\sigma\right)^2}\right] \quad (+C)$$

So we have

$$\alpha_b(t) = \frac{\sqrt{\pi}\, a_{max} D^2}{4i} \sum_\sigma \sigma\, e^{A_\sigma^2 - 9} \times$$

$$\times \left(\frac{t}{D} - A_\sigma\right)\mathrm{erf}\left(\frac{t}{D} - A_\sigma\right) + \frac{1}{\sqrt{\pi}}\, e^{-\left(\frac{t}{D}-A_\sigma\right)^2}$$

where $\sigma = \{1, -1\}$, $A_\sigma = 3 + \frac{\sigma i D\omega}{2}$