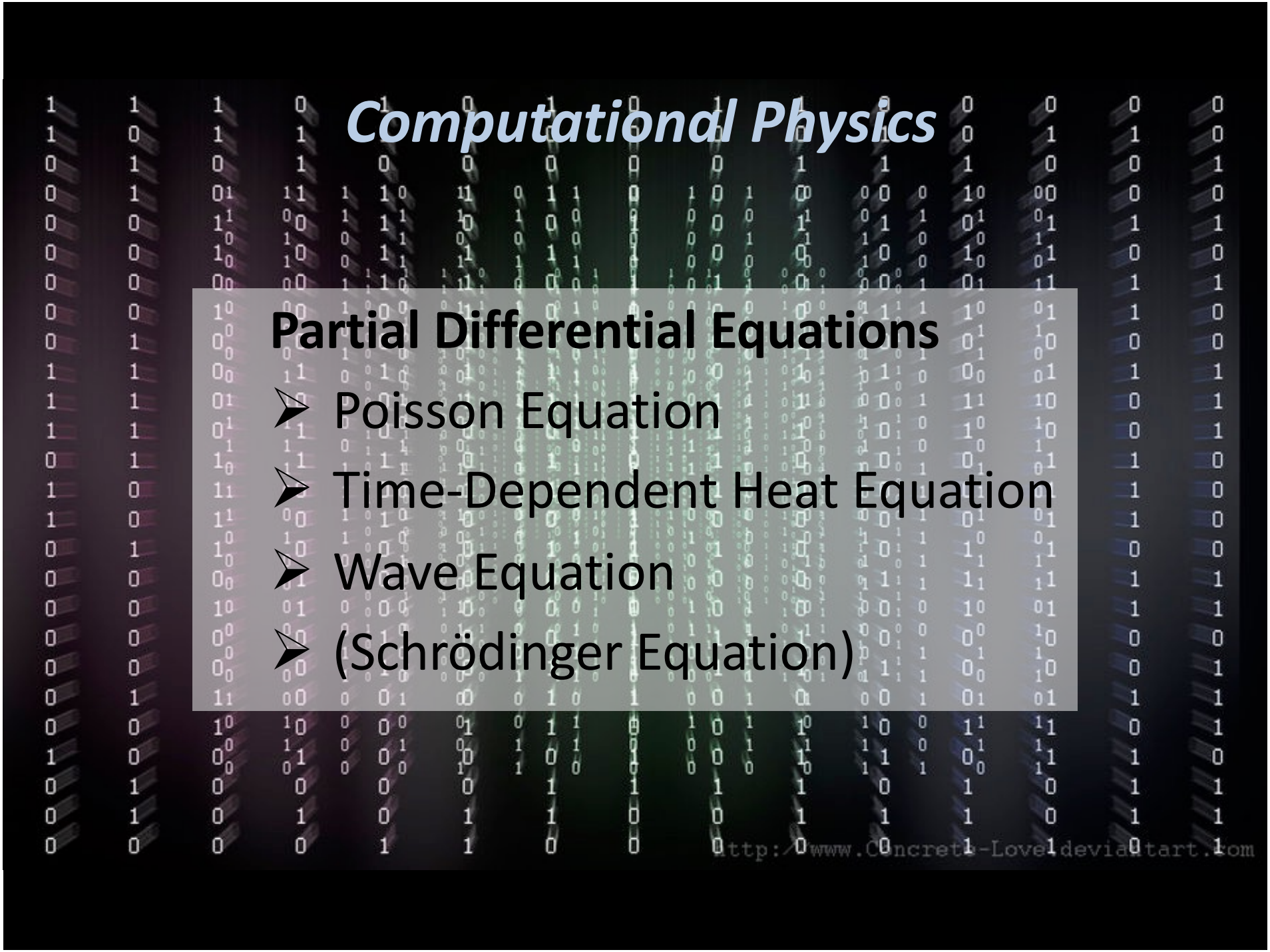

Computational Physics
Partial Differential Equations
Poisson Equation
Time-Dependent Heat Equation
Wave Equation
(Schrödinger Equation)

Introduction

- Here we consider **linear partial differential equations** (PDEs), which can be solved using finite difference methods.
- Nonlinear PDEs require more advanced methods like finite element or finite volume methods, or certain approximations. Examples are *Navier-Stokes* or *time-dependent Ginzburg-Landau* equations.
- Main concepts are already introduced. Here examples of elliptic, parabolic, and hyperbolic equations.
- In contrast to the numerical solution of ODEs, there exists no general recipe for the solution of PDEs.
- As for ODEs, the problem is only fully determined when initial and/or boundary conditions have been defined.

Classification

For a real-valued function u(x, y) of two independent real variables, x and y, a second-order, linear, constant-coefficient PDE for u(x,y) takes the form:

$$Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y + F = 0,$$

If interpreted as polynomial in two variables defines the following notation:

- **Elliptic** PDE: B^2-AC<0 : Poisson equation, Laplace equation
- **Parabolic** PDE: B^2-AC=0 : heat conduction, particle diffusion, and pricing of derivative investment instruments
- **Hyperbolic** PDE: B^2-AC>0 : wave equation

The Poisson equation

The Poisson Equation is an example of an **elliptic** PDE.

Here we consider the force of **F**(**r**,t) acting on a charged moving particle in 3D space, **r**, and time, t. Its velocity is v and the force is a result of the electric field, **E**(**r**,t), and magnetic field, **B**(**r**,t):

$$F(r,t) = q\left[E(r,t) + v \times B(r,t)\right]$$

In the electrostatic case [**B**=0, **E**(**r**,t)=**E**(**r**)]:

$$\nabla \cdot E(r) = \frac{1}{\epsilon_0}\rho(r)$$

with charge density ρ(**r**,t). Furthermore, with the electrostatic potential φ(**r**,t) we can write:

$$E(r) = -\nabla\varphi(r)$$

and as a result:

$$\Delta\varphi(r) = -\frac{\rho(r)}{\epsilon_0}$$

Poisson equation

...

For the case $\rho(\mathbf{r},t)=0$, the equation is called the Laplace equation.

Here we focus on the 2D static case, and solve the Poisson equation on the domain $\Omega=[0,L_x]\times[0,L_y]$, where we define the values of $\varphi(x,y)$ on the boundary of Ω, i.e. $\partial\Omega$, by a function g(x,y) and get the two-dimensional boundary value problem: (ϵ_0 is absorbed in ρ)

$$\begin{cases} \dfrac{\partial^2}{\partial x^2}\varphi(x,y) + \dfrac{\partial^2}{\partial y^2}\varphi(x,y) = -\rho(x,y)\ , & (x,y)\in\Omega \\ \varphi(x,y) = g(x,y)\ , & (x,y)\in\partial\Omega \end{cases}$$

We discretize x and y coordinates:

$$x_i = x_0 + ih_x, \quad i = 0,1,2,\ldots,n$$

$$y_j = y_0 + jh_y, \quad j = 0,1,2,\ldots,m$$

where h_x and h_y are the grid spacings in x and y directions

Poisson equation: discretization

The values of φ and ρ at a grid point (x_i,y_j) are defined by:

$$\varphi_{i,j} \equiv \varphi(x_i, y_j)$$
$$\rho_{i,j} \equiv \rho(x_i, y_j)$$

The Poisson equation, discretized by finite differences is then given by

$$\frac{\varphi_{i-1,j} - 2\varphi_{i,j} + \varphi_{i+1,j}}{h_x^2} + \frac{\varphi_{i,j-1} - 2\varphi_{i,j} + \varphi_{i,j+1}}{h_y^2} = -\rho_{i,j}$$

or:

$$\left(h_x^2 + h_y^2\right)\varphi_{i,j} - \frac{1}{2}\left[h_y^2\left(\varphi_{i-1,j} + \varphi_{i+1,j}\right) + h_x^2\left(\varphi_{i,j-1} + \varphi_{i,j+1}\right)\right] = \frac{(h_x h_y)^2}{2}\rho_{i,j}$$

for i=1,…n, j=1,…m

And the boundary conditions are:

$$\varphi_{0,j} = g_{0,j}\ , \qquad j = 0, 1, \ldots, m\ ,$$
$$\varphi_{n,j} = g_{n,j}\ , \qquad j = 0, 1, \ldots, m\ ,$$
$$\varphi_{i,0} = g_{i,0}\ , \qquad i = 1, 2, \ldots, n-1$$
$$\varphi_{i,m} = g_{i,m}\ , \qquad i = 1, 2, \ldots, n-1$$

Solution of the boundary problem

To solve for the unknown values of $\varphi_{i,j}$, one usually assigns a single index:

$$\varphi_{1,1} \rightarrow \varphi_1$$
$$\varphi_{1,2} \rightarrow \varphi_2$$
$$\vdots \quad \vdots$$
$$\varphi_{n,m} \rightarrow \varphi_\ell$$

with ℓ=nm

And then solves the resulting linear equation system for vector $\boldsymbol{\varphi}=(\varphi_1,\ldots,\varphi_\ell)^T$ directly (GE) or using iterative methods (see Appendix C). The boundary values are included in the matrix.

Here we will use an iterative method, which approximates the solution sequentially and thus introduces a "time" index for the φ_k which denotes the successively improved approximations for the real solution.

Iterative Methods (see also Appendix C)

- Iterative methods formally yield the solution x of a linear system after an infinite number of steps.
- At each step they require the computation of the residual of the system.
- In the case of a full matrix, their computational cost is therefore of the order of n^2 operations for each iteration, to be compared with an overall cost of the order of $2/3n^3$ operations needed by direct methods.

→ Iterative methods can therefore become competitive with direct methods provided the number of iterations that are required to converge (within a prescribed tolerance) is either independent of n or scales sublinearly with respect to n.

(Some) iterative methods can be parallelized!

Direct methods are typically sequential, and each step depends on the result of the previous one.

Main concept

The basic idea of iterative methods is to construct a sequence of vectors $\mathbf{x}^{(k)}$ that enjoy the property of convergence:

$$\mathbf{x} = \lim_{k \to \infty} \mathbf{x}^{(k)}$$

where **x** is the solution of A**x**=**b**

The iteration processes is stopped when $\|\mathbf{x}^{(n)} - \mathbf{x}\| < \varepsilon$ with a prescribed tolerance ϵ.

Problem with this conditions: Impractical, since we do not know **x**.

General scheme:

$$\mathbf{x}^{(0)} = \mathbf{f}_0(\mathrm{A}, \mathbf{b}),$$

$$\mathbf{x}^{(n+1)} = \mathbf{f}_{n+1}(\mathbf{x}^{(n)}, \mathbf{x}^{(n-1)}, \ldots, \mathbf{x}^{(n-m)}, \mathrm{A}, \mathbf{b}), \text{ for } n \geq m$$

Definitions

$$\mathbf{x}^{(0)} = \mathbf{f}_0(\mathrm{A}, \mathbf{b}),$$

$$\mathbf{x}^{(n+1)} = \mathbf{f}_{n+1}(\mathbf{x}^{(n)}, \mathbf{x}^{(n-1)}, \ldots, \mathbf{x}^{(n-m)}, \mathrm{A}, \mathbf{b}), \text{ for } n \geq m$$

In this general scheme f_i and $x^{(m)}$,…, $x^{(1)}$ are given functions and vectors, respectively.

- The number of steps which the current iteration depends on is called the *order of the method*.
- If the functions f_i are independent of the step index i, the method is called *stationary*, otherwise it is *nonstationary*.
- Finally, if f_i depends linearly on $x^{(0)}$, …, $x^{(m)}$, the method is called *linear*, otherwise it is *nonlinear*.

Linear iterative methods

Here we focus on stationary, linear iterative methods of order one.

- general technique: additive splitting of matrix A of form A=P−N
- P and N are two suitable matrices and P is nonsingular
- P is called preconditioning matrix or preconditioner

Here we consider an iteration of the form

$$\mathbf{x}^{(0)} \text{ given}, \quad \mathbf{x}^{(k+1)} = \mathrm{B}\mathbf{x}^{(k)} + \mathbf{f}, \quad k \geq 0$$

- B is an n x n square matrix called the iteration matrix
- **f** is a vector obtained from the right-hand side **b**
- Consistent with A**x**=**b** if **x**=B**x**+**f** or **f**=(I-B)A^{-1}**b**

Using the above splitting of A, we calculate $\mathbf{x}^{(k)}$ for k>0, solving

$$\mathrm{P}\mathbf{x}^{(k+1)} = \mathrm{N}\mathbf{x}^{(k)} + \mathbf{b}, \quad k \geq 0$$

i.e., B=P^{-1}N and **f**=P^{-1}**b**

...

This scheme can be written as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathrm{P}^{-1}\mathbf{r}^{(k)}$

with the ***residual*** $\mathbf{r}^{(k)} = \mathbf{b} - \mathrm{A}\mathbf{x}^{(k)}$

We note that:

1. P should be chosen such that it can be easily inverted
2. If P=A and N=0, the iteration would converge in one step
3. The residual is a measure of how good $\mathbf{x}^{(k)}$ approximates the real solution **x**

Jacobi iteration

If the diagonal entries of A are nonzero, we can single out in each equation the corresponding unknown on the diagonal and write:

$$x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \right], \qquad i = 1, \ldots, n$$

In the Jacobi method $\mathbf{x}^{(k+1)}$ is computed by [$\mathbf{x}^{(0)}$ can be an arbitrary initial guess]

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, \ldots, n$$

This corresponds to a splitting: P=D, N=D-A=E+F,

- D is a diagonal matrix having the diagonal elements of A
- E is the lower triangular matrix with elements: $e_{ij}=-a_{ij}$ for i>j, 0 else
- F the upper triangular matrix: $f_{ij}=-a_{ij}$ for i<j, 0 else

A generalization is the over-relaxation method (JOR): $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega \mathrm{D}^{-1} \mathbf{r}^{(k)}$

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x_i^{(k)}, \qquad i = 1, \ldots, n$$

where ω is a relaxation parameter $0<\omega \leq 1$

...

Remarks:

- In the Jacobi method P=D can be easily inverted
- Each iteration step required therefore only one matrix multiplication, i.e., $A\mathbf{x}^{(k)}$
- Therefore, it can be easily parallelized
- The method converges when A is strictly diagonally dominant, i.e., $|a_{ii}|$ is larger than the sum of all other absolute values of the elements in the row
- Standard convergence criterion: $\rho(D^{-1}N)<1$ (ρ is the spectral radius, i.e., the largest absolute value of this eigenvalues)
- Jacobi is convergent if A and (2D-A) are symmetric and positive definite
- The above convergence criterions are not always necessary for convergence…

Jacobi algorithm

Choose an initial guess $x^{(0)}$ to the solution
k = 0
check if convergence is reached, e.g., $||\mathbf{r}(k)||_\infty < \epsilon$
while convergence not reached do
  for i := 1 step until n do
    $\sigma = 0$
    for j := 1 step until n do
      if $j \neq i$ then
        $\sigma = \sigma + a_{ij}\, x_j^{(k)}$
      end if
    end (j-loop)
    $x_i^{(k+1)} = (b_i - \sigma)/ a_{ii}$
  end (i-loop)
  check if convergence is reached
  k = k + 1
loop (while convergence condition not reached)

Gauss-Seidel iteration

The Gauss-Seidel method differs from the Jacobi method in the fact that at the (k+1)-th step the available values of $x_i^{(k+1)}$ are being used to update the solution

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, \ldots, n$$

i.e., P=D-E , N=F

The related over-relaxation iteration (SOR) is

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega) x_i^{(k)}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \left(\frac{1}{\omega} \mathrm{D} - \mathrm{E} \right)^{-1} \mathbf{r}^{(k)}$$

Remarks:

- GS is monotonically convergent if A is symmetric and positive definite
- GS converges also for the same criteria as Jacobi
- GS is not parallelizable
- GS has less memory requirements than Jacobi, since the current iteration can overwrite elements of the previous approximation

Iterative solution of the Poisson equation

We denote the approximation for $\varphi_{i,j}$ as $\varphi_{i,j}^{t}$ after t iteration steps and define the explicit iteration rule:

$$\varphi_{i,j}^{t+1} = \frac{(h_x h_y)^2}{2(h_x^2 + h_y^2)} \rho_{i,j} + \frac{1}{2(h_x^2 + h_y^2)} \left[h_y^2 \left(\varphi_{i-1,j}^{t+1} + \varphi_{i+1,j}^{t} \right) + h_x^2 \left(\varphi_{i,j-1}^{t+1} + \varphi_{i,j+1}^{t} \right) \right] ,$$

Next, we solve an explicit Poisson equation.

Potential for an electric monopole, dipole, and quadropole

We use the Dirichlet boundary conditions:

$\varphi(0,y)=\varphi(L_x,y)=0$ in x-direction and

$\varphi(x,0)=\varphi(x,L_y)=0$ in y-direction

These boundary conditions can be integrated into the iteration rule by limiting the loop over the interior grid points and leave the boundary values of φ unchanged.

We use $L_x=L_y=10$ and n=m=100 and define the domains:

$$\Omega_1 = \left(x_{\frac{n}{2}-10}, x_{\frac{n}{2}}\right] \times \left(y_{\frac{m}{2}-10}, y_{\frac{m}{2}}\right]$$

$$\Omega_2 = \left(x_{\frac{n}{2}}, x_{\frac{n}{2}+10}\right] \times \left(y_{\frac{m}{2}-10}, y_{\frac{m}{2}}\right]$$

$$\Omega_3 = \left(x_{\frac{n}{2}-10}, x_{\frac{n}{2}}\right] \times \left(y_{\frac{m}{2}}, y_{\frac{m}{2}+10}\right]$$

$$\Omega_4 = \left(x_{\frac{n}{2}}, x_{\frac{n}{2}+10}\right] \times \left(y_{\frac{m}{2}}, y_{\frac{m}{2}+10}\right]$$

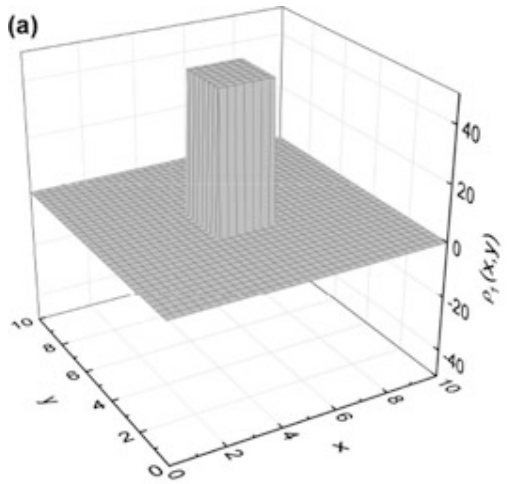
on which we define the charge density ρ

...

Here:

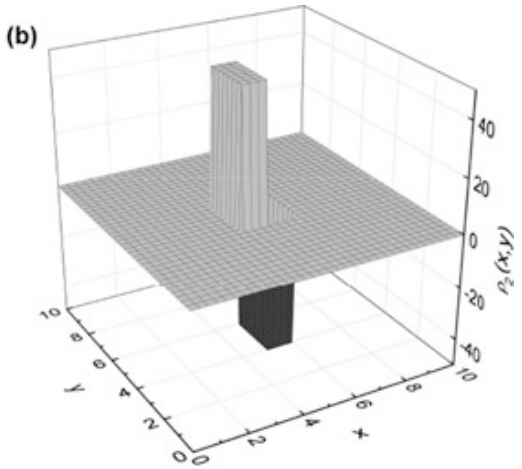
a) Monopole:

$$\rho_1(x,y) = \begin{cases} 50 & (x,y) \in \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4 \\ 0 & \text{elsewhere,} \end{cases}$$



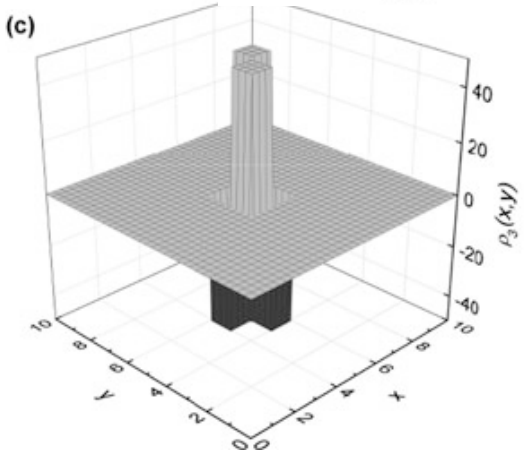
b) Dipole:

$$\rho_2(x,y) = \begin{cases} 50 & (x,y) \in \Omega_1 \cup \Omega_2 \\ -50 & (x,y) \in \Omega_3 \cup \Omega_4 \\ 0 & \text{elsewhere,} \end{cases}$$



c) Quadropole:

$$\rho_3(x,y) = \begin{cases} 50 & (x,y) \in \Omega_1 \cup \Omega_4 \\ -50 & (x,y) \in \Omega_2 \cup \Omega_3 \\ 0 & \text{elsewhere.} \end{cases}$$



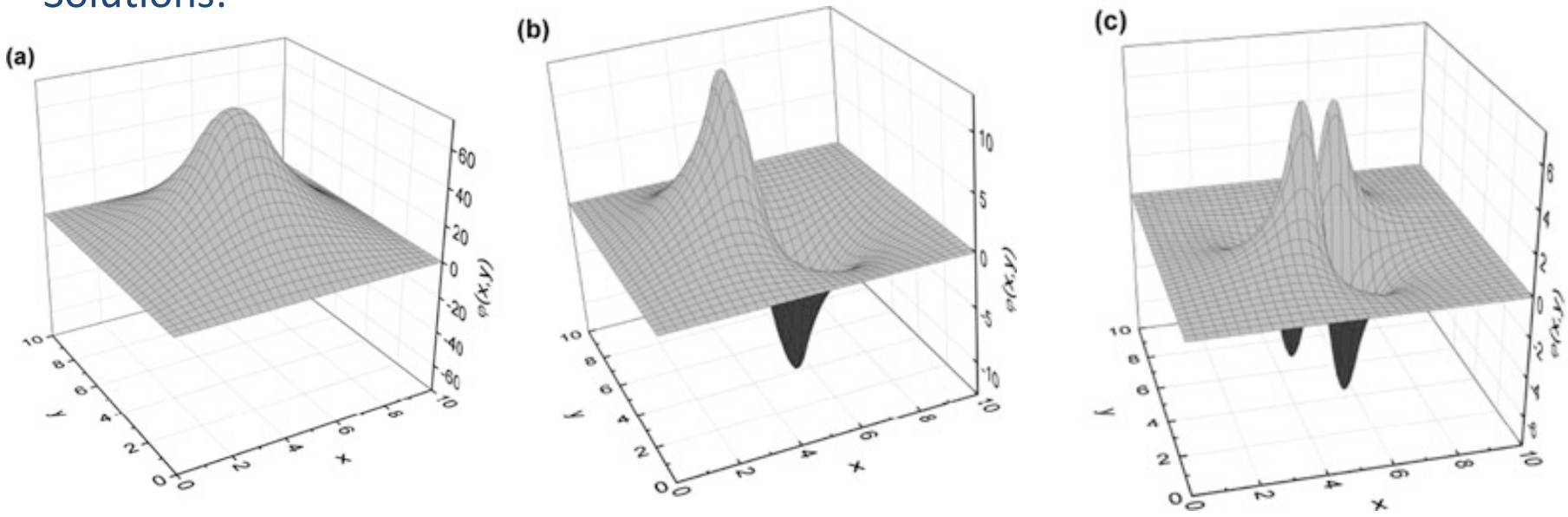
Convergence & Solutions

We consider the iteration converged at iteration t, when

$$\max_{i,j}\left(|\varphi_{i,j}^{t}-\varphi_{i,j}^{t-1}|\right)<\eta$$

We choose η=10^{-4}. From the solution for φ(x,y) we can then obtain **E.**

Solutions:



The time-dependent heat equation

We already discussed the stationary heat equation. Here we use the 1D time-dependent heat equation as an example of a parabolic PDE

$$\frac{\partial}{\partial t}T(x,t) = \kappa \frac{\partial^2}{\partial x^2}T(x,t)$$

With the central difference approximation for the rhs, we write:

$$\frac{\partial}{\partial t}T_k(t) = \kappa \frac{T_{k-1}(t) - 2T_k(t) + T_{k+1}(t)}{h^2}$$

where h is the grid spacing in x-direction: $x_k=x_0+k*h$, k=0,…,N and correspondingly $T_k(t)=T(x_k,t)$

time discretization

For the discretization of the time derivate we can choose any of the methods we have studied so far. Here we compare: explicit Euler, implicit Euler, and Crank-Nicolson. The latter is typically the best for parabolic PDEs. With

$$t_n = t_0 + n\Delta t \text{ and } T_k^n \equiv T_k(t_n)$$

we can write for the explicit Euler scheme:

$$\frac{T_k^{n+1} - T_k^n}{\Delta t} = \kappa \frac{T_{k-1}^n - 2T_k^n + T_{k+1}^n}{h^2}$$

or

$$T_k^{n+1} = T_k^n + \kappa \Delta t \frac{T_{k-1}^n - 2T_k^n + T_{k+1}^n}{h^2}$$

which is straightforward to solve, since the rhs only depends on the solution for the previous time step. This method is stable for

$$\frac{\kappa \Delta t}{h^2} \leq \frac{1}{2}$$

implicit Euler

For the implicit Euler scheme we get:

$$\frac{T_k^{n+1} - T_k^n}{\Delta t} = \kappa \frac{T_{k-1}^{n+1} - 2T_k^{n+1} + T_{k+1}^{n+1}}{h^2}$$

which again requires to solve a linear equation system to get all T_k^{n+1}, which has the form

$$AT^{n+1} = T^n + F \qquad \text{with} \quad T^n = (T_0^n, T_1^n, \ldots, T_N^n)^T$$

and

$$A = \begin{pmatrix} \ddots & \ddots & \ddots & & \\ & -\frac{\kappa\Delta t}{h^2} & 1 + \frac{2\kappa\Delta t}{h^2} & -\frac{\kappa\Delta t}{h^2} & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

boundary conditions are incorporated in A and F. This can be solved directly or iteratively

Example

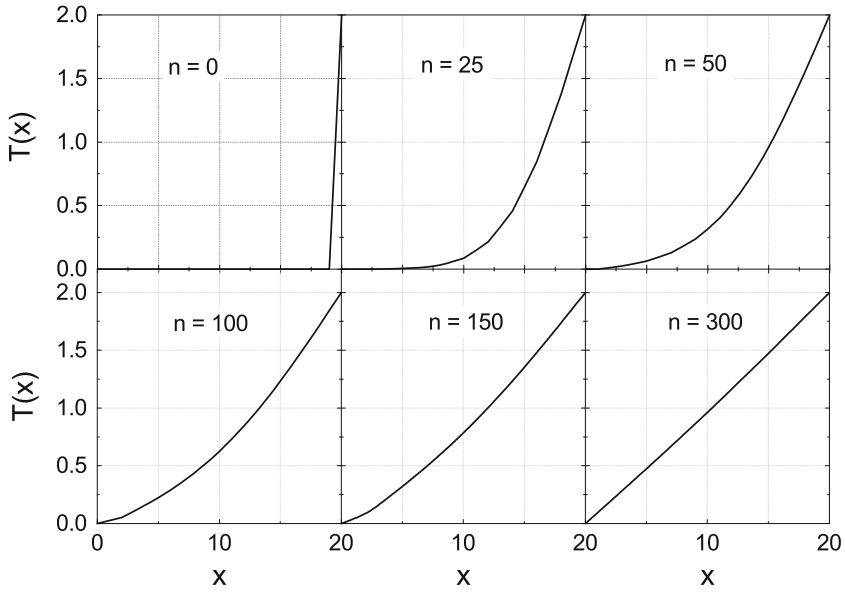
Here we solve the heat equation on the interval [0,L] in x-direction with boundary condition:

$$T(0) = T_0, \qquad T(L) = T_N$$

which we also supply with the initial condition

$$T(x,0) = 0, \qquad x \in [0, L]$$

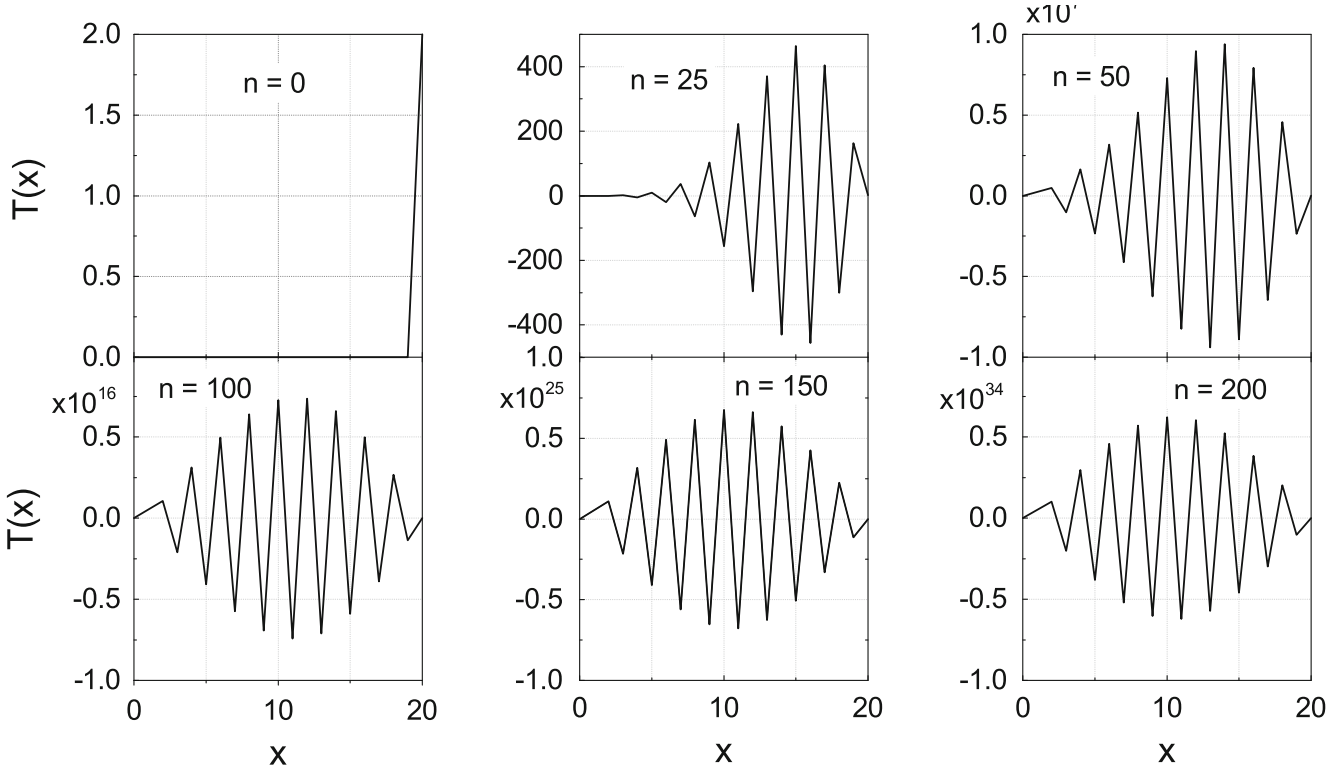
For the explicit Euler scheme with T_0=0, T_N=2, N=20, L=10, κ=1 and Δt=0.5



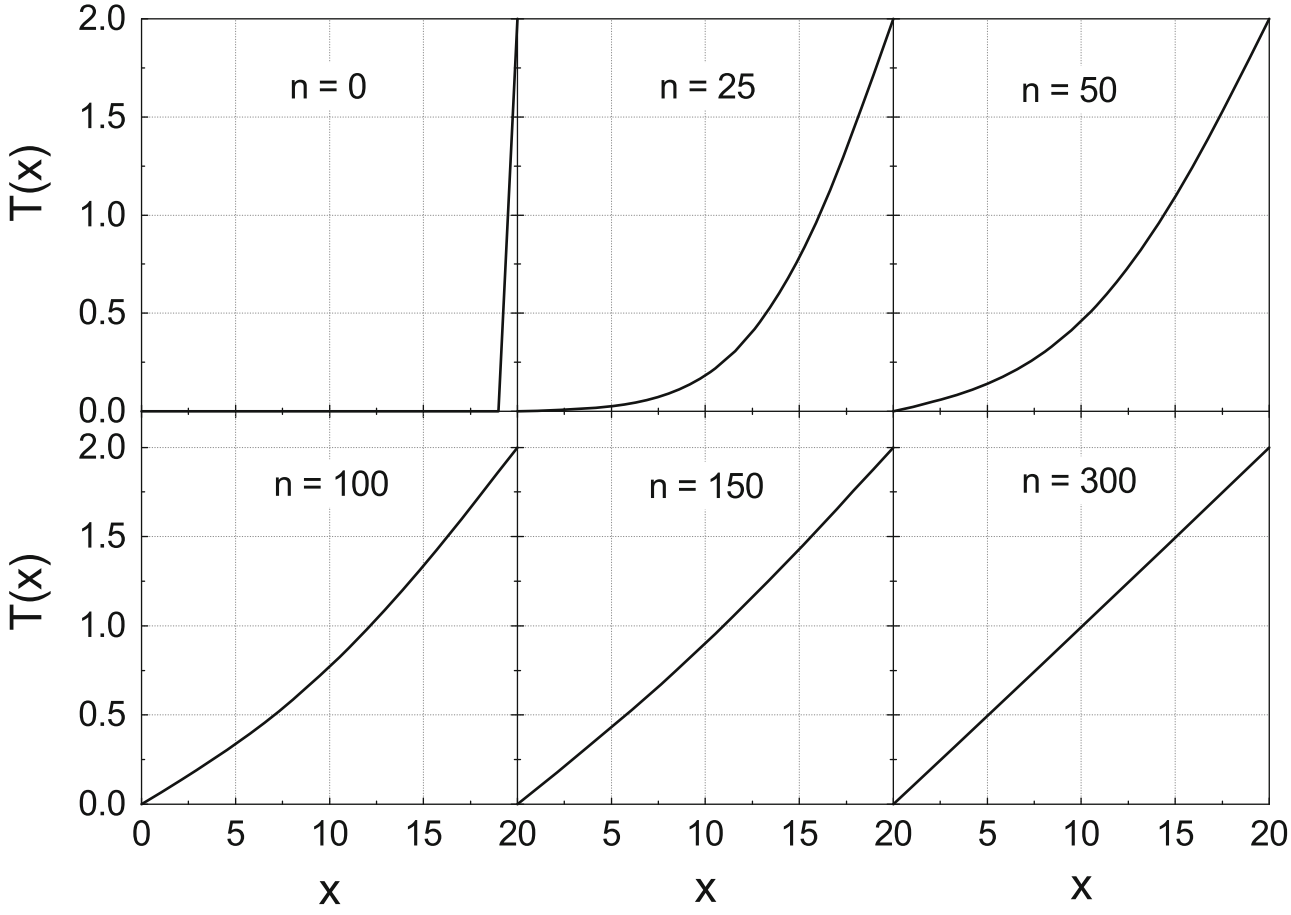
$$\frac{\kappa \Delta t}{h^2} \approx 0.45 \leq \frac{1}{2}$$

instability of explicit Euler

If we would have chosen Δt=0.7 → $\frac{\kappa \Delta t}{h^2} \approx 0.63 > \frac{1}{2}$



compared to implicit Euler



$\Delta t=0.7$

The wave equation

Finally, we use the wave equation as example for a hyperbolic PDE:

$$\frac{\partial^2}{\partial t^2}u(x,t) = c^2\frac{\partial^2}{\partial x^2}u(x,t)$$

Here c is the speed (e.g. of light) at which the wave u(x,t) propagates
We use the same discretization method as before and use the explicit Euler scheme:

$$\frac{u_k^{n-1} - 2u_k^n + u_k^{n+1}}{\Delta t^2} = c^2\frac{u_{k-1}^n - 2u_k^n + u_{k+1}^n}{h^2}$$

define

$$\lambda = \frac{c\Delta t}{h}$$

we get

$$u_k^{n+1} = 2(1-\lambda^2)u_k^n - u_k^{n-1} + \lambda^2(u_{k-1}^n + u_{k+1}^n)$$

...

- In order to solve this, we need to know u(x,t) at two previous time steps, in particular for n=0 and n=1
- This explicit Euler scheme is only stable for $\lambda = \frac{c\Delta t}{h} \leq 1$ (COURANT-FRIEDRICHS-LEWY or CFL condition)
- CFL holds for hyperbolic problems in general

If the following initial conditions are given:

$$u(x,0) = f(x), \qquad \frac{\partial}{\partial t}u(x,0) = g(x)$$

which are discretized as:

$$u_k^0 = f_k, \qquad \frac{u_k^1 - u_k^0}{\Delta t} = g_k$$

and therefore:

$$u_k^1 = u_k^0 + \Delta t g_k$$

Then we can solve the explicit equations.
One can also use higher order terms in the Taylor series for the discretization for the 2nd initial condition:

$$\frac{u_k^1 - u_k^0}{\Delta t} = \frac{\partial}{\partial t}u(x,0) + \frac{\Delta t}{2}\frac{\partial^2}{\partial t^2}u(x,0) + \mathcal{O}(\Delta t^2)$$

which in combination with the original wave equation yields for u^1_k:

$$u_k^1 = u_k^0 + \Delta t g_k + \frac{\Delta t^2 c^2}{2} f_k'' + \mathcal{O}(\Delta t^3)$$

Example

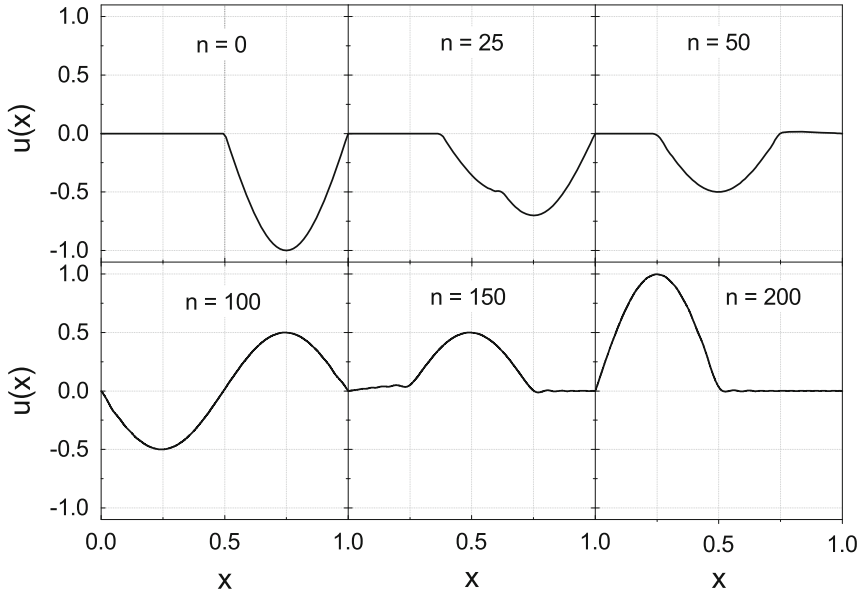
Here we consider a vibrating string of length L, which is fixed at its ends, i.e. u(0,t)=u(L,t)=0. Furthermore, we assume that the string was initially at rest:

$$\frac{\partial}{\partial t}u(x,0) = 0$$

plus initial condition

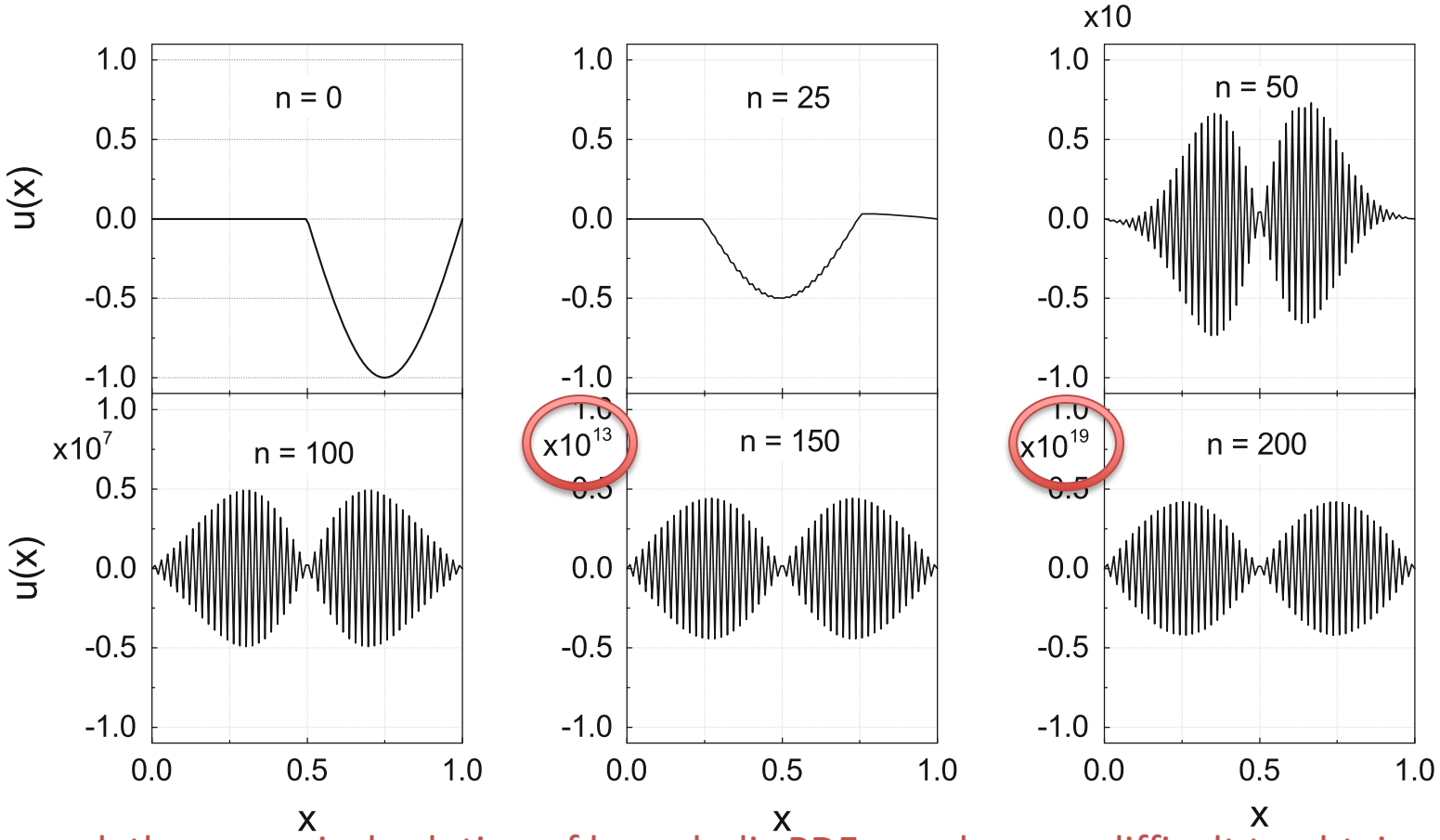
$$u(x,0) = \begin{cases} \sin\left(\frac{2\pi x}{L}\right) & x \in \left(\frac{L}{2}, L\right] , \\ 0 & \text{elsewhere.} \end{cases}$$

Solved for L=1; c=2, N=100, and Δt chosen such that λ=0.5 gives:



… instability

For λ=1.01 CFL is violated and it fails…



In general, the numerical solution of hyperbolic PDEs can be very difficult to obtain since in many cases these equations represent conservation laws. Typically, other methods are used (FEM, or finite volume)

The Time-Dependent Schrödinger Equation

The time-dependent Schrödinger equation is given by

$$i\hbar \frac{\partial}{\partial t}\Psi(x,t) = H\Psi(x,t)$$

which is formally solved by

$$\Psi(x,t) = \exp\left(-\frac{it}{\hbar}H\right)\Psi(x,0) = U(t)\Psi(x,0)$$

where U(t) is the unitary time-evolution operator. The unitarity preserves the norm of the complex wave function ψ(x,t).

Here just a brief sketch the main idea.

For more details read Chapters 10 and 11.5 and Appendix D

Schrödinger equation: explicit & implicit

for a single timestep: $$\Psi(x,t+\Delta t) = \exp\left[-\frac{i(t+\Delta t)}{\hbar}H\right]\Psi(x,0) = \exp\left(-\frac{i\Delta t}{\hbar}H\right)\Psi(x,t)$$

truncating the exponential series gives the relation:

$$\Psi(x,t+\Delta t) \approx \left(1-\frac{i\Delta t}{\hbar}H\right)\Psi(x,t)$$

and then the explicit scheme:

$$\Psi_k^{n+1} = \Psi_k^n - \frac{i\Delta t}{\hbar}\left(-\frac{\hbar^2}{2m}\frac{\Psi_{k-1}^n - 2\Psi_k^n + \Psi_{k+1}^n}{\Delta x^2} + V_k\Psi_k^n\right)$$

Using the conjugate we get the implicit version:

$$\Psi(x,t) = U^\dagger(\Delta t)\Psi(x,t+\Delta t) = \exp\left(\frac{i\Delta t}{\hbar}H\right)\Psi(x,t+\Delta t) \quad \rightarrow \quad \Psi_k^n = \left(1+\frac{i\Delta t}{\hbar}H\right)\Psi_k^{n+1}$$

Warning: These violate the unitarity, so the wave function will not remain normalized, so one needs to normalize after each time step

Crank-Nicholson

The Crank-Nicholson scheme can improve the situation, by rewriting

$$\begin{aligned} U(\Delta t) &= \exp\left(-\frac{i\Delta t}{\hbar}H\right) \\ &= \exp\left(-\frac{i\Delta t}{2\hbar}H\right)\exp\left(-\frac{i\Delta t}{2\hbar}H\right) \\ &= \exp\left(\frac{i\Delta t}{2\hbar}H\right)^{-1}\exp\left(-\frac{i\Delta t}{2\hbar}H\right) \\ &= \left[U^{\dagger}\left(\frac{\Delta t}{2}\right)\right]^{-1} U\left(\frac{\Delta t}{2}\right) . \end{aligned}$$

which gives:

$$U^{\dagger}\left(\frac{\Delta t}{2}\right)\Psi_k^{n+1} = U\left(\frac{\Delta t}{2}\right)\Psi_k^{n}$$

and after truncation:

$$\left(1+\frac{i\Delta t}{2\hbar}H\right)\Psi_k^{n+1} = \left(1-\frac{i\Delta t}{2\hbar}H\right)\Psi_k^{n}$$

This is then used with the explicit form of the Hamiltonian and rewritten as a liner equation system. This is quite convoluted → see book.

Example

Start with a Gaussian wave packet:

$$\Psi(x,0) = \exp(iqx)\exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]$$

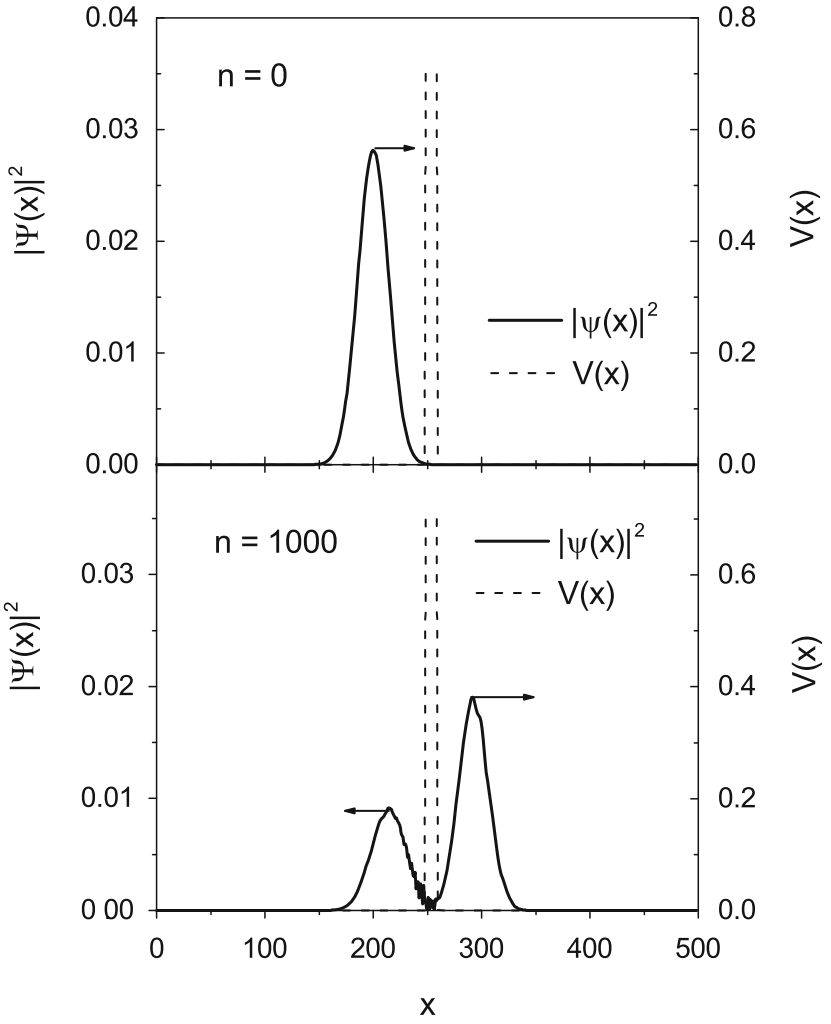
for a single potential barrier:

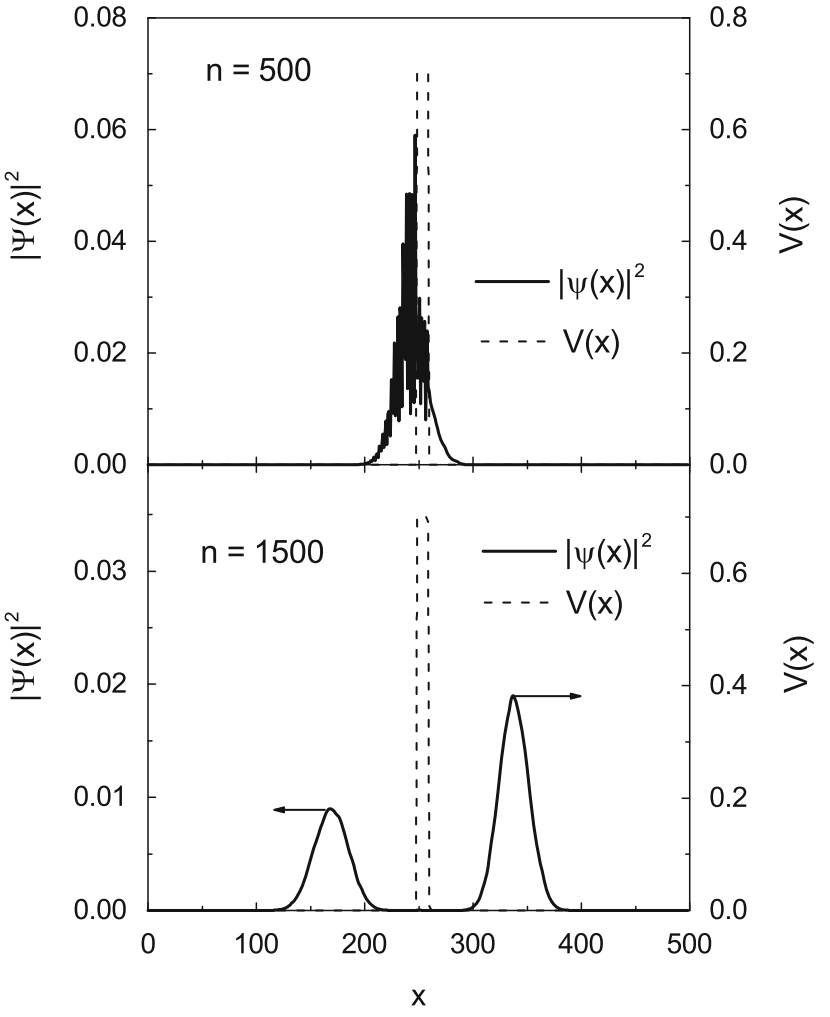
$$V_1(x) = \begin{cases} V_0 & x \in [a,b]\ , \\ 0 & \text{elsewhere,} \end{cases}$$

or a double barrier:

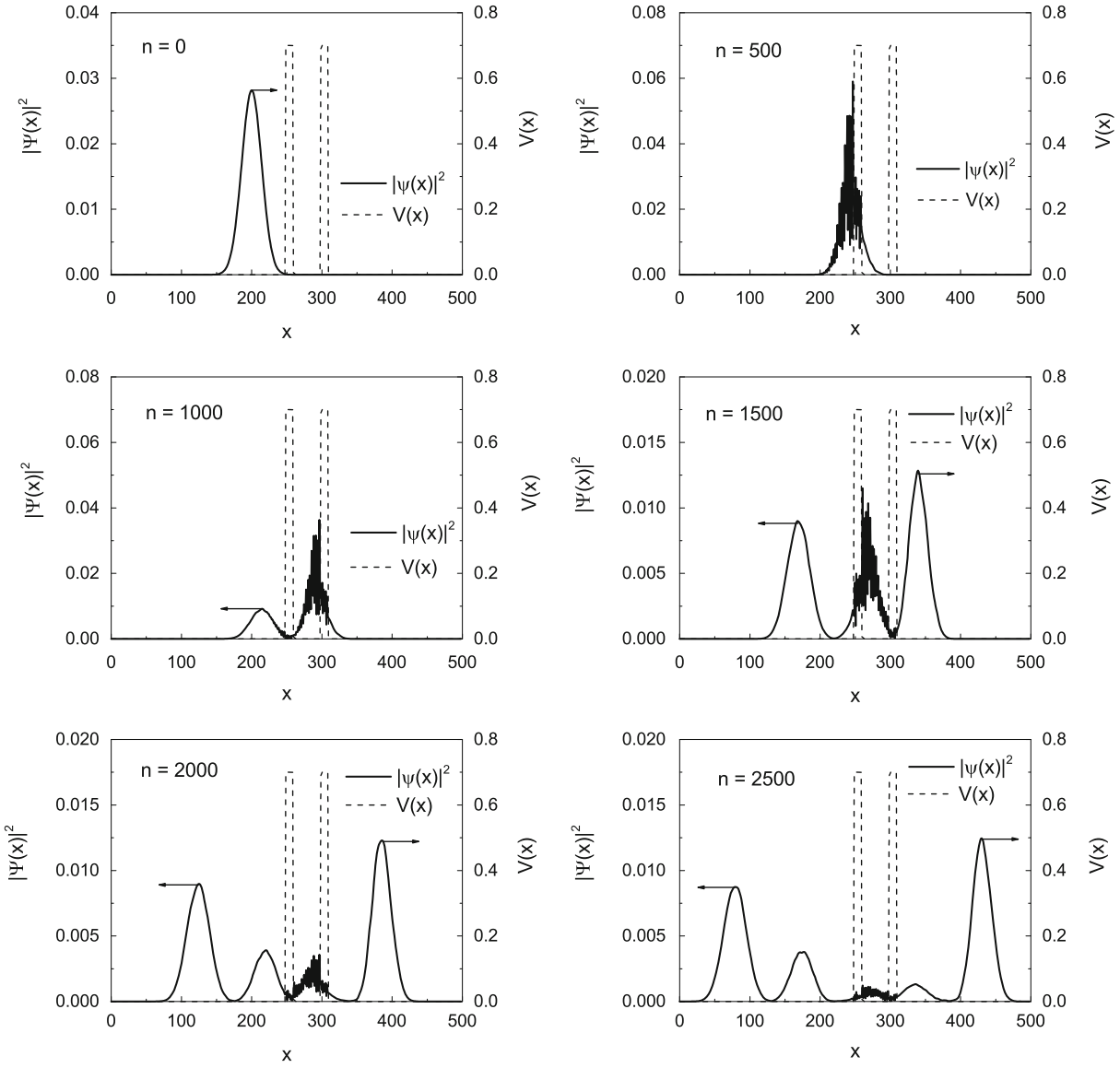
$$V_2(x) = \begin{cases} V_0 & x \in [a,b] \cup [c,d] \\ 0 & \text{elsewhere.} \end{cases}$$

tunneling through single barrier





tunneling through double barrier



This week's/Next homework task

- Implement the iterative Poisson solver for arbitrary charge density $\rho(x,y)$
- Implement a time-dependent heat equation solver