Basic Numerical Solution Methods for Differential Equations

Sebastian Merkel

September 15, 2019

1 Ordinary Differential Equations (ODEs)

1.1 Preliminaries

- A first-order ordinary differential equation is an equation of the form

$$f(x, y, y') = 0 \tag{1}$$

  with a function $f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$. A solution is a differentiable function $y : \mathbb{R} \to \mathbb{R}^n$ with the property

$$\forall x \in \mathbb{R} : \ f(y(x), y'(x), x) = 0.$$

  Definitions are adjusted in an obvious way, if f is just defined on a subset of $\mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$ and/or one is only interested in a solution y defined on a subset of $\mathbb{R}$.

- A k-th order ODE is similarly an equation of the form

$$f(y, y', ..., y^{(k)}, x) = 0$$

  with a fucntion $f : \mathbb{R}^n \times \cdots \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$. Such an equation can always be reduced to a $k \cdot n$-dimensional first-order ODE by associating it with an equivalent equation for the $k \cdot n$-dimensional vector function

$$z := \begin{pmatrix} y \\ y' \\ \vdots \\ y^{(k-1)} \end{pmatrix},$$

  whose first deriative is

$$z' = \begin{pmatrix} y' \\ y^{(1)} \\ \vdots \\ y^{(k)} \end{pmatrix}.$$

  The equivalent equation is

$$h(z, z', x) = 0$$

with

$$h_i\left(\begin{pmatrix} z_1 \\ \vdots \\ z_k \end{pmatrix}, \begin{pmatrix} z'_1 \\ \vdots \\ z'_k \end{pmatrix}, x\right) = \begin{cases} f_i\left(z_1, z'_1, \ldots, z'_k, x\right), & i = 1, \ldots, n \\ z'_{j-1,i} - z_{j,i}, & j \in \{2, ..., k\}, i = (j-1)n + 1, ..., jn \end{cases}$$

for $i = 1, ..., k \cdot n$. For this reason we focus in the following on first-order ODEs.

- An equation of the form (1) is called (fully) implicit. Often, the equation can be written in explicit form,
$$y' = g(x, y). \tag{2}$$
- For explicit ODEs there are simple sufficient conditions for existence and uniqueness of a solution satisfying $y(x_0) = y_0$ for a given initial value $(x_0, y_0) \in \mathbb{R} \times \mathbb{R}^n$:[1]
  - Existence (Peano theorem): if g is continuous, then a solution y to (2) such that $y(x_0) = y_0$ exists locally around x_0 (meaning a solution function exists on some interval $[x_0 - \varepsilon, x_0 + \varepsilon]$)
  - Existence and Uniqueness (Picard–Lindelöf theorem): if g is continuous and satisfies a local Lipschitz condition with respect to y, that is there is a constant $L < \infty$, such that $|g(x, y) - g(x, z)| \leq L|y - z|$ for all y, z in some neighborhood of y_0 and all x in some neighborhood of x_0, then there is a unique solution to (2) satisfying $y(x_0) = y_0$ – at least locally around x_0 (meaning there is an interval $[x_0 - \varepsilon, x_0 + \varepsilon]$ and a unique solution on that interval)

Local Lipschitz conditions are e.g. satisfied, if $\frac{\partial g}{\partial y}(x, y)$ exists and is bounded (locally around (x_0, y_0)), so regularly existence and uniqueness is not a problem in applications.

Note that these local solutions can usually be extended to a "global" solution, if they do not blow up/reach inadmissible values in finite time.

1.2 Numerical Solutions of ODEs

1.2.1 Explicit Euler Method

Let the following objects be given: some explicit ODE of the form (2), an initial condition (x_0, y_0) and a desired solution domain $[x_0, \bar{x}]$. A simple solution approach is to discretize the solution domain $[x_0, \bar{x}]$ into $N + 1$ points,

$$x_0 < x_1 < \cdots < x_N = \bar{x}$$

and approximate the derivatives $y'(x_{i-1}) = \frac{dy}{dx}(x_{i-1})$ by finite differences $\frac{y_i - y_{i-1}}{x_i - x_{i-1}}$ for $i = 1, ..., N$, where y_i is an approximation for the unknown $y(x_i)$. With this approximation, equation (2) evaluated at the points x_{i-1} becomes

$$\frac{y_i - y_{i-1}}{x_i - x_{i-1}} = g\left(x_{i-1}, y_{i-1}\right),$$

which can be solved for y_i:

$$y_i = y_{i-1} + g\left(x_{i-1}, y_{i-1}\right)\left(x_i - x_{i-1}\right). \tag{3}$$

[1]If the first-order system is a reduction from a k-th order system as above, this means that at x_0 the values of all first $k - 1$ derivatives must be pinned down by the initial value.

Together with the initial values y_0, equation (3) defines a simple recursion for the approximate function values $\{y_i\}_{i=0}^N$.

Alternatively, one may view the Euler method as a sequence of first-order Taylor approximations to the function y: a Taylor expansion around x_{i-1} evaluated in x_i yields

$$y(x_i) \approx y(x_{i-1}) + y'(x_{i-1})\,(x_i - x_{i-1}) = y(x_{i-1}) + g(x_{i-1}, y(x_{i-1}))\,(x_i - x_{i-1})\,,$$

where the second equation holds, because y satisfied ODE (3). Replacing the true function values $y(x_{i-1})$ and $y(x_i)$ by their approximations y_{i-1} and y_i yields again recursion (3).

1.2.2 Implicit Euler Method

Again, let an initial condition (x_0, y_0), a solution domain $[x_0, \bar{x}]$ and a discretization $\{x_i\}_{i=0}^N$ of that domain be given. The explicit Euler method approximates derivatives $y'(x_{i-1})$ by $\frac{y_i - y_{i-1}}{x_i - x_{i-1}}$ and uses the ODE in the points $\{x_0, ..., x_{N-1}\}$ to derive an explicit recursion for $\{y_i\}_{i=0}^N$. The implicit Euler method instead approximates $y'(x_i)$ by $\frac{y_i - y_{i-1}}{x_i - x_{i-1}}$ and uses the ODE in the points $\{x_1, ..., x_N\}$, which leads to the equations

$$\frac{y_i - y_{i-1}}{x_i - x_{i-1}} = g(x_i, y_i), \qquad i = 1, ..., N$$

Now, there is in general no explicit solution for y_i. Instead, one has to solve in each step a – potentially nonlinear – equation system to find them. While this looks much more complicated than the explicit method, the implicit method has often superior stability properties (compare Problem 3 of Problem Set 1).

This method does not require the ODE to be given in explicit form (2). It works equally for a fully implicit ODE (1). Then the equation system to solve in each step is

$$f\left(x_i, y_i, \frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right) = 0.$$

1.2.3 Other Methods

There are a lot of other methods, that we do not cover in this course. Two common classes of methods are (both exist in implicit and explicit variants):

- Runge-Kutta methods: like the Euler method they are single-step methods in that when computing the solution y_i over one discretization interval $[x_{i-1}, x_i]$, they only use the function value information from the last step (x_{i-1}, y_{i-1}) as an initial condition. However, each step is computed using multiple stages with the goal of increasing the order of consistency[2]. Each stage involves an evaluation of g or – in the case of implicit methods – an equation to be solved. Matlab's standard solvers for nonstiff problems are mainly based on these methods.

- Multi-step methods: use information (x_j, y_j, y_j') from multiple previous steps $j \leq i-1$ when computing (y_i, y_i') with the aim of gaining efficiency. Implicit versions of (linear) multi-step methods have worse stability properties than certain single-step methods for consistency orders larger than 2.

[2] In each step, a numerical scheme produces an error, the so-called truncation error. The method is said to be consistent of order p, if the truncation error in step i converges to 0 faster than $(x_i - x_{i-1})^p$ as the the step-width $x_i - x_{i-1}$ approaches 0. Euler methods are consistent of order 1.

Details on solvers pre-implemented in Matlab can be found in the ODE section of the Matlab help. You can also execute the command `odeexamples` for example code using the different Matlab solvers.

2 Partial Differential Equations (PDEs)

2.1 Some Basic Definitions

A general partial differential equation in n variables of order k is an equation of the form

$$F\left(x, u, Du, D^2u, \ldots, D^k u\right) = 0,$$

where $x \in \mathbb{R}^n$, $u : \mathbb{R}^n \to \mathbb{R}$ is a function and $D^1 u = \left(\frac{\partial u}{\partial x_i}\right)_{i=1,\ldots,n}$, $D^2 u = \left(\frac{\partial^2 u}{\partial x_{i_1}\partial x_{i_2}}\right)_{i_1,i_2=1,\ldots,n}$, $D^2 u = \left(\frac{\partial^3 u}{\partial x_{i_1}\partial x_{i_2}\partial x_{i_3}}\right)_{i_1,i_2,i_3=1,\ldots,n}$ etc. A (classical) solution u is a k times continuously differentiable function that satisfies the equation for all x. The solution theory of PDEs is much more complex than the one of ODEs and often, solutions only exist according to some weaker solution concept.[3] The focus here will be on second order PDEs[4] and in particular on PDEs that have the following *quasilinear* structure

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(x, u, Du)\frac{\partial^2 u}{\partial x_i \partial x_j}(x) + G(x, u, Du) = 0. \tag{4}$$

Despite the name "quasilinear", such PDEs can be highly nonlinear. These PDEs are typically classified into parabolic, hyperbolic and elliptic equations, although this classification does not cover all possible cases:[5]

- The equation is called parabolic, if the matrix $(a_{ij}(x, y, z))$ is (positive or negative) semidefinite with exactly one eigenvalue assuming the value 0 for all $x \in \mathbb{R}^n$, $y \in \mathbb{R}$, $z \in \mathbb{R}^n$
- The equation is called hyperbolic, if the matrix $(a_{ij}(x, y, z))$ has one eigenvalue that is positive or negative and the remaining $n-1$ eigenvalues have opposite sign (but are $\neq 0$) for all $x \in \mathbb{R}^n$, $y \in \mathbb{R}$, $z \in \mathbb{R}^n$
- The equation is called elliptic, if the matrix $(a_{ij}(x, y, z))$ is (positive or negative) definite for all $x \in \mathbb{R}^n$, $y \in \mathbb{R}$, $z \in \mathbb{R}^n$

Equations typically encountered in economics and finance are elliptic or parabolic. If the equation is parabolic, then the eigenvalue 0 of the $(a_{ij}(x, y, z))$ matrix usually has an Euklidian basis vector as an eigenvector. The associated coordinate can usually be interpreted as "time" and is separated in notation

[3] This may be also relevant for economists. For many PDEs in economic problems, particularly HJB-type equations, the concept of a viscosity solutions is useful.

[4] Continuous-time models with Brownian noise naturally lead to second-order PDEs. Higher order PDEs are less relevant for economic applications. You may sometimes encounter first-order PDEs. They can be viewed as degenerate second order PDEs and solved by the same methods (see however the discussion of upwind and monotone schemes below). Alternatively, it may sometimes be useful to apply the "method of characteristics" to gain insights into the solution structure of these equations.

[5] The terminology as it is defined here is also not the most general, but even more generally the classification does not exhaust all possible cases.

(so write (x,t) instead of just x). A quasilinear parabolic equation of the form (4) with a separate time dimension looks like

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(x,t,u,Du)\frac{\partial^2 u}{\partial x_i \partial x_j}(x,t) + G\left(x,t,u,D_x u, \frac{\partial u}{\partial t}\right) = 0, \tag{5}$$

where $D_x u$ is the vector of all first-order spatial derivatives. Because for a parabolic equation the eigenvalue 0 may appear only once, the matrix $(a_{ij}(x,t,y,z))$ must be (positive or negative) definite for all x,t,y,z. If one imposes the additional (stationary) condition $\frac{\partial u}{\partial t} = 0$, the dimensional parabolic equation for a n-dimensional function (in (x,t)) becomes an elliptic equation for a $(n-1)$-dimensional function (in x). For this reason, parabolic and elliptic equations are closely connected.

PDE problems that contain a time dimension are often initial value problems. Unlike for ODEs, the initial value does not just consist of a vector of function values at the initial time t_0, but of an initial function $u_0 : \mathbb{R}^n \to \mathbb{R}$ that assigns an initial value to each point x in the space domain of the problem. If the space domain has boundaries (say is an interval instead of $\mathbb{R}$) there may be additional (potentially time-varying) boundary conditions. The goal is to find a solution u to the PDE that satisfies the initial condition $u(x,t_0) = u_0(x)$ for all x and potentially some boundary conditions for all times $t \geq t_0$. A simple example of an initial value problem without boundary conditions is the heat equation in one space dimension,

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2}$$

with an initial temperature distribution $u(x,0) = u_0(x)$ over the whole real line (for all $x \in \mathbb{R}$). A simple example of an initial value problem with additional boundary conditions is the same heat equation over a finite interval $[\underline{x}, \overline{x}]$ where the temperature at the interval boundaries is held fixed over time, $u(t,\underline{x}) = \underline{u}$, $u(t,\overline{x}) = \overline{u}$ for all $t \geq 0$.

PDE problems without a time dimension are typically pure boundary value problems. The goal is to find a solution u to the PDE that satisfies some additional restrictions on function values or derivatives on the boundary of the solution domain. A simple example is the stationary heat equation

$$0 = a\frac{\partial^2 u}{\partial x^2}$$

over some interval $[\underline{x}, \overline{x}]$ with boundary conditions as in the example above, $u(\underline{x}) = \underline{u}$, $u(\overline{x}) = \overline{u}$.

2.2 Spatial Discretization and Method of Lines

2.2.1 Method of Lines for Parabolic PDEs

Consider a quasilinear parabolic PDE and assume for ease of exposition that the first time derivative enters the equation also linearly such that it can be written in the form

$$\frac{\partial u}{\partial t} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(x,t,u,Du)\frac{\partial^2 u}{\partial x_i \partial x_j}(x,t) + G\left(x,t,u,D_x u\right). \tag{6}$$

By interpreting the function $(x,t) \mapsto u(x,t)$ as a (infinite-dimensional) vector function $t \mapsto (u(x,\cdot))_{x\in\mathbb{R}^n}$, one can conceptually view this equation as an inifinite-dimensional system of ODEs in time and try to use ODE methods to solve the equation. The idea is to use a space discretization to approximate the infinite-dimensional vector $(u(x,\cdot))_{x\in\mathbb{R}^n}$ by some finite-dimensional object. Let $\mathcal{X}\subset\mathbb{R}^n$, $\mathcal{X}$ finite, be a grid of discretization points. For any function $y:\mathbb{R}^n\to\mathbb{R}$ define the vector $\hat{y}=(y(x))_{x\in\mathcal{X}}$ of function values on the grid and denote by $d_i(\hat{y},x)$ some approximation scheme for the first derivative $\frac{\partial y}{\partial x_i}(x)$ with respect to x_i that only uses information in the vector $\hat{y}$ (information of function values on the grid). Here, $x\in\mathcal{X}$ is a grid point. Similarly, let $d^2_{ij}(\hat{y},x)$ be some approximation scheme for second derivatives $\frac{\partial^2 y}{\partial x_i \partial x_j}(x)$. For $n=1$ and a uniform grid $x_0<x_1<\cdots<x_N$, $x_i=x_{i-1}+\Delta x$, common approximation schemes are (let for this example $\hat{y}_i=y(x_i)$)

$$d(\hat{y},x_i)=\begin{cases}\frac{\hat{y}_{i+1}-\hat{y}_{i-1}}{2\Delta x} & \text{(central differences)}\\ \frac{\hat{y}_{i}-\hat{y}_{i-1}}{\Delta x} & \text{(left differences)}\\ \frac{\hat{y}_{i+1}-\hat{y}_{i}}{\Delta x} & \text{(right differences)}\end{cases} \qquad d^2(\hat{y},x_i)=\frac{\hat{y}_{i+1}+\hat{y}_{i-1}-2\hat{y}_i}{\Delta x^2}.$$

Section 3 discusses in more detail how to design appropriate approximation schemes d, d^2. Given a space grid $\mathcal{X}$ and derivative approximation schemes, a discretized version of (6) is

$$\frac{d\hat{u}}{dt}(t)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}\left(x,t,\hat{u}(t),d_1(\hat{u}(t),x),...,d_n(\hat{u}(t),x)\right)d^2_{ij}(\hat{u}(t),x)+G\left(x,t,\hat{u}(t),d_1(\hat{u}(t),x),...,d_n(\hat{u}(t),x)\right), \qquad x\in\mathcal{X},$$

where $\hat{u}:\mathbb{R}\to\mathbb{R}^{|\mathcal{X}|}$. This is a $|\mathcal{X}|$-dimensional first-order ODE and can be solved using any of the ODE methods discussed in section 1.

A very common case is that both the space discretization and the time steps in the ODE solver are based on a finite-difference discretization (this is for example true for the example schemes d and d^2 given above and an Euler method for the time step). Such a solution method is called a finite difference method.

2.2.2 Method of Lines for Elliptic PDEs

Elliptic PDEs are often steady-state/stationary versions of some dynamics that are represented by a parabolic PDE.[6] If one has a good reason to believe that these dynamics eventually converge to the stationary solution,[7] then it may be possible to solve the elliptic equation by solving the underlying (dynamic) parabolic equation until convergence.

If there is no time-dependent representation of the system readily available, one can also convert an elliptic PDE into a parabolic PDE by adding a false time transient: take a generic PDE of the type (5) that does not contain a time variable and replace the 0 on the right-hand side by $\frac{\partial u}{\partial t}$. If the stationary solution has been found, then $\frac{\partial u}{\partial t}=0$ and u solves the original equation, otherwise the modified equation implies a time drift of u. Whether this procedure converges or not depends on the problem at hand.

Sometimes, it may be easier to solve elliptic PDEs directly without introducting an additional time dimension. This is regularly true for linear problems, but may not be the case in nonlinear problems.[8]

[6]Examples are a (stationary) value function of an infinite-horizon decision maker as the (backward) limit of (nonstationary) finite-horizon value functions as the time horizon becomes large or the invariant density of an equilibrium model as the (forward) limit of the equilibrium state distribution as the system runs for a long time.

[7]E.g. when iterating backward on a Bellman equation with discounting, intuition (and in many cases theory) suggests that the process should converge to the infinite-horizon solution. The same is true for continuous-time value function iteration based on the HJB equation.

[8]Even a more "direct" method must solve a nonlinear equation system which usually requires some iterative method. Designing an iterative algorithm based on a related parabolic PDE problem may be superior to a generic Newton-type

2.3 A Simple Linear Example

2.3.1 Implicit and Explicit Euler Methods for the Heat Equation

To make the abstract considerations of the last subsection more concrete and add some details on issues that come up in practice, consider the following simple linear parabolic PDE (one-dimensional heat equation)

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} \tag{7}$$

with some parameter $a > 0$. Suppose this is given together with a initial condition $u(x,0) = u_0(x)$ defined on the real line. To solve this problem using a finite difference method, one typically proceeds as follows:

1. Choose a space discretization, $x_0 < x_1 < \cdots < x_N$ and replace $\frac{\partial^2 u}{\partial x^2}(x_i,t)$ on the right-hand side of the PDE for all i by an expression that only depends on the vector $(u(x_0,t),\ldots,u(x_N,t))$.

   Example: choose the number N of gridpoints, arbitrary boundaries $x_0 < x_N$, and use a uniform grid, $x_i = x_0 + i\Delta x$ with $\Delta x = \frac{x_N - x_0}{N}$. Replace $\frac{\partial^2 u}{\partial x^2}(x_i,t)$ by a three-point finite difference approximation

   $$\frac{\partial^2 u}{\partial x^2}(x_i,t) \approx a\frac{u(x_{i+1},t)+u(x_{i-1},t)-2u(x_i,t)}{\Delta x^2}.$$

2. At the end points x_0 and x_N impose some artificial boundary conditions,[9] ideally using insights about the problem at hand.

   Example: choose constants consistent with the initial value $u(x_0,t) = u_0(x_0)$, $u(x_N,t) = u_0(x_N)$, consistent with long-run decay to 0, $u(x_0,t) = u(x_N,t) = 0$, or something in between (e.g. letting the function on the boundary decay from the initial condition to 0). In the following let $u_L(t)$, $u_R(t)$ be some generic boundary functions.

3. Solve the resulting $N-1$-dimensional ODE initial value problem for the vector function $\hat{u}(t) = (u(x_1,t),\ldots,u(x_{N-1},t))$ using a suitable ODE method, for example implicit or explicit Euler.

The last step is discussed in more detail. Given the choices in the example, the ODE to be solved is

$$\frac{d\hat{u}_i(t)}{dt} = a\frac{\hat{u}_{i+1}(t)+\hat{u}_{i-1}(t)-2\hat{u}_i(t)}{\Delta x^2} \qquad i = 1,\ldots,N-1 \tag{8}$$

and $\hat{u}_0(t) = u_L(t)$, $\hat{u}_N(t) = u_R(t)$ under the initial condition $\hat{u}_i(t) = u_0(x_i)$ for all $i = 1,\ldots,N-1$.

Solving this ODE problem using an explicit Euler discretization in time with time step width Δt means solving in each step the equation

$$\frac{\hat{u}_i(t_{j+1})-\hat{u}_i(t_j)}{\Delta t} = a\frac{\hat{u}_{i+1}(t_j)+\hat{u}_{i-1}(t_j)-2\hat{u}_i(t_j)}{\Delta x^2}$$

equation solver, if one can exploit additional (economic/physical/other) insights about the problem at hand to understand convergence behavior (economic insights often imply that a value function iteration of a single decision maker should converge to an infinite-horizon value function, physical insights may imply that something that looks like a heat diffusion process should eventually reach an equilibrium).

[9]If the problem is stated on a compact space domain and has actual boundary conditions, one should use these. But if the domain is unbounded like $\mathbb{R}$, additional boundary conditions have to imposed to transform the problem to a compact space domain.

for $\hat{u}_i(t_{j+1})$. The solution has an explicit recursive form (just as expected from the discussion in section 1),

$$\hat{u}_i(t_{j+1}) = \hat{u}_i(t_j) + \frac{a\Delta t}{\Delta x^2}\left(\hat{u}_{i+1}(t_j) + \hat{u}_{i-1}(t_j) - 2\hat{u}_i(t_j)\right). \tag{9}$$

For an implicit Euler discretization in time one has to solve for each j the equation system

$$\frac{\hat{u}_i(t_{j+1}) - \hat{u}_i(t_j)}{\Delta t} = a\frac{\hat{u}_{i+1}(t_{j+1}) + \hat{u}_{i-1}(t_{j+1}) - 2\hat{u}_i(t_{j+1})}{\Delta x^2}, \qquad i = 1, ..., N-1$$

Rearranging, writing the system in matrix form and imposing the boundary conditions yields the linear equation

$$\begin{pmatrix} 1+2\frac{a\Delta t}{\Delta x^2} & -\frac{a\Delta t}{\Delta x^2} & & & \\ -\frac{a\Delta t}{\Delta x^2} & 1+2\frac{a\Delta t}{\Delta x^2} & -\frac{a\Delta t}{\Delta x^2} & & \\ & & \ddots & \ddots & \ddots \\ & & & -\frac{a\Delta t}{\Delta x^2} & 1+2\frac{a\Delta t}{\Delta x^2} \end{pmatrix} \begin{pmatrix} \hat{u}_1(t_{j+1}) \\ \hat{u}_2(t_{j+1}) \\ \vdots \\ \hat{u}_{N-2}(t_{j+1}) \\ \hat{u}_{N-1}(t_{j+1}) \end{pmatrix} = \begin{pmatrix} \hat{u}_1(t_j) + \frac{a\Delta t}{\Delta x^2} u_L(t_{j+1}) \\ \hat{u}_2(t_j) \\ \vdots \\ \hat{u}_{N-2}(t_j) \\ \hat{u}_{N-1}(t_{j+1}) + \frac{a\Delta t}{\Delta x^2} u_R(t_{j+1}) \end{pmatrix}. \tag{10}$$

While the equation system (10) is certainly harder to solve than the explicit recursion (9), it is usually not too costly to solve this equation system even for large N, because one can exploit the special tridiagonal structure of the coefficient matrix. This structure also means that the matrix is very sparse for large N (most of its entries are 0) and there is no need to store a dense matrix with $(N-1)^2$ entries.[10] To work with sparse matrices in Matlab, check the function documentation of the functions `sparse` (generic creation function), `sparsealloc` (allocating space for sparse matrix to be populated later), `spdiags` (defining sparse matrices with a particular diagonal structure) and `speye` (sparse identity).

2.3.2 Stability Considerations

Through the ODE representation (8) the stability insights from Problem 3 of Problem Set 1 can be used to assess the stability of the two methods. As you have seen in the numerical examples there, a failure of stability can lead to bad error propagation properties of the algorithm and substantially distort the numerical solution.

The system (8) can be written in matrix form,

$$\frac{d\hat{u}(t)}{dt} = A\hat{u}(t) + b(t)$$

with

$$A = \frac{a}{\Delta x^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 \end{pmatrix}, \qquad b(t) = \frac{1}{\Delta x^2}\begin{pmatrix} u_L(t) \\ 0 \\ \vdots \\ 0 \\ u_R(t) \end{pmatrix}.$$

Ignoring the constant $b(t)$, the associated homogeneous equation $\frac{d\hat{u}(t)}{dt} = A\hat{u}(t)$ looks structurally identical to the one-dimensional test problem $y' = \lambda y$ analyzed in Problem 3. Indeed, the numerical stability of a

[10]The tridiagonal structure is a special case for $n = 1$ (one space dimension). In multiple space dimensions, the equation matrix will still be sparse, but not tridiagonal.

scheme for the vector ODE is related to numerical stability of the (same) scheme for the test equations $y' = \lambda y$, if λ are the eigenvalues of the matrix A. The matrix A has only negative eigenvalues, so by the analysis of Problem 3 one would expect the explicit Euler scheme to be stable, if $1 + \lambda\Delta t$ lies inside the unit circle for all eigenvalues λ of A. Here, it is only relevant to check the smallest eigenvalue, which is given by $\lambda = \frac{2a}{\Delta x^2}\left(\cos\left(\pi\frac{N-1}{N}\right) - 1\right) \approx -\frac{4a}{\Delta x^2}$ for N large.[11] Thus, the explicit Euler scheme is stable, if

$$\frac{a\Delta t}{\Delta x^2} \leq \frac{1}{2}.$$

This condition severely limits the size of the time step: whenever one refines the space grid by increasing the number of grid points by a factor m, the number of grid points in the time grid has to increase by a factor m^2 for the method to remain stable (at least asymptotically for large m). This is particularly bad, if one solves a parabolic, time-dependent problem only to get to the stationary solution of a related elliptic problem, because then one would like to take large time steps (for fast convergence), but have a fine space discretization (for accuracy of the stationary solution).

Because the implicit Euler scheme for ODEs is A-stable, a PDE algorithm based on implicit time steps as in equation (10) is stable regardless of the size of the eigenvalues of A, as long as they remain all negative. In this sense an implicit time discretization is often preferrable to an explicit one for problems with a large diffusion term (large a).[12]

2.3.3 Advection Terms/First-order Derivatives

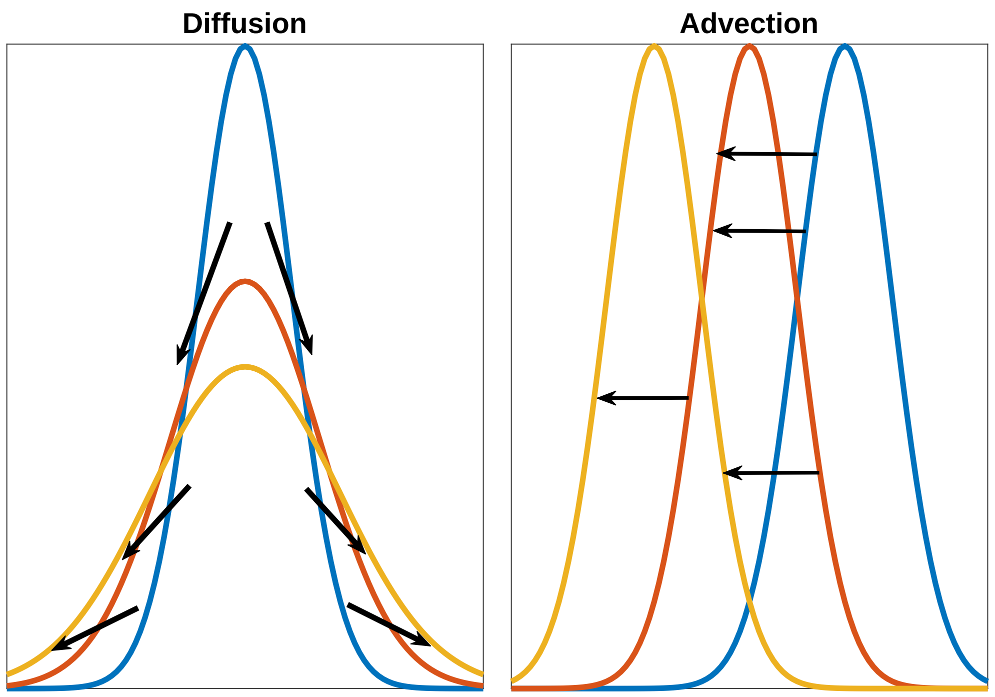
Additional care must be taken in the presence of first-order derivatives in the PDE, even for the linear case. To illustrate this point, consider the simple one-dimensional advection equation

$$\frac{\partial u}{\partial t} = b\frac{\partial u}{\partial x}. \tag{11}$$

The heat equation (7) models undirected movements of particles through diffusion or movement of probability mass through Brownian-type fluctuations, whereas equation (11) models a directed flow/stream of particles (say in a moving fluid) or the movement of probability mass through (deterministic) time drift:

[11] Details of the eigenvalue computation are omitted here.

[12] While the discussion was limited to the simple heat equation (7), these insights apply more generally.



In the physical example, suppose $u(x,t)$ describes some characteristic, say the mass, of the particle located at point x at time t. For $b > 0$, the equation describes a movement to the left at speed b, implying that the very same particle at location x at time t must be located at $x - b\Delta t$ at time $t + \Delta t$. If there is no interaction between different particles that changes the mass of a particle, then the mass observable at location $x - b\Delta t$ at time $t + \Delta t$ should exactly equal the mass observable at location x at time t, i.e. $u(x,t) = u(x - b\Delta t, t + \Delta t)$. This reasoning implies $u(x,t) = u(x + bt, 0) = f(x + bt)$ with some function f and it is easy to check that for any differentiable f, $u(x,t) = f(x + bt)$ solves indeed equation (11).

Now consider an explicit time discretization of (11),

$$u(x, t + \Delta t) = u(x,t) + \Delta t \cdot b u_x(x,t). \tag{12}$$

Using $u(x, t + \Delta t) = f(x + bt + b\Delta t) = u(x + b\Delta t, t)$, this implies

$$u_x(x,t) = \frac{u(x + b\Delta t, t) - u(x,t)}{b\Delta t}.$$

So if one was to choose $\Delta x = b\Delta t$ and approximate $u_x(x,t)$ in (11) by right first differences,

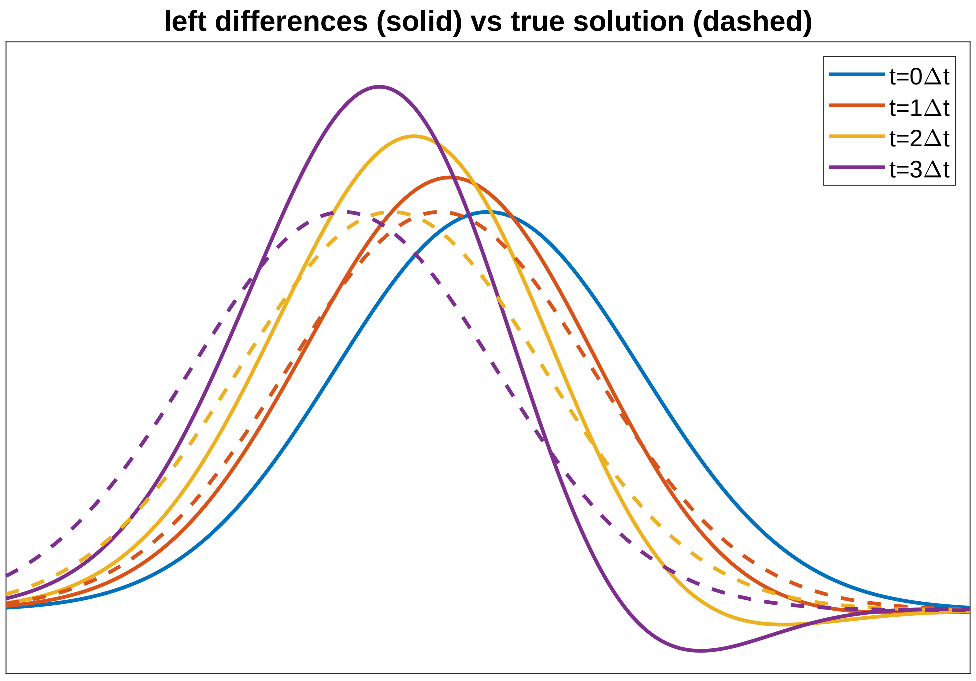
$$u_x(x,t) \approx \frac{u(x + \Delta x, t) - u(x,t)}{\Delta x},$$

then the Euler scheme (11) would give exactly the correct solution. If one had used instead left differences

$$u_x(x,t) \approx \frac{u(x,t) - u(x - \Delta x, t)}{\Delta x},$$

then the Euler scheme (12) would imply (still for $\Delta x = b\Delta t$) $u(x, t+\Delta t) = 2u(x,t) - u(x-\Delta x, t)$. To the extent that $u(x,t) - u(x-\Delta x, t) \approx u(x+\Delta x, t) - u(x,t)$, this does not matter too much, but if there is a substantial difference between the two (because there is substantial curvature in u), the left differencing scheme may introduce a significant error (a similar point applies to central differences). Of course, also the right differences do not lead to the exact solution, if $\Delta x \neq b\Delta t$, but one would nevertheless expect them to yield a better approximation, because one still looks into the right direction when searching for the time t position of the particle that is located at x at time $t + \Delta t$.

The following figure illustrates what could happen in a numerical solution based on an Euler scheme with left differences to approximate first derivatives:



The discussion so far focused on the case $b > 0$ and concluded that right differences are the superior choice. Of course, everything is symmetric in the orientation of the real axis and consequently for $b < 0$ the above discussion would have concluded that left differences are the superior choice. These insights suggest the following simple heuristic may be reasonable in the design of numerical schemes even beyond this simple example:

Upwind scheme: If (in some PDE) the coefficient multiplying $\frac{\partial u}{\partial x}$ is positive in the grid point (t_i, x_i), use right differences to approximate $\frac{\partial u}{\partial x}(t_i, x_i)$, if the coefficient multyplying $\frac{\partial u}{\partial x}$ is negative in (t_i, x_i), use left differences to approximate $\frac{\partial u}{\partial x}(t_i, x_i)$.[13]

[13]More generally, if the PDE is of the form (6) with G nonlinear in $\frac{\partial u}{\partial x}$ instead of a positive/negative coefficient the requirement would be a positive/negative derivative of G with respect to $\frac{\partial u}{\partial x}$.

2.4 Monotone Schemes

Upwind schemes are a special case of schemes that satisfy a certain monotonicity property that turns out to be key for results on the convergence of numerical schemes (to viscosity solutions). In particular, by the Barles-Souganidis theorem under certain conditions on the (parabolic/elliptic) equation, a numerical scheme converges, if it is consistent, stable and monotone.[14] Here, the discussion will be restricted to the monotonicity requirement. The Barles-Souganidis result applies to generic parabolic PDEs of the form

$$\frac{\partial u}{\partial t} = F(x, t, u, Du, D^2 u)$$

under the requirement that F is elliptic, which means in this case that for two symmetric matrices A, B the inequality $A \leq B$ (in the sense that $B - A$ is positive semidefinite) implies $F(x, t, y, z, A) \leq F(x, t, y, z, B)$ for all x, t, y, z.[15] A numerical scheme for this PDE is said to be monotone, if it satisfies a similar monotonicity condition as F.

Let S be a function that describes the numerical scheme in the following sense: for each grid point (x, t), the value $\hat{u}(x, t)$ is defined through the scheme by the equation

$$S(x, t, \hat{u}(x, t), (\hat{u}(\xi, \tau))_{(\xi,\tau)\neq(x,t)}) = 0,$$

where $(\hat{u}(\xi, \tau))_{(\xi,\tau)\neq(x,t)}$ is the vector of approximate solution values on all grid points (ξ, τ) other than (x, t). Examples of such functions are (here, $\hat{u}_{i,j} = \hat{u}(x_i, t_j)$)

$$S^{ee}(x_i, t_{j+1}, \hat{u}_{i,j}, (\hat{u}_{i',j'})_{(i',j')\neq(i,j)}) = a\frac{\hat{u}_{i-1,j} + \hat{u}_{i+1,j} - 2\hat{u}_{i,j}}{\Delta x^2} - \frac{\hat{u}_{i,j+1} - \hat{u}_{i,j}}{\Delta t} \tag{13}$$

for an explicit Euler scheme for equation (7) and

$$S^{ie}(x_i, t_{j+1}, \hat{u}_{i,j}, (\hat{u}_{i',j'})_{(i',j')\neq(i,j)}) = a\frac{\hat{u}_{i-1,j+1} + \hat{u}_{i+1,j+1} - 2\hat{u}_{i,j+1}}{\Delta x^2} - \frac{\hat{u}_{i,j+1} - \hat{u}_{i,j}}{\Delta t} \tag{14}$$

for an implicit Euler scheme for the same equation. A scheme S is called monotone, if it is (weakly) increasing in its last argument (the vector of function values other than at the current grid point).[16]

This section is concluded by checking the monotonicity property for some simple numerical schemes:

1. The scheme S^{ee} defined in (13) for the heat equation:

   Rearranging yields

   $$S^{ee}(\cdots) = \frac{a}{\Delta x^2}\hat{u}_{i-1,j} + \frac{a}{\Delta x^2}\hat{u}_{i+1,j} + \left(\frac{1}{\Delta t} - \frac{2a}{\Delta x^2}\right)\hat{u}_{i,j} - \frac{1}{\Delta t}\hat{u}_{i,j+1}.$$

   The coefficients in front of $\hat{u}_{i-1,j}$ and $\hat{u}_{i+1,j}$ are always positive, the coefficient in front of $\hat{u}_{i,j+1}$ can be ignored (this is the value this equation of the scheme is solving for), so the scheme is monotone, if and only if the coefficient in front of $\hat{u}_{i,j}$ is nonnegative, which is equivalent to

   $$\frac{1}{\Delta t} - \frac{2a}{\Delta x^2} \geq 0 \Leftrightarrow \frac{a\Delta t}{\Delta x^2} \leq \frac{1}{2}.$$

   This condition happens to coincide with the stability condition for the explicit Euler scheme.

[14]Convergence here does not mean that some (false) time transient method based on a parabolic equation converges to a (stationary) solution of an elliptic equation (like in "convergence of value function iteration"), but that the scheme for the PDE converges to the true solution of the full PDE (including transitory dynamics) as grids are refined.

[15]For an equation of the form (6) the definition of ellipticity (for the right-hand side) given above is sufficient for the condition stated here to be satisfied.

[16]"weakly decreasing" works, too: one can just replace S by $-S$.

2. The scheme S^{ie} defined in (14) for the heat equation:

   Rearranging yields

$$S^{ie}(\cdots) = \frac{a}{\Delta x^2}\hat{u}_{i-1,j+1} + \frac{a}{\Delta x^2}\hat{u}_{i+1,j+1} + \frac{1}{\Delta t}\hat{u}_{i,j} + \left(-\frac{1}{\Delta t} - \frac{2a}{\Delta x^2}\right)\hat{u}_{i,j+1}.$$

   Again, the coefficient in front of $\hat{u}_{i,j+1}$ is irrelevant for monotonicity and all other coefficients are positive. Consequently, the implicit Euler scheme for the heat equation is always monotone.

3. Consider three explicit schemes for the advection equation (11)

$$\begin{aligned} S^l(\cdots) &= b\frac{\hat{u}_{i,j} - \hat{u}_{i-1,j}}{\Delta x} - \frac{\hat{u}_{i,j+1} - \hat{u}_{i,j}}{\Delta t} \\ S^c(\cdots) &= b\frac{\hat{u}_{i+1,j} - \hat{u}_{i-1,j}}{2\Delta x} - \frac{\hat{u}_{i,j+1} - \hat{u}_{i,j}}{\Delta t} \\ S^r(\cdots) &= b\frac{\hat{u}_{i+1,j} - \hat{u}_{i,j}}{\Delta x} - \frac{\hat{u}_{i,j+1} - \hat{u}_{i,j}}{\Delta t} \end{aligned}$$

   It is easy to see that S^l can never be monotone for $b > 0$. If $b \leq 0$, S^l is monotone, if and only if $\frac{b}{\Delta x} + \frac{1}{\Delta t} \geq 0 \Leftrightarrow \frac{-b\Delta t}{\Delta x} \leq 1$. Symmetrically, S^r can never be monotone for $b < 0$ and if $b \geq 0$, it is monotone, if and only if $\frac{b\Delta t}{\Delta x} \leq 1$. Clearly, the scheme S^c based on central differencing is never monotone (unless $b = 0$).

4. You should check yourself that for a linear advection-diffusion equation

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

   an implicit Euler method that uses an upwind scheme for first derivatives is always monotone.

2.5 A Hybrid Implicit-Explicit Method for Nonlinear Equations

The discussion so far has concluded that implicit schemes are preferrable, but has restricted attention to linear equations. For nonlinear problems, an implicit scheme requires to solve a (high-dimensional) nonlinear equation in each step. While sometimes feasible, this is often hard.[17] A (very heuristic) idea to overcome this issue is to choose a time discretization that is explicit in the nonlinear parts and implicit in the linear parts, such that it is still sufficient to solve in each time step a sparse linear equation system of the form (10). While I am not aware of any rigorous analysis of the numerical properties of such a scheme, for the economic models studied in this course this procedure works sufficiently well. To illustrate the idea, consider a quasilinear equation of the form (5), but write the function G in a "more linear form",[18]

$$\frac{\partial u}{\partial t} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x,t,u,Du\right)\frac{\partial^2 u}{\partial x_i \partial x_j}(x,t) + \sum_{i=1}^{n} b_i\left(x,t,u,Du\right)\frac{\partial u}{\partial x_i}(x,t) + c\left(x,t,u,Du\right)u(x,t) + d(x,t,u,Du)$$

[17] And it may introduce an additional source of instability, if the equation system has multiple solutions and the solver finds the "wrong" one for some grid values of x.

[18] This can be done in an arbitrary way, but one could also use a Taylor expansion of G. In a particular model, there is often a relatively natural way how to do this.

The idea is now to use an explicit time discretization for the coefficients a, b, c, d,

$$\begin{aligned}
\hat{a}_{ij}(x,t_k) &:= a_{ij}\left(x,t_k,u(x,t_k),Du(x,t_k)\right)\\
\hat{b}_{i}(x,t_k) &:= b_{i}\left(x,t_k,u(x,t_k),Du(x,t_k)\right)\\
\hat{c}(x,t_k) &:= c\left(x,t_k,u(x,t_k),Du(x,t_k)\right)\\
\hat{d}(x,t_k) &:= d\left(x,t_k,u(x,t_k),Du(x,t_k)\right)
\end{aligned}$$

but an implicit time discretization for the remaining linear parts,

$$\frac{u(x,t_{k+1})-u(x,t_k)}{t_{k+1}-t_k} = \sum_{i=1}^{n}\sum_{j=1}^{n}\hat{a}_{ij}(x,t_k)\frac{\partial^2 u}{\partial x_i \partial x_j}(x,t_{k+1})+\sum_{i=1}^{n}\hat{b}_i(x,t_k)\frac{\partial u}{\partial x_i}(x,t_{k+1})+\hat{c}(x,t_k)u(x,t_{k+1})+\hat{d}(x,t_k)$$

After a suitable space discretization, this leads to a (sparse) linear equation system for the time step. The hope is that for sufficiently small time steps the error introduced by the explicit evaluation of the coefficient functions a, b, c, d is not too large.

3 Differentiation Formulas

Given a grid $\mathcal{X} = \{x_0, ..., x_N\} \subset \mathbb{R}$, $x_0 < \cdots < x_N$ and a vector of function values $\hat{u} = (\hat{u}_i)_{i=1,...,N} = (u(x))_{x\in\mathcal{X}}$ on that grid, what is a good way to compute derivatives u' and u'' of the unknown function using just information available from the vector $\hat{u}$?[19] There are mainly two ways to approach this question:

1. Choose a smooth global approximation function v and approximate the unknowns $u'(x_i)$ and $u''(x_i)$ by $v'(x_i)$ and $v''(x_i)$. This approach encompasses many specific methods, for example:

   (a) Choose for v a sum of smooth basis functions with global support and interpolate such that v solves $v(x_i) = u(x_i)$ for $i = 0, ..., N$. Typical choices for v are polynomials or trigonometric polynomials. This is called a spectral method with collocation.

   (b) Choose for v a sum of basis functions that have only small local support and interpolate such that v solves $v(x_i) = u(x_i)$ for $i = 0, ..., N$. An example would be spline interpolation. This is called a finite element method with collocation.

   In these approaches the derivative at every grid point x_i usually depends on the full vector $\hat{u}$. They may thus lead to dense equation systems in implicit time steps.

2. For each reference grid point x_i choose only a small set of adjacent grid points $\mathcal{Y}(x_i) = \{y_1, ..., y_k\} \subset \mathcal{X} \setminus \{x_i\}$ and use only local information $u(x_i), u(y_1), ..., u(y_k)$ around x_i, ignoring all other components of $\hat{u}$. The advantage is that this makes equations to be solved in implicit time steps sparser. There are two ways to work with the restricted information on $\mathcal{Y}(x_i) \cup \{x_i\}$:

   (a) Use any global approximation method on the smaller grid $\mathcal{Y}(x_i) \cup \{x_i\}$, e.g. a polynomial interpolation.

[19] The focus on the one-dimensional case is here for ease of exposition only. Everything applies analogously to multiple dimensions.

(b) Use a Taylor expansion of the unknown u to design a finite difference scheme that is accurate to the highest possible order.

Here, the method 2(b) is discussed in more detail. It leads to idenical results as method 2(a) with polynomial interpolation. The goal is to find a linear differentiation formula

$$d^\mu u(x_i) = \alpha_0^\mu u(x_i) + \sum_{j=1}^{k} \alpha_j^\mu u(y_j)$$

to approximate $u^{(\mu)}(x_i)$ ($\mu = 1, 2$). A m-th order Taylor expansion of u on a point $y \in \mathcal{Y}(x_i)$ around x_i yields

$$u(y) = u(x_i) + \sum_{\nu=1}^{m} \frac{(y - x_i)^\nu}{\nu!} u^{(\nu)}(x_i) + o\left((y - x_i)^m\right).$$

and substituting this into the differentiation formula implies

$$d^\mu u(x_i) = \left(\alpha_0^\mu + \sum_{j=1}^{k} \alpha_j^\mu\right) u(x_i) + \sum_{\nu=1}^{m} \left(\sum_{j=1}^{k} \alpha_j^\mu \frac{(y_j - x_i)^\nu}{\nu!}\right) u^{(\nu)}(x_i) + o\left(\max_j (y_j - x_i)^m\right). \tag{15}$$

The true derivative satisfies

$$u^{(\mu)}(x_i) = \sum_{\nu=0}^{\mu-1} 0 \cdot u^{(\nu)}(x_i) + 1 \cdot u^{(\mu)}(x_i) + \sum_{\nu=\mu+1}^{\infty} 0 \cdot u^{(\nu)}(x_i) \tag{16}$$

In order to maximize the order of convergence to the true derivative as $\max_j (y_j - x_i) \to 0$ (the order of consistency of the approximation), it makes sense to choose the unknown $k+1$ coefficients α_0^μ, α_1^μ, ..., α_k^μ in a way to equalize the first $k+1$ lowest order terms in equations (15) and (16). This results in the linear equation system

$$\alpha_0^\mu + \sum_{j=1}^{k} \alpha_j^\mu = 0$$

$$\sum_{j=1}^{k} \alpha_j^\mu \frac{(y_j - x_i)^\nu}{\nu!} = \begin{cases} 1, & \nu = \mu \\ 0, & \nu \le k, \nu \ne \mu \end{cases}$$

that can be solved for α_0^μ, α_1^μ, ..., α_k^μ. Obviously, this only makes sense for $\mu \le k$.

Examples:

- if $\mathcal{Y}(x_i) = \{x_{i-1}, x_{i+1}\}$ and the grid is uniform, $x_i - x_{i-1} = x_{i+1} - x_i = \Delta x$, solving the coefficient equations leads to

$$d^1 u(x_i) = \frac{u(x_{i+1}) - u(x_{i-1})}{\Delta x},$$
$$d^2 u(x_i) = \frac{u(x_{i+1}) + u(x_{i-1}) - 2u(x_i)}{\Delta x^2},$$

  the standard central differencing formulas for the first and second derivatives.

- if $\mathcal{Y}(x_i) = \{x_{i-1}\}$, then (also for non-uniform grids)

  $$d^1 u(x_i) = \frac{u(x_i) - u(x_{i-1})}{x_i - x_{i-1}},$$

  that is standard left differences. Similarly, $\mathcal{Y}(x_i) = \{x_{i+1}\}$ leads to right differences.