Numerical Analysis Hilary Term 2026
Lecture 12–13: Introduction to numerical methods for initial-value problems

Initial value problems (IVP)[1]**:** These arise everywhere in mathematics where we wish to model the evolution in time of a given system.

Definition 1. *Let $I = [x_0, X] \subset \mathbb{R}$ be a (time) interval and $D \subset \mathbb{R}^d$ be an open subset, where $d \in \mathbb{N}^+$ denotes the space dimension.*

- *A* first-order ordinary differential equation *(ODE) is an equation of the form*
$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}),$$
where the righthand side is a function $\mathbf{f} : I \times D \to \mathbb{R}^d$.

- *An* initial value problem *(IVP) is an ODE with an initial condition, that is,*
$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}), \qquad \mathbf{y}(x_0) = \mathbf{y}_0 .$$
The goal is to find $\mathbf{y}(x)$, either at the final (time) value $x = X$, or any $x \in [x, X]$.

We'll focus on IVPs in this course, but another important class of ODEs is boundary value problems (BVP), where the condition(s) are imposed at e.g. both the initial and final values $x = x$ and X (and more intricate conditions in domains $I \subseteq \mathbb{R}^2$ and $I \subseteq \mathbb{R}^3$).

Picard's Theorem gives sufficient conditions to ensure that the IVP admits a unique solution[2].

Picard's Theorem. Suppose that $\mathbf{f}$ is continuous in a neighborhood $U \subset \mathbb{R}^{1+d}$ of $(x_0, \mathbf{y}_0)$ that contains the (closed) cylinder
$$R = \{(x, \mathbf{y}) : x_0 \le x \le X_M, \|\mathbf{y} - \mathbf{y}_0\| \le Y_M\},$$
where $X_M > x_0$ and $Y_M > 0$ are constants. Suppose also that there exists a positive constant L such that
$$\|\mathbf{f}(x, \mathbf{y}) - \mathbf{f}(x, \mathbf{z})\| \le L\|\mathbf{y} - \mathbf{z}\|$$
holds whenever $(x, \mathbf{y})$ and $(x, \mathbf{z})$ lie in R. Finally, letting
$$M := \max\{\|\mathbf{f}(x, \mathbf{y})\| : (x, \mathbf{y}) \in R\},$$
suppose that $M(X_M - x_0) \le Y_M$. Then, there exists a unique continuously differentiable function
$$[x_0, X_M] \ni x \mapsto \mathbf{y}(x) \in \mathbb{R}^d$$
that is the solution to our IVP.

Remarks about Picard's theorem.

[1] The remaining lecture notes on IVP are based on notes by Patrick Farrell, Yuji Nakatsukasa, Alberto Paganini, and Endre Süli. Many of the figures presented are courtesy of Maike Meier.

[2] In the remainder, we only deal with problems that satisfy the conditions in Picard's theorem, and assume $\mathbf{y}$ is differentiable as many times as we take derivatives. For more details about Picard's theorem, including the proof, we refer to the lecture notes of A1 Differential Equations. Another source is chapter 11 of Nick Trefethen's book *Exploring ODEs*, which is freely available at `http://people.maths.ox.ac.uk/trefethen/Exploring.pdf`

- The solution, if it exists, can be expressed by the integral equation

$$\mathbf{y}(x) = \mathbf{y}(x_0) + \int_{x_0}^{x} \mathbf{f}(t, \mathbf{y}(t))\mathrm{d}t\,, \tag{1}$$

  which is obtained by integrating the IVP, and where the integration is to be understood componentwise.

- Picard's theorem guarantees the existence of a solution only up to a finite time X_M; consider : $y' = y^2$, $y(0) = 1$, which has solution $y(x) = (1-x)^{-1}$ and blows up at $x = 1$.

- If an IVP satisfies the assumptions of Picard's theorem, its solution is stable on the bounded interval $[x_0, X]$. This means that if $\mathbf{y} : [x_0, X] \to D$ solves the IVP

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y})\,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0\,, \tag{2}$$

  and $\tilde{\mathbf{y}} : [x_0, X] \to D$ solves the same ODE with a perturbed initial condition $\tilde{\mathbf{y}}_0$, that is,

$$\tilde{\mathbf{y}}'(x) = \mathbf{f}(x, \tilde{\mathbf{y}})\,, \qquad \tilde{\mathbf{y}}(x_0) = \tilde{\mathbf{y}}_0\,,$$

  then

$$\|\mathbf{y}(x) - \tilde{\mathbf{y}}(x)\| \le e^{L(X-x_0)}\|\mathbf{y}_0 - \tilde{\mathbf{y}}_0\| \quad \forall\, x \in [x_0, X]\,.$$

  (We'll prove a related result in the theorem below.) This implies that a small error in the initial condition does not compromise dramatically the solution of the IVP. This is an important property—small errors in $\mathbf{y}$, either at x_0 or some other $x \in [x_0, X)$, would not stop us from getting an accurate $\mathbf{y}$ at $x = X$. However, note that the constant $e^{L(X-x_0)}$ in the above bound grows exponentially as the final time X increases.

Note. Any higher-order IVP can be reformulated as a larger first-order IVP. The simplest way to do so is to define $\hat{\mathbf{y}} := \left[\ y, y', y'', \dots, y^{(k)}\ \right]^T$ and solve the IVP with respect to $\hat{\mathbf{y}}$. We therefore mainly consider numerical methods for first-order problems. It is also possible to reformulate nonautonomous problems as larger autonomous ones (wherein $\mathbf{y}' = \mathbf{f}(\mathbf{y})$, by working with $\hat{\mathbf{y}} := \begin{bmatrix} x \\ \mathbf{y} \end{bmatrix}$), so sometimes we will restrict ourselves to autonomous ones when it is convenient to do so.

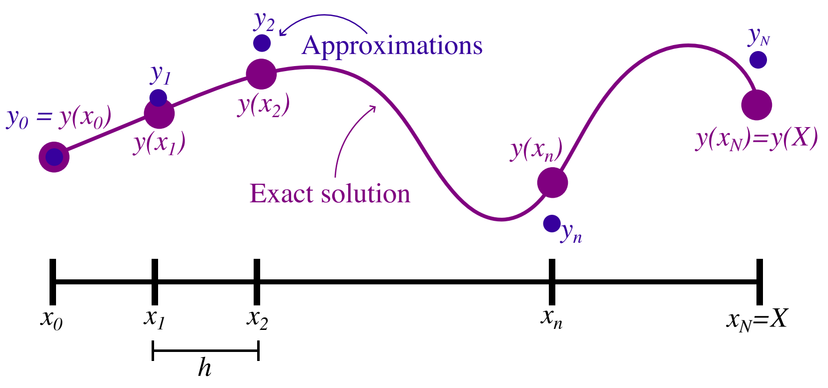
Numerical methods for IVPs: basic idea (Explicit and Implicit methods)

An inconvenient truth is that most IVPs (and most differential equations) cannot be solved analytically (i.e., exactly, to obtain closed-form solutions). It therefore becomes necessary to find approximate solutions with a numerical algorithm. Fortunately, a number of reliable and efficient methods are available for such solution. The remainder of this course is devoted to these methods and their analysis.

Assume that the IVP

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y})\,, \qquad \mathbf{y}(x_0) = \mathbf{y}_0\,,$$

admits a unique stable solution $\mathbf{y} : [x_0, X] \to D$ that is defined on the bounded interval $[x_0, X]$. How can we compute a numerical approximation of $\mathbf{y}$ that can be made arbitrarily accurate? A simple idea is to first divide the interval $[x_0, X]$ into $N \in \mathbb{N}^+$ subintervals defined by the equidistant points $x_n = x_0 + nh$, $n = 0, \ldots, N$, where the *step size* h is $h = (X - x_0)/N$. To each time step x_n, we associate an approximation $\mathbf{y}_n$ of $\mathbf{y}(x_n)$.



To define how to compute these approximations, we take inspiration from a variant of the integral equation (1)

$$\mathbf{y}(x_{n+1}) = \mathbf{y}(x_n) + \int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, \mathrm{d}x\,,$$

This equality suggests that, if we have already computed an approximation $\mathbf{y}_n$ of $\mathbf{y}(x_n)$, we could compute $\mathbf{y}_{n+1}$ by adding to $\mathbf{y}_n$ an approximation of the integral appearing on the right-hand side. There is therefore a deep connection between quadrature and the solution of IVPs. Indeed if $\mathbf{f}(x, \mathbf{y}) = \mathbf{f}(x)$, i.e., $\mathbf{f}$ does not depend on $\mathbf{y}$, then computing $\mathbf{y}$ is a standard quadrature problem, and can be solved by e.g. Gauss quadrature. Starting with $n = 0$, we could iterate such a strategy to compute the entire sequence $\{\mathbf{y}_n\}_{n=0}^N$. In what follows, we construct three different schemes based on three different (and still very similar) approximations of the integral and investigate the impact that this choice has on the properties of the resulting numerical method.

To construct an approximation of the integral, we recall that by the mean value theorem there is a $\xi \in [x_n, x_{n+1}]$ such that

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, \mathrm{d}x = h\mathbf{f}(\xi, \mathbf{y}(\xi))\,.$$

Therefore, we can construct an approximation by replacing ξ with a value s we like. The resulting numerical approximation rule is called a *rectangle rule*. A consequence is that, for any $s \in [x_n, x_{n+1}]$, the approximation error is at most

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, \mathrm{d}x - h\mathbf{f}(s, \mathbf{y}(s)) = h(\mathbf{f}(\xi, \mathbf{y}(\xi)) - \mathbf{f}(s, \mathbf{y}(s))) \leq h \max_{r\in[x_n,x_{n+1}]} |\mathbf{f}(r, \mathbf{y}(r)) - \mathbf{f}(s, \mathbf{y}(s))|.$$

For instance, we can choose $s = x_n$, so that

$$\int_{x_n}^{x_{n+1}} \mathbf{f}(x, \mathbf{y}(x))\, \mathrm{d}x \approx h\mathbf{f}(x_n, \mathbf{y}(x_n))\,.$$

Inserting this gives

$$\mathbf{y}(x_{n+1}) \approx \mathbf{y}(x_n) + h\mathbf{f}(x_n, \mathbf{y}(x_n))\,,$$

which motivates the definition of the *explicit Euler method*[3]

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n, \mathbf{y}_n)\,.$$

Two other interesting choices are $\xi = x_{n+1}$ and $\xi = (x_n + x_{n+1})/2$, which give rise to the *implicit Euler method*

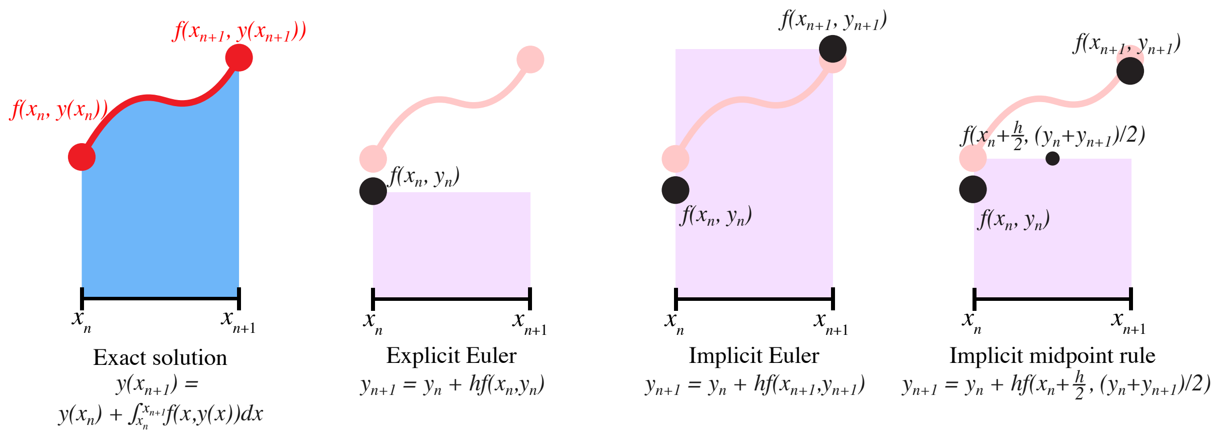
$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_{n+1}, \mathbf{y}_{n+1})$$

and the *implicit midpoint rule*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n + h/2, (\mathbf{y}_n + \mathbf{y}_{n+1})/2)\,,$$

respectively.

Here is an illustration of the three methods.



Note the occurrence of $\mathbf{y}_{n+1}$ on the right-hand side of these last two methods. These numerical methods are called *implicit*, because computing $\mathbf{y}_{n+1}$ requires solving a (generally nonlinear) system, which makes them more computationally expensive than explicit Euler. The arising equations are typically solved with Newton's method, which you saw in M4 Constructive Mathematics. Explicit methods are faster per timestep, but as we will see often suffer from severe timestep restrictions to retain stability, and implicit methods are usually faster for such problems.

Examples. We test these methods on two different examples. First, we consider the linear test case

$$y' = \lambda y\,, \quad y_0 = 1\,, \quad x \in [0,1]\,. \tag{3}$$

For $\lambda = 3$ and $N = 10$, we observe that all three methods compute a qualitatively correct solution, although the one computed with the implicit midpoint rule is way more accurate. Doubling the value of N, we see that the accuracy of the Euler methods improves, although they are not nearly as accurate as the implicit midpoint rule.

[3]The explicit and the implicit Euler methods have been known since 1768! Due, of course, to the great Leonhard Euler.

Next, we investigate what happens for negative values of λ. This case is interesting because the exact solution converges to 0 exponentially as $x \to \infty$. We fix $N = 10$ and investigate different values of λ. For $\lambda \in [-1, -10]$, we see that all methods provide a qualitatively correct solution. For $\lambda < -10$, we see that the explicit Euler solution start oscillating, becoming equioscillatory for $\lambda = -20$, and diverging to $\pm\infty$ for $\lambda < -20$.

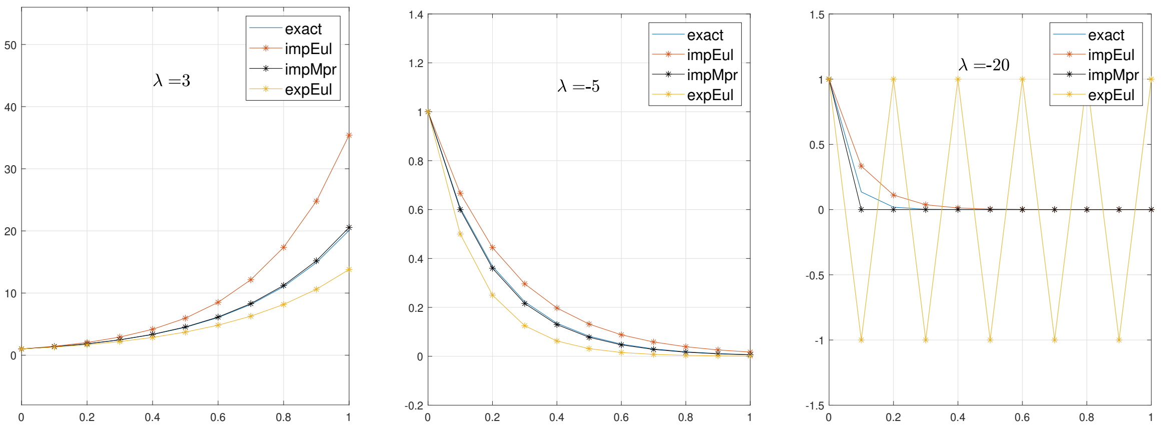


Figure 1: Solving (3) using explicit Euler, midpoint and implicit Euler methods.

For $\lambda < -20$, the solution computed with the implicit midpoint rule also starts to oscillate, although the level of these oscillations cannot be compared with the ones of the explicit Euler method, and the method does not diverge (not even for $\lambda = -9000$). On the other hand, it is surprising to see that the implicit Euler method provides excellent solutions for any negative number of λ. This example shows that the stability of a numerical method can vary drastically. We will explore this further in the upcoming lectures.

The second test case we consider is the following IVP:

$$\mathbf{y}' = \begin{pmatrix} y_2 \\ -y_1 \end{pmatrix}, \quad \mathbf{y}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad x \in [0, 2\pi], \tag{4}$$

whose analytical (exact) solution is $\mathbf{y}(x) = \binom{\cos(x)}{\sin(x)}$. This case is interesting because the quantity $Q(\mathbf{y}) := \|\mathbf{y}(x)\|$ is constant in time. We fix $N = 40$ and plot the orbit (that is, the curve $t \mapsto \mathbf{y}(x)$) of the numerical solutions computed with the three methods above and the evolution of their quantity Q.

The numerical solutions are illustrated in Figure 2. We see that the implicit midpoint rule is the only method that preserves Q, and that it does it up to machine precision! The laws of physics are typically formulated by considering the conservation laws of energy, mass, momentum, etc., and numerical methods that exactly preserve key structural properties of the underlying models are now of prime importance. This line of thinking has led to a beautiful confluence of numerical analysis with geometry and topology, leading to the field called geometric numerical integration (which is off-syllabus).

Convergence and consistency of a one-step method To discuss the convergence and convergence speed of methods, we formally define the class of one-step methods.

Definition 2. *A* one-step method *is a function* $\mathbf{\Psi}$ *that takes the triplet* $(s, \mathbf{y}, h) \subset \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ *and a function* $\mathbf{f}$, *and computes an approximation* $\mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f})$ *of* $\mathbf{y}(s + h)$, *which*

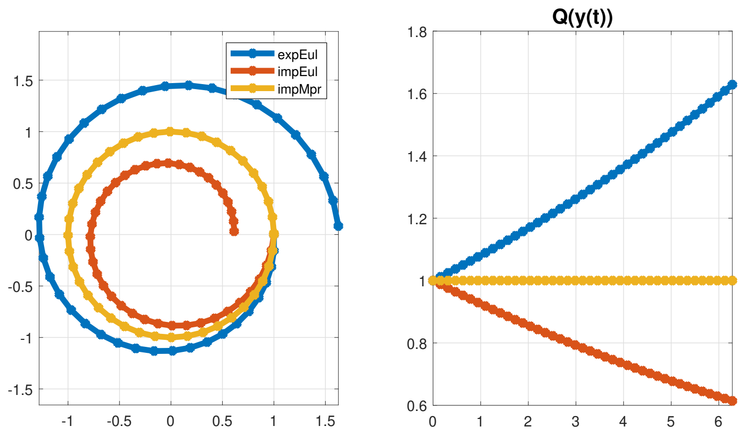


Figure 2: Numerical solutions for (4).

is the solution at $s + h$ of the IVP

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}), \quad \mathbf{y}(s) = \mathbf{y}.$$

Here, we tacitly assume that $\mathbf{y}(s + h)$ exists. Additionally, the timestep h may need to be sufficiently small for $\mathbf{\Psi}$ to be well defined.

Definition 3. *A one-step method $\mathbf{\Psi}$ is said to be* consistent *if*

$$\mathbf{\Psi}(s, \mathbf{y}, 0, \mathbf{f}) = \mathbf{y}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}h}\mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f})|_{h=0} = \mathbf{f}(s, \mathbf{y}).$$

Informally, consistency imposes that the derivative is computed sensibly. It can be seen as a minimum requirement for a good method—the more interesting question is the consistency order, which indicates how quickly the consistency error goes to 0 as $h \to 0$. To examine this, we define

Definition 4. *The* consistency error *(also known as the local truncion error) $\boldsymbol{\tau}$ is defined as*

$$\boldsymbol{\tau}(s, \mathbf{y}, h, \mathbf{f}) := \frac{\mathbf{y}(s + h) - \mathbf{y}}{h} - \frac{\mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f}) - \mathbf{y}}{h} = \frac{\mathbf{y}(s + h) - \mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f})}{h}, \tag{5}$$

where $\mathbf{y}(s + h)$ is the solution at $s + h$ of the IVP.

The following lemma gives additional insight about the definition of consistent one-step method. The gist of it is that a one-step method is consistent if the consistency error can be made arbitrarily small by reducing h.

Lemma. Assume that $h \mapsto \mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f})$ is continuously differentiable in a neighborhood of 0. Then, $\mathbf{\Psi}$ is consistent if and only if, for any fixed $\mathbf{f}$,

$$\|\boldsymbol{\tau}(\tilde{s}, \tilde{\mathbf{y}}, h, \mathbf{f})\| \to 0 \quad \text{as } h \to 0$$

locally uniformly in $(\tilde{s}, \tilde{\mathbf{y}}) \in R$, where R is the cylinder from Picard's Theorem.

It is sometimes convenient to represent an abstract one-step method via its *increment function.*

Lemma. Assume that $h \mapsto \mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f})$ is continuously differentiable in a neighborhood of 0. Then, $\mathbf{\Psi}$ is consistent if and only if there is a continuous increment function $h \mapsto \boldsymbol{\psi}(s, \mathbf{y}, h, \mathbf{f})$ such that

$$\mathbf{\Psi}(s, \mathbf{y}, h, \mathbf{f}) = \mathbf{y} + h\boldsymbol{\psi}(s, \mathbf{y}, h, \mathbf{f}), \quad \boldsymbol{\psi}(s, \mathbf{y}, 0, \mathbf{f}) = \mathbf{f}(s, \mathbf{y}).$$

We also define the *global error*

$$e_n := \mathbf{y}(x_n) - \mathbf{y}_n.$$

Then $e := e_N = \mathbf{y}(x_N) - \mathbf{y}_N$ is the (global) error in the solution at $x = X$. With a good method, we hope $e \to 0$ as the step size $h \to 0$ (and hence $\boldsymbol{\tau} \to 0$).

Theorem. Let $\mathbf{\Psi}$ be a consistent one-step method and assume that its increment function $\boldsymbol{\psi}$ is Lipschitz continuous with respect to $\mathbf{y}$, that is, that there exists a positive constant $L_{\boldsymbol{\psi}}$ such that, for $0 \leq h \leq h_0$ and for the same region R of Picard's theorem,

$$\|\boldsymbol{\psi}(x, \mathbf{y}, h, \mathbf{f}) - \boldsymbol{\psi}(x, \mathbf{z}, h, \mathbf{f})\| \leq L_{\boldsymbol{\psi}} \|\mathbf{y} - \mathbf{z}\| \quad \text{for} (x, \mathbf{y}), (x, \mathbf{z}) \text{ in } R.$$

Then, assuming that $(x_n, \mathbf{y}_n)$ remains in R, it follows that

$$e \leq \left(\frac{\exp\left(L_{\boldsymbol{\psi}}(x_N - x_0)\right) - 1}{L_{\boldsymbol{\psi}}} \right) \max_{n=0,\ldots,N-1} \|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\|.$$

Proof. For a generic $n \in \{1, \ldots, N-1\}$,

$$\begin{aligned}
e_{n+1} &= \|\mathbf{y}(x_{n+1}) - \mathbf{y}_{n+1}\|, \\
&= \|\mathbf{y}(x_{n+1}) - \mathbf{\Psi}(x_n, \mathbf{y}_n, h, \mathbf{f})\|, \\
&= \|\mathbf{y}(x_{n+1}) - \mathbf{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f}) + \mathbf{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f}) - \mathbf{\Psi}(x_n, \mathbf{y}_n, h, \mathbf{f})\|, \\
&\leq \|\mathbf{y}(x_{n+1}) - \mathbf{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + \|\mathbf{\Psi}(x_n, \mathbf{y}(x_n), h, \mathbf{f}) - \mathbf{\Psi}(x_n, \mathbf{y}_n, h, \mathbf{f})\|, \\
&= h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + \|\left(\mathbf{y}(x_n) + h\boldsymbol{\psi}(x, \mathbf{y}(x_n), h, \mathbf{f})\right) - \left(\mathbf{y}_n + h\boldsymbol{\psi}(x, \mathbf{y}_n, h, \mathbf{f})\right)\|, \\
&\leq h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + \|\mathbf{y}(x_n) - \mathbf{y}_n\| + h\|\boldsymbol{\psi}(x, \mathbf{y}(x_n), h, \mathbf{f}) - \boldsymbol{\psi}(x, \mathbf{y}_n, h, \mathbf{f})\|, \\
&= h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + e_n + h\|\boldsymbol{\psi}(x, \mathbf{y}(x_n), h, \mathbf{f}) - \boldsymbol{\psi}(x, \mathbf{y}_n, h, \mathbf{f})\|, \\
&\leq h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + e_n + hL_{\boldsymbol{\psi}}\|\mathbf{y}(x_n) - \mathbf{y}_n\|, \\
&= h\|\boldsymbol{\tau}(x_n, \mathbf{y}(x_n), h, \mathbf{f})\| + (1 + hL_{\boldsymbol{\psi}})e_n.
\end{aligned}$$

Iterating recursively, this implies that (note that $e_0 = 0$)

$$\begin{aligned}
e_{n+1} &\leq (1 + hL_{\boldsymbol{\psi}})^{n+1} e_0 + h \textstyle\sum_{k=0}^{n} (1 + hL_{\boldsymbol{\psi}})^k \max_{m=0,\ldots,n} \|\boldsymbol{\tau}(x_m, \mathbf{y}(x_m), h, \mathbf{f})\| \\
&= \tfrac{(1+hL_{\boldsymbol{\psi}})^{n+1} - 1}{L_{\boldsymbol{\psi}}} \max_{m=0,\ldots,n} \|\boldsymbol{\tau}(x_m, \mathbf{y}(x_m), h, \mathbf{f})\|.
\end{aligned}$$

To conclude the proof, note that $1 + hL_{\boldsymbol{\psi}} \leq \exp hL_{\boldsymbol{\psi}}$. □

Note that the global error accounts for the accumulation of errors over all steps, wheres the consistency error measures the error in a single step.

Order of accuracy and consistency order. In the following lectures we will examine methods that can give higher accuracy than Euler's methods. To explore these, let us state a few definitions. A method is said to have the *order of accuracy* (or just *order*) p if $e \le Ch^p$ for some constant C; $p > 0$ is usually an integer. The related notion of *consistency order* is the largest $\tilde{p}$ such that the consistency error $\|\boldsymbol{\tau}(s, \mathbf{y}, h, \mathbf{f})\| \le \tilde{C}h^{\tilde{p}}$ for some $\tilde{C}$. The consistency order $\tilde{p}$ measures the local error, whereas p does the global error; they are usually the same, as the above theorem suggests. Informally, at each step the computation incurs an error of $O(h^{\tilde{p}+1})$ (note the $+1$ from the definition (5)), and we perform $O(h^{-1})$ steps, so the global error is $O(h^{\tilde{p}}) = O(h^p)$; the theorem above makes this precise.

In the next lectures we will study two classes of methods—Runge-Kutta and multistep methods—that can achieve order of accuracy higher than 1. These usually result in more accurate solutions for a fixed computational budget. In studying these methods, the two overarching questions we shall address are:

(i) *Convergence*: Does the method converge to the exact solution as $h \to 0$?

(ii) *Stability*: How small does h need to be for the computed solution to start resembling the exact solution?

The theorem above shows that the answer to (i) is straightforward for one-step methods (including Runge-Kutta methods); roughly, convergence holds as $h \to 0$ if the method is consistent. The second question (ii) is intricate (as we saw already in the discussion of the test problem (3)), and we'll explore this question for Runge-Kutta methods in the next lecture.

For multistep methods, even (i) is not obvious; we'll treat both questions in the final lectures.

Listing 1: l11_ivp1.m

```
clear, set(0,'DefaultFigureWindowStyle','docked')
N = 10; h = 1/N; lambda = -20; %modify these parameters to experiment
expEul = nan(1, N+1); impEul = nan(1, N+1); impMpr = nan(1, N+1);
y0 = 1; expEul(1) = y0; impEul(1) = y0; impMpr(1) = y0;

for ii = 1:N
    expEul(ii+1) = expEul(ii)*(1+h*lambda);
    impEul(ii+1) = impEul(ii)/(1-h*lambda);
    impMpr(ii+1) = impMpr(ii)*(1+h*lambda/2)/(1-h*lambda/2);
end

t = linspace(0, 1, N+1);figure(1);
plot(t, exp(lambda*t), t, impEul, '*-', t, impMpr, 'k*-',t, expEul, '*-')
legend({'exact', 'impEul', 'impMpr','expEul'})
```

Listing 2: l11_ivp2.m

```
clear, set(0,'DefaultFigureWindowStyle','docked')
N = 40; T = 2*pi; h = T/N; A = [0 1; -1 0];
```

```
expEul = nan(2, N+1); impEul = nan(2, N+1); impMpr = nan(2, N+1);
y0 = [1; 0]; expEul(:,1) = y0; impEul(:,1) = y0; impMpr(:,1) = y0;

for ii = 1:N
    expEul(:,ii+1) = (eye(2)+h*A)*expEul(:,ii);
    impEul(:,ii+1) = (eye(2)-h*A)\impEul(:,ii);
    impMpr(:,ii+1) = (eye(2)-h*A/2)\((eye(2)+h*A/2)*impMpr(:,ii));
end
figure(2); subplot(1,2,1);
plot(expEul(1,:), expEul(2,:), '*-', 'linewidth', 4);
plot(impEul(1,:), impEul(2,:), '*-', 'linewidth', 4);
plot(impMpr(1,:), impMpr(2,:), '*-', 'linewidth', 4);
axis equal
legend({'expEul', 'impEul', 'impMpr'})
subplot(1,2,2), t = linspace(0, T, N+1); Q = @(y) sqrt(sum(y.^2, 1));
plot(t, Q(expEul), '*-', t, Q(impEul), '*-', t, Q(impMpr), '*-','linewidth', 4)
title('Q(y(t))','FontSize',24);
fprintf('max(abs(Q(impMpr)-1)) = %e\n', max(abs(Q(impMpr)-1)))
```