WI4014TU – Numerical Analysis for PDE's

Lecture 11
Numerical time integration, Forward- and Backward-Euler methods, convergence, stability, solution of the diffusion equation

Time-dependent problems

Diffusion equation:

$$\frac{\partial u}{\partial t} - \nabla \cdot (k \nabla u) = f$$

Wave equation:

$$\frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} - \Delta u = f$$

Unknown function $u(x, y, t) \Rightarrow$ grid values $u(x_i, y_j, t)$ stored in a vector $\boldsymbol{u}(t) \Rightarrow$ solution vector changes in time. How to compute the time evolution of $\boldsymbol{u}(t)$?

Numerical time integration

First-order time evolution

Let $\boldsymbol{u}(t) \in \mathbb{R}^n$ be a time-evolving vector and $\boldsymbol{f}(t, \boldsymbol{u}) \in \mathbb{R}^n$ a vector function depending on both time t and the vector $\boldsymbol{u}$. Consider a system of the first-order ODE's

$$\frac{d\boldsymbol{u}}{dt} = \boldsymbol{f}(t, \boldsymbol{u}), \quad t \geq t_0, \quad \boldsymbol{u}(t_0) = \boldsymbol{u}_0$$

$$\frac{d}{dt}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1(t, u_1, u_2, \dots) \\ f_2(t, u_1, u_2, \dots) \\ \vdots \end{bmatrix}$$

For example, a linear system of ODE's

$$\boldsymbol{u}' = -A\boldsymbol{u} + \boldsymbol{s},$$

where A is some matrix. But it can also be a nonlinear function $\boldsymbol{f}$.

Prediction by time integration

We know $\boldsymbol{u}(t_0)$ – initial condition. Suppose we want to compute $\boldsymbol{u}(t_0 + h)$, for some $h > 0$. Integrating the ODE from t_0 to $t_0 + h$, and using the Fundamental Theorem of Calculus (FTC), we get

$$\int_{t_0}^{t_0+h} \boldsymbol{u}' \, dt = \int_{t_0}^{t_0+h} \boldsymbol{f}(t, \boldsymbol{u}(t)) \, dt$$

$$\boldsymbol{u}(t_0 + h) - \boldsymbol{u}(t_0) = \int_{t_0}^{t_0+h} \boldsymbol{f}(t, \boldsymbol{u}(t)) \, dt$$

$$\boldsymbol{u}(t_0 + h) = \boldsymbol{u}(t_0) + \int_{t_0}^{t_0+h} \boldsymbol{f}(t, \boldsymbol{u}(t)) \, dt$$

Numerical approximation

Hence, the problem of finding $\boldsymbol{u}(t_0 + h)$ reduces to the problem of finding a good numerical approximation (quadrature) for the integral of $\boldsymbol{f}(t, \boldsymbol{u})$:

$$\int_{t_0}^{t_0+h} \boldsymbol{f}(t, \boldsymbol{u}(t))\, dt \approx \sum_{k=1}^{K} w_k\, \boldsymbol{f}(t_k, \boldsymbol{u}(t_k)), \quad t_0 \le t_k \le t_0 + h$$

Having found $\boldsymbol{u}(t_0 + h)$ we can apply the same procedure to find $\boldsymbol{u}(t_0 + h + h)$, etc. – time-iterations (recurrence).

Forward-Euler (Euler's Forward) method



Single-point approximation of the integral at the starting point:

$$\int_{t_0}^{t_0+h} \boldsymbol{f}(t, \boldsymbol{u}(t))\, dt \approx h\, \boldsymbol{f}(t_0, \boldsymbol{u}(t_0))$$

Explicit one-step prediction:

$$\boldsymbol{u}(t_0 + h) \approx \boldsymbol{u}(t_0) + h\, \boldsymbol{f}(t_0, \boldsymbol{u}(t_0))$$

Forward-Euler (FE) iterations:

$$\boldsymbol{u}(t_k + h) \approx \boldsymbol{u}(t_k) + h\, \boldsymbol{f}(t_k, \boldsymbol{u}(t_k)), \quad k = 0, 1, \ldots$$

Backward-Euler (Euler's Backward) method

Single-point approximation of the integral at the end point:

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(t, \boldsymbol{u}(t))\, dt \approx h\, \boldsymbol{f}(t_{k+1}, \boldsymbol{u}(t_{k+1}))$$

Implicit BE iterations

$$\boldsymbol{u}(t_{k+1}) \approx \boldsymbol{u}(t_k) + h\, \boldsymbol{f}(t_{k+1}, \boldsymbol{u}(t_{k+1})), \quad k = 0, 1, \ldots$$

- This requires solving a vector equation to find $\boldsymbol{u}(t_{k+1})$. The Backward-Euler is an *implicit* method, whereas, the Forward-Euler is an *explicit* method

Trapezoidal method

Two-point approximation of the integral:

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(t, \boldsymbol{u}(t))\, dt \approx \frac{1}{2} h \left[\boldsymbol{f}(t_k, \boldsymbol{u}(t_k)) + \boldsymbol{f}(t_{k+1}, \boldsymbol{u}(t_{k+1})) \right]$$

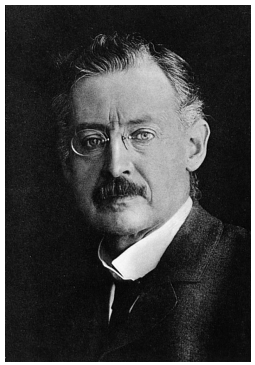
Implicit trapezoidal iterations

$$\boldsymbol{u}(t_{k+1}) \approx \boldsymbol{u}(t_k) + \frac{1}{2} h \left[\boldsymbol{f}(t_k, \boldsymbol{u}(t_k)) + \boldsymbol{f}(t_{k+1}, \boldsymbol{u}(t_{k+1})) \right], \quad k = 0, 1, \ldots$$

Other integration methods

$$\int_{t_0}^{t_0+h} \boldsymbol{f}(t, \boldsymbol{u}(t))\, dt \approx \sum_{i=1}^{k} w_i\, \boldsymbol{f}(t_k, \boldsymbol{u}(t_k)), \quad t_0 \le t_k \le t_0 + h$$

- Multistep methods: Adams-Bashforth, backward differentiation formulae (BDF)
- Explicit and implicit Runge-Kutta methods (ERK, IRK)




Convergence and order of time integration methods

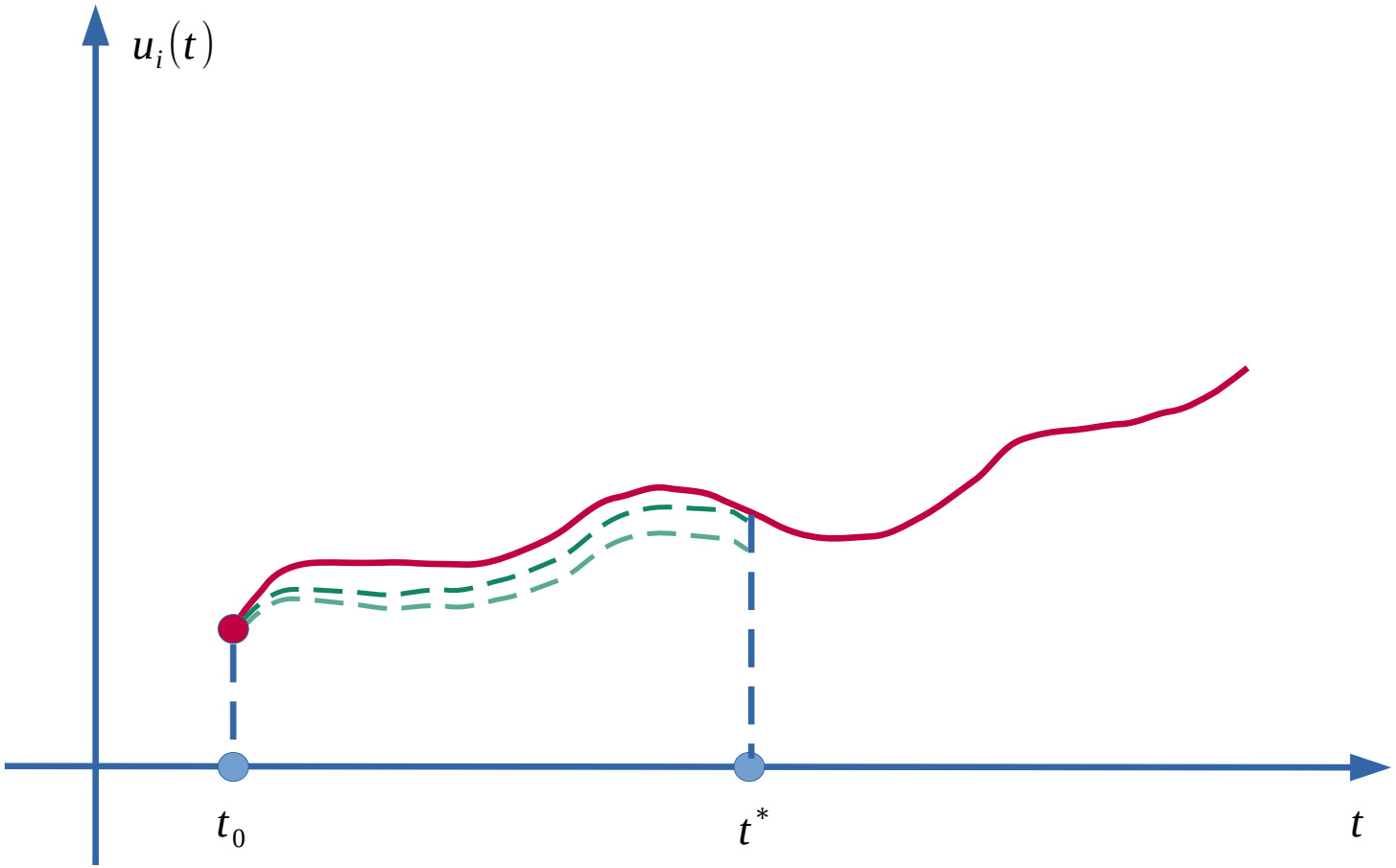
Convergence of time integration methods

- Suppose we have applied some time integration method on the interval $t \in [t_0, t^*]$ with the time step $h = (t^* - t_0)/K$, and obtained the numerical approximation $\boldsymbol{u}_h(t_k)$ of the actual $\boldsymbol{u}(t_k)$, for all $k = 0, 1, \ldots, K$.
- We say that the method is *convergent* if for every $\boldsymbol{f}(t, \boldsymbol{u})$ and every $t^* > t_0$ it is true that

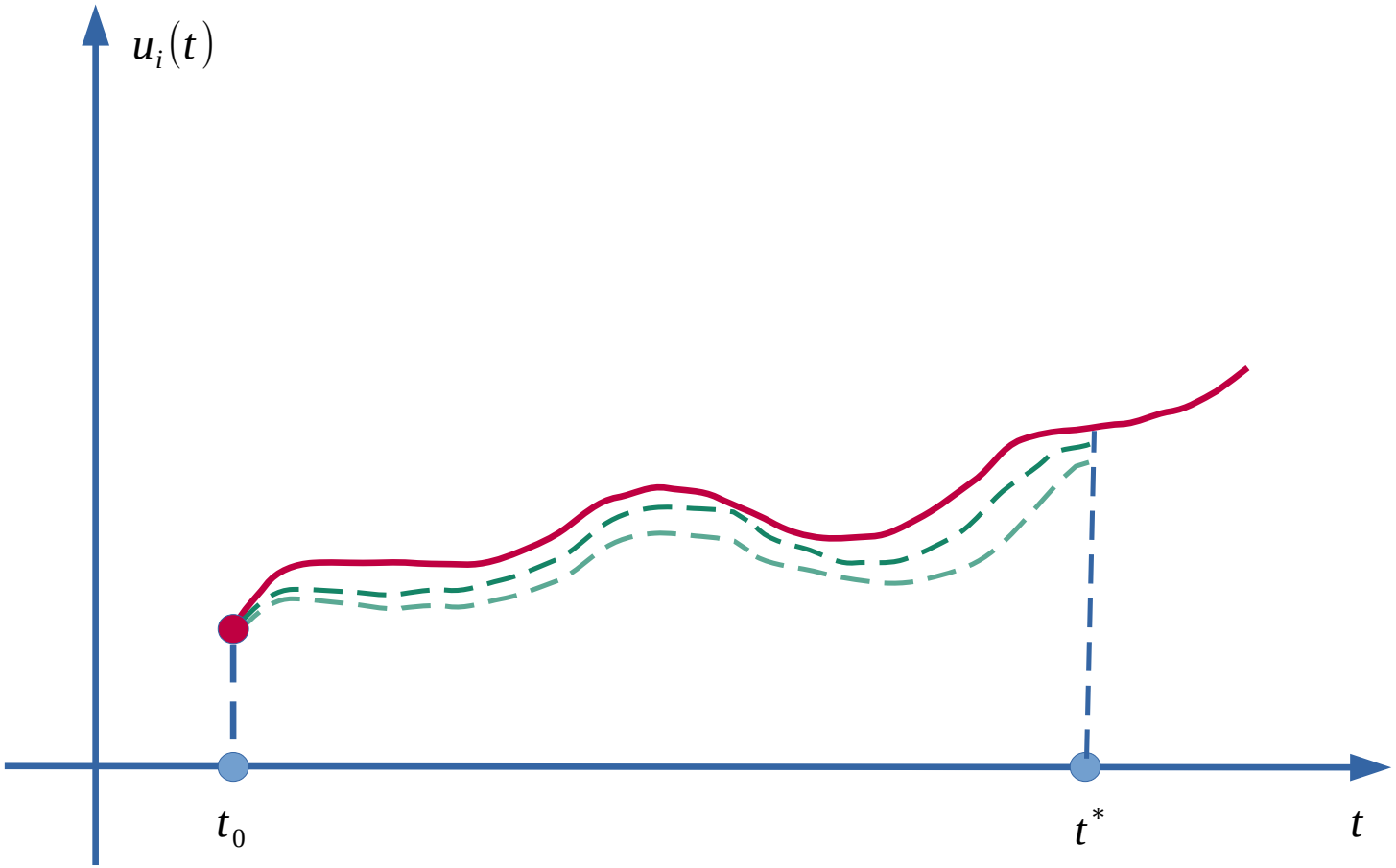
$$\lim_{h \to 0^+} \max_{k=0,1,\ldots,K} \|\boldsymbol{u}_h(t_k) - \boldsymbol{u}(t_k)\| = 0$$

  Global max-norm converges to zero as time step h decreases for any finite t^*.

Idea of convergence



Idea of convergence



Convergence and global error of the Forward-Euler method

Theorem: Let $\boldsymbol{u}(t)$ be continuously differentiable and $\boldsymbol{f}(t,\boldsymbol{u})$ a *Lipschitz* function, i.e., there is $\lambda > 0$ such that $\|\boldsymbol{f}(t,\boldsymbol{u}) - \boldsymbol{f}(t,\boldsymbol{v})\| \leq \lambda \|\boldsymbol{u} - \boldsymbol{v}\|$ for all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$. Then, the Forward-Euler method is convergent.

Proof: Denoting $\boldsymbol{e}_h(t_k) = \boldsymbol{u}_h(t_k) - \boldsymbol{u}(t_k)$

$$\boldsymbol{u}(t_{k+1}) = \boldsymbol{u}(t_k) + h\boldsymbol{u}'(t_k) + \mathcal{O}(h^2) = \boldsymbol{u}(t_k) + h\boldsymbol{f}(t_k, \boldsymbol{u}(t_k)) + \mathcal{O}(h^2)$$

$$\boldsymbol{e}_h(t_{k+1}) = \boldsymbol{e}_h(t_k) + h\left[\boldsymbol{f}(t_k, \boldsymbol{u}(t_k) + \boldsymbol{e}_h(t_k)) - \boldsymbol{f}(t_k, \boldsymbol{u}(t_k))\right] + \mathcal{O}(h^2)$$

$$\|\boldsymbol{e}_h(t_{k+1})\| \leq \|\boldsymbol{e}_h(t_k)\| + h\|\boldsymbol{f}(t_k, \boldsymbol{u}(t_k) + \boldsymbol{e}_h(t_k)) - \boldsymbol{f}(t_k, \boldsymbol{u}(t_k))\| + Ch^2$$

$$\|\boldsymbol{e}_h(t_{k+1})\| \leq (1 + h\lambda)\|\boldsymbol{e}_h(t_k)\| + Ch^2, \quad k = 0, 1, \ldots n$$

$$\|\boldsymbol{e}_h(t_k)\| \leq \frac{C}{\lambda} h\left[(1 + h\lambda)^k - 1\right] < \frac{C}{\lambda} h\left(e^{kh\lambda} - 1\right) \leq \frac{C}{\lambda}\left(e^{\lambda t^*} - 1\right) h$$

$$\lim_{h \to 0^+} \max_k \|\boldsymbol{e}_h(t_k)\| \leq \frac{C}{\lambda}\left(e^{\lambda t^*} - 1\right) \lim_{h \to 0^+} h = 0$$

Local and global orders of time integration methods

- We say that the time integration method is *locally of order* p if

$$\boldsymbol{u}_h(t_1) - \boldsymbol{u}(t_1) = \mathcal{O}(h^p)$$

- Method of local order p recovers exactly any polynomial of degree $p-1$ or less
- Forward Euler method is locally $\mathcal{O}(h^2)$:

$$\begin{aligned}\boldsymbol{u}_h(t_1) - \boldsymbol{u}(t_1) = \boldsymbol{u}_h(t_0) + h\boldsymbol{f}(t_0, \boldsymbol{u}_h(t_0)) + \mathcal{O}(h^2) \\ - \boldsymbol{u}(t_0) + h\boldsymbol{f}(t_0, \boldsymbol{u}(t_0)) - \mathcal{O}(h^2) = \mathcal{O}(h^2)\end{aligned}$$

since $\boldsymbol{u}_h(t_0) = \boldsymbol{u}(t_0)$. *Global* error typically drops to order $p-1$, and convergence analysis shows that the global error is, indeed, proportional to h.

Convergence and order of other methods

- Backward Euler method is convergent and locally $\mathcal{O}(h^2)$, globally $\mathcal{O}(h)$.
- Trapezoidal rule is an implicit, convergent, locally $\mathcal{O}(h^3)$ and globally $\mathcal{O}(h^2)$ method

$$\|\boldsymbol{e}_h(t_k)\| \leq \frac{Ch^2}{\lambda} \exp\left(\frac{\lambda t^*}{1-\lambda h/2}\right), \quad k = 0, 1, \ldots, K$$

Method of lines for diffusion equation

Method of lines discretizes in space, but keeps time continuous. Serves as a theoretical benchmark in stability analysis of numerical time-integration methods.

With FDM/FVM spatial discretization

Later we shall see that after FDM discretization in space: $-\Delta u(\boldsymbol{x}, t) \to A\boldsymbol{u}(t)$, $\boldsymbol{u}(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, the diffusion equation becomes the system of ODE's:

$$\boldsymbol{u}' = -A\boldsymbol{u}, \quad \boldsymbol{u}(0) = \boldsymbol{u}_0$$

Assume $A = V\Lambda V^{-1}$ exists. Then, $\boldsymbol{u}(t) = V\boldsymbol{b}(t)$, and $\boldsymbol{b}(t) = V^{-1}\boldsymbol{u}(t)$.

$$\begin{aligned}
V\boldsymbol{b}' &= -V\Lambda V^{-1}\boldsymbol{u}, \quad \boldsymbol{u}_0 = V\boldsymbol{b}_0 \\
\boldsymbol{b}' &= -\Lambda\boldsymbol{b}, \quad \boldsymbol{b}(0) = V^{-1}\boldsymbol{u}_0, \\
b_j'(t) &= -\lambda_j b_j(t), \quad j = 1, \ldots, n \\
b_j(t) &= b_j(0)e^{-\lambda_j t}, \quad j = 1, \ldots, n \\
\boldsymbol{b}(t) &= e^{-\Lambda t}\boldsymbol{b}(0) = e^{-\Lambda t}V^{-1}\boldsymbol{u}(0) \\
\boldsymbol{u}(t) &= Ve^{-\Lambda t}V^{-1}\boldsymbol{u}(0)
\end{aligned}$$

This is the exact solution of the discrete time-evolution problem.

Case of (negative) FD Laplacian

For $-\Delta \to A$ with the FDM method the eigenvalues are positive $\lambda_j > 0$, $j = 1, \ldots, n$. Writing out $\boldsymbol{u}(t) = V\boldsymbol{b}(t)$ and using the method-of-lines to find $\boldsymbol{u}(t)$, we get

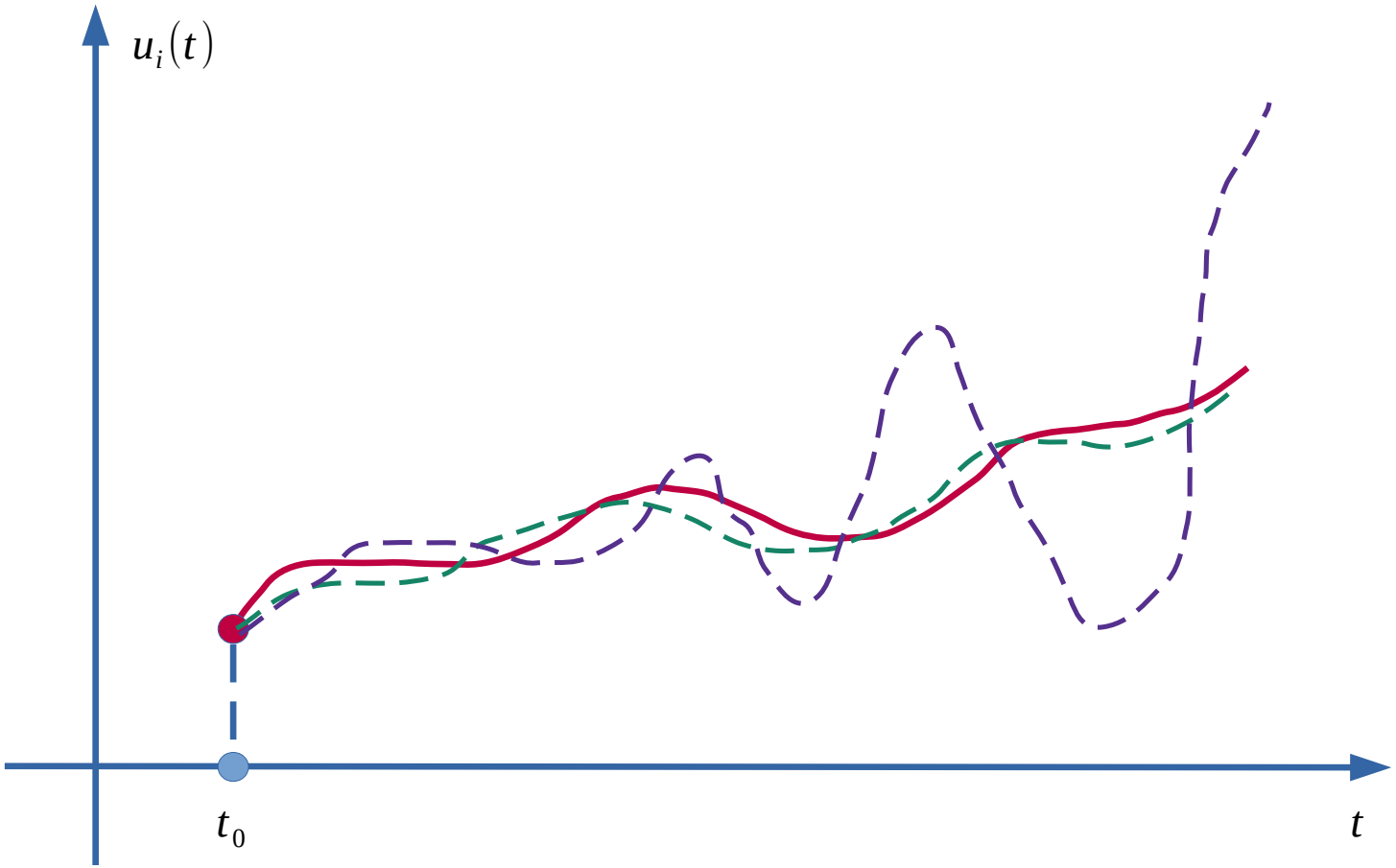
$$\boldsymbol{u}(t) = \sum_{j=1}^{n} b_j(t)\boldsymbol{v}_j$$

$$b_j(t) = b_j(0)e^{-\lambda_j t}$$

Shows that the amplitudes $b_j(t)$ of all expansion terms $b_j(t)\boldsymbol{v}_j$ of the solution decay exponentially in time.

Stability of time integration methods

Idea of stability



Stability of the forward Euler method

After FDM/FVM discretization the diffusion equation becomes

$$\boldsymbol{u}' = -A\boldsymbol{u}, \quad \boldsymbol{u}(t_0) = \boldsymbol{u}_0$$

the Forward-Euler method with the time step h produces iterations $k = 0, 1, \ldots$:

$$\boldsymbol{u}_h(t_k) = \boldsymbol{u}_h(t_{k-1}) - hA\boldsymbol{u}_h(t_{k-1}) = [I - hA]\boldsymbol{u}_h(t_{k-1}) = [I - hA]^k \boldsymbol{u}_0$$

Let A be diagonalizable: $A = V\Lambda V^{-1}$ with eigenvalues λ_j, $j = 1, \ldots, n$.

$$\boldsymbol{u}(t_k) = Ve^{\Lambda t_k} V^{-1}\boldsymbol{u}_0 = \sum_{j=1}^{n} b_j(0)e^{-\lambda_j t_k}\boldsymbol{v}_j \quad \text{(method of lines)}$$

$$\boldsymbol{u}_h(t_k) = [I - hA]^k \boldsymbol{u}_0 = V[I - h\Lambda]^k V^{-1}\boldsymbol{u}_0 = \sum_{j=1}^{n} b_j(0)(1 - h\lambda_j)^k \boldsymbol{v}_j$$

Hence, for $\lambda_j > 0$, $e^{-\lambda_j t_k}$ always decays with t_k, but $|1 - h\lambda_j|^k$ may be growing with k if h is too large. The Forward-Euler method is not always stable.

Conditional stability of the forward Euler method

Deriving the condition for the stability of the Forward-Euler method:

$$
\begin{aligned}
|1 - h\lambda_j| &< 1 \\
-1 < 1 - h\lambda_j &< 1 \\
-2 < -h\lambda_j &< 0 \\
0 < h\lambda_j &< 2 \\
0 < h &< \frac{2}{\lambda_j}, \quad j = 1, \ldots, n
\end{aligned}
$$

It is common to denote the time step h as h_t or Δt:

$$
0 < \Delta t < \frac{2}{\max_j \lambda_j}
$$

Unconditional stability of Backward-Euler method

Backward-Euler method with $\Delta t = h$ produces iterations $k = 0, 1, \ldots$:

$$\boldsymbol{u}_h(t_k) = \boldsymbol{u}_h(t_{k-1}) - hA\boldsymbol{u}_h(t_k)$$
$$[I + hA]\boldsymbol{u}_h(t_k) = \boldsymbol{u}(t_{k-1})$$
$$\boldsymbol{u}_h(t_k) = [I + hA]^{-1}\boldsymbol{u}_{t_{k-1}} = [I + hA]^{-k}\boldsymbol{u}_0$$
$$\boldsymbol{u}_h(t_k) = [I + hA]^{-k}\boldsymbol{u}_0 = V[I + h\Lambda]^{-k}V^{-1}\boldsymbol{u}_0 = \sum_{j=1}^{n} b_j(0)\frac{1}{(1 + h\lambda_j)^k}\boldsymbol{v}_j$$

Hence, for $\lambda_j > 0$, the Backward-Euler method is stable for any $h > 0$

Unconditional stability of trapezoidal method

Trapezoidal method with $\Delta t = h$ produces iterations $k = 0, 1, \ldots$:

$$\boldsymbol{u}_h(t_k) = \boldsymbol{u}_h(t_{k-1}) - \frac{1}{2} hA \left[\boldsymbol{u}_h(t_{k-1}) + \boldsymbol{u}_h(t_k)\right]$$

$$\left[I + \frac{h}{2}A\right] \boldsymbol{u}_h(t_k) = \left[I - \frac{h}{2}A\right] \boldsymbol{u}(t_{k-1})$$

$$\boldsymbol{u}_h(t_k) = \left[I + \frac{h}{2}A\right]^{-1} \left[I - \frac{h}{2}A\right] \boldsymbol{u}_{t_{k-1}} = \left[I - \frac{h}{2}A\right]^{k} \left[I + \frac{h}{2}A\right]^{-k} \boldsymbol{u}_0$$

$$\boldsymbol{u}_h(t_k) = \sum_{j=1}^{n} b_j(0) \left(\frac{1 - \lambda_j h/2}{1 + \lambda_j h/2}\right)^{k} \boldsymbol{v}_j$$

With $\lambda_j > 0$ trapezoidal method is stable for any $h > 0$

Solving the diffusion equation

Diffusion equation in 2D

$$\frac{\partial u}{\partial t} - \Delta u = f, \quad (x, y) \in \Omega, \quad 0 < t \leq T$$
$$u(x, y, t) = g(x, y, t), \quad (x, y) \in \partial\Omega, \quad 0 < t \leq T$$
$$u(x, y, 0) = u_0(x, y), \quad (x, y) \in \Omega, \quad t = 0$$

Sometimes it is written as $(x, y, t) \in \Omega \times (0, T]$

Time integration: Forward-Euler method

Although $u(x, y, t)$ is a function rather than a vector, all previously explained time integration methods can still be applied. With the time step h the Forward-Euler time iterations are

$$\frac{\partial u(x, y, t)}{\partial t} = \Delta u(x, y, t) + f(x, y, t)$$

$$u(x, y, t_{k+1}) = u(x, y, t_k) + h\left[\Delta u(x, y, t_k) + f(x, y, t_k)\right]$$

$$u^{k+1}(x, y) = u^k(x, y) + h\left[\Delta u^k(x, y) + f^k(x, y)\right]$$

$$k = 0, 1, \ldots; \quad u(x, y, t_0) = u_0(x, y)$$

The time-dependent boundary condition $u(x, y, t_k) = g(x, y, t_k)$, $(x, y) \in \partial\Omega$ must also be enforced at each t_k.

Time integration: Backward-Euler method

The Backward-Euler time iterations are

$$\frac{\partial u(x,y,t)}{\partial t} = \Delta u(x,y,t) + f(x,y,t)$$
$$u(x,y,t_{k+1}) = u(x,y,t_k) + h\left[\Delta u(x,y,t_{k+1}) + f(x,y,t_{k+1})\right]$$
$$u^{k+1}(x,y) - h\Delta u^{k+1}(x,y) = u^k(x,y) + hf^{k+1}(x,y)$$
$$k = 0,1,\ldots; \quad u(x,y,t_0) = u_0(x,y)$$

At the $(k+1)$-st iteration:

- $u^k(x,y)$ and $f^{k+1}(x,y)$ are known functions of $(x,y) \in \Omega$
- $u^{k+1}(x,y)$ is an unknown function

Space discretization: FDM or FVM

- In both FDM and FVM, introduce spatial grid and the solution vector $\boldsymbol{u}^k \in \mathbb{R}^n$, related to the grid values $u(x_i, y_j, t_k)$. Discretize the Laplacian operator $-\Delta \to A \in \mathbb{R}^{n \times n}$, moving Dirichlet boundary conditions to the right-hand side $\boldsymbol{f}^k$ as usual
- For example, Backward-Euler iteration becomes

$$
\begin{aligned}
& u^{k+1}(x,y) - h\Delta u^{k+1}(x,y) = u^k(x,y) + hf^{k+1}(x,y) \\
& \boldsymbol{u}^{k+1} + hA\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + h\boldsymbol{f}^{k+1} \\
& [I + hA]\,\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + h\boldsymbol{f}^{k+1} \\
& k = 0, 1, \ldots; \quad \boldsymbol{u}^0 = \boldsymbol{u}_0
\end{aligned}
$$

- To find the current grid values $\boldsymbol{u}^{k+1}$ in backward Euler we need to solve a linear system. This is computationally equivalent to solving one steady-state problem at each iteration

Solving the diffusion equation with FEM

Diffusion equation in 2D

$$\frac{\partial u}{\partial t} - \Delta u = f, \quad (x, y) \in \Omega, \quad 0 < t \leq T$$
$$u(x, y, t) = g(x, y, t), \quad (x, y) \in \partial\Omega, \quad 0 < t \leq T$$
$$u(x, y, 0) = u_0(x, y), \quad (x, y) \in \Omega, \quad t = 0$$

Sometimes it is written as $(x, y, t) \in \Omega \times (0, T]$

FEM with Forward Euler method

$$u^{k+1}(\boldsymbol{x}) = u^k(\boldsymbol{x}) + h\left[\Delta u^k(\boldsymbol{x}) + f^k(\boldsymbol{x})\right]$$

$$\int_\Omega u^{k+1} v \, d\Omega = \int_\Omega \left[u^k + h\Delta u^k + h f^k\right] v \, d\Omega$$

$$\int_\Omega u^{k+1} v \, d\Omega = \int_\Omega \left(u^k v - h\nabla u^k \cdot \nabla v + h f^k v\right) \, d\Omega$$

$$a(u^{k+1}, v) = L_k(v)$$

- u^k is known, solve for u^{k+1}
- System matrix is very simple in this case (mass matrix)
- Right-hand side changes every k-step, must be re-assembled
- A very simple linear system must be solved at every k-step

FEM with Backward-Euler method

:

$$u^{k+1}(\boldsymbol{x}) - h\Delta u^{k+1}(\boldsymbol{x}) = u^k(\boldsymbol{x}) + hf^{k+1}(\boldsymbol{x})$$

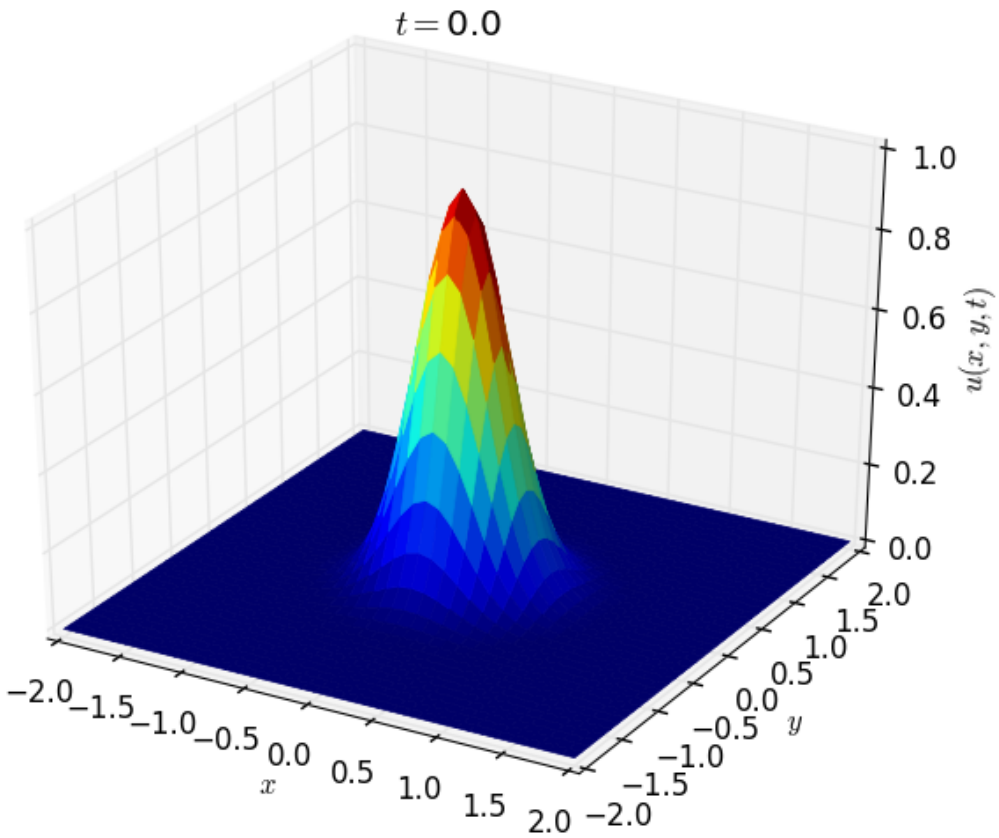
$$\int_\Omega \left[u^{k+1} - h\Delta u^{k+1}\right] v \, d\Omega = \int_\Omega \left[u^k + hf^{k+1}\right] v \, d\Omega$$

$$\int_\Omega \left(u^{k+1}v + h\nabla u^{k+1} \cdot \nabla v\right) \, d\Omega = \int_\Omega \left(u^k v + hf^{k+1}v\right) \, d\Omega$$

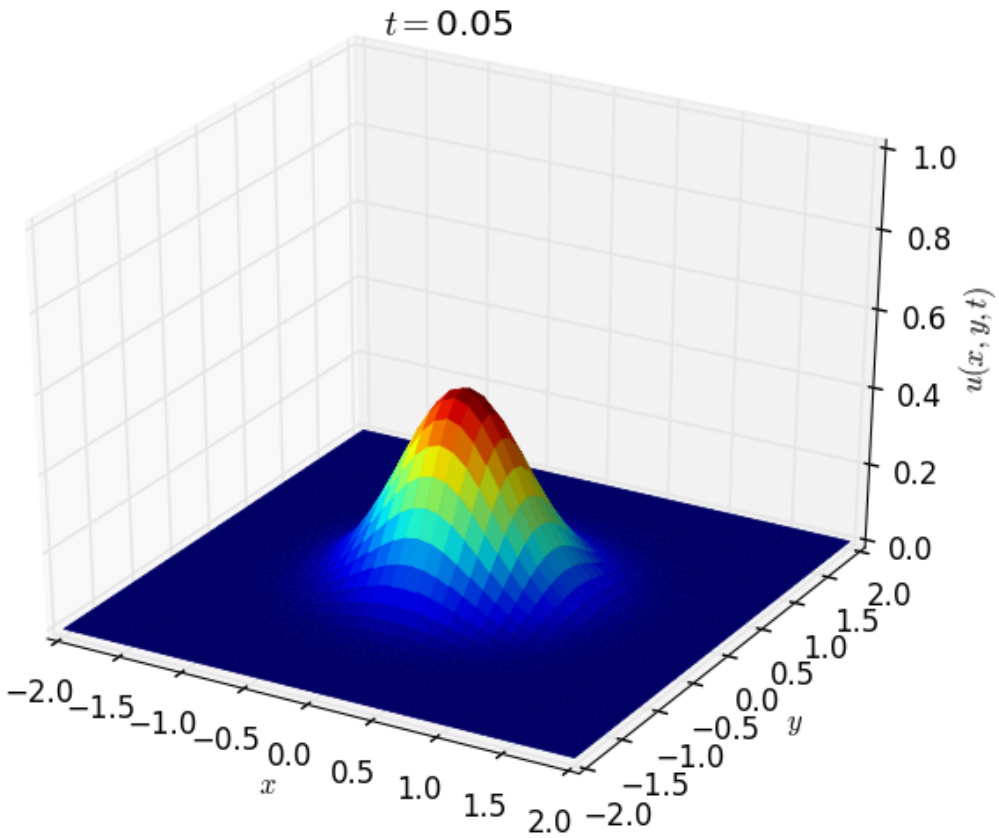
$$a(u^{k+1}, v) = L_k(v)$$

- u^k is known, solve for u^{k+1}
- System matrix is more complicated in this case, but its assembly may be performed only once
- Right-hand side changes in every k-step, must be re-assembled
- A linear system must be solved at every k-step

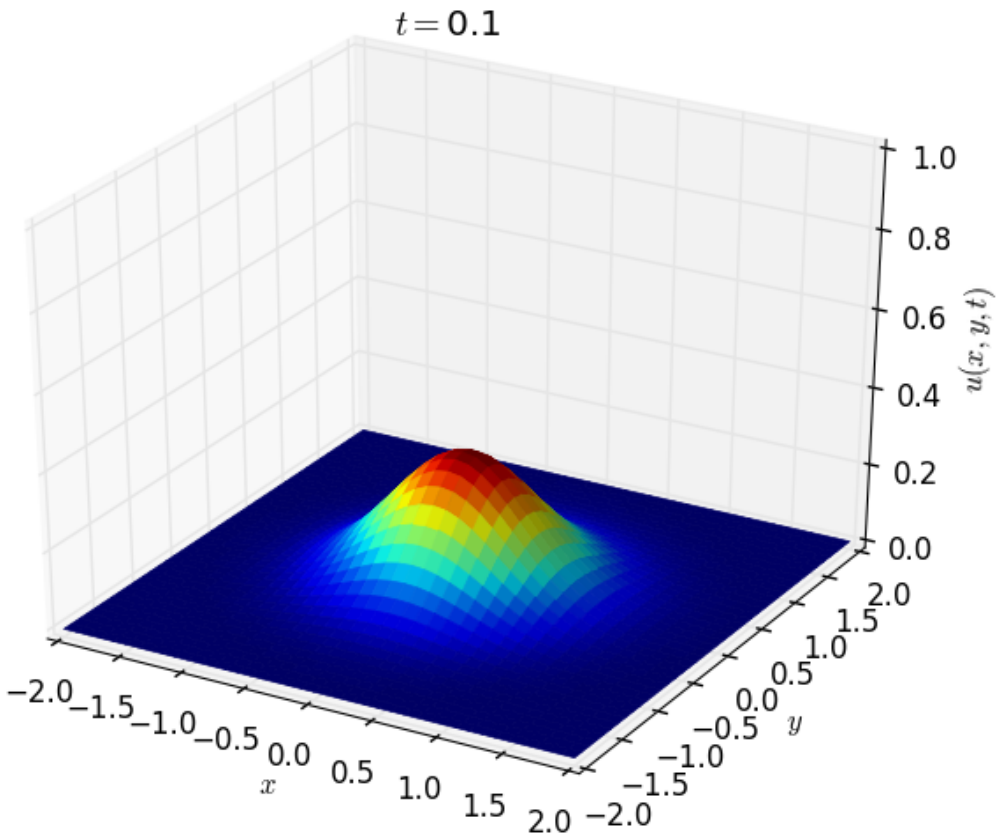
Diffusion: $t = 0.0$ (initial condition)



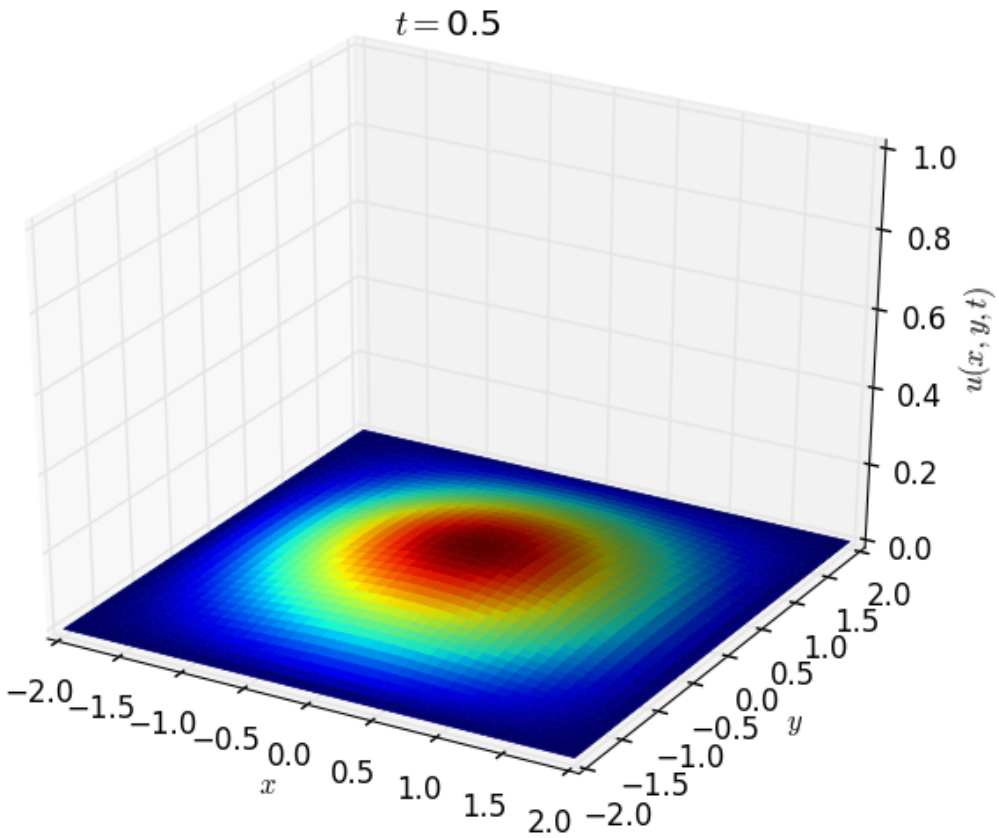
Diffusion: $t = 0.05$



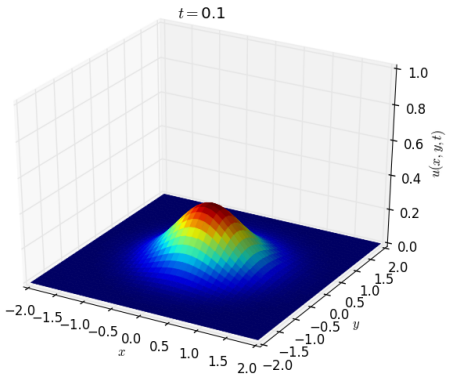
Diffusion: $t = 0.1$

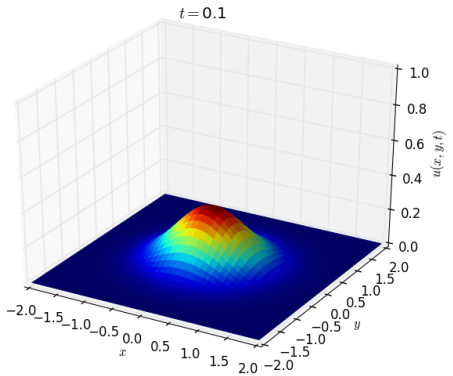


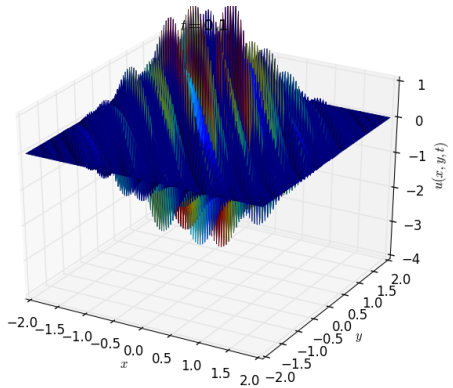
Diffusion: $t = 0.5$



Backward vs Forward Euler: $t = 0.1$







Left: backward Euler with $\Delta t = 0.01$ (10 steps)
Middle and right: forward Euler with $\Delta t = 0.0007$ (150 steps) and $\Delta t = 0.00095$ (105 steps)