Assignment 5

Angewandte Modellierung 25

Carl Colmant

May 25, 2025

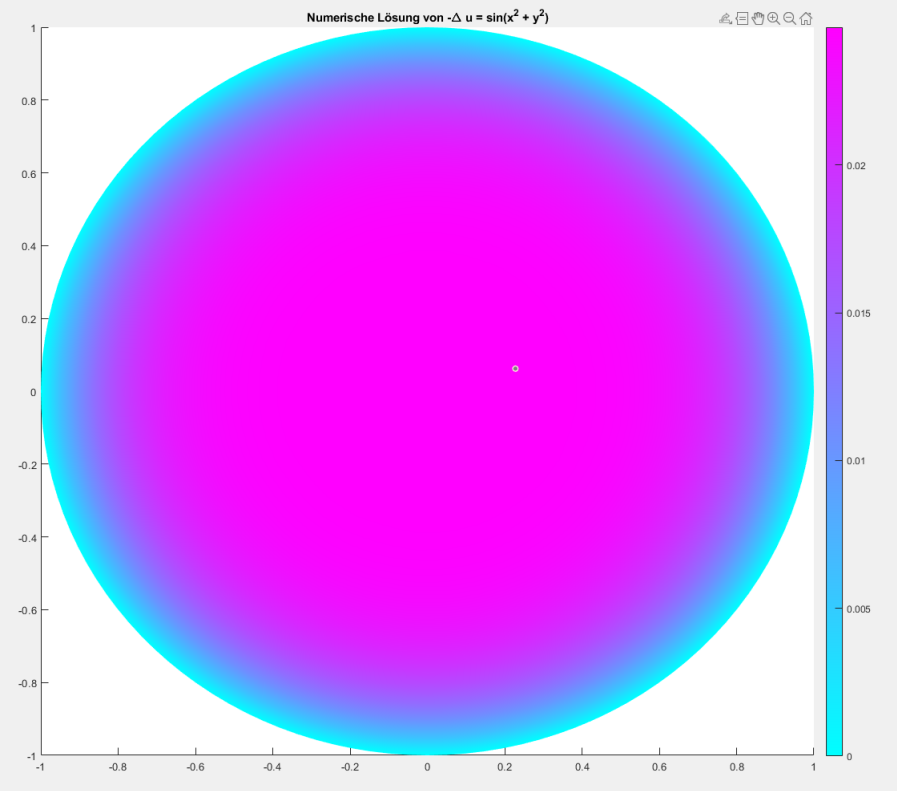
Exercise 1. Poisson equation

Mit der Poisson Gleichung will man eine Funktion f approximieren. In unserem Fall lösen wir die Gleichung für den Einheitskreis.

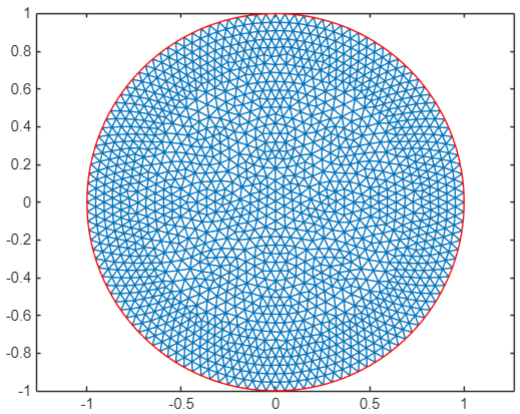
a)

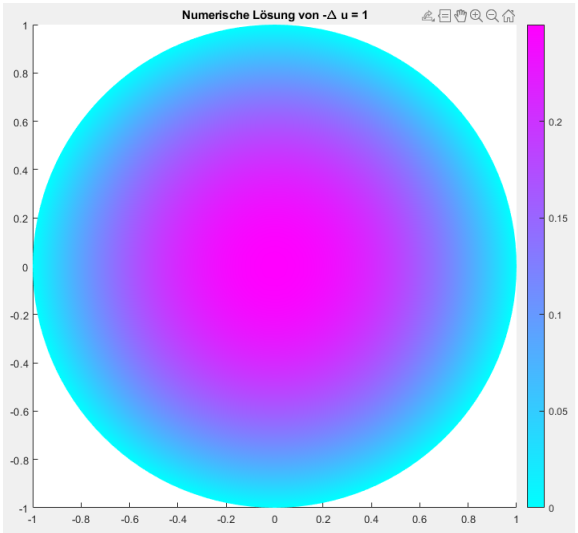
Ich habe die funktion $f(x, y) = sin^2(x^2 + y^2)$ gewählt.

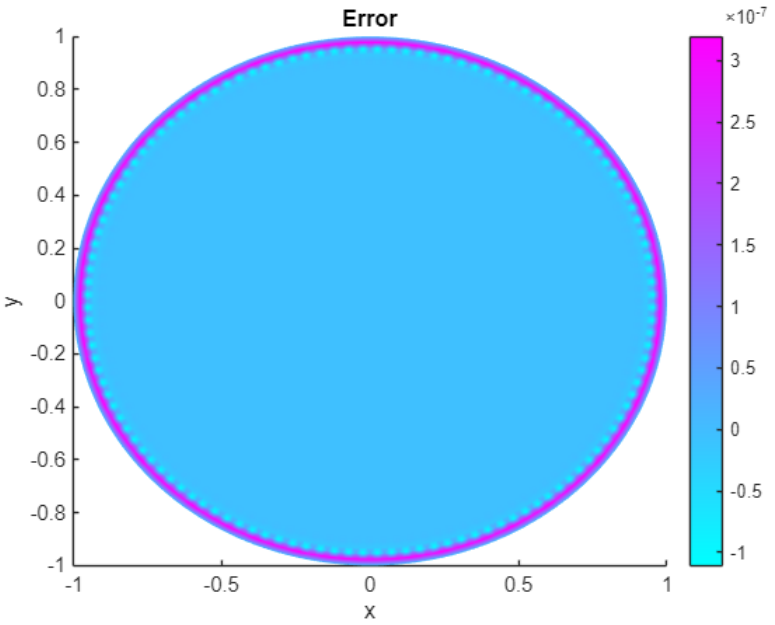
```
% Modell erstellen
model = createpde();
geometryFromEdges(model, @circleg);

% Sinus-Quellterm definieren
f = @(location, state) sin(location.x.^2 + location.y.^2).^2;


% Koeffizienten setzen
specifyCoefficients(model, "m", 0, "d", 0, "c", 1, "a", 0, "f",f);

% Randbedingung (u = 0 auf dem Rand)
applyBoundaryCondition(model, "dirichlet", "Edge", 1:model.Geometry.NumEdges, "u", 0);

% Mesh generieren
generateMesh(model, "Hmax", 0.1);

% Lösen
results = solvepde(model);

% Lösung visualisieren
pdeplot(model, "XYData", results.NodalSolution);
title("Numerische Lösung von -\Delta u = sin(x^2 + y^2)");
```



Im Gegensatz dazu, die Funktion $f(x, y) = 1$ also der Einheitskreis:

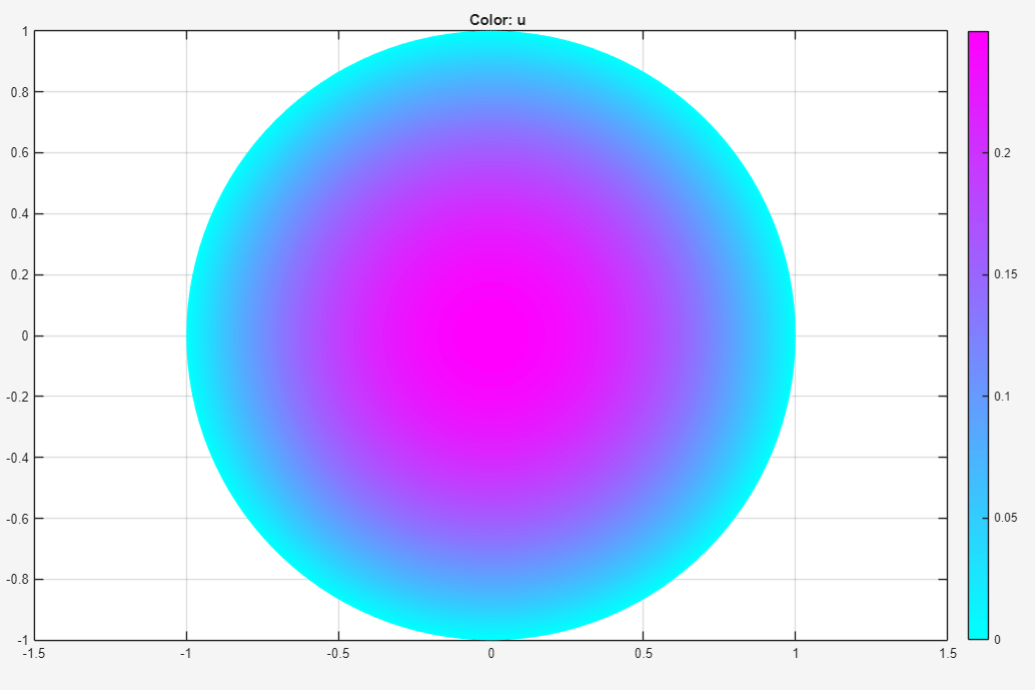


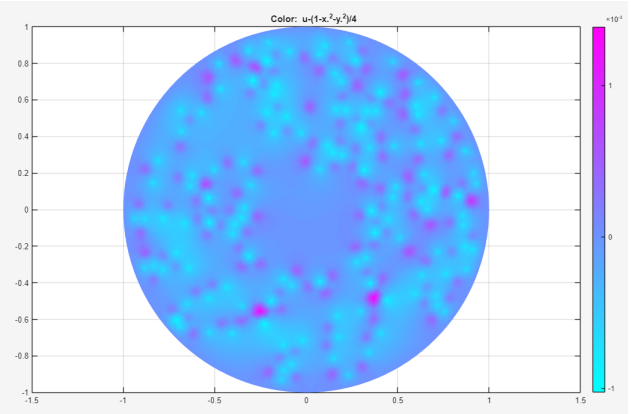


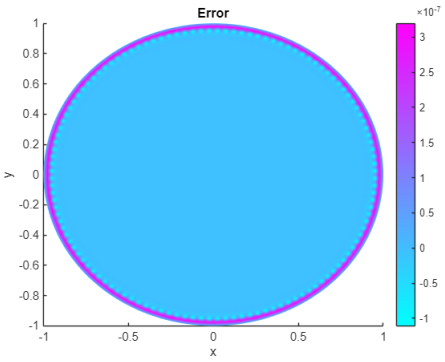


b)

Im gegensatz zu a) benutzt der pde Modeler eine andere Funktion zur berechnung des Meshes und somit entsteht eine etwas andere Lösung. Dies kann nur im Error gesehen werden also im Vergleich der Perfekten Lösung.







Wenn man beide Error plots gegenüber stellt sieht man das bei der Implementation aus dem Livescript die Fehler aussen am Kreis sich verteilen wehrend die Fehler bei der Implementation aus dem PDE Modeler sich eher im Kreis verteilen.

Exercise 2. Helmholtz's equation

Die Helmholtz Gleichung ist:

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times E \right) - k_0^2 \left(\epsilon_r - j \frac{\sigma}{\omega \epsilon_0} \right) E = 0$$

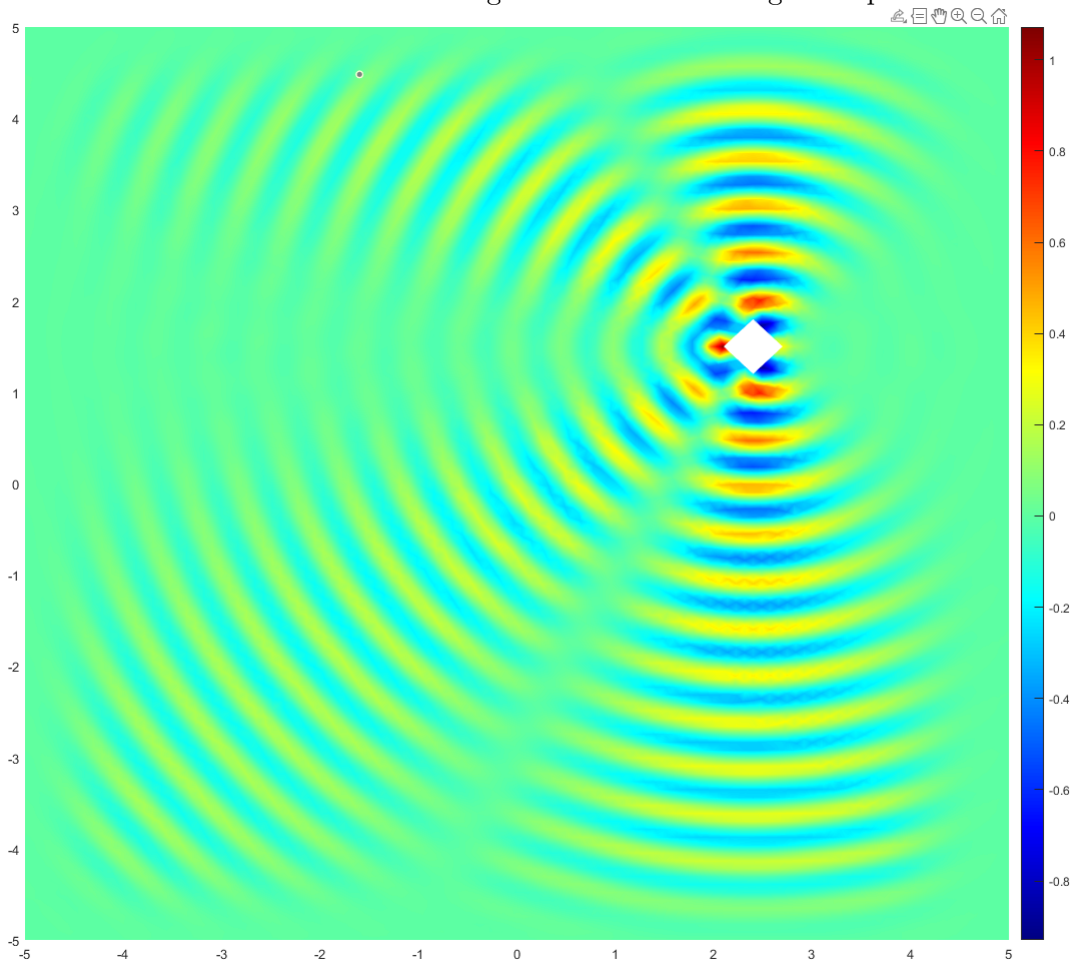
wobei

- E: elektrische Feld
- ∇ ist die Ableitung nach dem ort
- μ_r: relative Permeabilität in dem Script ist $\mu_r = 1$
- ϵ_r: relative Permittivität in dem Script ist $\epsilon_r = 1$
- σ: elektrische Leitfähigkeit in dem Script ist $\sigma = 0$
- ω: Frequenz in dem Script ist $\omega = 4\pi$
- ϵ_0: elektrische Feldkonstante in dem Script ist $\epsilon_0 = 1$
- μ_0: magnetische Feldkonstante in dem Script ist $\mu_0 = 1$
- k: Wellenzahl in dem Script ist $k = \frac{\omega}{c} = \frac{\omega}{\sqrt{\mu_0 \epsilon_0}}$

Das ist eine imaginäre Funktion.

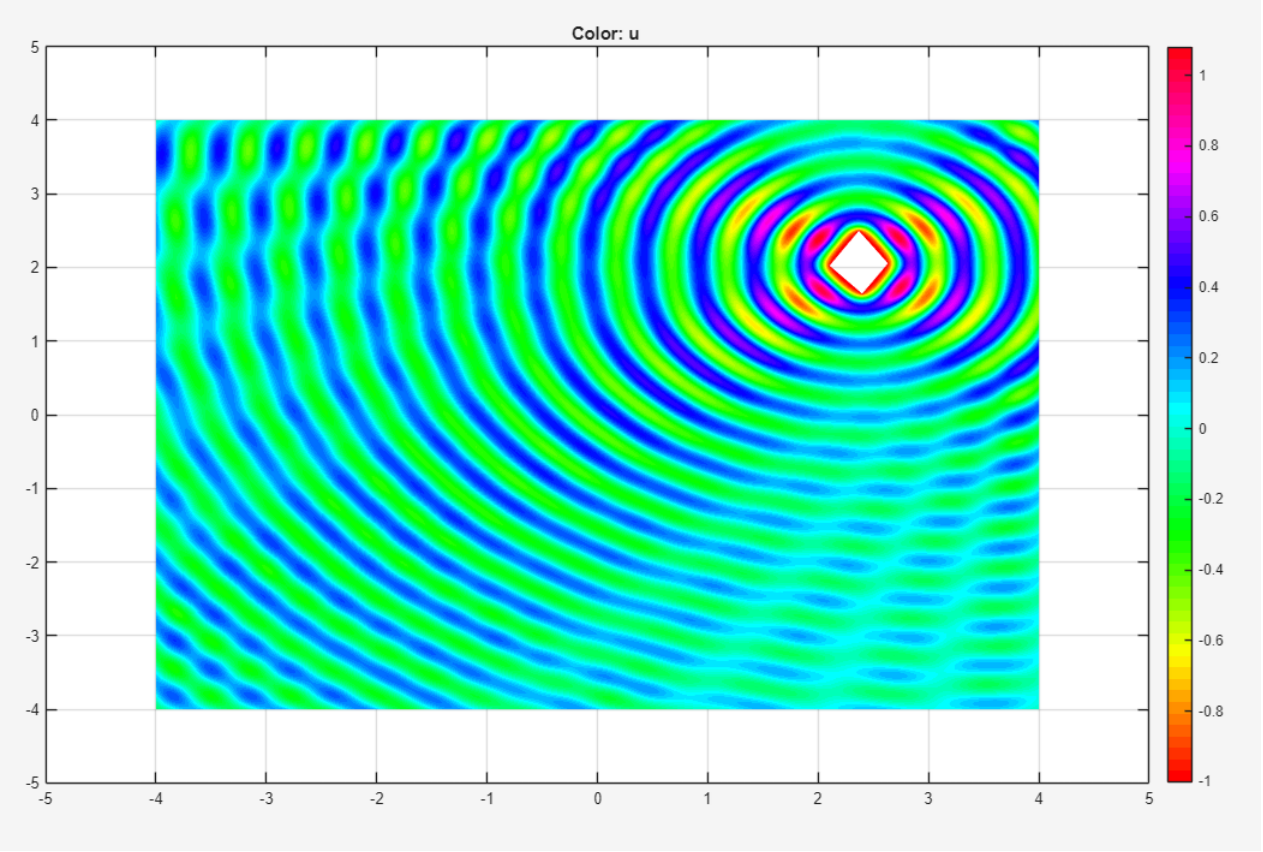
a)

wenn wir die schritte des Demo files folgen bekommen wir folgenden plot:



0.0.1 b)

Um das Ergebnis mit dem PDE Modeler zu erhalten müssen wir einmal das Rechteck als äussere Einschränkung definieren dazu setzten wir im PDE Modeler die boundary conditions im Neumann modus auf $g = 0$ und $q = -i * 4 * pi$ und für die eingezeichnete Raute im Dietrich modus auf $h = 1$ und $r = -e^{-i*4*pi*x}$. Nun muss noch die differential gleichung die approximiert werden soll in das system gegeben werden. Dazu müssen wir die Helmholtz Gleichung in eliptischer Form unter PDE specification definieren. Dazu setzen wir $c = 1, f = 0$ und $a = -(4 * pi)^2$. Aus diesen Einstellungen erhalten wir folgenden Polt:



Ich habe nur ein Quadrat von -4 bis 4 geplottet, sonst ist das Ergebnis aber sehr ähnlich.

Exercise 3. Heat equation

disclaimer: heavy AI usage.

1. Inhomogeneous Heat Equation on Square Domain ('pdedemo5')

Gleichung:

$$\frac{\partial u}{\partial t} - \Delta u = 1 \quad \text{auf dem Quadrat } \{-1 \le x, y \le +1\}$$

mit einer kreisförmigen Aussparung $x^2 + y^2 \le 0.4^2$.

Anfangsbedingung:

$$u(x, y, 0) = \begin{cases} 1, & \text{falls } x^2 + y^2 \le 0.4^2, \\ 0, & \text{sonst.} \end{cases}$$

Randbedingungen:

$$u(x, y, t) \;=\; 0 \quad \text{für alle Punkte an den vier Seiten des Quadrats.}$$

2. 1D Heat Equation mit `pdepe` (Beispiel 1)

Gleichung:

$$\pi^2 \, \frac{\partial u}{\partial t} \;=\; \frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le 1.$$

Anfangsbedingung:

$$u(x, 0) = \sin(\pi x).$$

Randbedingungen:

$$\begin{aligned} u(0, t) &= 0, && \text{(homogene Dirichlet)} \\ \frac{\partial u}{\partial x}(1, t) &= -\pi \, e^{-t}, && \text{(inhomogene Neumann).} \end{aligned}$$

diffusion equation

Lösung der Diffusionsgleichung mit implizitem Euler

Wir betrachten

$$\frac{\partial u}{\partial t} = D \, \frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 1], \; t \in [0, T],$$

mit

$$D = 0.01, \quad u(x, 0) = \exp\!\big(-100(x - 0.5)^2\big), \quad u(0, t) = u(1, t) = 0.$$

Diskretisierung

- Raumgitter: $N_x = 51$ Punkte, $\quad \Delta x = \frac{1}{N_x - 1} = 0.02.$
- Zeitschritte: $N_t = 200, \quad \Delta t = T / N_t = 0.2/200 = 0.001.$
- Implizites Euler:

  $$\mathbf{A} \, \mathbf{u}^{\,n+1} = \mathbf{u}^{\,n},$$

  wobei nur die inneren Gitterpunkte $i = 2, \ldots, N_x - 1$ betrachtet werden.

Matrixaufbau

Setze $N = N_x - 2$ und

$$\alpha = \frac{D\,\Delta t}{(\Delta x)^2}.$$

Dann ist $\mathbf{A} \in \mathbb{R}^{N\times N}$ tridiagonal mit

$$\mathbf{A} = \text{spdiags}\Big([\underbrace{0,\,-\alpha,\ldots,-\alpha}_{N}\,,\;\underbrace{1+2\alpha,\ldots,1+2\alpha}_{N}\,,\;\underbrace{-\alpha,\ldots,-\alpha,0}_{N}],\;-1:1,\;N,\;N\Big).$$

Explizit schreibt man in MATLAB:

```
N     = Nx-2;
alpha = D*dt/dx^2;
main  = (1 + 2*alpha) * ones(N,1);
off   = -alpha        * ones(N-1,1);
col_im1 = [0;         off];
col_ip1 = [off;       0 ];
A = spdiags([col_im1, main, col_ip1], -1:1, N, N);
```

```
% diffusion_implicit.m
% Löse du/dt = D * d2u/dx2 auf [0,1] mit u(0)=u(1)=0,
% Anfangsbedingung Gauß bei x=0.5, D=0.01, implizites Euler + Finite Volume

clear; close all; clc;

%% Parameter
D     = 0.01;      % Diffusionskoeffizient
L     = 1;         % Länge des Stabes
Tmax  = 0.2;       % Endzeit
Nx    = 51;        % Anzahl Gitterpunkte in x
Nt    = 200;       % Anzahl Zeitschritte

dx = L/(Nx-1);
dt = Tmax/Nt;
x  = linspace(0,L,Nx)';
t  = linspace(0,Tmax,Nt+1);

%% Anfangsbedingung
u0 = exp(-100*(x-0.5).^2);

% Matrix A für implizites Euler (innere Punkte 2..Nx-1)
N     = Nx-2;
alpha = D*dt/dx^2;
main  = (1 + 2*alpha) * ones(N,1);
off   = -alpha       * ones(N-1,1);
col_im1 = [0;        off];
col_ip1 = [off;      0 ];
A     = spdiags([col_im1, main, col_ip1], -1:1, N, N);


%% Zeitintegration
U = zeros(Nx, Nt+1);
U(:,1) = u0;

for n = 1:Nt
    b = U(2:end-1,n);         % RHS = alte Lösung innen
    U(2:end-1,n+1) = A\b;     % löse A * u^{n+1} = b
    % U(1,n+1)=0; U(end,n+1)=0; % BCs sind schon null initialisiert
end

%% 1) Oberflächenplot
[X,T] = meshgrid(x,t);  % Achtung: T(i,j) = t(i), X(i,j)=x(j)
figure;
surf(X, T, U', 'EdgeColor','none');
xlabel('x'); ylabel('t'); zlabel('u(x,t)');
title('Surface plot of u(x,t)');
view(45,30);
colorbar;
shading interp;

%% 2) Animation
figure;
h = plot(x, U(:,1), 'LineWidth',2);
axis([0 1 0 1]);
xlabel('x'); ylabel('u');
title(sprintf('t = %.3f',0));
drawnow;

for n = 1:Nt+1
    set(h, 'YData', U(:,n));
    title(sprintf('Diffusion t = %.3f', t(n)));
    pause(0.05);
end
```

Exercise 4. Wave equation

1. Command-Line Beispiel: `openExample('pde/pdedemo6')`

Wir lösen

$$m\,\frac{\partial^2 u}{\partial t^2} - \nabla\cdot(c\nabla u) + a\,u = f \quad \text{auf } (x,y)\in[-1,1]^2$$

mit

$$m = 1, \quad c = 1, \quad a = 0, \quad f = 0.$$

Randbedingungen

- **Dirichlet:**
$$u = 0 \quad \text{auf den linken und rechten Rändern.}$$
- **Neumann:**
$$\frac{\partial u}{\partial n} = 0 \quad \text{auf den oberen und unteren Rändern.}$$

Anfangsbedingungen

$$u(x,y,0) \;=\; \arctan\big(\cos(\tfrac{\pi}{2}x)\big), \qquad u_t(x,y,0) \;=\; 3\,\sin(\pi x)\,\exp\big(\sin(\tfrac{\pi}{2}y)\big).$$

2. PDE-App Beispiel: „Wave Equation on a Square Domain“

Das gleiche PDE und Gebiet, eingerichtet in der PDE Modeler App.

Randbedingungen

- **Dirichlet:**
$$u = 0 \quad \text{links und rechts.}$$
- **Neumann:**
$$\frac{\partial u}{\partial n} = 0 \quad \text{oben und unten.}$$

Anfangsbedingungen

$$u(x,y,0) = \arctan\big(\cos(\tfrac{\pi}{2}x)\big), \qquad u_t(x,y,0) = 3\,\sin(\pi x)\,\exp\big(\sin(\tfrac{\pi}{2}y)\big).$$

0.1 wave equation:

Ich habe es nicht hinbekommen die PDE Modeler App zu benutzen. Hier ist eine Lösung mit einem Matlab script stadtessen. $u(0,t) = u(1,t) = 0$,

```
% wave_fd.m
% Löse d u_tt = c * u_xx + f - a*u mit d=1, c=1, a=0, f=0
% auf [0,1], u(0,t)=u(1,t)=0,
% Anfang: u(x,0)=sin(2*pi*x), ut(x,0)=0

clear; close all; clc;

%% Parameter
c    = 1;      % Wellengeschwindigkeit^2
L    = 1;      % Länge des Intervalls
Tend = 1;      % Endzeit
Nx   = 101;    % Gitterpunkte in x
dx   = L/(Nx-1);
CFL  = 0.9;    % CFL-Zahl <1 für Stabilität
dt   = CFL*dx/sqrt(c);
Nt   = floor(Tend/dt);
dt   = Tend/Nt;   % präzisieren, damit Nt*dt=Tend

x = linspace(0,L,Nx)';
t = (0:Nt)*dt;

%% Initialbedingungen
U = zeros(Nt+1, Nx);
U(1,:) = sin(2*pi*x)';      % u(x,0)
% ut(x,0)=0 => erster Zeitschritt mit Taylor-Ansatz
r2 = c*dt^2/dx^2;
U(2, 2:end-1) = U(1,2:end-1) + ...
    0.5*r2*(U(1,3:end) - 2*U(1,2:end-1) + U(1,1:end-2));
% Randwerte bleiben null:
U(2,1) = 0;
U(2,end) = 0;
```

```
%% Zeitintegration (explizit)
for n = 2:Nt
    U(n+1,2:end-1) = 2*U(n,2:end-1) - U(n-1,2:end-1) + ...
        r2*(U(n,3:end) - 2*U(n,2:end-1) + U(n,1:end-2));
    % Dirichlet-BC
    U(n+1,1) = 0;
    U(n+1,end) = 0;
end

%% 1) Oberflächenplot
[X,T] = meshgrid(x, t);
figure;
surf(X, T, U, 'EdgeColor', 'none');
xlabel('x'); ylabel('t'); zlabel('u(x,t)');
title('Wellengleichung: u_{tt} = c u_{xx}');
view(45,30);
colorbar;
shading interp;

%% 2) Animation
figure;
h = plot(x, U(1,:), 'LineWidth', 2);
axis([0 1 -1 1]);
xlabel('x'); ylabel('u');
title(sprintf('t = %.3f',0));
drawnow;

for n = 1:Nt+1
    set(h, 'YData', U(n,:));
    title(sprintf('Wellenausbreitung bei t = %.3f', t(n)));
    pause(0.02);
end
```

Github

Wie immer sind alle meine benutzten Dateien auf meinem Github zu finden.