Simulating time-dependent problems

Our goal is to solve the heat equation

$$\partial_t u - \mu\Delta u = f \text{ in } \Omega = (-1,1)^2 \text{ for } t \in I = [0,2]$$

where $\mu = 0.1$ is a parameter controling the diffusion and with initial condition $u(0) = 0$ and Dirichlet condition $u = 0$ on $\partial\Omega$. The right hand side depends on the time

$$f(x,y,t) = \exp\Big(-10(x-0.5\cos(\pi t))^2 - 10(y-0.5\sin(\pi t))^2\Big)$$

We start with the implicit Euler method. The discrete variational formulation is

$$(u^n,\phi) + \Delta t\cdot 0.1(\nabla u^n,\nabla\phi) = (u^{n-1},\phi) + \Delta t(f,\phi)$$

Starting point is the script for stationary problems

We only add a parameter time passed to the right hand side. We solve the problem for discrete points in time. But, for each point in time, we solve the stationary Poisson problem

$$-0.1\Delta u = f(t)$$

In [1]:

```
%reset
from ngsolve import *
from ngsolve.webgui import Draw
from netgen.geom2d import SplineGeometry

import matplotlib.pyplot as plt
import numpy as np


def F(x,y,t):   # definition of a time-dependent right hand side
    mx = 0.5*cos(pi*t)
    my = 0.5*sin(pi*t)
    return exp(-10*(x-mx)*(x-mx)-10*(y-my)*(y-my))

def Matrix(fes, dt, mu):
    u,v = fes.TnT()
    A = BilinearForm((u*v+dt*0.5*mu*grad(u)*grad(v))*dx).Assemble()
    return A

def MassMatrix(fes, dt, mu):
    u,v = fes.TnT()
```

```
    M = BilinearForm(u*v*dx-0.5*dt*mu*grad(u)*grad(v)*dx).Assemble()
    return M


def solve(fes, guold, A, M, F,time): # solves -Delta u = F, u=G on boundary
    u,v = fes.TnT()         # get test and trial function


    f = LinearForm(dt * F(x,y,time) * v * dx).Assemble()  # rhs

    r = f.vec + M.mat * guold.vec                    # add old solution

    gu = GridFunction(fes)  # to store solution
    gu.vec.data += A.mat.Inverse(freedofs=fes.FreeDofs()) * r
    return gu

def domain(hmax):
    geo = SplineGeometry()
    pnts =[(-1,-1), (1,-1), (1,1), (-1,1)]
    p1,p2,p3,p4 = [geo.AppendPoint(*pnt) for pnt in pnts]
    lines = [[["line",p1,p2],"bd"],[["line",p2,p3],"bd"],
              [["line",p3,p4],"bd"],[["line",p4,p1],"bd"]]
    [geo.Append(c,bc=bc) for c,bc in lines]
    return Mesh(geo.GenerateMesh(maxh=hmax))

### Definition of the mesh
hmax = 0.1
mesh = domain(hmax)
fes = H1(mesh, order=1 , dirichlet="bd")

### Definition of the time-mesh
Tmax = 2
Ntime = 20                         # Number of steps
tt = np.linspace(0,Tmax,Ntime+1)   # discrete points in time
dt = Tmax/Ntime                    # time step size

### Problem parameters
mu = 0.01

### Definition of Matrix
A = Matrix    (fes, dt, mu)
M = MassMatrix(fes, dt, mu)

### Solution and vector to store all solutions for visualization
gu=GridFunction(fes)
gu_all = GridFunction(gu.space,multidim=0)
gu_all.AddMultiDimComponent(gu.vec)

for i in range(Ntime):                       # loop over time
    gu = solve(fes, gu, A, M, F, tt[i+1])    # solve at new time tt[i+1]
    gu_all.AddMultiDimComponent(gu.vec)

Draw(gu_all, mesh, interpolate_multidim=False, animate=True,
     order = 1, deformation = True, min=0, max=1, autoscale = True)
```

```
/home/dayron/.local/lib/python3.10/site-packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g. as a system package and as a pip package). As a result, the 3D projection is not available.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions of "
WebGuiWidget(layout=Layout(height='50vh', width='100%'), value={'gui_settings': {}, 'ngsolve_version': '6.2.24…
```

Out[1]:

```
BaseWebGuiScene
```

--> Task

Goal is to change the program into an implicit Euler solution. Two steps are required:

1. The BilinearForm must be modified Change it to

$$(u, \phi) + \Delta t(\nabla u, \nabla \phi)$$

   The time step dt is a variable. Pass it as extra argument to solve

2. The right hand side must be modified to include the old solution

In [ ]:

```
```