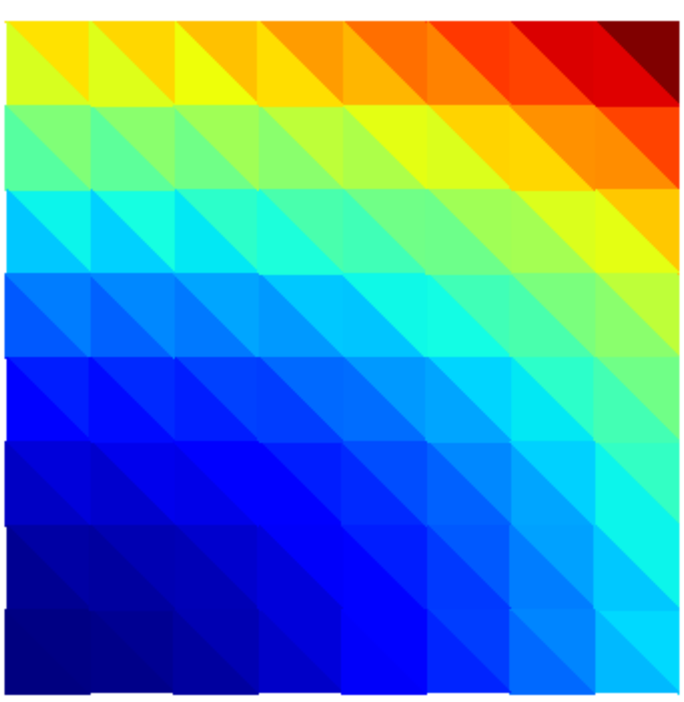
FEnics exercise #1 below, executed by scikit-fem.

In [1]:

```
import numpy as np

from skfem import *
from skfem.models.poisson import laplace, unit_load, mass

mesh = MeshTri().refined(3)
mesh

V = InteriorBasis(mesh, ElementTriP1())

u_D = 1 + [1, 2] @ mesh.p ** 2

boundary = mesh.boundary_nodes()

u = np.zeros_like(u_D)
u[boundary] = u_D[boundary]

a = asm(laplace, V)
L = -6.0 * asm(unit_load, V)

u = solve(*condense(a, L, u, D=boundary))

ax = mesh.plot(u)
ax.get_figure().savefig("poisson.svg")

mesh.save("fenics01.ply")

error = u - u_D
print("error_L2  =", np.sqrt(error.T @ asm(mass, V) @ error))
print("error_max =", np.linalg.norm(error, np.inf))

```

```
Warning: PLY doesn't support 64-bit integers. Casting down to 32-bit.

error_L2  = 3.090730095650652e-16
error_max = 1.1102230246251565e-15
```



Type *Markdown* and LaTeX: α^2

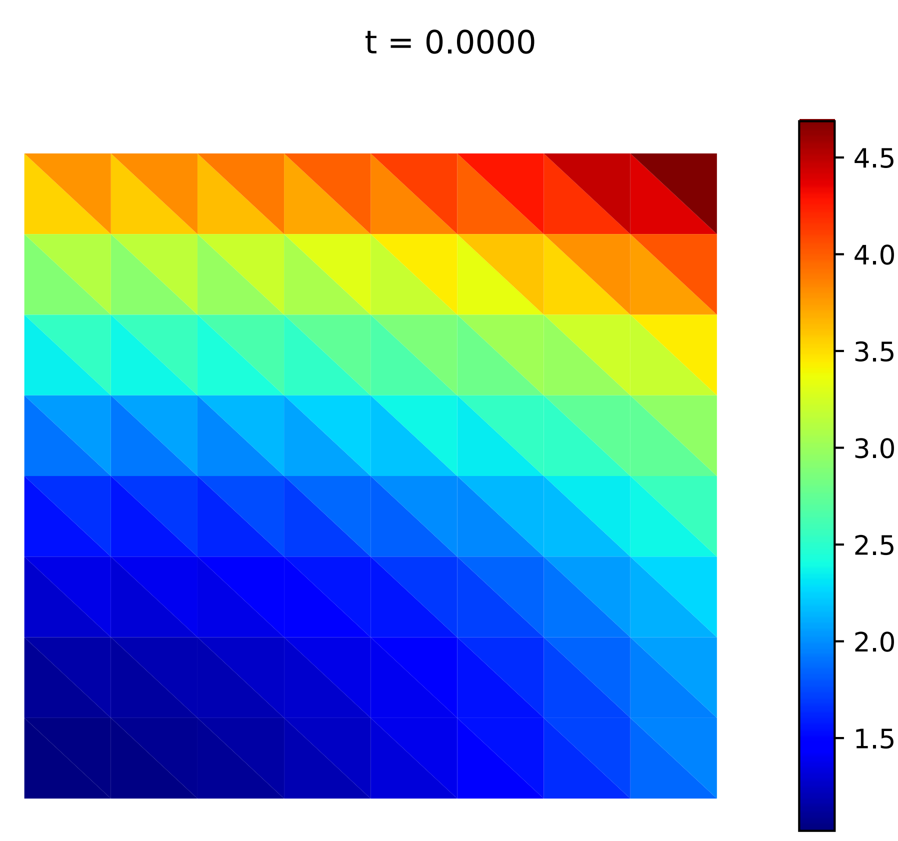
FEnics exercise #3 below, executed by scikit-fem.

In [2]:

```
from matplotlib.pyplot import subplots, pause
# %matplotlib qt
import numpy as np
%config InlineBackend.figure_formats = ['svg']

from skfem import *
from skfem.models.poisson import laplace, mass, unit_load

alpha = 3.0
beta = 1.2

nx = ny = 2 ** 3

time_end = 2.0
```

```python
num_steps = 10
dt = time_end / num_steps

mesh = (
    MeshLine(np.linspace(0, 1, nx + 1)) * MeshLine(np.linspace(0, 1, ny + 1))
).to_meshtri()
basis = InteriorBasis(mesh, ElementTriP1())

boundary = basis.get_dofs().all()
interior = basis.complement_dofs(boundary)

M = asm(mass, basis)
A = M + dt * asm(laplace, basis)
f = (beta - 2 - 2 * alpha) * asm(unit_load, basis)

fig, ax = subplots()


def dirichlet(t: float) -> np.ndarray:
    return 1.0 + [1.0, alpha] @ mesh.p ** 2 + beta * t


t = 0.0
u = dirichlet(t)

zlim = (0, np.ceil(1 + alpha + beta * time_end))

for i in range(num_steps + 1):

    ax.cla()
    ax.axis("off")
    fig.suptitle("t = {:.4f}".format(t))
    mesh.plot(u, ax=ax, zlim=zlim)
    if t == 0.0:
        fig.colorbar(ax.get_children()[0])
    fig.show()
    pause(1.0)

    t += dt
    b = dt * f + M @ u

    u_D = dirichlet(t)
    u = solve(*condense(A, b, u_D, D=boundary))
    error = np.linalg.norm(u - u_D)
    print("t = %.2f: error = %.3g" % (t, error))

```

```
C:\Users\gary\AppData\Local\Temp/ipykernel_7460/3832573378.py:50: UserWarning: Matplotlib is curren
tly using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the
figure.
  fig.show()
```



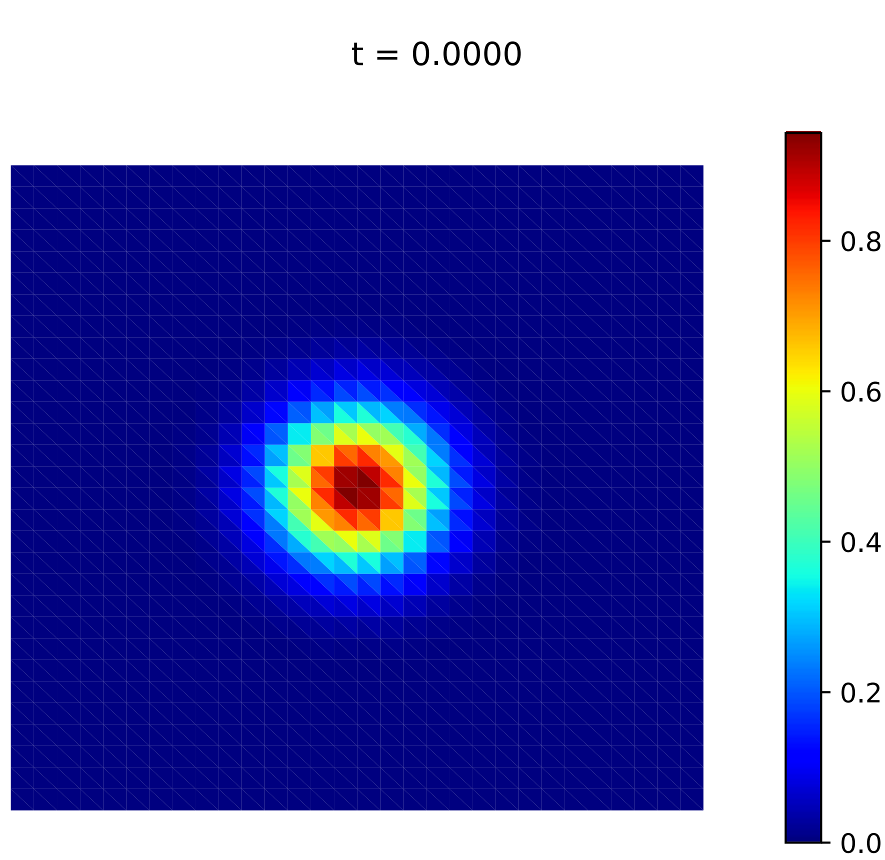

```
t = 0.20: error = 9.44e-15
t = 0.40: error = 1.25e-14
t = 0.60: error = 1.45e-14
t = 0.80: error = 1.7e-14
t = 1.00: error = 1.68e-14
t = 1.20: error = 1.89e-14
t = 1.40: error = 2.01e-14
t = 1.60: error = 2.01e-14
t = 1.80: error = 2.16e-14
t = 2.00: error = 2.34e-14
t = 2.20: error = 2.46e-14
```

FEnics exercise #4 below, executed by scikit-fem.

In [3]:

```python
from pathlib import Path

from matplotlib.pyplot import subplots, pause
import numpy as np

from skfem import *
from skfem.models.poisson import laplace, mass

a = 5.0

nx = ny = 30

time_end = 2.0
num_steps = 50
dt = time_end / num_steps

mesh = (
    MeshLine(np.linspace(-2, 2, nx + 1)) * MeshLine(np.linspace(-2, 2, ny + 1))
).to_meshtri()
basis = InteriorBasis(mesh, ElementTriP1())

boundary = basis.get_dofs().all()
interior = basis.complement_dofs(boundary)

M = asm(mass, basis)
A = M + dt * asm(laplace, basis)

fig, ax = subplots()

t = 0.0
u = np.exp(-a * (np.sum(mesh.p ** 2, axis=0)))  # initial condition, P1 only

output_dir = Path("heat_gaussian")
try:
    output_dir.mkdir()
except FileExistsError:
    pass

for i in range(num_steps + 1):

    ax.cla()
    ax.axis("off")
    fig.suptitle("t = {:.4f}".format(t))
    mesh.plot(u, ax=ax, zlim=(0, 1))
    if t == 0.0:
        fig.colorbar(ax.get_children()[0])
        fig.savefig("initial.png")
    fig.show()
    pause(0.01)

    t += dt
    b = M @ u

    u = solve(*condense(A, b, D=boundary))

    mesh.save(str(output_dir.joinpath(f"solution{i:06d}.msh")), {"temperature": u})

```

```
C:\Users\gary\AppData\Local\Temp/ipykernel_7460/3973729471.py:48: UserWarning: Matplotlib is curren
tly using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the
figure.
  fig.show()
```



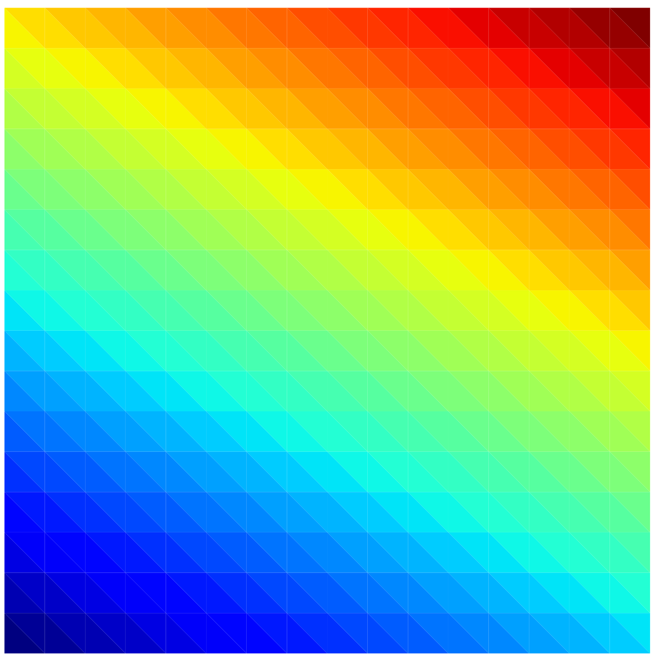
FEnics exercise #5 below, executed by scikit-fem.

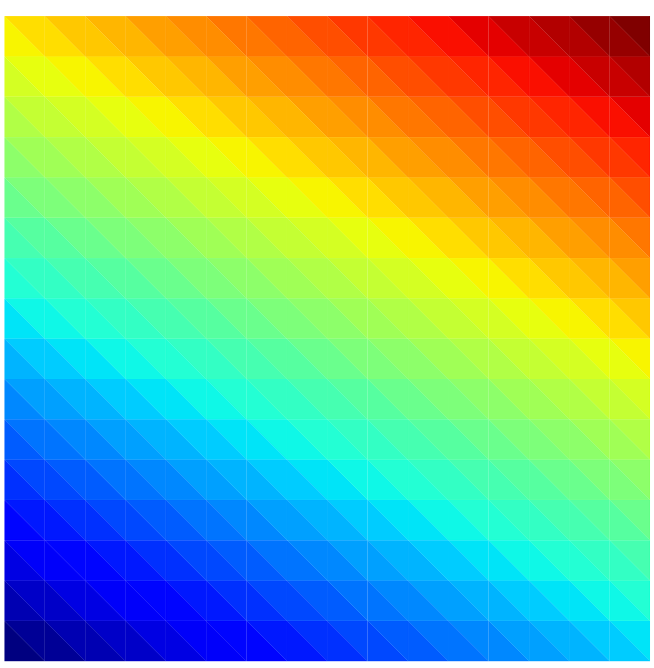
In [18]:

```python
from pathlib import Path

import numpy as np
from scipy.optimize import root

from sympy import symbols
from sympy.vector import CoordSys3D, gradient, divergence
from sympy.utilities.lambdify import lambdify

from skfem import (
    MeshTri,
    InteriorBasis,
    ElementTriP1,
    BilinearForm,
    LinearForm,
    asm,
    solve,
    condense,
)
from skfem.models.poisson import laplace
from skfem.visuals.matplotlib import plot
%config InlineBackend.figure_formats = ['svg']


output_dir = Path("poisson_nonlinear")

try:
    output_dir.mkdir()
except FileExistsError:
    pass


def q(u):
    """Return nonlinear coefficient"""
    return 1 + u * u


R = CoordSys3D("R")


def apply(f, coords):
    x, y = symbols("x y")
    return lambdify((x, y), f.subs({R.x: x, R.y: y}))(*coords)


u_exact = 1 + R.x + 2 * R.y  # exact solution
f = -divergence(q(u_exact) * gradient(u_exact))  # manufactured RHS

mesh = MeshTri().refined(3)  # refine thrice

V = InteriorBasis(mesh, ElementTriP1())

boundary = V.get_dofs().all()
interior = V.complement_dofs(boundary)


@LinearForm
def load(v, w):
    return v * apply(f, w.x)


b = asm(load, V)


@BilinearForm
def diffusion_form(u, v, w):
    return sum(v.grad * (q(w["w"]) * u.grad))


def diffusion_matrix(u):
    return asm(diffusion_form, V, w=V.interpolate(u))


dirichlet = apply(u_exact, mesh.p)  # P1 nodal interpolation
plot(V, dirichlet).get_figure().savefig(str(output_dir.joinpath("exact.png")))


def residual(u):
    r = b - diffusion_matrix(u) @ u
    r[boundary] = 0.0
    return r


```

```
u = np.zeros(V.N)
u[boundary] = dirichlet[boundary]
result = root(residual, u, method="krylov")

if result.success:
    u = result.x
    print("Success.  Residual =", np.linalg.norm(residual(u), np.inf))
    print("Nodal Linf error =", np.linalg.norm(u - dirichlet, np.inf))
    plot(V, u).get_figure().savefig(str(output_dir.joinpath("solution.png")))
else:
    print(result)

```

```
Success.  Residual = 1.4058151617812875e-07
Nodal Linf error = 1.2228777324096995e-08
```





FEnics exercise #7 below, executed by scikit-fem.

In [3]:

```
from pathlib import Path

import numpy as np

import skfem
from skfem.models.poisson import vector_laplace, laplace
from skfem.models.general import divergence

from meshio.xdmf import TimeSeriesWriter


@skfem.BilinearForm
def vector_mass(u, v, w):
    return sum(v * u)


@skfem.BilinearForm
def port_pressure(u, v, w):
    return sum(v * (u * w.n))


p_inlet = 8.0

```