Numerical Analysis Assignment 2

Name: Nate Stemen (20906566)
Email: nate.stemen@uwaterloo.ca

Due: Fri, Sep 25, 2020 5:00 PM
Course: AMATH 740

Problem 3. 2D model problem.

Solution. First a spy plot.

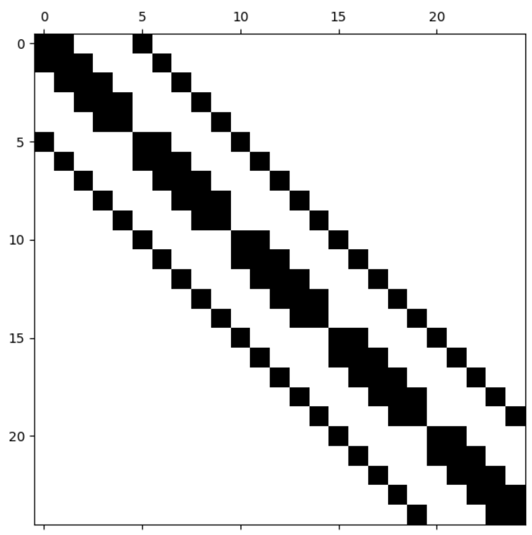


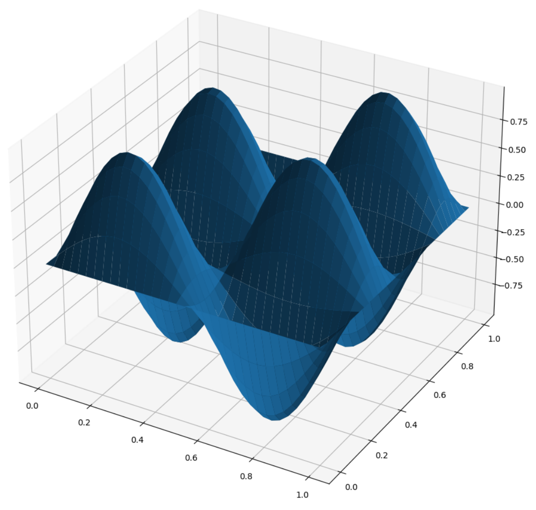
Figure 3.1: Spy plot for 2D Laplacian Matrix with $N = 5$

Here's the code from build_laplace_2d.py.

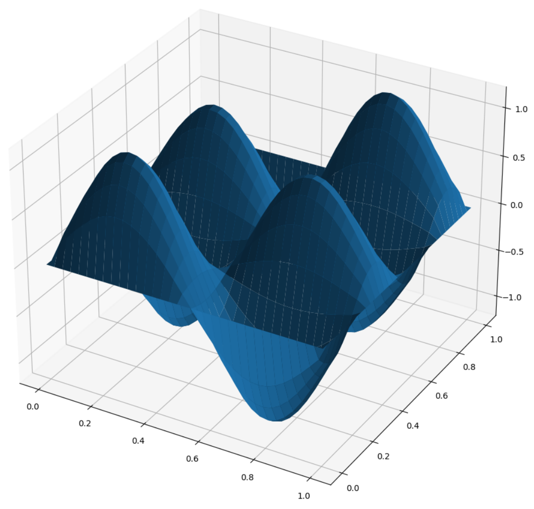
```python
import numpy as np
from scipy import sparse


def build_laplace_2D(N):
    dimension = N ** 2
    main_diag = np.full(dimension, -4.0)
    side_diag = np.ones(dimension - 1)
    side_diag[np.arange(1, dimension) % N == 0] = 0
    identity_diag = np.ones(dimension - N)
    diagonals = [main_diag, side_diag, side_diag, identity_diag,
   identity_diag]
    laplacian = sparse.diags(diagonals, [0, -1, 1, N, -N], shape=(
   dimension, dimension))
    return laplacian
```

Laplace's equation

Below are the exact and approximate solutions along with the error. Note the vertical axis has changed quite a bit for the error. Here is the code that generated these



(a) Exact solution



(b) Approximate solution

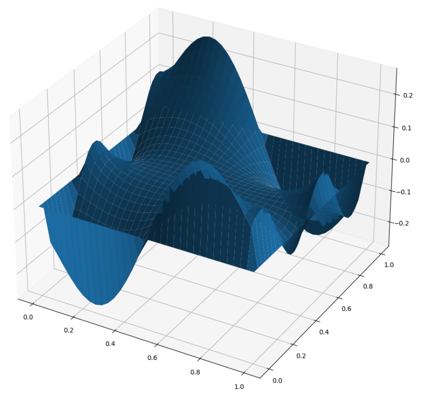


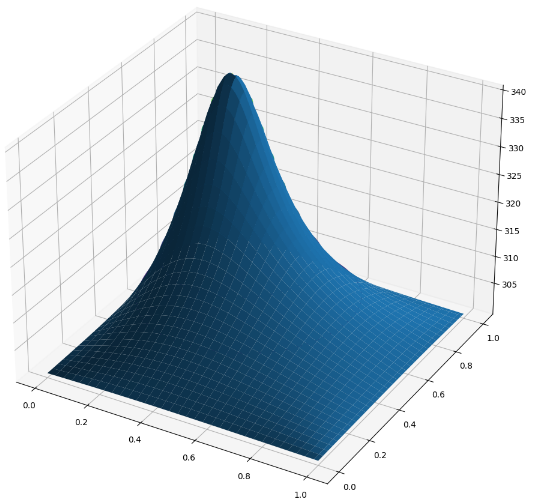
Figure 3.3: Difference of exact and approximate solution

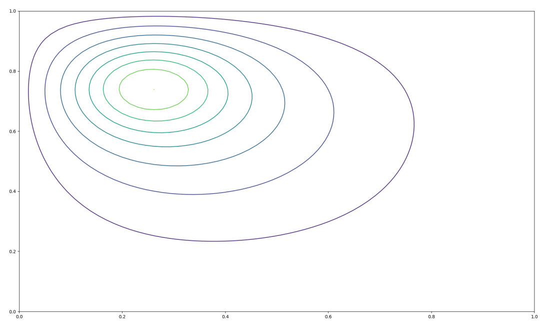
(`laplace_zeroBC.py`).

```python
from build_laplace_2d import build_laplace_2D
import numpy as np
import matplotlib.pyplot as plt

right_hand_side = (
    lambda x, y: 20 * (np.pi ** 2) * np.sin(2 * np.pi * x) * np.sin(4 *
    np.pi * y)
)

n = 32
N = n + 2
h = 1 / (n + 1)

b_vec = []

x = np.linspace(0, 1, N)
y = np.linspace(0, 1, N)

xv, yv = np.meshgrid(x, y, indexing="ij")

for i in range(N):
    for j in range(N):
        x_point, y_point = xv[i, j], yv[i, j]
        b_vec.append((-(h ** 2)) * right_hand_side(x_point, y_point))

laplace_matrix = build_laplace_2D(N).toarray()
solution = np.linalg.solve(laplace_matrix, b_vec).reshape((N, N))

solution[0, :] = np.zeros(N)
solution[-1, :] = np.zeros(N)
solution[:, 0] = np.zeros(N)
solution[:, -1] = np.zeros(N)


fig = plt.figure()
ax = fig.gca(projection="3d")

exact = np.sin(2 * np.pi * xv) * np.sin(4 * np.pi * yv)
surf = ax.plot_surface(xv, yv, solution)
surf = ax.plot_surface(xv, yv, exact)
surf = ax.plot_surface(xv, yv, exact - solution)

plt.show()
```

Heat Equation

First, the mesh/contour plots. The maximum temperature is 340.0056172901325.


(a) Mesh plot for heat equation


(b) Contour plot for heat equation

The following code generated these (`laplace_heat.py`).

```python
from build_laplace_2d import build_laplace_2D
import numpy as np
import matplotlib.pyplot as plt

right_hand_side = lambda x, y: 5000 * np.exp(-100 * ((x - 0.25) ** 2 +
    (y - 0.75) ** 2))

n = 64
N = n + 2
h = 1 / (n + 1)
```

```python

b_vec = []

x = np.linspace(0, 1, N)
y = np.linspace(0, 1, N)

xv, yv = np.meshgrid(x, y, indexing="ij")

for i in range(N):
    for j in range(N):
        x_point, y_point = xv[i, j], yv[i, j]
        b_vec.append((-(h ** 2)) * right_hand_side(x_point, y_point))

boundary_term = 300
b_vec = np.array(b_vec).reshape((N, N))
b_vec[0, :] -= boundary_term
b_vec[-1, :] -= boundary_term
b_vec[:, 0] -= boundary_term
b_vec[:, -1] -= boundary_term
b_vec = np.array(b_vec).reshape((N ** 2,))


laplace_matrix = build_laplace_2D(N).toarray()
solution = np.linalg.solve(laplace_matrix, b_vec).reshape((N, N))

max_temp = np.amax(solution)
print(f"maximum temperature acheived: {max_temp}")

fig = plt.figure()
ax = fig.gca(projection="3d")

surf = ax.plot_surface(xv, yv, solution)
cp = plt.contour(xv, yv, solution)

plt.show()
```

Problem 4. Conditioning of linear systems.

Solution. (a)

We begin by showing $\kappa_2(A)$ is always equal to 1 if A is an orthogonal matrix.

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 \tag{4.1}$$

We can expand these norms out as follows where we use $A^{-1} = A^\intercal$ because for an orthogonal matrix $AA^\intercal = A^\intercal A = \mathbf{1}$.

$$\begin{aligned}\|A\|_2^2 &= \lambda_{\max}(A^\intercal A) & \|A^{-1}\|_2^2 &= \lambda_{\max}\left(\left(A^{-1}\right)^\intercal A^{-1}\right)\\ &= \lambda_{\max}(\mathbf{1}) & &= \lambda_{\max}(AA^\intercal)\\ &= 1 & &= \lambda_{\max}(\mathbf{1}) = 1\end{aligned}$$

Plugging these two equations into eq. (4.1) we see $\kappa_2(A) = 1$ for all orthogonal A.

(b)

Here we show the matrix $A = \left[\begin{smallmatrix}1 & -1\\ 1 & 1\end{smallmatrix}\right]$ is not orthogonal.

$$AA^\intercal = \begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix} = \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix} = 2\mathbf{1} \neq \mathbf{1}$$

We can see here that $A^{-1} = \frac{1}{2}A^\intercal$ which we will use below in our calculation of $\kappa_2(A)$.

$$\begin{aligned}\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 &= \sqrt{\lambda_{\max}(A^\intercal A)}\sqrt{\lambda_{\max}\left((A^{-1})^\intercal A^{-1}\right)}\\ &= \sqrt{\lambda_{\max}(2\cdot\mathbf{1})}\sqrt{\tfrac{1}{4}\lambda_{\max}(AA^\intercal)}\\ &= \sqrt{2}\sqrt{\tfrac{1}{2}} = 1\end{aligned}$$

Because $\kappa_2(A) = 1$, we can conclude the matrix is well-conditioned.

When computing the solution to the perturbed problem, the change in solution from the unperturbed problem is very small. This is because the matrix is well conditioned (κ_2 is small)

(c)

First, is the matrix $B = \left[\begin{smallmatrix}1 & -1+\delta\\ 1 & -1\end{smallmatrix}\right]$ orthogonal?

$$BB^\intercal = \begin{bmatrix}1 & -1+\delta\\ 1 & -1\end{bmatrix}\begin{bmatrix}1 & 1\\ -1+\delta & -1\end{bmatrix} = \begin{bmatrix}1+(\delta-1)^2 & 2-\delta\\ 2-\delta & 2\end{bmatrix} \neq \mathbf{1}$$

We conclude the matrix is not orthogonal. Via python we can calculate the matrix condition number to be $\kappa_2(B) = 39{,}999{,}947{,}698.45045$, and hence we conclude the matrix is not well conditioned.

When computing the solution to the perturbed problem, the change in solution from the unperturbed problem is massive (on the order 10^6). This is because the matrix is very ill conditioned (κ_2 is *very* large).

Below is the code I used to compute things for this question (`conditioning.py`).

```python
import numpy as np

print("--------------------------")
print("------- P A R T  B -------")
print("--------------------------")

A = np.matrix([[1, -1], [1, 1]])
b = np.matrix([[1], [1]])

x = np.linalg.solve(A, b)

print("solution for unperturbed problem:")
print(x)

print(f"matrix condition number: {np.linalg.cond(A, 2)}")

perturbed_b = np.matrix([[1.001], [1]])
x_perturbed = np.linalg.solve(A, perturbed_b)

print("solution for perturbed problem:")
print(x_perturbed)

delta_x = x - x_perturbed
norm_delta_x = np.linalg.norm(delta_x, 2)
norm_x = np.linalg.norm(x, 2)

delta_b = b - perturbed_b
norm_delta_b = np.linalg.norm(delta_b, 2)
norm_b = np.linalg.norm(b, 2)

relative_cond = (norm_delta_x * norm_b) / (norm_delta_b * norm_x)

print(f"relative condition number of perturbation: {relative_cond}")

print("--------------------------")
print("------- P A R T  C -------")
print("--------------------------")

delta = 10 ** (-10)
B = np.matrix([[1, -1 + delta], [1, -1]])
b = np.matrix([[1], [1]])

x = np.linalg.solve(B, b)

print("solution for unperturbed problem:")
print(x)

print("inverse of B:")
print(np.linalg.inv(B))
print(f"matrix condition number: {np.linalg.cond(B, 2)}")

perturbed_b = np.matrix([[1.001], [1]])
x_perturbed = np.linalg.solve(B, perturbed_b)

print("solution for perturbed problem:")
print(x_perturbed)

delta_x = x - x_perturbed
```

```python
norm_delta_x = np.linalg.norm(delta_x, 2)
norm_x = np.linalg.norm(x, 2)

delta_b = b - perturbed_b
norm_delta_b = np.linalg.norm(delta_b, 2)
norm_b = np.linalg.norm(b, 2)

relative_cond = (norm_delta_x * norm_b) / (norm_delta_b * norm_x)

print(f"relative condition number of perturbation: {relative_cond}")
```

And here's what it outputs when run.

```
--------------------------
------- P A R T  B -------
--------------------------
solution for unperturbed problem:
[[1.]
 [0.]]
matrix condition number: 1.0000000000000004
solution for perturbed problem:
[[ 1.0005e+00]
 [-5.0000e-04]]
relative condition number of perturbation: 1.0
--------------------------
------- P A R T  C -------
--------------------------
solution for unperturbed problem:
[[ 1.]
 [-0.]]
inverse of B:
[[ 9.99999917e+09 -9.99999917e+09]
 [ 9.99999917e+09 -9.99999917e+09]]
matrix condition number: 39999947698.45045
solution for perturbed problem:
[[10000000.17259526]
 [ 9999999.17259526]]
relative condition number of perturbation: 19999998345.19272
```