trial-exam-f24

May 14, 2024

1 Written (Trial) Exam for 01002/01004 Mathematics 1b, Suggested Solutions

By shsp@dtu.dk, 05/05-2024

[3]:
```
from sympy import *
from dtumathtools import *

init_printing()
```

1.1 Exercise 1

We are given the two partial derivatives, so the following gradient, of a function $f : \mathbb{R}^2 \to \mathbb{R}$:

[4]:
```
x, y = symbols("x y")
fx = 6 * x - 6 * y
fy = 6 * y**2 - 6 * x
fx, fy
```

[4]:

$$(6x - 6y, \ -6x + 6y^2)$$

1.1.1 (a)

Setting them equal to zero and solving for all solutions results in all stationary points:

[5]:
```
statpt = solve([Eq(fx, 0), Eq(fy, 0)])
statpt
```

[5]:

$$[\{x : 0, \ y : 0\}, \ \{x : 1, \ y : 1\}]$$

So, f has the two stationary points, $(0, 0)$ and $(1, 1)$.

1.1.2 (b)

Second-order partial derivatives:

[6]:
```
fxx = diff(fx, x)
fxy = diff(fx, y)
fyx = diff(fy, x)
fyy = diff(fy, y)
```

```
fxx, fxy, fyx, fyy
```

[6]:

$(6,\ -6,\ -6,\ 12y)$

We see that the two partial mixed double derivatives are equal. Since f also is defined on all of $\mathbb{R}^2$, then f is two-time differentiable (smooth).

The Hessian matrix $H_f(x, y)$:

[7]:
```
H = Lambda(tuple([x, y]), Matrix([[fxx, fxy], [fyx, fyy]]))
H(x, y)
```

[7]:

$$\begin{bmatrix} 6 & -6 \\ -6 & 12y \end{bmatrix}$$

With no boundary given, extrema can only be found at stationary points or execptional points. Since f is smooth and defined on all of $\mathbb{R}^2$, there are no exceptional points. So, we investigate the eigenvalues of the Hessian matrix at the stationary points:

[8]:
```
H(0, 0).eigenvals()
```

[8]:

$$\left\{3 - 3\sqrt{5} : 1,\ 3 + 3\sqrt{5} : 1\right\}$$

The eigenvalues have different signs, so according to Theorem 5.2.4, $(0, 0)$ is a saddel point.

[9]:
```
lambdas = H(1, 1).eigenvals(multiple=True)
lambdas[0].evalf(), lambdas[1].evalf()
```

[9]:

$(2.29179606750063,\ 15.7082039324994)$

The eigenvalues are both positive, indicating a local minimum at $(1, 1)$.

There are no more possible extremum points, so f has no maximum.

1.1.3 (c)

We are now informed that $f(0, 0) = 1$. For the 2nd-degree Taylor approximating expanded from $x_0 = (0, 0)$, we need the 1st-order and 2nd-order partial derivatives evaluated at $(0, 0)$:

$$\frac{\partial f(0,0)}{\partial x} = 0, \frac{\partial f(0,0)}{\partial y} = 0, \frac{\partial^2 f(0,0)}{\partial x^2} = 6, \frac{\partial^2 f(0,0)}{\partial y^2} = 0, \frac{\partial^2 f(0,0)}{\partial x \partial y} = \frac{\partial^2 f(0,0)}{\partial y \partial x} = -6$$

Setting up the approximation:

$$P_2(x, y) = f(0,0) + \frac{\partial f(0,0)}{\partial x}(x-0) + \frac{\partial f(0,0)}{\partial y}(y-0) + \frac{1}{2}\frac{\partial^2 f(0,0)}{\partial x^2}(x-0)^2 + \frac{1}{2}\frac{\partial^2 f(0,0)}{\partial y^2}(x-0)^2 + \frac{\partial^2 f(0,0)}{\partial x \partial y}(x-0)(y-0)$$

$$= 1 + 0 + 0 + \frac{1}{2}6x^2 + 0 - 6xy$$

$$= 3x^2 - 6xy + 1$$

1.2 Exercise 2

A function $f : \mathbb{R} \to \mathbb{R}$ is given by $f(0) = 1$ and $f(x) = \sin(x)/x$ when $x \neq 0$.

1.2.1 (a)

3rd-degree Taylor polynomial of $\sin(x)$ expanded from $x_0 = 0$:

[10]:
```
sin(x).series(x, 0, 4)
```

[10]:

$$x - \frac{x^3}{6} + O\left(x^4\right)$$

So, the Taylor polynomial of degree 3 is $P_3(x) = x - \frac{x^3}{6}$.

[11]:
```
P3 = x - x**3 / 6
P3, sin(x).series(x, 0, 4).removeO()
```

[11]:

$$\left(-\frac{x^3}{6} + x,\ -\frac{x^3}{6} + x\right)$$

1.2.2 (b)

The Taylor expansion (Taylor's limit formula) of $\sin(x)$ is:

$$\sin(x) = x - \frac{x^3}{6} + \varepsilon(x)x^3$$

where $\varepsilon(x)$ is an epsilon function.

We find the following limit value:

$$\lim_{x\to 0} \frac{\sin(x)}{x} = \lim_{x\to 0} \frac{x - \frac{x^3}{6} + \varepsilon(x)x^3}{x} = \lim_{x\to 0} \left(1 - \frac{x^2}{6} + \varepsilon(x)x^2\right) = 1.$$

1.2.3 (c)

According to remark to theorem 3.1.1 in the note, f is continuous in all points in the interval $\mathbb{R}\ \{0\}$. In (b) we showed that $\sin(x)/x$ converges towards 1 for $x \to 0$. By the given definition, $f(0) = 1$, and thus $f(x) \to f(0)$ for $x \to 0$, so f is also continuous in $x = 0$.

1.2.4 (d)

Defining the function for $]0, 1]$:

[12]:
```
def f(x):
    return sin(x) / x


f(x)
```

[12]:

$\frac{\sin(x)}{x}$

Computing a decimal approximation of $\int_0^1 f(x)\,\mathrm{d}x$ using SymPy:

[13]:
```
integrate(f(x), (x, 0, 1)).evalf()
```

[13]: 0.946083070367183

1.2.5 (e)

We will compute a Riemann sum as an approximation of the area under the graph of f by subdividing the interval $[0,1]$ into $J = 30$ subintervals with equal widths of $\Delta x_j = 1/30$ and finding the right-sum. For such a sum, $x_j = j/J$ for $j = 1, \ldots, J$:

[14]:
```
j = symbols("j")

delta_xj = 1 / 30
J = 30
xj = j / J

Sum(f(xj) * delta_xj, (j, 1, 30)).evalf()
```

[14]: 0.943413033821518

Alternatively, using at for loop:

[15]:
```
riemann_sum = 0
N = 30
for i in range(1, N + 1):
    riemann_sum += sin(i / N) / (i / N) * 1 / N

riemann_sum
```

[15]: 0.943413033821518

1.2.6 (f)

Computing $\int_0^1 P_3(x)\,\mathrm{d}x$:

[16]:
```
integrate(P3, (x, 0, 1)).evalf()
```

[16]: 0.458333333333333

This approximation of the integral is worse than the approximation using a Riemann sum in the previous question, since a Taylor polynomial of $\sin(x)$ does not approximate f very well. However, it would have been sensible to use:

[17]:
```
integrate(P3 / x, (x, 0, 1)).evalf()
```

[17]: 0.944444444444444

1.3 Exercise 3

Given matrix C_t where $t \in \mathbb{R}$:

[18]:
```
t = symbols("t")
Ct = Matrix([[1, 2, 3, 4], [4, 1, 2, 3], [3, 4, 1, 2], [t, 3, 4, 1]])
Ct
```

[18]:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ t & 3 & 4 & 1 \end{bmatrix}$$

1.3.1 (a)

The unitary matrix C_t^* is the transposed and conjugated matrix. Since $t \in \mathbb{R}$, there are no non-real numbers involved, and the conjugation can be ignored. The unitary matrix is thus the transposed matrix, $C_t^* = C_t^T$:

[19]:
```
Ct_uni = Ct.T
Ct_uni
```

[19]:

$$\begin{bmatrix} 1 & 4 & 3 & t \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

C_t is a normal matrix if $C_t C_t^* = C_t^* C_t$, so if $C_t C_t^T = C_t^T C_t$, which is solved for t:

[20]:
```
Ct_uni * Ct
```

[20]:

$$\begin{bmatrix} t^2 + 26 & 3t + 18 & 4t + 14 & t + 22 \\ 3t + 18 & 30 & 24 & 22 \\ 4t + 14 & 24 & 30 & 24 \\ t + 22 & 22 & 24 & 30 \end{bmatrix}$$

[21]:
```
Ct * Ct_uni
```

[21]:

$$\begin{bmatrix} 30 & 24 & 22 & t + 22 \\ 24 & 30 & 24 & 4t + 14 \\ 22 & 24 & 30 & 3t + 18 \\ t + 22 & 4t + 14 & 3t + 18 & t^2 + 26 \end{bmatrix}$$

[22]:
```
solve(Eq(Ct * Ct_uni, Ct_uni * Ct))
```

[22]:

$[\{t : 2\}]$

So, only for $t = 2$ is C_t normal.

1.3.2 (b) and (c)

Defining $A = C_2$:

[23]:
```
A = Ct.subs(t, 2)
A
```

[23]:

$$\begin{bmatrix}1 & 2 & 3 & 4\\4 & 1 & 2 & 3\\3 & 4 & 1 & 2\\2 & 3 & 4 & 1\end{bmatrix}$$

Given eigenvectors:

[24]:
```
v1 = Matrix([1, 1, 1, 1])
v2 = Matrix([1, I, -1, -I])
```

Treating A as a mapping matrix and mapping the eigenvectors:

[25]:
```
A * v1, A * v2
```

[25]:

$$\left(\begin{bmatrix}10\\10\\10\\10\end{bmatrix}, \ \begin{bmatrix}-2-2i\\2-2i\\2+2i\\-2+2i\end{bmatrix}\right)$$

From this we read the scaling factors, which are the eigenvalues corresponding to the given eigenvectors, to be $\lambda_1 = 10$ and $\lambda_2 = -2 - 2i$:

[26]:
```
lambda1 = 10
lambda2 = -2 - 2 * I
```

Check:

[27]:
```
A * v1 == lambda1 * v1, A * v2 == simplify(lambda2 * v2)
```

[27]: (True, True)

1.3.3 (d)

Orthogonality is equivalent to an inner product of zero. The inner product of two complex vectors from $\mathbb{C}^4$ is a dot product with one vector complex conjugated, $\langle v_1, v_2\rangle = v_1 \cdot \overline{v_2}$:

[28]:
```
v1.dot(v2.conjugate())
```

[28]:

0

We conclude that they are orthogonal, $v_1 \perp v_2$.

1.3.4 (e)

The norm is the root of the inner product of a vector with itself, e.g. $||v_1|| = \sqrt{< v_1, v_1 >}$. Since $v_1 \in \mathbb{R}^4$ we can use the usual dot product without conjugation as the inner product for that one. We compute the norms of both eigenvectors:

[29]:
```
sqrt(v1.dot(v1))
```

[29]: 2

[30]:
```
sqrt(v2.dot(v2.conjugate()))
```

[30]: 2

As their norms are not 1, they are not normalized. The list v_1, v_2 is hence orthogonal but *not* orthonormal.

1.4 Exercise 4

Given quadratic form $q : \mathbb{R}^2 \to \mathbb{R}$:

[31]:
```
def q(x1, x2):
    return 2 * x1**2 - 2 * x1 * x2 + 2 * x2**2 - 4 * x1 + 2 * x2 + 2


x1, x2 = symbols("x1,x2")
q(x1, x2)
```

[31]: $2x_1^2 - 2x_1x_2 - 4x_1 + 2x_2^2 + 2x_2 + 2$

1.4.1 (a)

For rewriting to matrix form $q(x_1, x_2) = x^TAx + x^Tb + c$, then A, b and c can be as follows:

[32]:
```
A = Matrix([[2, -1], [-1, 2]])
b = Matrix([-4, 2])
c = 2
A, b, c
```

[32]: $\left(\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \begin{bmatrix} -4 \\ 2 \end{bmatrix}, 2\right)$

Checking:

[33]:
```
x = Matrix([x1, x2])

simplify(list(x.T * A * x + x.T * b)[0] + c)
```

[33]: $2x_1^2 - 2x_1x_2 - 4x_1 + 2x_2^2 + 2x_2 + 2$

[34]:
```
simplify(list(x.T * A * x + x.T * b)[0] + c) == q(x1, x2)
```

[34]: True

1.4.2 (b)

We will now reduce the quadratic form q to new form called q_1 without "mixed double terms" by changing the basis using an orthogonal change-of-basis matrix Q that changes from new to original coordinates, meaning $\tilde{x} = Q^T x$. Such Q consists of orthonormalized eigenvectors of A as columns.

[35]:
```
A.eigenvects()
```

[35]:

$$\left[\left(1, \ 1, \ \left[\begin{bmatrix} 1 \\ 1 \end{bmatrix} \right] \right), \ \left(3, \ 1, \ \left[\begin{bmatrix} -1 \\ 1 \end{bmatrix} \right] \right) \right]$$

A has the two linearly independent eigenvectors:

[36]:
```
v1 = Matrix([1, 1])
v2 = Matrix([-1, 1])
v1, v2
```

[36]:

$$\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \ \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right)$$

Also, A has a corresponding eigenvalue to each eigenvector:

[37]:
```
lambda1 = 1
lambda2 = 3
lambda1, lambda2
```

[37]:

$$(1, \ 3)$$

Since A is symmetric, then v_1 and v_2 are orthogonal, according to Theorem xx. We normalize them:

[38]:
```
q1 = v1.normalized()
q2 = v2.normalized()
q1, q2
```

[38]:

$$\left(\begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}, \ \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} \right)$$

A change-of-basis matrix Q is then:

[39]:
```
Q = Matrix.hstack(q1, q2)
Q
```

[39]:

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

This can also be found directly by

[40]:
```
Qmat, Lamda = A.diagonalize(normalize=True)
Qmat
```

[40]:

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

1.4.3 (c)

The new coordinates $\tilde{x}$ are in code denoted by k:

[41]:
```
k1, k2 = symbols("k1 k2")
k = Matrix([k1, k2])
k
```

[41]:

$$\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

In the new coordinates, the squared terms have coefficients equal to the eigenvalues of A that correspond to the eigenvectors in Q, which were found above, in the same order. We set up the new form q_1 in the new coordinates, where the original linear terms from $x^T b$ are changed to the new basis by performing $\tilde{x}^T Q^T b$:

[42]:
```
q1 = lambda1 * k1**2 + lambda2 * k2**2 + list(k.T * Q.T * b)[0] + c
q1
```

[42]:

$$k_1^2 - \sqrt{2}k_1 + 3k_2^2 + 3\sqrt{2}k_2 + 2$$

Check:

[43]:
```
simplify(list(k.T * Q.T * A * Q * k + k.T * Q.T * b)[0] + c)
```

[43]:

$$k_1^2 - \sqrt{2}k_1 + 3k_2^2 + 3\sqrt{2}k_2 + 2$$

Factorizing by completing the square gives us the following suggestions to the constants:

[44]:
```
alpha = 1
gamma = sqrt(2) / 2
beta = 3
delta = -sqrt(2) / 2
alpha, gamma, beta, delta
```

[44]:

$$\left(1, \frac{\sqrt{2}}{2}, 3, -\frac{\sqrt{2}}{2}\right)$$

Setting up the suggested factorized form of q_1 to see if it fits:

[45]:
```
q1_fact = (
    alpha * (k1 - gamma) ** 2
    - alpha * gamma**2
    + beta * (k2 - delta) ** 2
    - beta * delta**2
    + 2
)
q1_fact
```

[45]:

$$\left(k_1 - \frac{\sqrt{2}}{2}\right)^2 + 3\left(k_2 + \frac{\sqrt{2}}{2}\right)^2$$

[46]:
```
expand(q1_fact)
```

[46]:

$$k_1^2 - \sqrt{2}k_1 + 3k_2^2 + 3\sqrt{2}k_2 + 2$$

[47]:
```
expand(q1_fact) == q1
```

[47]: True

We see that the above listed four constants give us the wanted factorized form from the problem text, which is a correct factorization of q_1.

1.4.4 (d)

We are informed that q_1 in the new coordinates has a stationary point at (γ, δ) with the values of the constants found in (c):

[48]:
```
k_statpt = Matrix([gamma, delta])
k_statpt
```

[48]:

$$\begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}$$

The point written in the original coordinates:

[49]:
```
x_statpt = Q * k_statpt
x_statpt
```

[49]:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The Hessian matrix of q is by definition $H_q = 2A$. Since the eigenvalues of A are positive at all points, then the eigenvalues of H_q are also positive at all points. Thus, also positive at any stationary points. According to Theorem 5.2.4, if the point $(1, 0)$ is a stationary point, then two positive eigenvalues indicate that it is a local minimum.

1.5 Exercise 5

Given parametrization of a solid region, for $u \in [0, 1], v \in [0, 1], w \in [0, \pi/2]$:

[50]:
```
def r(u, v, w):
    return Matrix([v * u**2 * cos(w), v * u**2 * sin(w), u])


u, v, w = symbols("u v w")
r(u, v, w)
```

[50]:

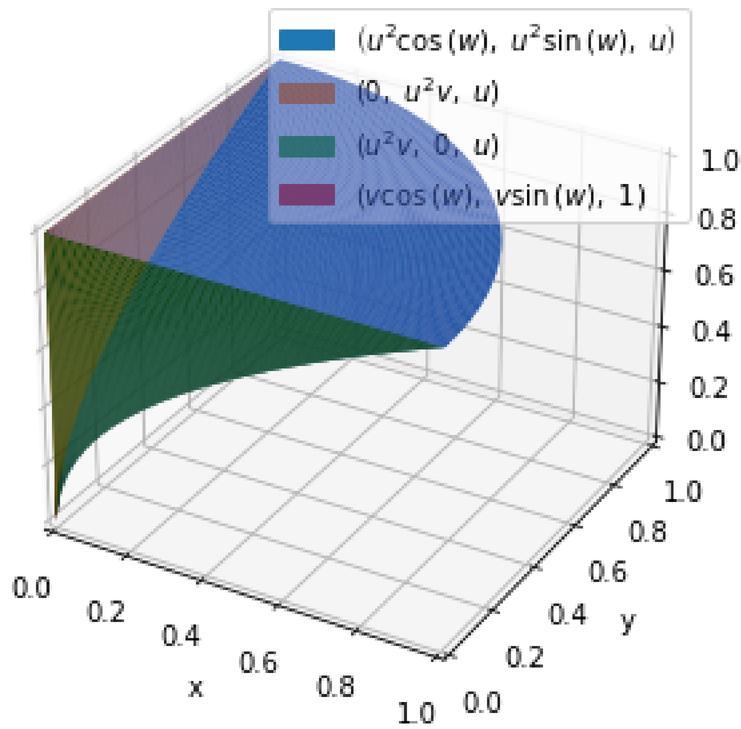
$$\begin{bmatrix}u^2 v \cos(w)\\ u^2 v \sin(w)\\ u\end{bmatrix}$$

We note that r is injective within the interior of the given parameter intervals.

1.5.1 (a)

Plotting the region:

[86]:
```
from sympy.plotting import *

pa = dtuplot.plot3d_parametric_surface(
    *r(u, v, w).subs(v, 1), (u, 0, 1), (w, 0, pi / 2), show=False
)
pb = dtuplot.plot3d_parametric_surface(
    *r(u, v, w).subs(w, pi / 2), (u, 0, 1), (v, 0, 1), show=False
)
pc = dtuplot.plot3d_parametric_surface(
    *r(u, v, w).subs(w, 0), (u, 0, 1), (v, 0, 1), show=False
)
pd = dtuplot.plot3d_parametric_surface(
    *r(u, v, w).subs(u, 1),
    (v, 0, 1),
    (w, 0, pi / 2),
    {"color": "royalblue", "alpha": 0.7},
    show=False
)
(pa + pb + pc + pd).show()
```



The Jacobian matrix:

[51]:
```
Jac_mat = Matrix.hstack(diff(r(u, v, w), u), diff(
    r(u, v, w), v), diff(r(u, v, w), w))
Jac_mat
```

[51]:

$$\begin{bmatrix} 2uv\cos(w) & u^2\cos(w) & -u^2v\sin(w) \\ 2uv\sin(w) & u^2\sin(w) & u^2v\cos(w) \\ 1 & 0 & 0 \end{bmatrix}$$

The Jacobian determinant:

[52]:
```
Jac_det = simplify(Jac_mat.det())
Jac_det
```

[52]:

u^4v

1.5.2 (b)

Given vector field:

[53]:
```
x, y, z = symbols("x y z")
V = Matrix([x + exp(y * z), 2 * y - exp(x * z), 3 * z + exp(x * y)])
V
```

[53]:

$$\begin{bmatrix} x + e^{yz} \\ 2y - e^{xz} \\ 3z + e^{xy} \end{bmatrix}$$

Given function:

[54]:
```
f = Lambda(tuple((x, y, z)), diff(V[0], x) + diff(V[1], y) + diff(V[2], z))
f(x, y, z)
```

[54]: 6

1.5.3 (c)

We see above that f is a constant and thus continuous function. A continuous function satisfying the conditions (I) and (II) on page 140,are guaranteed to be Riemann integrable, according to the remark after definition 6.3.1.

1.5.4 (d)

Since r is injective and since the Jacobian determinant is non-zero within the interior of the parameter intervals, then we can compute the volume integral of f over the solid region by integrating along the axis-parallel u, v, w region and adjusted by the Jacobian function, which is the absolute value of the Jacobian determinant in this case:

[55]:
```
integrate(f(*r(u, v, w)) * abs(Jac_det), (u, 0, 1), (v, 0, 1), (w, 0, pi / 2))
```

[55]: $\frac{3\pi}{10}$

1.6 Exercise 6

Given elevated surface: $G = \{(x, y, h(x, y)) | 0 \leq x \leq 2, 0 \leq y \leq 1\}$, where h is given as:

[56]:
```
def h(x, y):
    return 2 * x - y + 1


x, y = symbols("x y")
h(x, y)
```

[56]: $2x - y + 1$

1.6.1 (a)

Parametrisation of G:

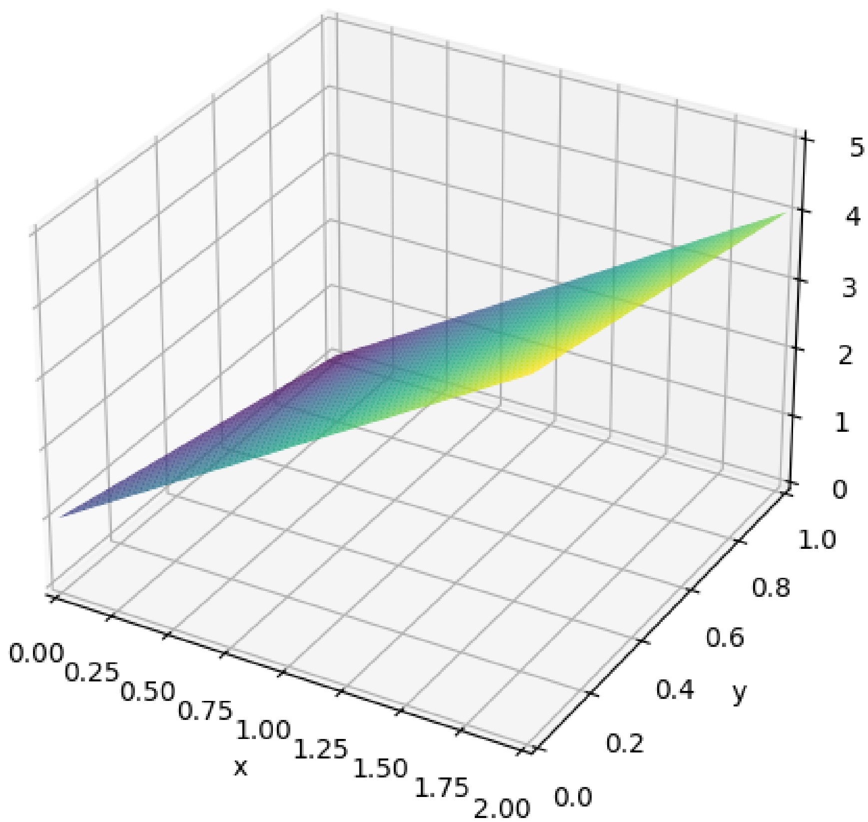
[59]:
```
r = Lambda(tuple((u, v)), Matrix([u, v, h(u, v)]))

u, v = symbols("u v")
r(u, v)
```

[59]:

$$\begin{bmatrix} u \\ v \\ 2u - v + 1 \end{bmatrix}$$

wich parameter intervals $u \in [0, 2], v \in [0, 1]$. This parametrization is injective in the interior. Plot:

[62]:
```
plot3d_parametric_surface(*r(u, v), (u, 0, 2), (v, 0, 1))
```



[62]: <sympy.plotting.plot.Plot at 0x1ade1d2a1e0>

Normal vector to the surface:

[63]:
```
N = diff(r(u, v), u).cross(diff(r(u, v), v))
N
```

[63]:
$$\begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}$$

The Jacobian function in case of surface integrals is the length (norm) of the normal vector:

[64]:
```python
Jac = N.norm()
Jac
```

[64]: $\sqrt{6}$

The area of G is found as a surface integral of the scalar 1 over the surface. Since r is injective and the Jacobian function is non-zero on the interior, then we will carry out the surface integral along u and v and adjust by the Jacobian:

[65]:
```python
integrate(Jac, (u, 0, 2), (v, 0, 1))
```

[65]: $2\sqrt{6}$

1.6.2 (b)

The region is now cut in two by a vertical plane through the points $(0,1)$ and $(2,0)$. This cuts the region in the (x,y) plane into two triangles, of which we denote the "lower" triangle by Γ_1. Parametrized, where $u \in [0,2], v \in [0,1]$:

[66]:
```python
s = Matrix([u, (1 - u/2) * v])
s
```

[66]:
$$\begin{bmatrix} u \\ v\left(1 - \frac{u}{2}\right) \end{bmatrix}$$

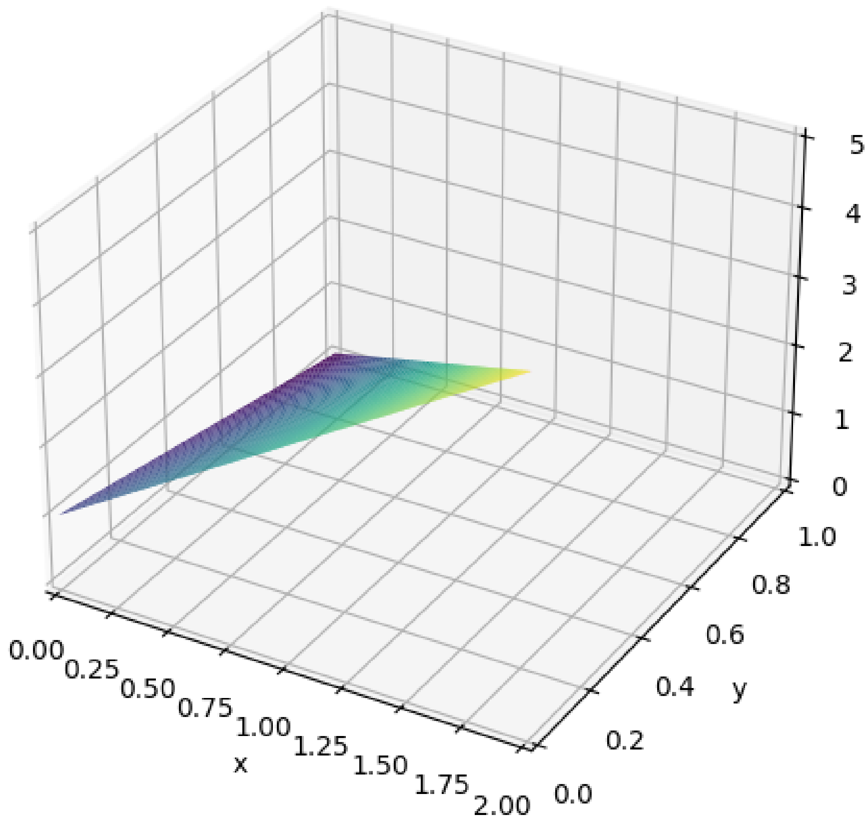
The elevated surface above Γ_1 is denoted G_1. A parametrization of G_1, where $u \in [0,2], v \in [0,1]$:

[67]:
```python
r1 = Lambda(tuple((u, v)), Matrix([*s, h(*s)]))
r1(u, v)
```

[67]:
$$\begin{bmatrix} u \\ v\left(1 - \frac{u}{2}\right) \\ 2u - v\left(1 - \frac{u}{2}\right) + 1 \end{bmatrix}$$

Plot:

[68]:
```python
plot3d_parametric_surface(*r1(u, v), (u, 0, 2), (v, 0, 1))
```



[68]: <sympy.plotting.plot.Plot at 0x1ade4632570>

Normal vector:

[69]:
```
N1 = simplify(diff(r1(u, v), u).cross(diff(r1(u, v), v)))
N1
```

[69]:

$$\begin{bmatrix} u-2 \\ 1-\frac{u}{2} \\ 1-\frac{u}{2} \end{bmatrix}$$

The Jacobian function:

[70]:
```
simplify(N1.norm())
```

[70]:

$$\frac{\sqrt{6}\,|u-2|}{2}$$

Since $u \leq 2$, we simplify to:

[71]:
```
Jac1 = -sqrt(6) * (u - 2)/2
Jac1
```

[71]:

$$-\frac{\sqrt{6}\,(u-2)}{2}$$

1.6.3 (c)

Given function

[72]:
```python
def f(x, y, z):
    return x + y + z - 1


f(x, y, z)
```

[72]:

$$x + y + z - 1$$

Surface integral of f over G_1 is performed over the parameter region since r_1 is injective and the Jacobian function non-zero on the interior of Γ_1:

[73]:
```python
integrate(f(*r1(u, v)) * Jac1, (u, 0, 2), (v, 0, 1))
```

[73]:

$$2\sqrt{6}$$