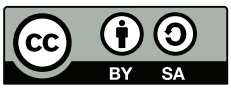
Regularization by Tikhonov Functionals

Andreas Stahel

Version of 24th May 2022



©Andreas Stahel, 2020
"Regularization by Tikhonov Functionals" by Andreas Stahel, BFH, Biel, Switzerland is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.
You are free: to copy, distribute, transmit the work, to adapt the work and to make commercial use of the work. Under the following conditions: You must attribute the work to the original author (but not in any way that suggests that the author endorses you or your use of the work). Attribute this work as follows:
Andreas Stahel: Regularization by Tikhonov Functionals
If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Contents

1 Introduction

This note is a documentation for the *Octave* codes `regularization.m` and `regularization2D.m`. With these commands a Tikhonov regularization is used to approximate given data points by smooth curves or surfaces.

- With the command `regularization()`, for given data points $(x_i, y_i) \in \mathbb{R}^2$, given functions $g_1(x)$ and $g_2(x)$ on the interval $a \le x \le b$, using the regularization parameters $\lambda_1, \lambda_2 \ge 0$, the functional

$$F(u) = \sum_{i=1}^{M} (y_i - u(x_i))^2 + \lambda_1 \int_a^b (u'(x) - g_1(x))^2 \, dx + \lambda_2 \int_a^b (u''(x) - g_2(x))^2 \, dx$$

  is minimized.

- With the command `regularization2D()`, for given data points $(x_i, y_i, z_i) \in \mathbb{R}^3$ and then regularization parameters $\lambda_1 \ge 0$, $\lambda_2 > 0$, minimize the functional

$$\begin{aligned} F(u) &:= F_D(u) + \lambda_1 F_1(u) + \lambda_2 F_2(u) \\ &= \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \iint_\Omega u_x^2 + u_y^2 \, dA + \lambda_2 \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2 \, dA \end{aligned}$$

  on a bounded domain $\Omega \subset \mathbb{R}^2$.

The organization of the notes is as follows:

- In section 2 the setup and documentation for the regularization with functions of one independent variable is explained. A few examples are given and connections to splines shown.
- In section 3 the algorithm and the resulting code is explained. The corresponding Euler–Lagrange equations are derived and the jump conditions determined.
- In section 4 the setup and documentation for the regularization with functions of two independent variables is explained.
  - The code and its documentation is shown.
  - A few examples are given, illustration essential effects.
  - An elementary proof of existence of a unique solution is given and a few remarks to similar results in the literature given. The Euler–Lagrange equations and the natural boundary conditions are derived.

2 Introduction to Regularization in One Dimension

For the data points (x_i, y_i) for $1 \le i \le M$, positive regularization parameters $0 \le \lambda_1$ and $0 \le \lambda_2$ and the target functions $g_1(x)$ and $g_2(x)$ find a function $u : [a, b] \to \mathbb{R}$ such that the Tikhonov functional

$$\begin{aligned} F(u) &= F_D(u) + \lambda_1 F_1(u) + \lambda_2 F_2(u) \\ &= \sum_{i=1}^{M} (y_i - u(x_i))^2 + \lambda_1 \int_a^b (u'(x) - g_1(x))^2 \, dx + \lambda_2 \int_a^b (u''(x) - g_2(x))^2 \, dx \end{aligned} \tag{1}$$

is minimized.

- The functional F_D pushes the optimal solution towards a function going through the given data points (x_i, y_i).
- The functional $\lambda_1 F_1$ pushes the optimal solution towards a function with derivative $g_1(x)$. For large values of λ_1 the result satisfies $u'(x) \approx g_1(x)$. For small values of $\lambda_1 > 0$ and $\lambda_2 = 0$, the data will be approximated by straight line segments between the data points.
- The functional $\lambda_2 F_2$ pushes the optimal solution towards a function with second derivative $g_2(x)$. With $g_2 = 0$ the solution is pushed towards straight line segments.

The above optimization is realized in *Octave* by using finite difference approximations for the derivatives and elementary numerical integration. A linear system of equations is used to determine discrete approximations $\vec{u} \in \mathbb{R}^{N+1}$ to the exact minimizer $u(x)$.

The built–in help is shown below and the next section provides a few examples on how to use the function `regularization()`.

```
help regularization
-->
 -- Function File: [GRID,U] = regularization (DATA, INTERVAL, N, F1, F2)

     Apply a Tikhonov regularization, the functional to be minimized is
     F = FD + LAMBDA1*F1 + LAMBDA2*F2
       = sum_(i=1)^M (y_i-u(x_i))^2 + LAMBDA1*int_a^b (u'(x) - G1(x))^2 dx
                                    + LAMBDA2*int_a^b (u"(x) - G2(x))^2 dx

     With LAMBDA1 = 0 and G2(x) = 0 this leads to a smoothing spline.

     Parameters:
        * DATA is a M*2 matrix with the x values in the first column and
          the y values in the second column.
        * INTERVAL = [a,b] is the interval on which the regularization
          is applied.
        * N is the number of subintervals of equal length.  GRID will
          consist of N+1 grid points.
        * F1 is a structure containing the information on the first
          regularization term, integrating the square of the first
          derivative.
            * F1.LAMBDA is the value of the regularization parameter
              LAMBDA1>=0.
            * F1.G is the function handle for the function G1(X).  If
              not provided G1=0 is used.
        * F2 is a structure containing the information on the second
          regularization term, integrating the square of the second
          derivative.  If F2 is not provided LAMBDA2=0 is assumed.
             * F2.LAMBDA is the value of the regularization parameter
               LAMBDA2>=0.
             * F2.G is the function handle for the function G2(X).  If
               not provided G2=0 is used.

     Return values:
        * GRID is the grid on which U is evaluated.  It consists of N+1
          equidistant points on the INTERVAL.
        * U are the values of the regularized approximation to the DATA
          evaluated at GRID.

     See also: csaps, regularization2D, demo regularization.
```

2.1 Examples

2.1.1 A first example

As a first example consider the data points $x = y = (3.2,\ 4,5,\ 5.2,\ 5.6)$ on the interval $[0, 10]$ with 101 grid points. Then two regularizations are perfomed, leading to the results in Figure 1.

1. Use $\lambda_1 = 10^{-2}$ and aim for a slope of 0.1 by setting $g_1(x) = 0.1$. For $\lambda_2 = 0$ this leads to the piecewise straight lines in Figure 1 with slope 0.1, as soon as outside of the domain with specified values.

2. For the second attempt change $\lambda_2 = 0$ to $\lambda_2 = 10^{-2} = 0.01$ leading to the smoother curve in Figure 1. Again the slope on the outside approaches 0.1 .

Observe that the values of regularized curves do **not** coincide with the given data points, just zoom into the figure.

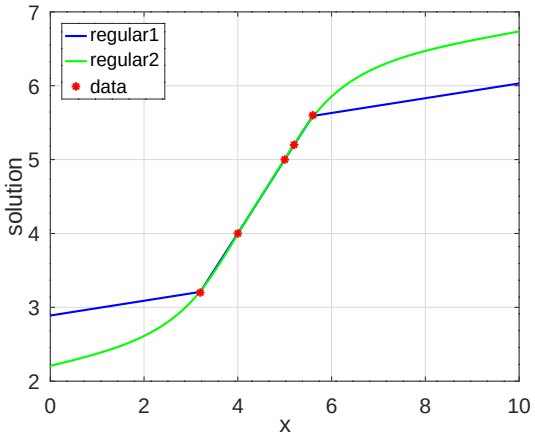


Figure 1: Results of the first example for regularization

Example1.m

```
N = 100;
interval = [0,10];
x = [3.2,4,5,5.2,5.6]'; y = x;
F1.lambda = 1e-2;
F1.g = @(x)0.1*ones(size(x));
%% regularize towards slope 0.1, no smoothing
F2.lambda = 0;
[grid,u1] = regularization([x,y],interval,N,F1);
%% regularize towards slope 0.1, with some smoothing
F2.lambda = 1e-2;
[grid,u2] = regularization([x,y],interval,N,F1,F2);

figure(1); plot(grid,u1,'b',grid,u2,'g',x,y,'*r')
           xlabel('x'); ylabel('solution');
           legend('regular1','regular2','data','location','northwest')
```

2.1.2 A second example

As a second example consider the data points $y = \sin(x)$ for 15 points $\frac{\pi}{4} \le x \le \frac{3\pi}{4}$, with some random noise added. Then two regularizations on the interval $[0, \pi]$ with 1000 subintervals are performed, leading to the results in Figure 2.

1. Use $\lambda_2 = 10^{-3}$ and $\lambda_1 = 0$, leading to the blue curve in Figure 1. Observe that the curvature outside of $\frac{\pi}{4} \le x \le \frac{3\pi}{4}$ vanishes, i.e. find straight line segments with the slope determined by the closest data points.

2. For the second attempt change to $\lambda_1 = 10^{-2}$, this causes the slope of the green curve to tend towards 0 on the outside.

Both calls use $g_1 = g_2 = 0$. Observe that the situation is asymmetric, caused by the randomness of the data points.

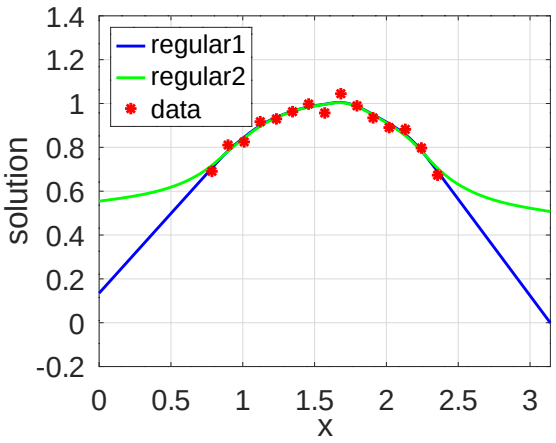


Figure 2: Results of the second example for regularization

Example2.m

```
N = 1000;                          interval = [0,pi];
x = linspace( pi/4,3*pi/4,15)';    y = sin(x)+ 0.03*randn(size(x));
clear F1 F2
F1.lambda = 0;      %% regularize by smoothing only
F2.lambda = 1e-3;
[grid,u1] = regularization([x,y],interval,N,F1,F2);
                    %% regularize by smoothing and aim for slope 0
F1.lambda = 1*1e-2;
[grid,u2] = regularization([x,y],interval,N,F1,F2);

figure(1); plot(grid,u1,'b',grid,u2,'g',x,y,'*r')
           xlabel('x'); ylabel('solution'); xlim(interval);
           legend('regular1','regular2','data','location','northwest')
```

2.1.3 A third example

As third example consider 200 data points, located on four straight line segments with slope -2, and some random noise added. Then three regularizations are performed, using F_1 only with $\lambda_1 = 0.001$, 0.1 and 3, leading to the results in Figure 3. Thus the functional to be minimized is

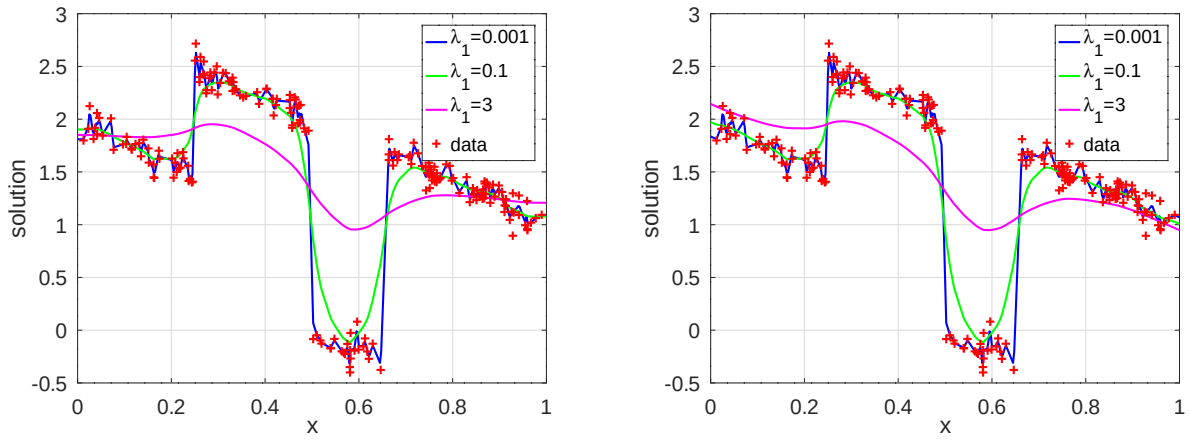
$$F(u) = \sum_{i=1}^{200} (y_i - u(x_i))^2 + \lambda_1 \int_0^1 (u'(x) - g_1(x))^2 \, dx$$

for different values of the regularization parameter $\lambda_1 > 0$ and either $g_1 = 0$ or $g_1 = -2$.

1. The blue curve in Figure 3(a) with $\lambda_1 = 0.001$ is not very regular and has many spikes. The curve is rather close to the data points. Thus $\lambda_1 = 0.001$ is on the small side.

2. The green curve in Figure 3(a) with $\lambda_1 = 0.1$ is rather smooth and not very far away from the data points. Thus $\lambda_1 = 0.1$ might be a good choice.

3. The magenta curve in Figure 3(a) with $\lambda_2 = 3$ is very smooth, but far away from the data points. The zero slopes at the end points are visible. Thus $\lambda_1 = 3$ is on the large side.



(a) $g_1(x) = 0$ (b) $g_1(x) = -2$

Figure 3: Results of the third example for regularization

Since it is "known" that the slope of the curve should be -2, rerun the above regularization using $g_1(x) = -2$, i.e. minimize

$$F(u) = \sum_{i=1}^{200} (y_i - u(x_i))^2 + \lambda_1 \int_0^1 (u'(x) + 2)^2 \, dx \, .$$

This leads to Figure 3(b), where the slopes at the endpoints are closer to the desired value of -2 .

Example3.m

```
interval = [0,1];  N = 400;
x = rand(200,1);
y = 2 - 2*x + (x>0.25) - 2*(x>0.5).*(x<0.65)+ 0.1*randn(length(x),1);
clear F1
%% F1.g = @(x)-2*ones(size(x));  %% use this when aiming for a slope -2
F1.lambda = 1e-3; [grid,u1] = regularization([x,y],interval,N,F1);
F1.lambda = 1e-1; [grid,u2] = regularization([x,y],interval,N,F1);
F1.lambda = 5e+0; [grid,u3] = regularization([x,y],interval,N,F1);

figure(1); plot(grid,u1,'b',grid,u2,'g',grid,u3,'m',x,y,'+r')
           xlabel('x'); ylabel('u and f')
           legend('\lambda_1=0.001','\lambda_1=0.1','\lambda_1=3','data')
```

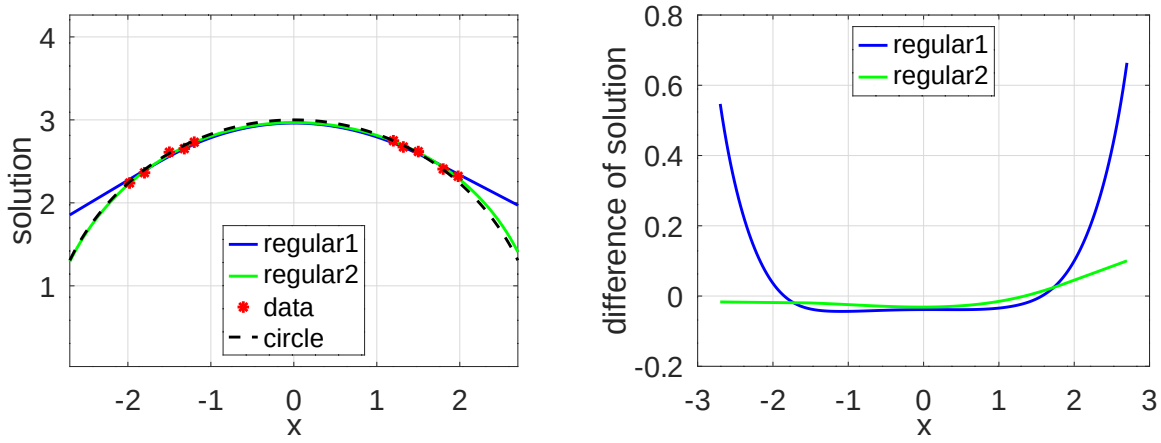
2.1.4 Regularization to a circle

If we assume that the data points should sit on a given circle, we can feed this information to the regularization algorithm by providing the function $g_2(x)$. The regularization is used twice, with the results in Figure 4.

1. First with $\lambda_1 = 0$, $\lambda_2 = 10^{-1}$ and $g_1(x) = g_2(x) = 0$.

2. Then we provide the exact second derivative of the circle function as $g_2(x)$.

Observe that outside of the domain with data points, the second result deviates considerably less from the circle.



(a) the solutions (b) difference to the circle

Figure 4: Regularization towards a given circle

CircleRegularization.m

```
R = 3;
x = R*[ 0.4, 0.44, 0.6, 0.66, 0.5];
x = [-fliplr(x),x]';
N = 200;
interval = 0.9*[-R,R];
y = sqrt(R^2-x.^2) + 0.03*randn(size(x));
clear F1 F2
F1 = 0;
F2.lambda = 1e-1;
[grid,u1] = regularization([x,y],interval,N,F1,F2);
F2.g = @(x)-R^2./(R^2-x.^2).^(3/2);
[grid,u2] = regularization([x,y],interval,N,F1,F2);

circle = sqrt(R^2-grid.^2);

figure(1);  plot(grid,u1,'b',grid,u2,'g',x,y,'*r',grid,circle,'k')
            xlabel('x'); ylabel('solution')
            axis equal
            legend('regular1','regular2','data','circle')

figure(2);  plot(grid,u1-circle,'b',grid,u2-circle,'g')
            xlabel('x'); ylabel('difference of solution')
            legend('regular1','regular2')
```

2.2 Regularization and splines

In this subsection it is pointed out that smoothing splines are a special case of the presented regularization algorithm.

2.2.1 Smoothing splines

Using **regularization()** with $\lambda_1 = g_1 = g_2 = 0$ leads to results coinciding with a smoothing spline, computed by the command `csaps()` from the splines package of *Octave*. As an example Figure 5 shows the result of a smoothing spline (with parameter $p = 0.99$) and the regularization result with $\lambda_2 = 10^{-2}$. For `csaps()` use the parameter $p = \frac{1}{1+\lambda_2} = \frac{1}{1.01} \approx 0.99$.

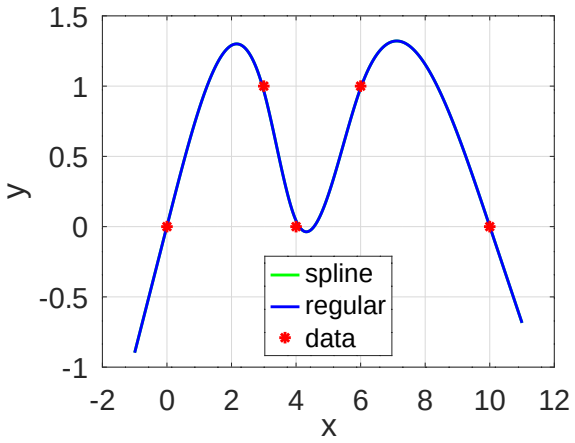


Figure 5: Comparison of regularization and a smoothing spline

SmoothingSpline.m

```
pkg load splines
N = 1000;            interval = [-1,11];
x = [0 3 4 6 10]';   y = [0 1 0 1 0]';
clear F1 F2
F1 = 0; F2.lambda = 1e-2; p = 1/(1+F2.lambda)

[grid,u] = regularization([x,y],interval,N,F1,F2);
yspline  = csaps(x,y,p,grid);  % smoothing spline

figure(1);  plot(grid,yspline,'g',grid,u,'b',x,y,'*r')
            legend('spline','regular','data','location','south')
```

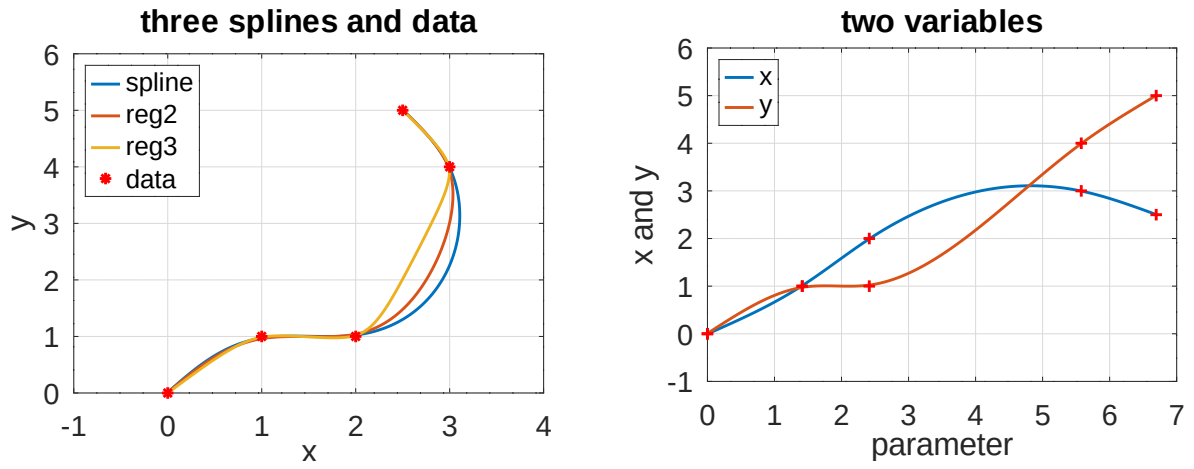
2.2.2 A 2D spline and variations by using `regularization()`

The command `regularization()` can be used to generate 2D splines, connecting points in the plane $\mathbb{R}^2$. The example below uses the points

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} , \begin{pmatrix} 1 \\ 1 \end{pmatrix} , \begin{pmatrix} 2 \\ 1 \end{pmatrix} , \begin{pmatrix} 3 \\ 4 \end{pmatrix} \text{ and } \begin{pmatrix} 2.5 \\ 5 \end{pmatrix} .$$

To obtain a uniform parametrization the distance between these points is used to construct the splines. The code uses three sets of parameters and the results are shown in Figure 6.

1. With $\lambda_1 = 0$ and $\lambda_2 = 10^{-2}$ a smoothing spline is generated, i.e. only the integral based on the second derivatives is controlled.
2. With $\lambda_1 = \lambda_2 = 10^{-2}$ the first and second derivatives are used in the Tikhonov functional.
3. With $\lambda_1 = 10^{-2}$ and $\lambda_2 = 10^{-3}$ the integral of the second derivatives is allowed to grow, which leads to sharper corners in Figure 6.



(a) splines in 2D

(b) x and y components

Figure 6: 2D splines generated by `regularization()`

Spline2D.m

```
Points = [0 0; 1 1; 2 1; 3 4; 2.5 5];  %% the given data points
%% determine distance between points
dPoints = diff(Points);
Dist = sqrt(dPoints(:,1).^2+dPoints(:,2).^2);
Positions = [0;cumsum(Dist)];
N = length(Points)*10+1; %% number of points on curve to be created

F1.lambda = 0e-2; F2.lambda = 1e-2;  %% smoothing spline
[tx,x] = regularization([Positions,Points(:,1)],[0,max(Positions)],N,F1,F2);
[ty,y] = regularization([Positions,Points(:,2)],[0,max(Positions)],N,F1,F2);

figure(2);
    plot(tx,x,ty,y,Positions,Points(:,1),'+r',Positions,Points(:,2),'+r')
    xlabel('parameter'); ylabel('x and y');
    legend('x','y'); title('the two variables')

F1.lambda = 1e-2; F2.lambda = 1e-2;  %% keep first derivative small
[tx,x2] = regularization([Positions,Points(:,1)],[0,max(Positions)],N,F1,F2);
[ty,y2] = regularization([Positions,Points(:,2)],[0,max(Positions)],N,F1,F2);

F1.lambda = 1e-2; F2.lambda = 1e-3;  %% allow large second derivative
[tx,x3] = regularization([Positions,Points(:,1)],[0,max(Positions)],N,F1,F2);
[ty,y3] = regularization([Positions,Points(:,2)],[0,max(Positions)],N,F1,F2);

figure(1); plot(x,y,x2,y2,x3,y3,Points(:,1),Points(:,2),'*r')
           xlabel('x'); ylabel('y');
           legend('spline','reg2','reg3','data')
           title('three splines and data')
```

3 The Algorithm and the Code

3.1 The computational grid

The interval $[a, b]$ is divided in N segments of equal length $\Delta x = \frac{b-a}{N}$ and $\bar{x}_j = a + (j-1)\,\Delta x$ for $j = 1, 2, 3, \ldots, N+1$. This leads to a vector $\vec{u} \in \mathbb{R}^{N+1}$ with $u_j = u(\bar{x}_j)$.

3.2 The data functional and its interpolation matrix

For $\bar{x}_j \le x < \bar{x}_{j+1}$ use a piecewise linear interpolation of the function $u(x)$ to determine an approximate value at x.

$$u(x) \approx (1-\theta)\,u(\bar{x}_j) + \theta\,u(\bar{x}_{j+1}) \quad \text{where} \quad \theta = \frac{x - \bar{x}_j}{\Delta x}\,.$$

This leads to an interpolation matrix $\mathbf{I} \in \mathbb{M}^{M\times(N+1)}$ with the nonzero entries $I_{i,j} = 1-\theta$ and $I_{i,j+1} = \theta$. For the algorithm the matrix $\mathbf{I}^T\mathbf{I} \in \mathbb{M}^{(N+1)\times(N+1)}$ will be used. The above leads to a contribution of

$$\begin{bmatrix} (1-\theta)^2 & \theta\,(1-\theta) \\ \theta\,(1-\theta) & \theta^2 \end{bmatrix}$$

in rows/columns j and $j+1$ of $\mathbf{I}^T\mathbf{I}$. This leads to

$$\begin{aligned} F_D(u) &= \sum_{i=1}^{M} (y_i - u(x_i))^2 \approx \langle \vec{y} - \mathbf{I}\,\vec{u}\,,\, \vec{y} - \mathbf{I}\,\vec{u} \rangle \\ \frac{\partial\, F_D}{\partial \vec{u}} &\approx 2\,\mathbf{I}^T(\mathbf{I}\vec{u} - \vec{y}) \end{aligned}$$

If a data point satisfies $x_i = \bar{x}_j$, then $\theta = 0$ and the j-th component of the above gradient equals $2\,(u_j - y_i)$. This is the only nonzero component.

3.3 Integration of first derivatives

3.3.1 The continuous formulation

The contribution

$$F_1(u) = \int_a^b (u'(x) - g_1(x))^2\,dx$$

to the Euler–Lagrange equation is given by $\frac{d}{dx}\,(u'(x) - g_1(x)) = 0$ and the natural boundary conditions are $u'(a) - g_1(a) = u'(b) - g_1(b) = 0$.

3.3.2 The discrete implementation

To evaluate $\int_a^b (u'(x) - g(x))^2\,dx$ use the finite difference approximation

$$u'(\frac{\bar{x}_j + \bar{x}_{j+1}}{2}) \approx \frac{u(\bar{x}_{j+1}) - u(\bar{x}_j)}{\Delta x} = \frac{u_{j+1} - u_j}{\Delta x}$$

Thus multiplying the vector $\vec{u}$ by the matrix

$$\mathbf{A}_1 = \frac{1}{\Delta x} \begin{bmatrix} -1 & 1 & & & & & \\ & -1 & 1 & & & & \\ & & -1 & 1 & & & \\ & & & \ddots & \ddots & & \\ & & & & -1 & 1 & \\ & & & & & -1 & 1 \end{bmatrix} \in \mathbb{M}^{N\times(N+1)}$$

will evaluate the first derivatives at the midpoints of the grid. In the algorithm the matrix

$$\mathbf{A}_1^T \mathbf{A}_1 = \frac{1}{(\Delta x)^2} \begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & & -1 & 1 \end{bmatrix} \in \mathbb{M}^{(N+1)\times(N+1)}$$

is used. This is the matrix generated by a finite difference method to solve $u''(x) = f(x)$ with the boundary conditions $u'(a) = u'(b) = 0$. The vector $\vec{g}_1 \in \mathbb{R}^N$ contains the values of the function $g_1(x)$ at the midpoints of the grid. Then use

$$\begin{aligned} F_1 &= \int_a^b (u'(x) - g_1(x))^2 \, dx \approx \Delta x \sum_{j=1}^{N} \left(\frac{u_{j+1} - u_j}{\Delta x} - g_j \right)^2 = \Delta x \, \langle \mathbf{A}_1 \vec{u} - \vec{g}_1 \, , \, \mathbf{A}_1 \vec{u} - \vec{g}_1 \rangle \\ \frac{\partial F_1}{\partial \vec{u}} &\approx 2 \, \Delta x \, \mathbf{A}_1^T \, (\mathbf{A}_1 \vec{u} - \vec{g}_1) \in \mathbb{R}^{N+1} \, . \end{aligned}$$

3.3.3 Boundary contributions

If the boundary conditions should be $u'(a) - g_1(a) = m_a$ and $u'(b) - g_1(b) = m_b$, then modify the functional F_1 by[1]

$$F_1(u) = \int_a^b (u'(x) - g_1(x))^2 \, dx + u(a) \, 2 \, m_a - u(b) \, 2 \, m_b \, .$$

Thus the first and last component of the gradient $\frac{\partial F_1}{\partial \vec{u}}$ obtain an additional contribution of $+2\, m_a$, resp. $-2\, m_b$.

3.4 Integration of second derivatives

3.4.1 The continuous formulation

The contribution

$$F_2(u) = \int_a^b (u''(x) - g_2(x))^2 \, dx$$

to the Euler–Lagrange equation[2] is given by $\frac{d^2}{dx^2} \, (u''(x) - g_2(x)) = 0$ and the natural boundary conditions are $u'' - g_2 = u''' - g_2' = 0$ at $x = a$ and $x = b$.

[1] Minimize $F_1(u + \varepsilon\phi)$ with respect to ε for all test functions ϕ.

$$\begin{aligned} F_1(u + \varepsilon\phi) &= \int_a^b (u' + \varepsilon\phi' - g_1)^2 \, dx + (u(a) + \varepsilon\phi(a)) \, 2 \, m_a - (u(b) + \varepsilon\phi(b)) \, 2 \, m_b \\ &= F_1(u) + 2 \, \varepsilon \int_a^b (u' - g_1) \, \phi' \, dx + 2 \, \varepsilon\phi(a) \, m_a - 2 \, \varepsilon\phi(b) \, m_b + O(\varepsilon^2) \\ &= F_1(u) + 2 \, \varepsilon \left((u' - g_1) \, \phi \Big|_{x=a}^{b} - \int_a^b (u' - g_1)' \, \phi \, dx + \phi(a) \, m_a - \phi(b) \, m_b \right) + O(\varepsilon^2) \end{aligned}$$

This leads to the Euler–Lagrange equation $(u' - g_1)' = 0$ and the boundary conditions $u'(a) - g_1(a) - m_a = 0$ and $u'(b) - g_1(b) - m_b = 0$.

[2] Two integrations by part lead to

$$F_2(u + \varepsilon\phi) = \int_a^b (u'' + \varepsilon\phi'' - g_2)^2 \, dx = F_2(u) + 2 \, \varepsilon \int_a^b (u'' - g_2) \, \phi'' \, dx + O(\varepsilon^2)$$

3.4.2 The discrete implementation

To evaluate $\int_a^b (u''(x) - g_2(x))^2\,dx$ use the finite difference approximation

$$u''(\bar{x}_j) \approx \frac{u(\bar{x}_{j-1}) - 2\,u(\bar{x}_j) + u(\bar{x}_{j+1})}{(\Delta x)^2} = \frac{u_{j-1} - 2\,u_j + u_{j+1}}{(\Delta x)^2}$$

Thus multiplying the vector $\vec{u}$ by the matrix

$$\mathbf{A}_2 = \frac{1}{\Delta x^2}\begin{bmatrix} -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \end{bmatrix} \in \mathbb{M}^{(N-1)\times(N+1)}$$

leads to approximations of the second derivatives at the interior grid points. In the algorithm use the matrix

$$\mathbf{A}_2^T\mathbf{A}_2 = \frac{1}{\Delta x^4}\begin{bmatrix} 1 & -2 & 1 & & & & & \\ -2 & 5 & -4 & 1 & & & & \\ 1 & -4 & 6 & -4 & 1 & & & \\ & 1 & -4 & 6 & -4 & 1 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & 1 & -4 & 6 & -4 & 1 \\ & & & & 1 & -4 & 5 & -2 \\ & & & & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{M}^{(N+1)\times(N+1)}$$

This is the matrix generated by a finite difference method to solve $u^{(4)}(x) = f(x)$ with the natural boundary conditions $u''(a) = u'''(a) = u''(b) = u'''(b) = 0$. The vector $\vec{g}_2 \in \mathbb{R}^{N-1}$ contains the values of the function $g_2(x)$ at the interior grid points. Then use

$$\begin{aligned} F_2 &= \int_a^b (u''(x) - g_2(x))^2\,dx \approx \Delta x \sum_{j=2}^{N}\left(\frac{u_{j-1} - 2\,u_j + u_{j+1}}{\Delta x} - g_j\right)^2 \\ &= \Delta x\,\langle \mathbf{A}_2\vec{u} - \vec{g}_2\,,\,\mathbf{A}_2\vec{u} - \vec{g}_2\rangle \\ \frac{\partial F_2}{\partial \vec{u}} &\approx 2\,\Delta x\,\mathbf{A}_2^T\,(\mathbf{A}_2\vec{u} - \vec{g}_2) \in \mathbb{R}^{N+1}\,. \end{aligned}$$

3.5 The Euler–Lagrange equation, natural boundary and jump conditions

3.5.1 The continuous formulation

Since the functional to be minimized is

$$F(u) = F_D(u) + \lambda_1\,F_1(u) + \lambda_2\,F_2(u)\,,$$

$$\begin{aligned} &= F_2(u) + 2\,\varepsilon\left((u'' - g_2)\,\phi'\Big|_{x=a}^{b} - \int_a^b (u'' - g_2)'\,\phi'\,dx\right) + O(\varepsilon^2) \\ &= F_2(u) + 2\,\varepsilon\left((u'' - g_2)\,\phi' - (u'' - g_2)'\,\phi\Big|_{x=a}^{b} + \int_a^b (u'' - g_2)''\,\phi\,dx\right) + O(\varepsilon^2)\,. \end{aligned}$$

the Euler–Lagrange equation, based on the Lagrangian $\lambda_1 F_1 + \lambda_2 F_2$, is given by

$$\lambda_2 (u''(x) - g_2(x))'' - \lambda_1 (u'(x) - g_1(x))' = 0 \,. \tag{2}$$

The natural boundary conditions are

$$0 = \lambda_2 (u'' - g_2)' - \lambda_1 (u' - g_1 - m_a) \quad \text{and} \quad 0 = -\lambda_2 (u'' - g_2) \,.$$

The discrete data (x_i, y_i) has to be taken into account too. Use the functional from equation (1) and pay special attention to the points $x = x_i$. Let $x_0 = a$ and $x_{M+1} = b$. Then use integration by parts on the sub-intervals $[x_i, x_{i+1}]$.

$$\begin{aligned}
F(u+\phi) &= F_D(u+\phi) + \lambda_1 F_1(u+\phi) + \lambda_2 F_2(u+\phi) \\
&= \sum_{i=1}^{M} (u(x_i) + \phi(x_i) - y_i)^2 + \\
&\quad + \lambda_1 \int_a^b (u'(x) + \phi'(x) - g_1(x))^2 \, dx + \lambda_2 \int_a^b (u''(x) + \phi''(x) - g_2(x))^2 \, dx \\
&\approx F(u) + 2 \sum_{i=1}^{M} (u(x_i) - y_i)\, \phi(x_i) + \\
&\quad + 2 \sum_{i=0}^{M} \int_{x_i}^{x_{i+1}} \lambda_1 (u'(x) - g_1(x))\, \phi'(x) + \lambda_2 (u''(x) - g_2(x))\, \phi''(x) \, dx \\
&= F(u) + 2 \sum_{i=1}^{M} (u(x_i) - y_i)\, \phi(x_i) + \\
&\quad + 2\,\lambda_1 \sum_{i=0}^{M} (u'(x_{i+1-}) - g_1(x_{i+1-}))\, \phi(x_{i+1-}) - (u'(x_{i+}) - g_1(x_{i+}))\, \phi(x_{i+}) \\
&\quad + 2\,\lambda_2 \sum_{i=0}^{M} (u''(x_{i+1-}) - g_1(x_{i+1-}))\, \phi'(x_{i+1-}) - (u''(x_{i+}) - g_1(x_{i+}))\, \phi'(x_{i+}) \\
&\quad + 2 \sum_{i=0}^{M} \int_{x_i}^{x_{i+1}} -\lambda_1 (u'(x) - g_1(x))'\, \phi(x) - \lambda_2 (u''(x) - g_2(x))'\, \phi'(x) \, dx \\
&= F(u) + 2 \sum_{i=1}^{M} (u(x_i) - y_i)\, \phi(x_i) + \\
&\quad + 2\,\lambda_1 \sum_{i=0}^{M} (u'(x_{i+1-}) - g_1(x_{i+1-}))\, \phi(x_{i+1-}) - (u'(x_{i+}) - g_1(x_{i+}))\, \phi(x_{i+}) \\
&\quad + 2\,\lambda_2 \sum_{i=0}^{M} (u''(x_{i+1-}) - g_2(x_{i+1-}))\, \phi'(x_{i+1-}) - (u''(x_{i+}) - g_2(x_{i+}))\, \phi'(x_{i+}) \\
&\quad - 2\,\lambda_2 \sum_{i=0}^{M} (u''(x_{i+1-}) - g_2(x_{i+1-}))'\, \phi(x_{i+1-}) - (u''(x_{i+}) - g_2(x_{i+}))'\, \phi(x_i) \\
&\quad + 2 \sum_{i=0}^{M} \int_{x_i}^{x_{i+1}} -\lambda_1 (u'(x) - g_1(x))'\, \phi(x) + \lambda_2 (u''(x) - g_2(x))''\, \phi(x) \, dx
\end{aligned}$$

To start out use test functions $\phi(x)$ vanishing at the data points $x = x_i$, i.e. $\phi_i(x_i) = \phi'(x_i) = 0$ for $i = 1, 2, 3, \ldots, M$. For $x \neq x_i$ this leads to the Euler–Lagrange equation (2). Since we are free to choose the values of $\phi(x_i)$ and $\phi'(x_i)$ there are boundary and jump conditions.

- At $x = x_0 = a$ find the natural boundary conditions

$$-\lambda_1 \left(u'(a) - g_1(a)\right) + \lambda_2 \left(u''(a) - g_2(a)\right)' = 0 \quad \text{and} \quad -\lambda_2 \left(u''(a) - g_2(a)\right) = 0\,.$$

- At $x = x_{M+1} = b$ find the natural boundary conditions

$$+\lambda_1 \left(u'(b) - g_1(b)\right) - \lambda_2 \left(u''(b) - g_2(b)\right)' = 0 \quad \text{and} \quad +\lambda_2 \left(u''(b) - g_2(b)\right) = 0\,.$$

- At a data point $x = x_i$ use $\phi(x_i) = 0$ and $\phi'(x_i) \neq 0$ to find

$$u''(x_i+) - g_2(x_i+) = u''(x_i-) - g_2(x_i-)\,.$$

  Assuming that g_2 is smooth this implies that the second derivative $u''(x)$ is continuous at $x = x_i$

- At a data point $x = x_i$ use $\phi(x_i) \neq 0$ and $\phi'(x_i) = 0$ to find

$$\begin{aligned} u(x_i) - y_i \;\; = \;\; & -\lambda_1 \left(u'(x_i-) - g_1(x_i-)\right) + \lambda_2 \left(u''(x_i-) - g_2(x_i-)\right)' \\ & +\lambda_1 \left(u'(x_i+) - g_1(x_i+)\right) - \lambda_2 \left(u''(x_i+) - g_2(x_i+)\right)'\,. \end{aligned}$$

  Assuming that g_1 and g_2 are smooth and u' and u'' are continuous this implies

$$u(x_i) - y_i = -\lambda_2 \left(u'''(x_i+) - u'''(x_i-)\right)\,.$$

  This is a jump condition for the third derivative, i.e.

$$u'''(x_i+) - u'''(x_i-) = \frac{1}{\lambda_2} \left(y_i - u(x_i)\right)\,.$$

Thus the exact minimizer $u(\cdot)$ of the functional (1) solves the Euler–Lagrange equation (2) piecewise on all sub–intervals $x_i < x < x_{i+1}$ and it satisfies the above natural boundary conditions and the jump conditions for the third derivative $u'''(x)$ at $x = x_i$.

3.5.2 A numerical example to illustrate the jumps

The above jump condition on the third derivative is illustrated by a simple example with the resulting graphs in Figure 7. Observe that the values of $\frac{1}{\lambda_2}\left(y_i - u(x_i)\right)$ equal the jumps of the third derivative.

Example4.m

```
N = 1000;  interval = [0,10];
x = [2,4,5,9]'; y = [0 0 1 0]';
F1.lambda = 1e-2; F2.lambda = 1e-2;
[grid,u] = regularization([x,y],interval,N,F1,F2);
figure(1); plot(grid,u,'g',x,y,'or')
           xlabel('x'); ylabel('solution');
           legend('regular','data','location','northwest')

u_points = (y-interp1(grid,u,x))/F2.lambda;    h = diff(interval)/N;
diff3 = fliplr([-1 2 0 -2 1]/(2*h^3));         dddu = conv(u,diff3);
figure(2); plot(grid(3:end-2),dddu(5:end-4),'-',x,u_points,'or')
           xlabel('x'); ylabel('u"')
           legend('third deriavtive','(y-u)/\lambda_2')
```

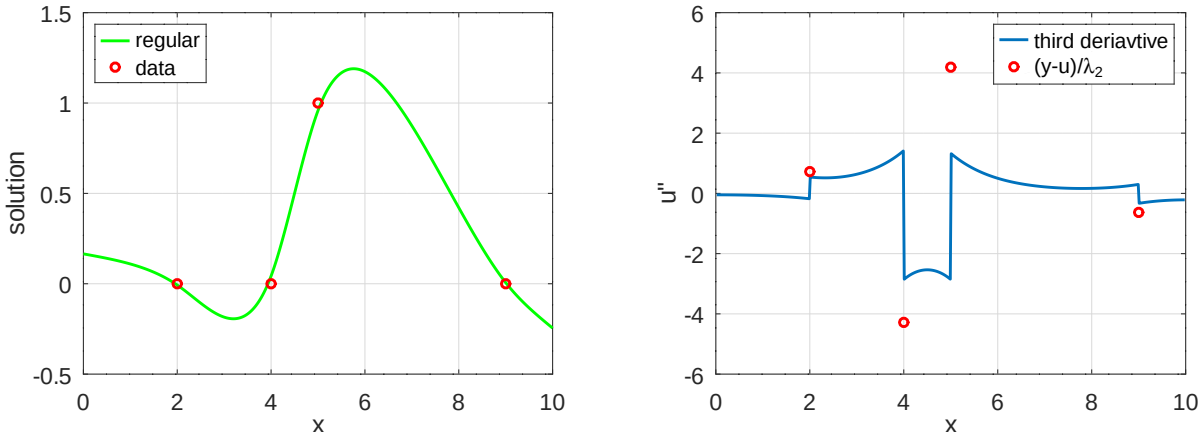


Figure 7: A regularization with the solution, the third derivative and the jumps

3.5.3 The analytical solution if $g_1(x) = g_2(x) = 0$

If $g_1(x) = g_2(x) = 0$ the Euler–Lagrange equation (2) simplifies to

$$\lambda_2\, u^{(4)}(x) - \lambda_1\, u''(x) = 0$$

on each subinterval $x_i < x < x_{i+1}$. To find the explicit solutions of the ODE use the characteristic equation

$$\lambda_2\, \alpha^4 - \lambda_1\, \alpha^2 = \alpha^2\, (\lambda_2\, \alpha^2 - \lambda_1) = 0$$

with the four solutions $\alpha_{1,2} = 0$ and $\alpha_{3,4} = \pm\sqrt{\lambda_1/\lambda_2}$. Thus the four linearly independent solution can be determined, depending on λ_1.

- If $\lambda_1 = 0$ find $\alpha_i = 0$ and the solution is a polynomial of degree 3. This is obvious, since the ODE simplfies to $u^{(4)}(x) = 0$.
- If $\lambda_1 > 0$ use the solutions of the form

$$u(x) = c_1 + c_2\, x + c_3\, \exp(+\sqrt{\frac{\lambda_1}{\lambda_2}}\, x) + c_4\, \exp(-\sqrt{\frac{\lambda_1}{\lambda_2}}\, x)$$

or

$$u(x) = c_1 + c_2\, x + c_3\, \cosh(\sqrt{\frac{\lambda_1}{\lambda_2}}\, x) + c_4\, \sinh(\sqrt{\frac{\lambda_1}{\lambda_2}}\, x)\ .$$

On each of the $M+1$ subintervals four coefficient have to be determined. The continuity of u, u', u'' and the jump condition for u''' lead to $4\,M$ equations and the natural boundary conditions at the end points add 4 more equations. Thus a system of $4\,(M+1)$ linear equations could be set up to determine the solution of the regularization problem.

3.5.4 The discrete implementation

For the functional $F(u)$ find the discretization

$$\begin{aligned}
F(u) &= F_D(u) + \lambda_1\, F_1(u) + \lambda_2\, F_2(u) \\
&= \sum_{i=1}^{M} (y_i - u(x_i))^2 + \lambda_1 \int_a^b (u'(x) - g_1(x))^2\, dx + \lambda_2 \int_a^b (u''(x) - g_2(x))^2\, dx \\
&\approx \langle \vec{y} - \mathbf{I}\vec{u}\,,\, \vec{y} - \mathbf{I}\vec{u}\rangle + \lambda_1\, \Delta x\, \langle \mathbf{A}_1\vec{u} - \vec{g}_1\,,\, \mathbf{A}_1\vec{u} - \vec{g}_1\rangle + \lambda_2\, \Delta x\, \langle \mathbf{A}_2\vec{u} - \vec{g}_2\,,\, \mathbf{A}_2\vec{u} - \vec{g}_2\rangle \\
\frac{1}{2}\,\frac{\partial F}{\partial \vec{u}} &\approx \mathbf{I}^T(\mathbf{I}\vec{u} - \vec{y}) + \lambda_1\, \Delta x\, \mathbf{A}_1^T\, (\mathbf{A}_1\vec{u} - \vec{g}_1) + \lambda_2\, \Delta x\, \mathbf{A}_2^T\, (\mathbf{A}_2\vec{u} - \vec{g}_2) \in \mathbb{R}^{N+1}\,.
\end{aligned}$$

This leads to a system of linear equations for $\vec{u} \in \mathbb{R}^{N+1}$.

$$\left(\mathbf{I}^T\mathbf{I} + \Delta x \, \left(\lambda_1 \, \mathbf{A}_1^T\mathbf{A}_1 + \lambda_2 \, \mathbf{A}_2^T\mathbf{A}_2\right)\right) \, \vec{u} = \mathbf{I}^T\vec{y} + \Delta x \, \left(\lambda_1 \, \mathbf{A}_1^T\vec{g}_1 + \lambda_2 \, \mathbf{A}_2^T\vec{g}_2\right) \in \mathbb{R}^{N+1}$$

3.6 The code for the function `regularization()`

The above algorithm is implemented in an *Octave* function `regularization.m`. Not shown below are the copyright, the documentation and the demos, just the implementation of the algorithm.

regularization.m

```
## Copyright (C) 2019 Andreas Stahel

function [grid,u] = regularization (data, interval, N, F1, F2)
  a = interval (1);  b = interval (2);
  grid = linspace (a, b, N + 1)';
  dx = grid (2) - grid (1);
  x = data (:, 1);
  ## select points in interval only
  ind = find ((x >= a) .* (x <= b));
  x = x (ind);
  y = data (:, 2);  y = y (ind);
  M = length (x);
  Interp = sparse (M, N + 1);   ## interpolation matrix
  pos = floor ((x - a) / dx) + 1;
  theta = mod ((x - a) / dx, 1);
  for ii = 1:M
    if theta (ii) > 10*eps
      Interp (ii, pos (ii)) = 1 - theta (ii);
      Interp (ii, pos (ii) + 1) = theta (ii);
    else
      Interp (ii, pos (ii)) = 1;
    endif
  endfor
  mat = Interp' * Interp;
  rhs = (Interp' * y);
  if isfield (F1, 'lambda')
    A1 = spdiags ([-ones(N, 1), +ones(N, 1)], [0, 1], N, N + 1) / dx;
    mat = mat + F1.lambda * dx * A1' * A1;
    if isfield (F1, 'g')
      g1 = F1.g (grid (1:end - 1) + dx / 2);
      rhs = rhs + F1.lambda * dx * A1' * g1;
    endif
  endif
  if exist ('F2')
    A2 = spdiags (ones (N, 1) * [1, -2, 1], [0, 1, 2], N - 1, N + 1) / dx^2;
    mat = mat + F2.lambda * dx * A2' * A2;
    if isfield (F2, 'g')
      g2 = F2.g (grid (2:end - 1));
      rhs = rhs + F2.lambda * dx * A2' * g2;
      endif
  endif
  u = mat \ rhs;
endfunction
```

4 Regularization with Two Independent Variables

For a problem with two independent variables $(x, y) \in \Omega \subset \mathbb{R}^2$ and parameters $\lambda_1 \geq 0$, $\lambda_2 > 0$ and data points $(x_i, y_i, z_i) \in \Omega \times \mathbb{R}$ the functional to be minimized is

$$\begin{aligned} F(u) &:= F_D(u) + \lambda_1 F_1(u) + \lambda_2 F_2(u) \\ &= \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \iint_\Omega u_x^2 + u_y^2 \, dA + \lambda_2 \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2 \, dA \,. \end{aligned}$$

- The functional F_D pushes the optimal solution towards a function going through the given data points (s_i, y_i, z_i).
- The functional $\lambda_1 F_1$ pushes the optimal solutions towards a horizontal plane.
- The expression F_2 corresponds to the bending energy in a thin plate. Thus $\lambda_2 F_2$ pushes the optimal solutions towards surfaces with minimal curvature.

The above optimization is realized in *Octave* by using finite difference approximations for the derivatives and elementary numerical integration. A system of linear equations is used to determine the discrete approximation $\vec{u} \in \mathbb{R}^N$ to the exact minimizer $u(x, y)$.

4.1 Documentation for the usage of the code

The built–in help is shown below and the next section provides a few examples on how to use the function `regularization2D()`.

```
help regularization2D
-->
-- Function File:
    [GRID, U, DATA_VALID] = regularization2D(DATA, BOX, N, LAMBDA1, LAMBDA2)

  Apply a Tikhonov regularization, the functional to be minimized is
  F = FD + LAMBDA1 * F1 + LAMBDA2 * F2
    = sum_(i=1)^M (y_i-u(x_i))^2+
      + LAMBDA1 * dintegral (du/dx)^2+(du/dy)^2 dA +
      + LAMBDA2 * dintegral (d^2u/dx^2)^2+(d^2u/dy^2)^2 +2*(d^2u/dxdy) dA

  With LAMBDA1 = 0 and LAMBDA2>0 this leads to a thin plate smoothing spline.

  Parameters:
    * DATA is a M*3 matrix with the (x,y) values in the first two
      columns and the z values in the third column.
      Only data points strictly inside the BOX are used
    * BOX = [x0,x1;y0,y1] is the rectangle x0<x<x1 and y0<y<y1 on
      which the regularization is applied.
    * N = [N1,N2] determines the number of subintervals of equal length.
          GRID will consist of (N1+1)x(N2+1) grid points.
    * LAMBDA1 >= 0 is the value of the first regularization parameter
    * LAMBDA2 > 0 is the value of the second regularization parameter

  Return values:
    * GRID is the grid on which U is evaluated.  It consists of
      (N1+1)x(N2+1) equidistant points on the rectangle BOX.
    * U are the values of the regularized approximation to the DATA
      evaluated on the GRID.
```

```
     * DATA_VALID returns the values data points used and the values
       of the regularized function at these points

   See also: tpaps, regularization, demo regularization2D.
```

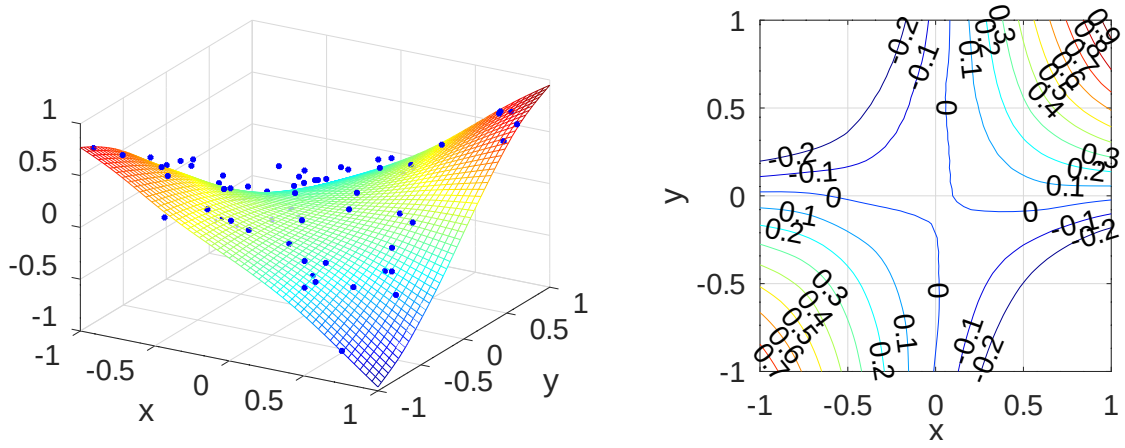
4.2 Examples

4.2.1 A first example, approximating a surface with random noise

On the square $-1 < x, y < 1$ examine points given by $z_i = x_i \cdot y_i$ with some random noise added. Then a smoothing surface is fit through those points.

Octave

```
lambda1 = 0; lambda2 = 0.05
M = 100; x = 2*rand(M,1)-1; y = 2*rand(M,1)-1;  z = x.*y + 0.1*randn(M,1);
data = [x,y,z];
[grid,u] = regularization2D(data,[-1 1;-1 1],[50 50],lambda1,lambda2);
figure(1);  mesh(grid.x, grid.y,u)
            xlabel('x'); ylabel('y');
            hold on
            plot3(data(:,1),data(:,2),data(:,3),'*b','Markersize',2)
            view([30,30]); hold off
```



(a) the surface (b) the contour lines

Figure 8: A regularized surface, determined by random data

4.2.2 A second example, determined by a few points

As a second example examine five data points, four on the x and y axis at $x, y = \pm 1$ at height 0 and at the origin at height 1. Then a regulatization on the square $-1.5 \le x, y \le +1.5$ with parameters $\lambda_1 = 0$ and $\lambda_2 = 0.05$ is applied.

Octave

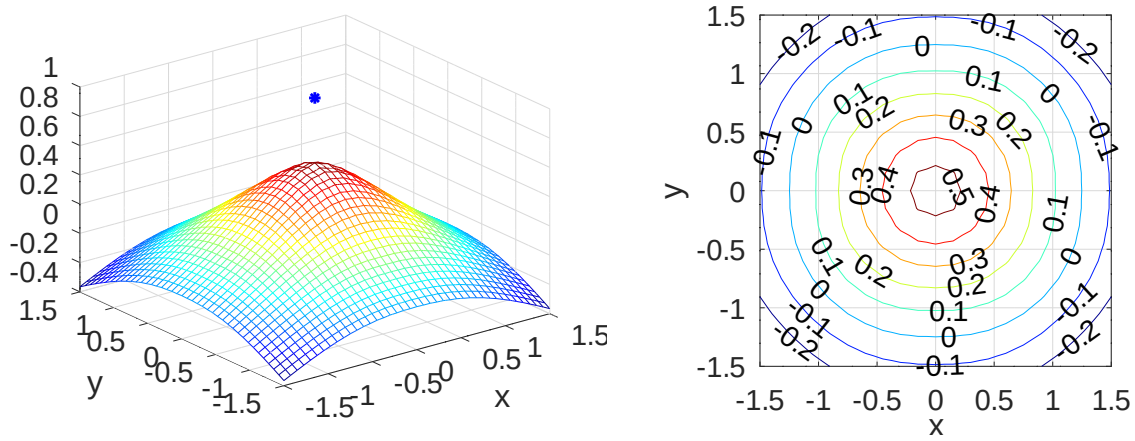
```
lambda1 = 0; lambda2 = 0.05
M = 4; angles = [1:M]/M*2*pi; data = zeros(M+1,3);
data(1:M,1) = cos(angles);  data(1:M,2) = sin(angles); data(M+1,3) = 1;
[grid,u] = regularization2D(data,[-1.5 1.5;-1.5 1.5],[40 40],lambda1,lambda2);
```

Use the usual *Octave* commands to generate the surface plot and contour lines in Figure 9.

Octave

```
figure(1)l mesh(grid.x, grid.y,u)
           xlabel('x'); ylabel('y');
           hold on
           plot3(data(:,1),data(:,2),data(:,3),'*b')
           hold off

figure(2); [c,h] = contour(grid.x, grid.y,u,[-0.2:0.1:1]); clabel(c,h);
           xlabel('x'); ylabel('y');  axis equal
```



(a) the surface

(b) contour lines

Figure 9: A surface determined by 5 data points, with regularization parameter $\lambda_2 = 0.05$

The above can be recomputed with

- a smaller parameter $\lambda_2 = 0.005$, leading to a surface with larger curvature, but closer to the data points.
- a large parameter $\lambda_2 = 0.5$, leading to a surface with smaller curvature, but further away from the data points.

Find the resulting graphs in Figure 10.

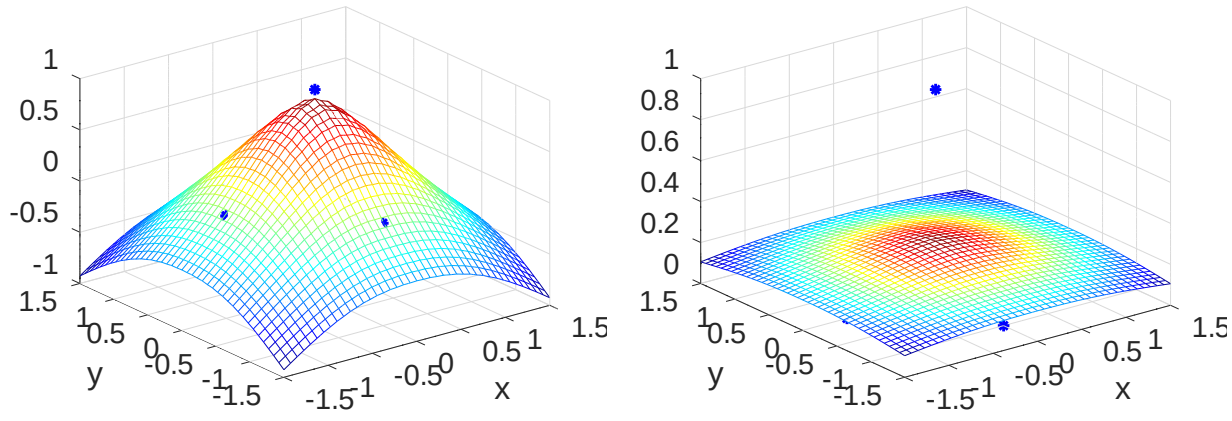
4.2.3 A third example, to illustrate that $\lambda_2 > 0$ is necessary

Remark 4–4 states that the parameter $\lambda_2 > 0$ has to be positive, otherwise the problem is not well defined and the discrete solution generated depends on the grid size used. The code below uses the same data as the previous example, but with parameters $\lambda_1 = 0.1$ and $\lambda_2 = 0$. The code is used twice, once with a 50×50 grid, and once with a 100×100 grid. Figure 11 shows the results. Observe that the contour lines for the finer grid are closer together. This is consistent with the observation in Remark 4–4 and illustrates that $\lambda_2 > 0$ is necessary to obtain well defined results.

Octave

```
lambda1 = 0.1; lambda2 = 0;
M = 4; angles = [1:M]/M*2*pi;  data = zeros(M+1,3);
data(1:M,1)= cos(angles);  data(1:M,2)= sin(angles); data(M+1,3)=1;

[grid,u] = regularization2D(data,[-1.5 1.5;-1.5 1.5],[50 50],lambda1,lambda2}
```



(a) $\lambda_2 = 0.005$ (b) $\lambda_2 = 0.5$

Figure 10: A surface determined by 5 data points with different regularization parameters λ_2

4.2.4 A fourth example, approximating a given function

Generate data on a 21×21 grid on the unit square $-1 \le x, y \le +1$ of the function $f(x, y) = \exp(-(x^2 + y^2))$. Then a regularization with $\lambda_1 = 0$ and $\lambda_2 = 0.05$ is applied on the larger rectangle $(x, y) \in [-1, +1] \times [-1, +2]$. Find the resulting surface and the difference between the regularized function and the original function $f(x, y) = \exp(-(x^2+y^2))$ in Figure 12. Observe that the difference on the unit square is rather small and for $y > 1$ the regularized surface shows very little curvature, but a clear slope.

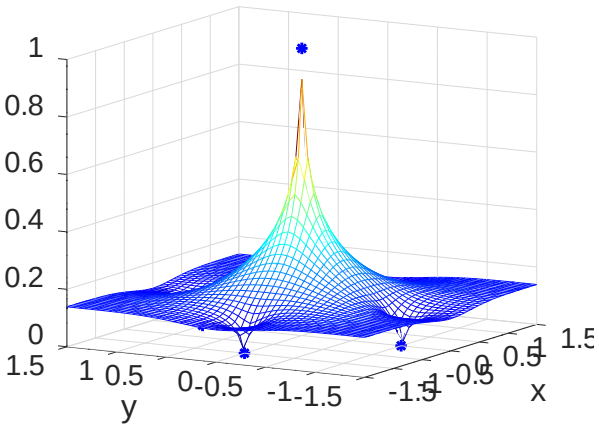
Octave

```
lambda1 = 0; lambda2 = 0.05;
N = 21;    [xx,yy] = meshgrid(linspace(-1,1,N));
function z = ff(x,y)
  z = exp(-(x.^2+y.^2));
endfunction
zz = ff(xx,yy);
data = [xx(:),yy(:),zz(:)];
Nreg = 100;
[grid,u] = regularization2D(data,[-1,1;-1 2],[Nreg,Nreg],lambda1,lambda2);
```

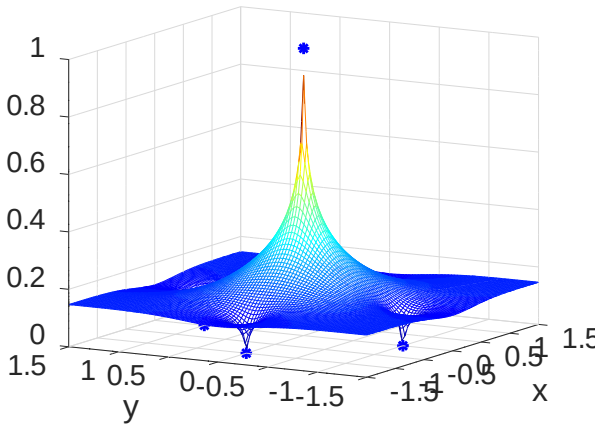
The above is repeated with the function $f(x, y) = \exp(-9\,(x^2 + y^2))$, leading to Figure 13. The graph of this function is almost horizontal on the boundary of the unit square, thus the extension of the regularized surface for $y > 1$ is almost horizontal. The curvature of $f(x, y)$ close to the origin $(0, 0)$ is large and as a consequence the regularized surface deviates from $f(x, y)$.

The same setup can be examined using the function `tpaps()` from the package splines, leading to Figure 14. Observe the that shape of the regularization is similar and the construction by `tpaps()` leads to a smaller difference to the original function $\exp(-9\,(x^2 + y^2))$.

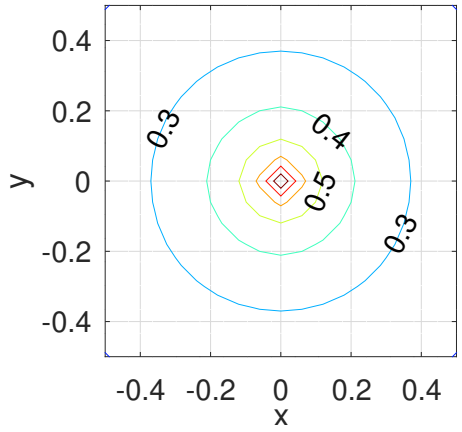
```
pkg load splines
p = 1/(1+lambda2);
[u2,p] = tpaps(data(:,1:2),data(:,3),p,[grid.x(:),grid.y(:)]);
u2 = reshape(u2,Nreg+1,Nreg+1);
```



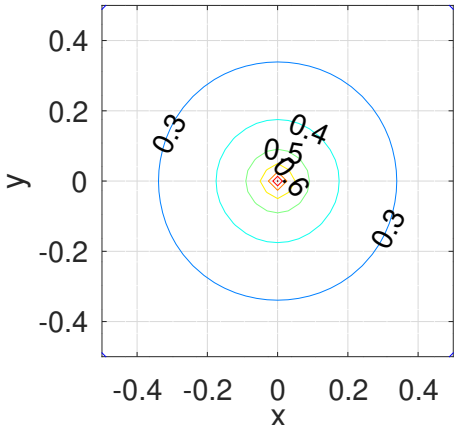
(a) Graph with a 50×50 grid



(b) Graph with a 100×100 grid



(c) Contour with a 50×50 grid



(d) Contour with a 100×100 grid

Figure 11: Regularization with $\lambda_1 = 0.1$ and $\lambda_2 = 0$ on different grids

4.2.5 A fifth example, comparing `regularization2d()` and `tpaps()`

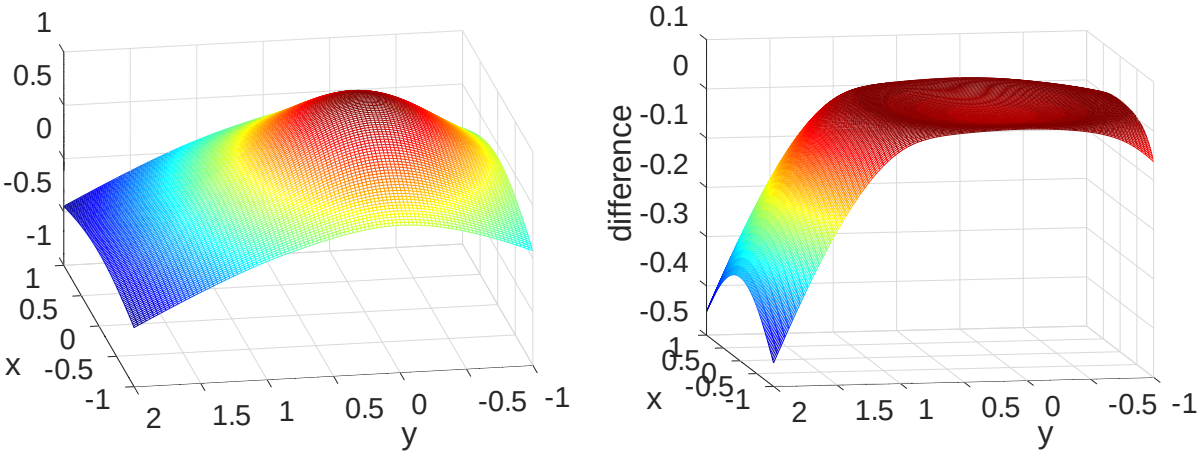
The function `tpaps()` is using smoothing splines, minimizing the bending energy of the thin plate over all of $\mathbb{R}^2$, and not the rectangle used by `regularization2D()`. Thus the functional

$$F(u) = F_D(u) + \lambda_2 F_2(u) = \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_2 \iint_{\Omega} u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2 \, dA$$

is minimized

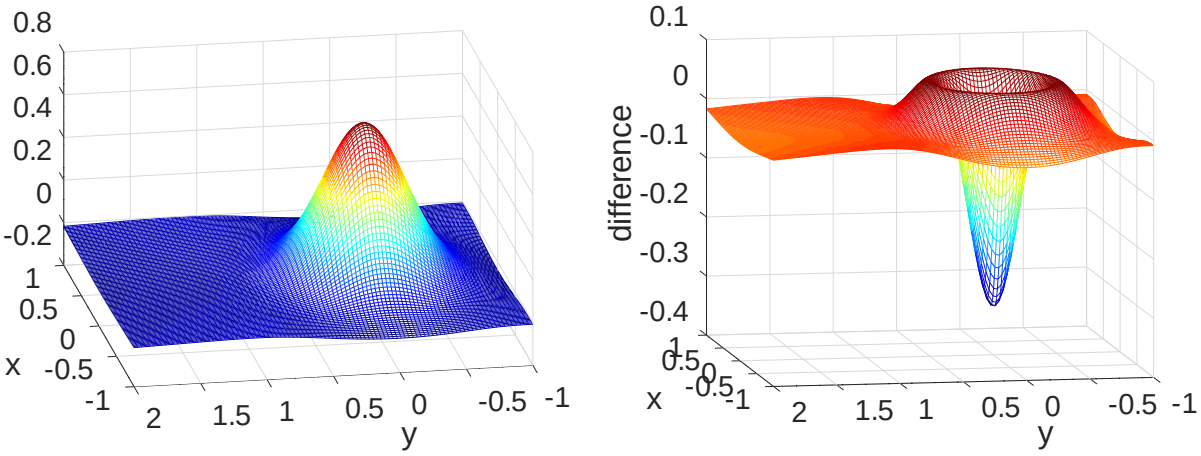
- with the rectangle $\Omega = [x_0, x_1] \times [y_0, y_1]$ for `regularization2D()`.
- with $\Omega = \mathbb{R}^2$ for `tpaps()`.

In Figure 15 find the bending plate energy $F_2(u)$ (integration over the rectangle) and the data difference $F_D(u)$ displayed as function of the parameter λ_2. Figure 16 shows that `regularization2D()` leads to smaller thin plate energies on the rectangle, compared to `tpaps()`, as is to be expected since `regularization2D()` minimizes over the rectangle.



(a) the surface $u(x, y)$ (b) the difference $u(x, y) - f(x, y)$

Figure 12: A regularized surface determined by $\exp(-(x^2 + y^2))$ on the unit square



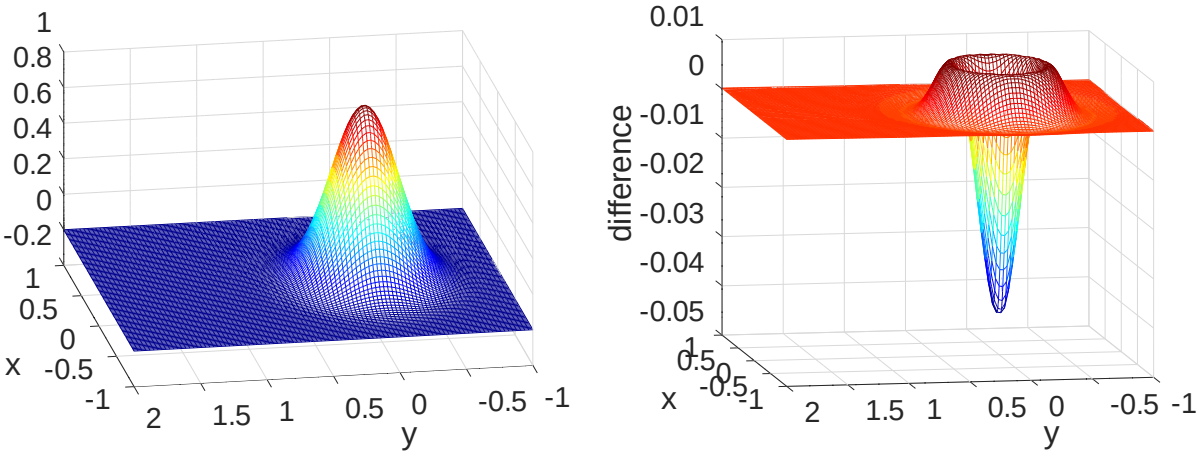
(a) the surface $u(x, y)$ (b) the difference $u(x, y) - f(x, y)$

Figure 13: A regularized surface determined by $\exp(-9\,(x^2 + y^2))$ on the unit square

4.3 The algorithm and the code for the function `regularization2D()`

The functional to be minimized is

$$
\begin{aligned}
F(u) \;&:=\; F_D(u) + \lambda_1 F_1(u) + \lambda_2 F_2(u) \\
&=\; \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \langle u, u\rangle_1 + \lambda_2 \langle u, u\rangle_2 \\
&=\; \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \iint\limits_{\Omega} (u_x - s_x)^2 + (u_y - s_y)^2 \, dA + \lambda_2 \iint\limits_{\Omega} u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2 \, dA \\
&\approx\; \langle \mathbf{Interp}\, \vec{u} - \vec{z}, \mathbf{Interp}\, \vec{u} - \vec{z}\rangle \\
&\qquad +\lambda_1 \langle (\mathbf{D}_x \vec{u} - \vec{s}_x), \mathbf{W}_x (\mathbf{D}_x \vec{u} - \vec{s}_x)\rangle + \lambda_1 \left(\langle (\mathbf{D}_y \vec{u} - \vec{s}_y), \mathbf{W}_y (\mathbf{D}_y \vec{u} - \vec{s}_y)\rangle \right) + \\
&\qquad +\lambda_2 \left(\langle \mathbf{D}_{xx} \vec{u}, \mathbf{W}_x \mathbf{D}_{xx} \vec{u}\rangle + \langle \mathbf{D}_{yy} \vec{u}, \mathbf{W}_y \mathbf{D}_{yy} \vec{u}\rangle + 2 \langle \mathbf{D}_{xy} \vec{u}, \mathbf{W}_{xy} \mathbf{D}_{xy} \vec{u}\rangle \right) \;=:\; F(\vec{u})
\end{aligned}
$$



(a) the surface $u(x, y)$ (b) the difference $u(x, y) - f(x, y)$

Figure 14: A regularized surface determined by $\exp(-9\,(x^2+y^2))$ on the unit square, using `tpaps()`

Then the necessary analytical condition $\nabla F(\vec{u}) = \vec{0}$ leads to the discrete condition

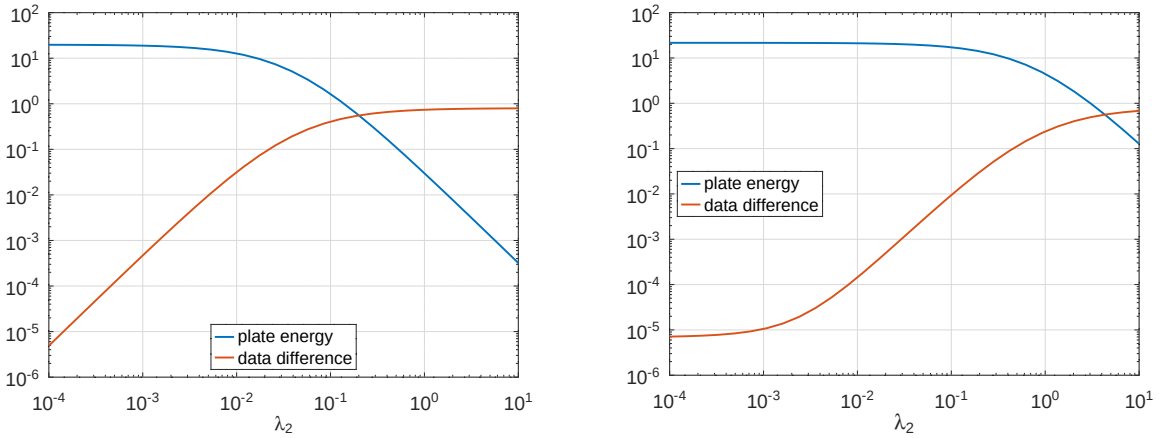
$$\begin{aligned}\mathbf{Interp}^T\,\vec{z} \;=\; & \mathbf{Interp}^T\mathbf{Interp}\,\vec{u} + \lambda_1\,\left(\mathbf{D}_x^T\mathbf{W}_x\mathbf{D}_x + \mathbf{D}_y^T\mathbf{W}_y\mathbf{D}_y\right)\,\vec{u} - \lambda_1\,\left(\mathbf{D}_x\mathbf{W}\,\vec{s}_x + \mathbf{D}_y\mathbf{W}\,\vec{s}_y\right) \\ & +\lambda_2\,\left(\mathbf{D}_{xx}^T\mathbf{W}_x\mathbf{D}_{xx} + \mathbf{D}_{yy}^T\mathbf{W}_y\mathbf{D}_{yy} + 2\,\mathbf{D}_{xy}^T\mathbf{W}_{xy}\mathbf{D}_{xy}\right)\,\vec{u}\,.\end{aligned}$$

The above algorithm is implemented in the *Octave* function `regularization2D.m`. Not shown below are the copyright, the documentation and the demos, just the implementation of the algorithm.

regularization2D.m

```
## Copyright (C) 2021 Andreas Stahel
function [grid,u,data_valid] = regularization2D (data,box,N,lambda1,lambda2)

%% generate the grid
  N = N+1;  %%%  now N is the number of grid points in either direction
  x = linspace(box(1,1),box(1,2),N(1));
  y = linspace(box(2,1),box(2,2),N(2));
  [xx,yy] = meshgrid(x,y);
  dx = diff(box(1,:))/(N(1)-1);  dy = diff(box(2,:))/(N(2)-1);
  grid.x  = xx;  grid.y  = yy;
  x = data (:,1); y = data(:,2); z = data(:,3);
  ## select points in box only
  ind = find((x>box(1,1)).*(x<box(1,2)).*(y>box(2,1)).*(y<box(2,2)));
  x = x (ind); y = y (ind); z = z (ind);
  %% generate the sparse interpolation matrix
  M = length (x);
  x_ind = floor((x-box(1,1))/dx);
  xi = mod(x,dx)/dx;
  y_ind = floor((y-box(2,1))/dy);
  nu = mod(y,dy)/dy;
  row = ones(4,1)*[1:M];
  index_base = N(2)*x_ind+y_ind+1;
  index = index_base + [0,N(2),1,N(2)+1]; index = index';
  coeff = [(1-xi).*(1-nu),xi.*(1-nu),(1-xi).*nu,xi.*nu]; coeff = coeff';
  Interp = sparse(row(:),index(:),coeff(:),M,N(1)*N(2));
  mat = Interp' * Interp;
```



(a) using `regularizion2D()` (b) using `tpaps()`

Figure 15: Comparing the contributions by the data differences F_D and the bending energy F_2, for different values of $\lambda_2 > 0$

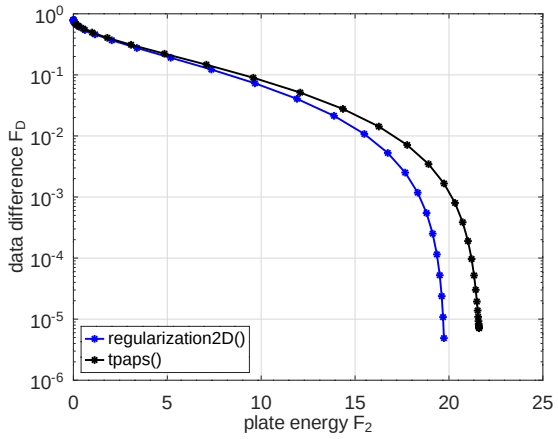


Figure 16: Comparing the energy contributions by `regularization2D()` and `tpaps()`, for different values of $\lambda_2 > 0$

```
  rhs = (Interp' * z);

%%% derivative with respect to x
  Dx = kron(spdiags(ones(N(1),1)*[-1 1],[0 1],N(1)-1,N(1))/dx,speye(N(2)));
  Wx = ones(N(2),1); Wx(1) = 1/2; Wx(N(2)) = 1/2;
  Wx = kron(speye(N(1)-1),diag(Wx))*dx*dy;
%%% derivative with respect to y
  Dy = kron(speye(N(1)),spdiags(ones(N(2),1)*[-1 1],[0 1],N(2)-1,N(2))/dy);
  Wy = ones(N(1),1); Wy(1) = 1/2; Wy(N(1)) = 1/2;
  Wy = kron(diag(Wy),speye(N(2)-1))*dx*dy;
  mat += lambda1*(Dx'*Wx*Dx + Dy'*Wy*Dy);

%%% second derivative with respect to x
  Dxx = spdiags(ones(N(1),1)*[1 -1 -1 1],[-1 0 1 2],N(1)-1,N(1));
  Dxx(1,1:4) = [3 -7 5  -1]; Dxx(N(1)-1,N(1)-3:N(1)) = [-1 5 -7 3];
  Dxx = Dxx/(2*dx^2);
```

```
  Dxx = kron(Dxx,speye(N(2)));
%%% second derivative with respect to y
  Dyy = spdiags(ones(N(2),1)*[1 -1 -1 1],[-1 0 1 2],N(2)-1,N(2));
  Dyy(1,1:4) = [3 -7 5  -1]; Dyy(N(2)-1,N(2)-3:N(2)) = [-1 5 -7 3];
  Dyy = Dyy/(2*dy^2);
  Dyy = kron(speye(N(1)),Dyy);
%%% mixed second derivative
  Dy2 = kron(speye(N(1)-1),spdiags(ones(N(2),1)*[-1 1],[0 1],N(2)-1,N(2))/dy);
  Dxy = Dy2*Dx*sqrt(dx*dy);
  mat += lambda2*(Dxx'*Wx*Dxx + Dyy'*Wy*Dyy + 2*Dxy'*Dxy);

%%%  solve
  u = reshape(mat \ rhs,N(2),N(1));
  if nargout>2
    data_valid =[x,y,Interp*u(:)];
  endif
endfunction
```

4.4 The mathematics for the result in two dimensions

4.4.1 Analytical results and the proof of unique existence

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with piecewise smooth boundary, satisfying the cone condition. Thus the usual Sobolev imbedding result apply ([Adam03]). Let $(x_i, y_i) \in \Omega$ and z_i for $i = 1, 2, \ldots, M$ be a set of data points. Let u and v be real valued functions defined on Ω and use the notations

$$\begin{aligned}
\vec{u} &:= (u_1, u_2, \ldots, u_M)^T \\
\langle \vec{u}, \vec{v} \rangle &= \sum_{i=1}^{M} u_i \, v_i \\
\langle u, v \rangle &= \iint_{\Omega} u(x,y) \, v(x,y) \, dA \\
\langle u, v \rangle_1 &= \langle u_x, v_x \rangle + \langle u_y, v_y \rangle = \iint_{\Omega} \frac{\partial \, u(x,y)}{\partial x} \frac{\partial \, v(x,y)}{\partial x} + \frac{\partial \, u(x,y)}{\partial y} \frac{\partial \, v(x,y)}{\partial y} \, dA \\
\langle u, v \rangle_2 &= \langle u_{xx}, v_{xx} \rangle + \langle u_{yy}, v_{yy} \rangle + 2 \, \langle u_{xy}, v_{xy} \rangle
\end{aligned}$$

This leads to the usual L_2 and Sobolev norms.

$$\begin{aligned}
\|u\|^2_{L_2(\Omega)} &= \langle u, u \rangle \\
\|u\|^2_{H^1(\Omega)} &= \langle u, u \rangle + \langle u, u \rangle_1 \\
\|u\|^2_{H^2(\Omega)} &= \langle u, u \rangle + \langle u, u \rangle_1 + \langle u, u \rangle_2
\end{aligned}$$

The expression

$$\langle u, u, \rangle_2 = u_{xx}^2 + u_{yy}^2 + 2 \, u_{xy}^2$$

should be invariant under rotations of the coordinate system. To verify this examine

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{bmatrix} \\
\mathbf{A}^2 &= \begin{bmatrix} u_{xx}^2 + u_{xy}^2 & (u_{xx} + u_{yy}) \, u_{xy} \\ (u_{xx} + u_{yy}) \, u_{xy} & u_{xy}^2 + u_{yy}^2 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned} \operatorname{trace}(\mathbf{A}^2) &= u_{xx}^2 + u_{yy}^2 + 2\,u_{xy}^2 \\ \langle u, u\rangle_2 &= u_{xx}^2 + u_{yy}^2 + 2\,u_{xy}^2 \;=\; \operatorname{trace}(\mathbf{A}^2) \end{aligned}$$

Now the transformation rule $\mathbf{A}' = \mathbf{R}^T\mathbf{A}\mathbf{R}$ leads to $(\mathbf{A}')^2 = \mathbf{R}^T\mathbf{A}\mathbf{R}\mathbf{R}^T\mathbf{A}\mathbf{R} = \mathbf{R}^T\mathbf{A}^2\mathbf{R}$ and since the trace equals the sum of the eigenvalues the expression $\langle u, u\rangle_2$ is invariant under rotations.[3]

To prove the main result use a Poincaré type inequality, based on the value of the function u at one point $x_0 \in \Omega$, instead of the usual zero values on the boundary $\partial\Omega$ or the average value of u (Poincaré–Wirtinger inequality).

4–1 Lemma : *For a convex, bounded domain $\Omega \subset \mathbb{R}^2$ with piecewise smooth boundary examine functions $u \in W^{1,p}(\Omega)$ with $p > 4$. For any $x_0 \in \Omega$ with $u_0 := u(x_0)$ there exists a constant c such that*

$$\|u - u_0\|_{L_2(\Omega)} \le c\,\|\nabla u\|_{L_p(\Omega)}\,, \tag{3}$$

where the constant c depends on p and Ω.

Instead of working with a convex domain $\Omega \subset \mathbb{R}^2$ one may require that the domain is star-shaped with respect to the point $x_0 \in \Omega$.

Proof : The standard Sobolev imbedding implies $u \in C^0(\overline{\Omega})$ and the evaluation at one point $x_0 \in \Omega$ is well defined. For the convex domain $\Omega \subset \mathbb{R}^2$ and a parameter $0 < t \le 1$ use a rescaled, contracted domain

$$\Omega_t := \{w = x_0 + t\,y \,|\, \text{for some } y \in \Omega\} \subset \Omega \subset \mathbb{R}^2\,.$$

Integrating the inequality

$$\begin{aligned} |u(x) - u(x_0)| &= |\int_0^1 \langle \nabla u(x_0 + t\,(x - x_0)), (x - x_0)\rangle\; dt| \le \int_0^1 \|x - x_0\|\,\|\nabla u(x_0 + t\,(x - x_0))\|\; dt \\ |u(x) - u(x_0)|^2 &\le \max_{x\in\overline{\Omega}} \|x - x_0\|^2 \int_0^1 \|\nabla u(x_0 + t\,(x - x_0))\|^2\; dt \end{aligned}$$

over the domain Ω leads to

$$\begin{aligned} \iint_\Omega |u(x) - u_0|^2\; dA_x &\le \max_{x\in\overline{\Omega}} \|x - x_0\|^2 \left(\iint_\Omega \int_0^1 \|\nabla u(x_0 + t\,(x - x_0))\|^2\; dt\; dA_x \right) \\ &= C(\Omega) \int_0^1 \left(\iint_\Omega \|\nabla u(x_0 + t\,(x - x_0))\|^2\; dA_x \right)\; dt \\ &\qquad \text{substitution } w = x_0 + t\,(x - x_0),\; dA_w = t^2\,dA, \operatorname{vol}(\Omega_t) = t^2\operatorname{vol}(\Omega) \\ &= C(\Omega) \int_0^1 \left(\iint_{\Omega_t} \frac{1}{t^2}\,\|\nabla u(w)\|^2\; dA_w \right)\; dt \end{aligned}$$

[3] In the case this should ever be tried for three independet variables:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} u_{xx} & u_{xy} & u_{xz} \\ u_{xy} & u_{yy} & u_{uz} \\ u_{xz} & u_{yz} & u_{zz} \end{bmatrix} \\ \mathbf{A}^2 &= \begin{bmatrix} u_{xx}^2 + u_{xy}^2 + u_{xz}^2 & \cdot & \cdot \\ \cdot & u_{xy}^2 + u_{yy}^2 + u_{yz}^2 & \cdot \\ \cdot & \cdot & u_{xz}^2 + u_{yz}^2 + u_{zz}^2 \end{bmatrix} \\ \operatorname{trace}(\mathbf{A}^2) &= u_{xx}^2 + u_{yy}^2 + u_{zz}^2 + 2\,(u_{xy}^2 + u_{yz}^2 + u_{zx}^2) \end{aligned}$$

$$
\begin{aligned}
& \text{Hölder's inequality on } \Omega_t \text{ with } r = \frac{p}{2}\,,\ \frac{1}{r} + \frac{1}{r'} = 1\,,\ 1 < r' < 2 \\
& \text{use } \left(\iint\limits_{\Omega_t} t^{-2\,r'}\, dA_w \right)^{1/r'} = t^{-2}\, \mathrm{vol}(\Omega_t)^{1/r'} = \mathrm{vol}(\Omega)^{1/r'}\, t^{2/r'-2} \\
\leq\ & C(\Omega) \int_0^1 c(p,\Omega)\, t^{\frac{2}{r'}-2} \left(\iint\limits_{\Omega_t} \|\nabla u(w)\|^{2\,r}\, dA_w \right)^{1/r} dt \\
\leq\ & C(\Omega)\, c(p,\Omega) \left(\int_0^1 t^{\frac{2}{r'}-2}\, dt \right) \|\nabla u\|^2_{L_p(\Omega)} \\
=\ & C(\Omega)\, c(p,\Omega)\, (\frac{2}{r'} - 1)\, \|\nabla u\|^2_{L_p(\Omega)} \quad \text{, use } \quad \frac{2}{r'} - 2 > -1 \\
\leq\ & C_1(p,\Omega)\, \|\nabla u\|^2_{L_p(\Omega)}
\end{aligned}
$$

Thus inequality (3) is verified. □

Examine a convex, bounded domain $\Omega \subset \mathbb{R}^2$ and data points $(x_i, y_i) \in \Omega$ for $i = 1, 2, 3, \ldots, M$. At those points the values of a function u should be close to given values z_i. For a function $u \in H^2(\Omega)$ the evaluation at the points (x_i, y_i) is well defined and the values $u(x_i, y_i)$ lead to a vector $\vec{u} \in \mathbb{R}^M$. For positive regularization parameters $0 < \lambda_1$ and $0 < \lambda_2$ search for a minimizer of the functional

$$
\begin{aligned}
F(u) &:= F_D(u) + \lambda_1\, F_1(u) + \lambda_2\, F_2(u) \\
&= \langle \vec{u} - \vec{z}, \vec{u} - \vec{z} \rangle + \lambda_1\, \langle u, u \rangle_1 + \lambda_2\, \langle u, u \rangle_2 \qquad (4) \\
&= \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \iint\limits_{\Omega} u_x^2 + u_y^2\, dA + \lambda_2 \iint\limits_{\Omega} u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2\, dA\,.
\end{aligned}
$$

4–2 Theorem : *If at least one data point is given, then the functional $F(u)$ in (4) has exactly one minimizer $u_* \in H^2(\Omega)$.*

Proof : Proceed in three steps: first establish an a-priori estimate, then verify existence, followed by the proof of uniqueness.

A-priori estimate:
Let $\bar{z} = \frac{1}{M} \sum_{i=1}^{M} z_i$ be the average value and then use the variance σ^2 of the z values

$$
M\, \sigma^2 = \sum_{i=1}^{M} (z_i - \bar{z})^2\,.
$$

The constant function $u(x, y) = \bar{z}$ satisfies $F(u) = M\, \sigma^2$. Any function $u \in H^2(\Omega)$ with $F(u) \leq M\, \sigma^2$ satisfies

$$
\langle u, u \rangle_1 \leq \frac{1}{\lambda_1}\, M\, \sigma^2 \quad \text{and} \quad \langle u, u \rangle_2 \leq \frac{1}{\lambda_2}\, M\, \sigma^2\,.
$$

Define a subset $\mathrm{A} \subset H^2(\Omega)$, based on $m = \max_i |z_i|$.

$$
\mathrm{A} := \{ u \in H^2(\Omega) \mid \min_{x \in \overline{\Omega}} u(x) \leq +m\,,\ \max_{x \in \overline{\Omega}} u(x) \geq -m\,,\ \langle u, u \rangle_1 \leq \frac{1}{\lambda_1}\, M\, \sigma^2\,,\ \langle u, u \rangle_2 \leq \frac{1}{\lambda_2}\, M\, \sigma^2 \}
$$

This set $A \subset H^2(\Omega)$ is not empty, since the constant function $u(x, y) = \bar{z} \in A$. The Sobolev imbedding for the gradient ∇u leads to

$$
\|\nabla u\|_{L_p(\Omega)} \leq c\, \|\nabla u\|_{H^1(\Omega)} = c\, \sqrt{\langle u, u \rangle_1 + \langle u, u \rangle_2} \quad \text{for some } 4 \leq p < \infty \quad .
$$

For the set A to be bounded in $H^2(\Omega)$ an a-priori bound on $\|u\|_{L_2(\Omega)}$ is needed. Since $u \in A$ use an $x_0 \in \Omega$ with $|u(x_0)| = |u_0| \leq m$. Inequality (3) in the above Lemma implies

$$\|u\|_{L_2(\Omega)} \leq \|u_0\|_{L_2(\Omega)} + \|u - u_0\|_{L_2(\Omega)} \leq m\,\sqrt{\mathrm{vol}(\Omega)} + \sqrt{c}\,\|\nabla u\|_{L_p(\Omega)}\,.$$

Thus the set $A \subset H^2(\Omega)$ is uniformly bounded in the L_2 norm, and consequently in the H^2 norm.

Existence:
Use a minimizing sequence $u_n \in A$ of the functional $F(u)$ in (4). Based on the above a-priori estimate the sequence is bounded in $H^2(\Omega)$. Since $H^2(\Omega)$ is compactly imbedded in $W^{1,p}(\Omega)$ for some $p > 4$, there exists a convergent subsequence (again denoted by u_n) in $W^{1,p}(\Omega)$ and $u_n \longrightarrow u_*$. Since $H^2(\Omega)$ is a Hilbert space there exists a weakly convergent subsequence u_n and $u_* \in H^2(\Omega)$. The weak convergence implies

$$\langle \vec{u}_n - \vec{z}, \vec{u}_* - \vec{z}\rangle + \lambda_1 \langle u_n, u_*\rangle_1 + \lambda_2 \langle u_n, u_*\rangle_2 \overset{n\to\infty}{\longrightarrow} \langle \vec{u}_* - \vec{z}, \vec{u}_* - \vec{z}\rangle + \lambda_1 \langle u_*, u_*\rangle_1 + \lambda_2 \langle u_*, u_*\rangle_2 = F(u_*)$$

and $u_* \in H^2(\Omega)$ is a minimizer of the functional F.

Uniqueness:
Assume $u_1 \in H^2(\Omega)$ is a second minimizer. Let $v = u_1 - u_*$ and examine

$$\begin{aligned} g(t) &:= F(u_* + t\,v) \\ &= \langle \vec{u}_* + t\,\vec{v} - \vec{z}, \vec{u}_* + t\,\vec{v} - \vec{z}\rangle + \lambda_1 \langle u_* + t\,v, u_* + t\,v\rangle_1 + \lambda_2 \langle u_* + t\,v, u_* + t\,v\rangle_2\,. \end{aligned}$$

Thus g is a polynomial of degree 2 with $g(0) = g(1) = F(u_*)$. Since u_* is a minimizer, conclude $\frac{d^2 g}{dt^2} = 0$, i.e. $0 = \langle \vec{v}, \vec{v}\rangle + \lambda_1 \langle v, v\rangle_1 + \lambda_2 \langle v, v\rangle_2$. Thus $\langle v, v\rangle_1 = \iint\limits_\Omega \|\nabla v\|^2\, dA = 0$ and the function v has to be a constant. Then $\langle \vec{v}, \vec{v}\rangle = 0$ implies that $v = 0$ and thus the minimizer u_* is unique. □

4–3 Remark : The Euler–Lagrange equation and natural boundary conditions
To find the corresponding PDE and the natural boundary conditions use standard calculus of variations arguments. For smooth perturbations ϕ examine

$$\begin{aligned} F(u+\phi) &= \langle \vec{u} + \vec{\phi} - \vec{z}, \vec{u} + \vec{\phi} - \vec{z}\rangle + \lambda_1 \iint\limits_\Omega \|\nabla u + \nabla\phi\|^2\, dA + \\ &\quad + \lambda_2 \iint\limits_\Omega (u_{xx} + \phi_{xx})^2 + (u_{yy} + \phi_{yy})^2 + 2\,(u_{xy} + \phi_{xy})^2\, dA \\ &\approx F(u) + 2\,\langle \vec{u} - \vec{z}, \vec{\phi}\rangle + 2\,\lambda_1 \iint\limits_\Omega \langle \nabla u, \nabla\phi\rangle\, dA + \\ &\quad + 2\,\lambda_2 \iint\limits_\Omega u_{xx}\phi_{xx} + u_{yy}\phi_{yy} + 2\,u_{xy}\phi_{xy}\, dA \\ &= F(u) + 2\,\langle \vec{u} - \vec{z}, \vec{\phi}\rangle + 2\,\lambda_1 \iint\limits_\Omega \langle \nabla u, \nabla\phi\rangle\, dA + \\ &\quad + 2\,\lambda_2 \iint\limits_\Omega + \langle \begin{pmatrix} u_{xx} \\ u_{xy} \end{pmatrix}, \nabla\phi_x\rangle + \langle \begin{pmatrix} u_{xy} \\ u_{yy} \end{pmatrix}, \nabla\phi_y\rangle\, dA \\ &= F(u) + 2\,\langle \vec{u} - \vec{z}, \vec{\phi}\rangle + 2\,\lambda_1 \oint_{\partial\Omega} \phi \cdot \langle \nabla u, \vec{n}\rangle\, ds - 2\,\lambda_1 \iint\limits_\Omega \Delta u \cdot \phi\, dA \\ &\quad + 2\,\lambda_2 \oint_{\partial\Omega} \langle \begin{pmatrix} u_{xx} \\ u_{xy} \end{pmatrix}, \vec{n}\rangle\, \phi_x + \langle \begin{pmatrix} u_{xy} \\ u_{yy} \end{pmatrix}, \vec{n}\rangle\, \phi_y\, ds - \end{aligned}$$

$$
\begin{aligned}
& -2\,\lambda_2 \iint_\Omega \langle \begin{pmatrix} u_{xxx}+u_{xyy} \\ u_{xxy}+u_{yyy} \end{pmatrix}, \nabla\phi\rangle \, dA \\
= \;& F(u) + 2\,\langle \vec{u}-\vec{z}, \vec{\phi}\rangle + 2\,\lambda_1 \oint_{\partial\Omega} \phi \cdot \langle \nabla u, \vec{n}\rangle \, ds - 2\,\lambda_1 \iint_\Omega \Delta u \cdot \phi \, dA \\
& +2\,\lambda_2 \oint_{\partial\Omega} \langle \begin{pmatrix} u_{xx} \\ u_{xy} \end{pmatrix}, \vec{n}\rangle \, \phi_x + \langle \begin{pmatrix} u_{xy} \\ u_{yy} \end{pmatrix}, \vec{n}\rangle \, \phi_y - \langle \begin{pmatrix} u_{xxx}+u_{xyy} \\ u_{xxy}+u_{yyy} \end{pmatrix}, \vec{n}\rangle \, \phi \, ds + \\
& +2\,\lambda_2 \iint_\Omega (u_{xxxx} + 2\,u_{xxyy} + u_{yyyy})\,\phi \, dA
\end{aligned}
$$

The resulting Euler–Lagrange equation is

$$
-\lambda_1 \,\Delta u + \lambda_2 \,\Delta^2 u = \Delta\,(-\lambda_1\, u + \lambda_2\,\Delta u) = 0\,. \tag{5}
$$

The effect of the data at (x_i, y_i) is not taken into account with the above Euler–Lagrange equation and the equation is valid away from (x_i, y_i).

To derive the natural boundary conditions use [LandLifs75, §12, p.50ff]. Examine directional derivatives on the boundary $\partial\Omega$ in normal direction $\vec{n}$ and tangential direction $\vec{t}$.

$$
\begin{aligned}
\vec{n} &= \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \qquad \frac{\partial}{\partial n} := n_1 \frac{\partial}{\partial x} + n_2 \frac{\partial}{\partial y} \\
\vec{t} &= \begin{pmatrix} -n_2 \\ n_1 \end{pmatrix} = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}, \qquad \frac{\partial}{\partial t} := -n_2 \frac{\partial}{\partial x} + n_1 \frac{\partial}{\partial y} \\
\frac{\partial}{\partial x} &= n_1 \frac{\partial}{\partial n} - n_2 \frac{\partial}{\partial t}, \qquad \frac{\partial}{\partial y} = n_2 \frac{\partial}{\partial n} + n_1 \frac{\partial}{\partial t}
\end{aligned}
$$

One of the above boundary contributions can be written in the form

$$
\begin{aligned}
\text{BC} &= \oint_{\partial\Omega} \langle \begin{pmatrix} u_{xx} \\ u_{xy} \end{pmatrix}, \vec{n}\rangle \, \phi_x + \langle \begin{pmatrix} u_{xy} \\ u_{yy} \end{pmatrix}, \vec{n}\rangle \, \phi_y \, ds \\
&= \oint_{\partial\Omega} u_{xx}\, n_1\, \phi_x + u_{xy}\, n_2\, \phi_x + u_{xy}\, n_1\, \phi_y + u_{yy}\, n_2\, \phi_y \, ds \\
&= \oint_{\partial\Omega} (u_{xx}+u_{yy})\, n_1\, \phi_x - u_{yy}\, n_1\, \phi_x + u_{xy}\, n_2\, \phi_x + u_{xy}\, n_1\, \phi_y + (u_{xx}+u_{yy})\, n_2\, \phi_y - u_{xx}\, n_2\, \phi_y \, ds \\
&= \oint_{\partial\Omega} \Delta u \frac{\partial\,\phi}{\partial n} + (u_{xy}\,\phi_y - u_{yy}\,\phi_x)\, n_1 - (u_{xx}\,\phi_y - u_{xy}\,\phi_x)\, n_2 \, ds \\
&= \oint_{\partial\Omega} \Delta u \frac{\partial\,\phi}{\partial n} + (u_{xy}\,(n_2 \frac{\partial\,\phi}{\partial n} + n_1 \frac{\partial\,\phi}{\partial t}) - u_{yy}\,(n_1 \frac{\partial\,\phi}{\partial n} - n_2 \frac{\partial\,\phi}{\partial t}))\, n_1 - \\
&\qquad -(u_{xx}\,(n_2 \frac{\partial\,\phi}{\partial n} + n_1 \frac{\partial\,\phi}{\partial t}) + u_{xy}\,(n_1 \frac{\partial\,\phi}{\partial n} - n_2 \frac{\partial\,\phi}{\partial t}))\, n_2 \, ds \\
&= \oint_{\partial\Omega} \Delta u \frac{\partial\,\phi}{\partial n} + \left(u_{xy}\, n_1\, n_2 - u_{yy}\, n_1^2 - u_{xx}\, n_2^2 + u_{xy}\, n_1\, n_2\right) \frac{\partial\,\phi}{\partial n} + \\
&\qquad + \left(u_{xy}\, n_1^2 + u_{yy}\, n_1\, n_2 - u_{xx}\, n_1\, n_2 - u_{xy}\, n_2^2\right) \frac{\partial\,\phi}{\partial t} \, ds \\
&= \oint_{\partial\Omega} \Delta u \frac{\partial\,\phi}{\partial n} - \left(n_1^2\, u_{yy} + n_2^2\, u_{xx} - 2\, n_1\, n_2\, u_{xy}\right) \frac{\partial\,\phi}{\partial n} + \\
&\qquad + \left((n_1^2 - n_2^2)\, u_{xy} + n_1\, n_2\, (u_{yy} - u_{xx})\right) \frac{\partial\,\phi}{\partial t} \, ds
\end{aligned}
$$

The second contribution can be integrated by parts over the closed curve $\partial\Omega$, leading to

$$\oint_{\partial\Omega} \left((n_1^2 - n_2^2)\, u_{xy} + n_1\, n_2\, (u_{yy} - u_{xx})\right)\, \frac{\partial\, \phi}{\partial t}\, ds = -\oint_{\partial\Omega} \frac{\partial}{\partial\, t}\, \left((n_1^2 - n_2^2)\, u_{xy} + n_1\, n_2\, (u_{yy} - u_{xx})\right)\, \phi\, ds\,.$$

For the second boundary contribution use

$$\oint_{\partial\Omega} \langle \begin{pmatrix} u_{xxx} + u_{xyy} \\ u_{xxy} + u_{yyy} \end{pmatrix}, \vec{n}\rangle\, \phi\, ds = \oint_{\partial\Omega} \frac{\partial\, (u_{xx} + u_{yy})}{\partial n}\, \phi\, ds\,.$$

The sum of all boundary integrals has to vanish for all ϕ, i.e.

$$\begin{aligned} 0 \;\; = \;\; & \oint_{\partial\Omega} \left(\lambda_1\, \frac{\partial\, u}{\partial n} - \lambda_2\, \frac{\partial\, \Delta u}{\partial n} - \lambda_2\, \frac{\partial}{\partial\, t}\, \left((n_1^2 - n_2^2)\, u_{xy} + n_1\, n_2\, (u_{yy} - u_{xx})\right)\right)\, \phi\, ds + \\ & + \lambda_2 \oint_{\partial\Omega} \left(\Delta u - \left(n_1^2\, u_{yy} + n_2^2\, u_{xx} - 2\, n_1\, n_2\, u_{xy}\right)\right)\, \frac{\partial\, \phi}{\partial n}\, ds\,. \end{aligned}$$

As consequence find the two natural boundary conditions

$$\lambda_2 \left(\frac{\partial\, \Delta u}{\partial n} + \frac{\partial}{\partial\, t}\, \left((n_1^2 - n_2^2)\, u_{xy} + n_1\, n_2\, (u_{yy} - u_{xx})\right)\right) \;=\; \lambda_1\, \frac{\partial\, u}{\partial n} \tag{6}$$

$$\Delta u - \left(n_1^2\, u_{yy} + n_2^2\, u_{xx} - 2\, n_1\, n_2\, u_{xy}\right) \;=\; 0\,. \tag{7}$$

4–4 Remark : Second order derivatives are necessary
One might be attempted to use first order derivatives only in the regularizing functional, i.e.

$$F(u) = \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \iint_{\Omega} u_x^2 + u_y^2\, dA\,.$$

But this does not work at all, caused by $H^1(\Omega)$ not being imbedded in $C^0(\overline{\Omega})$. To illustrate this use radial functions depending on $r = \sqrt{x^2 + y^2}$ and examine for $p > 0$

$$v_\varepsilon(x, y) = \begin{cases} 1 - \frac{r^p}{\varepsilon^p} & \text{for} \quad r < \varepsilon \\ 0 & \text{for} \quad r \geq \varepsilon \end{cases}\,.$$

Then $v_\varepsilon(0, 0) = 1$ for all $\varepsilon > 0$ and $\lim_{\varepsilon \to 0+} v_\varepsilon(x, y) = 0$ for $(x, y) \neq (0, 0)$. Use $\nabla r = \frac{1}{r}\, (x, y)$ to calculate $\|\nabla v_\varepsilon\| = \frac{p}{\varepsilon^p}\, r^{p-1}$ for $r \leq \varepsilon$ and an integration in polar coordinates to conclude

$$\begin{aligned} \|v_\varepsilon\|^2_{L_2(\mathbb{R}^2)} \;\; & \leq \;\; \pi\, \varepsilon^2 \\ \|\nabla v_\varepsilon\|^2_{L_2(\mathbb{R}^2)} \;\; & = \;\; \frac{2\, \pi\, p^2}{\varepsilon^{2p}} \int_0^\varepsilon r^{2p-2}\, r\, dr = \frac{2\, \pi\, p^2}{\varepsilon^{2p}\, 2\, p}\, \varepsilon^{2p} = p\, \pi\,. \end{aligned}$$

For separated data points (x_i, y_i) the function

$$u_\varepsilon(x, y) := \sum_{i=1}^{M} z_i\, v_\varepsilon(x - x_i, y - y_i)$$

satisfies (for $\varepsilon > 0$ small enough) $u_\varepsilon(x_i, y_i) = z_i$ and $\lim_{\varepsilon \to 0+} u_\varepsilon(x, y) = 0$ for all $(x, y) \neq (x_i, y_i)$. All functions satisfy $u_\varepsilon \in H^1(\Omega)$ since $\iint\limits_{\Omega} \|\nabla u_\varepsilon\|^2\, dA = M\, p\, \pi$. Using $\varepsilon = p$ find

$$\lim_{\varepsilon \to 0+} F(u_\varepsilon) = \lim_{\varepsilon \to 0+} \lambda_1\, \langle u_\varepsilon, u_\varepsilon\rangle_1 = 0 \quad \text{and} \quad \lim_{\varepsilon \to 0+} \langle u_\varepsilon, u_\varepsilon\rangle = 0\,.$$

The above functions u_ε (with $p = \varepsilon$) converge to zero, but a similar construction can lead to convergence towards any constant h by using

$$u_\varepsilon(x,y) := h + \sum_{i=1}^{M} (z_i - h)\, v_\varepsilon(x - x_i, y - y_i)\,.$$

Thus a regularization with $\lambda_2 = 0$ is not well defined and consequently a numerical implementation can not be stable. This is illustrated by an example in Section 4.2.3 on page 20.

4–5 Remark : First order derivatives are not necessary, $\lambda_1 \geq 0$ is acceptable
One might be attempted to set $\lambda_1 = 0$ and use second order derivatives only in the regularizing functional, i.e.

$$F(u) = \sum_{i=1}^{M} (u(x_i,y_i) - z_i)^2 + \lambda_2 \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2\,u_{xy}^2 \; dA$$

- In most real world cases this will work just fine and the result in [Arca04] states just that. A proof is shown in Theorem 4–7 below. The essential step is a result by Necas in [Neca12] and is shown in Theorem 4–6 below.

- Use a slight modification of the proof and work with $\lambda := \max\{\lambda_1, \lambda_2\}$ to establish the a-priori estimate. In the proof of uniqueness use $\lambda_2 > 0$ to conclude $\langle v, v\rangle_2 = \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2\,u_{xy}^2 \; dA = 0$. This implies (see Necas?) that the function v has to be linear. If the data points are not colinear then $\langle \vec{v}, \vec{v}\rangle = 0$ implies $v = 0$, i.e. uniqueness.

- With the keyword smoothing D splines in [Arca04, Theorem 3.1, p.66] a proof is given, using $H^2(\Omega)$ ellipticity and the Lax–Milgram Theorem with the norm

$$\|u\|_{A,2,\Omega}^2 := \sum_{i=1}^{M} |u(x_i,y_i)|^2 + \langle u, u\rangle_2\,.$$

  Proposition 1.1 verifies that this is an equivalent norm to the usual Sobolev norm on $H^2(\Omega)$. The key argument is that $\langle u, u\rangle_2 = 0$ implies that u is a polynomial of degree 1 and [Neca12, Theorem 2.7.1] to show the equivalence of the norms. This replaces the a-priori estimate in Lemma 4–1 and is the key point of the proof. Below find the proof of the key result in [Neca12]. Observe that the proof is very elegant, but not constructive.

- The proof of the above Lemma with inequality (3) requires an estimate for $\|\nabla u\|_{L_2(\Omega)}$ and without additional assumptions on the data points (x_i, y_i) and z_i this is not available. Examine the situation when all points (x_i, y_i) are on a straight line. Then one can add a linear function with arbitrary slope without changing the value of the functional $F(u)$. Thus there is no unique solution and no bound on $\|\nabla u\|_{L_2(\Omega)}$.

- In [GreeSilv94, Lemma 7.1, p.149] a proof is given with domain $\mathbb{R}^2$ that for non colinear data a unique solution exists.

4–6 Theorem : *[Neca12, Theorem 2.7.1] Assuming that the data points (x_i, y_i) are not colinear there exists a constant c such that*

$$\|u\|^2_{H^1(\Omega)} \le c \left(\langle u, u\rangle_2 + \sum_{i=1}^{M} u^2(x_i, y_i) \right) \quad \text{for all} \quad u \in H^2(\Omega)\,. \tag{8}$$

This leads easily to

$$\|u\|^2_{H^2(\Omega)} \le c_1 \left(\langle u, u\rangle_2 + \sum_{i=1}^{M} u^2(x_i, y_i) \right) \quad \text{for all} \quad u \in H^2(\Omega)\,. \tag{9}$$

Proof : Proof by contradiction. If no such constant exists then there exists a sequence $u_n \in H^2(\Omega)$ with $\|u_n\|_{H^1(\Omega)} = 1$ and

$$\langle u_n, u_n\rangle_2 + \sum_{i=1}^{M} u_n^2(x_i, y_i) \le \frac{1}{n}\,.$$

This implies $\|u_n\|^2_{H^2(\Omega)} = \|u_n\|^2_{H^1(\Omega)} + \langle u_n, u_n\rangle_2 \le 1 + \frac{1}{n}$. Since the imbedding $H^2(\Omega)$ in $H^1(\Omega)$ is compact use a subsequence (again denoted by u_n), such that for all $|\alpha| = 2$

$$\begin{aligned} D^\alpha u_n &\to 0 \quad \text{in } L_2(\Omega) \\ u_n &\to u \quad \text{in } H^1(\Omega) \end{aligned}$$

Thus $u_n \to u$ in $H^2(\Omega)$ and $D^\alpha u = 0$. This implies (use [Neca12, Theorem 1.1.6, p.10]) that u is a polynomial of degree 1. Since the data points are not colinear $\sum_{i=1}^{M} u_n^2(x_i, y_i) \le \frac{1}{n}$ implies $u = 0$. This is in contradition to $\|u\|_{H^1(\Omega)} = 1$. □

For sake of completeness the statement and proof of the results with $\lambda_1 \ge 0$ is given, using a presentation very close to Theorem 4–2. Observe that this result is applicable to more situations, the domain $\Omega \subset \mathbb{R}^2$ is not required to be convex and $\lambda_1 = 0$ is permissible.

4–7 Theorem : *On a bounded, connected domain $\Omega \subset \mathbb{R}^2$ with lipschitzian boundary examine the functional*

$$\begin{aligned} F(u) &:= F_D(u) + \lambda_1 F_1(u) + \lambda_2 F_2(u) \\ &= \langle \vec{u} - \vec{z}, \vec{u} - \vec{z}\rangle + \lambda_1 \langle u, u\rangle_1 + \lambda_2 \langle u, u\rangle_2 \\ &= \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_1 \iint_\Omega u_x^2 + u_y^2 \, dA + \lambda_2 \iint_\Omega u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2 \, dA\,. \end{aligned}$$

The data points $(x_i, y_i) \in \Omega$ are not colinear, i.e. for $u \in H^2(\Omega)$ the equality $\sum_{i=1}^{M} u^2(x_i, y_i) + \langle u, u\rangle_2 = 0$ implies $u = 0$. For regularization parameters $\lambda_1 \ge 0$ and $\lambda_2 > 0$ the functional has exactely one minimizer $u_* \in H^2(\Omega)$.

Proof : Proceed in three steps: first establish an a-priori estimate, then verify existence, followed by the proof of uniqueness.

A-priori estimate:
Let $u_0(x, y) = c_0 + c_1\, x + c_2\, y$ be the unique solution of the least square problem $\min \sum_{i=1}^{M} (z_i - u_0(x_i, y_i))^2$. Then $F(u_0) < \infty$ implies that the set

$$\mathrm{A} := \{u \in H^2(\Omega) \,|\, F(u) \le F(u_0)\}$$

is not empty and the above Theorem implies that it is uniformly bounded in $H^2(\Omega)$.

Existence:
Use a minimizing sequence $u_n \in A$ of the functional $F(u)$. Since $H^2(\Omega)$ is compactly imbedded in $H^1(\Omega)$ there exists a convergent subsequence (again denoted by u_n) in $H^1(\Omega)$ and $u_n \longrightarrow u_*$. Since $H^2(\Omega)$ is a Hilbert space there exists a weakly convergent subsequence u_n and $u_* \in H^2(\Omega)$. The weak convergence implies

$$\langle \vec{u}_n - \vec{z}, \vec{u}_* - \vec{z} \rangle + \lambda_1 \langle u_n, u_* \rangle_1 + \lambda_2 \langle u_n, u_* \rangle_2 \overset{n\to\infty}{\longrightarrow} \langle \vec{u}_* - \vec{z}, \vec{u}_* - \vec{z} \rangle + \lambda_1 \langle u_*, u_* \rangle_1 + \lambda_2 \langle u_*, u_* \rangle_2 = F(u_*)$$

and $u_* \in H^2(\Omega)$ is a minimizer of the functional F.

Uniqueness:
Assume $u_1 \in H^2(\Omega)$ is a second minimizer. Let $v = u_1 - u_*$ and examine

$$\begin{aligned} g(t) &:= F(u_* + t\,v) \\ &= \langle \vec{u}_* + t\,\vec{v} - \vec{z}, \vec{u}_* + t\,\vec{v} - \vec{z} \rangle + \lambda_1 \langle u_* + t\,v, u_* + t\,v \rangle_1 + \lambda_2 \langle u_* + t\,v, u_* + t\,v \rangle_2 . \end{aligned}$$

Thus g is a polynomial of degree 2 with $g(0) = g(1) = F(u_*)$. Since u_* is a minimizer, conclude $\frac{d^2 g}{dt^2} = 0$, i.e. $0 = \langle \vec{v}, \vec{v} \rangle + \lambda_1 \langle v, v \rangle_1 + \lambda_2 \langle v, v \rangle_2$. Thus $\langle v, v \rangle_2 = 0$ and all second order derivatives vanish and the function v has to be linear, use Result 4–10. Then $\langle \vec{v}, \vec{v} \rangle = 0$ and the non-colinear data points implies that $v = 0$ and thus the minimizer u_* is unique. □

4–8 Remark : Find a solution of the problem with $\lambda_1 = 0$ in a document by David Eberly [Eber96], i.e. `ThinPlateSplines.pdf`, Geometric Tools, Redmond WA 98052,
https://www.geometrictools.com/ or more precise
https://www.geometrictools.com/Documentation/Documentation.html.

The expression to be minimized is

$$F(u) = \sum_{i=1}^{M} (u(x_i, y_i) - z_i)^2 + \lambda_2 \iint\limits_{\mathbb{R}^2} u_{xx}^2 + u_{yy}^2 + 2\,u_{xy}^2 \, dA \,,$$

i.e. integration over the plane $\mathbb{R}^2$ and not a bounded domain $\Omega \subset \mathbb{R}^2$. The main source seems to be [Wahb90], see also [Duch77],[GreeSilv94] (on my HD).

As Green's function use $G(r) = \frac{1}{6\,\pi}\, r^2 \ln(r)$, where $r = \sqrt{x^2 + y^2}$. Then the ansatz for the optimizer is

$$u(x, y) = c_0 + c_1\, x + c_2\, y + \sum_{i=1}^{M} \alpha_i\, G(\|(x - x_i, y - y_i)\|) \,.$$

This approach is very specific to the domain $\Omega = \mathbb{R}^2$.

4.4.2 Simple proof of $\nabla u = \vec{0}$ implies u is constant

Let $\Omega \subset \mathbb{R}^N$ be a bounded, open domain, consisting of a finite number of connected components. The results are to be verified on each of the components.

4–9 Result : *For $u \in W^{1,p}(\Omega)$ use $\int_\Omega \|\nabla u\|^p \, dA = 0$ to conclude $u = const$ on each of the components.*

Proof : This is a trivial consequence of the Poincaré–Wirtinger inequality (e.g. [Evan98, §5.8.1])

$$\|u - u_\Omega\|_{L_p(\Omega)} \le c\, \|\nabla u\|_{L_p(\Omega)}$$

where the average value u_Ω is given by $u_\Omega = \frac{1}{|\Omega|} \int_\Omega u \, dA$. □

4–10 Result : *For $u \in W^{2,p}(\Omega)$ use $\sum_{|\alpha|=2} \int_\Omega |D^\alpha u|^p \; dA = 0$ to conclude that u is a linear function on each connected component of Ω.*

Proof : Use the above result on the gradient ∇u. □

4.4.3 Numerical differentiation and integration on rectangles

Examine the partial derivatives of functions defined on rectangles $[a,b] \times [c,d] \subset \mathbb{R}^2$, using uniform grids. Use

$$\begin{aligned} x_i &= a + i\,\Delta x \quad \text{for } i = 0,1,2,3,\ldots,N_x \text{ with } \Delta x = \tfrac{b-a}{N_x} \\ y_j &= c + j\,\Delta y \quad \text{for } j = 0,1,2,3,\ldots,N_y \text{ with } \Delta y = \tfrac{d-c}{N_y} \end{aligned}$$

and the usual matrices to determine partial derivatives with finite difference approximations. With these the centered difference approximation is used determine the values of the derivatives at the mid points between the grid points, e.g. at $(x,y) = (x_i + \frac{\Delta x}{2}, y_j)$ for $\frac{\partial u}{\partial x}$ or at $(x,y) = (x_i, y_j + \frac{\Delta y}{2})$ for $\frac{\partial u}{\partial y}$.

$$\mathbf{D}_x = \frac{1}{\Delta x} \begin{bmatrix} -1 & 1 & & & & & \\ & -1 & 1 & & & & \\ & & -1 & 1 & & & \\ & & & \ddots & \ddots & & \\ & & & & -1 & 1 \end{bmatrix} \in \mathbb{M}^{N_x \times (N_x+1)}$$

$$\mathbf{D}_y = \frac{1}{\Delta y} \begin{bmatrix} -1 & 1 & & & & & \\ & -1 & 1 & & & & \\ & & -1 & 1 & & & \\ & & & \ddots & \ddots & & \\ & & & & -1 & 1 \end{bmatrix} \in \mathbb{M}^{N_y \times (N_y+1)}$$

With these matrices and a Kronecker product with an identity matrix the first partial derivatives can be estimated. To perform the integration use a midpoint rule in x direction and a trapezoidal rule in y direction.

$$\begin{aligned} \langle \frac{\partial u}{\partial x}, \frac{\partial u}{\partial x} \rangle &= \int_c^d \left(\int_a^b \left(\frac{\partial u(x,y)}{\partial x} \right)^2 dx \right) dy \approx \int_c^d \left(\sum_{i=0}^{N_x-1} \left(\frac{\partial u(x_i + \Delta x/2, y)}{\partial x} \right)^2 \Delta x \right) dy \\ &\approx \frac{\Delta y}{2} \left(\sum_{i=0}^{N_x-1} \left(\frac{\partial u(x_i + \Delta x/2, y_0)}{\partial x} \right)^2 \Delta x \right) + \\ &\quad + \Delta y \sum_{j=1}^{N_y-1} \left(\sum_{i=0}^{N_x-1} \left(\frac{\partial u(x_i + \Delta x/2, y_j)}{\partial x} \right)^2 \Delta x \right) + \\ &\quad + \frac{\Delta y}{2} \left(\sum_{i=0}^{N_x-1} \left(\frac{\partial u(x_i + \Delta x/2, y_{N_y})}{\partial x} \right)^2 \Delta x \right) \end{aligned}$$

For the seond order derivatives at the midpoints between the grid points use the matrix

$$\mathbf{D}_{xx} = \frac{1}{2\,(\Delta x)^2} \begin{bmatrix} +3 & -7 & +5 & -1 & & & & \\ +1 & -1 & -1 & +1 & & & & \\ & +1 & -1 & -1 & +1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & +1 & -1 & -1 & +1 & \\ & & & & +1 & -1 & -1 & +1 \\ & & & & -1 & +5 & -7 & +3 \end{bmatrix} \in \mathbb{M}^{N_x \times (N_x+1)}$$

and a similar matrix $\mathbf{D}_{yy}$ for $\frac{\partial^2 u}{\partial y^2}$. A Kronecker product is used to generate the matrix approximating the second order derivatives on the grid. These finite difference approximations are consistent of order 2. The integration over the rectangle is very similar to the first order derivatives. Similar algorithms are used to integrate $\frac{\partial^2 u}{\partial y^2}$ and $\frac{\partial^2 u}{\partial x\,\partial y}$, resp. their squares.

4.4.4 Bilinear interpolation on rectangular grids

To construct the interpolation operator the values at the four corners of a rectangle are used to interpolate at a point in the rectangle. For this bilinear functions are used.

If a function u is defined on the unit square $[0,1] \times [0,1] \subset \mathbb{R}^2$ the **bilinear interpolation** based on the four corners is given by

$$\begin{aligned} u(\xi,\eta) &= (1-\xi)\,(1-\eta)\,u_{i,j} + \xi\,(1-\eta)\,u_{i+1,j} + (1-\xi)\,\eta\,u_{i,j+1} + \xi\,\eta\,u_{i+1,j+1} \\ &= \left\langle \begin{pmatrix} (1-\xi)\,(1-\eta) \\ \xi\,(1-\eta) \\ (1-\xi)\,\eta \\ \xi\,\eta \end{pmatrix}, \begin{pmatrix} u_{i,j} \\ u_{i+1,j} \\ u_{i,j+1} \\ u_{i+1,j+1} \end{pmatrix} \right\rangle . \end{aligned}$$

This can be used to define the (sparse) interpolation matrix $\mathbf{Interp} \in \mathbb{M}^{(N_x \cdot N_y) \times M}$ to determine the interpolated values at the data points (x_i, y_i). It is best constructed with the command `sparse()` in *Octave*.

4.4.5 Compare rescaling and different weight factors

- Use external data x_i and u_i with the goal $u_i = f(x_i)$ and then replace x_i by $z_i = \mu\,x_i$. The new goal is $u_i = f(\frac{1}{\mu}\,\mu\,x_i) = f(\frac{1}{\mu}\,z_i)$. Thus the argument of the function f is rescaled by $\frac{1}{\mu}$.
- First order derivatives

$$\begin{aligned} g(x) &= f(\mu\,x) \\ 0 < x < \frac{1}{\mu} \qquad & \qquad 0 < z = \mu\,x < 1 \\ g'(x) &= \mu\,f'(\mu\,x) \\ F_1(f) &= \int_0^1 (f'(x))^2\,dx \\ G_1(g) &= \int_0^{1/\mu} (g'(x))^2\,dx \;=\; \int_0^{1/\mu} (\mu\,f'(\mu\,x))^2\,dx \\ z = \mu\,x \qquad & \qquad dz = \mu\,dx \\ &= \int_0^1 (\mu\,f'(z))^2\,\frac{1}{\mu}\,dz \;=\; \frac{1}{\mu}\,\mu^2\,F_1(f) \end{aligned}$$

Consequence: rescaling an independent variable by a factor μ is equivalent to a factor μ for the functional contribution of the first derivative. Observe that there is a contribution of $\frac{1}{\mu}$ by the integration.

- Second order derivatives

$$\begin{aligned} g(x) &= f(\mu\, x) \\ 0 < x < \frac{1}{\mu} &\qquad 0 < z = \mu\, x < 1 \\ g''(x) &= \mu^2\, f''(\mu\, x) \\ F_2(f) &= \int_0^1 (f''(x))^2\, dx \\ G_2(g) &= \int_0^{1/\mu} (g''(x))^2\, dx \;=\; \int_0^{1/\mu} (\mu^2\, f''(\mu\, x))^2\, dx \\ z = \mu\, x &\qquad dz = \mu\, dx \\ &= \int_0^1 (\mu^2\, f''(z))^2\, \frac{1}{\mu}\, dz \;=\; \frac{1}{\mu}\, \mu^4\, F_2(f) \end{aligned}$$

Consequence: rescaling an independent variable by a factor μ is equivalent to a factor μ^3 for the functional contribution of the second derivative. Observe that there is a contribution of $\frac{1}{\mu}$ by the integration.

- With the rescaling $(x, y) \to (\mu\, x, y)$, $u(x, y) \to u(\mu\, x, y)$, $\Omega \to \Omega'$ and $dA \to \frac{1}{\mu}\, dA'$ observe the effects on the functional F_2.

$$\lambda_2\, F_2(u) = \lambda_2 \iint\limits_{\Omega} u_{xx}^2 + u_{yy}^2 + 2\, u_{xy}^2\, dA \to \frac{\lambda_2}{\mu} \iint\limits_{\Omega'} \left(\mu^4 u_{xx}^2 + u_{yy}^2 + 2\, \mu^2 u_{xy}^2\right)\, dA'$$

- Using the above obervations a change of scale for an independent variable can be replaced by modifying the functional F_2 by

$$\lambda_2\, F_2(u) \quad \longrightarrow \quad F_{2,\vec{\lambda}}(u) = \iint\limits_{\Omega} \lambda_{xx} u_{xx}^2 + \lambda_{yy} u_{yy}^2 + 2\, \lambda_{xy} u_{xy}^2\, dA$$

  - The relative size of the three contributions is not changed by the rescaling of the domain of integration.
  - The effect of rescaling the independent variable x by a factor of μ can be achieved by $\lambda_{xx} = \mu^4\, \lambda_{yy}$.
  - The effect of rescaling the independent variable y by a factor of μ can be achieved by $\lambda_{yy} = \mu^4\, \lambda_{xx}$.
  - Based on the above observation I suggest to then use $\lambda_{xy} = \sqrt{\lambda_{xx}\, \lambda_{yy}}$. This is implemented in the code `regularization2Dslopes.m` . You may overwrite this default behavior and specify all three of λ_{xx}, λ_{yy} and λ_{xy}.

List of Figures

Bibliography

[Adam03] R. Adams and J. Fournier. *Sobolev Spaces*. ISSN. Elsevier Science, 2003.

[Arca04] R. Arcangéli, M. C. L. de Silanes, and J. J. Torrens. *Mulitimensional Minimizing Splines, Theory and Applications*. Kluwer Academic Publishers, 2004.

[Duch77] J. Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Constructive Theory of Functions of Several Variables*, volume 571 of *Lecture Notes in Mathematics*, pages 85–100. Math. Res. Inst., Oberwolfach, Springer, Berlin, 1977.

[Eber96] D. Eberly. Thin–Plate Splines. Technical report, Geometric Tools, Redmond WA 98052, 1996. www.geometrictools.com/Documentation/ThinPlateSplines.pdf.

[Evan98] L. C. Evans. *Partial Differential Equations*. AMS, 1998.

[GreeSilv94] P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: A roughness penalty approach*. Chapman and Hall, United Kingdom, 1994.

[LandLifs75] L. D. Landau and E. M. Lifshitz. *Lehrbuch der Theoretischen Physik, Band VII, Elastizitätstheorie*. Akademie Verlag, Berlin, 1975.

[Neca12] J. Necas, C. Simader, Š. Necasová, G. Tronel, and A. Kufner. *Direct Methods in the Theory of Elliptic Equations*. Springer Monographs in Mathematics. Springer Berlin Heidelberg, 2012.

[Wahb90] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia, 1990.