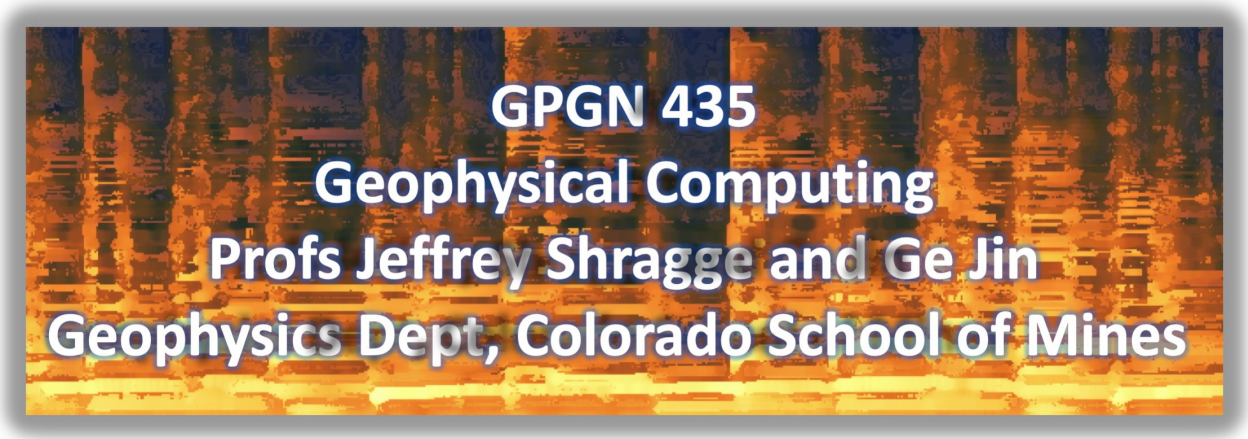


Module 1: Introduction to Geophysical Computing

For the modern professional geophysicists, the use of computers and algorithms is ubiquitous in all aspects of the daily routine. Whether using a "canned" commercial geophysical software application to or developing new code to address a particular task, employing/implementing some sort of computational algorithm is almost always at the heart of any geophysical task.

The challenge of canned software

One of the key challenges geophysicsts face when using commercial geophysical software is that it is often difficult to determine what is going on "under the hood". In many instances it may seem that geophysical results are found in the following way:

$$\boxed{\begin{array}{c}\text{Field}\\ \text{Geophysical}\\ \text{Data}\end{array}} \Longrightarrow \boxed{\begin{array}{c}\text{Magic Black}\\ \text{Box of Geophysical}\\ \text{Data Processing}\end{array}} \Longrightarrow \boxed{\begin{array}{c}\text{Top}-\text{notch}\\ \text{Geophysical}\\ \text{Result}\end{array}}$$

Often there are numerous parameters that go into the blackbox, various combinations of which can greatly affect the output results! It can be extremely frustrating for users - especially when the documentation and usage examples cannot be found, and there is no open-source code to help figure it out.

Python and Scientific Libraries

One of the major goals of this course is to strip away some of the **magic black box** nature of how students approach geophysical computing, and to provide a better conceptual idea of what is going on when software is applied to tackle geophysical problems. That said, it is not our goal to "reinvent the wheel" on absolutely every algorithm because this would take far too long and one would never finish finding solutions to the problem at hand.

The approach we are taking in this course tries to find a middle way. In particular, we will focus on the **Python** language and leverage its open-source, diverse and well validated computational toolkits including **Numpy** and **Scipy** as well as the strong plotting library **Matplotlib**.

However, to fully appreciate the theory behind some of these tools, it is important to code up some algorithms by oneself to help deepen the understanding of just what the algorithm is doing. Moreover, there will be scenarios where one has to develop a full computer program for a specialized task for which no community-based solution exists. In these cases, Python may not be the optimal language in which to develop and thus there will be the need to include algorithms directly embedded into the code. Having a deeper understanding of the geophysical algoritms will be greatly helpful in these scenarios. Overall, it is our hope that this will assist students in designing better geophysical computing solutions that will facilitate achiement of the geophysical data processing goals.

Course Aims

The course is intended to provide senior-level undergraduate and first-year graduate students with material that aimed at improving the algorithmic, programming and computing skills, while simultaneously enhancing skills in handling geophysical data sets. In some ways, these skills are those that are not really taught in any one particular geophysics course; however, they are often the "glue" that students need to help stick together disparate components learned in specific courses into a workable solution.

If we were to broadly classify the material presented in this course, it could be broken down into the following three skill sets:

- Handling Geophysical Data:
  - **Data Interpolation**
  - **Regression (Model Fitting)**
- Applied Numerical Algorithms
  - **Numerical Quadrature (Integration)**

    - **Applied Linear Algebra for Geophysical Problems**
    - **Solutions to Ordinary Differential Equations (ODEs)**
    - **Differentiation and Discretization**

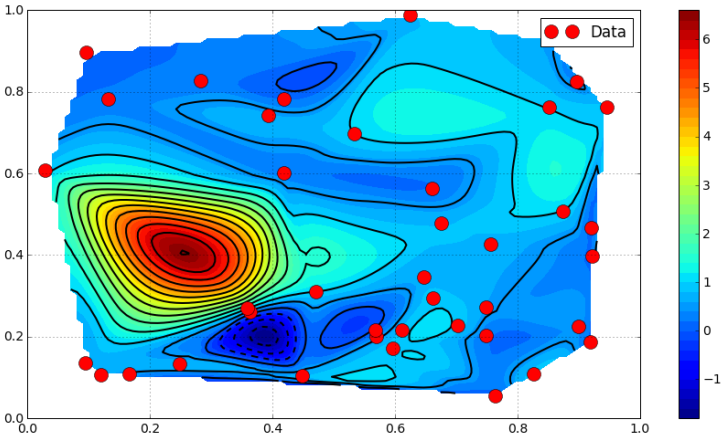
- Numerical Solutions of Geophysical Partial Differential Equations (PDEs):
    - **Elliptical PDEs (Laplace and Poisson Equations)**
    - **Parabolic PDEs (2D Heat Flow)**
    - **Hyperbolic PDEs (2D Acoustic wave equation)**

We briefly summarize each of these sections below.

Module 2: Interpolation

Geophysicists handle data that are acquired at regular and irregular (spatial and/or temporal intervals); however, most of the time the locations where data are acquired are insufficient in number or not in all of the desired locations required to make the corresponding geophysical or geological interpretation.

Let's look a the following example. For simplicitly, let's say that we have 30+ elevation data points that were acquired over a 1 km by 1 km area. An important question that will no doubt be asked is: *Based on this data, can we determine the elevation profile throughout the entire area?*



The good news is that yes we can create high-density maps using the acquired data points; however, the challenge is that there is no one unique map that we can create from the data that would be arguably correct. To do this, we we will explore some of the fundamentals of **interpolation** and see a number of different ways in which plausible maps can be created in 1D and 2D. Many of these algorithms even can be exported to higher dimensions for those cases where 3D, 4D and even 5D interpolation is required; however, these fall beyond the scope of this module.

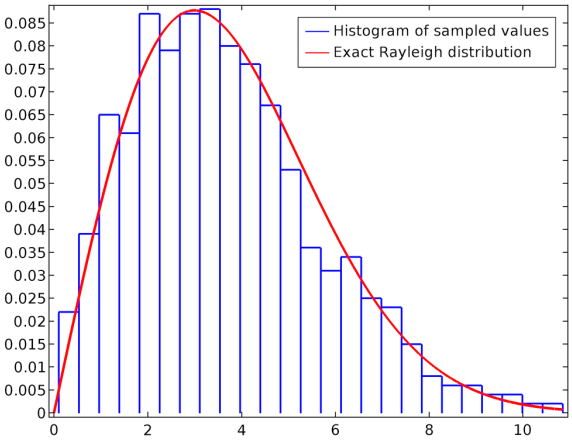
Module 3: Numerical Quadrature / Integration

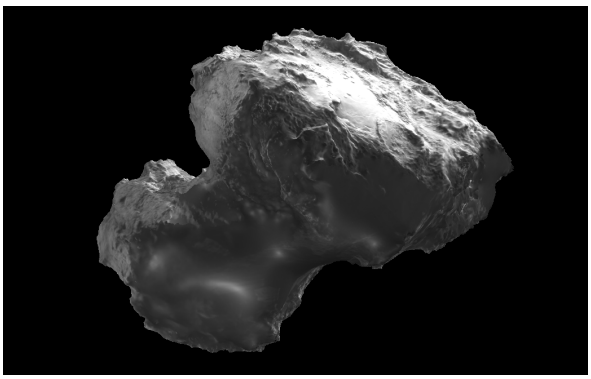
One of the fundamental building blocks of modern calculus is the **integral**. We have all no doubt sat though numerous classes where the instructors went over analytic evaluation of 1D, 2D and perhaps 3D integrals

$$J_1 = \int_a^b f(x)\,dx, \qquad J_2 = \int_c^d \int_a^b g(x,y)\,dx\,dy, \qquad J_3 = \int_e^f \int_c^d \int_a^b h(x,y,z)\,dx\,dy\,dz$$

using many different tricks. However, for these scenarios a key constraint is that the functions to be evaluated, here $f(x)$, $g(x,y)$ and $h(x,y,z)$ were always specified *analytically*. This represents a strong constraint on your ability to evaluate integrals!

Here's a couple of interesting examples:





The left-hand side figure represents a frequency histogram of values of some recorded data set. We see that some scientists have decided that this histogram can be represented by the model curve shown in red. Now, let's interpret these at probability density functions (PDFs) and say that we wanted to integrate this data and/or model curve over the interval $J = [0, a]$ to find out what the probabilty of something happening on the interval J. Arguably, it would easier to integrate the red curve if it were given by some analytic function. However, how accurate would this actually be given that the model doesn't exact match data!

The right-hand figure shows the an image of the Rosetta comet. Let's say that you were involved in a space mining operation that needed some key information regarding the comet. If the operation is run entirely by solar power, an important question would be how much of the 2D surface could be facing the sun and could be covered by solar panels? Another question would be if the comet contains 1% per volume of iron ore, how much iron ore is there available in the comet? Evidently, these questions would require evaluating 2D and 3D integrals over some irregular surface that is not easily described by analytic functions $g(x, y)$ and $h(x, y, z)$!

To address scenarios like this one, we will be exploring some of the **numerical quadrature** tools that are available in the python Scipy library.

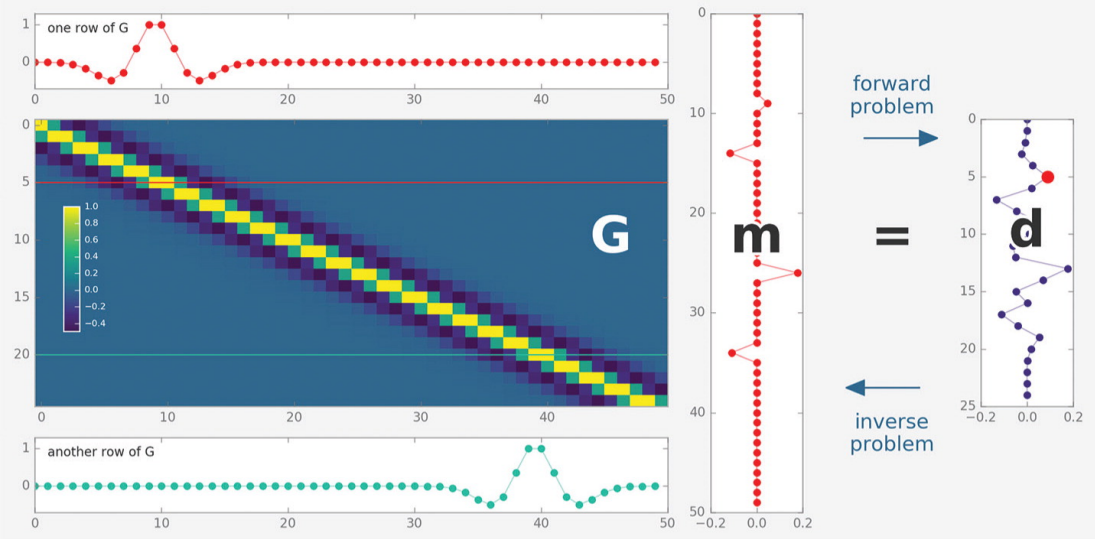
Module 4: Applied Linear Algebra for Geophysical Problems

In your academic career you have probably taken a semester-long course in Linear Algebra. It is equally likely that this course was pretty abstract and didn't really make a strong connection with geophysics. However, applied linear algebra comes up all of the time in geophysical computing - especially in the context of linearized system of equations.

In many situations we will investigate this semester, the geophysical problem can be represeted by a straightforward matrix equation:

$$\mathbf{Gm} = \mathbf{d}$$

where $\mathbf{m}$ is some geophysical **model parameter** (e.g., acoustic wave speed, thermal conductivity, electric charge), $\mathbf{d}$ is some sort of geophysical **data** (e.g., acoustic pressure, heat distibution, electric potential), and $\mathbf{G}$ is a numerical representation of the **physics** and often **experimental geometry** that is used to forward model $\mathbf{d}$ given $\mathbf{m}$. The following figure illustrates this concept using graphical depictions of $\mathbf{G}$, $\mathbf{m}$ and $\mathbf{d}$.



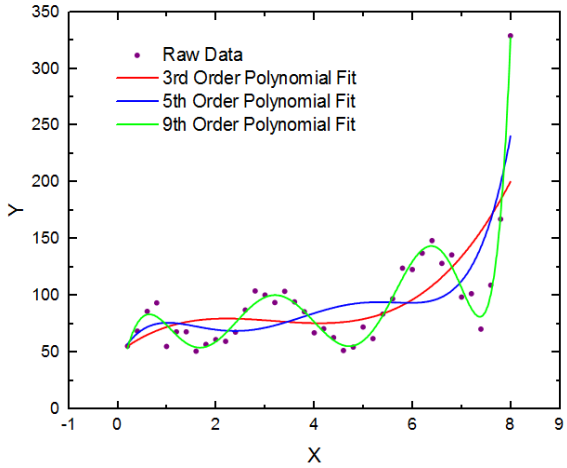
A good portion of this course looks at how one can solve these types of forward modeling equations in an efficient and straightforward manner using different methods of **applied linear algebra**. We will also examine scenarios where solutions cannot actually be generated due to various numerical challenges.

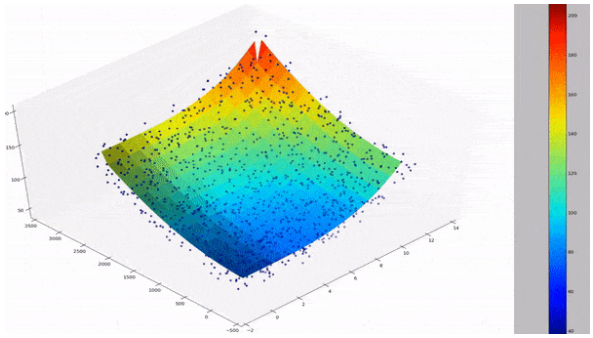
Module 5: Regression and Model Fitting

Another important skill set to develop is fitting models to a given data set. It's likely that we've all done this with using the linear regression tool in Excel that allows us to find the parameters m and d such that the linear relation $y = mx + d$ is optimal in the least-squares sense. You may have played around with higher order polynomials up to and including the same order of fit as the number of data points!

While it may be numerically possible to perfectly fit an underlying model (except perhaps when interpolating!), this is seldom the optimal thing to do. One important reason for this is that data $f(x)$ almost *noisy* to some degree such that it is better represented by $y = f(x + \Delta x_N) + \Delta f$ where Δx_N is the noise in the position measurement and Δf_N is the noise in the measurement (e.g., instrument fluctuaions).

Thus, we need to develop tools and experience in fitting data sets like those in the figure below.





The left-hand image shows a number of data points (dots) along with three different curves that are fit to higher orders. The right-hand image shows data acquired over an 2D area with significant scatter along with a 2D curve fit to it. Of course we may be interested in fitting models to much higher order data in order to be able to generate some more complex model that we could possibly hope to visualize in 1D or 2D images. This type of analysis is proving to be fundamental in **data science** and **machine learning** applications in the geosciences.

Module 6: Ordinary Differential Equations (ODEs)

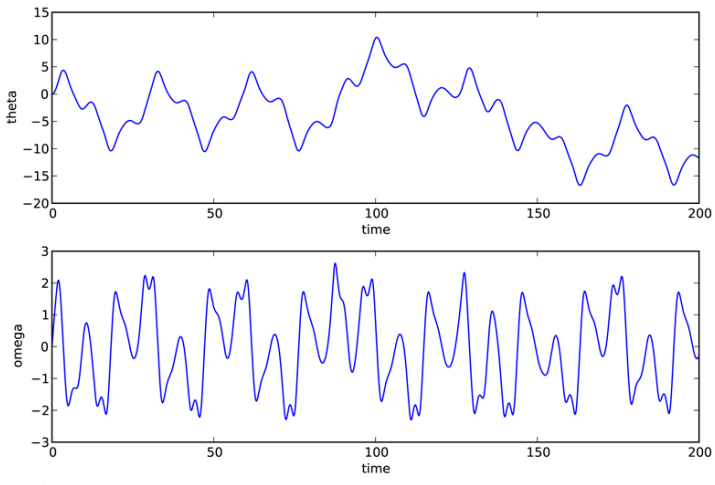
The next module in the course starts to move in a slightly different direction: we being examining numerical approaches to solving differential equations. We first start with ordinary differential equations (ODEs), which again should be one of the math courses you have taken during your first two years. Your "diffy-e-q" course probably largely focused on increasingly complex analytic solutions to increasingly more involved ODEs. You probably memorized the formulae and the solutions approaches for your exams, and then promptly forgot these solution approaches after walking out of the exam hall. (Don't worry, your instructors did this too ...)

However, as you no doubt expect, very powerful numerical solvers for ODEs that can be used in place of going back to your Differential Equations textbook. Importamtly, in many situations these routines can solve ODEs that cannot be solved by analytic methods! Take for example the problem of a driven damped pendulum. The equation of motion for the angle θ that the pendulum makes with the vertical is given by

$$\frac{d^2\theta}{dt^2} = -\frac{1}{Q}\frac{d\theta}{dt} + \sin\theta + d\cos\Omega t$$

where t is time, Q is the quality factor, d is the forcing amplitude, and Ω is the driving frequency of the forcing. The ODE is nonlinear owing to the $\sin\theta$ term and there one needs numerical approaches to solve this system!

The following figure shows the solution computed using Scipy ODE solvers in only a few lines of code. The lower panel shows a complex driving frequency Ω, while upper panel shows the computed solution of the pendulum.



ODE solvers can be quite useful in geophysical and geoscience problems - especially in scenarios where you have a whole number of coupled ODEs!

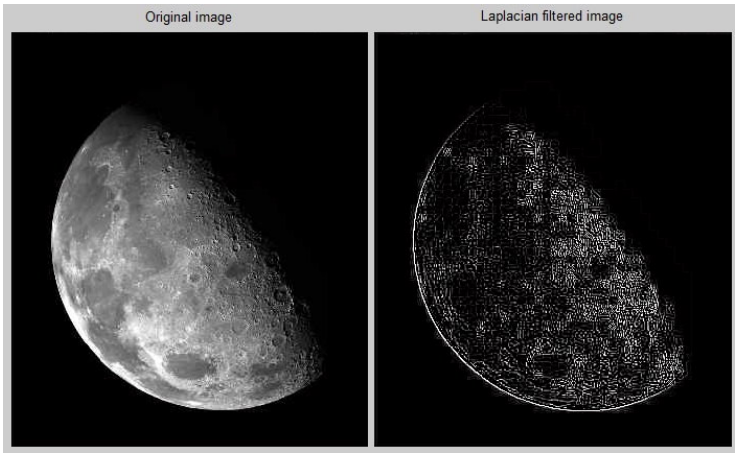
Module 7: Numerical Differentiation / Discretization

The next module continues our exploration of numerical methods to used to solve equations - especially partial differential equations (PDEs) such as the acoustic wave equation

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}$$

To solve these equations numerically in a computer, we need to solve PDEs on a discrete solution grid that necessarily requires the process of **discretizing the physical system** including the partial differential operators (e.g., $\frac{\partial^2}{\partial x^2}$) comprising the PDE. This is commonly done using **finite-difference** approximations, which commonly lead to numerical **stencils** and very efficient solution algorithms.

In fact, for those of you who have taken digital signal processing, you hopefully will already be familiar with the concept of discretization. To remind you, the left panel below shows an image of the moon. The center panel shows this picture after applying a **convolutional Laplacian filter** for the purpose of emphasizing the discontinuous structure. The right panel shows the **2D finite-difference approximation** of the continuous Laplacian operators ∇^2.



0	−1	0
−1	4	−1
0	−1	0

In this module we will be taking a deeper look at the numerical approximations and discretization with the goal of developing the skills required to solve PDEs of interest like those in Modules 8, 9 and 10.

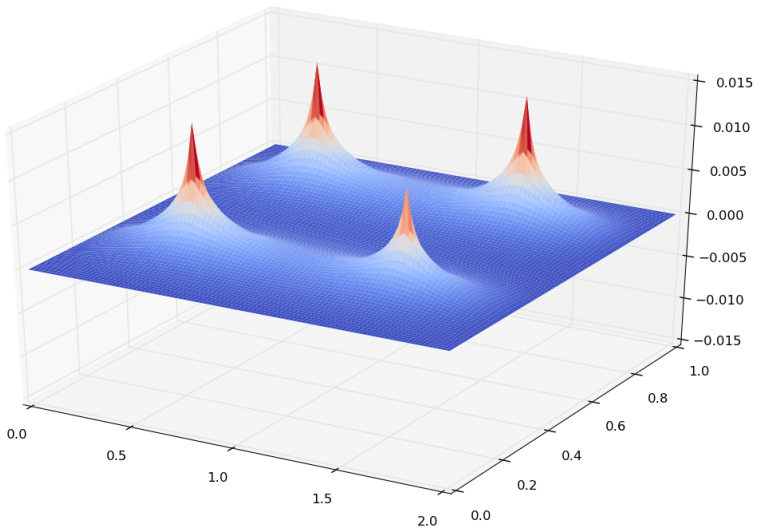
Module 8: Elliptical PDEs (Solutions to Laplace and Poisson Equations)

Having tackled issues pertaining to discretization of partial differential operators and thereby PDEs, we are now ready to move on to actually solving some PDEs of interest in Geophysics. One of the most straightforward 2D PDEs to solve is Poisson's equation for a potential surface $U(x, y)$:

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = f(x, y)$$

which reduced to Laplace's equation in the instance where $f(x, y) = 0$ throughout the solution domain. You may recall that one must specify the boundary conditions prior to calcuating solutions to thes types of equations.

the example below shows the potential surface solution related to the distribution of either four point masses (gravitational potential) or the response to four positive point charges (electrical potential).



In this course we will be exploring solutions to such equations using some of the approaches we investigated in the applied linear algebra section (Module 4).

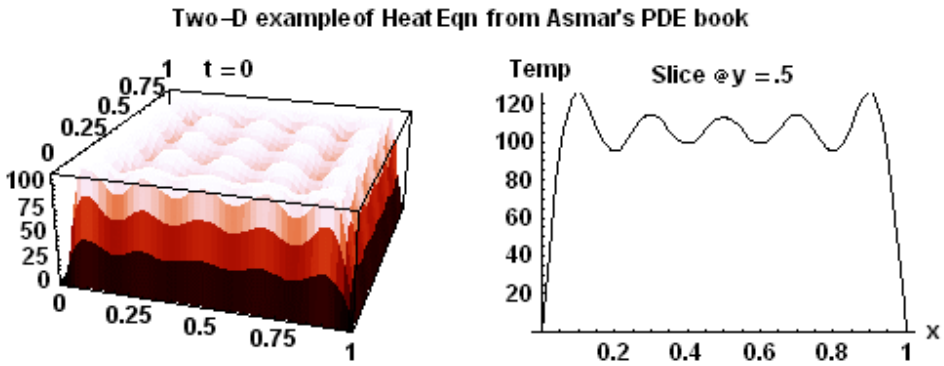
Module 9: Parabolic PDEs (Solutions to Heat Flow / Diffusion Equations)

The next set of PDEs we will study fall into the family of parabolic equations. In particular, we are looking to generate numerical solutions of 2D heat flow and 2D diffusion equations, both of which can be written in the following form:

$$\frac{\partial \phi(x,y,t)}{\partial t} = \nabla \cdot \left[D(x,y) \, \nabla \phi(x,y,t) \right]$$

where $\phi(x,y,t)$ is, e.g., the distribution of heat through the 2D solution domain, and $D(x,y)$ is the heterogeneous thermal conductivity field. Again, computing solutions requires setting both the initial condition (i.e., at time $t = 0s$) and the boundary conditions at the edges of the computational domain.

The left panels shows example below shows the time evoluation of the heat distribution $U(x,y,t)$ for a square computational domain where the four edges are held at 0°C and all interior points initially start at 100°C. The right panel shows a cross-section through the solution and is easier to see how the $U(x,y,t)$ evolves over the computed solution time.



In this module we will be developing numerical solutions to the 2D heat flow equation, validating them against analytical solutions to a PDE system, and then using your validated code to solve some interesting geophysical problems for which there are no analytical solutions.

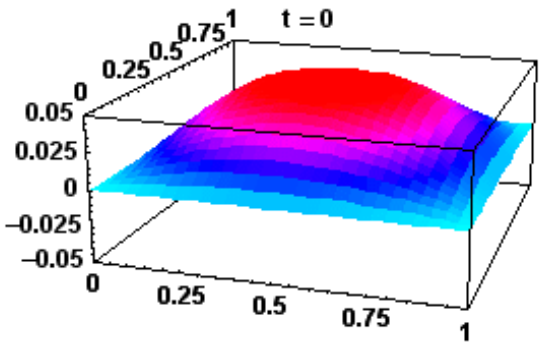
Module 10: Hyperbolic PDEs (Solutions to Acoustic Wave Equation)

The final set of PDEs we will study fall into the family of hyperbolic equations. In particular, we are looking to generate numerical solutions of 2D acoustic wave equation, which is written as the following:

$$\left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial x^2} - \frac{1}{c^2(x,y)} \frac{\partial^2}{\partial t^2} \right] U(x,y,t) = f(x,y,t)$$

where $U(x,y,t)$ is the displacement of the acoustic wave disturbance, and $c(x,y)$ is the heterogeneous acoustic wave speed, and $f(x,y,t)$ is the force source distribution (e.g., what is causing the acoustic wave disturbance). Computing solutions requires setting the initial conditions and the boundary conditions at the edges of the computational domain.

The example below shows a numerical solution to the 2D acoustic wave equation that is modeling the temporal and spatial evolution of a taut square drum head (i.e., clamped boundaries with zero displacement) and some initial amplitude and/or velocity distribution.



In this module we will be developing numerical solutions to the 2D acoustic, validating them against analytical solutions to a PDE system like in the example above, and then using our validated code to solve some interesting geophysical wave propagation problems for which there are no analytical solution.