Notes for MCTP mini course on numerical analysis
Summer 2012 (draft)

Daniel Appelö

August 15, 2012

Abstract

Contents

1 Introduction

The aim of this course is to give practical and theoretical knowledge of numerical approximations and how such approximations can be used when solving non-linear equations, ordinary or partial differential equations and when computing integrals. The course will consist of a set of laboratory assignments and lectures centered around various problems (academic versions) of importance in engineering and applied sciences.

As the course is very short we will focus on one or two numerical methods for each topic, but give references for in depth studies. In addition we will be somewhat informal when proving that a certain method has this or that property.

2 Ordinary differential equations, interpolation and integration and the search for a minimum energy path.

Our first application will be the problem of finding a *minimum energy path* (MEP) between minima in a potential energy landscape. The minimum energy path is defined as the path from one minima to another minima that attains the smallest maximal value of the potential energy along the path.

Knowledge of the MEP is important in many applications arising in chemistry, physics and material science as the MEP gives information if a system will transition from a stable state to another. In the first part of this course we will learn about the necessary numerical tools to implement a method (the string method [5, 3]) that can be used to find such MEPs.

The string method (see Algorithm 1) has two basic ingredients, an evolution step that relaxes the system into equilibria, and a reparametrization step that connects states or images along the MEP. We will talk more about the latter step later but for now we will focus on how to numerically evolve a system whose motion depends on a potential energy.

2.1 An example of a system with a potential energy landscape, a particle in a force field

Consider the two dimensional motion of a single particle of mass m (which we will take to be 1) at a point $(x(t), y(t))^T = \mathbf{x}(t)$ influenced by a force field $\mathbf{F}(\mathbf{x})$. The energy $\mathcal{E}$ of the particle is a sum of its kinetic and potential energy, that is,

$$\mathcal{E} = \mathcal{E}_{\mathrm{k}} + \mathcal{E}_{\mathrm{p}}.$$

Here the kinetic energy is given by the expression

$$\mathcal{E}_{\mathrm{k}} = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2), \tag{1}$$

where we use the notation $\dot{x} = \frac{dx(t)}{dt}$ for the velocity of the particle in the x-direction and $\dot{y} = \frac{dy(t)}{dt}$ in the y-direction.

The motion of the particle is governed by the equations of motion

$$m\ddot{x} = F_x, \qquad m\ddot{y} = F_y, \tag{2}$$

with initial conditions

$$x(0) = x^0, \qquad y(0) = y^0, \tag{3}$$

$$\dot{x}(0) = v_x^0, \qquad \dot{y}(0) = v_y^0, \tag{4}$$

If the force field is conservative the energy will remain constant for all times, that is, the time derivative of $\mathcal{E}$ is zero. If we take the time derivative of the energy we find that

$$0 = \frac{d}{dt}\mathcal{E}(x(t), y(t)) = m(\dot{x}\ddot{x} + \dot{y}\ddot{y}) + \frac{d}{dt}\mathcal{E}_{\mathrm{p}}(x(t), y(t)) = \dot{x}\left(m\ddot{x} + \frac{\partial \mathcal{E}_{\mathrm{p}}}{\partial x}\right) + \dot{y}\left(m\ddot{y} + \frac{\partial \mathcal{E}_{\mathrm{p}}}{\partial y}\right). \tag{5}$$

Thus, we find that the field is conservative if $F_x = -\frac{\partial \mathcal{E}_\mathrm{p}}{\partial x}$ and $F_y = -\frac{\partial \mathcal{E}_\mathrm{p}}{\partial y}$ and that the motion of the particle is described by the solution of the system of ordinary equations

$$m\ddot{x} = -\frac{\partial \mathcal{E}_\mathrm{p}}{\partial x}, \qquad m\ddot{y} = -\frac{\partial \mathcal{E}_\mathrm{p}}{\partial y}. \tag{6}$$

As most numerical methods for solving ordinary differential equations are designed for first order systems of ODEs, we rewrite the above system as the equivalent first order system:

$$\begin{aligned} m\dot{v}_x &= -\frac{\partial \mathcal{E}_\mathrm{p}}{\partial x}, \\ m\dot{v}_y &= -\frac{\partial \mathcal{E}_\mathrm{p}}{\partial y}, \\ \dot{x} &= v_x, \\ \dot{y} &= v_y. \end{aligned} \tag{7}$$

Rewriting a higher order ODE as an equivalent first order system is always possible (is the converse statement true?) and follows a standard recipe. For example to rewrite a nth order ODE $\frac{d^n y(t)}{dt^n} = F$, n new variables, $y_k = \frac{d^k y(t)}{dt^k}$, $k = 0, \ldots, n-1$, satisfying

$$\frac{dy_k(t)}{dt} = y_{k+1}, \quad k = 0, \ldots, n-2, \tag{8}$$

$$\frac{dy_{n-1}(t)}{dt} = F, \tag{9}$$

are introduced. For further details see p. 5 in [1].

2.2 Numerical solution of (7)

Suppose we know the potential energy and want to find the solution to (7). Only in very rare cases we can find the solution by pen and paper and for the vast majority of potentials we will have to use numerical techniques to find an approximate solution to (7).

In fact, in many real life applications even the potential energy itself will not be in closed form but rather have to be computed. This of course makes it impossible to compute the spatial derivatives in (7) by pen and paper. Fortunately we can can use numerical techniques to find approximate solutions to (7) by first approximating the time derivatives of x, y and the space derivatives of $\mathcal{E}_\mathrm{p}$.

2.2.1 Taylor expansion and finite difference approximations

The perhaps most useful tool from calculus when trying to design numerical methods is Taylor's formula

$$f(z) = f(a) + \frac{x-a}{1!}f'(a) + \frac{(x-1)^2}{2!}f^{(2)}(a) + \ldots + \frac{(x-1)^{n-1}}{(n-1)!} + R_n,$$

where the reminder is

$$\frac{(x-1)^n}{n!}f^{(n)}(\xi), \quad \xi \in [a, x].$$

Alternatively a more useful formulation of Taylor expansion can be found by instead expanding $f(z)$ around $z + h$. Then we find that

$$f(z+h) = f(z) + hf'(z) + \frac{h^2}{2!}f''(z) + \mathcal{O}(h^3). \tag{10}$$

If we now subtract off $f(z)$ and divide through with h on both sides we obtain

$$\frac{f(z+h) - f(z)}{h} = f'(z) + \frac{h}{2!}f''(z) + \mathcal{O}(h^2). \tag{11}$$

Thus $\frac{f(x+h)-f(x)}{h}$ is an approximation of $f'(x)$ with and error that is proportional to h.

The error in the approximation depends on the size of h and can be defined as

$$e(h) = |f'(z) - \frac{f(z+h)-f(z)}{h}| \approx \mathcal{O}(h),$$

that is, if we take h half as big we expect the error to be half as big as well.

In general an error in a numerical approximation is on the form $e(h) \approx \mathcal{O}(h^p)$. The *order of accuracy* of the approximation is then said to be p, where p often is an integer. The approximation above is first order accurate.

If you have derived and implemented a numerical approximation for computing some quantity and you expect the error to behave as $e(h) \approx \mathcal{O}(h^p)$ it is useful to try your program for a set of successively smaller values of h and plot the error as a function of h in a `loglog` plot. For example if you have stored the different values of h and the error in `H` and `E` you can plot them using the command `loglog(H,E)`. Now the slope of the plotted curve is the order of accuracy (recall $\log h^p = p \log h$) and if it matches up with your expectations you might have a bug free program.

Note that the higher the order of the approximation the faster the error is reduced if h is reduced. For example using a half as big h gives one quarter or one sixteenth the error when a second or fourth order method is used.

Classroom exercises

1. Combine the expansion for $f(z+h)$ with $f(z-h) = f(z) - hf'(z) + \frac{h^2}{2!}f''(z) + \mathcal{O}(h^3)$ to find an approximation to $f'(z)$. How does the size of the leading order error term for the two approximations if change if $h = 1/2, 1/4, 1/8$ are used? What is the order of approximation?
2. Use your two approximations to find the derivative of the functions
   (a) $f(x) = \tanh x$,
   (b) $f(x) = e^{\sin(\sqrt{\log(x)})}$,
   at $x = 2$ for some different values of h.
3. Use the more accurate approximation to estimate the less accurate error, does it decay as expected?
4. With pen and paper sketch a Matlab function that computes the derivative of a function `myfun(x)` using one of the finite difference approximations.

2.3 Dynamics and steady state equilibria

To make things concrete we consider the potential energy

$$\begin{aligned}\mathcal{E}_{\mathrm{p}} &= \left(1 - e^{-\left(\frac{x^2+y^2}{2^2}\right)^{10}}\right) \\ &\quad -e^{-((x-\frac{1}{2})^2+(y-\frac{1}{2})^2)^2} - e^{-((x+\frac{1}{2})^2+(y+\frac{1}{2})^2)^2} + e^{-((x+\frac{1}{4})^2+(y-\frac{1}{3})^2)^2} + e^{-((x-\frac{1}{4})^2+(y+\frac{1}{5})^2)^2},\end{aligned} \tag{12}$$

which is also displayed in Figure 1.

2.3.1 Lossless system

If we release a particle in the potential energy landscape it will move according to the system (7) with it's energy being conserved throughout the process. Therefore if we start at a point where the potential energy is high, the particle will start to move and its kinetic energy will increase as it gets closer to an energy minima and then decrease as it leaves the minima towards states that have a higher potential energy. As the system is lossless this process of conversion between the kinetic and potential energy will continue indefinitely and the particle will only stay in the potential well (minima) if it starts there. An example of a particle trajectory can be found in Figure 2.

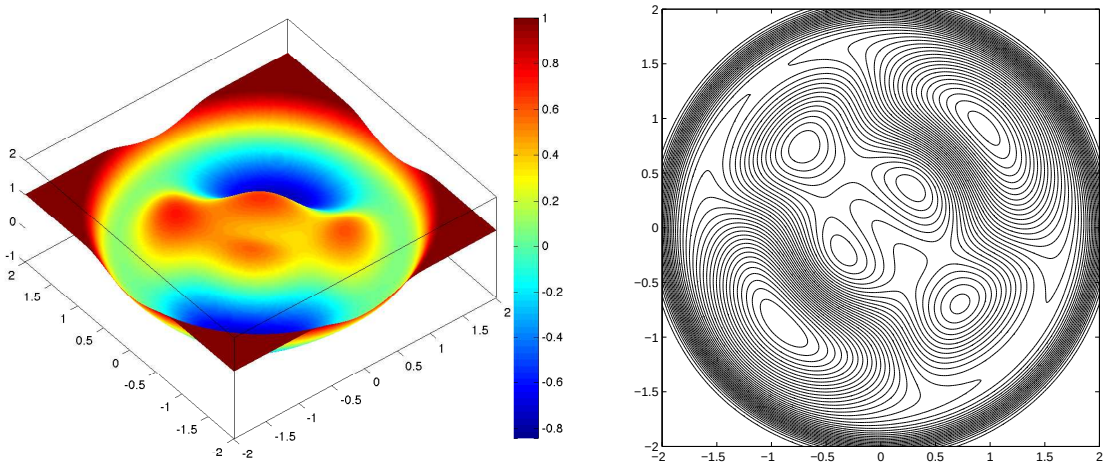

Figure 1: A potential energy landscape.

2.3.2 Lossy (damped) system

If the system has a little bit of damping from, say, friction the equations may look something like

$$\begin{aligned} m\dot{v}_x &= -\beta v_x - \frac{\partial \mathcal{E}_\mathrm{p}}{\partial x}, \\ m\dot{v}_y &= -\beta v_y - \frac{\partial \mathcal{E}_\mathrm{p}}{\partial y}, \\ \dot{x} &= v_x, \\ \dot{y} &= v_y, \end{aligned} \tag{13}$$

where β controls the about of damping in the system. To the right in Figure 2 and to the left in Figure 3 we have plotted the particle trajectories for a small and a large β.

Now the particle falls into the well and stays there. During this process the time derivatives are tending to zero and when they vanish we say that the system reaches an *equilibrium state* or steady state.

With damping included in the model we find that the time derivative of the energy is:

$$\frac{d}{dt}\mathcal{E}(x(t),y(t)) = m(\dot{x}\ddot{x}+\dot{y}\ddot{y}) + \frac{d}{dt}\mathcal{E}_\mathrm{p}(x(t),y(t)) = \dot{x}\left(m\ddot{x}+\frac{\partial \mathcal{E}_\mathrm{p}}{\partial x}\right) + \dot{y}\left(m\ddot{y}+\frac{\partial \mathcal{E}_\mathrm{p}}{\partial y}\right) = -\beta(\dot{x}^2+\dot{y}^2). \tag{14}$$

That is, the energy decreases as long as the particle is moving ($\dot{x} \neq 0, \dot{y} \neq 0$.) In other words as long as the particle has some potential energy that it can convert to kinetic energy the particle will move, the damping will gradually slow it down until it is at a position where the potential energy has a minimum and the particle is at rest.

This can also be seen directly from the equations of motions (13) that, with zero velocity, reduces to

$$\frac{\partial \mathcal{E}_\mathrm{p}}{\partial x} = 0, \qquad \frac{\partial \mathcal{E}_\mathrm{p}}{\partial y} = 0,$$

i.e. $\mathcal{E}_\mathrm{p}$ has a minimum (strictly speaking an inflection point.) We call this state an *equilibrium state*.

2.3.3 Classroom exercises

1. For the potential

$$\mathcal{E}_\mathrm{p} = (x-1)^2(x+1)^2 + y^4,$$

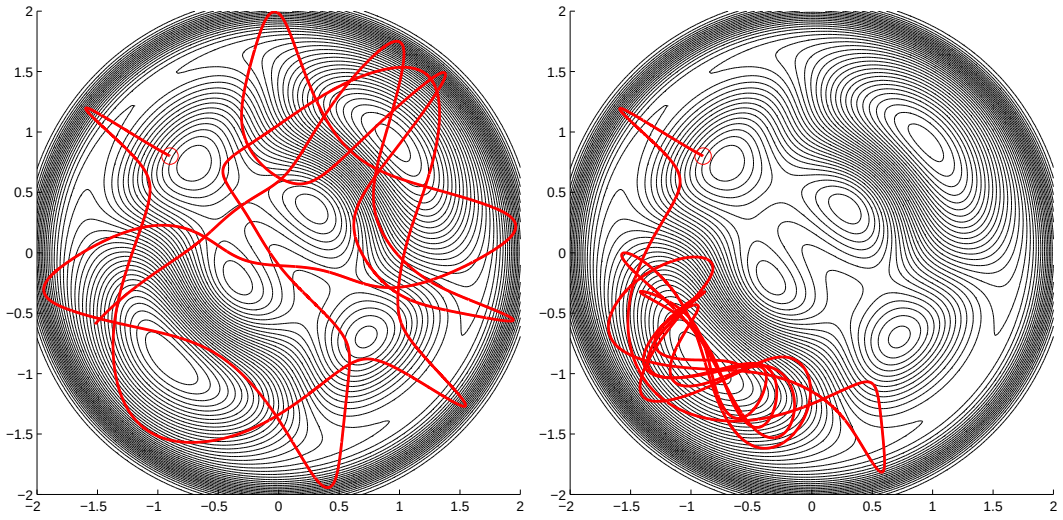

Figure 2: Particle trajectories in an undamped (left) and slightly damped (right) system.

find the steady states $(x_{\text{ss}}, y_{\text{ss}})$ of the damped system.

2. Assume that a particle is at rest in one of the potential wells. Sketch the "elevation" contours of the potential and draw some different paths between the two wells. For each path make a sketch of the potential energy along the path.

3. For a lossless system, what is the smallest kinetic energy a particle (starting in one of the wells) must have to be able to reach the other well? What is the initial velocity in x and y?

2.3.4 Numerical solution of ODEs

So how can we compute the trajectories of a particle governed by the equations of motion? For notational convenience let us consider the ordinary differential equation

$$\dot{u}(t) = f(u(t), t), \quad u(0) = u_0,$$

Again we may use a finite difference approximation to approximate the time derivative on the left

$$\frac{u(t + \Delta t) - u(t)}{\Delta t} \approx \dot{u}(t) = f(u(t), t).$$

Rearranging the equation allows us to compute $u_n \equiv u(t_n) = u(n\Delta t)$ according to

$$\begin{aligned} u_0 &= u_0, \\ u_{n+1} &= u_n + \Delta t\, f(u_n, t_n), \quad n = 0, 1, \ldots \end{aligned}$$

The above method is called *forward Euler*.

Accuracy and stability

It turns out that if we compute the solution up to some fixed time the error decreases with order of accuracy of the above method is one, or in other words the error is proportional to Δt. This might be surprising as the error in the approximation of one time step is actually proportional to $(\Delta t)^2$. Roughly speaking the explanation goes like this, if we compute to some time T with a time step Δ we have to take $N_t = \frac{T}{\Delta t}$ steps,

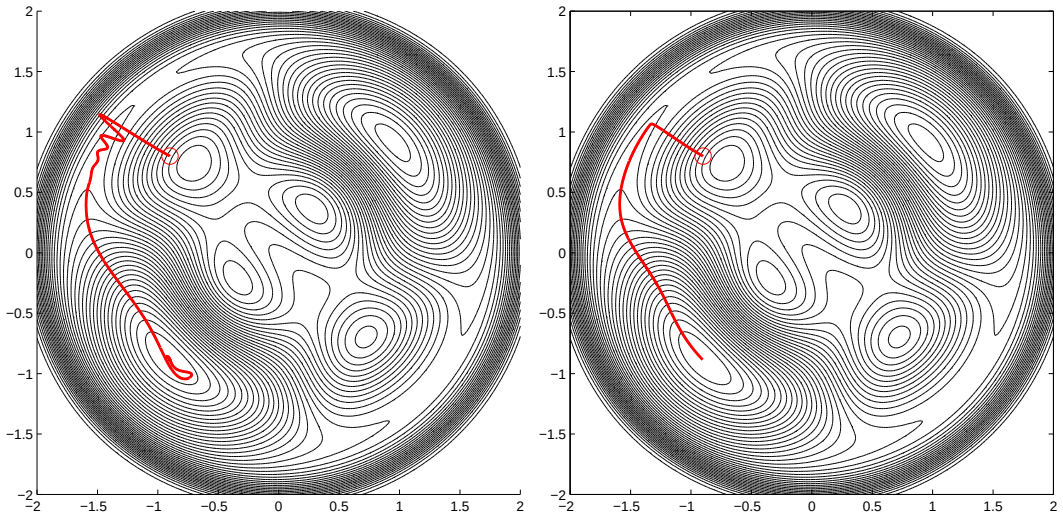

Figure 3: Particle trajectories in a strongly damped system (left).

if we commit an error of order $(\Delta t)^2$ in each of those steps the final error can be as large as $T\Delta t$ and hence of order of accuracy one.

In any case, any numerical method for solving ODEs has an error expansion which, assuming there is a maximum allowable error, sets an upper limit on Δt. This is the accuracy limit. To illustrate the stability limit we consider the following classroom exercise.

Classroom exercise

As just mentioned, there is also an upper limit on Δt that is dictated by stability considerations. Consider the special case when $f(u(t), t) = \lambda u(t), \quad \lambda < 0$ and $u_0 = 1$:

- Find the solution to
$$\dot{u}(t) = \lambda u(t), \quad u(0) = 1.$$
- What is the steady state? Is the solution of the continuous problem stable?
- Use Euler's method to compute the approximate solution at the first few time steps. What is the stability limit on Δt?
- Repeat the analysis (applied to $f(u(t), t) = \lambda u(t)$) for *backward Euler*

$$\begin{aligned} u_0 &= u_0, \\ u_{n+1} &= u_n + \Delta t\, f(u_{n+1}, t_{n+1}), \quad n = 0, 1, \ldots \end{aligned}$$

When can this method be useful? What is the additional complication compared to forward Euler when applied to an ODE with a more complicated right hand side?

2.3.5 A better method, the fourth order accurate Runge-Kutta method.

To be written

2.4 Project 1

- Write a Matlab function `potential(X,Y)` that returns the potential (12). Make sure the function can be evaluated for scalar and vector valued input, that is use `.` notation (`X.^2` not `X^2`).
- Use the `meshgrid` command to generate a grid on the domain $(x, y) \in [-1.6, 1.6]^2$. For example you can use `[X,Y] = meshgrid(-1.6:0.02:1.6,-1.6:0.02:1.6);`
- Compute the potential at all the points in your grid and use the `contour` command to plot your potential.
- Extend the function we wrote in class to return both the x and y derivative (the gradient) of the potential. The input to the function is the step size in x and y and two matrices `x` and `y` that contains the points where the gradient is desired, `[Epx,Epy]=Gradfun(x,y,hx,hy)`. Experiment to find a good value for `hx` and `hy`.
- Use the `quiver` command to plot the directional field for the ODE

```
[Epx,Epy]=Gradfun(X,Y,hx,hy);
id = 1:10:size(X,1);
quiver(X(id,id),Y(id,id),-Epx(id,id),-Epy(id,id),'r','LineWidth',2)
```

- Let `x` and `y` be the coordinates of a particle and write a program that uses Euler's method to evolve the position of the particle. Include the lossy term, β, and set it to zero for lossless systems.
- Experiment with different initial positions and different β to see if you can make the particle end up in both wells. How about the time step, is there a stability limit? Can you find a initial location that is attracted to different wells for different step sizes?
- Save the trajectory, for example using

```
X=[]; Y=[];
for ...
...
X=[X;x]; Y=[Y;y];
...
end
plot(X,Y)
```

- Once you have found the two wells, construct some curves between them and plot the potential energy along the curve. What is the smallest transition energy you can find?

2.5 Minimum energy paths via the string method

The above systems are representative for many physical systems in that they have more than one equilibrium state that it *relaxes* to as time passes. In our example there are two equilibrium states close to the the points $(-1,-1), (1,1)$. Now suppose we would like to pass from one to the other along some curve / path in the x, y-plane what is the maximum potential energy that we would see? This, of course, depends on the path we take and the potential energy itself. In Figure 4, where we display the energy along the paths $y = x$ and $y = x^9$ between $(-1,-1)$ and $(1,1)$. It can be seen that the shorter blue path also requires us to pass through a region with significantly higher potential energy than the red, longer, path. In other words the the longer (red) path has a lower transition energy (the maximum difference in potential energy from the start point) than the blue path. The path with the lowest possible transition energy will pass through the (or multiple) saddle point of the energy landscape.

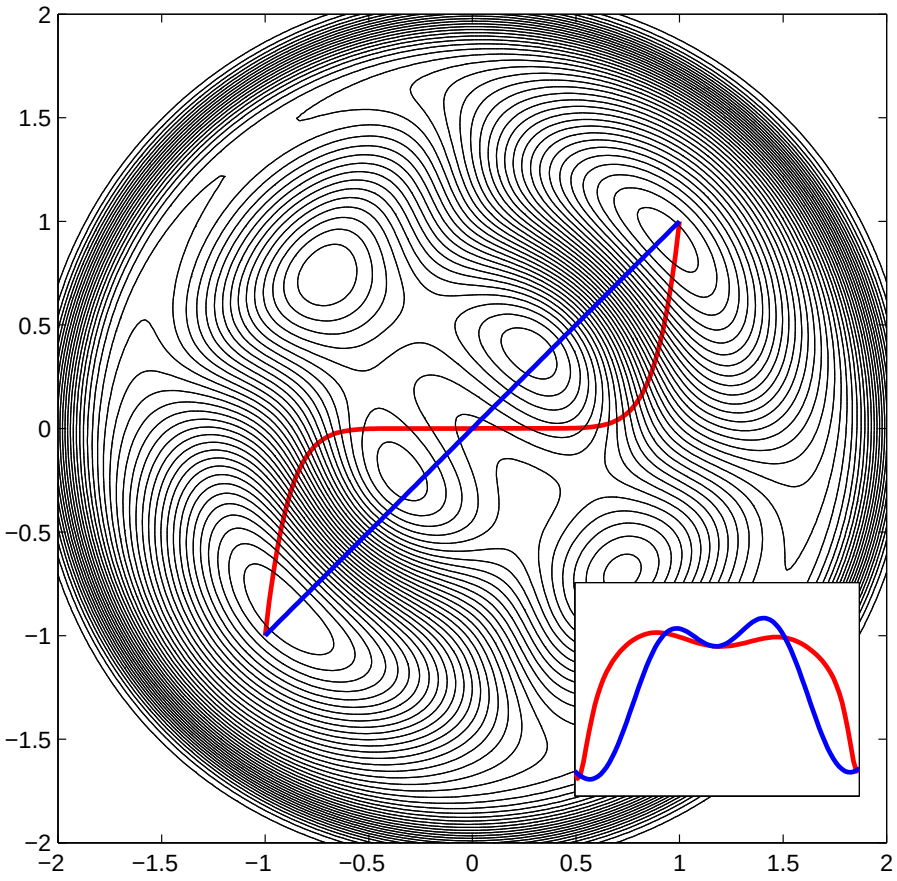

Figure 4:

From the inset we can also see that the end points of the curves are not the minima, thus neither of the paths are the minimum energy path (recall the MEP goes from one minima to the other through the saddle point). How can we find the MEP and the minima at the same time? One option is to use the string method.

The string method was introduced by E, Ren and Vanden-Eijnden [5, 3] is a convenient and intuitive method to find transition pathways or Minimum Energy Paths (MEPs) for physical systems whose stable states, $u^\star$, can be identified as equilibrium solutions of an evolution equation

$$\frac{\partial u(t,x)}{\partial t} = F(t,x,u). \tag{15}$$

That is $u^\star$ is a solution to

$$F(t,x,u^\star) = 0. \tag{16}$$

The string method consists of two steps:

1. An evolution step where a set of independent states (or images) are evolved according to some evolution equation. This is essentially the same procedure as discussed in project 1 but with more than one particle. If this step is performed by itself all of the images will end up in either of the minima.

2. A repametrization step. The basic idea is behind the reparametrization step is to prevent the images from "sliding down the string". Here the "string" refers to the curve obtained by connecting the

images by an interpolating curve and the reparametrization is simply to redistribute the images to new positions at equal arc length distance along the string.

We now describe the different steps of the string method and discuss the numerical methods used to perform the different steps. We start by the evolution step.

2.6 Evolution by gradient flow

Consider the following scenario. You are at the top of Sandia peak and a thunderstorm is approaching rapidly, you need to get down fast! Fortunately someone has given you a function $h(x, y)$ describing the height above the sea level. How should you descend down the mountain? This is the classic problem of the *law of the steepest descent*, solved first by Cauchy 1847. The solution is found by considering a path $(x(t), y(t)) = \mathbf{x}(t)$ for which the instantaneous change in height is $\frac{d}{dt}h(\mathbf{x}(t))$ defined as

$$\frac{d}{dt}h(\mathbf{x}(t)) = \frac{\partial h(\mathbf{x}(t))}{\partial x}x'(t) + \frac{\partial h(\mathbf{x}(t))}{\partial y}y'(t) = \nabla h(\mathbf{x}(t)) \cdot \mathbf{x}'(t). \tag{17}$$

Thus to descend as fast as possible is equivalent to making the right hand side of (17) as negative as possible, or in other words, making sure that the fastest path from a point $\mathbf{x}(0) = \mathbf{x}_0$, $\mathbf{x}(t)$ satisfies[1]

$$\mathbf{x}'(t) = -\nabla h(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{18}$$

The gradient flow equation (18) can be used for the evolution step in the string method.

Again, consider the potential

$$\mathcal{E}_{\mathrm{p}}(x, y) = (x-1)^2(x+1)^2 + y^4,$$

clearly the minimum energy path is the straight line $y = 0$. The gradient of the potential energy is

$$\nabla\mathcal{E}_{\mathrm{p}}(x, y) = \begin{pmatrix} 2(x-1)(x+1)^2 + 2(x+1)(x-1)^2 \\ 4y^3 \end{pmatrix},$$

which, along the minimum energy path, reduces to

$$\nabla\mathcal{E}_{\mathrm{p}}(x, 0) = \begin{pmatrix} 2(x-1)(x+1)^2 + 2(x+1)(x-1)^2 \\ 0 \end{pmatrix}.$$

Thus if a particle whose motion is governed by (18) is *on the MEP* it will only experience a force in the direction *tangent to the MEP* and directed towards either of the minima. This is an alternative definition of the MEP.

What happens if we are not on the MEP? Again we can look at the gradient at a point, say, $(1/2, 1)$

$$\nabla\mathcal{E}_{\mathrm{p}}(1/2, 1) = \begin{pmatrix} -3/2 \\ 4 \end{pmatrix}.$$

Now the direction of steepest descent is in the direction of the negative gradient, i.e. $[3/2. - 4]$ so the particle will move towards the MEP and towards the minima. To summarize, we have made the following two observations:

1. If we are on the MEP we stay on it but fall down to a minima.
2. If we are not on the MEP we fall towards it and towards the minima.

Thus if we could find a force that acts *along the string* but in the opposite direction of the gradient descent we could combine it with the forcing of the evolution step so that once the string is on the MEP they will balance and a discrete approximation to the MEP can be found. Enter reparametrization step!

[1] Recall that the inner product between two vectors $\mathbf{a}$, $\mathbf{b}$ can be written $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta$ which is maximally negative when the vectors are parallel and opposed.

2.7 Reparametrization

Assume that we had placed 11 particles placed at $x_j = -1 + j/5, \quad j = 0, \ldots, 10 = n, y_j = 0$ and that we evolved them according to (18) using forward Euler with a stepsize $\Delta t = 0.1$. The outcome of this process is displayed in Figure 5. As can be seen the particles are clustering towards the end of the curve and, if continued, the evolution will bring them into either of the minima. Note that the particles are still on the minimum energy path but give a worse representation of the potential along curve itself close to the saddle point where the density of particles is lower.

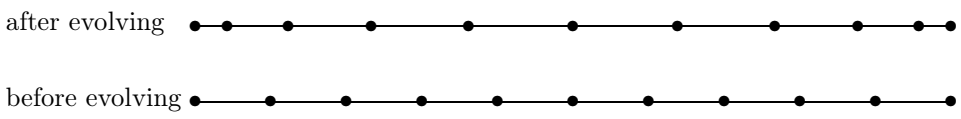

Figure 5:

To represent the curve better we would like to redistribute the particles in a way so that they are have the same individual spacing and stay on the curve. That is, if the curve is $(x(s), y(s)), \quad s \in [0, 1]$ with the evolved particles located at $s_0, s_1, \ldots, s_n$ we would like to place the particles at new arc lengths $\tilde{s}_i = i/n, \quad i = 0, \ldots, n$.

In our case the curve is a straight line but in a more general situation the curve itself will not be known only the location of each particle and we must find an approximation to the curve based on these locations. This is the interpolation problem which we discuss next.

2.8 Polynomial interpolation

The polynomial interpolation problem can be stated:

Problem 1 (Polynomial interpolation)**.** *Given a continuous function $f(x)$ defined on $a \leq x \leq b$ and $n+1$ ordered nodes*

$$a \leq x_0 < x_1 < \ldots < x_n \leq b,$$

in that interval. Find a polynomial, $p(x)$, of degree n such that the interpolation conditions

$$f(x_j) = p(x_j), \quad j = 0, \ldots, n, \tag{19}$$

hold.

The naive approach to finding such a polynomial is to consider the monomial

$$p(x) = c_0 + c_1 x + \ldots + c_n x^n,$$

and forming the equations

$$\begin{aligned} f(x_0) &= p(x_0) = c_0 + c_1 x_0 + \ldots + c_n x_0^n, \\ f(x_1) &= p(x_0) = c_0 + c_1 x_1 + \ldots + c_n x_1^n, \\ &\vdots \\ f(x_n) &= p(x_0) = c_0 + c_1 x_n + \ldots + c_n x_n^n. \end{aligned}$$

Or equivalently on matrix form $\mathbf{f} = \mathbf{A}\mathbf{c}$, where $\mathbf{f} = [f(x_0), \ldots, f(x_n)]^T$, $\mathbf{c} = [c_0, \ldots, c_n]^T$

$$\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}. \tag{20}$$

Here the matrix $\mathbf{A}$ is called the Vandermonde matrix. Solving for $\mathbf{c} = \mathbf{A}^{-1}\mathbf{f}$ solves the interpolation problem. Solving for the coefficients requires Gauss elimination which costs $\mathcal{O}(n^3)$ operations and each evaluation of $p(x)$ at a new x value costs $\mathcal{O}(n^2)$. This is an unacceptably high cost and as we will see there are more efficient ways to find and evaluate $p(x)$.

Lagrange interpolation

In fact, given the *data* $\mathbf{f}$ we can write down $p(x)$ directly

$$p(x) = \sum_{j=0}^{n} f(x_j) l_j(x), \tag{21}$$

where

$$l_j(x) = \prod_{i=0,\, j\neq i}^{n} \left(\frac{x - x_i}{x_j - x_i} \right), \quad j = 0, \ldots, n, \tag{22}$$

are the Lagrange polynomials. The existence and uniqueness of the polynomial $p(x)$ can be easily established from (21). Existence is obvious from the fact that each $l_j(x)$ is polynomial of degree n (and so is a linear combination of them) and have the property

$$l_j(x_i) = \begin{cases} 1 & i = j, \\ 0 & i \neq j. \end{cases}$$

On to uniqueness! If there was another polynomial $\tilde{p}(x)$, also of degree n, satisfying the interpolation conditions we could form $r(x) = p(x) - \tilde{p}(x)$ which would also be a polynomial of degree n. However $r(x)$ would have $n+1$ zeros (at the nodes) and a polynomial of degree n with more than n zeros must be identically zero, hence $p(x) \equiv \tilde{p}(x)$. This imply uniqueness.

The formula (21) is not suitable for computations as it requires $\mathcal{O}(n^2)$ operations for each new value of x. An alternative formulation requiring only $\mathcal{O}(n)$ operations is the **modified** Lagrange interpolation formula

$$p(x) = \Phi_n(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j).$$

Here

$$\Phi_n(x) = \prod_{i=0}^{n} (x - x_i),$$

and

$$w_j = \frac{1}{\prod_{i=0}^{n} (x_j - x_i)}.$$

Another, equally fast, alternative is the **barycentric formula**

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x - x_j} f(x_j)}{\sum_{j=1}^{n} \frac{w_j}{x - x_j}}, \quad x \neq x_j.$$

These and other forms of interpolation are discussed in the review article [2].

It is natural to ask how accurate the polynomial approximates the function $f(x)$. Assuming that $f(x) \in C^{n+1}([a,b])$, i.e. it is $n+1$ times continuously differentiable in the interval it can be shown that the error formula is

$$f(x) - p(x) = \frac{f^{(n+1)}(\rho)}{(n+1)!}\Phi_n(x),$$

for some $\xi \in [a,b]$.

Add a proof sketch using Rolle's theorem and the $(n+1)$th zero of

$$f(t) - p(t) - \Phi_n(t)\frac{f(x)-p(x)}{\Phi_n(x)}.$$

If we denote the maximum distance between two nearby points $h = \max|x_{j+1} - x_j|$ we can estimate $\|f(x) - p(x)\|_\infty$ by finding an upper bound on

$$|\Phi_n(x)| = |x - x_0| \cdots |x - x_j| \cdots |x - x_n|.$$

Suppose $x_j \le x \le x_{j+1}$, then to the right of x_{j+1} we can replace x with x_j. Similarly we can replace x with x_{j+1} to the left of x_j (note that if we try to replace them with x_j to the right etc. we do not get the $\le$ sign). We get:

$$\begin{aligned}&|x - x_0| \cdots |x - x_j| \cdots |x - x_n| \le \\ &|x_{j+1} - x_0| \cdots |x_{j+1} - x_{j-1}||x - x_j||x - x_{j+1}||x_j - x_{j+2}| \cdots |x_j - x_n|.\end{aligned}$$

With $h = \max_{0\le k\le n-1}[x_{k+1} - x_k]$ we can write

$$\begin{aligned}&|x - x_0| \cdots |x - x_j| \cdots |x - x_n| \\ \le &\underbrace{|x_{j+1} - x_1|}_{jh} \cdots \underbrace{|x_{j+1} - x_{j-1}|}_{2h}|x - x_j||x - x_{j+1}|\underbrace{|x_j - x_{j+2}|}_{\le 2h} \cdots \underbrace{|x_j - x_n|}_{(n-j)h} \\ = &|x - x_j||x - x_{j+1}|h^{n-1}j!(n-j)! \le \frac{|x_{j+1} - x_j|}{4}h^{n-1}j!(n-j)! \le \frac{h^{n+1}}{4}n!.\end{aligned}$$

Using the estimate above together with the error formula we get the bound

$$\|f - p\|_\infty \le \frac{h^{n+1}}{4(n+1)}\|f^{(n+1)}\|_\infty. \tag{23}$$

Piecewise linear interpolation

The formula (23) indicates that a large n implies a small error but this is only true if $\|f^{(n+1)}\|_\infty$ does not grow to fast. Add example using Runge's example and radius of convergence of analytic functions It is however not uncommon that the derivatives grow fast and as a result it is necessary to use a low order piecewise polynomial interpolation instead.

In piecewise polynomial interpolation a low order interpolating polynomial is constructed from the values of a set of nearby points. The most basic version is piecewise linear interpolation where one simply draws a straight line between two adjacent points. More formally, the piecewise linear interpolation between x_i and x_{i+1} interpolating y_i and y_{i+1} is

$$p_i(x_i + th_i) = y_i + t\Delta y_i, \quad t \in [0,1], \tag{24}$$

$$h_i = x_{i+1} - x_i, \tag{25}$$

$$\Delta y_i = y_{i+1} - y_i. \tag{26}$$

Temp	63	46	♣	41
Day	Nov. 1	Nov. 3	Nov. 5	Nov. 7

Classroom exercise

I recorded the temperature in Nob Hill for the first few days of November but unfortunately I spilled some coffee on my records. Help me find the missing temperature by interpolating a second degree polynomial to the known data.

(a) Formulate the interpolation problem using the naive (Vandermonde) approach.

(b) Write a program that computes the missing temperature. Solving a linear system in Matlab can be done by `x=A\b`

(c) Use linear interpolation and pen and paper to fill in the missing value.

Splines

Another form of piecewise interpolation are *splines*. The formula for a cubic spline between x_i and x_{i+1} and interpolating y_i and y_{i+1} with slopes k_i and k_{i+1} is

$$\begin{aligned} p_i(x_i + th_i) &= y_i + t\Delta y_i + t(1-t)g_i + t^2(1-t)c_i, \quad t \in [0,1], && (27)\\ h_i &= x_{i+1} - x_i, && (28)\\ \Delta y_i &= y_{i+1} - y_i, && (29)\\ g_i &= h_i k_i - \Delta y_i, && (30)\\ c_i &= 2\Delta y_i - h_i(k_i + k_{i+1}). && (31) \end{aligned}$$

Now suppose data y_i are given at x_i for $i = 1, \ldots, n$. We then have $n-1$ polynomials defined on intervals $I_i = \{x \in [x_i, x_{i+1}]\}$. We requite the polynomials to satisfy the interpolation conditions

$$p_i(x_i) = y_i, \quad i = 1 \ldots, n-1, \quad p_{n-1}(x_n) = y_n. \tag{32}$$

We also require continuity of the polynomial and its first two derivatives at the inner nodes

$$\begin{aligned} p_i(x_i) &= p_{i+1}(x_i), \quad i = 2 \ldots, n-1, && (33)\\ p_i'(x_i) &= p_{i+1}'(x_i), \quad i = 2 \ldots, n-1, && (34)\\ p_i''(x_i) &= p_{i+1}''(x_i), \quad i = 2 \ldots, n-1. && (35) \end{aligned}$$

In order to evaluate the formula (27) we must find the slopes $k_i, \quad i = 1, \ldots, n$. We can use (35) to construct $n-2$ equations. The second derivatives of the ith polynomial is

$$p_i''(x_i + th_i) = \frac{1}{h_i^2}\left(-2g_i + (2-6t)c_i\right), \tag{36}$$

so equating $p_i''(x_i + 1h_i) = p_{i+1}''(x_{i+1} + 0h_{i+1})$ yields

$$\frac{-2g_i - 4c_i}{h_i^2} = \frac{-2g_{i+1} + 2c_{i+1}}{h_{i+1}^2},$$

which can be rearranged into

$$h_{i+1}k_i + 2(h_i + h_{i+1})k_{i+1} + h_i k_{i+2} = 3\left(\frac{h_{i+1}\Delta y_i}{h_i} + \frac{h_i \Delta y_{i+1}}{h_{i+1}}\right), \quad i = 1, \ldots, n-2. \tag{37}$$

The expression (37) generates $n-2$ equations from the $n-2$ interior nodes but n k_i values are required. Two more equations from the boundary nodes x_1 and x_n must be added to close the system. There are several possible wasps to close the system, which one to use depends on the situation.

Natural Splines

In natural splines the second derivative is set to zero at the end points, i.e. $p_i''(x_1) = p_i''(x_n) = 0$ yielding the two equations

$$-2g_1 + 2c_1 = 0 \quad \Leftrightarrow \quad 2h_1k_1 + h_1k_2 = 3\Delta y_1, \tag{38}$$

$$g_{n-1} + 2c_{n-1} = 0, \quad \Leftrightarrow \quad h_{n-1}k_{n-1} + 2h_{n-1}k_n = 3\Delta y_{n-1}. \tag{39}$$

Clamped Splines

For clamped splines we set the first derivative at the end points

$$k_1 = \alpha, \tag{40}$$

$$k_n = \beta. \tag{41}$$

2.9 Reparametrization step (continued)

Let us now return to the reparametrization of the curve, $(x(s), y(s)), \quad s \in [0, 1]$, connecting the $n+1$ particles at $s_0, s_1, \ldots, s_n$. Now, before proceeding we must find $s_0, s_1, \ldots, s_n$, how can we do this? It is reasonable to set $s_0 = 0$ to begin with but how are we to find s_1? The formula for the arc length between two points along the parametrized curve is

$$s = \int_{s_j}^{s_{j+1}} \sqrt{(x'(s))^2 + (y'(s))^2}\, ds,$$

so if we could approximate this we would be ready to go! The numerical approximation of integrals is often referred to as numerical quadrature or simply quadrature.

2.10 Numerical quadrature

Add description of interpolatory quadrature, Euler and Trapetzoidal

Classroom exercise

Derive a composite quadrature rule that uses the midpoint of each interval. How does the error depend on the step size? What is the highest order polynomial that can be integrated exact?

2.11 Reparametrization step (continued)

Let us now return to the computation of $s_0, s_1, \ldots, s_n$ which in the case of piecewise linear interpolation can be approximated in the following compact way

$$l_0 = 0, \quad l_i = s_{i-1} + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2, \quad i = 1, \ldots, n,$$

which can then be normalized to

$$s_i = l_i/l_n, \quad i = 0, \ldots, n.$$

Project 2

Evolution step in the string method.

- Write a new program (you can use some of the functions from the first project) that solves the gradient flow equations

$$\dot{\mathbf{x}}_i = -\nabla \mathcal{E}_p(\mathbf{x}_i), \quad i = 0, \ldots, n, \tag{42}$$

Algorithm 1 String method

Require: Let $\{u_0, u_1, \ldots, u_n\}$ be an ordered set of states. At time $t = t_0$, let u_0 and u_n be two distinct sets within the domain of attraction of two distinct solutions, $u_0^\star$ and $u_n^\star \neq u_0^\star$, to (16). Let the intermediate states u_i, $i = 1, \ldots, n-1$, be defined as

$$u_i = (1 - \frac{i}{n})u_0 + \frac{i}{n}u_n, \quad i = 1, \ldots, n-1.$$

while $\tau > TOL$ **do**

Evolve (15) until time $t = t_0 + \Delta t$, i.e. find

$$u_i(t_0 + \Delta t, x), \quad i = 0, \ldots, n.$$

Compute the "arc length" s of the set of states:

$$l_0 = 0,\ l_i = s_{i-1} + \|u_i - u_{i-1}\|_2, \quad i = 1, \ldots, n,$$
$$s_i = l_i / l_n, \quad i = 0, \ldots, n.$$

Interpolate new states so that they are equidistant on the arc s:

$$v_i = \mathcal{I}(u_0, u_1, \ldots, u_n, i/n), \quad i = 0, \ldots, n.$$

Compute tolerance:

$$\tau = \max_i \|u_i - v_i\|.$$

Update the old states with the interpolated states: $u_i = v_i, \quad i = 0, \ldots, n.$

end while

for $n+1$ independent particles. You can use the potential from day 1 or you can try to come up with one yourself. Try your program with different sets of initial data corresponding to some different curves (a straight line, a parabola etc.). Where does the points end up?

Reparametrization step To prevent the particles from falling down into the wells we will now connect them by placing them on a curve (string) and redistribute them with equidistant spacing in the arclength. The curve is described by $\mathbf{x}(s)$ where s is the arc length starting at zero at one end and ending at $s_{\max}$ at the other. The location of the particles are $\mathbf{x}_i \equiv \mathbf{x}(s_i)$, $i = 0, \ldots, n$.
Now, the redistributing consists of replacing the $n+1$ particles $\mathbf{x}(s_i)$ with $n+1$ new particles $\tilde{\mathbf{x}}(\tilde{s}_i)$ located at $\tilde{s}_i = i\frac{s_{\max}}{n}$, $i = 0, \ldots, n$.
To perform the redistribution we must be able to compute the length of the string and all the individual locations of the particles $\mathbf{x}(s_i)$.

- Write a function that computes the arc length and all the individual locations of the particles on a (discretized) curve with $n+1$ points. For example `s = getAL(x,y)`, where `x` and `y` are vectors with the positions in x and y and `s` is a vector with the arc length positions for the points, starting with 0 and ending with $s_{\max}$.

- Write a function `xi=myintrp(x,s,si)` that performs piecewise linear interpolation at the points in the vector `si`. Note that to get the $n+1$ new particles $\tilde{\mathbf{x}}(\tilde{s}_i)$ you should use your interpolation routine for both x and y.

- Combine the evolution step and the reparametrization (do one step at the time) step to obtain the MEP for your potential.

- Once the computation is converged you should evaluate the potential along the sting and find the

energy barrier from one well to the other. Interpolate the potential along the curve to find a more accurate value for the maxima. How does the maxima depend on the number of particles?

- Experiment with different initial conditions, potentials, step sizes etc.
- Use Matlabs built in interpolation routines `interp1` (do `help interp1`) and evolve the equations with fourth order Runge-Kutta. Test your two programs on a problem where you know what the MEP is. Which method is more accurate?

3 An artsy interlude, Bezier curves

Bezier curves are splines that allow the user (you!) to control the slope at the endpoints. They were developed by Pierre Bezier who was an automobile designer at Renault and independently by Paul de Casteljau at Citroen. Today, Bezier curves are one of the cornerstones of computer aided design (CAD) and can be used to create curves and surfaces that are appealing to the eye.

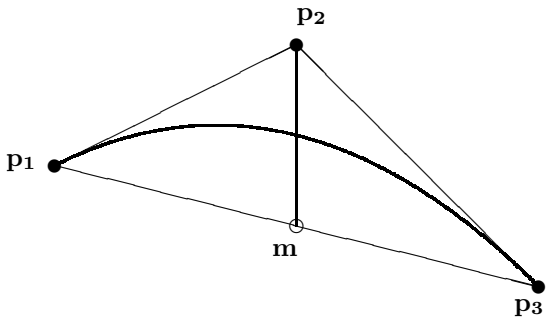

Figure 6: A quadratic Bezier curve.

The quadratic Bezier curve

Bezier curves can be of any degree but the most common are quadratic and cubic. Common to all Bezier curves are that they are made up of a start and an end point and a a number of control points. For example the quadratic Bezier curve, see Figure 6, is fully described by a start point $\mathbf{p_1}$, a control point $\mathbf{p_2}$ and an end point $\mathbf{p_3}$. The formula describing the curve is

$$\mathcal{B}_q(t) = (1-t)^2\mathbf{p_1} + 2(1-t)t\mathbf{p_2} + t^2\mathbf{p_3}, \quad t \in [0,1].$$

Bezier curves have some interesting properties. Clearly the curve starts in $\mathcal{B}_q(0) = \mathbf{p_1}$ and ends in $\mathcal{B}_q(1) = \mathbf{p_3}$. What about the tangent at those points? We have that

$$\mathcal{B}'_q(t) = -2(1-t)\mathbf{p_1} + (2-4t)\mathbf{p_2} + 2t\mathbf{p_3}, \quad t \in [0,1],$$

and find that $\mathcal{B}'_q(0) = 2(\mathbf{p_2} - \mathbf{p_1})$ and $\mathcal{B}'_q(1) = -2(\mathbf{p_2} - \mathbf{p_3})$. That is, the control point decides the tangent of the curve at the start and endpoint.

Another interesting fact is that the point $\mathcal{B}_q(1/2)$ is at the midpoint along the straight line from $\mathbf{m} = (\mathbf{p_1} + \mathbf{p_3})/2$ to $\mathbf{p_2}$ (can you show this?, what is the slope?).

The cubic Bezier curve

The cubic bezier curve is

$$\mathcal{B}_q(t) = (1-t)^3\mathbf{p_1} + 3(1-t)^2t\mathbf{p_2} + 3(1-t)t^2\mathbf{p_3} + t^3\mathbf{p_4}, \quad t \in [0,1].$$

Two examples of a cubic Bezier curve are given in Figure 7. Again the tangent at the start and endpoint can be found from the line connecting the start and endpoint and the closest control point. Note that the curve can overlap itself as it does to the right in Figure 7. The cubic Bezier curve will cut the line connecting the midpoints $\mathbf{m}_1 = (\mathbf{p}_1 + \mathbf{p}_4)/2$ and $\mathbf{m}_2 = (\mathbf{p}_2 + \mathbf{p}_3)/2$ at $3/4$ and at $t = 1/2/$ The slope of $\mathcal{B}_q(t)$ at $t = 1/2$ can be found from the line connecting the other two midpoints.

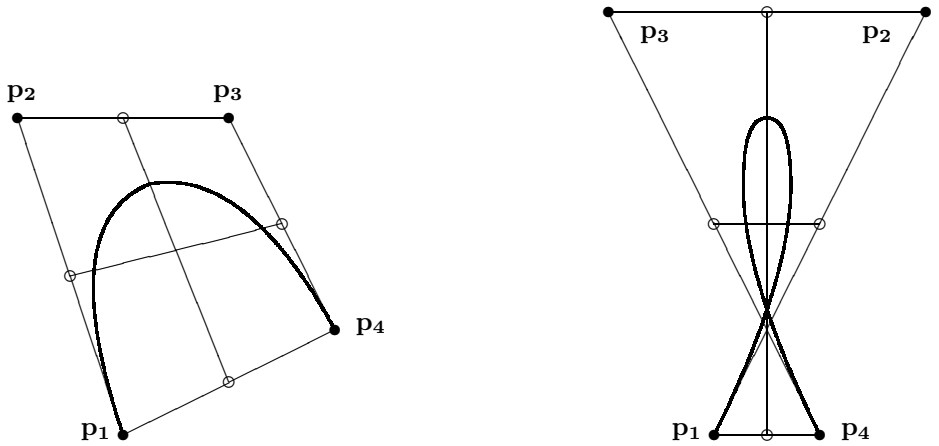

Figure 7: Two cubic Bezier curves.

Project 3

Bezier curves (pretty pictures)

- Draw a cascade of m cubic Bezier curves, all starting in the same point (x_1, y_1) and ending at a vertical tangent in the point (x_4, y_4). The angel of the tangent at the start should be $2\pi j/m$, $j = 1, \ldots, m$ with m somewhere in-between 60 and 120. Choose the distance to the control points for "maximal artistic impact".
- Use four cubic Bezier curves to approximate the unit circle. How big is the maximal error in your approximation?
- Use the results from the classroom exercise to create a sequence of Bezier curves describing a letter, your full name or a shape you like. Plot the Bezier curves and fine tune your design.
- Add a calligraphic effect by plotting the curve many times with a small displacement added. If the curve is stored in `X` and `Y` do `for k = 1:10, plot(X+k*d,Y+k*d)` for some small value of `d`.
- No need to stay in two dimensions, the formulas extend to three dimensions. Use `plot3` to create some art!
- Look up string art on Google. Experiment with different rotations and displacements of your curves to see if you can create some art.

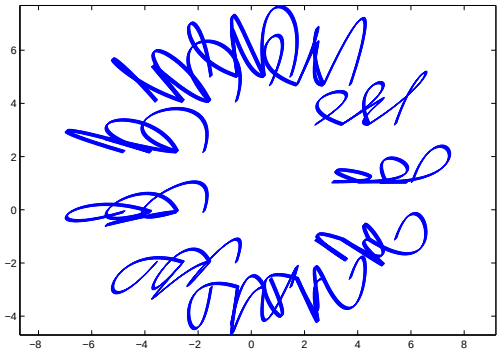

Figure 8: A cascade of calligraphed Bezier curves.

4 Scalar conservation laws, numerics and theory

The scalar conservation law is

$$u_t + f(u)_x = 0, \quad -\infty < x < \infty, t > 0, \tag{43}$$
$$u(0, x) = u_0(x). \tag{44}$$

This is a time dependent partial differential equation that describes the conservation of the quantity u. Assuming $f(u)$ vanishes as $|x| \to \infty$ we find that

$$\frac{d}{dt}\int_{-\infty}^{\infty} u(t, x)dx = [f(u)]_{-\infty}^{+\infty} = 0,$$

hence the integral of u is conserved in time. If we restrict the integration to an interval $x \in [a, b]$ we further find the change of the amount of u inside that interval is decided by the flux of u at b subtract the flux at a:

$$\frac{d}{dt}\int_{a}^{b} u(t, x)dx = f(u(t, b)) - f(u(t, a)).$$

The most basic example of a flux function $f(u)$ is $f(u) = au$ where a is a real number. With this choice of flux the conservation law reduces to the advection equation

$$u_t + au_x = 0, \quad -\infty < x < \infty, t > 0, \tag{45}$$
$$u(0, x) = u_0(x), \tag{46}$$

which is an example of a linear hyperbolic equation.

Understanding the properties of the advection equation is crucial for the analysis of more complicated flux functions. The importance of (45) is clear if we first rewrite (43) on so called quasi-linear form

$$u_t + A(u)u_x = 0. \tag{47}$$

Here $A(u) = f'(u)$. An important tool in applied mathematics used to analyze local properties of the solution u is linearization. Linearization simply means that we split $u = U + v$ into a "small" and a "big" part v and U, stick it into the equation expand non-linear terms in Taylor series and neglect quadratic and higher order terms. In the particular case when U is a constant the nonlinear equation (47) reduces to the advection equation

$$v_t + A(U)v_x = 0. \tag{48}$$

For further details on linearization see Chapter 1.3 in [4].

The solution to the advection equation is the *traveling wave*

$$u(x, t) = u_0(x - at),$$

which can be seen by differentiating with respect to time and space

$$\frac{\partial u(t, x)}{\partial t} = -au_0'(x - at),$$

$$\frac{\partial u(t, x)}{\partial x} = u_0'(x - at).$$

The name advection equation is very a appropriate as the initial data simply gets advected as time goes by. For example, consider the solution at $x = 0$ and $t = 0$, $u_0(0)$, at a later time $t = \tau$ the value $u_0(0)$ is located at $x = a\tau$, i.e. it has been advected in space. More generally, the value at some $x = \tilde{x}$, $u_0(\tilde{x})$ at $t = 0$ will thus advect to $x = \tilde{x} + at$ at a later t. The solution is constant along the straight lines $x = \tilde{x} + at$ in the $x - t$ plane. These straight lines are the characteristics of the advection equation.

Characteristics

The characteristics (or characteristic curves) are the curves in the $x-t$ plane defined by

$$\frac{dX(t)}{dt} = A(u(x,t)). \tag{49}$$

By using the chain rule and this definition we find

$$\frac{du(t,X(t)))}{dt} = \frac{\partial u}{\partial t} + \frac{dX(t)}{dt}\frac{\partial u}{\partial x} = \frac{\partial u}{\partial t} + A(u)\frac{\partial u}{\partial x} = 0,$$

that as long as the solution is smooth it is constant along the characteristic. In the case of the advection equation $A(u) = a$ and it is easy to integrate (49) in time to find $X(t) = at + C$.

4.1 Numerical solution of the advection equation

We have already seen how to approximate derivatives using Taylor expansion and we could try the same approach here. For example we could approximate $u_t + au_x = 0$ by a finite difference approximation using

$$\frac{\partial u}{\partial t} \approx \frac{u(t+\Delta t, x) - u(t,x)}{\Delta t},$$

$$\frac{\partial u}{\partial x} \approx \frac{u(t, x+h) - u(t, x-h)}{2h}.$$

It will be convenient to introduce a more compact notation for the different methods we will describe. We therefore introduce the grid $x_j = jh$ and a "time grid" $t_n = n\Delta t$ which allows us to introduce a grid function $u_j^n = u(t_n, x_j)$. Finally, we also introduce $\lambda = \frac{a\Delta t}{h}$. With this notation we may use the above approximations to obtain the following scheme:

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2}(u_{j+1}^n - u_{j-1}^n), \tag{50}$$

To test the method we restrict the domain to $x \in [0,1]$ and use periodic boundary conditions $u(t,0) = u(t,1)$ and set the initial condition to be $u(0,x) = \sin 2\pi x$. We choose $h = 1/100$ and $\lambda = 0.8$ and evolve the solution up to time $t = 1.2$. In Figure 9 we have plotted snapshots of the solution at $t = 1.0$ and 1.2. As can be seen the solution looks smooth at the first time but very jagged at $t = 1.2$. If we continue to evolve

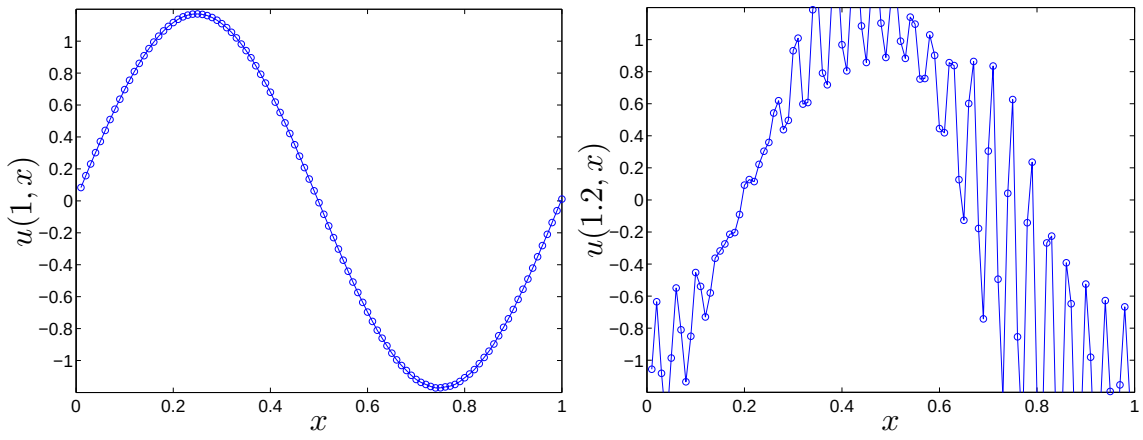

Figure 9:

the solution it will grow to a state where it is no longer resembling the true solution (recall the solution is just advection of the initial data).

Before discussing why the numerical solution behaves like this we consider a small modification of the scheme (50), replacing u_j^n by the average of two nearby points yielding the Lax-Friedrich scheme:

$$u_j^{n+1} = \frac{u_{j+1}^n + u_{j-1}^n}{2} - \frac{\lambda}{2}(u_{j+1}^n - u_{j-1}^n), \tag{51}$$

Performing the same computation with the Lax-Friedrich scheme we see in Figure 10 that the solution now stays smooth and bounded. Is there a way to understand when a scheme blows up or not?

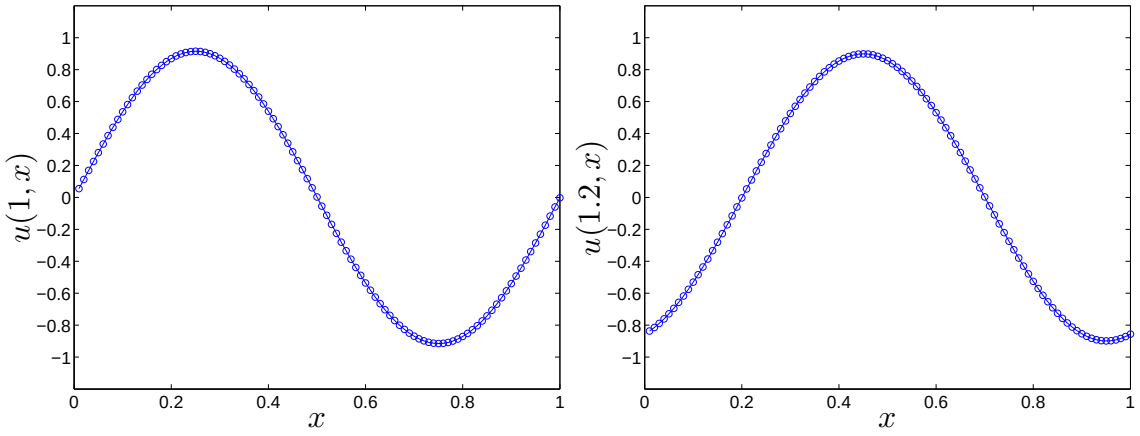

Figure 10:

von Neumann analysis

If we could find a compact way to relate the solution at a previous time step to the solution to the next, say, $u_j^{n+1} = gu_j^n$, we could analyze if the magnitude of the solution, $|u_j^n|$, was growing or not.

If the scheme is linear and the problem is linear and periodic von Neumann analysis will help us answer this question. In this context we define a linear scheme as a scheme that can be written as $u_j^{n+1} = F(\ldots, u_{j-1}^n, u_j^n, u_{j+1}^n, \ldots)$ and for which it holds true that

$$F(\ldots, v_{j-1}^n + w_{j-1}^n, v_j^n + w_j^n, v_{j+1}^n + w_{j+1}^n, \ldots) = F(\ldots, v_{j-1}^n, v_j^n, v_{j+1}^n, \ldots) + F(\ldots, w_{j-1}^n, w_j^n, w_{j+1}^n, \ldots).$$

The von Neumann analysis uses the linearity together with the fact that any L-periodic function, $q(x)$, can be expressed as a Fourier series

$$q(x) = \sum_{k=-\infty}^{\infty} c_k e^{\frac{i2\pi kx}{L}},$$

recognizing that linearity makes it sufficient to analyze a single mode $e^{\frac{i2\pi kx}{L}}$. With $\xi = \frac{2\pi k}{L}$ and $x_j = jh$ we may plug in $u_j^n = e^{ijh\xi}$ into the scheme. We get

$$u_j^{n+1} = g(\xi)u_j^n = g(\xi)e^{ijh\xi}. \tag{52}$$

Now if $|g(\xi)| \leq 1$ then the solution will not blow up as it will not get magnified from one step to the next.

We try this out for the Lax-Friedrich scheme

$$\begin{aligned} u_j^{n+1} &= \frac{e^{i(j-1)h\xi} + e^{i(j+1)h\xi}}{2} - \lambda \frac{e^{i(j+1)h\xi} - e^{i(j+1)h\xi}}{2} \\ &= \left(\frac{e^{-ih\xi} + e^{ih\xi}}{2} - \lambda \frac{e^{-ih\xi} - e^{ih\xi}}{2} \right) u_j^n \\ &= \underbrace{(\cos \xi h - i\lambda \sin h\xi)}_{g(\xi)} u_j^n. \end{aligned}$$

With $|g(\xi)|^2 = \cos^2 h\xi + \lambda^2 \sin^2 h\xi$ we see that a requirement for stability is that $|\lambda| \leq 1$ or equivalently $\Delta t \leq \frac{h}{|a|}$.

Classroom exercise

Repeat the von Neumann analysis for the following schemes.

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2}(u_{j+1}^n - u_{j-1}^n), \tag{53}$$

$$u_j^{n+1} = u_j^n - \lambda(u_j^n - u_{j-1}^n), \tag{54}$$

$$u_j^{n+1} = u_j^n - \lambda(u_{j+1}^n - u_j^n). \tag{55}$$

4.2 Geometric interpretation - domain of dependence and the CFL condition

Let us again consider the solution to the advection equation at some point X and some time T. The solution $u(T, X)$ then depends only on the initial data at a single point $u_0(X - aT)$ as is illustrated to the left in Figure 11. The *domain of dependence* of the a general hyperbolic problem is defined as all the points on the x-axis that influence the solution at some later time (here $t = T$). For the advection equation the domain of dependence is simply the point $X - aT$.

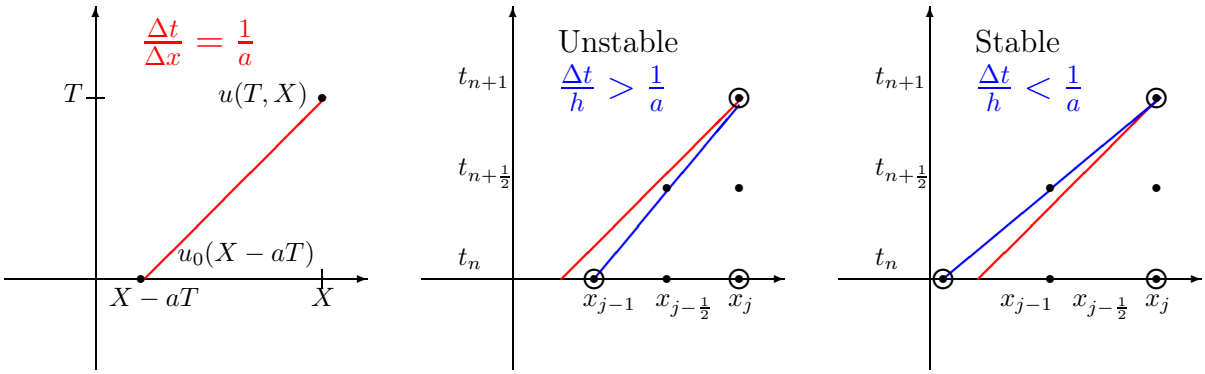

Figure 11: Geometric interpretation of stability.

The sketches at the center and right of Figure 11 illustrates the numerical method (54) which uses u_{j-1}^n and u_j^n to compute u_j^{n+1}. The open circles are the grid points used on a coarse grid and the solid circles indicate the points used on a refined grid. The *domain of dependence* of this numerical scheme is the triangle made up of the open circles. In the center figure the time step is so large that $\Delta t > h/a$ and the intersection of the left edge of the numerical domain of dependence and the x-axis is to the right of $X - aT$. As the grid is refined more and more points will be added to the base of the triangle but the point closest to the red

line will always be x_{j-1}. The approximation will therefore not be convergent as we can change the value of $u_(X - aT)$ without changing the numerical solution at (t_{n+1}, x_j).

On the other hand for the case when $\Delta t < h/a$ depicted to the right refining the grid results in a better and better approximation as the number of points in the base increase. It turns out that a *necessary* condition for stability of a numerical scheme (of the type considered here) for a conservation law is that the domain of dependence of the equation is contained within numerical domain of dependence. This is the *Courant - Friedrich -Levy* (CLF) condition. Associated to the CFL condition is the CFL number $|a_{\max}|\frac{\Delta t}{h}$, where $|a_{\max}|$ is the wave speed with largest magnitude. The CFL condition can be expressed $|a_{\max}|\frac{\Delta t}{h} \leq 1$. The fact that the CFL condition is only necessary is illustrated by the scheme (50).

4.3 A non-linear conservation law - Burgers equation

Let us now return to the scalar conservation law (43)-(44)

$$u_t + f(u)_x = 0, \quad -\infty < x < \infty, t > 0, \tag{56}$$
$$u(0, x) = u_0(x), \tag{57}$$

and consider the flux $f(u) = \frac{u^2}{2}$. This choice of flux yields the famous Burgers equation.

If we carry out the differentiation we find

$$u_t + uu_x = 0,$$

thus the characteristics of this equation are solutions to

$$\frac{dX(t)}{dt} = u. \tag{58}$$

Let us first draw the characteristics at $t = 0$ for some smooth initial data, see Figure 12. We know that the solution is constant along the characteristics as long as it is smooth, but looking at the right sketch in Figure 12 we see that if we follow the characteristics they will eventually cross. When the characteristics cross the solution will no longer be smooth as there will be two values of u that will occupy the same x value. This is a geometrical interpretation of the loss of smoothness that can occur in Burgers equation.

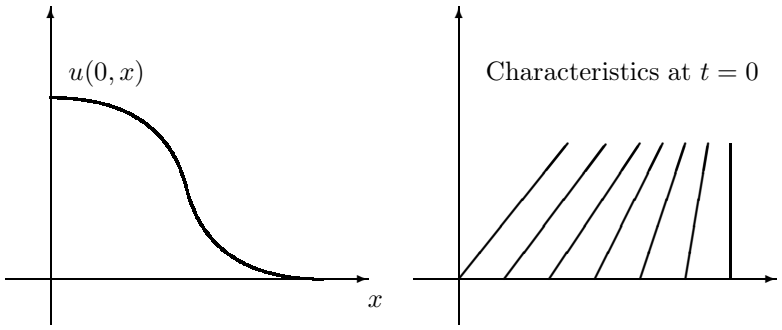

Figure 12:

Alternatively we can see the loss of smoothness from the exact solution $u(t, x) = u_0(x - A(u)t)$. If we differentiate with respect to time we get

$$u_t = -\frac{u_0' a}{1 + u_0' a' t},$$

thus there will be a blowup of the derivatives when $t = -\frac{1}{u_0' a'}$.

The set of possible solutions for a non-linear conservation law is much richer than for a linear problem but a full description of the theory of non-linear conservation laws is beyond the scope of this class. A central issue is that the uniqueness of the solution is not obvious as you will see next.

Classroom exercises

1. Check that $u(t,x) = u_0(x - A(u)t)$ is a solution to (43)-(44).
2. Show that both
$$u(t,x) = \begin{cases} -1, & x < 0, \\ 1, & x > 0, \end{cases} \tag{59}$$
and
$$u(t,x) = \begin{cases} -1, & x < -t, \\ \frac{x}{t}, & -t < x < t, \\ 1, & x > t, \end{cases} \tag{60}$$
are solutions (ignore $x = 0$, $t = 0$ and the other places where the derivative does not exist.) to Burgers equation.
3. Draw the characteristics for both the solutions above.

4.3.1 Some schemes for non-linear conservation laws

As you have seen the conservation law (43) can support more than one solution. How can we know which one is the right solution?

One way to single out the right solution is to first consider a slightly viscous problem

$$u_t + f(u)_x = \varepsilon u_{xx}, \quad -\infty < x < \infty, t > 0, \tag{61}$$
$$u(0,x) = u_0(x), \tag{62}$$

which has a unique solution, and then send the "viscosity" ε to zero. This solution is called the vanishing viscosity solution and will be the physically relevant solution.

Another option is to use a "good" numerical method to solve the conservation law. Here we will consider two schemes, upwind and Lax-Friedrich. Both can be cast in the form

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{h}(h_{j+\frac{1}{2}} - h_{j-\frac{1}{2}}).$$

Where $h_{j+\frac{1}{2}}$ is the numerical flux function and is defined as

$$h_{j+\frac{1}{2}} = \frac{1}{2}(f(u_{j+1} + f(u_j)) - \frac{1}{2}\left|\frac{f(u_{j+1} - f(u_j)}{u_{j+1} - u_j}\right|(u_{j+1} - u_j), \tag{63}$$

for the upwind method and

$$h_{j+\frac{1}{2}} = \frac{1}{2}(f(u_{j+1} + f(u_j)) - \frac{h}{2\Delta t}(u_{j+1} - u_j) \tag{64}$$

for the Lax-Friedrich method.

Project day 4

The advection equation

- Write a program that solves the advection equation $u_t + u_x = 0$, $t > 0$, $x \in [0, 1]$ with initial data $u(0, x) = u_0(x)$ and periodic boundary conditions, $u(t, 0) = u(t, 1)$ using the upwind scheme. Try with different initial data

$$u_0(x) = \sin(2\pi x),$$

$$u_0(x) = \begin{cases} 1 & |x - 0.5| < 0.25, \\ 0 & \text{otherwise.} \end{cases}$$

- Compute the solution at $t = 1$ with CFL number $\Delta t/h = 0.5$ and compare how the error decays as you make the grid size h smaller and smaller (you can use the max error, `max(abs(uinitial-u))`). It is suitable to plot this in a loglog plot, `loglog(H,E)` where `H` is a vector containing all the step sizes you used and `E` is a vector with the corresponding errors.

- Repeat with Lax-Friedrich.

- Repeat for Burger's equation.

Burger's turbulence[2].

Add a viscous term to Burger's equation

$$u_t + (\frac{u^2}{x})_x = \varepsilon u_{xx},$$

and discretize the it with a centered difference $u_{xx} \approx (u_{j+1} - 2u_j + u_{j-1})/h^2$. Fix, h (fairly small), use random initial data and solve until $t = 15$. Monitor the norm of u (`norm(u)`) and plot it as a function of time in a loglog plot. Does the decay of $\|u\|$ follow a power law? if so what is the exponent?

- Do this experiment for three values of ε. First choose ε a bit larger than h (you might have to reduce the CFL number for the solution to be stable). For this case you may use either of the methods. Next choose ε a fraction of h and use both methods. Finally, set $\varepsilon = 0$ and repeat with both methods. Can you say something about the built in dissipation of the two methods?

- For the inviscid equation, if you start with random data, how many shocks will there be after long time?

[2]There is a somewhat inaccessible review at http://arxiv.org/pdf/0704.1611v1.pdf

5 Wellposedness, energy estimates and the energy method for finite difference approximations

to be written

5.1 Project 5

Heat Equation

- Consider the heat equation
$$u_t = u_{xx}, \quad x \in [0,1], t > 0,$$
with initial and boundary conditions
$$u(x,0) = 0, \quad u(0,t) = \sin(\omega t), \quad u(1,t) = 0.$$
Discretize the equation using Crank-Nicolson, i.e.
$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{1}{2} D_+ D_- (u_j^{n+1} + u_j^n).$$
Here $D_+ D_- v_j = (v_{j+1} - 2v_j + v_{j-1})/h^2$.
- *Classroom exercise.* Sketch a matlab program that implements the scheme.
- Implement your solver in Matlab. For a set of different frequencies, ω, solve the equation until it reaches a "steady state". How does the envelope depend on the frequency?
- This program solves the the heat equation in two dimensions, $u_t = \nabla^2 u$. Try to run it and try to understand it well enough so that you can add a source to the equations, $u_t = \nabla^2 u + S(x,y,t)$.

```
n=100;
nt = 100;
N = n*n;
dt = tend/(nt-1);
h = 1/(n+1);
x = h*(1:n)';
y = h*(1:n)';
[X,Y] = meshgrid(x,y);
% Initial data
u = -(1+X.*Y);

e = ones(n,1);
A = spdiags([e -2*e e], -1:1, n, n);
A = A/h^2;
I = speye(n);
L = kron(A,I) + kron(I,A);
DF = speye(N) + 0.5*dt*L;
DB = speye(N) - 0.5*dt*L;
u = reshape(u,N,1);
for iter = 1:nt
  u = DB\(DF*u);
  mesh(X,Y,reshape(u,n,n))
  drawnow
end
```

References

[1] Carl M Bender and Steven A Orszag, *Advanced mathematical methods for scientists and engineers*, Springer, New York, 1999.

[2] Jean-Paul Berrut and Lloyd N. Trefethen, *Barycentric lagrange interpolation*, SIAM Review **46** (2004), no. 3, pp. 501–517 (English).

[3] Weinan E, Weiqing Ren, and Eric Vanden-Eijnden, *Simplified and improved string method for computing the minimum energy paths in barrier-crossing events*, Journal of Chemical Physics **126** (2007), no. 16, 164103.

[4] H Kreiss and Jens Lorenz, *Initial-boundary value problems and the navier-stokes equations*, Pure and applied mathematics, vol. v. 136, Academic Press, Boston, 1989.

[5] E. W, WQ Ren, and E Vanden-Eijnden, *String method for the study of rare events*, Physical Review B **66** (2002), no. 5, 052301.