Massachusetts
Institute of
Technology

# Advection problems

## Flow equations etc.

18.303 Linear Partial Differential Equations: Analysis and Numerics

# Elementary advection problem

Let $u : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. This is a $1+1$ dimensional field that is varying in time and space. We will look at a differential equation

$$u_t(t,x) + cu_x(t,x) = 0$$

with some initial condition $u(0,x) = u_0(x)$. Here $c \in \mathbb{R}$ is a model parameter.

# Elementary advection problem

Let $u : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. This is a $1+1$ dimensional field that is varying in time and space. We will look at a differential equation

$$u_t(t,x) + cu_x(t,x) = 0$$

with some initial condition $u(0,x) = u_0(x)$. Here $c \in \mathbb{R}$ is a model parameter.

Let us make a change of coordinates $x = y + ct$. $y$ will be the coordinate that is moving at a constant velocity $c$. Let $v(t,y) = u(t,x)$.

# Elementary advection problem

Let $u : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. This is a $1+1$ dimensional field that is varying in time and space. We will look at a differential equation

$$u_t(t,x) + cu_x(t,x) = 0$$

with some initial condition $u(0,x) = u_0(x)$. Here $c \in \mathbb{R}$ is a model parameter.

Let us make a change of coordinates $x = y + ct$. $y$ will be the coordinate that is moving at a constant velocity $c$. Let $v(t,y) = u(t,x)$.

Using the chain rule gives

$$v_t(t,y) + \frac{\mathrm{d}y}{\mathrm{d}t} v_y(t,y) + c\frac{\mathrm{d}y}{\mathrm{d}x} v_y(t,y) = v_t(t,y) - cv_y(t,y) + cv_y(t,y) = 0.$$

# Elementary advection problem

Let $u : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. This is a $1+1$ dimensional field that is varying in time and space. We will look at a differential equation

$$u_t(t,x) + cu_x(t,x) = 0$$

with some initial condition $u(0,x) = u_0(x)$. Here $c \in \mathbb{R}$ is a model parameter.

Let us make a change of coordinates $x = y + ct$. $y$ will be the coordinate that is moving at a constant velocity $c$. Let $v(t,y) = u(t,x)$.

Using the chain rule gives

$$v_t(t,y) + \frac{\mathrm{d}y}{\mathrm{d}t} v_y(t,y) + c\frac{\mathrm{d}y}{\mathrm{d}x} v_y(t,y) = v_t(t,y) - cv_y(t,y) + cv_y(t,y) = 0.$$

We see that the solution is given by $v_t(t,y) = 0$, which can be solved by integrating from 0 to $t$ giving

$$v(t,y) = u_0(y).$$

# Elementary advection problem

Let $u : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. This is a $1+1$ dimensional field that is varying in time and space. We will look at a differential equation

$$u_t(t,x) + cu_x(t,x) = 0$$

with some initial condition $u(0,x) = u_0(x)$. Here $c \in \mathbb{R}$ is a model parameter.

Let us make a change of coordinates $x = y + ct$. $y$ will be the coordinate that is moving at a constant velocity $c$. Let $v(t,y) = u(t,x)$.

Using the chain rule gives

$$v_t(t,y) + \frac{\mathrm{d}y}{\mathrm{d}t} v_y(t,y) + c\frac{\mathrm{d}y}{\mathrm{d}x} v_y(t,y) = v_t(t,y) - cv_y(t,y) + cv_y(t,y) = 0.$$

We see that the solution is given by $v_t(t,y) = 0$, which can be solved by integrating from 0 to $t$ giving

$$v(t,y) = u_0(y).$$

This means that $v$ is constant at every point $y$. The solution is given by the packet $u_0(x)$ moving with a velocity $c$. This can be also written as $u(t,x) = u_0(x - ct)$.

# Discrete Fourier transform

In order to talk about the numerical stability of the advection problem we'll introduce the discrete Fourier transform. Assume we have data $x_j = x(\Delta x j)$ on some interval and a function $f(x_j) = f_j$, where $j = 0, 1, ..., N-1$. We define the discrete Fourier transform as

$$\hat{f}_j = \sum_{n=0}^{N-1} e^{-2\pi i n j/N} f_n.$$

In order to talk about the numerical stability of the advection problem we'll introduce the discrete Fourier transform. Assume we have data $x_j = x(\Delta x j)$ on some interval and a function $f(x_j) = f_j$, where $j = 0, 1, ..., N-1$. We define the discrete Fourier transform as

$$\hat{f}_j = \sum_{n=0}^{N-1} e^{-2\pi i n j/N} f_n.$$

The inverse transform is given by

$$f_j = \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i n j/N} \hat{f}_n.$$

# Discrete Fourier transform

In order to talk about the numerical stability of the advection problem we'll introduce the discrete Fourier transform. Assume we have data $x_j = x(\Delta x j)$ on some interval and a function $f(x_j) = f_j$, where $j = 0, 1, ..., N-1$. We define the discrete Fourier transform as

$$\hat{f}_j = \sum_{n=0}^{N-1} e^{-2\pi i n j / N} f_n.$$

The inverse transform is given by

$$f_j = \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i n j / N} \hat{f}_n.$$

We can show that this works by inserting the definition for $\hat{f}$:

$$\frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi i n j / N} \sum_{k=0}^{N-1} e^{-2\pi i n k / N} f_k.$$

Changes the order of the summation gives

$$\frac{1}{N}\sum_{k=0}^{N-1} f_k \sum_{n=0}^{N-1} e^{2\pi i n(j-k)/N}.$$

Changes the order of the summation gives

$$\frac{1}{N}\sum_{k=0}^{N-1} f_k \sum_{n=0}^{N-1} e^{2\pi in(j-k)/N}.$$

We notice that $j-k$ is an integer. The latter sum is a geometric series so we have

$$\sum_{n=0}^{N-1} e^{2\pi in(j-k)/N} = \frac{1-e^{2\pi iN(j-k)/N}}{1-e^{2\pi i(j-k)/N}}.$$

Changes the order of the summation gives

$$\frac{1}{N}\sum_{k=0}^{N-1} f_k \sum_{n=0}^{N-1} e^{2\pi i n(j-k)/N}.$$

We notice that $j-k$ is an integer. The latter sum is a geometric series so we have

$$\sum_{n=0}^{N-1} e^{2\pi i n(j-k)/N} = \frac{1-e^{2\pi i N(j-k)/N}}{1-e^{2\pi i(j-k)/N}}.$$

The enumerator gives $1-\exp(2\pi i(j-k)) = 0$ if $j-k \neq 0$. If $j-k=0$ this will be a sum of ones giving $N$. Altogether we have

$$\sum_{n=0}^{N-1} e^{2\pi i n(j-k)/N} = N\delta_j^k.$$

Changes the order of the summation gives

$$\frac{1}{N}\sum_{k=0}^{N-1} f_k \sum_{n=0}^{N-1} e^{2\pi in(j-k)/N}.$$

We notice that $j-k$ is an integer. The latter sum is a geometric series so we have

$$\sum_{n=0}^{N-1} e^{2\pi in(j-k)/N} = \frac{1-e^{2\pi iN(j-k)/N}}{1-e^{2\pi i(j-k)/N}}.$$

The enumerator gives $1-\exp(2\pi i(j-k)) = 0$ if $j-k \neq 0$. If $j-k=0$ this will be a sum of ones giving $N$. Altogether we have

$$\sum_{n=0}^{N-1} e^{2\pi in(j-k)/N} = N\delta_j^k.$$

Now

$$\sum_{k=0}^{N-1} f_k\delta_j^k = f_j$$

just as we want.

The discrete Fourier transform is one of the most useful numerical tools. There is an algorithm called the fast Fourier transform (FFT) that is readily implemented in all major programming languages and can be used to calculate the discrete Fourier transform efficiently. This is especially useful for periodic data.

The discrete Fourier transform is one of the most useful numerical tools. There is an algorithm called the fast Fourier transform (FFT) that is readily implemented in all major programming languages and can be used to calculate the discrete Fourier transform efficiently. This is especially useful for periodic data.

Note: the FFT can be also used to transform the functions into cosine or sine bases for solving e.g. Dirichlet problems.

# Back to the advection problem

The advection problem we introduced was solved almost trivially but it turns out that it is a very hard problem for numerical treatment.

# Back to the advection problem

The advection problem we introduced was solved almost trivially but it turns out that it is a very hard problem for numerical treatment.

Let's assume that we solve the problem for periodic boundaries. The analytical solution is basically the same: the initial data $u_0(x)$ is moved forward (or backward) with a constant speed $c$. Let's write an implicit finite difference scheme

$$\frac{u_j^{(n+1)} - u_j^{(n)}}{s} = -c\frac{u_{j+1}^{(n)} - u_j^{(n)}}{h}.$$

# Back to the advection problem

The advection problem we introduced was solved almost trivially but it turns out that it is a very hard problem for numerical treatment.

Let's assume that we solve the problem for periodic boundaries. The analytical solution is basically the same: the initial data $u_0(x)$ is moved forward (or backward) with a constant speed $c$. Let's write an implicit finite difference scheme

$$\frac{u_j^{(n+1)} - u_j^{(n)}}{s} = -c\frac{u_{j+1}^{(n)} - u_j^{(n)}}{h}.$$

Shifting things around a bit gives

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

where $\sigma = cs/h$.

# Back to the advection problem

The advection problem we introduced was solved almost trivially but it turns out that it is a very hard problem for numerical treatment.

Let's assume that we solve the problem for periodic boundaries. The analytical solution is basically the same: the initial data $u_0(x)$ is moved forward (or backward) with a constant speed $c$. Let's write an implicit finite difference scheme

$$\frac{u_j^{(n+1)} - u_j^{(n)}}{s} = -c\frac{u_{j+1}^{(n)} - u_j^{(n)}}{h}.$$

Shifting things around a bit gives

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

where $\sigma = cs/h$.

Next we will take the discrete Fourier transform.

The discrete Fourier transform is a linear transformation but in order to continue we need the discrete Fourier transform of $u_{j+1}^{(n)}$. We have

$$u_{j+1}^{(n)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi i k(j+1)/N}\hat{u}_k^{(n)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi i kj/N} e^{2\pi i k/N}\hat{u}_k^{(n)}.$$

The discrete Fourier transform is a linear transformation but in order to continue we need the discrete Fourier transform of $u_{j+1}^{(n)}$. We have

$$u_{j+1}^{(n)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi ik(j+1)/N}\hat{u}_k^{(n)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi ikj/N}e^{2\pi ik/N}\hat{u}_k^{(n)}.$$

We see that the discrete Fourier transform is

$$\hat{u}_{k+1}^{(n)} = e^{2\pi ik/N}\hat{u}_k^{(n)}.$$

The discrete Fourier transform is a linear transformation but in order to continue we need the discrete Fourier transform of $u_{j+1}^{(n)}$. We have

$$u_{j+1}^{(n)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi ik(j+1)/N}\hat{u}_k^{(n)} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi ikj/N}e^{2\pi ik/N}\hat{u}_k^{(n)}.$$

We see that the discrete Fourier transform is

$$\hat{u}_{k+1}^{(n)} = e^{2\pi ik/N}\hat{u}_k^{(n)}.$$

Similarly,

$$\hat{u}_{k-1}^{(n)} = e^{-2\pi ik/N}\hat{u}_k^{(n)}.$$

Inserting this result gives

$$\hat{u}_k^{(n+1)} = (1+\sigma)\hat{u}_k^{(n)} - \sigma e^{2\pi ik/N}\hat{u}_k^{(n)} = (1+\sigma-\sigma e^{2\pi ik/N})\hat{u}_k^{(n)}.$$

Inserting this result gives

$$\hat{u}_k^{(n+1)} = (1+\sigma)\hat{u}_k^{(n)} - \sigma e^{2\pi ik/N}\hat{u}_k^{(n)} = (1+\sigma-\sigma e^{2\pi ik/N})\hat{u}_k^{(n)}.$$

This algorithm is unstable if $|1+\sigma-\sigma e^{2\pi ik/N}| > 1$. Let's calculate

$$|1+\sigma-\sigma e^{2\pi ik/N}|^2 = (1+\sigma)^2-(1+\sigma)\sigma(e^{2\pi ik/N}+e^{-2\pi ik/N})+\sigma^2 = 1+2\sigma+2\sigma^2-2(1+\sigma)\sigma\cos(2\pi ik/N).$$

Inserting this result gives

$$\hat{u}_k^{(n+1)} = (1+\sigma)\hat{u}_k^{(n)} - \sigma e^{2\pi ik/N}\hat{u}_k^{(n)} = (1+\sigma-\sigma e^{2\pi ik/N})\hat{u}_k^{(n)}.$$

This algorithm is unstable if $|1+\sigma-\sigma e^{2\pi ik/N}| > 1$. Let's calculate

$$|1+\sigma-\sigma e^{2\pi ik/N}|^2 = (1+\sigma)^2-(1+\sigma)\sigma(e^{2\pi ik/N}+e^{-2\pi ik/N})+\sigma^2 = 1+2\sigma+2\sigma^2-2(1+\sigma)\sigma\cos(2\pi ik/N).$$

In the worst possible case $\cos(2\pi ik/N) = -1$ (when $k \approx N/2$). We get

$$1+2\sigma+2\sigma^2+2(1+\sigma)\sigma \leq 1,$$

which is solved by

$$-1 \leq \sigma \leq 0 \Leftrightarrow -1 \leq \frac{sc}{h} \leq 0.$$

Inserting this result gives

$$\hat{u}_k^{(n+1)} = (1+\sigma)\hat{u}_k^{(n)} - \sigma e^{2\pi i k/N}\hat{u}_k^{(n)} = (1+\sigma-\sigma e^{2\pi i k/N})\hat{u}_k^{(n)}.$$

This algorithm is unstable if $|1+\sigma-\sigma e^{2\pi i k/N}| > 1$. Let's calculate

$$|1+\sigma-\sigma e^{2\pi i k/N}|^2 = (1+\sigma)^2 - (1+\sigma)\sigma(e^{2\pi i k/N}+e^{-2\pi i k/N})+\sigma^2 = 1+2\sigma+2\sigma^2-2(1+\sigma)\sigma\cos(2\pi i k/N).$$

In the worst possible case $\cos(2\pi i k/N) = -1$ (when $k \approx N/2$). We get

$$1+2\sigma+2\sigma^2+2(1+\sigma)\sigma \leq 1,$$

which is solved by

$$-1 \leq \sigma \leq 0 \Leftrightarrow -1 \leq \frac{sc}{h} \leq 0.$$

We notice that the system is unconditionally unstable if $c > 0$!

We can repeat the calculation for an explicit scheme

$$\frac{u_j^{(n+1)} - u_j^{(n)}}{s} = -c\frac{u_j^{(n)} - u_{j-1}^{(n)}}{h}$$

giving

$$\hat{u}_k^{(n+1)} = (1-\sigma)\hat{u}_k^{(n)} + \sigma e^{-2\pi ik/N}\hat{u}_k^{(n)} = (1-\sigma+\sigma e^{-2\pi ik/N})\hat{u}_k^{(n)}.$$

We can repeat the calculation for an explicit scheme

$$\frac{u_j^{(n+1)} - u_j^{(n)}}{s} = -c\frac{u_j^{(n)} - u_{j-1}^{(n)}}{h}$$

giving

$$\hat{u}_k^{(n+1)} = (1-\sigma)\hat{u}_k^{(n)} + \sigma e^{-2\pi i k/N}\hat{u}_k^{(n)} = (1-\sigma+\sigma e^{-2\pi i k/N})\hat{u}_k^{(n)}.$$

Going through the same calculation gives

$$0 \leq \sigma \leq 1.$$

We can repeat the calculation for an explicit scheme

$$\frac{u_j^{(n+1)} - u_j^{(n)}}{s} = -c\frac{u_j^{(n)} - u_{j-1}^{(n)}}{h}$$

giving

$$\hat{u}_k^{(n+1)} = (1-\sigma)\hat{u}_k^{(n)} + \sigma e^{-2\pi ik/N}\hat{u}_k^{(n)} = (1-\sigma+\sigma e^{-2\pi ik/N})\hat{u}_k^{(n)}.$$

Going through the same calculation gives

$$0 \leq \sigma \leq 1.$$

One would think that the central difference discretization would give better results but a simimlar analysis shows that it's actually *unconditionally unstable*. This is also true for doing the calculation in Fourier space. These really are hard equations.

# Upwind scheme

Let's generalize the problem a little bit and assume that $c$ depends on the spatial point. We can discretize the system using an implicit scheme wherever $c(x) < 0$ and an explicit scheme when $c(x) > 0$. This is called upwinding.

# Upwind scheme

Let's generalize the problem a little bit and assume that $c$ depends on the spatial point. We can discretize the system using an implicit scheme wherever $c(x) < 0$ and an explicit scheme when $c(x) > 0$. This is called upwinding.

This can be generalized to higher dimensions. The name comes from the fact that we define the finite difference stencil in the opposite direction of $c$ i.e. upwind. There are also higher order upwind schemes with larger finite difference stencils but we will not cover that here.

# Upwind scheme

Let's generalize the problem a little bit and assume that $c$ depends on the spatial point. We can discretize the system using an implicit scheme wherever $c(x) < 0$ and an explicit scheme when $c(x) > 0$. This is called upwinding.

This can be generalized to higher dimensions. The name comes from the fact that we define the finite difference stencil in the opposite direction of $c$ i.e. upwind. There are also higher order upwind schemes with larger finite difference stencils but we will not cover that here.

These methods are important for true advection equations where the time evolution of a field is given by

$$\frac{D\phi(t, \mathsf{x})}{\mathrm{d}t} = \partial_t \phi + \mathsf{v} \cdot \nabla \phi$$

with some velocity field $\mathsf{v}$.