NUMERICAL METHODS WEEK 9

ODES AND DECOMPOSITIONS

Learning outcomes:

- Practice using finite differences to solve end point differential equations.
- Look at decompositions of matrix eqautions and how they can be used to solve linear equations.

MATT WATKINS MWATKINS@LINCOLN.AC.UK

DIFFERENTIAL EQUATIONS

Some differential equations can be solved numerically as follows (more details next term).

We can approximate a function $y = f(x)$ on a grid of n points that are spaced by Δx.

We can approximate derivatives of y by finite differences

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}\Big|_i = \frac{(y_{i+1} - y_i)/\Delta x - (y_i - y_{i-1})/\Delta x}{\Delta x} = \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}$$

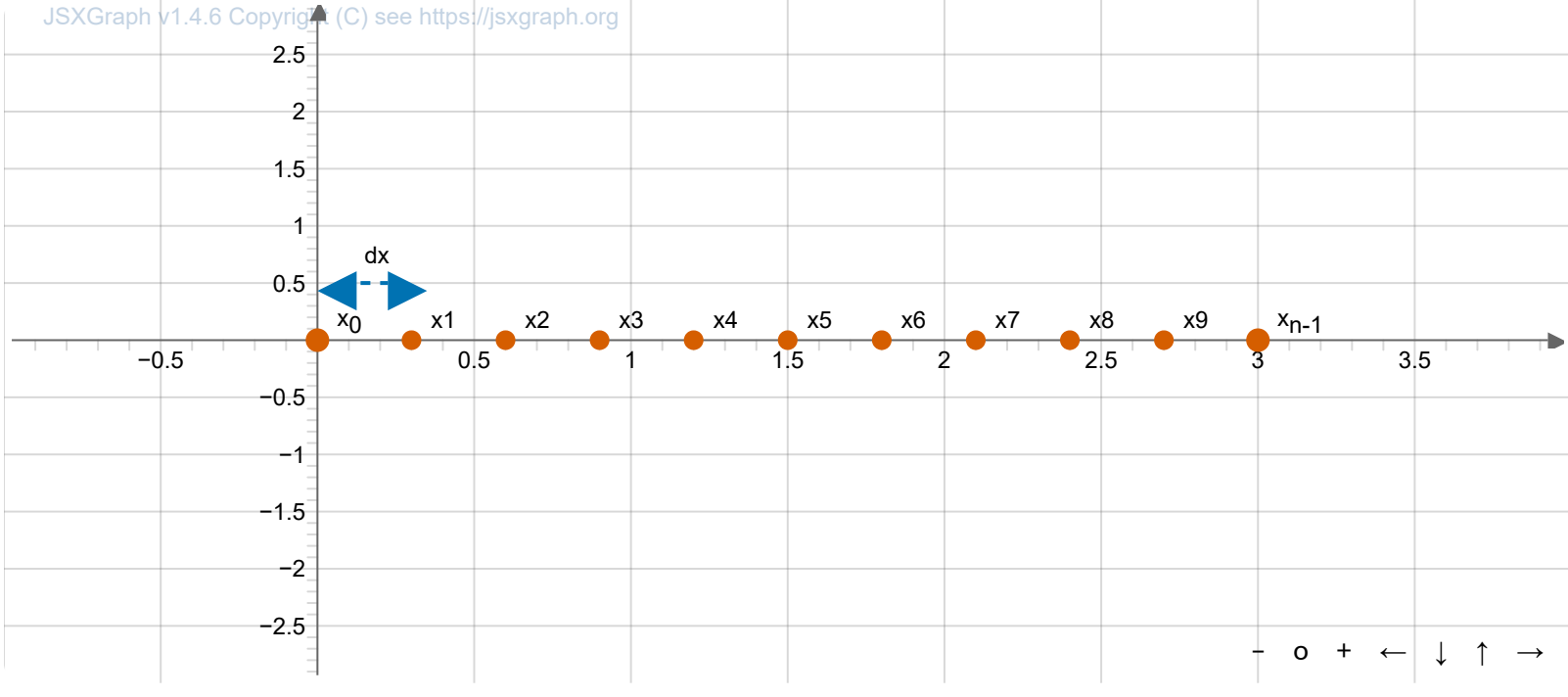
In matrix form we can write this as

$$\frac{1}{\Delta x^2}\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

FINITE DIFFERENCES

We change our continuous independent variable x into a discrete set of n equally spaced points

Add point



The gap between the points, Δx, is given by the range of the interval divided by the number of points -1, i.e.

$$x_i = x_{min} + \frac{i(x_{max} - x_{min})}{(n-1)}$$

DIFFERENTIAL EQUATIONS

Some differential equations can be solved numerically as follows (more details next term).

We can approximate a function $y(x) = f(x)$ on a grid of n points that are spaced by Δx.

We can approximate derivatives of $y(x_i)$ by finite differences

$$\frac{dy(x_i)}{dx} \approx \frac{(y_{i+\Delta x/2} - y_{i-\Delta x/2})}{\Delta x}$$

and then using this approximation again we get:

$$\frac{\mathrm{d}^2 y(x_i)}{\mathrm{d}x^2} \approx \left(\frac{dy(x_i + \Delta x/2)}{dx} - \frac{dy(x_i - \Delta x/2)}{dx} \right) / \Delta x = \frac{(y_{i+1} - y_i)/\Delta x - (y_i - y_{i-1})/\Delta x}{\Delta x}$$

$$= \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}$$

This is known as central differences - you can also work with forward or backward differences to get different approximations.

In matrix form we can write this as

$$\frac{1}{\Delta x^2}\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

If we multiply out the 2nd row we get

$$\frac{1}{\Delta x^2}(y_0 - 2y_1 + y_2) \approx \frac{d^2y(x_1)}{dx^2}$$

and the 3rd row gives

$$\frac{1}{\Delta x^2}(y_1 - 2y_2 + y_3) \approx \frac{d^2y(x_2)}{dx^2}$$

...

DIFFERENTIAL EQUATION

We can write

$$\frac{\mathrm{d}^2 y(x)}{\mathrm{d}x^2} = g(x)$$

where $g(x)$ contains everything that is not a function of y as

$$\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} -2y_L \\ g_1 \Delta x^2 \\ \vdots \\ g_i \Delta x^2 \\ \vdots \\ g_{n-2} \Delta x^2 \\ -2y_R \end{pmatrix}$$

where y_L and y_R are the values of y at the left and right boundaries.

This is a set of linear equations that can be solved using Gauss Elimination, or other methods such as LU decomposition (see later).

Note we multiplied through by Δx^2 to avoid dividing by a very small number.

$$\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} -2y_L \\ g_1\Delta x^2 \\ \vdots \\ g_i\Delta x^2 \\ \vdots \\ g_{n-2}\Delta x^2 \\ -2y_R \end{pmatrix}$$

Multiplying out the 2nd row and dividing by Δx^2 we get

$$\frac{1}{\Delta x^2}(y_0 - 2y_1 + y_2) \approx \frac{d^2y(x_1)}{dx^2} = g(x_1)$$

and for the third row

$$\frac{1}{\Delta x^2}(y_1 - 2y_2 + y_3) \approx \frac{d^2y(x_2)}{dx^2} = g(x_2)$$

the same for other rows.

Again, I multiplied through by Δx^2 because this will be small and can cause numerical problems if you divide numbers by it.

BOUNDARY CONDITIONS

Note that the first and last rows are different. They are completely decoupled from the others and just read

$$-2y_0 = -2y_L$$

and

$$-2y_{N-1} = -2y_R$$

i.e. that the boundary conditions are satisfied at the edges of the range we are solving over.

The convention of putting -2 on the LHS is only to keep the matrix looking pretty, you could put 1s and just yL and yR on the RHS.

The boundary values are fixed, but they effect the other points because they appear in other rows

$$\frac{1}{\Delta x^2}(y_L - 2y_1 + y_2) \approx \frac{d^2y(x_1)}{dx^2} = g(x_1)$$

EXAMPLE: $\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = x$

Find a solution for y in the interval [0,1] when:
$g(x) = x, y_L = 0.2, y_R = 1.5$ with $n = 10, 100, 500$

If we discretize into n points we get get our points $x_i = 0 + \frac{i(1-0)}{(n-1)}$.

We can write our equations at each point as:

$$\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} -2 \times 0.2 \\ x_1 \Delta x^2 \\ \vdots \\ x_i \Delta x^2 \\ \vdots \\ x_{n-2} \Delta x^2 \\ -2 \times 1.5 \end{pmatrix}$$

EXAMPLE $\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = g(x)$

Solve when $g(x_{n/2}) = 1/\Delta x$, all other $g_i = 0$ and $y_L = 0, y_R = 0$ in the interval $[-5, 5]$

If we discretize into n points we get get our points $x_i = -5 + \frac{i(5--5)}{(n-1)}$.

We can write our equations at each point as:

$$\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n/2} \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} -2 \times 0 \\ 0 \times \Delta x^2 \\ \vdots \\ \frac{1}{\Delta x}\Delta x^2 \\ \vdots \\ 0 \times \Delta x^2 \\ -2 \times 0 \end{pmatrix}$$

EXAMPLE $\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + ky = C$

Solve $\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + ky = g(x)$ when $g(x) = C, y_L = 5, y_R = 0$ in the interval $[-3, 3]$

To get our linear equation we must have $\mathbf{D(x)}y(x) = g(x)$, in this case the LHS must approximate $(\frac{\mathrm{d}^2}{\mathrm{d}x^2} + k)y$

$$\begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2+k\Delta x^2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2+k\Delta x^2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2+k\Delta x^2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix}$$

TASK SOLVE THE QUESTION IN THE WEEK 9 ONLINE TEST.

Note, you will need to think how to include the first derivative of y into the LHS too.

LU DECOMPOSITION

Again we want to solve $\mathbf{Ax - b} = 0$

Suppose we can rewrite this as

$$\begin{pmatrix} u_{00} & u_{01} & u_{02} \\ 0 & u_{11} & u_{12} \\ 0 & 0 & u_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} d_0 \\ d_1 \\ d_2 \end{pmatrix} = 0$$

which looks similar to Gauss elmination. In matrix notation

$$\mathbf{Ux - d = 0}$$

.

Now, assume there is a lower diagonal matrix $\mathbf{L}$ with '1's on the diagonal

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{10} & 1 & 0 \\ l_{20} & l_{21} & 1 \end{pmatrix}$$

that has the property that, premultiplying by $\mathbf{L}$ we get

$$\mathbf{LUx - Ld = Ax - b}$$

LU DECOMPOSITION

To ensure that

$$\mathbf{LUx} - \mathbf{Ld} = \mathbf{Ax} - \mathbf{b}$$

we require that

$$\mathbf{LU} = \mathbf{A}$$

and

$$\mathbf{Ld} = \mathbf{b}$$

GAUSSIAN ELIMINATION

TRIANGULARIZATION

When we do the Gauss elimination method, we actually find all the elements of $\mathbf{L}$, we just need to store them.
It is the inverse of the matrix we would need to multiply $\mathbf{b}$ by to get the correct RHS in the Gauss elmination method.

Let us take an initial augmented matrix, and record the values that we would scale the $\mathbf{b}$ matrix by if it were there:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

pivoting around row 0, we remove all entries below the diagonal entry in column 0, doing this we scaled the $\mathbf{b}$ matrix by the ratios shown on the right

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & a'_{21} & a'_{22} & a'_{23} \\ 0 & a'_{31} & a'_{32} & a'_{33} \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{10}}{a_{00}} & & & \\ \frac{a_{20}}{a_{00}} & & & \\ \frac{a_{30}}{a_{00}} & & & \end{pmatrix}$$

Matrix after pivoting around row 0

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & a'_{21} & a'_{22} & a'_{23} \\ 0 & a'_{31} & a'_{32} & a'_{33} \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{10}}{a_{00}} & & & \\ \frac{a_{20}}{a_{00}} & & & \\ \frac{a_{30}}{a_{00}} & & & \end{pmatrix}$$

Then pivoting around row 1 we remove elements below the diagonal in column 1, and subtract multiples of the 2nd row of $\mathbf{b}$ as shown in the right hand matrix

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & a''_{32} & a''_{33} \end{pmatrix} \qquad \begin{pmatrix} 1 & & \\ \frac{a_{10}}{a_{00}} & 1 & \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{10}} & \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{10}} & \end{pmatrix}$$

pivoting around row 2

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & 0 & a'''_{33} \end{pmatrix} \qquad \begin{pmatrix} 1 & & \\ \frac{a_{10}}{a_{00}} & 1 & \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{11}} & 1 \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{11}} & \frac{a''_{32}}{a''_{22}} \end{pmatrix}$$

we had got to

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & 0 & a'''_{33} \end{pmatrix} \qquad \begin{pmatrix} 1 & & & \\ \frac{a_{10}}{a_{00}} & 1 & & \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{11}} & 1 & \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{11}} & \frac{a''_{32}}{a''_{22}} & \end{pmatrix}$$

if we fill in the rest of the matrix on the right we have

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & 0 & a'''_{33} \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{10}}{a_{00}} & 1 & 0 & 0 \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{11}} & 1 & 0 \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{11}} & \frac{a''_{32}}{a''_{22}} & 1 \end{pmatrix}$$

We can show for a real system that this new matrix is $\mathbf{L}$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{10}}{a_{00}} & 1 & 0 & 0 \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{11}} & 1 & 0 \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{11}} & \frac{a''_{32}}{a''_{22}} & 1 \end{pmatrix} \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & 0 & a'''_{33} \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\mathbf{LU} = \mathbf{A}$$

and

$$\mathbf{Ld} = \mathbf{b}$$

```
void gauss_elim(std::ofstream &myout, double (&a)[rows][cols]) {
    double coeff;
    double x[rows];
    for (int i =0; i < rows-1; i++) {
        for (int j = i+1; j < rows; j++) {
            coeff = a[j][i]/a[i][i];
            for (int k = i; k < cols; k++) {
                a[j][k] -= a[i][k]*coeff;
            }
        }
        myout << "\n pivoting around row " << i << "\n\n";
        output_matrix(myout,a);
    }
}
```

highlight: hljs cs

The entries in $\mathbf{L}$ are just what I called `coeff` in the code

```
void gauss_elim(std::ofstream &myout, double (&a)[rows][cols]) {
    double coeff;
    double x[rows];
    for (int i =0; i < rows-1; i++) {
        for (int j = i+1; j < rows; j++) {
            coeff = a[j][i]/a[i][i];
      a[j][i] = coeff;
            for (int k = i+1; k < cols; k++) {
                a[j][k] -= a[i][k]*coeff;
            }
        }
    }
}
```

highlight: hljs cs

So only very minor changes in the Gauss elimination code are needed.
Note, I have stored the nondiagonal elements of $\mathbf{L}$ in what would be the zero elements of the row echelon matrix.

LU DECOMPOSITION

The major advantage of LU decompostion is we calculate $\mathbf{L}$ and $\mathbf{U}$ once, then we can easily find $\mathbf{x}$ for any $\mathbf{b}$

Having $\mathbf{L}$ we can solve $\mathbf{Ld} = \mathbf{b}$ for $\mathbf{d}$ by forward substitution

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{10}}{a_{00}} & 1 & 0 & 0 \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{11}} & 1 & 0 \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{11}} & \frac{a''_{32}}{a''_{22}} & 1 \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

i.e. multiplying out the equations starting from the top row.

Then, having constructed $\mathbf{d}$ for the given $\mathbf{b}$ we continue exactly like in Gauss Elimination using back substitution to find $\mathbf{x}$

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & 0 & a'''_{33} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

This is what the following call does in Eigen

```
#include
#include
using namespace std;
using namespace Eigen;
int main()
{
   Matrix3f A;
   Vector3f b;
   A << 1,2,3,  4,5,6,  7,8,10;
   b << 3, 3, 4;
   cout << "Here is the matrix A:\n" << A << endl;
   cout << "Here is the vector b:\n" << b << endl;
   Vector3f x = A.fullPivLu().solve(b);
   cout << "The solution is:\n" << x << endl;
}
```

highlight: hljs cpp

LU DECOMPOSITION

We can use LU decomposition to find the inverse of a matrix, by setting b to the columns of the identity matrix, I.

For example we get the first column of $\mathbf{A}^{-1}$ by using $\mathbf{b}^T = (1, 0, 0, 0)$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{a_{10}}{a_{00}} & 1 & 0 & 0 \\ \frac{a_{20}}{a_{00}} & \frac{a'_{21}}{a'_{11}} & 1 & 0 \\ \frac{a_{30}}{a_{00}} & \frac{a'_{31}}{a'_{11}} & \frac{a''_{32}}{a''_{22}} & 1 \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

i.e. multiplying out the equations starting from the top row.

Then, having constructed $\mathbf{d}$ for the given $\mathbf{b}$ we continue exactly like in Gauss Elimination using back substitution to find $\mathbf{x}$

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a'_{11} & a'_{12} & a'_{13} \\ 0 & 0 & a''_{22} & a''_{23} \\ 0 & 0 & 0 & a'''_{33} \end{pmatrix} \begin{pmatrix} A^{-1}_{00} \\ A^{-1}_{10} \\ A^{-1}_{20} \\ A^{-1}_{30} \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

The we build the other columns of $\mathbf{A}^{-1}$ from the other unit vectors.

If you want to understand the LU decomposition clearly, adapt a gauss-elim type code to also perform LU decomposition

- Find the inverse of the matrix using LU decomposition

$$\begin{pmatrix} 2 & 2 & 4 & -2 \\ 1 & 3 & 2 & 4 \\ 3 & 1 & 3 & 1 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$

Check your solution is correct by comparison to Gauss-Jordan or Eigen / numpy libraries. Note you can also get the determinant from LU decomposition.

QR DECOMPOSITION

WHY ANOTHER DECOMPOSITION???

This decomposition is part of the 'QR' algorithm to find the eigenvalues and vectors of a matrix.

Any real square matrix A may be decomposed as

$$\mathbf{A} = \mathbf{QR},$$

where

- $\mathbf{Q}$ is an orthogonal matrix (its columns are orthogonal unit vectors meaning $\mathbf{Q}^t\mathbf{Q} = I$)
- $\mathbf{R}$ is an upper triangular matrix (also called right triangular matrix).

If $\mathbf{A}$ is invertible, then the factorization is unique if we require the diagonal elements of $\mathbf{R}$ to be positive.

QR DECOMPOSITION

GRAM-SCHMIDT ORTHOGONALISATION.

You may remember this from linear algebra - we build an orthogonal basis by sucessively projecting out previous basis vectors

We can scan across the columns of our matrix $\mathbf{A}$ and remove the components of any previous column by subtracting the dot product.

Take an initial matrix $\mathbf{A}$

$$\mathbf{A} = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$$

First we'll make an orthogonal matrix $\mathbf{U}$, we'll keep the first column.

$$\mathbf{U} = \begin{pmatrix} 12 & ? & ? \\ 6 & ? & ? \\ -4 & ? & ? \end{pmatrix}$$

To get the 2nd column orthogonal, we subtract the projection of the second column onto the first column from the 2nd column.

To get the 2nd column orthogonal, we subtract the projection of the second column onto the first column from the 2nd column.

$$\mathbf{U} = \begin{pmatrix} 12 & -51 - \text{proj}_{col_1} & ? \\ 6 & 167 - \text{proj}_{col_1} & ? \\ -4 & 24 - \text{proj}_{col_1} & ? \end{pmatrix}$$

where the projection onto the first column is given by

$$\begin{aligned} proj_{col_1} &= \left\langle \mathbf{col}_2 \cdot \frac{\mathbf{col}_1}{||col_1||} \right\rangle \frac{\mathbf{col\_1}}{||col_1||} \\ &= \left\langle (-51, 167, 24) \cdot \begin{pmatrix} 12 \\ 6 \\ 4 \end{pmatrix} \right\rangle (12, 6, 4) \frac{1}{(12^2 + 6^2 + 4^2)} \\ &= \frac{294}{196}(12, 6, 4) = (18, 9, -6) \end{aligned}$$

so

$$\mathbf{U} = \begin{pmatrix} 12 & -51 - 18 & ? \\ 6 & 167 - 9 & ? \\ -4 & 24 - -6 & ? \end{pmatrix} = \begin{pmatrix} 12 & -69 & ? \\ 6 & 158 & ? \\ -4 & 30 & ? \end{pmatrix}$$

Two get the 3rd column orthogonal, we subtract the projection of the third column onto the first two columns.

$$\begin{aligned}\mathbf{U} &= \begin{pmatrix} 12 & -69 & 4-(\text{proj}_{col_1}+\text{proj}_{col_2}) \\ 6 & 158 & -68-(\text{proj}_{col_1}+\text{proj}_{col_2}) \\ -4 & 30 & -41-(\text{proj}_{col_1}+\text{proj}_{col_2}) \end{pmatrix} \\ &= \begin{pmatrix} 12 & -69 & -58/5 \\ 6 & 158 & 6/5 \\ -4 & 30 & -33 \end{pmatrix}\end{aligned}$$

Now we make each column a unit vector to get the orthogonal matrix $\mathbf{Q}$

$$\mathbf{Q} = \begin{pmatrix} \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} & \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} & \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \end{pmatrix} = \begin{pmatrix} 6/7 & -69/175 & -58/175 \\ 3/7 & 158/175 & 6/175 \\ -2/7 & 6/35 & -33/35 \end{pmatrix}.$$

Remember, we wanted

$$\mathbf{A} = \mathbf{QR},$$

if we premultiply by $\mathbf{Q}^T$ we get

$$\mathbf{Q}^T\mathbf{A} = \mathbf{Q}^T\mathbf{QR},$$

but for an orthogonal matrix $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ so we have

$$\mathbf{Q}^T\mathbf{A} = \mathbf{R},$$

and we can find $\mathbf{R}$ my mulitplying our original matrix $\mathbf{A}$ and the (transposed) orthogonal matrix we just found.

$$R = Q^T A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & 35 \end{pmatrix}.$$

QR ALGORITHM

It can be shown that the sequence

$$\mathbf{A}^0 = \mathbf{A}$$
$$\mathbf{A}^{k+1} = \mathbf{R}^k\mathbf{Q}^k$$

converges to a upper triangular matrix with the eigenvalues of $\mathbf{A}$ as its diagonal entries.

In the sequence $\mathbf{Q}^k$ and $\mathbf{R}^k$ are formed by the QR decomposition of $\mathbf{A}^k$

$$\mathbf{A}^k = \mathbf{Q}^k\mathbf{R}^k$$

This algorithm needs modifications to be generally efficient, but gives the general shape of real eigenvalue finding routines.

If we have an eigenvalue we can find the corresponding eigenvector using Gauss Elimination.

Find the eigenvalues of the following matrices using the QR decomposition code on Bb (or write your own of course):

$$\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 \\ -1 & 0 & 3 & 1 \\ -1 & 0 & 1 & 4 \end{pmatrix}$$

.

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1/2 \\ 0 & 0 & 1 & 1/4 \\ 1 & 1/2 & 1/4 & 1 \end{pmatrix}$$

.

Check that your eigenvalues are eigenvalues by running Gauss elimination on the equation $(\mathbf{A} - \lambda_i \mathbf{I})\mathbf{x} = 0$. The augmented matrix should be of reduced rank, and you can deduce the eigenvectors from the output.

Check that an eigenvector you found is an eigenvector with eigenvalue λ_i by direct multiplication.

Modify the code to output the condition number of a matrix. The condition number is the ratio of the largest to smallest eigenvalue. This controls how numerically stable inversion of the matrix is.

SUMMARY AND FURTHER READING

You should be reading additional material to provide a solid background to what we do in class

All the textbooks contain sections on solving linear equations, for instance chapter 2 of Numerical Recipes (http://www.nrbook.com/a/bookcpdf.php).

SNAKE

Use the arrow keys

start game