NUMERICAL METHODS WEEK 6

RECAP

Learning outcomes:

- Practice using some of the methods we have covered so far.
- See how to easily and tidyly reuse code.

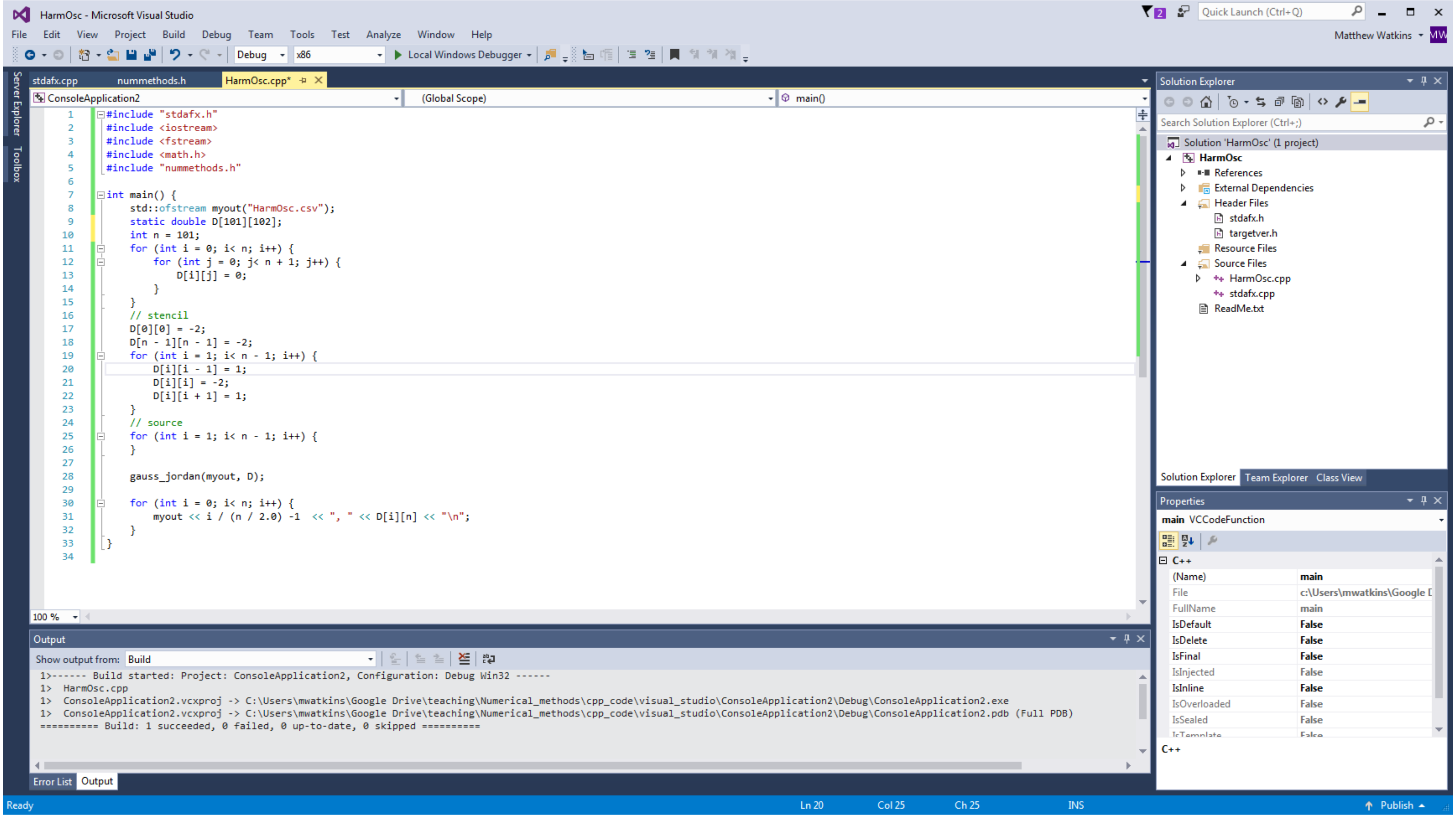
MATT WATKINS MWATKINS@LINCOLN.AC.UK

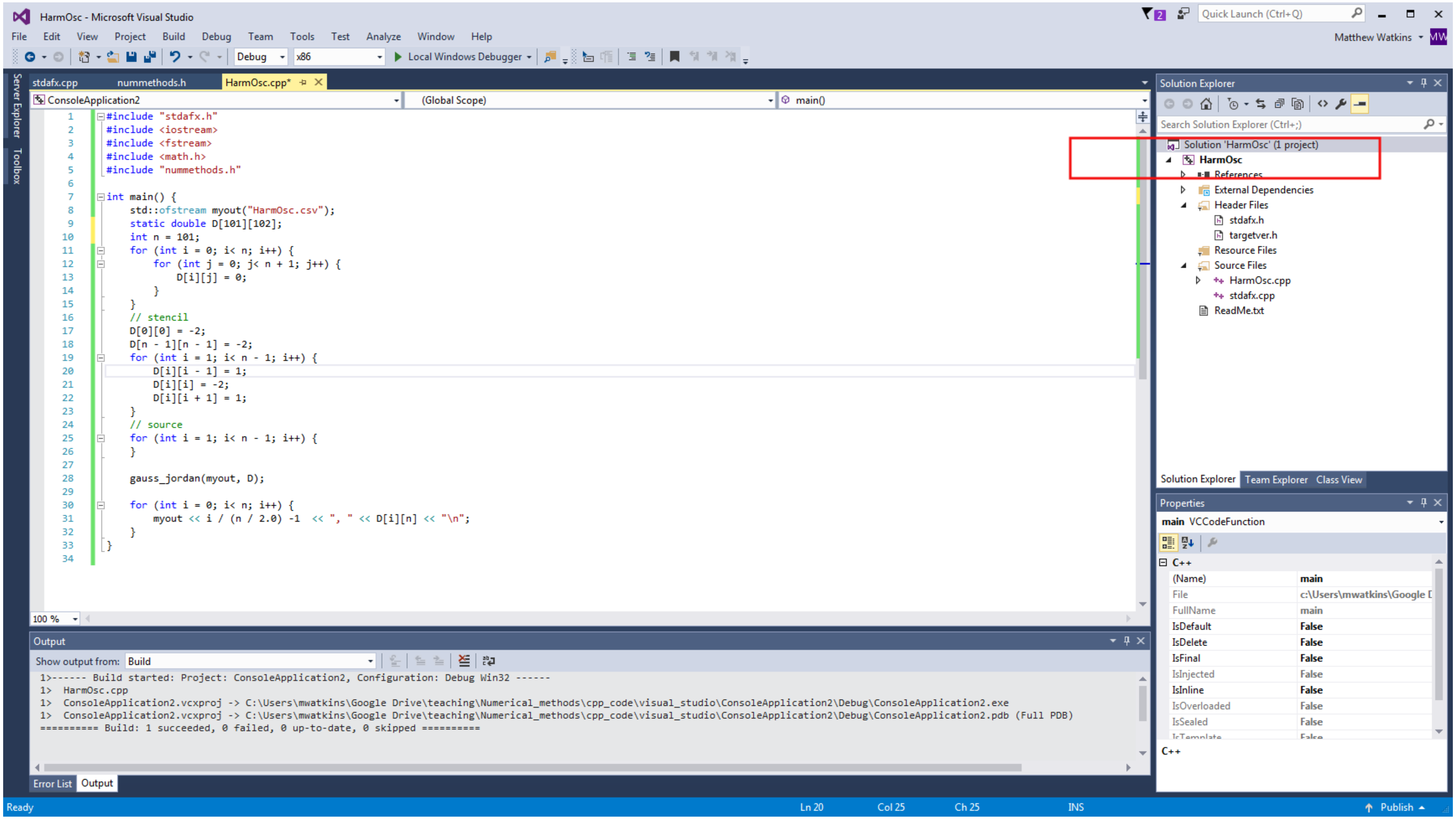
REUSING CODE

You may dislike having lots of code / routines copied around everytime we reuse something

The solution is to **include** other files or, generally, libraries.

We can make our own library file by copying functions other than `main` into a file called `'my_library.h'`

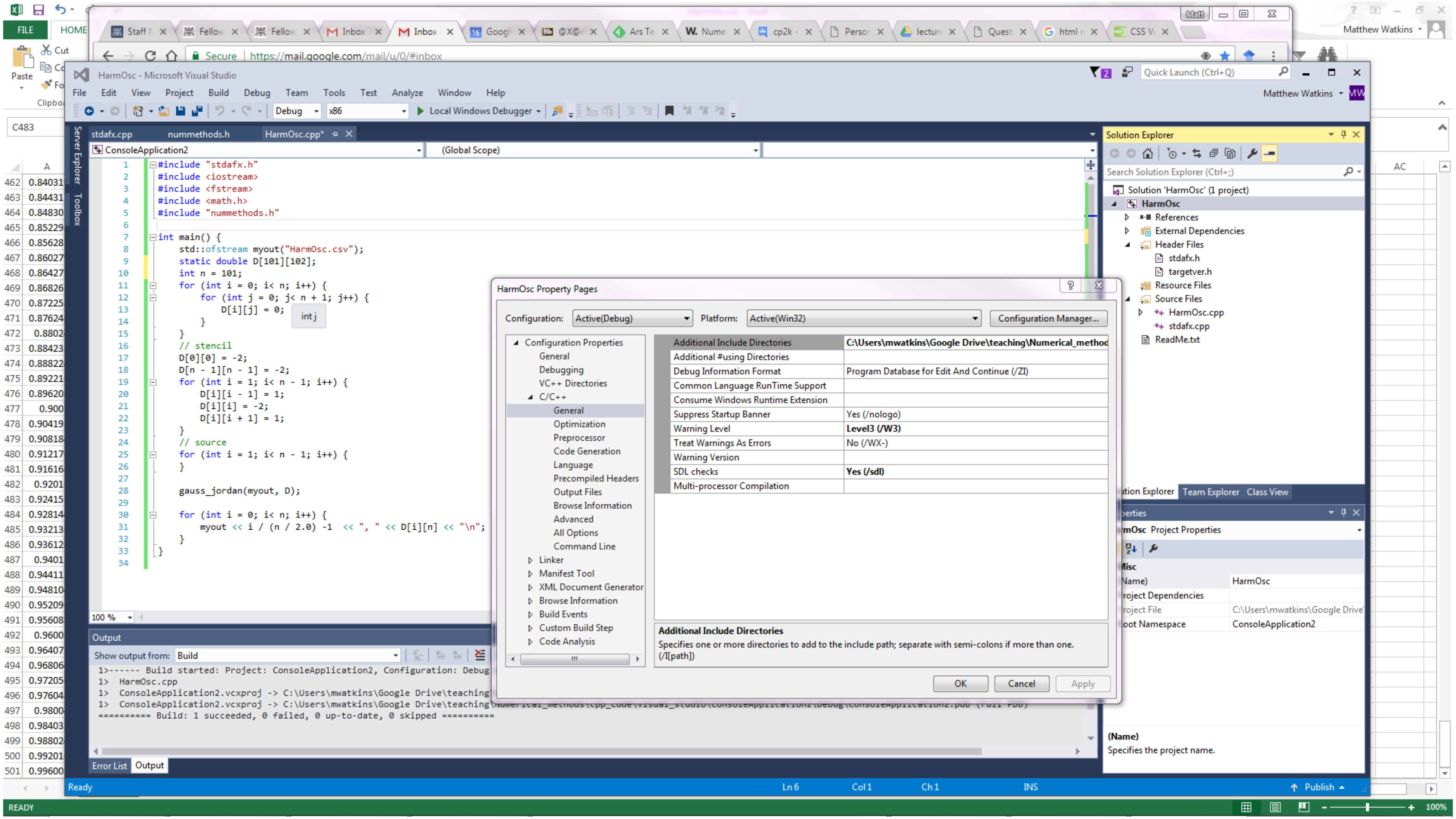

HarmOsc - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Matthew Watkins
Debug
x86
Local Windows Debugger
stdafx.cpp
nummethods.h
HarmOsc.cpp*
ConsoleApplication2
(Global Scope)
main()
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <math.h>
#include "nummethods.h"
int main() {
std::ofstream myout("HarmOsc.csv");
static double D[101][102];
int n = 101;
for (int i = 0; i< n; i++) {
for (int j = 0; j< n + 1; j++) {
D[i][j] = 0;
}
}
// stencil
D[0][0] = -2;
D[n - 1][n - 1] = -2;
for (int i = 1; i< n - 1; i++) {
D[i][i - 1] = 1;
D[i][i] = -2;
D[i][i + 1] = 1;
}
// source
for (int i = 1; i< n - 1; i++) {
}
gauss_jordan(myout, D);
for (int i = 0; i< n; i++) {
myout << i / (n / 2.0) -1 << ", " << D[i][n] << "\n";
}
}
100 %
Output
Show output from: Build
1>------ Build started: Project: ConsoleApplication2, Configuration: Debug Win32 ------
1> HarmOsc.cpp
1> ConsoleApplication2.vcxproj -> C:\Users\mwatkins\Google Drive\teaching\Numerical_methods\cpp_code\visual_studio\ConsoleApplication2\Debug\ConsoleApplication2.exe
1> ConsoleApplication2.vcxproj -> C:\Users\mwatkins\Google Drive\teaching\Numerical_methods\cpp_code\visual_studio\ConsoleApplication2\Debug\ConsoleApplication2.pdb (Full PDB)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'HarmOsc' (1 project)
HarmOsc
References
External Dependencies
Header Files
stdafx.h
targetver.h
Resource Files
Source Files
HarmOsc.cpp
stdafx.cpp
ReadMe.txt
Solution Explorer Team Explorer Class View
Properties
main VCCodeFunction
C++
(Name) main
File c:\Users\mwatkins\Google D
FullName main
IsDefault False
IsDelete False
IsFinal False
IsInjected False
IsInline False
IsOverloaded False
IsSealed False
C++
Ready
Ln 20
Col 25
Ch 25
INS
Publish



This is my example project. Right click on the **bolded** project name in the pane to right.

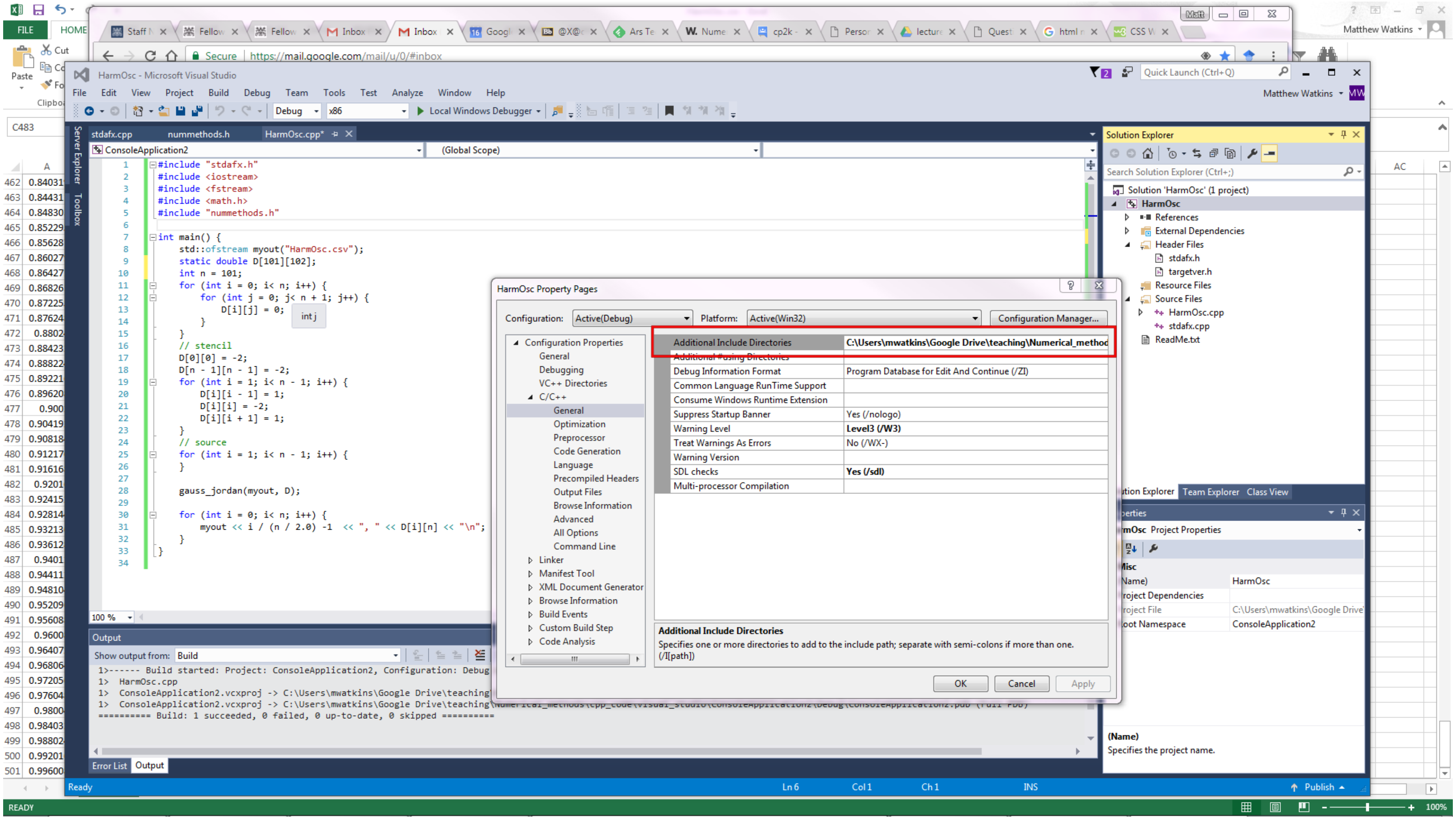
Then select `properties`.

A new window should have appeared, like this:

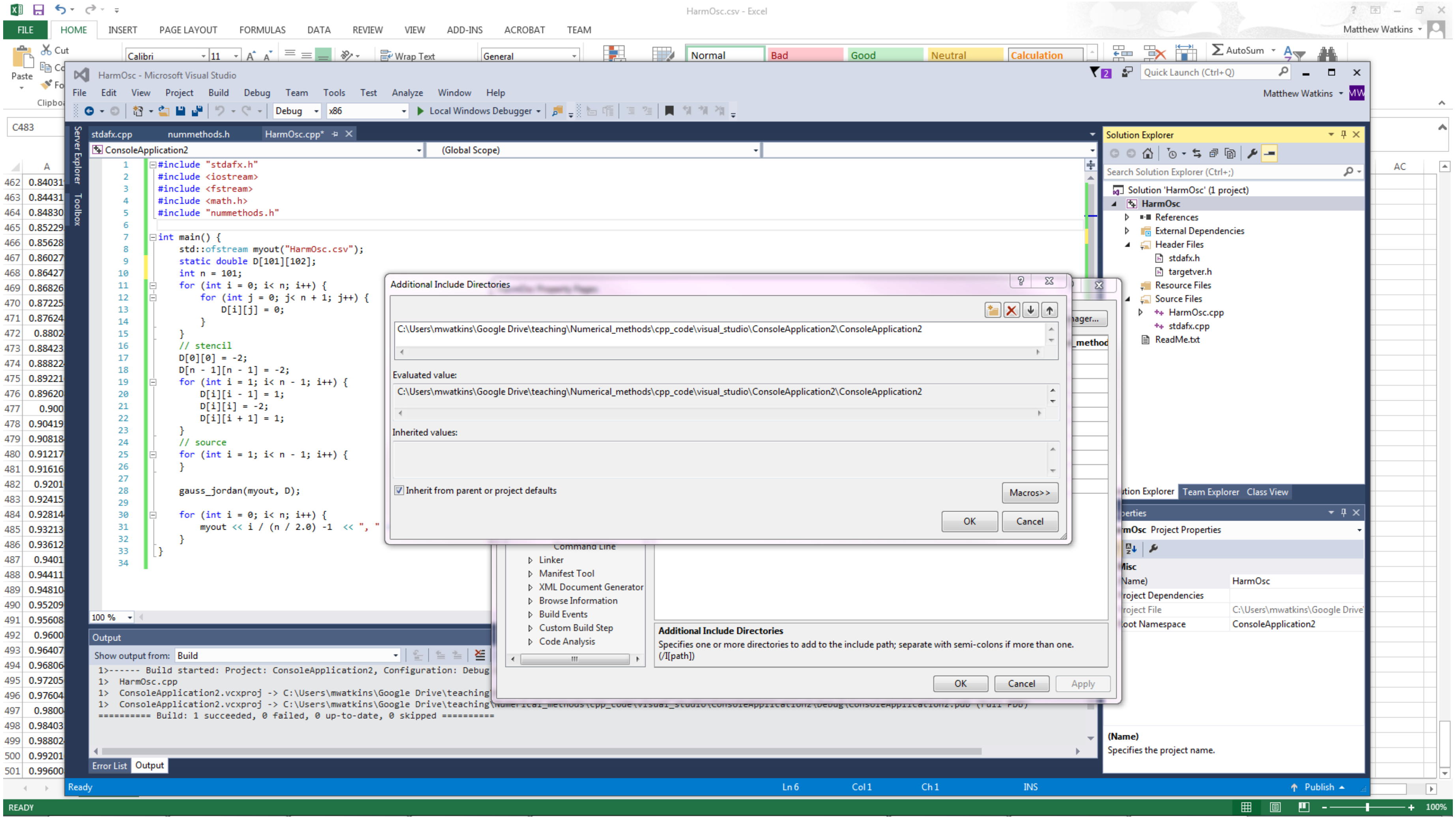


Open the `C/C++` then the `General` tab

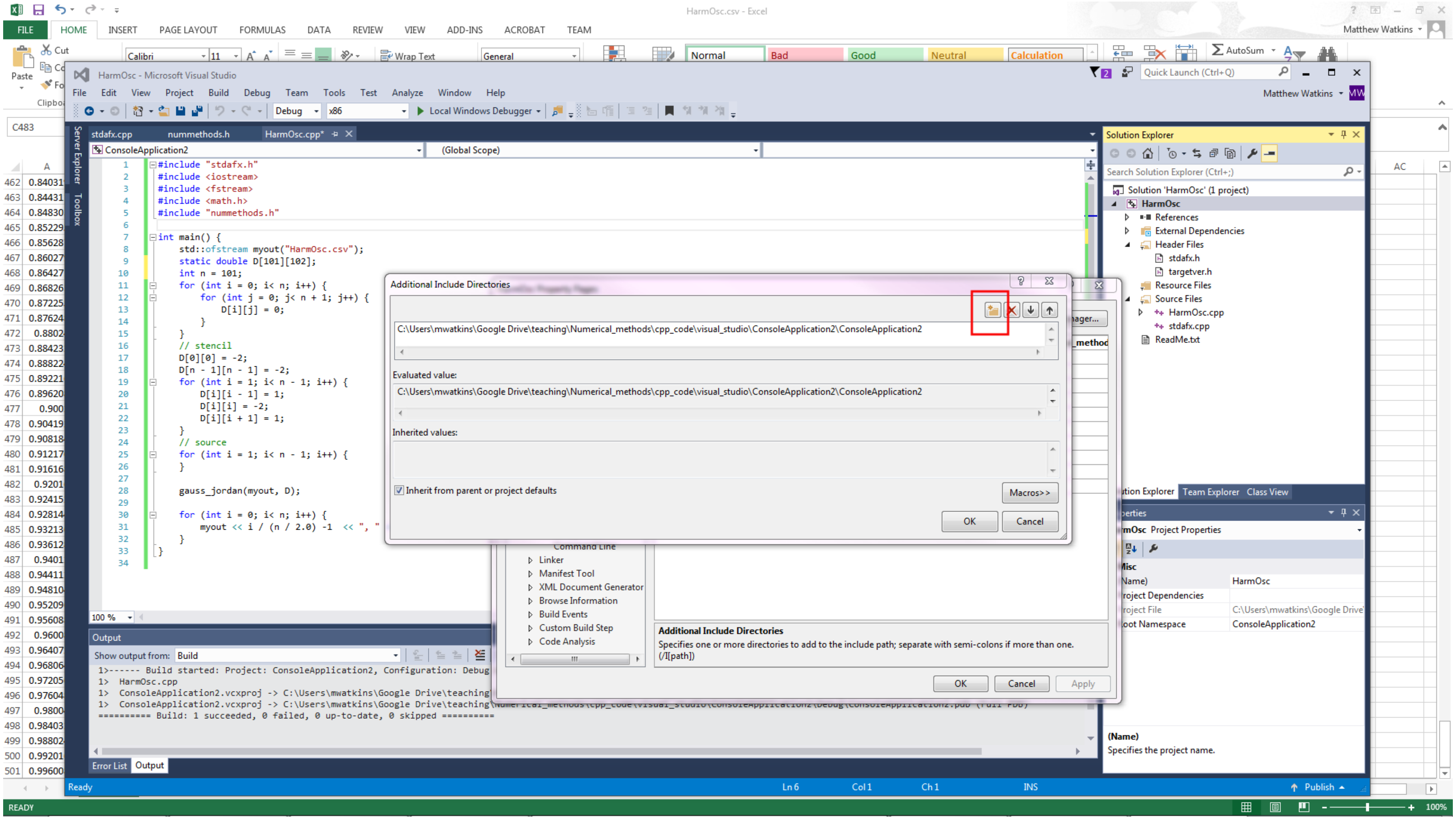
Then click on drop down button in the second column of the the 'Additional Include Directories' row and select edit

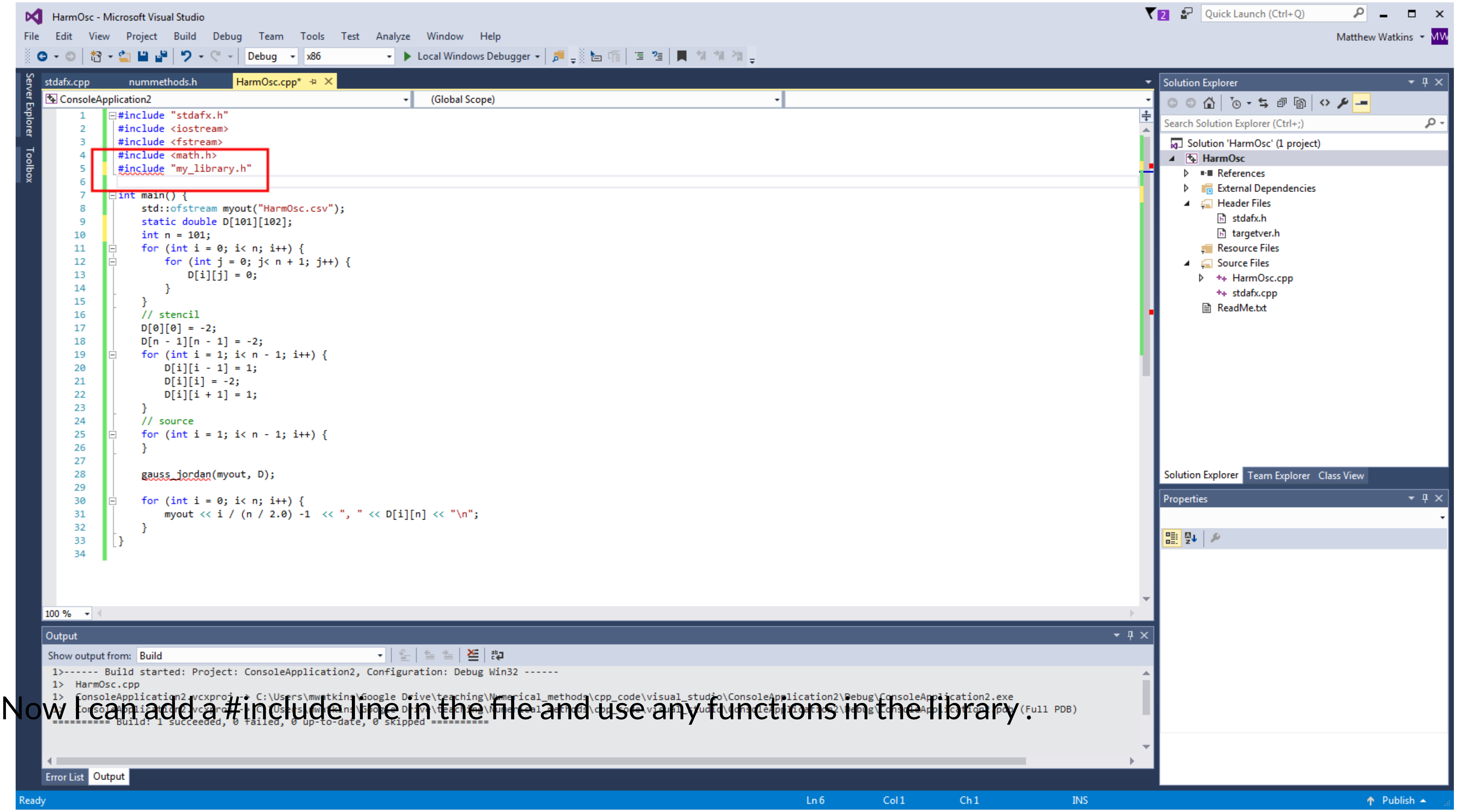


You should see something like this:



Click on the folder icon, and select the directory where you saved your 'my_library.h' file in


HarmOsc.csv - Excel
HarmOsc - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug x86 Local Windows Debugger
stdafx.cpp nummethods.h HarmOsc.cpp*
ConsoleApplication2 (Global Scope)
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <math.h>
#include "nummethods.h"
int main() {
std::ofstream myout("HarmOsc.csv");
static double D[101][102];
int n = 101;
for (int i = 0; i< n; i++) {
for (int j = 0; j< n + 1; j++) {
D[i][j] = 0;
}
}
// stencil
D[0][0] = -2;
D[n - 1][n - 1] = -2;
for (int i = 1; i< n - 1; i++) {
D[i][i - 1] = 1;
D[i][i] = -2;
D[i][i + 1] = 1;
}
// source
for (int i = 1; i< n - 1; i++) {
}
gauss_jordan(myout, D);
for (int i = 0; i< n; i++) {
myout << i / (n / 2.0) -1 << ", "
}
}
Additional Include Directories
C:\Users\mwatkins\Google Drive\teaching\Numerical_methods\cpp_code\visual_studio\ConsoleApplication2\ConsoleApplication2
Evaluated value:
C:\Users\mwatkins\Google Drive\teaching\Numerical_methods\cpp_code\visual_studio\ConsoleApplication2\ConsoleApplication2
Inherited values:
Inherit from parent or project defaults
Macros>>
OK
Cancel
Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])
OK Cancel Apply
Solution Explorer
Solution 'HarmOsc' (1 project)
HarmOsc
References
External Dependencies
Header Files
stdafx.h
targetver.h
Resource Files
Source Files
HarmOsc.cpp
stdafx.cpp
ReadMe.txt
HarmOsc Project Properties
(Name) HarmOsc
Project Dependencies
Project File C:\Users\mwatkins\Google Drive
Root Namespace ConsoleApplication2
(Name)
Specifies the project name.
Output
Show output from: Build
1>------ Build started: Project: ConsoleApplication2, Configuration: Debug
1> HarmOsc.cpp
1> ConsoleApplication2.vcxproj -> C:\Users\mwatkins\Google Drive\teaching
1> ConsoleApplication2.vcxproj -> C:\Users\mwatkins\Google Drive\teaching
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
Error List Output
Ready Ln 6 Col 1 Ch 1 INS
READY



Now I can add a #include line in the file and use any functions in the library.

POWER METHOD

This is a way of finding the eigenvector with the largest eigenvector

- Start with a guess $\mathbf{x}_0$ at the eigenvector (it must not be orthogonal to the true eigenvector).
- Update the eigenvector $\mathbf{x}_i = \frac{\mathbf{A}\mathbf{x}_{n-1}}{||\mathbf{A}\mathbf{x}_{n-1}||}$

Find the largest eigenvector of

$$\mathbf{A} = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 \\ -1 & 0 & 3 & 1 \\ -1 & 0 & 1 & 4 \end{pmatrix}.$$

using the power method.

We basically used this when studying Markov Chains in year 1.

GENERAL LINEAR LEAST SQUARES

Simple linear, polynomial and multiple linear regression can be generalised to the following linear least-squares model

$$y_i(a_0, a_1, a_2, a_3 \ldots a_n; x_0, x_1, x_2, x_3 \ldots x_n) = a_0 z_0 + a_1 z_1 + a_2 z_2 + \cdots + a_m z_m + e$$
are now not just coordinates x_i but some *predefined* functions of those positions $z_0(x_0, x_1, x_2, x_3 \ldots x_n), z_1(x_0, x_1, x_2, x_3 \ldots x_n), \ldots, z_m(x_0, x_1, x_2, x_3 \ldots x_n)$ are $m+1$ $\mathbf{basis\ functions}$.

The linear refers to the parameters $a_0, a_1, \ldots, a_m$, the zs can be highly non-linear

For instance.
$$y(a_0, a_1, a_2; x_0) = a_0 + a_1 \cos(\omega x_0) + a_2 \sin(\omega x_0)$$
fits this model with $z_0 = 1, z_1 = \cos(\omega x_0)$ and $z_2 = \sin(\omega x_0)$. Where ω is a single independent variable and ω is a predefined constant.

I'll supress the functional dependence notation as it gets too much. Remember the zs are predefined - only the as are optimized.

GENERAL LINEAR LEAST SQUARES EXAMPLE

Let's redo the quadratic fit example using the general linear least squares formalism. The general expression
$y_i(a_0, a_1, a_2, a_3 \ldots a_n; x_0, x_1, x_2, x_3 \ldots x_n) = a_0 z_0(x_i) + a_1 z_1(x_i) + a_2 z_2(x_i) + \cdots + a_m z_m(x_i) +$

becomes

$$y_i(a_0, a_1, a_2; x_0, x_1, x_2, x_3 \ldots x_n) = a_0 z_0(x_i) + a_1 z_1(x_i) + a_2 z_2(x_i) + e$$

as we only have three parameters.

Comparing to the formula for a quadratic fit

$$y_i(a_0, a_1, a_2; x_0, x_1, x_2, x_3 \ldots x_n) = a_0 + a_1 x_i + a_2 x_i^2 + e$$

we see that $z_0(x_i) = 1, z_1(x_i) = x_i$ and $z_2(x_i) = x_i^2$

We write the set of equations for the all the $n+1$ data points in matrix form

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} z_0(x_0) & z_1(x_0) & \cdots & z_m(x_0) \\ z_0(x_1) & z_1(x_1) & \cdots & z_m(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ z_0(x_n) & z_1(x_n) & \cdots & z_m(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} e(x_0) \\ e(x_1) \\ \vdots \\ e(x_n) \end{bmatrix}$$

we'll define

$$\{Y\}=\begin{bmatrix} y(x_0) \\ y(x_1) \\ \vdots \\ y(x_n) \end{bmatrix}, [Z]=\begin{bmatrix} z_0(x_0) & z_1(x_0) & \cdots & z_m(x_0) \\ z_0(x_1) & z_1(x_1) & \cdots & z_m(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ z_0(x_n) & z_1(x_n) & \cdots & z_m(x_n) \end{bmatrix}, \{e\}=\begin{bmatrix} e(x_0) \\ e(x_1) \\ \vdots \\ e(x_n) \end{bmatrix}$$

which contain the values of the basis functions and the errors at each point, and

$$\{A\}=\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}$$

which contains the $m+1$ parameters we optimize.

We can also express error in our model as a sum of the squares much like before:

$$S_r=\sum_{i=0}^{n}\left(y_i-\sum_{j=0}^{m} a_j z_j(x_i)\right)^2$$

You can show by taking partial derivatives that S_r is minimised when

$$[[Z]^T[Z]]\{A\} = \{[Z^T]\{Y\}\}$$

More details can of the derivation can be found here (http://fourier.eng.hmc.edu/e176/lectures/NM/node35.html), though the notation is a little different.

POWER METHOD, WHY DOES IT WORK?

If our matrix is diagonalizable it has a set of orthogonal eigenvectors $\nu_1, \nu_2, \ldots, \nu_n$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ with $|\lambda_1| > |\lambda_j|$ for $j > 1$.

Any vector can be expanded as a linear combination the ν_i:

$$b_0 = c_1 v_1 + c_2 v_2 + \cdots + c_m v_m.$$

If we apply our matrix $\mathbf{A}$ to b_0 k times we get

$$\begin{aligned} A^k b_0 &= c_1 A^k v_1 + c_2 A^k v_2 + \cdots + c_m A^k v_m \\ &= c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + \cdots + c_m \lambda_m^k v_m \\ &= c_1 \lambda_1^k \left(v_1 + \frac{c_2}{c_1} \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \cdots + \frac{c_m}{c_1} \left(\frac{\lambda_m}{\lambda_1} \right)^k v_m \right). \end{aligned}$$

using the fact that for the eigenvectors $\mathbf{A}\nu_i = \lambda_i \nu_i$.

Because λ_1 is larger than any other λ, all the terms except the first in the bracket tend to zero as k gets larger.

SUMMARY AND FURTHER READING

Ensure that you can do the material tested today - make and multiply matrices, solve linear equations - we will be using these repeatedly in the coming weeks.

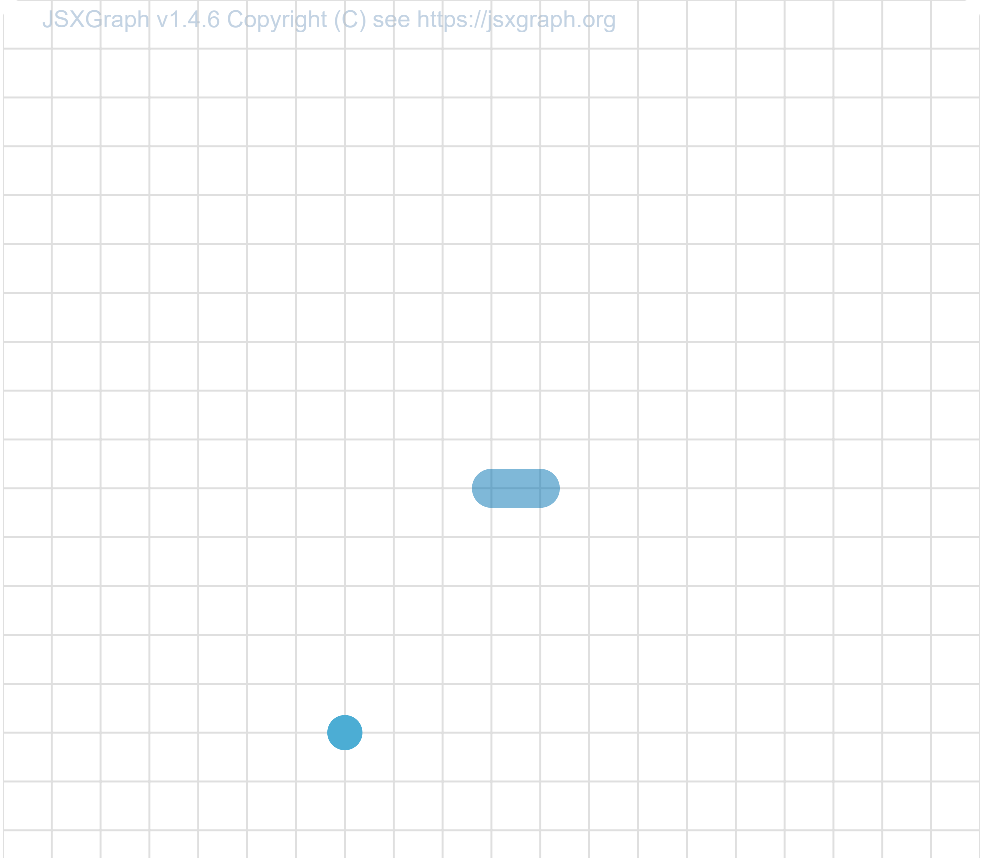
You should be reading additional material to provide a solid background to what we do in class

I suggest starting with Chapra and Canale, but there are many other text books in the library - find the one that works for you.

SNAKE

Use the arrow keys

start game


JSXGraph v1.4.6 Copyright (C) see https://jsxgraph.org