Numerical Methods Assignment 4

Andrew Martin

October 9, 2017

1. Polynomial regression and model selection

   1.1. Assuming $m > n$, write the system of equations in matrix form, identify A, x and b. How many solutions to $Ax = b$ exist in general?
   Solution

$$A = \begin{bmatrix} 1 & x_1 & \dots & x_1^m \\ 1 & x_2 & \dots & x_2^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^m \end{bmatrix}$$

$$x = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

$$b = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

...

   1.2. `fitpoly.m` code:
   Solution

```
function [y,coefs,c] = fitpoly(x,xj,yj,n)
%function fitpoly performs polynomial regression of degree n to the data xj and yj
%INPUTS
%----
%x - vector to calculate y values for the polynomial generated
%xj - column vector of known x values
```

```
%yj − column vector of known y values corresponding to the xj vector
%n − degree of the polynomial to generate
%OUTPUTS
%−−−−
%y − column vector of the values corresponding to the inputted x vector
%coefs − the coeficients corresponding to the solutions to the LSE
%c − the condition number of the regression matrix
%
%−−−−
%Andrew Martin
%a1704466
%9/10


i = 0:n;
A = xj.^(i);
coefs = A\yj;
c = cond(A);
y = x.^(i) *coefs ;
end
```

...

1.3. Choosing $n = m - 1$ gives $||e|| = 0$ Why?

Solution

This gives m equations with m-1 variables - which can be solved directly. This means the set will pass through all points - giving 0 error. A polynomial of degree k requires $k + 1$ points to generate.

E.g. a linear equation requires 2 points, a quadratic needs 3 points, etc. etc.. **...**

1.4. `testfitpoly.m` code:

Solution

```
data = importdata('facebook.dat');
xj = data(:,1);
yj = data(:,2);
maxn=length(xj); %458 as that is n=m
AIC = inf; %initialise to a maximum possible value
x = xj;
for n=1:maxn
    [y,coefs,c] = fitpoly(x,xj,yj,n);
    ej = y − yj;
    AICn = 2*n + 458*log(sum(ej.^2));
```

```
        if(AICn > AIC)
            AIC = AICn;
            break;
        end
        AIC = AICn;

end
hold on
plot(x,y,'r-')
plot(xj,yj,'b.')
title("Polynomial regression on Facebook search data")
xlabel("Number of years since 1 January 2007")
ylabel("Normalised weekly search volume counts for Facebook")
legend(["Polynomial corresponding to minimum AIC, n = " + num2str(n)],"Given Fac
hold off
%%1.6
figure
x = linspace(min(xj),11,500)';
[y,~,~] = fitpoly(x,xj,yj,n);
plot(x,y,'b',x(end),y(end),'r.')
title("Prediction for 2017")
xlabel("Number of years since 1 January 2007")
ylabel("Normalised weekly search volume counts for Facebook")

%%1.7
[y,coefs,c] = fitpoly(xj,xj,yj,20);
```

...

1.5. Plot the data and nth degree polynomial over the same time period. Label x axis as "number of years since 1 January 2007"

Solution

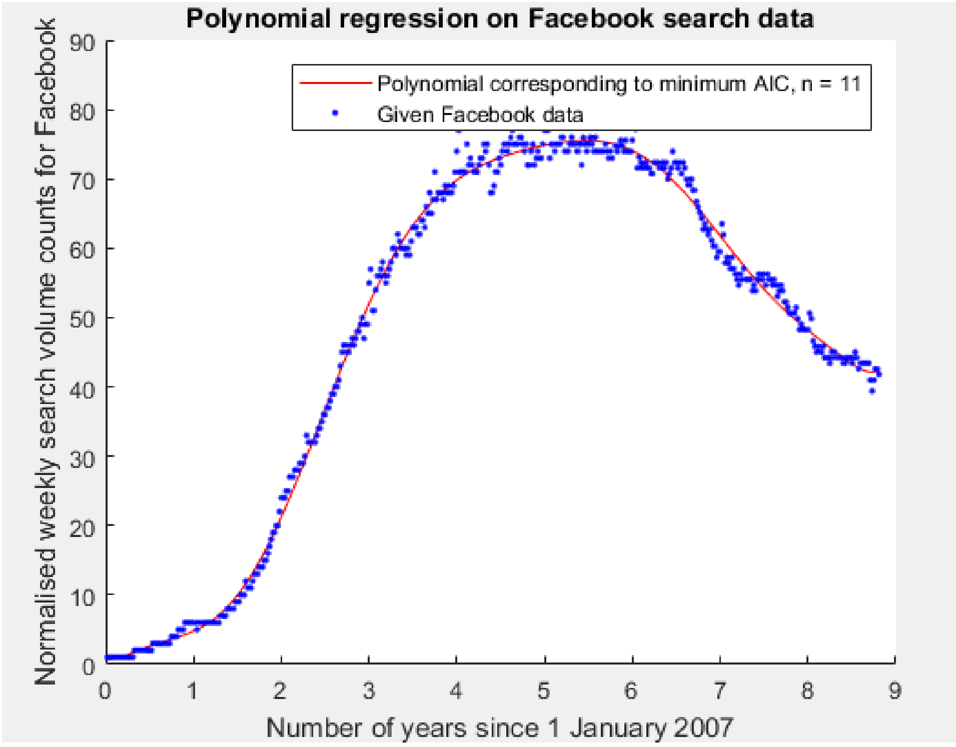
Above is the n-th degree polynomial over the same period **...**

1.6. Could the model reasonably predict Facebook search volume today in September 2017? Why or why not?

Solution

Firstly the graph generated by the model for 2017 is below:

From this it appears it is not very effective in predicting present or future search volume. This would have several reasons including: A lot of people use the Facebook app, many people save bookmarks rather than using google to find it, and facebook's use may have even peaked. **...**



1.7. Fit a degree 20 polynomial to the dataset. With reference to the outputs of `fitpoly`, explain why Matlab throws a warning message here.

Solution

The error message states:

> In fitpoly (**line** 23)
  In testfitpoly (**line** 36)
Warning: Rank deficient, **rank** = 7, tol = 2.743044e+06.

This error corresponds to the line `coefs = A\yj` , where A is the matrix A shown in 1.1. The issue is A is rank deficient. It has rank 7, when it needs rank 21 (to give solutions). The issue arises as a number of the values in the rows of A become too small for machine precision and become 0 throughout. **...**

2. Solution of the heat equation using Jacobi iteration

2.1. Referring to the relevant theorem - explain why Jacobi iteration is guaranteed to converge to the solution $Au = b$ when $\alpha > 0$

Solution

Theorem 5.32 states that Jacobi iteration will converge if the matrix A is diagonally dominant. If $\alpha > 0$ then the terms on the

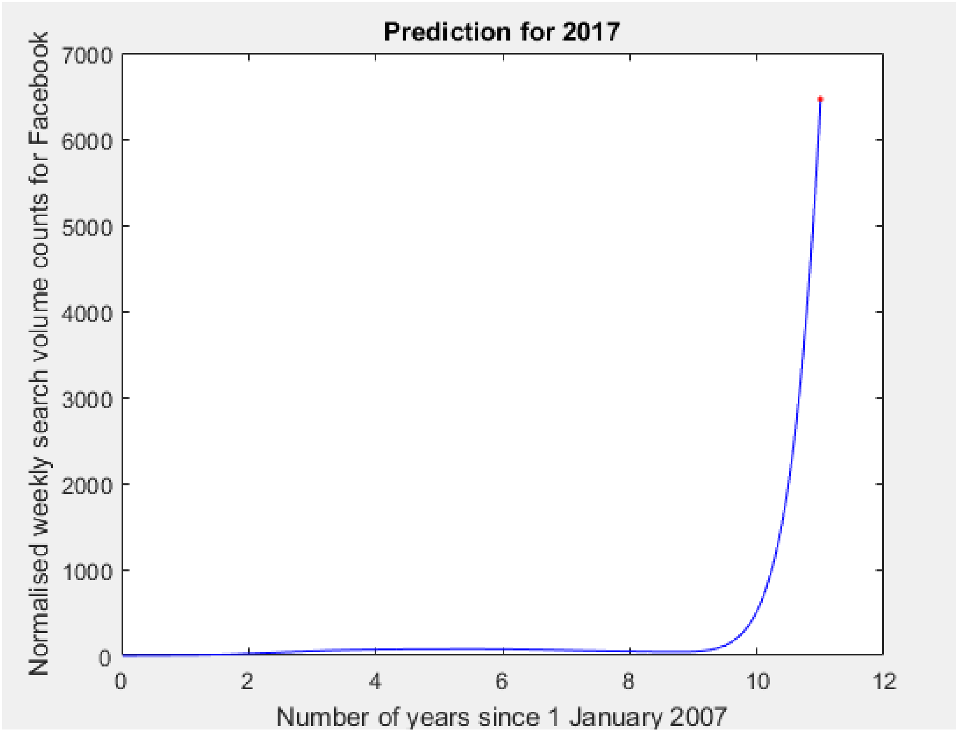


Figure 1: Trend from extrapolation

diagonal will have absolute value $2 + \epsilon$ for $\epsilon = \alpha h^2$. The matrix is diagonally dominant if

$$|-2 - \alpha h^2| \geq |1| + |1|$$

So for $\alpha > 0$

$$2 + \epsilon > 2$$

So the matrix is diagonally dominant.

...

2.2. Using the information given, write down a formula for the elements of the Jacobi iterate $u^{(k)}$ in the form

$$u_i^{(k)} = u_i^{(k-1)} - ?, \quad i = 2, \ldots, N-1$$

Solution

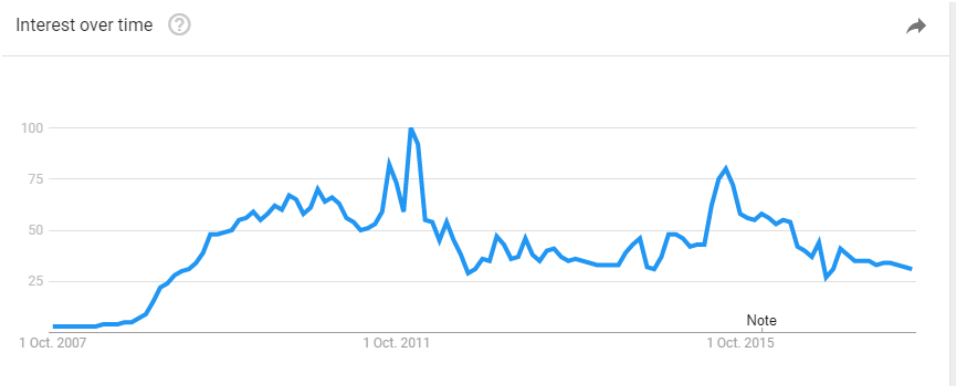


Figure 2: True trend from Google trends

Given the normal form: $u^{(k)} = u^{(k-1)} - D^{-1}r^{(k-1)}$

$$
\begin{aligned}
u^{(k)} &= u^{(k-1)} - D^{-1}r^{(k-1)} \\
&= u_i^{(k-1)} - \frac{-1}{2+\alpha h^2} r_i^{(k-1)} \\
u_i^{(k)} &= u_i^{(k-1)} + \frac{r_i^{(k-1)}}{2+\alpha h^2}
\end{aligned}
$$

...

2.3. Write a function `solveheat.m` to solve $Au = b$ using Jacobi iteration.

Solution

As below:

```
function [u,x,rnorm] = solveheat(alpha,func,L,u0,uL,N,tol,maxits)
%Function solveheat
%solves the heat PDE in one dimension in the form d^2u/dx^2 - alpha * u = - f(x)
%Andrew Martin
%8/10
%a1704466
%INPUTS
%----
%alpha - the value corresponding to alpha in the PDE
%func - the function f(x) to equate to
%L    - length of the 'heated rod'
%u0   - initial condition corresponding to the LHS of the rod
%uL   - initial condition corresponding to the RHS of the rod
%N    - number of points to solve over
%tol  - solution tolerance - if the equation varies by less than this over
```

```
%        an iteration, break.
%OUTPUTS
%----
%u     - the solution to the PDE at x
%x     - the vector of N evenly distributed points between 0 and L
%rnorm - the value

h = L/(N-1);
u=zeros(N,1);
u(1) = u0;
u(N) = uL;
j=1:N;
i=2:N-1;
x = (j-1)'*h;
fx = func(x);
r = zeros(N,1);
Dinv = 1/(2+alpha*h^2);
for iteration=1:maxits
    r(i)= u(i-1) - (2+alpha*h^2)*u(i) + u(i+1) + h^2 * fx(i);
    u(i) = u(i) + Dinv * r(i);
    rnorm = norm(r);
    if(rnorm<tol)
        disp(iteration);
        break;
    end
end

end
```

...

2.4. Considering the BVP given, verify the exact solution is

$$u(x) = \frac{1 + \frac{1}{2}\sin(1)}{\sinh(1)}\sinh(x) - \frac{1}{2}\sin(x)$$

Solution

$$
\begin{aligned}
\frac{\partial^2 u}{\partial x^2} - u &= sin(x) \\
\lambda^2 - \lambda &= 0 \\
\lambda &= 0, \quad \lambda = 1 \\
u &= c_1 e^x + c_2 e^{-x} + d\sin(x) \\
\frac{\partial^2 u}{\partial x^2} &= c_1 e^x + c_2 e^{-x} - d\sin(x) \\
\frac{\partial^2 u}{\partial x^2} - u &= -2d\sin(x) = \sin(x) \implies d = \frac{-1}{2} \\
u &= c_1 e^x + c_2 e^{-x} - \frac{\sin(x)}{2}
\end{aligned}
$$

Applying BCs:

$$
\begin{aligned}
u &= c_1 e^x + c_2 e^{-x} - \frac{\sin(x)}{2} \\
u(0) = 0 &= c_1 e^0 + c_2 e^0 - \frac{\sin(0)}{2} \\
&= c_1 + c_2 \implies c_2 = -c_1 \\
u(1) = 1 &= c_1 e^1 - c_1 e^{-1} - \frac{\sin(1)}{2} \\
c_1(e^1 - e^{-1}) &= 1 + \frac{\sin(1)}{2} \\
\implies c_1 &= \frac{1 + \frac{1}{2}\sin(1)}{(e^1 - e^{-1})}
\end{aligned}
$$

Which gives

$$
\begin{aligned}
u &= -\frac{1 + \frac{1}{2}\sin(1)}{(e^1 - e^{-1})} e^x + \frac{1 + \frac{1}{2}\sin(1)}{(e^1 - e^{-1})} e^{-x} - \frac{\sin(x)}{2} \\
&= -\frac{1 + \frac{1}{2}\sin(1)}{(e^1 - e^{-1})} \left(e^x + e^{-x}\right) - \frac{\sin(x)}{2} \\
&= -\frac{1 + \frac{1}{2}\sin(1)}{\sinh(1)} \sinh(x) - \frac{1}{2}\sin(x)
\end{aligned}
$$

...

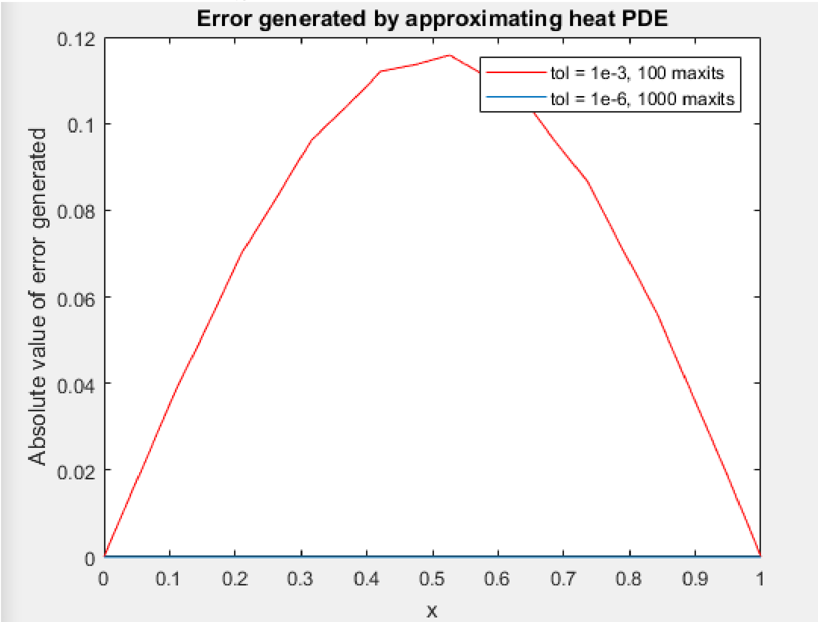
2.5. Write `testsolveheat.m` and create two plots:

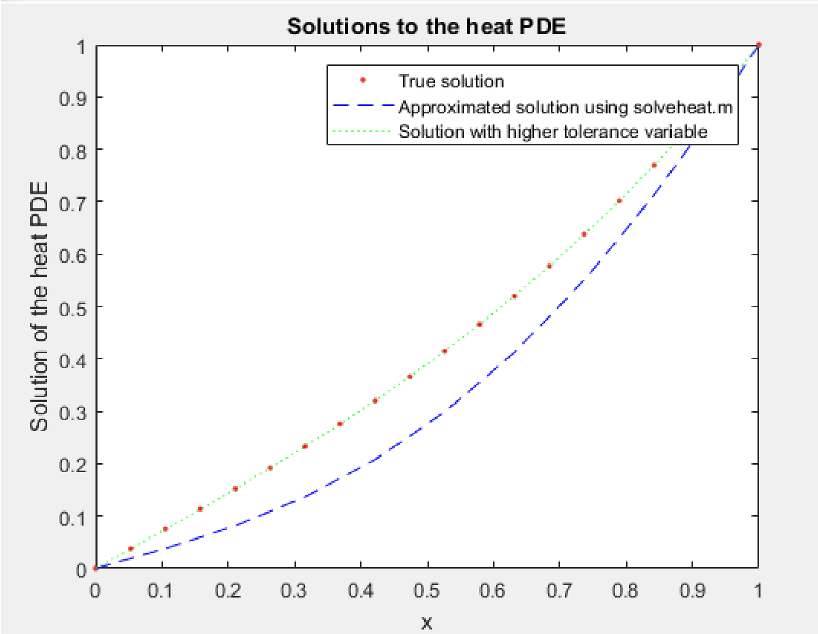
a. Showing exact solution and approximate solutions as functions of x

b. Another showing the absolute value of the difference between those two.

Make sure plots are labelled.

Solution





...

2.6. Add code to `testsolveheat.m` that uses Jacobi iteration to solve the heat equation with given ICs and BCs

Solution

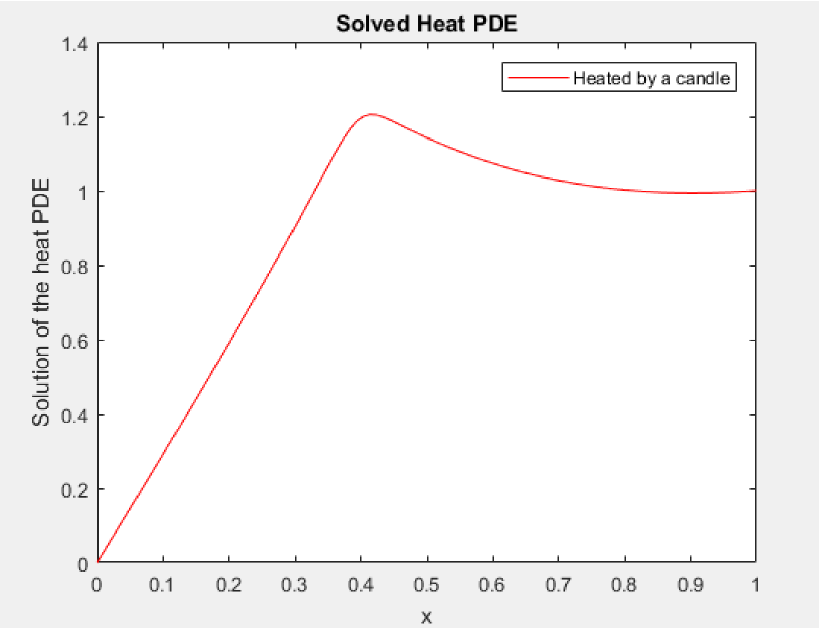
The entirety of `testsolveheat.m` is below

```
fun = @(x) (1+0.5*sin(1))/sinh(1) *sinh(x) - 0.5*sin(x);
func =@(x) -sin(x);


%%Iterate solutions to Heat PDE
[u,x,rnorm] = solveheat(1,func ,1,0,1,20,1 e-3,100);
[u2,~,~] = solveheat(1,func ,1,0,1,20,1 e-6,1000);
plot(x,fun(x),'r.',x,u,'b--',x,u2,'g:')
title("Solutions to the heat PDE")
legend("True solution","Approximated solution using solveheat.m","Solution with highe
xlabel("x");
ylabel("Solution of the heat PDE");

%%Error generated
figure
diff = abs(fun(x)-u);
diff2 = abs(fun(x)-u2);
plot(x,diff,'r',x,diff2 );
title("Error generated by approximating heat PDE")
legend("tol = 1e-3, 100 maxits","tol = 1e-6, 1000 maxits")
xlabel("x");
ylabel("Absolute value of error generated");

%2.6
N=100;
L=1;
alpha = 1;
u0 = 0;
uL = 1;
f = @(x) 80* exp(-(x-0.4).^2/0.001);
tol = 1e-3;
maxits = 10000;
[u3,x,rnorm] = solveheat(alpha,f,L,u0,uL,N,tol,maxits);
figure
plot(x,u3,'r-');
title("Solved Heat PDE")
legend("Heated by a candle")
xlabel("x");
ylabel("Solution of the heat PDE");
```


Solved Heat PDE
Heated by a candle
Solution of the heat PDE
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9

...