MATH661 Project 2 - Nonlinear model reduction

Posted: 10/31/22

Due: 11/14/22, 11:55PM (first draft, comments will be returned, revision due on 10/18)

- Julia HDF5 package installation (only needs to be invoked once, will be used in this assignment)

1 Introduction

1.1 Overview

In Project 1 you used an interpolation of noisy data to extrapolate forward in time. The data was

$$\mathcal{D} = \{(k, \boldsymbol{A}_k), k = 1, 2, ..., N, \boldsymbol{A} \in \mathbb{R}^{n_x \times n_y}\},$$

and the interpolations were done by linear and quadratic polynomials. In this project you are invited to construct a nonlinear approximation of a problem that has historically attracted a great deal of interest: fluid-structure interaction (FSI). Such problems are representative of coupled dynamics of two systems, and are encountered throughout the sciences. In FSI problems the two systems are:

1. A fluid system, described in some approximation of its behavior: potential flow, Euler flow, Navier-Stokes flow, viscoelastic flow;

2. A solid system, again described in some approximation of its behavior: Hookean elastic, hyperelastic, viscohyperelastic.

1.2 Historical context

Famous examples of FSI are:

1. The 1940 Tacoma Narrows Bridge collapse in which wind gusts down a narrow value into Puget Sound led to the collapse of a suspension bridge;

2. The 1960 crash of a Lockheed L-188 Electra aircraft operated as Flight 710 by Northwest Orient Airlines that experienced resonant vibration of an engine strut during flight leading to engine separation and resultant crash;

3. Hemodynamics, including sickle cell disease in which congenital erythrocyte malformation affects deformability needed to pass through capillaries, and obstruction-induced pulsations of a blood vessel associated with thrombosis or aneurysms.

1.3 Hemodynamics data

This project investigates model reduction for aneurysms. A complicated model is available based upon numerical solution of the PDEs governing fluid flow and blood vessel elasticity. It is implemented in my BEARCLAW code, and has been placed in `~/courses/MATH661/projects/P02/flexjet`. It contains the FSI application as a module that interacts with the PDE solve in `/opt/bearclaw` in the SciComp@UNC virtual machine.

- Commands to build the flexjet executable, using the GNU make utility that all scientific computational researchers should know how to use.
- Command to run the executable to produce a data set in the `flextjet/out` subdirectory as a sequence of time snapshots saved in the platform-portable HDF4 data format. Depending on your machine speed, this might take up to 10 minutes to complete. If you want intermediate printouts of execution stage, simply use the command `xbear`, without redirection of output to the null device.
- Command to visualize the results. Both output and error messages are redirected to null device. The OpenDX public-domain visualization is used, a package that allows visual programming to process data into images. After investigating data set, close the visualization application.
- Commands to reformat the data to HDF5 format so that it is readable in Julia. BASH shell commands are used to loop through the files.
- Data from HDF5 files can be read into Julia, exemplified here for the snapshot after one time step, stored in `q00001.h5`.

1.4 Least squares model reduction

From the above data sets $\mathcal{D} = \{(k, u_k, v_k, p_k, x_k, y_k), k = 1, ..., N\}$ are available with u_k, v_k, $p_k, x_k, y_k \in \mathbb{R}^{n_x \times n_y}$, $n_x = 160$, $n_y = 40$, k a time index. The data can be used to construct a reduced, one-dimensional model described by the functions:

- $D(t, x)$ the diameter of the blood vessel at position x, and time t;
- $U(t, x)$ the average velocity in the x-direction.

The diameter at time k and position x_i is obtained as

$$D_k(i) = y_k(i, n_y) - y_k(i, 1), 1 \leqslant i \leqslant n_x.$$

The average velocity is obtained as

$$U_k(i) = \frac{1}{n_y} \sum_{j=1}^{n_y} u_k(i, j), 1 \leqslant i \leqslant n_x.$$

2 Track 1 & 2 common problems

1. Write some utility routines to work with the data:

   a) A function to read u_k, v_k, p_k, x_k, y_k for given k;

   b) A function to compute $D_k(i)$;

   c) A function to compute $U_k(i)$.

   Solution:

   - A function `readtime` to read u_k, v_k, p_k, x_k, y_k for given k
   - A function `D(k,i)` to compute $D_k(i) = y_k(i, n_y) - y_k(i, 1), 1 \leqslant i \leqslant n_x$

```
∴ function D(k,i)
    y = readtime(k,5); nx,ny = size(y)
    return y[i,ny]-y[i,1]
  end
```

```
D
```

```
∴
```

   - A function `U(k,i)` to compute $U_k(i) = \frac{1}{n_y} \sum_{j=1}^{n_y} u_k(i, j), 1 \leqslant i \leqslant n_x$.

```
∴ using Statistics
```

```
∴ function U(k,i)
    ρu = readtime(k,1); nx,ny = size(ρu)
    return mean(ρu[i,:])
  end
```

```
U
```

```
∴
```

2. Construct least squares polynomial approximants of $D_n(t,x)$, $U_n(t,x)$ of degrees $n=3$, 4 at 10 randomly chosen time snapshots. The data for the least squares approximation of $D(k,x)$ is $\mathcal{D}=\{(x_i, D_k(i)), i=1,...,n_x\}$.

   Solution:

   In P01, we have constructed a linear and a quadratic prediction model for the weather image data $\{(t, B_t)\}$ at time $k+1$ as

$$B_{k+1}(t) = B_k + (t-k)(B_k - B_{k-1})$$

$$B_{k+1}(t) = \alpha\,(t-k+1)^2 + \beta\,(t-k+1) + \gamma,$$

   where $\gamma = \boldsymbol{B_{k-1}}$, $\alpha = \frac{1}{2}(\boldsymbol{B_k - 2B_{k-1} + B_{k-2}})$, $\beta = \frac{1}{2}(\boldsymbol{B_k - B_{k-2}})$.

   In this project, we aim to estimate nonlinear polynomial approximations of D and U as a function of time t. We first choose 10 randomly chosen time snapshots and assume that the dataset $\mathcal{D}_D=\{(x_i, D_k(i)), i=1,...,n_x\}, \mathcal{D}_U$ are given. Then, we apply and evaluate the least squares polynomial approxiamtion of degree 3 and 4. In this way, we can construcct the one-dimensional model of the diameter of the blood vessel and the average velocity as a function of position.

   a) Choose 10 time snapshots and construct a polynomial basis M. We will use the Monomial basis.

```
∴ function MonomialBasis(t,n)
    m=size(t)[1]; A=ones(m,1);
    for j=1:n
      A = [A t.^j]
    end
    return A
  end;
```

```
∴ import Pkg; Pkg.add("StatsBase");
```

```
    Updating registry at `~/.julia/registries/General`
    Updating git-repo
`https://github.com/JuliaRegistries/General.git`
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.6/Project.toml`
  No Changes to `~/.julia/environments/v1.6/Manifest.toml`
```

```
∴ using StatsBase;N=97;t_sampled=sample(1:N,10,replace=false);
```

```
∴ k=t_sampled[1]; xk=readtime(k,4);n=3;M =
  MonomialBasis(xk[1:nx,1],n);
```

```
∴
```

b) The least squares estimator can be obtained using the backslash operator. We can set the data vector at time k to be $d=(D_k(i))$, and $u=(U_k(i))$ to get each estimated coefficients a, by solving $Ma=d\,(\text{or}\,u)$. We write this procedure as a function `polyEst`, which gets the dataset as an argument.

```
∴ function polyEst(k, n, data)
      xk=readtime(k,4);M = MonomialBasis(xk[1:nx,1],n);
      return M\data.(k,1:nx)
  end
```

```
polyEst
```

```
∴ Dlsq=polyEst(t_sampled[1], 3, D);
```

$$[\,4\,] \qquad (3)$$

```
∴
```

c) Now we can evaluate the polynomial estimation at time t using the evaluation scheme.

```
∴ function Horner(x,a)
    m = size(a)[1]
    p=a[m]
    for k= (m-1):-1:1
      p = a[k]+p*x
    end
    return p
  end
```

```
Horner
```

```
∴ function polyEval(t,n,data)
      aLSQ=polyEst(t,n, data)
      p_est = zeros(nx);xk=readtime(t,4)
      for i=1:nx
        p_est[i]=Horner(xk[i,1],aLSQ)
      end
      return p_est
  end
```

```
polyEval
```

```
∴
```

3. Assess the accuracy of your one-dimensional model by plotting $D_n(t,x), U_n(t,x)$ and the data, and computing errors

$$\varepsilon_D(k,n) = \left(\sum_{i=1}^{n_x} (D_n(k,x_i) - D_k(i))^2 / \sum_{i=1}^{n_x} (D_k(i))^2 \right)^{1/2},$$

similar for ε_U.

Solution:

First of all, we plot the dataset and the one-dimensional model of polynomial degree 3(yellow) and 4(green) using the function `plotPolyData`. Figure 1 and 2 shows the results at 10 randomly chosen time snapshots. It shows that the polynomial approximation shows some limitations in fitting wiggly fluctuations and constant trends. Higher degree tends to fit the data trend better, for example, the 6th plot for fitting D or the 10th plot for fitting U shows the green line is representing the data better than the yellow. Figure 3 further shows the comprehensive error results by evaluating the performance numerically using the given error criteria.

```
∴ function plotPolyData(data,t)
      d=data.(t,1:nx)
      xk=readtime(t,4)
      h1 = polyEval(t,3,data)
      h2 = polyEval(t,4,data)
      plot(xk[1:nx,1],d, label="data"); grid("on")
      plot(xk[1:nx,1],h1, label="n=3");plot(xk[1:nx,1],h2,
  label="n=4");
    ylabel("time␣"*string(t));xlabel("x");legend();
  end
```

```
plotPolyData
```

We can finally define the error function `polyErr`, getting the dataset, time, and the degree of polynomial as argument. Figure 3 indicates that both results show that degree 4 shows less error for the approximation than the degree 3 results.

```
∴ function polyErr(t,n,data)
    d=data.(t,1:nx)
    return norm((polyEval(t,n,data).- d))/norm(d)
  end
```

```
polyErr
```

∴

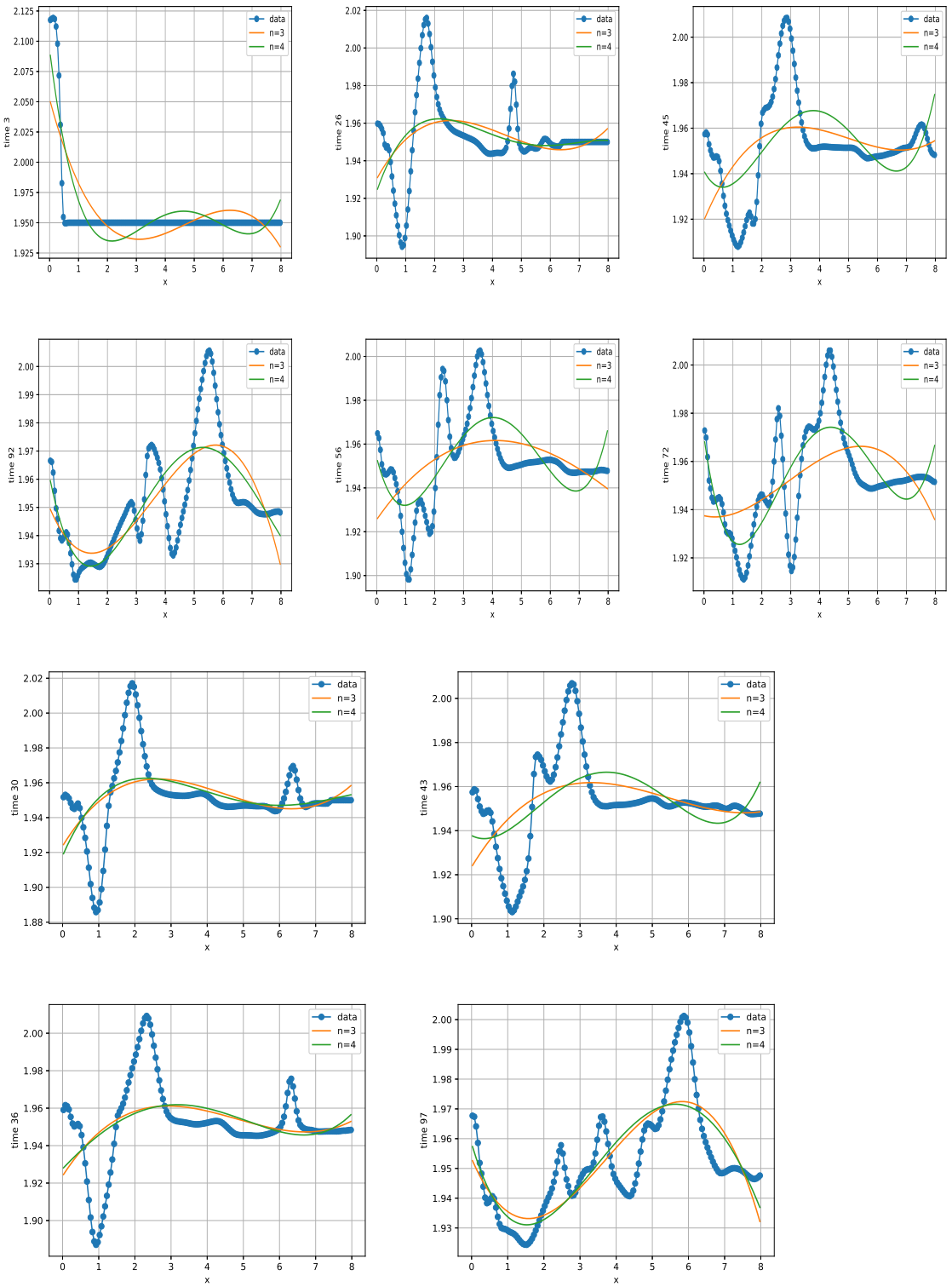

Figure 1. Polynomial approximation results for $D(t, x)$ at 10 randomly chosen time t's.

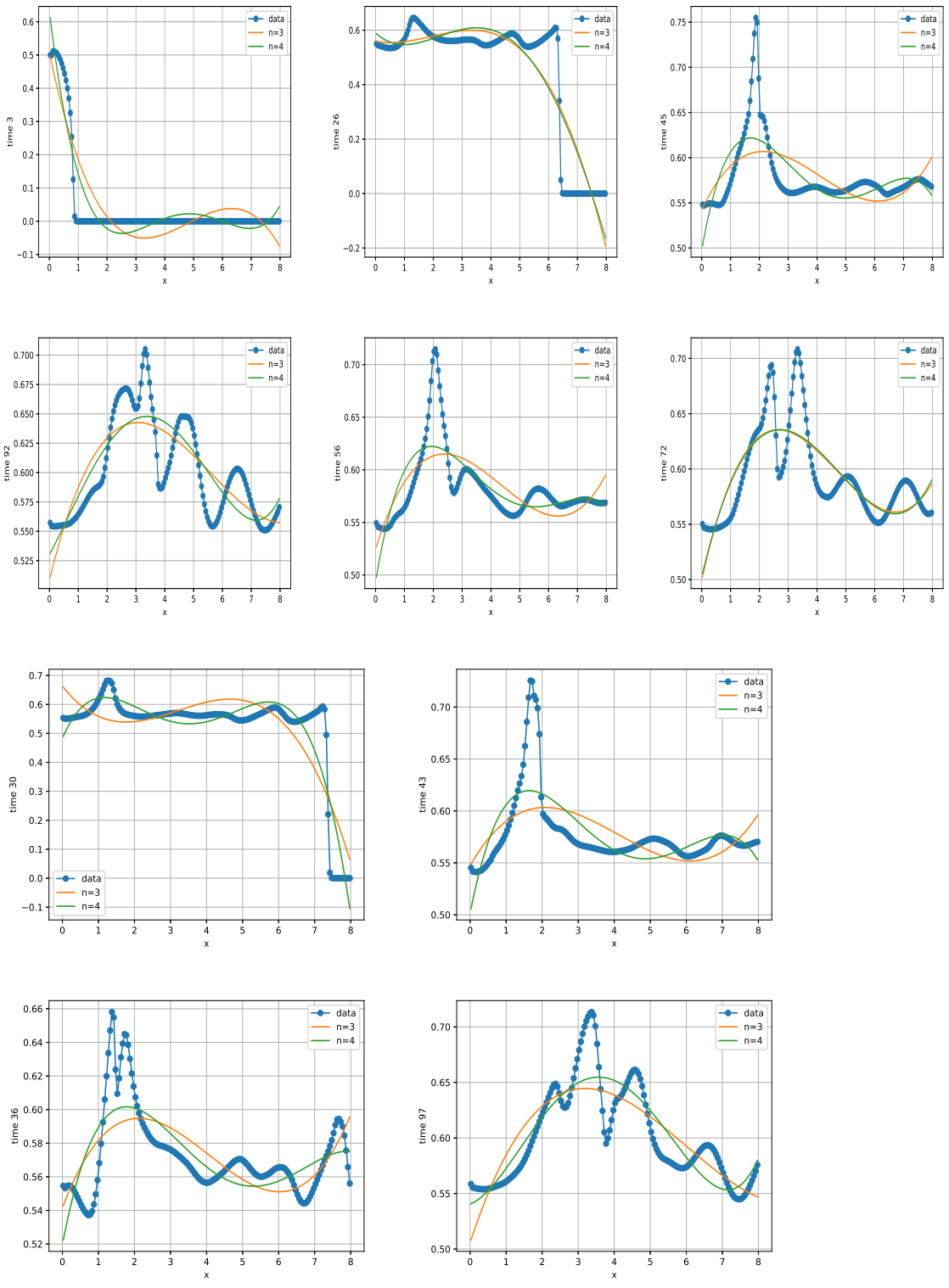

Figure 2. Polynomial approximation results for $U(t, x)$ at 10 randomly chosen time t's.

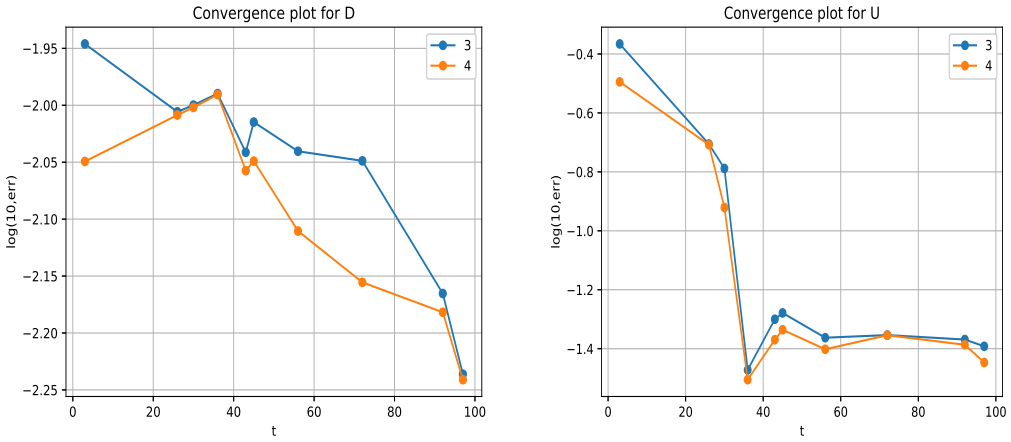

Figure 3. Error plot of polynomial approximation of D and U at 10 different time stapshots colored by the degree of the polynomial.

3 Track 2 additional problems

1. Read [1] and write a one-page synopsis of methods used to construct one-dimensional models of blood flow in arteries.

2. Compare the above 1D model with those presented in [1]. Carry out a critical assessment of P02-1D model capabilities with those from [1].

Bibliography

[1] Luca Formaggia, Daniele Lamponi, and Alfio Quarteroni. One-dimensional models for blood flow in arteries. *Journal of Engineering Mathematics*, 47(3):251–276, dec 2003.