MATH661 Project 1
Data-driven weather prediction

Posted: 09/09/21

Due: 09/23/21, 11:55PM (first draft, comments will be returned, revision due on 09/30)

1 Introduction

1.1 Numerical weather prediction

Weather prediction is currently based upon numerical solution of the equations of fluid dynamics, i.e., the Navier-Stokes equations

$$\boldsymbol{v}_t + \boldsymbol{v}\nabla\boldsymbol{v} = -\nabla p + \nu\nabla^2\boldsymbol{v},$$

to find the velocity $\boldsymbol{v}$ and pressure p at grid points $(x_i,\ y_j,\ z_k)$. Typical mesh sizes are $1\,\text{km} \leqslant w \leqslant 10\,\text{km}$ in the "horizontal" directions, $10m \leqslant h \leqslant 100m$ in the vertical direction, and the key computational bottleneck is the solution of a linear system

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b}$$

to find grid pressure values.

An intriguing alternative to the above physics-based model is to process weather data measurements, identify principal features, and extrapolate into the future based upon current trends. This is known as a data-driven model, and is a current research topic, e.g. consult this machine learning approach that applied a non-linear neural network model, or the type of job opportunities requiring mastery of concepts explored in this project.

This project investigates a data-driven approach using tools of linear algebra. The objectives of this project are:

- introduce practical techniques of working with large data sets
- understand how data can be decomposed using matrix factorizations
- learn to distinguish between major features and noise
- highlight the capabilities of additive corrections (linear algebra), but also its limitations
- gain an appreciation of how the many tools available in a Unix-like environment (e.g., Linux in SciComp@UNC or OS/X fully configured for scientific use) work together to build complex models

1.2 Weather data

- The following commands within a BASH shell brings up an animation of weather over the Americas from NOAA. A weather model will be constructed using this data
- The model will use various Julia packages to process image data, carry out linear algebra operations. The following commands build the appropriate Julia environment. The Julia packages are downloaded, compiled and stored in your local Julia library.
- Once compiled, packages are imported into the current environment
- With appropriate packages in place and available, the satellite imagery can be imported into the Julia environment

1.3 Data preprocessing

- A first step in all data-driven models is to transform the raw data so that it becomes suitable for some specific goal. This is exemplified by the simple operation of concentrating on a rectangular area of the Earth within the imagery and extracting a floating point data window.

1.4 Computing the SVD

The SVD of the array of floats obtained from an image identifies correlations in gray-level intensity between pixel positions, an encoded description of weather physics.

```
∴ n=32; A=data[:,:,n]; U,S,V=svd(A);
```

```
∴
```

- Define a function `rsvd` to sum the first p rank-one updates from an SVD, containing the dominant correlations,

$$\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^T \cong \boldsymbol{B} = \sum_{j=1}^{p} \sigma_j\,\boldsymbol{u}_j\,\boldsymbol{v}_j^T.$$

  These can be identified as large scale weather patterns.

- Define a function `lsvd` to sum the last p rank-one updates from an SVD, containing the least-correlated modes

$$\boldsymbol{C} = \sum_{j=r-p}^{r-1} \sigma_j\,\boldsymbol{u}_j\,\boldsymbol{v}_j^T.$$

  These can be interpreted as "noise" with respect to large-scale weather patterns, i.e., local variations in weather.

It is possible to construct linear combinations of large-scale and local modes

```
∴ clf(); imshow(0.9*B+0.1*C,cmap="gray");
∴
```

2 Common problems (both tracks)

1. Consider large-scale weather patterns $\boldsymbol{B}_k$, $\boldsymbol{B}_{k-1}$, $\boldsymbol{B}_{k-2}$ obtained with p most significant modes from frames $k, k-1, k-2$ (i.e., different times in the past). Assuming a constant rate of change leads to the prediction

$$\boldsymbol{P}_{k+1} = \boldsymbol{B}_k + (\boldsymbol{B}_k - \boldsymbol{B}_{k-1}) \tag{3}$$

with the known large-scale weather $\boldsymbol{B}_{k+1}$. Experiment with various values of k, p. Present images from your numerical experiments and comment them.

Solution:

To evaluate the performance of image approximation, we use an appropriate norm that measures difference in images. Images are in a matrix form and for this case, the extension of a vector norm, i.e. the matrix Frobenius norm is appropriate.

$$\| \boldsymbol{A} \|_F = \left(\sum_i \sum_j a_{ij}^2 \right)^{1/2}$$

Define weather prediction error as

$$\epsilon(k,p) = \| \boldsymbol{A}_k - \boldsymbol{B}(k,p) \|_F, \boldsymbol{B}(k,p) = \boldsymbol{B}_{k-1}^{(p)} + (\boldsymbol{B}_{k-1}^{(p)} - \boldsymbol{B}_{k-2}^{(p)})$$

with

$$\boldsymbol{A}_k \text{: data at time } k, \boldsymbol{B}_k^{(p)} = \sum_{i=1}^{p} \sigma^{(k)}{}_i \boldsymbol{u}_i^{(k)} \boldsymbol{v}_i^{(k)T}$$

```
∴ function err(k,p)
            global data
            U,S,V=svd(data[:,:,k-1]); B = rsvd(p,U,S,V);
            U,S,V=svd(data[:,:,k-2]);
            B = 2*B - rsvd(p,U,S,V);
            m,n = size(B); mn=m*n; A = reshape(data[:,:,k],mn,1);
            return norm(A-reshape(B,mn,1));
  end;
∴
```

The above function compute the error for the rsvd approximation for a given frame number k and degrees of approximation p. The whole performance can be summarized in the following plot. Figure 1 shows the log error over different frames with various p values in different colors. The constant rate prediction using rsvd approximation

works well with small errors at earlier frames and showed bad performance at later frames. It also shows monotonely decreasing error trend when we increase p values. It would be better to plot over p values at given frames to see how rank one update performs. Also, calling err(k,p) over various p values recomputes many SVDs inefficiently. Since for rank one update, one does not need to recompute SVD again, we define another function doing successive rank one updates and return the vector of resulted approximation errors.

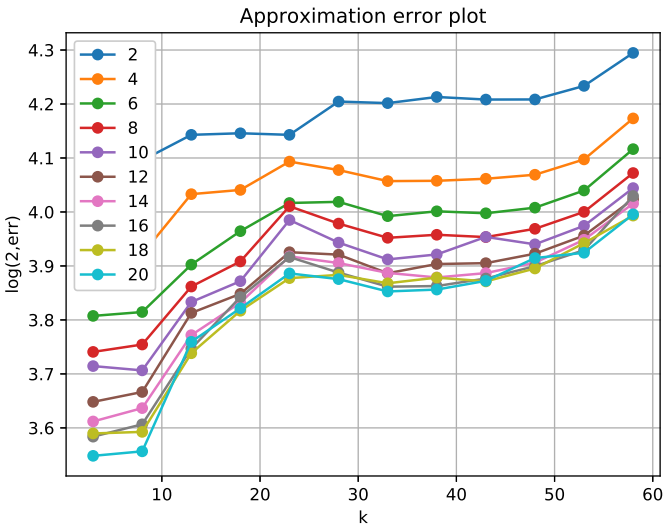


Figure 1. Error plot over data frames colored by various numbers of p.

```
∴ function computeErr(k,approx,p_array=1:30)
            global data
            m,n = size(data[:,:,k]);mn=m*n;err=[]
            A = reshape(data[:,:,k],mn,1);
            U1,S1,V1=svd(data[:,:,k-1]);
            U2,S2,V2=svd(data[:,:,k-2]);
            for p=p_array
              B = approx(p,U1,S1,V1);
              B = 2*B - approx(p,U2,S2,V2);
              err=[err; norm(A-reshape(B,mn,1))]
            end
            plot(p_array,log.(err), label="L␣k:"*string(k),"o-");
  grid("on")
    xlabel("p"); ylabel("log(2,err)");
    title("Convergence␣plot");
            return err;
  end;
```

```
∴
```

Figure 2 shows the error trend over additional rank one updates. It shows that the decrease is rapid at first, but tend to get slower. $k = 5$ result at the top further shows that the error can increase over the additional update after it reaches its local minimum around $p = 15$.

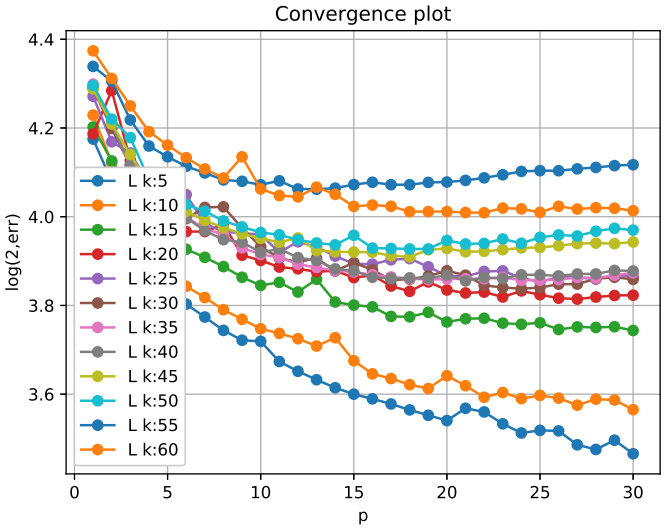


Figure 2. Error plot over data frames colored by various numbers of p.

```
∴ clf();
```

```
∴ for k=5:5:60
    computeErr(k,rsvd)
    legend()
  end
```

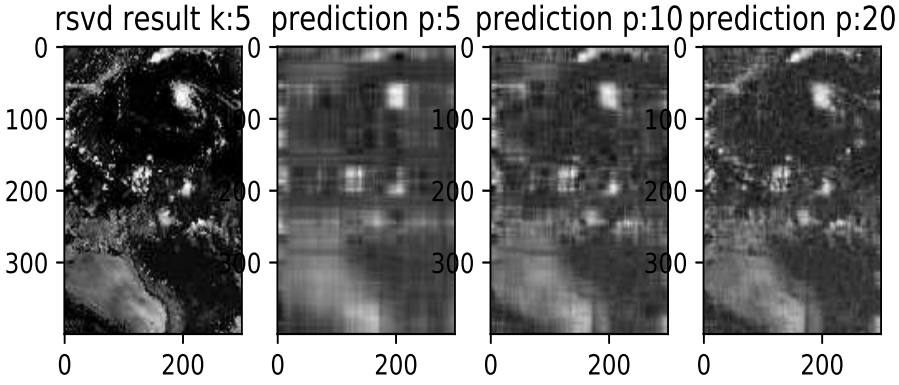
```
∴ savefig(FigPrefix*"FigErrp.eps")
```

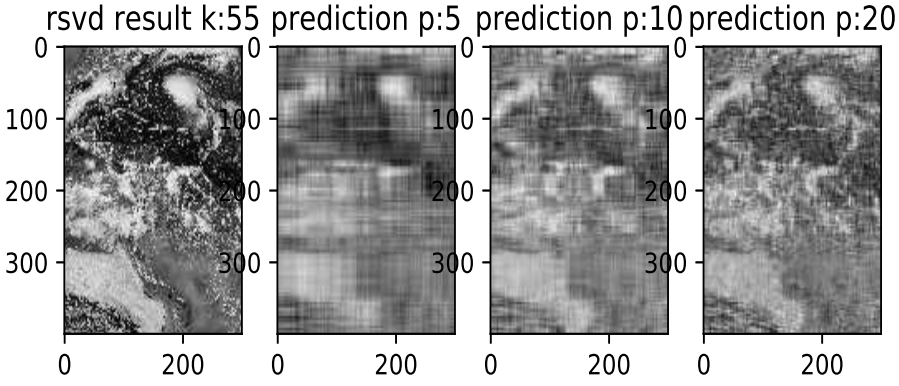
```
The PostScript backend does not support transparency; partially
transparent artists will be rendered opaque.
The PostScript backend does not support transparency; partially
transparent artists will be rendered opaque.
```

```
∴
```

```
∴
```

The image approximation result at some k and p results are in Figure 3. $k = 5$ (top) and 55 (bottom) are presented as they are the frame where the approximation showed comparably smaller error ($k = 5$) and larger error ($k = 55$). For each case, $p = 5, 20$ results are plotted to visually see how using more ranks performs well in image approximation. $k = 55$ frame shows large size of hurricanes and the clouds growing trend might be hard to predict assuming the constant rate of change here.


rsvd result k:5
prediction p:5
prediction p:10
prediction p:20
0
100
200
300
0
200


rsvd result k:55
prediction p:5
prediction p:10
prediction p:20
0
100
200
300
0
200

```
∴ function plotImage(p,k,approx)
            f, ax = subplots(1,length(p)+1)

            ax[1].imshow(data[:,:,k],cmap="gray");
            ax[1].set_title("rsvd␣result␣k:$(k)")
            for i in 1:length(p)
              A=data[:,:,k-1]; U,S,V=svd(A); B =
   approx(p[i],U,S,V);
              A=data[:,:,k-2]; U,S,V=svd(A); B = 2*B -
   approx(p[i],U,S,V);
              ax[i+1].imshow(B,cmap="gray");
              ax[i+1].set_title("prediction␣p:$(p[i])")
            end
   end;
```

```
∴ k=5;plotImage([5 10 20] ,k,rsvd);
   savefig(FigPrefix*"Fig01k="*string(k)*".eps")
```

```
∴ k=55;plotImage([5 10 20] ,k,rsvd);
   savefig(FigPrefix*"Fig01k="*string(k)*".eps")
```

```
∴
```

2. Repeat the above for local weather patterns from q least significant modes.

   Solution:

   lsvd approximation uses the least correlated modes which can be interpreted as "noise". We use the same error norm criteria to measure the differences in images. Figure 4 shows the comprehensive result. Here, larger range of q degree of approximation is evaluated, $q = 5, 15, 25, ..., 295$. Figure 5 further shows the image approximation result at $k = 5$ (top) and 55 (bottom).

   The following are some observations and interpretations.

   a) Error values in Figure 4 are quite large even when large number of nodes are used. We can observe that The error starts to decrease at a very large number of q, around 290. Figure 5 shows that the information contained in the least nodes are mostly noises.

   b) $q = 290$ result contains peripheral information like geographical features. This means that most pivotal whether information are summarized in the first few linear subspaces of the dataset. Since we have noticed that some modes contain geographic features as opposed to small-scale weather patterns, we would like to investigate their effects upon predictions? Could we eliminate geographic

features if needed? Are the geographic features actually important to keep in the model? These are further investigated in the question 3.

c) Using $q = 297$ seems to show some major whether features, which means that most information are summarized in the first 3 vectors, since the full dimension is 300.

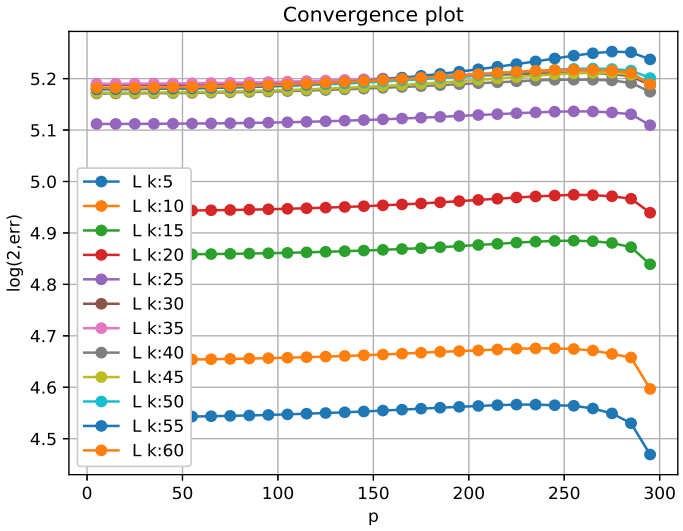


Figure 4. Error plot over data frames colored by various numbers of p.

```
∴ clf();
```

```
∴ for k=5:5:60
    computeErr(k,lsvd,5:10:300)
    legend()
  end
```

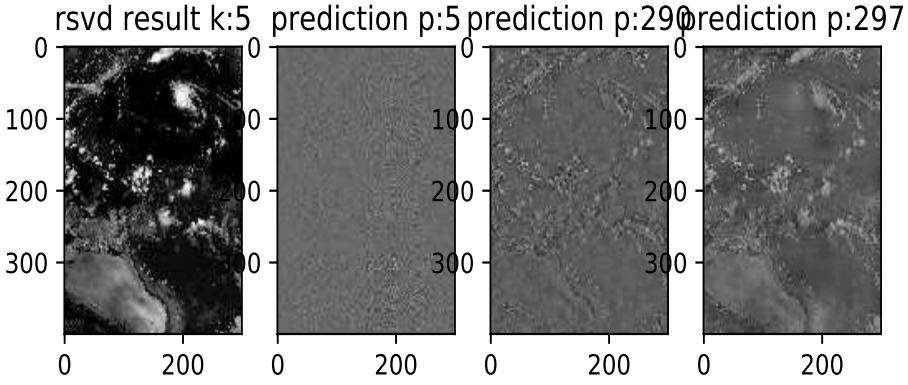
```
∴ savefig(FigPrefix*"FigErr_lsvd.eps")
```

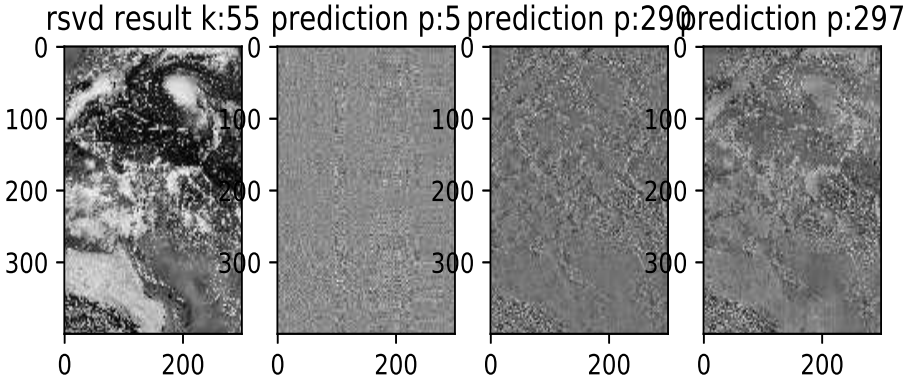
```
The PostScript backend does not support transparency; partially
transparent artists will be rendered opaque.
The PostScript backend does not support transparency; partially
transparent artists will be rendered opaque.
```

```
∴
```

```
∴
```

The image approximation result at some k and p results are in Figure 5. $k = 5$ (top) and 55 (bottom) are presented as they are the frame where the approximation showed comparably smaller error ($k = 5$) and larger error ($k = 55$). For each case, $p = 5, 290, 297$ results are plotted to visually see how using more ranks performs in peripheral image approximation.


rsvd result k:5
prediction p:5
prediction p:290
prediction p:297
0
100
200
300
0
200


rsvd result k:55
prediction p:5
prediction p:290
prediction p:297
0
100
200
300
0
200

```
∴ k=5;plotImage([5 290 297] ,k,lsvd);
  savefig(FigPrefix*"Fig02k="*string(k)*".eps")
∴ k=55;plotImage([5 290 297] ,k,lsvd);
  savefig(FigPrefix*"Fig02k="*string(k)*".eps")
∴
```

3. Repeat the above for combined large ($\boldsymbol{B}$) and small scale ($\boldsymbol{C}$) weather patterns. Experiment with different weights u, v in the linear combination $u\boldsymbol{B} + v\boldsymbol{C}$, $u + v = 1$.

   Solution:

   The following two settings are considered. Here the frame is fixed to $k = 55$ for simplicity. $u = 1$ result shows the rsvd result.

   - Define a function `avgsvd` to get the linear combination $u\boldsymbol{B} + v\boldsymbol{C}$ using first and last p rank-one updates from an SVD respectively. Since $\boldsymbol{C}$ image contains noise, one might be interested in $u \geqslant 1$ case so that we can see the effect of removing the noise. Although u near 1 results showed small errors, it could not outperform the rsvd result.

   - Define a function `geosvd` to get the linear combination $u\boldsymbol{B} + v\boldsymbol{G}$, where

$$\boldsymbol{B} = \sum_{j=1}^{p} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T, \boldsymbol{G} = \sum_{j=p+1}^{2p} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T.$$

     G is considered in the context of including or removing the geographical information. Figure 7 shows that although u near 1 results showed small errors, it could not outperform the rsvd result.

   Should geographical information be eliminated? Should the noise q be included in weather prediction? Overall, there was no evidence to support that eliminating geo-

graphical information is needed for the performance.

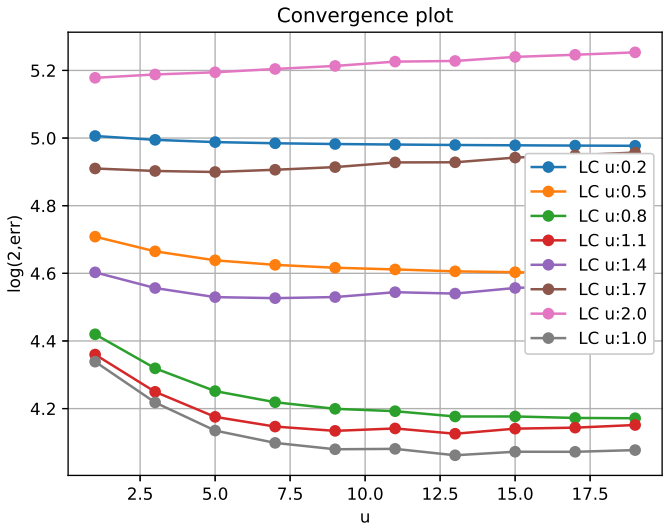


Figure 6. Error plot over data frames colored by various numbers of p.

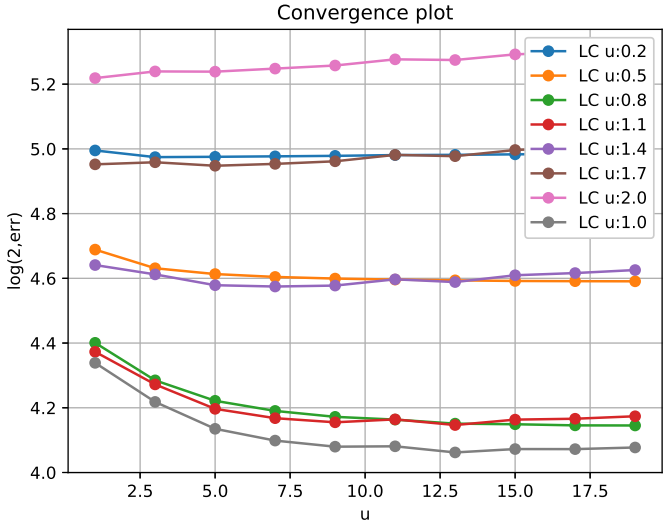


Figure 7. Error plot over data frames colored by various numbers of p.

4. Formula (3) expresses a linear prediction

$$\boldsymbol{B}(t) = \boldsymbol{B}_k + (t-k)(\boldsymbol{B}_k - \boldsymbol{B}_{k-1}),$$

evaluated at $t = k+1$. Construct a quadratic prediction based upon data $\boldsymbol{B}_k, \boldsymbol{B}_{k-1}$, $\boldsymbol{B}_{k-2}$, and compare with linear prediction in question 1. Again, experiment with various values of k, p.

Solution:

- We get the linear prediction formula by the following step (unfold to see)
- Now to do the quadratic prediction, do the following to get

$$B(t) = \mathbf{3}\boldsymbol{B_k} - \mathbf{3}\boldsymbol{B_{k-1}} + \boldsymbol{B_{k-2}}$$

This predicts using the previous observation added by the difference and the difference of differences.

- Define a function `computeErr_Qd` to measure the performance. Figure 8 indicates that the linear prediction outperforms the quadratic prediction.

```
∴ function computeErr_Qd(k,approx,p_array=1:30)
            global data
            m,n = size(data[:,:,k]);mn=m*n;err=[]
            A = reshape(data[:,:,k],mn,1);
            U1,S1,V1=svd(data[:,:,k-1]);
            U2,S2,V2=svd(data[:,:,k-2]);
            U3,S3,V3=svd(data[:,:,k-3]);
            for p=p_array
              B = approx(p,U1,S1,V1);
              B2= approx(p,U2,S2,V2);
              B = 3*B - 3*B2 + approx(p,U3,S3,V3);
              err=[err; norm(A-reshape(B,mn,1))]
            end
            plot(p_array,log.(err), label="Q␣k:"*string(k),"o-");
  grid("on")
    xlabel("p"); ylabel("log(2,err)");
    title("Convergence␣plot");
            return err;
  end;
```

```
∴
```

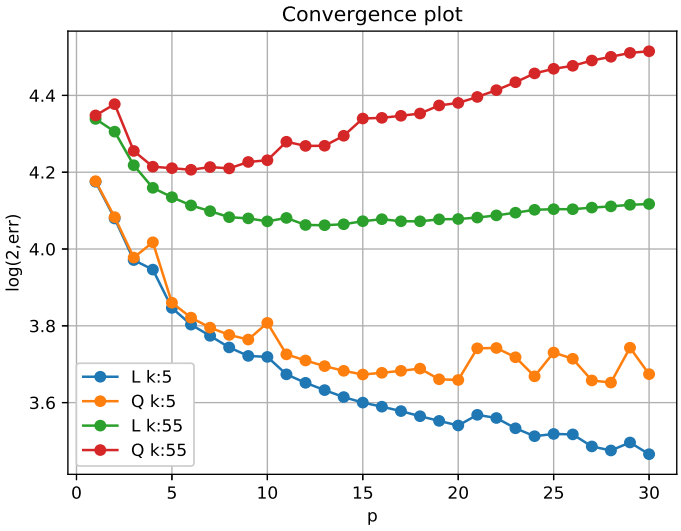


Figure 8. Error plot over data frames colored by various numbers of p.

```
∴ clf();
```

```
∴ for k=[5 55]
    computeErr(k,rsvd)
    computeErr_Qd(k,rsvd)
    legend()
  end
```

```
∴ savefig(FigPrefix*"FigErr_Qd.eps")
```

```
The PostScript backend does not support transparency; partially
transparent artists will be rendered opaque.
The PostScript backend does not support transparency; partially
transparent artists will be rendered opaque.
```

```
∴
```

```
∴ function plotImage(p,k,approx)
            f, ax = subplots(1,length(p)+1)

            ax[1].imshow(data[:,:,k],cmap="gray");
            ax[1].set_title("rsvd␣result␣k:$(k)")
            A=data[:,:,k-1]; U,S,V=svd(A); B = approx(p[i],U,S,V);
            A=data[:,:,k-2]; U,S,V=svd(A); B = 2*B - approx(p[i],U,S,V);
            ax[i+1].imshow(B,cmap="gray");
              ax[i+1].set_title("prediction␣p:$(p[i])")
            end
  end;
```

```
∴ k=5;plotImage([5 10 20] ,k,rsvd);
  savefig(FigPrefix*"Fig01k="*string(k)*".eps")
```

∴

3 Track 2 additional problems

Weyn, Durran & Caruana [4] describe a data-driven approach to weather prediction.

1. Write a short esssay (2-4 paragraphs) on the essential differences between the SVD approach in this project compared to [4]. .

2. Carry out bibliographic research, looking for papers that cite [4]. Choose a few (2 or 3), add them to the bibliography (P01.bib) and comment on further research ideas, citing your sources.

Bibliography

[1] Camille et.al Besombes. Producing realistic climate data with generative adversarial networks GANS. *Nonlinear Processes in Geophysics*, pages 347–370, 2021.

[2] Alexander Bihlo. A generative adversarial network approach to (ensemble) weather prediction. 2020.

[3] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K. Gupta. Clifford neural layers for pde modeling. 2022.

[4] Jonathan A. Weyn, Dale R. Durran, and Rich Caruana. Improving Data-Driven Global Weather Prediction Using Deep Convolutional Neural Networks on a Cubed Sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):0, sep 2020.

d