Mathematica Project , Math21b Spring 2009

Oliver Knill, Harvard University, Spring 2009

Support

Welcome to the Mathematica computer project! In case problems with this assignment, please email Oliver at math21b@fas.harvard-.edu. *Mathematica* can demand a lot of resources from computers. Therefore, **save the notebook frequently** during your work. The actual assignments are in the last blue box of this notebook. This lab is due at the last lecture of the semester. You will print the assignment part with your solutions and hand it in. All problems are listed at the end of this document. All problems have sample code in this notebook somewhere. The total work should be minimal. It can be done in a 2-3 hours if you are in a hurry. Of course we hope you get interested in one or the other topic and hang around for more. If you think you need too much time, network around or ask. Check the syllabus to remind yourself how the lab is used for your grade.

Cells and Expressions

The notebook is subdivided into cells, bounded by a blue bracket. The text you right right now is a "text" cell, the next gren cell is an "input" cell.
It computes the dot product of two vectors.

```
{3, 4, 5, 6}.{2, 3, 4, 5}
```

To evalute an expression, click anywhere on the expresson, hold down the <Shift> key and press <Enter>. The next example shows how we compute the inverse of a matrix:

```
Inverse[{{3, 2, 5}, {3, 2, 1}, {1, 1, 1}}]
```

In this notebook, input cells are boxed in green for better readability.

Basic Computations

Mathematica deals vectors and matrices in the same way as with numbers. A dot . is the matrix multiplication or dot product. The following examples should be selfexplanatory. We start with multiplying a matrix with a vector.

```
{{1, 4}, {0, 1}}.{2, 3}
```

If you want to see a result, say of a matrix operation, like it would appear in a text book, add // TraditionalForm

```
{{1, 2, 3}, {3, 4, 5}, {4, 5, 6}}.
  Inverse[{{2, 3, 4}, {1, 1, 1}, {1, 3, 6}}] // TraditionalForm
```

```
A = {{2, 3}, {1, 2}}; b = {3, 4}; Inverse[A].b
```

```
Solve[{x + y == 1, x - y == 3}, {x, y}]
```

```
Det[{{2, 3, 5, 6}, {1, 2, 3, 4}, {1, 1, 1, 1}, {12, 3, 2, 1}}]
```

```
Tr[{{a, b}, {c, d}}]
```

```
Transpose[{{1, 2, 3}, {3, 4, 5}, {1, 2, 1}}] // MatrixForm
```

```
A = {{1, 2}, {3, 4}}; QR = QRDecomposition[A]; Q = QR[[1]];
R = QR[[2]]; {TraditionalForm[Q], TraditionalForm[R]}
```

Eigenvalues

With Eigenvalues[A], you compute the eigenvalues, with Eigensystem[A], both the eigenvalues and the eigenvectors. What happens for a random matrix? Lets make an experiment and plot the complex eigenvalues of a random 50x50 matrix. You might want to try with larger matrices to see a pattern.

```
A = Table[Random[] - 1 / 2, {50}, {50}];
eigenvalues = Eigenvalues[A];
point[z_] := Point[{Re[z], Im[z]}];
Graphics[Map[point, eigenvalues]]
```

Explore this for larger and larger matrices. What do you observe?

An other question. Sergey Sadov of the Memorial university in Canada asked:
Suppose A is an NxN matrix with entries independently and uniformly distributed on[0, 1].
What is the distribution of the absolute value of the largest eigenvalues of A for large N?

```
le := Module[{},
   A = Table[Random[], {100}, {100}]; Max[Abs[Eigenvalues[A]]]];
```

Lets do an experiment, take 1000 matrices, compute the largest eigenvalue in each case and look at the distribution. This computation takes some time.

```
s = Sort[Table[le, {1000}]];
```

If we plot these data, we see the distribution function:

```
ListPlot[s]
```

The density function can be visualized as follows:

```
ListPlot[BinCounts[s, {Min[s], Max[s], 0.1}], Ticks → None]
```

While we are at it, lets also compute some statistical properties:

```
{Mean[s], Median[s], StandardDeviation[s], Min[s], Max[s]}
```

Determinants

What is the distribution of the determinant of a random matrix? To see the distribution of a random 10x10 matrix, lets do 100 experiments and plot the results in a sorted manner.

```
S = Sort[Table[Det[RandomReal[{0, 1}, {10, 10}]], {100}]];
ListPlot[S]
```

This distribution approaches the arcsin-distribution.

```
Plot[ArcSin[x], {x, -1, 1}]
```

Here is how we compute the determinant of the 100 x 100 matrix which contains the entry i*j at the entry (i,j).

```
A = Table[i * j, {i, 100}, {j, 100}];
Det[A]
```

Differential Equations

Differential equations can be solved with the command "DSolve". Here is an example, where one does not specify the initial conditions:

```
ClearAll[f]; DSolve[f'''[x] - 2 f''[x] + f'[x] == Exp[3 x], f[x], x]
```

Here is an example, where the initial conditions are specified. This is the solution of the forced driven harmonic oscillator:

```
S = DSolve[{f''[x] + f'[x] + f[x] == Cos[x],
    f[0] == 10, f'[0] == 0}, f[x], x]
```

You can then plot the solution

```
g[t_] := S[[1, 1, 2]] /. x → t; Plot[g[t], {t, 0, 20}]
```

Sometimes, it is good to clear all variables. We have used f,g before and the next solution of a differential equation would make problems if

```
Remove["Global`*"]
```

```
S = DSolve[{f''[x] == -f[x] + g[x], g''[x] == f[x] - g[x],
    f[0] == 10, f'[0] == 0, g[0] == 1, g'[0] == 0}, {f[x], g[x]}, x]
```

Markov Processes

A vector is called a probability vector if each entry is nonnegative and the sum of all its entry is 1. A matrix A for which each column is a probability vector is called a Markov matrix or stochastic matrix.

```
A = {{1 / 2, 1 / 5, 1 / 4, 1 / 3}, {1 / 6, 1 / 5, 1 / 4, 0},
   {1 / 6, 1 / 5, 1 / 4, 1 / 3}, {1 / 6, 2 / 5, 1 / 4, 1 / 3}};
MatrixForm[
 A]
```

A Markov matrix has an eigenvalue 1 because its transpose has the eigenvector [1,1,..., 1]. The eigenvector of the eigenvalue 1 of A is called the steady state.

```
N[Eigensystem[A]]
```

When iterating the matrix A, it converges to a matrix which has the eigenvector as column vectors.

```
N[MatrixPower[A, 10]]
```

Here is a magical square:

```
A = {{17, 24, 1, 8, 15}, {23, 5, 7, 14, 16},
   {4, 6, 13, 20, 22}, {10, 12, 19, 21, 3}, {11, 18, 25, 2, 9}};
MatrixForm[
 A]
```

If we divide it by the sum 65 over each column, we obtain a stochastic matrix.

Data fitting

The following example produces random data and fits them with a polynomial of degree 3.

```
h[x_] := x + 2 * Random[]; M = 400;
data = Table[{k / M, h[k / M]}, {k, M}];
f = Function[y, Fit[data, Table[z^j, {j, 0, 3}], z] /. z → y];
S1 = ListPlot[data];
S2 = Plot[f[x], {x, 0, 1}, PlotStyle → {Red}];
Show[{S1, S2}]
```

Products of random matrices

If we multiply random 2x2 matrices, how fast does the product grow? The answer depends on the size of the random numbers. Lets take random numbers between -c and c:

```
F[c_] :=
 Module[{}, randommatrix := Table[c (2 Random[] - 1), {2}, {2}];
  s = IdentityMatrix[2]; lambda = 0;
  Do[s = randommatrix.s; u = Max[Abs[Flatten[s]]];
   s = s / u; lambda = lambda + Log[u], {100}]; lambda]
```

```
Plot[F[c], {c, 0.5, 3}]
```

For which range of c, does the product appear to grow exponentially?

Nonlinear Systems

Here is an example, of how to plot a phase portrait of a nonlinear system. It is a Volterra system which is a model in population dynamics.

```
VectorPlot[{0.4 * x - 0.4 * y * x, -0.1 * y + 0.2 * x * y},
 {x, 0, 2}, {y, 0, 2}, VectorScale → 0.1,
 VectorColorFunction → Hue, StreamPoints → 20]
```

Fourier Series

Lets compute some Fourier series. In this example, we compute the series of an odd function so that we only have to compute the sin- series.

```
ClearAll[f, a, aa, s]
f[x_] := If[x < 0, 1, -1];
a[n_] := Integrate[(1 / Pi) f[x] Sin[n x], {x, -Pi, Pi}];
aa = Table[a[k], {k, 1, 30}];
s[x_, n_] := Sum[aa[[k]] Sin[k x], {k, 1, n}];
```

For every n, we have a Function s[x,n] which is a Fourier approximation of the function f:

```
Plot[{s[x, 9], f[x]}, {x, -Pi, Pi}]
```

If you plot several plots, you observe that while the Fourier approximation converges pointwise, there is an overshoot near the discontinuity. This is called the Gibbs phenomenon.

```
Plot[{f[x], s[x, 1], s[x, 20], s[x, 30]}, {x, -Pi, Pi}]
```

Partial Differential Equations

Here is how *Mathematica* can solve a partial differential equation. Here it is the wave equation. The result will be given in terms of variables C[1],C[2] which can be general functions:

```
ClearAll[u];
DSolve[{D[u[x, t], {t, 2}] == D[u[x, t], {x, 2}]}, u, {x, t}]
```

Note that the solution capabilities of *Mathematica* are rather limited in partial differential equations.
Here is a numerical computation of the one dimensional wave equation with inhomogeneous right hand side on the unit interval with boundary conditions zero at 0 and 1.

```
ClearAll[u]
s = NDSolve[{∂{t,2}u[x, t] == ∂{x,2}u[x, t] - 20 Sin[2 Pi x],
    u[x, 0] == Sin[4 Pi x] + Sin[6 π x], u^(0,1)[x, 0] == 0,
    u[0, t] == 0, u[1, t] == 0}, u, {x, 0, 1}, {t, 0, 2}];
Plot3D[Evaluate[u[x, t] /. s[[1]]], {t, 0, 1}, {x, 0, 1}]
```

Sound and Fourier

Instead plotting a function, you can also play it. A song is just a function.

```
Play[ Sin[1000 Sqrt[x]], {x, 0, 2}]
```

If you superimpose several functions, you can hear several "melodies" together.

```
w = Play[
  Abs[Sin[200 Sin[x^2]]] + Sin[1000 x] + Cos[2000 Sqrt[x]], {x, 0, 6}]
```

You can export it as a Wave file and use it as a cellphone ring tone ....

```
Export["sound.wav", w, "WAV"]
```

The smoothness of a function depends on how nice it sounds. A discontinuous function sounds horrible. The fourier series decays slowely.

```
Play[ Sign[Sin[1000 * x]], {x, 0, 2}]
```

Sound generation

Here is how you can play a tune with *Mathematica* with a given wave form. In the example below, we take the Sin function as the wave form and play a tune.

```
PlaySong[hull_, tune_, ground_] :=
 Module[{soundfilename, scale, songlength, frequency, song, P},
  scale[n_] := ground * 2^(n / 12); beatlength = 1 / 5;
  songlength = Length[tune] * beatlength;
  frequency[x_] := tune[[1 + Floor[x / beatlength]]];
  song[t_] := hull[scale[frequency[t]] * t];
  P = Play[song[t], {t, 0, songlength}]]

t1 = {0, 0, 0, 2, 1, 2, 1, 2, 1, 2, 0, 0, 0, 4, 10,
   7, 7, 5, 4, 2, 2, 0, 4, 7, 4, 0, 0, 0, 0, 0, 2, 0, 0};
f1[x_] := Abs[Sin[x]]; g1 = 2000;
PlaySong[f1, t1, g1]
```

Sound manipulation

Here is an involution on the linear space of all songs.

```
a = Import[
    "http://www.courses.fas.harvard.edu/~math21b/labs/yesterday.wav"]
type = a[[1, 2]]; n = Length[a[[1, 1]]];
song = {Table[a[[1, 1, 1, n - k]] / 2, {k, 0, n - 1}],
    Table[a[[1, 1, 2, n - k]] / 2, {k, 0, n - 1}]};
c = Sound[SampledSoundList[song, type]];
Export["yesterday2.wav", c, "WAV"]
Show[c]
```

Image manipulation

Quite a few image manipulation routines are built in. The first example is a filter. Every point is averaged with weights of neighboring points.

```
AA1 = Import[
  "http://www.courses.fas.harvard.edu/~math21b/labs/hall.jpg"];
Show[AA1]
AA2 = Dilation[ImageMultiply[LaplacianFilter[AA1, 1], 1], 1] //
  ImageAdjust
```

The next procedure splits the picture into small pieces of size 80 pixels and then produces a graph connecting closest pieces.

```
im = Flatten[ImagePartition[AA1, 80]];
cm = Map[Mean[Mean[ImageData[#]]] &, im];
nf = Nearest[cm → im];
GraphPlot[
 Flatten[Table[Thread[im[[i]] → nf[cm[[i]], 3]], {i, Length[im]}]],
 VertexRenderingFunction → (Inset[#2, #, Center, 0.5] &),
 SelfLoopStyle → None]
```

Here is an example on how to produce a movie morphing one picture to an other: the morph will take some time.

```
A1 = Import[
    "http://www.courses.fas.harvard.edu/~math21b/labs/mozart.jpg"];
A2 = Import[
    "http://www.courses.fas.harvard.edu/~math21b/labs/oliver.jpg"];
w = Length[A1[[1, 1]]]; h = Length[A1[[1]]]; ClearAll[S]
S[t_] := Image[(t A1[[1]] + A2[[1]] (1 - t)),
    Byte, ColorSpace → RGB, ImageSize → {w, h}];
Export["mix.gif", Table[S[k / 50], {k, 0, 50}], "GIF"]
```

Movie manipulation

The following few lines reads in a movie, identifies and colors morphological components and exports a new movie. This computation will take a while. We simplified procedures of http://blog.wolfram.com/2008/12/01/the-incredible-convenience-of-mathematica-image-processing and import and export
directly gif movies.

```
A = Import[
   "http://www.courses.fas.harvard.edu/~math21b/labs/geese.gif"];
n = Length[A]; w = Length[A[[1, 1, 1]]]; h = Length[A[[1, 1]]];
c = Prepend[
    Table[i → N[Chop[Append[Apply[List, ColorConvert[Hue[i / 2.3],
           RGBColor]], 0.5]]], {i, 100}], 0 → {0, 0, 0, 1}];
G[n_] := StringTake["00000" <> ToString[n], -4] <> ".gif";
T[B_] := Image[MorphologicalComponents[ColorNegate[B], 0.1] /. c,
    ColorSpace → "RGB"];
Export["geese1.gif", Table[T[A[[k]]], {k, n}],
 "GIF", ImageSize → {w, h}]
```

Auto Tune Pitch correction

The following procedures simulates "autotune". A sound file is split into pieces, a Fourier transform performed to cut out unwanted frequencies. Then the new sound is exported.

```
A = Import[
   "http://www.courses.fas.harvard.edu/~math21b/labs/yesterday.wav"
   ]; M = 5000; S = A[[1, 1, 1]]; n = Length[S];
tunes = Floor[Table[440 * 2^(k / 12), {k, -36, 55}]];
filter = Table[If[MemberQ[tunes, k]  || k > 400, 1, 0], {k, M}];
ClearAll[S1]; S1[l_] := Table[S[[l * M + k]], {k, M}];
ClearAll[S2]; S2[l_] := FourierDCT[filter * FourierDCT[S1[l], 2], 3];
S3 = Flatten[Table[S2[l], {l, 0, Floor[n / M] - 1}]];
U = Sound[SampledSoundList[{S3, S3}, 64000]];
 Export["yesterday1.wav", U, "WAV"]
Show[U]
```

Random music

Lets play some random tune:

```
T[x_] := Module[{}, y = x + RandomInteger[4] - 2; y];
s = NestList[T, 5, 100];
song1 = Sound[SoundNote[#, 1, RandomChoice[{"Cello"}]] & /@ s, 20]
```

You can export this into a midi file:

```
Export["song1.midi", song1, "Midi"]
```

Here is an example with random accords, where time is also chosen at random:

```
r := RandomChoice[{4, 7, 12}]
g[x_] :=
  RandomChoice[{0.5, 0.3, 0.2} → {x, {x[[1]]}, {x[[1]], x[[2]]}}];
accord := Module[{}, r1 = r; {{1 + r1, 5 + r1, 9 + r1}, {1, 5, 9}}];
scale = {1 + r, 5 + r, 9 + r};
s := g[RandomChoice[accord]] + RandomChoice[scale]
A = Table[{s, Random[Integer, 3] + 1}, {40}];
sn[{a_, t_}] := SoundNote[s, t];
song2 = Sound[{"Piano", sn /@ A}, 12]
```

```
Export["song2.midi", song2, "Midi"]
```

The *Mathematica* documentation gives the following cool example of a sound generation using cellular automaton 30:

```
sound3 =
 Sound[SoundNote[DeleteCases[3 Range[31] Reverse[#], 0] - 48, .1] & /@
   Transpose[CellularAutomaton[90, {{1}, 0}, 30]]]
Export["sound3.midi", sound3, "MIDI"]
```

Markov music

We are getting a bit off topic here but it illustrates an unexpected use of matrices. The following mathematica demo has been written for a talk given in February 2008. It demonstrates how to use a transition matrix to generate music which is less than random.

```
A = {(*c c# d d# e f f# g g# a
     a# h C C# D D# E F F# G G# A A# H*) (*C*)
    {1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0},
    (*C#*){0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0}, (*d*){0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, (*d#*)
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    (*e*){0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0}, (*f*){1, 0, 0, 0, 0, 1, 0, 1,
```

```
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0}, (*f#*)
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    (*g*){1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 1, 0, 0, 0, 0}, (*g#*){0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, (*a*)
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    (*a#*){0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0}, (*h*){0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, (*C*)
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    (*C#*){0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0}, (*D*){0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, (*D#*)
    {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
    (*E*){0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0}, (*F*){0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, (*F#*)
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    (*G*){1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 1, 0, 0, 1, 0}, (*G#*){0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0}, (*A*)
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    (*A#*){1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 1, 1, 0, 0, 0}, (*H*){0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
AT = Transpose[A]; T[x_] := RandomChoice[Position[AT[[x]], 1]][[1]];
markovchain[m_] := NestList[T, 1, m];
cminor = {1, 4, 8, 13}; cmajor = {1, 5, 8, 13}; e = {1, 1, 1, 1}; dim = 4;
cisminor = cminor + e; cismajor = cmajor + e;
dminor = cminor + 2 e; dmajor = cmajor + 2 e;
disminor = cminor + 3 e; dismajor = cmajor + 3 e;
eminor = cminor + 4 e; emajor = cmajor + 4 e;
fminor = cminor + 5 e; fmajor = cmajor + 5 e;
fisminor = cminor + 6 e; fismajor = cmajor + 6 e;
gminor = cminor + 7 e; gmajor = cmajor + 7 e;
gisminor = cminor + 8 e; gismajor = cmajor + 8 e;
aminor = cminor + 9 e; amajor = cmajor + 9 e;
aisminor = cminor + 10 e; aismajor = cmajor + 10 e;
hminor = cminor + 11 e; hmajor = cmajor + 11 e;
minors = {cminor, cisminor, dminor, disminor, eminor, fminor,
```

```
    fisminor, gminor, gisminor, aminor, aisminor, hminor};
majors = {cmajor, cismajor, dmajor, dismajor, emajor, fmajor,
    fismajor, gmajor, gismajor, amajor, aismajor, hmajor};
accordlist = Flatten[{minors, majors}];
accords = Partition[accordlist, dim]; songlength = 40;
sn[{s_, t_}] := SoundNote[s, t]; chain = markovchain[songlength];
aa1 = Table[sn[{accords[[chain[[k]]]], 1}], {k, songlength - 1}];
uu2 = Flatten[Table[{RandomSample[accords[[chain[[k]]]],
      RandomSample[accords[[chain[[k]]]]]}, {k, songlength}]];
uuu2 = Table[If[Random[] < 0.2, uu2[[k]] + 12, uu2[[k]]],
    {k, Length[uu2]}];
aa2 = Table[sn[{uuu2[[l]], 1 / (2 dim)}], {l, Length[uu2]}];
uu3 = Flatten[
    Table[RandomSample[accords[[chain[[k]]]]], {k, songlength}]];
aa3 = Table[sn[{uu3[[l]], 1 / dim}], {l, Length[uu3]}];

ss1 = Sound[Prepend[aa1, "Piano"]];
ss2 = Sound[Prepend[aa2, "Harp"]];
ss3 = Sound[Prepend[aa3, "Cello"]];
Export["niceaccords1.midi", ss1, "MIDI"];
```

```
Show[{ss1, ss2, ss3}]
```

Random poems

Well, now things are going really off topic. This random poem generator was adapted from Java program by Nandy Millan. http://www.cs.bham.ac.uk/~nxm/mscPoetry/Poetry/CGPoetry.html. It generates a poem, which can be considered as a vector of word vectors. What does it have to do with linear algebra? A more sophisticated poem generator would not do random choices but use a Markov chain approach to generate the sentences. Maybe you can build such a poet?

```
ClearAll["Global`*"];
Adjectives = {placid, mellow, nice, sad, happy,
    tight, noisy, cold, warm, beautiful, delicate, shining,
sweet, soft, graceful, pretty, captivating, desiring,
    languishing, fragile, exquisite, silver,
graceful, deafening, spiritless, heartbroken,
    sealed, stormy, carnal, frenetic, burning};
Adverbs = {soon, late, desperately, passionately, slowly,
    softly, carefully, tenderly, gently, quickly, kindly};
```

```
Articles = {the, a}; Ditverbs = {gave, told};
Opronouns = {me, you, him, her, us, them};
Spronouns = {i, you, he, she, we};
Itverbs = {cried, fell, died, came, anguished, shivered,
   surrendered, shouted, whispered, danced, dreamed};
Nouns = {heart, girl, mouth, soul, hand, saint, maiden,
   love, lover, flower, garden, sun, bird, breeze, rose,
   kiss, misery, despair, passion, desire, warmth,
   fire, flame, fantasy, wing, kingdom, light, shadow};
Preps = {from, for, with, in} ; Pronouns =
 {i, you, he, she, it, we, they, me, him, her, us, them};
Tverbs = {loved, kissed, touched, wanted,
   felt, desired, cuddled, fondled, enjoyed};
Verbs = {loved, escaped, rose, coped, sang, cried, hoped,
   fell, died, came, slept, shouted, kissed, touched, felt,
whispered, danced, desired, dreamed, cried, shouted};

Adjective := RandomChoice[Adjectives];
Adverb := RandomChoice[Adverbs]; Ditverb := RandomChoice[Ditverbs];
Itverb := RandomChoice[Itverbs]; Noun := RandomChoice[Nouns];
Opronoun := RandomChoice[Opronouns];
Prep := RandomChoice[Preps]; Pronoun := RandomChoice[Pronouns];
Spronoun := RandomChoice[Spronouns];
OPronoun := RandomChoice[OPronouns];
Tverb := RandomChoice[Tverbs]; Verb := RandomChoice[Verbs];
Article = RandomChoice[Articles]; Subject := RandomChoice[
  {{Article, Noun}, {Spronoun}, {Article, Adjective, Noun}}];
Predicatelist := {{Adverb}, {Prep, Article, Adjective, Noun},
   {Prep, Opronoun}, {Article, Adjective, Noun},
{Opronoun}, {Article, Adjective, Noun, Adverb},
   {Opronoun, Adverb}, {Opronoun, Article, Adjective, Noun}};
Verblist := {{Itverb}, {Itverb}, {Itverb}, {Tverb},
   {Tverb}, {Tverb}, {Tverb}, {Ditverb}};
Predicate := RandomChoice[Table[{Verblist[[i]], Predicatelist[[i]]},
    {i, Length[Verblist]}]];
Object := RandomChoice[{{}, {Adverb}, {Subject},
    {Opronoun}, {Prepsubject}}];
Verbobject := {Verb, Object}; Prepsubject := {Prep, Subject};
Subjectpredicate := {Subject, Predicate};
Continuation := RandomChoice[{{}, {"and", Subjectpredicate}}];
Sentence :=
```

```
 Flatten[RandomChoice[{{Subject, Verbobject, Continuation}}]];
Poem = Table[Sentence, {10}];
Poem
```

Assignment

To get full credit for this assignment, you have to hand in solutions to the following problems. We kept the assignment short. Once you get again a grip at *Mathematica*, it should take you not long to do it. You have sample code for each of the problems in the text above.

1) Plot the distribution of the eigenvalues of a random matrix, where each entry is a random number in [-0.4,0.6]. You can modify the given example in this lab but chosing random numbers with Random[]-0.4.

2) Find the solution of the ordinary differential equation f''[x]+f'[x]+f[x]== Cos[x]+x^4 with initial conditions f[0]==1;f'[0]== 0.

3) Find the Fourier series of the function f[x_]: = x^7 on [-Pi,Pi]. You need at least 20 coefficients of the Fourier series.

4) Fit the prime numbers data=Table[{k,Prime[k]/k]},{k,100000}] with functions {1,x,Log[x]}. The result will be a function. The result will suggest a growth rate of the prime numbers which is called the prime number theorem.

5) Freestyle: anything goes here. Nothing can be wrong. You can modify any of the above examples a bit to write a little music tune or poem or try out some image manipulation function. It can be very remotely related to linear algebra.

In order to work on your project it is a good idea to save this notebook first as a different document, do the assignment directly in that notebook and print out the relevant pages at the very end on a printer. The assignment hase to be printed out and turned in the last class of the semester. The printout does not to be colored but feel free to print it in color, if you like.