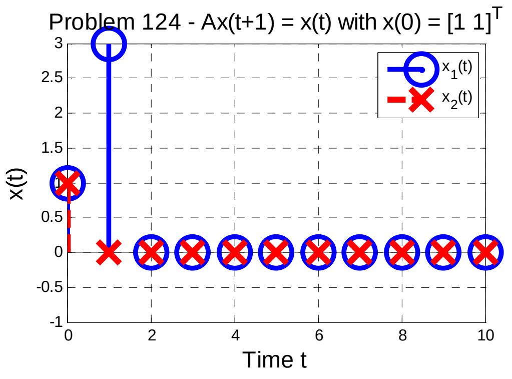
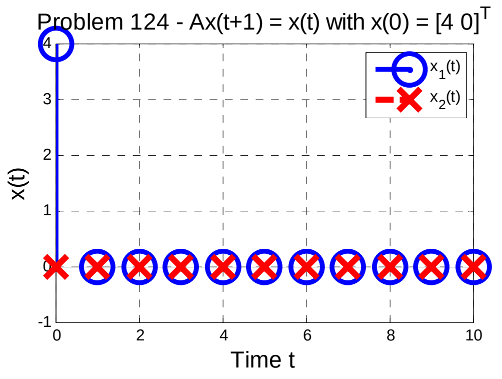
Christopher Tsai
June 4, 2007
EE 263 – Linear Dynamical Systems

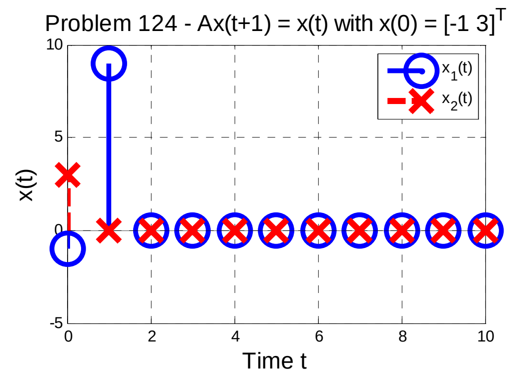
Problem Set VII

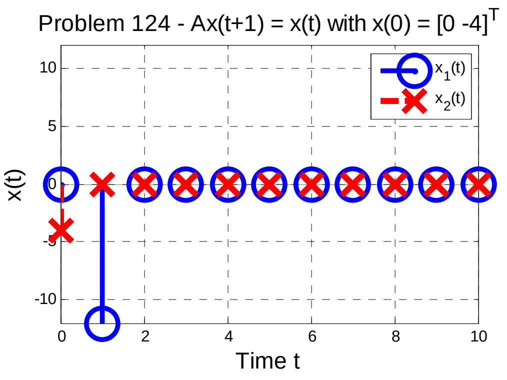
Problem #124: A Method for Rapidly Driving the State to Zero

As we computed in hand calculations, the system $x(t+1) = Ax(t)$ is stable because the zero eigenvalues all have less than unit magnitude; also, as noted in hand calculations, the dynamics terminate after only two iterations, since $A^2 = 0$, meaning that all trajectories end at zero within two seconds no matter where they begin:





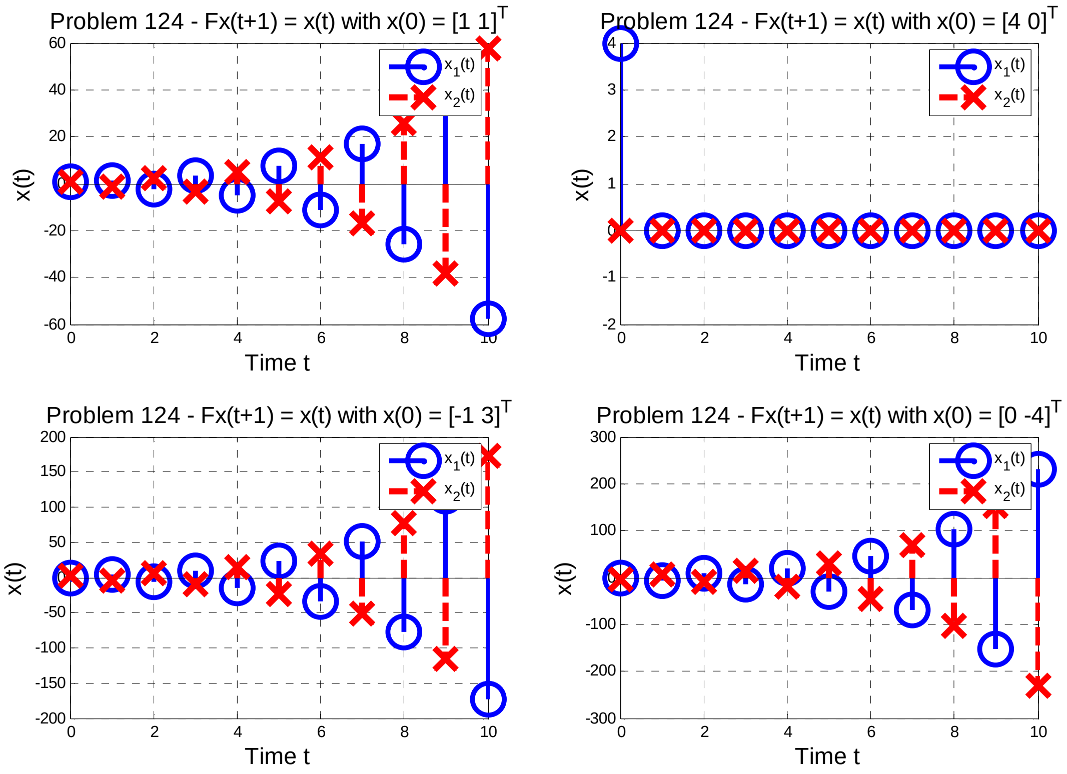




All four initial-value problems ascertain this theory; no matter which values $x_1(0)$ and $x_2(0)$ assume, they converge to zero by $t = 2$ at the latest.

Examining the discrete-time autonomous linear dynamical system $x(t+1) = Fx(t)$, however, yields no such stability. As we observed during eigenvalue calculation, the negative eigenvalue $-\frac{3}{2}$ leads the system to instability, resulting in rapidly increasing trajectory coordinates.

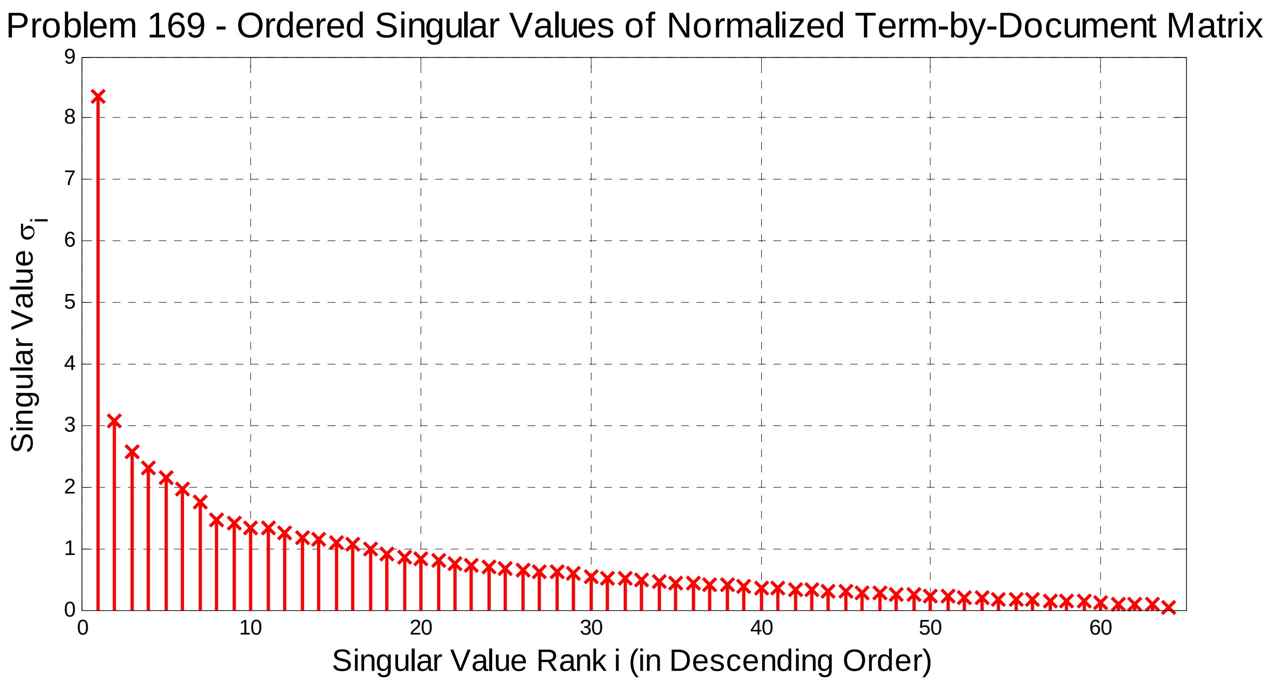
However, because the matrix power $F^t = \begin{bmatrix} 0 & -\left(-\frac{3}{2}\right)^t \\ 0 & \left(-\frac{3}{2}\right)^t \end{bmatrix}$ explodes only in the second column, or, put differently, only due to the second eigenvalue, trajectories that never initially excite the second mode will never diverge. For example, as the second (upper-right) initial condition of $\begin{bmatrix} 4 \\ 0 \end{bmatrix}$ reveals, the system can be stable when the trajectory begins with zero second component, or zero y-coordinate:



However, in all cases in which even the slightest perturbation occurs to the second value, the system quickly grows unstable, as the second eigenmode eventually dominates the first, pushing the system to ever-larger values of *x(t)*. Notice that the values of the coordinates oscillate in sign because the second eigenvalue is negative, which causes sign flipping between odd and even matrix powers. Overall, however, all trajectories resemble one another in their unstable oscillation except the second, which begins with no excitation of the second eigenmode (associated with $\lambda_2 = -\frac{3}{2}$.

Problem #169: The EE 263 Search Engine

After normalizing the term-by-document matrix A, we obtain the singular values:



Clearly, the first singular value (8.3465) dwarves the others, with the separation between successive singular values decreasing for the smaller values. The full list of singular values follows:

```
% singularValues =
%     8.3465
%     3.0859
%     2.5679
%     2.3155
%     2.1527
%     1.9630
%     1.7471
%     1.4710
%     1.4191
%     1.3439
%     1.3244
%     1.2565
%     1.1844
%     1.1595
%     1.0873
%     1.0793
%     0.9943
%     0.9126
%     0.8650
%     0.8269
%     0.7951
```

```
%     0.7419
%     0.7242
%     0.7099
%     0.6798
%     0.6358
%     0.6293
%     0.6157
%     0.5825
%     0.5389
%     0.5190
%     0.5142
%     0.4834
%     0.4707
%     0.4469
%     0.4381
%     0.4184
%     0.4150
%     0.3898
%     0.3633
%     0.3539
%     0.3343
%     0.3313
```

```
%     0.2988
%     0.2951
%     0.2828
%     0.2686
%     0.2510
%     0.2440
%     0.2258
%     0.2163
%     0.1913
%     0.1866
%     0.1781
%     0.1645
%     0.1608
%     0.1527
%     0.1480
%     0.1349
%     0.1246
%     0.1028
%     0.0951
%     0.0870
%     0.0301
```

Upon performing a query on the word 'students' and comparing the top five results with the results from queries on the low-rank approximations of normalized term-by-document matrix A, we obtain the top five results lists:

FULL RANK	RANK 32	RANK 16	RANK 8	RANK 4
103562	103562	103562	101252	101252
103561	103561	103563	103562	103561
103563	103563	103561	106106	103563
101252	101252	101252	103563	104173
100107	100107	100107	100107	100651

with the accompanying relevance values tabulated below:

FULL RANK	RANK 32	RANK 16	RANK 8	RANK 4
0.600035	0.644528	0.547383	0.375116	0.221239
0.551308	0.49516	0.451924	0.366579	0.196093
0.495772	0.472597	0.431237	0.246216	0.191225
0.493834	0.429912	0.391048	0.332164	0.189414
0.405854	0.384641	0.374455	0.299823	0.182386

Upon inspecting the search results, we see that the Rank 32 Approximation yields the same query results as the full rank search, although we begin to see the artifacts of approximation at the Rank 16 Approximation level, as the second and third most relevant returns no longer order correctly. However, the most relevant result – 103562 – still appears first, so this page ranking is nevertheless viable, even though we begin to detect some slippage in the decreased relevance of the top find. When we descend to the Rank 8 Approximation, the system begins to break down, with the top choice falling one slot and yielding its top position to the *fourth* most relevant result. Meanwhile, the previous irrelevant result 106106 materializes out of nowhere to appear as the third most relevant result, and the second-most relevant result falls two slots; the true ranking completely convolutes. Lowering the approximation to Rank 4 precipitates pandemonium, as the truly most relevant result – the one with the highest proportion of 'students' – completely vanishes from the top five list, while less relevant results such as 104173 appear despite their rightful position outside the top five.

All in all, it seems like excessively lowering the rank debilitates the accuracy of the full search, which properly orders the pages according to their relevance to the search query; however, some approximation – such as the Rank 32 and Rank 16 Approximations – may actually see little degradation from the original algorithm.

Of course, the low-rank approximations exchange compromised accuracy for boosted speed, as the lower-rank approximations permit offline calculations and decrease the number of multiplication operations necessary during a search. For one, with all its rows and columns included in the singular value decomposition, the full rank approximation requires *nm* operations, totaling (128)(64) = 8192 operations in our particular Matlab example. Meanwhile, with reduced nonzero rows, the Rank *r* Approximation requires only *r*(*n* + *m*) operations, which could be comparatively smaller when juxtaposed beside the full rank count. For example, the Rank 32 Approximation necessitates only 32(128 + 64) = 6144 operations, while the Rank 16 Approximation, slightly compromised but nonetheless reliable, consumes only 16(128 + 64) = 3072 calculations, less than half that required by the full rank search. One can imagine that, for larger values of *n* and *m* – more complicated search terms over a wider array of documents – the computational advantage of a low-rank approximation can vastly expedite search.

Assuming that a very large number of searches are performed before the term-by-document matrix is updated, the singular value decomposition can be performed offline, allowing simple plug-substitution of the query into a matrix to involve fewer computations:

$$\mathrm{c} = \widetilde{\mathrm{A}}^{\mathrm{T}}\tilde{\mathrm{q}}$$

Resolving $\widetilde{\mathrm{A}}^{\mathrm{T}}$ into its singular value decomposition, $\widetilde{\mathrm{A}}^{\mathrm{T}} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}$, we can write the relevance as

$$\mathrm{c} = (\mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}})^{\mathrm{T}}\tilde{\mathrm{q}}$$

$$\mathrm{c} = \mathrm{V}\Sigma\mathrm{U}^{\mathrm{T}}\tilde{\mathrm{q}}$$

The first Cayley product – $V\Sigma$ – always yields the same calculations for the same term-by-document matrix, so we can perform them offline, while the portion $U^T\tilde{q}$ processes the query before premultiplication by the pre-computed matrix. This pre-computation saves online runtime, allowing the Rank *r* Approximation to return in only $r(n + m)$ operations, a vast savings for large term-by-document matrices in the World Wide Web. The downside, of course, is that the lower we make the rank, the more relevance-ordering accuracy we sacrifice.