Carry out the solution of each problem: show steps of any required calculations; state reasons that justify any conclusions. Mere oracular answers will receive no credit.

1. Give a MATLAB command that creates a $2 \times 128$ matrix $\text{Dice}$ of random integers from one to six (inclusive), representing 128 throws of a pair of dice.

Given the array $\text{Dice}$ described above, what does the following MATLAB fragment do?

```matlab
eights = 0;
for i = 1:128
    if (Dice(1,i)+Dice(2,i) == 8)
        eights = eights + 1;
    end
end
```

Give a one-line MATLAB command that calculates the quantity $\text{eights}$ without a for loop.

2. Using only the information in the following table, give three different ways to approximate the derivative $\Gamma'(1)$. Which approximation do you expect to be the most accurate?

<table>
<thead>
<tr>
<th>$x$</th>
<th>$\Gamma(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{3}{2}$</td>
<td>$\sqrt{\pi}$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$\frac{5}{2}$</td>
<td>$\frac{1}{2} \sqrt{\pi}$</td>
</tr>
<tr>
<td>$2$</td>
<td>$1$</td>
</tr>
</tbody>
</table>

3. Find the Newton form of the cubic polynomial that interpolates the data in the table below.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>$1$</td>
<td>$1$</td>
<td>$1$</td>
<td>$25$</td>
</tr>
</tbody>
</table>
4. Both function M-files shown here, downloaded from Van Loan’s web site, compute the matrix product \( y = A \cdot x \). Although they obviously use different algorithms, the descriptive comments are identical! Edit the comments so that they correctly describe the algorithms used.

```matlab
function y = MatVecR0(A,x)    function y = MatVecC0(A,x)
% y = MatVecR0(A,x)            % y = MatVecC0(A,x)
% Computes the matrix-vector    % This computes the matrix-vector
% product \( y = A \cdot x \) (via saxpys)  % product \( y = A \cdot x \) (via saxpys)
% where \( A \) is an \( m \)-by-\( n \) matrix  % where \( A \) is an \( m \)-by-\( n \) matrix
% and \( x \) is a column \( n \)-vector.  % and \( x \) is a column \( n \)-vector.
[m,n] = size(A);             [m,n] = size(A);
y = zeros(m,1);             y = zeros(m,1);
for i=1:m                   for j=1:n
    y(i) = A(i,:)*x;          y = y + A(:,j)*x(j);
end                           end
```

5. On pages 25–26 Van Loan gives four ways to set up an \( n \times m \) matrix \( A \) with \( A(k,j) = \sin(jx_k) \). The script M-file `SumOfSines.m` in the middle of page 25 handles the case \((n,m) = (200, 4)\) without a for loop. Why would that scheme be awkward to program if \( m = 25 \)?

In §5.3 we learned that MATLAB’s built-in functions accept matrix arguments. Give a one-line MATLAB command that computes \( A \) as the sine of an outer product matrix.

[Exam begins on Obverse]