An appropriate substitution here is
\[ t = \sqrt{x} \quad \Rightarrow \quad dt = \frac{1}{2\sqrt{x}} \, dx, \quad \text{or} \quad dx = 2t \, dt. \]

Thus,
\[
\int_0^1 \frac{e^x}{\sqrt{x}} \, dx = 2 \int_0^1 e^{t^2} \, dt \overset{!}{=} 2(1.4627) = 2.9253,
\]
with this approximate value coming from our g1Quad() routine.

**Answer:**

First, we make our substitutions so that each iterated integral is carried out over the region \([-1, 1]:\)

\[ x = \frac{1}{2} \left( u + 1 \right) \quad \Rightarrow \quad dx = \frac{1}{2} \, du, \]
\[ y = \frac{\pi}{2} \left( v + 1 \right) \quad \Rightarrow \quad dx = \frac{\pi}{2} \, dv, \]
\[ z = \frac{1}{2} \left( w + 3 \right) \quad \Rightarrow \quad dx = \frac{1}{2} \, dw. \]

Thus,
\[
\int_0^1 \int_0^\pi \int_1^2 x^2 \sin(xy \sqrt{z}) \, dz \, dy \, dx = \frac{\pi}{32} \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 (u+1)^2 \sin\left(\frac{\pi}{4}(u+1)(v+1)\sqrt{\frac{w+3}{2}}\right) \, dw \, dv \, du.
\]

My script (i.e., .m-file) to carry out the computation is:

```matlab
curr = cputime();
n = 6;
[xns, wts] = GLNodeWt(n);

function out = f(u,v,w)
    out = (u + 1).^2 .* sin(pi*(u+1).*(v+1).*sqrt((w+3)/2)/4) * pi/32;
end

S = 0;
for ii = 1:n
    for jj = 1:n
        for kk = 1:n
```

```matlab
end
end
end
S
```
S = S + f(xns(ii), xns(jj), xns(kk)) * wts(ii) * wts(jj) * wts(kk);  
end  
end  
end  
S  
cputime() - curr

producing the output

S = 0.63806  
an = 0.037283

(The first and last lines of code, both of which make calls to the Octave function cputime(), are to determine how many seconds of CPU processing time have been used to carry out the computation. Here, it is less than 4-hundredths of a second.)

\*37 (a) Let \((t, x_1, \ldots, x_m), (t, y_1, \ldots, y_m)\) be points in \(D\). We note that \(D\) is convex, and that

\[
|f(t, x_1, \ldots, x_m) - f(t, y_1, \ldots, y_m)| = |f(t, x_1, \ldots, x_m) - f(t, x_1, \ldots, x_{m-1}, y_m) + f(t, x_1, \ldots, x_{m-1}, y_m) - f(t, x_1, \ldots, x_{m-2}, y_{m-1}, y_m) + \cdots + f(t, x_1, y_2, \ldots, y_m) - f(t, y_1, \ldots, y_m)|
\]

\[
= \left| \frac{\partial f}{\partial x_m}(t, x_1, \ldots, x_{m-1}, \xi_m)(x_m - y_m) + \frac{\partial f}{\partial x_{m-1}}(t, x_1, \ldots, \xi_{m-1}, y_m)(x_{m-1} - y_{m-1}) + \cdots + \frac{\partial f}{\partial x_1}(t, \xi_1, y_2, \ldots, y_m)(x_1 - y_1) \right| \quad \text{(by the MVT; } \xi_j \text{ between } x_j, y_j) 
\]

\[
\leq \left| \frac{\partial f}{\partial x_m}(t, x_1, \ldots, x_{m-1}, \xi_m) \right| |x_m - y_m| + \left| \frac{\partial f}{\partial x_{m-1}}(t, x_1, \ldots, \xi_{m-1}, y_m) \right| |x_{m-1} - y_{m-1}| + \cdots + \left| \frac{\partial f}{\partial x_1}(t, \xi_1, y_2, \ldots, y_m) \right| |x_1 - y_1| \quad \text{(by triangle inequality)}
\]

\[
\leq L |x_m - y_m| + L |x_{m-1} - y_{m-1}| + \cdots + L |x_1 - y_1| 
\]

\[
= L \sum_{j=1}^{m} |x_j - y_j|. 
\]

(b) The word equivalent really means tautology here, in the sense that if we know each \(f_j\) is Lipschitz on \(D\), then we know \(\|f(t, x) - f(t, y)\|_\infty \leq M \|x - y\|_1\), and the
latter implies the former as well. To prove this, let us first assume that each $f_k$ is Lipschitz. That is, there exist numbers $L_k > 0, k = 1, \ldots, m$ such that

$$|f_k(t, x_1, \ldots, x_m) - f_k(t, y_1, \ldots, y_m)| \leq L_k \sum_{j=1}^{m} |y_j - x_j|. \quad (1)$$

Define $M = \max_{1 \leq k \leq m} L_k$. Then clearly (1) implies

$$|f_k(t, x_1, \ldots, x_m) - f_k(t, y_1, \ldots, y_m)| \leq M \sum_{j=1}^{m} |y_j - x_j| = M\|x - y\|_\infty, \quad (2)$$

where the expression on the RHS is a uniform bound independent of $k$. Since this bound holds for each $k = 1, \ldots, m$, it must hold for the maximum entry, in absolute value, from $f(t, x) - f(t, y)$, which is another way of saying

$$\|f(t, x) - f(t, y)\|_\infty \leq M\|x - y\|_1. \quad (3)$$

We omit the details of how, if one assumes the inequality (3), one might show each $f_j(t, x)$ is Lipschitz on $D$. This is pretty easy to see, particularly if you understand the half that we have proved already.

AP 10.6 (a) With predefined functions like

```octave
function y = f(theta)
    y = 1;
end
```

```octave
function y = kern(r, theta, phi)
    y = (1 - r.^2) ./ (1 - 2*r.*cos(phi - theta) + r.^2) / (2*pi);
end
```

we may use a command like

```octave
glQuad(@(phi) kern(.5, pi/2, phi).*f(phi), 0, 2*pi, 10)
ans = 1.0036
```

(along with the glQuad() routine from PS16) to evaluate the steady-state temperature distribution at $(r, \theta) = (1/2, \pi/2)$. Here are the values at several other points: $(0.1, \pi/8), (0.95, 1), (0.62, 2.1), \text{and} (0.33, 5)$.

```octave
glQuad(@(phi) kern(.1, pi/8, phi).*f(phi), 0, 2*pi, 10)
an = 1.0000
```

```octave
glQuad(@(phi) kern(.95, 1, phi).*f(phi), 0, 2*pi, 10)
an = 4.2683
```

---

**PS17—Solutions**
You see I re-did the calculation for the point \((r, \theta) = (0.95, 1)\) incorporating more nodes to get an answer close to 1.

(b) Now we make some changes and calculate some differences of the form \(r^2 (\cos^2 \theta - \sin^2 \theta) - u(r, \theta)\), differences we expect to be near zero.

\[
\text{octave:71} \> \text{function } y = f(\theta) \\
> y = \cos(\theta)^2 - \sin(\theta)^2;
> \text{end}
\]

\[
\text{octave:73} \> \text{function } y = \text{expectedDist}(r, \theta) \\
> y = r^2 \cdot (\cos(\theta)^2 - \sin(\theta)^2);
> \text{end}
\]

\[
\text{octave:74} \> \text{expectedDist}(0.33, 5) - \text{glQuad}(\theta) \text{kern}(0.33, 5, \phi) \cdot f(\phi), 0, 2\pi, 10) \\
> \text{ans} = -8.9553e-04
\]

\[
\text{octave:75} \> \text{expectedDist}(0.1, 2) - \text{glQuad}(\theta) \text{kern}(0.1, 2, \phi) \cdot f(\phi), 0, 2\pi, 10) \\
> \text{ans} = -2.7089e-05
\]

\[
\text{octave:76} \> \text{expectedDist}(0.8, 3.2) - \text{glQuad}(\theta) \text{kern}(0.8, 3.2, \phi) \cdot f(\phi), 0, 2\pi, 10) \\
> \text{ans} = 0.39191 \quad \% \text{r is too close to 1; better use more nodes}
\]

\[
\text{octave:77} \> \text{expectedDist}(0.8, 3.2) - \text{glQuad}(\theta) \text{kern}(0.8, 3.2, \phi) \cdot f(\phi), 0, 2\pi, 30) \\
> \text{ans} = 0.013749
\]

(c) The commands

\[
\text{octave:78} \> \text{function } y = f(\theta) \\
> y = \theta \cdot (2\pi - \theta);
> \text{end}
\]

\[
\text{octave:79} \> \text{thetas} = 0:.1:2\pi;
\]

\[
\text{octave:87} \> \text{distAtThetas} = [ ];
\]

\[
\text{octave:88} \> \text{for jj} = 1:\text{length} \text{(thetas)} \\
> \text{distAtThetas} = \text{[distAtThetas glQuad}(\theta) \text{kern}(0.2, \text{thetas(jj)}, \phi) \cdot f(\phi), 0, 2\pi, 10]));
> \text{end}
\]

\[
\text{octave:89} \> \text{subplot}(2,2,1) \\
\text{octave:90} \> \text{plot} \text{(thetas, distAtThetas)} \\
\text{octave:91} \> \text{title('u(0.2, \theta)', 'fontsize', 22)
\]

\[
\text{PS17—Solutions}
\]
produce the plot in the upper left corner. Using commands similar to 88–91 (changing only the value of \( r \), which subplot receives the plot and, for the larger \( r \)-values, increasing the number of nodes used), we produce the other three plots as requested.

![Graphs of u(0.2, \( \theta \)), u(0.4, \( \theta \)), u(0.6, \( \theta \)), u(0.8, \( \theta \))](image)

We note that both the peaks and low points become more extreme as \( r \) gets closer to 1.

As for the value of \( u(0, \theta) \), we may evaluate directly or, if we do not trust the answer, we may choose a path towards the origin (like a line with constant \( \theta \)) and look at the value of \( u(r, \theta) \) as \( r \to 0 \). The commands below involve direct evaluation of \( u(0, \theta) \) and the “limit” along two straight-line paths to \((0, 0)\).

```octave
octave:119> glQuad(@(phi) kern(0, thetas(jj), phi).*f(phi), 0, 2*pi, 10)
ans = 6.5797
octave:120> rs = [.1 .01 .001 .0001];
octave:121> for jj = 1:4
>     glQuad(@(phi) kern(rs(jj), 0, phi).*f(phi), 0, 2*pi, 20)
> end
ans = 6.1693
ans = 6.5396
ans = 6.5757
ans = 6.5793
```

---

**PS17—Solutions**
octave:122> for jj = 1:4
> glQuad(@(phi) kern(rs(jj), 2.1, phi).*f(phi), 0, 2*pi, 20)
> end
ans = 6.7861
ans = 6.6000
ans = 6.5818
ans = 6.5799

All of these suggest \( u(0, \theta) \approx 6.5797 \).

(d) We get the following table of values

\[
\begin{array}{c|c|c|c|c}
& \theta = 0 & \theta = \pi/4 & \theta = 5\pi/6 & \theta = 3\pi/2 \\
\hline
r = 0.9 & 1.3809 & 4.4404 & 9.3288 & 7.2669 \\
r = 0.95 & 0.8172 & 4.3753 & 9.4630 & 7.3337 \\
r = 0.99 & 0.2252 & 4.3288 & 9.5691 & 7.3884 \\
\end{array}
\]

which is consistent (though by no means conclusive evidence) with the claims about \( u(r, \theta) \) always being between 0 and \( \pi^2 \approx 9.8696 \).