Solutions for the Hundred-Dollar, Hundred-Digit Challenge

29 April 2002

Kim McInturff
Raytheon
Goleta, CA
mcint@esd.ray.com

Peter S. Simon
Space Systems/Loral
Palo Alto, CA
peter_simon@ieee.org

Certification of Originality

We certify that we developed these solutions ourselves, without assistance from any former students of Oxford or from anyone else.

Summary of Numeric Answers

Here we summarize the ten "magic" numbers. In each case, we report the solution using the number of significant digits that we believe is correct.

Problem 1: 0.32336743167778

Problem 2: 0.995262919443354

Problem 3: 1.27422415282

Problem 4: -3.3068686474752

Problem 5: 0.214335234590

Problem 6: ± 0.061913954473991

Problem 7: 0.72507834626840

Problem 8: 0.42401138703369

Problem 9: 0.78593367435037

Problem 10: $3.8375879792512 \times 10^{-7}$

Description of Solutions

1 Problem 1

What is $\lim_{\epsilon \to 0} \int_{\epsilon}^{1} x^{-1} \cos(x^{-1} \log x)\, dx$?

1.1 Solution Method

After introducing the change of variables $u = -\log x$ the integral becomes

$$\int_0^{\infty} \cos(ue^u)\, du.$$

Zeros of the integrand are located at the points $\{u_m\}$, where u_m is the root of

$$u_m e^{u_m} - (2m-1)\frac{\pi}{2} = 0, \quad (m = 1, 2, 3, \ldots).$$

The first 30 roots were determined using the MATLAB routine `fzero`. The integral was converted to an alternating series by integrating along segments bounded by adjacent roots:

$$S = \sum_{m=1}^{\infty} \int_{u_{m-1}}^{u_m} \cos(ue^u)\, du,$$

where $u_0 = 0$. The terms in the series were computed using the MATLAB function `quadl` and the resulting slowly converging alternating series was accelerated using repeated averaging on the last 20 partial sums.

1.2 Code Listing

The solution to this problem was coded in MATLAB. It consists of the single file `problem1.m` listed below.

```
function  problem1
%
% Peter Simon
% 3/1/2002
%
Nterm = 30;  % # terms in the alternating series to compute.
Navg =  20; % # partial sums to include in the averaging process.
```

```
[terms,nodes] = findterms(Nterm); % Obtain alternating terms to be
                                  % summed.
psum = cumsum(terms); % Obtain partial sums of series.
lozenge = zeros(Navg,Navg); % Set up computational lozenge.
lozenge(:,1) = psum(Nterm-Navg+1:end); % 1st col is original partial sum.
for i = 2:Navg  % Loop over the columns of the lozenge
  % Perform averaging:
  lozenge(1:(Navg-i+1), i) ...
      = (lozenge(1:(Navg-i+1),i-1) + lozenge(2:(Navg-i+2),i-1)) / 2;
end
% Display results:
disp(' ')
disp('Zeros of the integrand: ')
fprintf('%18.14f \n\n\n', nodes);
disp('Terms in the resulting alternating series: ')
fprintf('%18.14f \n\n\n', terms);
disp('Partial sums of the alternating series: ')
fprintf('%18.14f \n\n\n', psum);
disp('Last four columns of the repeated averaging lozenge:')
for i = 1:4
  fprintf('%18.14f ', lozenge(i,end-3:end-i+1))
  fprintf('\n')
end
return

function [terms,nodes] = findterms(Nterm)
% Find Nterm terms in the series to be summed.
nodes = findnodes(Nterm); % Obtain the integrand zeros.
terms = zeros(Nterm,1); % Preallocation
for i = 1:Nterm
  if i == 1
    a = 0;
  else
    a = nodes(i-1);
  end
  b = nodes(i);
  % Integrate over the panel (a,b):
  terms(i) = quadl(@integrand, a, b, eps);
end
return

function y = integrand(u)
y = cos(u .* exp(u));
return

function u = findnodes(Nu)
% Find Nu nodes (locations where the integrand changes sign).
u = zeros(Nu,1);  % Preallocation
options = optimset('fzero'); % Get default options.
% Set function tolerance to machine epsilon:
options = optimset(options, 'TolFun', eps);
for i = 1:Nu
  p = (2*i-1) * pi/2;
  [u(i), fval, exitflag] = fzero(@f,[0 Nu],options,p);
  if exitflag < 0
```

```
    error('Bad exitflag in findnodes!')
  end
end
return

function y = f(x,p)
% Function used to find zeros of the integrand.
y = x .* exp(x) - p;
return
```

2 Problem 2

A photon moving at speed 1 in the x–y plane starts at $t = 0$ at $(x, y) = (0.5, 0.1)$ heading due east. Around every integer lattice point (i, j) in the plane, a circular mirror of radius $1/3$ has been erected. How far from the origin is the photon at $t = 10$?

2.1 Solution Method

We use bold face variables to represent two-vectors, and a caret accent to denote unit vectors. Thus, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are unit vectors in the x and y directions, respectively. Furthermore, we assume the standard convention that east and north correspond to $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, respectively. The starting point is $\boldsymbol{r}_0 = x_0\hat{\boldsymbol{x}} + y_0\hat{\boldsymbol{y}} = 0.5\hat{\boldsymbol{x}} + 0.1\hat{\boldsymbol{y}}$. Denote by $\{M_i\}_{i=1}^{\infty}$ the sequence of mirrors with which the photon collides, with $\boldsymbol{r}_i = x_i\hat{\boldsymbol{x}} + y_i\hat{\boldsymbol{y}}$ being the point of collision and $\overline{\boldsymbol{r}}_i = \overline{x}_i\hat{\boldsymbol{x}} + \overline{y}_i\hat{\boldsymbol{y}}$ the center of M_i. Let $\hat{\boldsymbol{\ell}}_i$ denote the direction the photon travels between $\boldsymbol{r}_{i-1}$ and $\boldsymbol{r}_i$. At each collision point $\boldsymbol{r}_i$ let $\{\hat{\boldsymbol{u}}_i, \hat{\boldsymbol{v}}_i\}$ be a local coordinate system defined by

$$\hat{\boldsymbol{u}}_i = \frac{\boldsymbol{r}_i - \overline{\boldsymbol{r}}_i}{|\boldsymbol{r}_i - \overline{\boldsymbol{r}}_i|}, \qquad \hat{\boldsymbol{u}}_i \times \hat{\boldsymbol{v}}_i = \hat{\boldsymbol{z}}.$$

Specifically,

$$\begin{aligned}\hat{\boldsymbol{u}}_i &= u_{xi}\hat{\boldsymbol{x}} + u_{yi}\hat{\boldsymbol{y}} = 3(x_i - \overline{x}_i)\hat{\boldsymbol{x}} + 3(y_i - \overline{y}_i)\hat{\boldsymbol{y}},\\ \hat{\boldsymbol{v}}_i &= v_{xi}\hat{\boldsymbol{x}} + v_{yi}\hat{\boldsymbol{y}} \quad \text{where} \quad v_{xi} = -u_{yi},\ v_{yi} = u_{xi}.\end{aligned}$$

The direction after collision with mirror M_i is given by

$$\hat{\boldsymbol{\ell}}_{i+1} = -(\hat{\boldsymbol{\ell}}_i \cdot \hat{\boldsymbol{u}}_i)\hat{\boldsymbol{u}}_i + (\hat{\boldsymbol{\ell}}_i \cdot \hat{\boldsymbol{v}}_i)\hat{\boldsymbol{v}}_i.$$

Clearly, $\boldsymbol{r}_1 = (1 - \frac{1}{3}\sqrt{0.91})\hat{\boldsymbol{x}} + 0.1\hat{\boldsymbol{y}}$, and the time at which the photon reaches mirror M_1 is $t_1 = (\frac{1}{2} - \frac{1}{3}\sqrt{0.91})$. For $i \geq 2$ the time t_i between collisions with mirrors M_{i-1} and M_i is found as the solution of a quadratic equation:

$$t_i = B - \sqrt{B^2 - C} \qquad \text{where} \qquad B = \hat{\boldsymbol{\ell}}_i \cdot (\overline{\boldsymbol{r}}_i - \boldsymbol{r}_{i-1}),\ \ C = |\overline{\boldsymbol{r}}_i - \boldsymbol{r}_{i-1}|^2 - \frac{1}{9},$$

and the point of collision on M_i is $\boldsymbol{r}_i = \boldsymbol{r}_{i-1} + t_i \hat{\boldsymbol{\ell}}_i$.

In the first ten seconds, the photon hits 14 circular mirrors, specified by the matrix of center coordinates

$$C = \begin{bmatrix} \overline{x} \\ \overline{y} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & -1 & -1 \\ 0 & 1 & 2 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & -1 \end{bmatrix}$$

The photon hits the fourteenth mirror M_{14} at time $t = 9.89050300334379$ at the point

$$\boldsymbol{r}_{14} = -0.83875273011336\hat{\boldsymbol{x}} - 0.70826308244376\hat{\boldsymbol{y}}.$$

At time $t = 10$ the photon is at

$$\boldsymbol{r} = -0.70826308244376\hat{\boldsymbol{x}} - 0.66964269636357\hat{\boldsymbol{y}}$$

and the distance from the origin is

$$|\boldsymbol{r}| = 0.995262919443354.$$

A plot of the photon's trajectory is shown in Figure 2.1.

2.2 Code Listing

The solution to this problem was first coded in MATLAB. We then found that because of the ill-conditioned nature of the problem, it was necessary to code it in a language supporting greater precision. The solution shown below was coded in an extension to Fortran 77 that supports sixteen-byte ("quad precision" or "real*16") floating point variables. The answer we obtained from the quad precision code was checked against one found using a multiprecision code [4] set to sixty decimal digits of accuracy. The solutions agree to more than 20 digits.

```
      program prob2
c  quad precision version.  intended for DEC Alpha machine.

      implicit none

      real*16  x0, y0

      real*16  lx, ly
      dimension lx(15), ly(15)

      real*16  x, y, t, ttotal
      dimension x(15), y(15), t(15)

      integer k, kp1

      real*16  ux, uy, vx, vy
      dimension ux(14), uy(14), vx(14), vy(14)
```

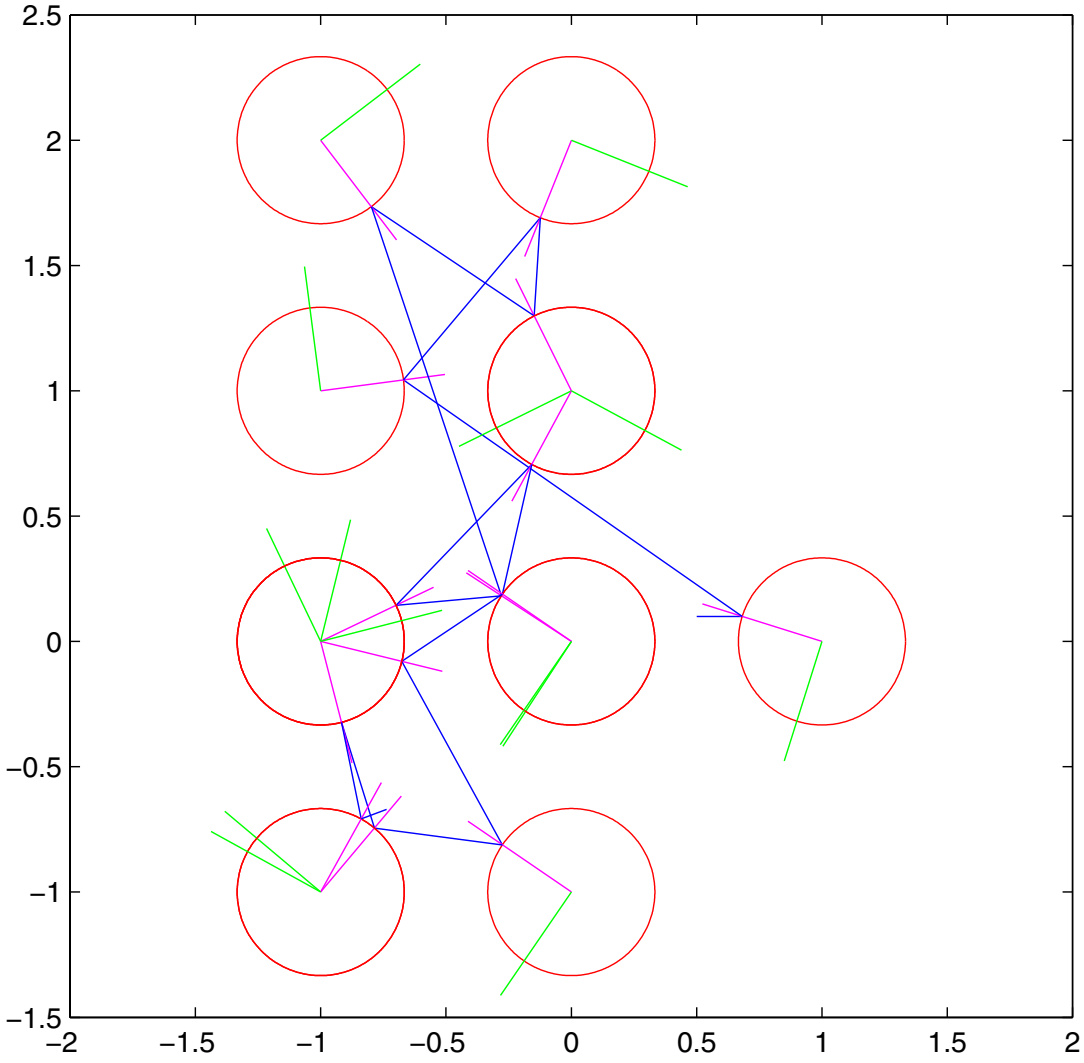

Figure 2.1: Photon trajectory for Problem 2.

```
      real*16  b, c, d

      integer xbar(14), ybar(14)
      data xbar /1, -1, 0, 0, -1, 0, -1, 0, 0, -1,  0, -1, -1, -1/
      data ybar /0,  1, 2, 1,  2, 0,  0, 1, 0,  0, -1, -1,  0, -1/

      x0 = 0.5q0
      y0 = 0.1q0
      lx(1) = 1.q0
      ly(1) = 0.q0
      x(1) = 1.q0 - sqrt(0.91q0)/3.q0
      y(1) = 0.1q0

      t(1) = x(1) - x0
      ttotal = t(1)

      do k=1,14
        ux(k) = 3.q0*(x(k)-xbar(k))
        uy(k) = 3.q0*(y(k)-ybar(k))

        vx(k) = -uy(k)
        vy(k) = ux(k)

        kp1 = k + 1
        lx(kp1) = -(lx(k)*ux(k)+ly(k)*uy(k))*ux(k)
     *             + (lx(k)*vx(k)+ly(k)*vy(k))*vx(k)
        ly(kp1) = -(lx(k)*ux(k)+ly(k)*uy(k))*uy(k)
     *             + (lx(k)*vx(k)+ly(k)*vy(k))*vy(k)

        if (k.lt.14) then
          b = lx(kp1)*(x(k)-xbar(kp1)) + ly(kp1)*(y(k)-ybar(kp1))
          c = (x(k)-xbar(kp1))**2 + (y(k)-ybar(kp1))**2 - 1.q0/9.q0
          t(kp1) = -b-sqrt(b*b-c)
          ttotal = ttotal + t(kp1)
        else
          t(kp1) = 1.0q1 - ttotal
          ttotal = 1.0q1
        end if
        x(kp1) = x(k) + t(kp1)*lx(kp1)
        y(kp1) = y(k) + t(kp1)*ly(kp1)
      end do

      do k=1,15
        write(6,*) ' k, t = ', k, t(k)
      end do
      do k=1,15
        write(6,*) ' k, x = ', k, x(k)
      end do
      do k=1,15
        write(6,*) ' k, y = ', k, y(k)
      end do

      d = sqrt(x(15)**2+y(15)**2)
      write(6,*) ' d = ', d

c  d = 0.99526291944335
```

```
stop
end
```

3 Problem 3

The infinite matrix A with entries $a_{11} = 1$, $a_{12} = 1/2$, $a_{21} = 1/3$, $a_{13} = 1/4$, $a_{22} = 1/5$, $a_{31} = 1/6$, etc., is a bounded operator on ℓ^2. What is $\|A\|$?

3.1 Solution Method

In general the matrix elements a_{mn} of A satisfy

$$\frac{1}{a_{mn}} = \frac{m(m+1)}{2} + \sum_{k=2}^{n}(m+k-2).$$

Let $\{s_m\}_{m=1}^{\infty}$ denote the sequence of singular values of A, arranged in decreasing magnitude. The norm of A, $||A||$, is s_1, the largest singular value. We may approximate the norm of A as the limit of norms of truncated $N \times N$ matrices A_N obtained by taking the first N rows and columns of A. That is,

$$||A|| = \lim_{N\to\infty} ||A_N||.$$

In particular, taking $N = 12000$ yields

$$||A_N|| = 1.27422415281846,$$

and by examining the rate at which $||A_N||$ is changing, we may conclude that

$$||A|| = 1.27422415282.$$

3.2 Code Listing

The solution to this problem was coded in MATLAB. It consists of the single file `problem3.m` listed below.

```
N=12000
A=zeros(N);
for m=1:N
  A(m,1)=m*(m+1)/2;
  for n=2:N
    A(m,n)=A(m,n-1)+m+n-2;
  end
end
A=1./A;
S=svd(A);
```

```
specradius=S(1)
Frobeniusnorm=norm(S)
S(1)=0;
tailnorm=norm(S)

% pi/sqrt(6) = 1.28254983016186

%    1 1                 0                 1
%    2 1.18335017655166  0.02816861314076  1.18368539363765
%    3 1.23364450098849  0.05829213364944  1.23502094219675
%    4 1.25253739751680  0.07961122325431  1.25506489037269
%    5 1.26121761618336  0.09429867303881  1.26473796302164
%    6 1.26577861497541  0.10465791170738  1.27009794134626
%    7 1.26841343166642  0.11219049285116  1.27336536010604
%    8 1.27004630585408  0.11782647267371  1.27550017509850
%    9 1.27111452000473  0.12215019277009  1.27697016118648
%   10 1.27184401925072  0.12553938623811  1.27802478332812
%   11 1.27235992531705  0.12824570375436  1.27880676416895
%   12 1.27273551126191  0.13044161325622  1.27940247619583
%   13 1.27301571056500  0.13224837945970  1.27986664665309
%   14 1.27322916385796  0.13375323548374  1.28023530325515
%   15 1.27339473723705  0.13502022221709  1.28053294265730
%   16 1.27352521545013  0.13609721384114  1.28077668857714
%   17 1.27362947870672  0.13702058341550  1.28097880127279
%   18 1.27371383043724  0.13781837060756  1.28114824478825
%   19 1.27378283184330  0.13851247614341  1.28129169541760
%   20 1.27383984054130  0.13912021006473  1.28141420789639
%   21 1.27388736652228  0.13965540240665  1.28151966586800
%   22 1.27392731241415  0.14012921216214  1.28161109288891
%   23 1.27396113877254  0.14055072474007  1.28169087120397
%   24 1.27398997977557  0.14092739888131  1.28176089826614
%   25 1.27401472548763  0.14126540491432  1.28182270044844
%   50 1.27419383698309  0.14455999813969  1.28236793755452
%  100 1.27421999123493  0.14553632556821  1.28250435013790
%  200 1.27422360135321  0.14580505626452  1.28253845972659
%  500 1.27422411595291  0.14588455202076  1.28254801087299
% 1000 1.27422414812948  0.14589626625122  1.28254937533896
% 2000 1.27422415222862  0.14589922912363  1.28254971645610
% 5000 1.27422415278303  0.14590006389973  1.28254981196894
% 8000 1.27422415281188  0.14590016110307  1.28254982305526
%10000 1.27422415281644  0.14590018355296  1.28254982561364
%12000 1.27422415281846  0.14590019575197  1.28254982700337

upperbound = sqrt(pi*pi/6-tailnorm^2)

% (conservative) upper bound on norm:       1.27422415599759
% (extremely likely) upper bound on norm:   1.274224152825
```

4 Problem 4

What is the global minimum of the function

$$\exp(\sin(50x)) + \sin(60e^y) + \sin(70\sin x) + \sin(\sin(80y)) - \sin(10(x+y)) + \frac{1}{4}(x^2+y^2)?$$

4.1 Solution Method

The partial derivatives of

$$f(x, y) = e^{\sin 50x} + \sin(60e^y) + \sin(70 \sin x) + \sin(\sin 80y) - \sin(10(x + y)) + \frac{x^2 + y^2}{4}$$

are

$$\frac{\partial f}{\partial x}(x, y) = 50e^{\sin 50x} \cos 50x + 70 \cos x \cos(70 \sin x) - 10 \cos(10(x + y)) + \frac{x}{2}, \quad (4.1)$$

$$\frac{\partial f}{\partial y}(x, y) = 60e^y \cos(60e^y) + 80 \cos 80y \cos(\sin 80y) - 10 \cos(10(x + y)) + \frac{y}{2}. \quad (4.2)$$

The global minimum of f occurs at a point at which $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are zero. Thus, we restrict attention to points (x, y) simultaneously satisfying the equations

$$\cos(10(x + y)) = 5e^{\sin 50x} \cos 50x + 7 \cos x \cos(70 \sin x) + \frac{x}{20},$$
$$\cos(10(x + y)) = 6e^y \cos(60e^y) + 8 \cos 80y \cos(\sin 80y) + \frac{y}{20}.$$

We may also write

$$f(x, y) = f_1(x) + f_2(y) + f_3(x) + f_4(y) + f_5(x, y) + f_6(x, y)$$

where

$$f_1(x) = e^{\sin 50x}, \quad f_2(y) = \sin 60e^y, \quad f_3(x) = \sin(70 \sin x),$$
$$f_4(y) = \sin(\sin 80y), \quad f_5(x, y) = -\sin(10(x + y)), \quad f_6(x, y) = \frac{x^2 + y^2}{4}.$$

Define h_k to be the minimum of f_k $(k = 1, 2, \ldots, 6)$. Thus,

$$h_1 = e^{-1},\ h_2 = h_3 = -1,\ h_4 = -\sin 1,\ h_5 = -1,\ h_6 = 0,$$

so

$$h = \sum_{k=1}^{6} h_k = e^{-1} - 3 - \sin 1$$

is less than the global minimum of f. We search for the minimum among points (x, y) such that $f_k(x, y) < h_k + 0.167$ for each $k = 1, 2, \ldots, 6$. This will be justified if our final answer exceeds h by less than 0.167.

The restriction $f_6(x, y) < 0.167$ implies

$$\max\{|x|, |y|\} \le \sqrt{x^2 + y^2} < \sqrt{0.668} < 0.82. \tag{4.3}$$

When $f_5(x, y) < -0.833$, we find $\sin(10(x + y)) > 0.833$, so for integer m

$$\frac{\arcsin(0.833)}{10} + \frac{m\pi}{5} < x + y < \frac{\pi - \arcsin(0.833)}{10} + \frac{m\pi}{5}, \tag{4.4}$$

and comparison with (4.3) shows that only $m = -2, -1, 0, 1$ need be considered. When $f_4(y) < 0.167 - \sin 1$,

$$\sin 80y < \arcsin(0.167 - \sin 1)$$

and so

$$-\frac{\pi + \arcsin(\arcsin(0.167 - \sin 1))}{80} + \frac{n\pi}{40} < y < \frac{\arcsin(\arcsin(0.167 - \sin 1))}{80} + \frac{n\pi}{40} \tag{4.5}$$

where only integer n satisfying $-10 \le n \le 10$ need be considered. When $f_3(x) < -0.833$,

$$\frac{-\pi + \arcsin(0.833)}{70} + \frac{\ell\pi}{35} < \sin x < -\frac{\arcsin(0.833)}{70} + \frac{\ell\pi}{35}$$

and so

$$\arcsin\left(\frac{-\pi + \arcsin(0.833)}{70} + \frac{\ell\pi}{35}\right) < x < \arcsin\left(-\frac{\arcsin(0.833)}{70} + \frac{\ell\pi}{35}\right)$$

where only integer ℓ satisfying $-7 \le \ell \le 8$ need be considered. When $f_2(y) < -0.833$,

$$\frac{-\pi + \arcsin(0.833)}{60} + \frac{p\pi}{30} < e^y < -\frac{\arcsin(0.833)}{60} + \frac{p\pi}{30}$$

and so

$$\log\left(\frac{-\pi + \arcsin(0.833)}{60} + \frac{p\pi}{30}\right) < y < \log\left(-\frac{\arcsin(0.833)}{60} + \frac{p\pi}{30}\right)$$

where only integer p satisfying $5 \le p \le 21$ need be considered. When $f_1(x) < e^{-1} + 0.167$,

$$\sin 50x < \log(e^{-1} + 0.167)$$

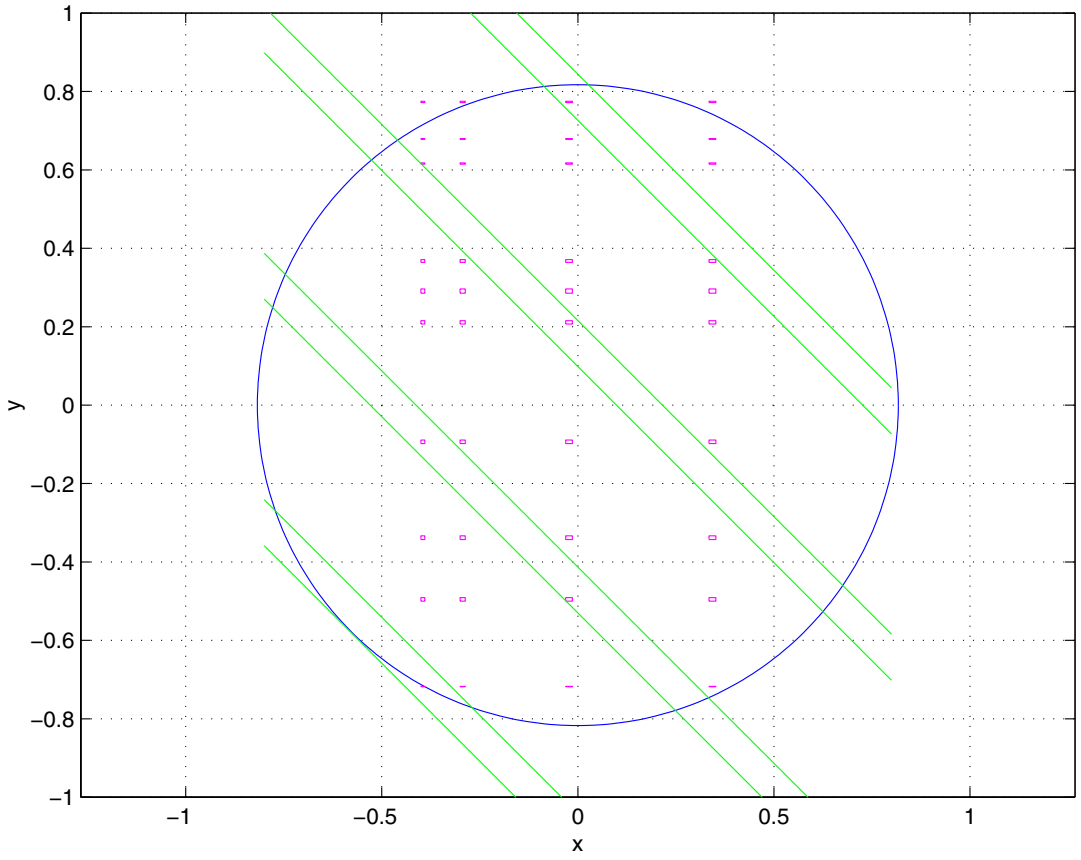

Figure 4.1: The rectangles, strips, and circle that define the search region.

and so

$$\frac{\arcsin(\log(e^{-1}+0.167))}{50}+\frac{q\pi}{25}<x<-\frac{\pi+\arcsin(\log(e^{-1}+0.167))}{50}+\frac{q\pi}{25} \tag{4.6}$$

where only integer q satisfying $-6 \le q \le 6$ need be considered.

Equations (4.5)-(4.6) combine to yield forty rectangular regions in the x-y plane. Only six of those rectangles ntersect the four strips defined by (4.4), as shown in Figure 4.1. Among those six rectangular regions the inequality $f_4+f_6 > h_4+h_6+0.167$ is satisfied on the three furthest from the origin. Hence, the global minimum being sought must lie within one of the three rectangles closest to the origin, which also lie within the two strips closest to the origin. Within each of these rectangles the curves represented by (4.1) and (4.2) intersect to yield a point at which $f(x, y)$ achieves a local minimum. Among these minima the one found at

$$(x_0, y_0) = (-0.02440307969843,\ 0.21061242715243)$$

yields the desired global minimum, which is

$$f(x_0, y_0) = -3.3068686474752.$$

It should be noted that

$$f(x_0, y_0) - h = 0.16672289616125 < 0.167,$$

as necessary to validate the analysis.

4.2 Code Listing

The solution to this problem was coded in MATLAB. It consists of the single file `problem4.m` listed below.

```
c1=0.1*asin(0.833);
c2=0.1*(pi-asin(0.833));
xplusymin=zeros(1,4);
xplusymax=zeros(1,4);
for k=1:4
  m=k-3;
  xplusymin(k)=c1+0.2*m*pi;
  xplusymax(k)=c2+0.2*m*pi;
end
xplusymin
xplusymax

figure(1)
y=zeros(101);
f4=y;
c2=0.0125*asin(asin(0.167-sin(1)));
c1=0.0125*(-pi-asin(asin(0.167-sin(1))));
```

```
y2min=zeros(1,21);
y2max=zeros(1,21);
for k=1:21
  m=k-11;
  y2min(k)=c1+0.025*m*pi;
  y2max(k)=c2+0.025*m*pi;
  for n=1:101
    y(n)=y2min(k)+(n-1)*0.01*(y2max(k)-y2min(k));
    f4(n)=sin(sin(80*y(n)));
  end
  plot(y,f4,'b')
  hold on
end
y2min
y2max

figure(2)
x=zeros(101);
f3=x;
c2=-asin(0.833)/70;
c1=(-pi+asin(0.833))/70;
x2min=zeros(1,16);
x2max=zeros(1,16);
for k=1:16
  m=k-8;
  x2min(k)=asin(c1+m*pi/35);
  x2max(k)=asin(c2+m*pi/35);
  for n=1:101
    x(n)=x2min(k)+(n-1)*0.01*(x2max(k)-x2min(k));
    f3(n)=sin(70*sin(x(n)));
  end
  plot(x,f3,'b')
  hold on
end
x2min
x2max

figure(1)
f2=y;
c2=-asin(0.833)/60;
c1=(-pi+asin(0.833))/60;
y1min=zeros(1,17);
y1max=zeros(1,17);
for k=1:17
  m=k+4;
  y1min(k)=log(c1+m*pi/30);
  y1max(k)=log(c2+m*pi/30);
  for n=1:101
    y(n)=y1min(k)+(n-1)*0.01*(y1max(k)-y1min(k));
    f2(n)=sin(60*exp(y(n)));
  end
  plot(y,f2,'r')
  hold on
end
y1min
y1max
```

```
grid on

figure(2)
f1=x;
c2=0.02*asin(log(exp(-1)+0.167));
c1=0.02*(-pi-asin(log(exp(-1)+0.167)));
x1min=zeros(1,13);
x1max=zeros(1,13);
for k=1:13
  m=k-7;
  x1min(k)=c1+0.04*m*pi;
  x1max(k)=c2+0.04*m*pi;
  for n=1:101
    x(n)=x1min(k)+(n-1)*0.01*(x1max(k)-x1min(k));
    f1(n)=exp(sin(50*x(n)));
  end
  plot(x,f1,'r')
  hold on
end
x1min
x1max
grid on

ymin=zeros(1,10);
ymax=ymin;
ymin=[ y1min( 1) y2min( 5) y2min( 7) y1min( 5) y2min(14) ...
       y2min(15) y2min(16) y1min(14) y2min(20) y1min(17) ]
ymax=[ y2max( 2) y1max( 2) y1max( 3) y2max(10) y1max( 8) ...
       y1max( 9) y1max(10) y2max(19) y1max(15) y2max(21) ]

xmin=zeros(1,4);
xmax=xmin;
xmin=[ x2min( 4) x1min( 5) x2min( 8) x2min(12) ]
xmax=[ x1max( 4) x2max( 5) x2max( 8) x2max(12) ]

r2=0.668
r=sqrt(r2);
% max{x,y}<r<0.82
phi=zeros(1,361);
xplot=zeros(1,361);
yplot=zeros(1,361);
for k=1:361
  phi(k)=(k-1)*pi/180;
  xplot(k)=r*cos(phi(k));
  yplot(k)=r*sin(phi(k));
end

figure(3)
plot(xplot,yplot,'b')
hold on
axis equal

clear phi xplot yplot

xplot=zeros(1,5);
yplot=zeros(1,5);
```

```
for m=1:4
  xplot(1)=xmin(m);
  xplot(2)=xmax(m);
  xplot(3)=xmax(m);
  xplot(4)=xmin(m);
  xplot(5)=xplot(1);
  for n=1:10
    yplot(1)=ymin(n);
    yplot(2)=ymin(n);
    yplot(3)=ymax(n);
    yplot(4)=ymax(n);
    yplot(5)=yplot(1);
    plot(xplot,yplot,'m')
    hold on
  end
end
grid on

xplot=[-0.8 0.8];
yplot=zeros(1,2);
for k=1:4
  m=k-3;
  yplot(1)=-xplot(1)+0.1*asin(0.833)+0.2*m*pi;
  yplot(2)=-xplot(2)+0.1*asin(0.833)+0.2*m*pi;
  plot(xplot,yplot,'g')
  yplot(1)=-xplot(1)+0.1*(pi-asin(0.833))+0.2*m*pi;
  yplot(2)=-xplot(2)+0.1*(pi-asin(0.833))+0.2*m*pi;
  plot(xplot,yplot,'g')
end

figure(4)
for m=1:101
  x(m)=xmin(3)+(m-1)*0.01*(xmax(3)-xmin(3));
end
y=-x-0.2*pi+0.1*acos(5*cos(50*x).*exp(sin(50*x))+7*cos(x) ...
    .*cos(70*sin(x))+0.05*x);
plot(x,y,'r')
hold on
grid on

clear x y

for m=1:101
  y(m)=ymin(2)+(m-1)*0.01*(ymax(2)-ymin(2));
end
x=-y-0.2*pi+0.1*acos(6*exp(y).*cos(60*exp(y))+8*cos(80*y) ...
    .*cos(sin(80*y))+0.05*y);
plot(x,y,'m')

clear x y

figure(5)
for m=1:101
  x(m)=xmin(1)+(m-1)*0.01*(xmax(1)-xmin(1));
end
y=-x-0.2*pi+0.1*acos(5*cos(50*x).*exp(sin(50*x))+7*cos(x) ...
```

```
    .*cos(70*sin(x))+0.05*x);
plot(x,y,'r')
hold on
grid on

clear x y

for m=1:101
  y(m)=ymin(4)+(m-1)*0.01*(ymax(4)-ymin(4));
end
x=-y-0.2*pi+0.1*acos(6*exp(y).*cos(60*exp(y))+8*cos(80*y) ...
    .*cos(sin(80*y))+0.05*y);
plot(x,y,'m')

clear x y

figure(6)
for m=1:101
  x(m)=xmin(3)+(m-1)*0.01*(xmax(3)-xmin(3));
end
y=-x+0.1*acos(5*cos(50*x).*exp(sin(50*x))+7*cos(x) ...
    .*cos(70*sin(x))+0.05*x);
plot(x,y,'r')
hold on
grid on

clear x y

for m=1:101
  y(m)=ymin(5)+(m-1)*0.01*(ymax(5)-ymin(5));
end
x=-y+0.1*acos(6*exp(y).*cos(60*exp(y))+8*cos(80*y) ...
    .*cos(sin(80*y))+0.05*y);
plot(x,y,'m')

clear x y

lowerbnd=exp(-1)-3-sin(1);
fmin=0;

for x=-0.024:0.00001:-0.022
  for y=-0.496:0.00001:0.-0.494
    t1=exp(sin(50*x));
    t2=sin(60*exp(y));
    t3=sin(70*sin(x));
    t4=sin(sin(80*y));
    t5=-sin(10*(x+y));
    r=sqrt(x*x+y*y);
    t6=0.25*r*r;
    f=t1+t2+t3+t4+t5+t6;
    if (f<fmin)
      xmin=x;
      ymin=y;
      fmin=f;
      epsilon=fmin-lowerbnd;
      rmax=2*sqrt(epsilon);
```

```
%      ['For x, y: ' num2str(x) ' ' num2str(y)]
%      ['    t1 t2 t3: ' num2str(t1) ' ' num2str(t2) ' ' num2str(t3)]
%      ['    t4 t5 t6: ' num2str(t4) ' ' num2str(t5) ' ' num2str(t6)]
%      ['    fmin epsilon rmax: ' num2str(fmin) ' ' num2str(epsilon) ' ' ...
%                           num2str(rmax)]
    end
  end
end

for x=-0.396:0.00001:-0.394
  for y=-0.094:0.00001:0.-0.092
    t1=exp(sin(50*x));
    t2=sin(60*exp(y));
    t3=sin(70*sin(x));
    t4=sin(sin(80*y));
    t5=-sin(10*(x+y));
    r=sqrt(x*x+y*y);
    t6=0.25*r*r;
    f=t1+t2+t3+t4+t5+t6;
    if (f<fmin)
      xmin=x;
      ymin=y;
      fmin=f;
      epsilon=fmin-lowerbnd;
      rmax=2*sqrt(epsilon);
%      ['For x, y: ' num2str(x) ' ' num2str(y)]
%      ['    t1 t2 t3: ' num2str(t1) ' ' num2str(t2) ' ' num2str(t3)]
%      ['    t4 t5 t6: ' num2str(t4) ' ' num2str(t5) ' ' num2str(t6)]
%      ['    fmin epsilon rmax: ' num2str(fmin) ' ' num2str(epsilon) ' ' ...
%                           num2str(rmax)]
    end
  end
end

for x=-0.026:0.00001:-0.024
  for y=0.21:0.00001:0.212
    t1=exp(sin(50*x));
    t2=sin(60*exp(y));
    t3=sin(70*sin(x));
    t4=sin(sin(80*y));
    t5=-sin(10*(x+y));
    r=sqrt(x*x+y*y);
    t6=0.25*r*r;
    f=t1+t2+t3+t4+t5+t6;
    if (f<fmin)
      xmin=x;
      ymin=y;
      fmin=f;
      epsilon=fmin-lowerbnd;
      rmax=2*sqrt(epsilon);
%      ['For x, y: ' num2str(x) ' ' num2str(y)]
%      ['    t1 t2 t3: ' num2str(t1) ' ' num2str(t2) ' ' num2str(t3)]
%      ['    t4 t5 t6: ' num2str(t4) ' ' num2str(t5) ' ' num2str(t6)]
%      ['    fmin epsilon rmax: ' num2str(fmin) ' ' num2str(epsilon) ' ' ...
%                           num2str(rmax)]
    end
```

```
  end
end

figure(7)
for m=1:101
  x(m)=-0.02440307970+(m-1)*(1e-13);
end
y=-x+0.1*acos(5*cos(50*x).*exp(sin(50*x))+7*cos(x) ...
    .*cos(70*sin(x))+0.05*x);
plot(x,y,'r')
hold on
grid on

for m=1:101
  y(m)=0.21061242715+(m-1)*(1e-13);
end
x=-y+0.1*acos(6*exp(y).*cos(60*exp(y))+8*cos(80*y) ...
    .*cos(sin(80*y))+0.05*y);
plot(x,y,'m')

axis([-0.02440307970 -0.02440307969 0.2106124271 0.2106124272])

clear x y
```

5 Problem 5

Let $f(z) = 1/\Gamma(z)$, where $\Gamma(z)$ is the gamma function, and let $p(z)$ be the cubic polynomial that best approximates $f(z)$ on the unit disk in the supremum norm $\|\cdot\|_\infty$. What is $\|f - p\|_\infty$?

5.1 Solution Method

Because of the symmetry exhibited by f, we restrict the search to cubic polynomials with real coefficients.

Let $\hat{p}$ be any trial cubic polynomial. Since f and $\hat{p}$ are entire functions so is their difference $f - \hat{p}$, and thus by the maximum principle the maximum value of $\left|f - \hat{p}\right|$ over the unit disk will occur somewhere on the unit circle $|z| = 1$.

Since f is entire, it can be approximated on the unit disk by a truncated Maclaurin series. We first attempted to use the coefficients listed in [1, Formula 6.1.34], but found that some of the listed coefficients were erroneous. We instead adopted the following method for computing the series coefficients. It is clear that $f(0) = 0$ and $f'(0) = 1$. Higher order derivatives of f at 0 satisfy a recursion formula [5] involving the Euler-Mascheroni constant

$$\gamma = -\Gamma'(1) \approx 0.57721566490153286$$

and the Riemann zeta function

$$\zeta(z) = \sum_{k=1}^{\infty} \frac{1}{k^z}$$

evaluated at $z = 2, 3, \ldots$. The recursion formula is

$$\frac{f^{(n)}(0)}{n} = \frac{\gamma}{n-1} f^{(n-1)}(0) - \sum_{k=2}^{n-1} (-1)^k \frac{(n-2)!}{(n-k)!} \zeta(k) f^{(n-k)}(0) \qquad n = 2, 3, \ldots.$$

The 27-term power series $F(z)$ with polynomial coefficients obtained by numerically evaluating the recursion formula approximates the reciprocal gamma function over the unit disk with a maximum error of less than 10^{-15}. Let

$$P(z) \equiv F(z) - \hat{p}(z) = \sum_{n=1}^{N} c_n z^{N-n}, \qquad |P(z)|^2 = \sum_{m=1}^{N} \sum_{n=1}^{N} c_m c_n z^{N-n} (z^*)^{N-m}$$

where $N = 27$ and the $\{c_n\}$ are known real numbers. For points on the unit circle, $z = e^{i\theta}$, so that

$$\left|P(e^{i\theta})\right|^2 = \sum_{m=1}^{N} \sum_{n=1}^{N} c_m c_n e^{i(m-n)\theta}.$$

The maximum modulus of P on the unit disk will thus occur at one of the critical points defined by

$$\begin{aligned} 0 = -i \frac{d}{d\theta} \left(\left|P(e^{i\theta})\right|^2 \right) &= \sum_{m=1}^{N} \sum_{n=1}^{N} (m-n) c_m c_n e^{i(m-n)\theta} \\ &= \sum_{m=1}^{N} \sum_{n=1}^{N} (m-n) c_m c_n z^{m-n} = \sum_{s=1-N}^{N-1} d_s z^s \end{aligned}$$

where $d_s = s \sum_{n=1}^{N-|s|} c_n c_{n+|s|}$. Therefore, the critical points can be easily located by finding the roots of a polynomial (using a built-in MATLAB function `roots`) and individually examined. The polynomial coefficients in $\hat{p}$ are then adjusted to minimize the maximum modulus of P on the unit circle using an optimization program.

5.2 Code Listing

The solution to this problem was coded in MATLAB and consists of the file `problem5.m` which is listed below.

```
function problem5
%p0 = [-0.655878 0.577216 1 0];
%p0 = [-0.60334321133579 0.62521193360141 1.01976187015187 0.00554195980514];
%p0 = [-0.61005474734507 0.61347731512549 1.00887636582242 0];
%p0 = [-0.60334325127370 0.62521186092055 1.01976179747105 0.00554191986731];
p0 = [ -0.60334325127986 0.62521185940116 1.01976179595166 0.00554191986116];
p = p0;
dold = 10;
d = 1;
diary ndsimplex.log
% We found that the following loop was necessary to avoid stagnation in
% the optimization of the coefficients.  It optimizes first
% over a several subsets of the polynomial coefficients and then repeats.
% We found that when we tried to optimize all four polynomial coefficients
% simultaneously, the optimizers bogged down.  This occurred for three
% different choices of optimizer, including fminimax (from the
% Optimization Toolbox) and ga from the Genetic Algorithm Optimization
% Toolbox (GAOT).
while d < dold
  dold = d;
  [p,d] = ndsimplexsolve([1 2 3],p);
  fprintf('%17.14f %16.14f %16.14f %16.14f %16.14f\n',[p,d]);
  [p,d] = ndsimplexsolve([2 3 4],p);
  fprintf('%17.14f %16.14f %16.14f %16.14f %16.14f\n',[p,d]);
  [p,d] = ndsimplexsolve([1 2 3 4],p);
  fprintf('%17.14f %16.14f %16.14f %16.14f %16.14f\n',[p,d]);
end
diary off
return

function [p,d] = ndsimplexsolve(dimens,p0)
% This version uses fminsearch (Nelder-Meade).
% Optimize on 1, 2, 3, or 4 coefficients at a time.
% dimens is a vector containing a subset of {1,2,3,4} denoting which
% dimensions of p are to be optimized over. p0 is the initial starting
% guess for p (including all four dimensions). p and d are the optimized
% values of the polynomial coefficients and distance, resp.
%
% Find the coefficients of a cubic polynomial with real coefficients
% that best approximates (in the sense of the infinity norm) the
% reciprocal gamma function over the unit disk in the complex plane.
%
% Peter Simon
% 3/12/2002
%
%

% Now polish the coefficients
options = optimset('fminsearch'); % Get default options.
                                  % Set function tolerance:
options = optimset(options, ...
                   'TolFun', 1e-14, ...
                   'TolX', 1e-14, ...
                   'Display', 'none', ...
                   'MaxFunEvals', 10000, ...
                   'MaxIter', 10000, ...
```

```
                  'LargeScale', 'off');
[preduced, fval, exitflag] = fminsearch(@func0,p0(dimens), options, dimens, p0);
if exitflag < 0
  error(['Negative exitflag: ', num2str(exitflag)])
end
if exitflag == 0
  disp(['Warning: Termination due to max number of iterations or function' ...
        ' evaluations exceeded!'])
end
p = p0; p(dimens) = preduced;
d = fval;
return


function y = func0(preduced,dimens,p0)
p = p0;
p(dimens) = preduced;
y = func1(p);
return

function y = func1(p)
% Return the maximum magnitude of the difference f(z)-P(z)
% evaluated over the unit disk, where P(z) is the polynomial
% whose coefficients are stored in vector p in standard Matlab format,
% and f(z) is 1/Gamma(z), the reciprocal of the gamma function.  We use a
% 26 term polynomial approximation of the reciprocal gamma function
% obtained from its Maclaurin expansion.  We found that some of
% the coefficients listed in Table 6.1.34 of Abramowitz and Stegun were
% erroneous.
%
%
% Polynomial coefficients for reciprocal gamma function:

c = fliplr(...
           [ 0, ...
             1, ...
             0.57721566490153286, ...
            -0.65587807152025388, ...
            -0.04200263503409524, ...
             0.16653861138229149, ...
            -0.04219773455554434, ...
            -0.00962197152787697, ...
             0.00721894324666310, ...
            -0.00116516759185907, ...
            -0.00021524167411495, ...
             0.00012805028238812, ...
            -0.00002013485478079, ...
            -0.00000125049348214, ...
             0.00000113302723198, ...
            -0.00000020563384170, ...
             0.00000000611609510, ...
             0.00000000500200764, ...
            -0.00000000118127457, ...
             0.00000000010434267, ...
             0.00000000000778226, ...
            -0.00000000000369681, ...
```

```
             0.00000000000051004, ...
            -0.00000000000002058, ...
            -0.00000000000000535, ...
             0.00000000000000123, ...
            -0.00000000000000012]);

Nc = length(c);
Np = length(p);
if Nc >= Np
  c(Nc-Np+1:Nc) =  c(Nc-Np+1:Nc) - p; % Form f(z) - P(z)
  y = dmax(c);
else
  p(Np-Nc+1:Np) =  p(Np-Nc+1:Np) - c; % Form P(z) - f(z)
  y = dmax(p);
end
return

function d = dmax(c)
% d = dmax(c)
%
% Find the maximum magnitude of a polynomial over the unit disk.
% c is a vector of polynomial coefficients stored in standard Matlab
% form. Method: Use maximum principle to limit examination to the unit
% circle (not the disk). Examine zeros of the derivative of P(exp(i*theta))
%
% P. Simon
% 3/14/2002
%
if size(size(c)) > 2
  error('c must be a vector!')
end
N = length(c);
d = zeros(1,N-1); % Allocate space for d coefficients of positive
                  % subscript.
for s = 1:(N-1)
  for n = 1:(N-s)
    d(s) = d(s) + c(n) * conj(c(n+s));
  end
  d(s) = s*d(s);
end

D = [fliplr(d) 0 -conj(d)];  % Set up polynomial coefficients
R = roots(D); % Locations on unit circle where derivative wrt theta is
              % zero.
R(abs(R) > 1.01 | abs(R) < 0.99) = []; % Eliminate points off unit circle.

d = polyval(c,R); % Evaluate polynomial at critical points.
d = max(abs(d)); % Locate maximum magnitude.
return
```

6 Problem 6

A flea starts at $(0, 0)$ on the infinite 2D integer lattice and executes a biased random walk: At each step it hops north or south with probability $1/4$, east with probability $1/4 + \varepsilon$, and west with probability $1/4 - \varepsilon$. The probability that the flea returns to $(0, 0)$ sometime during its wanderings is $1/2$. What is ε?

6.1 Solution Method

Let $q = 1/4$; for fixed $\varepsilon \in [-q, q]$, let $d = \sqrt{q^2 - \varepsilon^2}$. Let P_n denote the probability that the flea finds itself at $(0, 0)$ after $2n$ steps. In general,

$$P_n = \binom{2n}{n\ \ n\ \ 0\ \ 0} q^{2n} + \sum_{k=1}^{n} \binom{2n}{n-k\ \ n-k\ \ k\ \ k} q^{2(n-k)} d^{2k} \tag{6.1}$$

for positive integer n. For $n \geq 2$, we may write

$$P_n = Q_n + R_n$$

where R_n denotes the probability that the flea is arriving at $(0, 0)$ at step $2n$ for the first time since it was originally there, while Q_n denotes the probability that the flea has previously returned to $(0, 0)$ as well as being there after $2n$ steps. Also, for integer $n \geq 2$

$$Q_n = P_1 P_{n-1} + \sum_{k=2}^{n-1} R_k P_{n-k} \qquad \text{and} \qquad R_n = P_n - Q_n. \tag{6.2}$$

We may use (6.1) and (6.2) to compute Q_n and R_n for small n. The probability that the flea eventually returns to $(0, 0)$ is

$$P = P_1 + \sum_{n=2}^{\infty} R_n. \tag{6.3}$$

Truncation of the infinite series in (6.3) yields

$$P \approx P_1 + \sum_{n=2}^{N} R_n \tag{6.4}$$

where N may be selected to achieve any desired degree of accuracy. Moreover, we may choose ε so that (6.4) is satisfied with $P = 1/2$. The desired ε is ± 0.061913954473991.

6.2 Code Listing

The solution to this problem was coded in a multiprecision extension to Fortran 77 [4] and consists of the single file `problem6.fpd`, listed below.

```
cmp+ precision level 250
      program prob6

      implicit none

      integer k, m, n

cmp+ multip real c, d, e

      integer nterms, ngterms
      parameter(nterms=1200, ngterms=2*nterms+1)

cmp+ multip real qnomial, num, denom
      dimension qnomial(nterms,nterms+1)

cmp+ multip real p, q, r, ptot
      dimension p(nterms), q(nterms), r(nterms)

cmp+ multip real gamma
      dimension gamma(ngterms)

      gamma(1)=1.d0
      do n=2,ngterms
        gamma(n)=(n-1)*gamma(n-1)
c        write(6,*) 'n, gamma(n): ', n, gamma(n)
      end do

      do n=1,nterms
        num=gamma(2*n+1)
        do m=1,n+1
          k=m-1
          denom=gamma(n-k+1)*gamma(n-k+1)*gamma(k+1)*gamma(k+1)
          qnomial(n,m)=num/denom
        end do
      end do

      c=0.25d0
      e=0.06191395447399085d0
      write(6,*) 'e: ', e
      d=sqrt(c*c-e*e)

      p(1)=2.d0*(c*c+d*d)
      n=1
      ptot=p(1)
      do n=2,nterms
        p(n)=qnomial(n,1)*c**(2*n)
        do k=1,n
          p(n)=p(n)+qnomial(n,k+1)*c**(2*(n-k))*d**(2*k)
        end do
        q(n)=p(1)*p(n-1)
        do k=2,n-1
```

```
    q(n)=q(n)+r(k)*p(n-k)
  end do
  r(n)=p(n)-q(n)
  ptot=ptot+r(n)
end do
write(6,*) 'nterms, ptot: ', nterms, ptot

e=0.06191395447399095d0
write(6,*) 'e: ', e
d=sqrt(c*c-e*e)

p(1)=2.d0*(c*c+d*d)
n=1
ptot=p(1)
do n=2,nterms
  p(n)=qnomial(n,1)*c**(2*n)
  do k=1,n
    p(n)=p(n)+qnomial(n,k+1)*c**(2*(n-k))*d**(2*k)
  end do
  q(n)=p(1)*p(n-1)
  do k=2,n-1
    q(n)=q(n)+r(k)*p(n-k)
  end do
  r(n)=p(n)-q(n)
  ptot=ptot+r(n)
end do
write(6,*) 'nterms, ptot: ', nterms, ptot

stop
end
```

7 Problem 7

Let A be the $20,000 \times 20,000$ matrix whose entries are zero everywhere except for the primes $2, 3, 5, 7, \ldots, 224737$ along the main diagonal and the number 1 in all the positions a_{ij} with $|i - j| = 1, 2, 4, 8, \ldots, 16384$. What is the $(1, 1)$ entry of A^{-1}?

7.1 Solution Method

Consider the general problem wherein $A = [a_{mn}]$ is the $N \times N$ matrix with entries defined by

$$\begin{aligned} a_{mm} &= m^{\text{th}} \text{ prime (in ascending order)}, \\ a_{mn} &= 1 \text{ if } |m - n| = 2^p \text{ for some integer } p, \\ a_{mn} &= 0 \text{ otherwise.} \end{aligned}$$

We are to solve the matrix equation $Ax = b$ for x given the column vector $b = [b_m]$ of length N defined by $b_1 = 1$ and $b_m = 0$ otherwise. The x_1 entry of the solution vector $x = [x_m]$ is then the $(1, 1)$ entry of A^{-1}. For $N \geq 14$ over half the entries of A are

zero, so sparse matrix techniques are desirable. The built-in MATLAB construct $A\backslash b$ allows the solution of $Ax = b$ with A in sparse format. However, to extend the range of N for which solutions may be found, the recently formulated UMFPACK routines (found at www.cise.ufl.edu/research/sparse) were implemented. The results are presented in tabular form as follows:

N	x_1
1	0.5
2	0.6
5	0.68241834607366
10	0.70765055512457
20	0.71742894560079
50	0.72171519597936
100	0.72359427773595
200	0.72441297988280
500	0.72478203845865
1000	0.72494532189647
2000	0.72501883262526
5000	0.72506786914184
10000	0.72507501773406
20000	0.72507834626840

7.2 Code Listing

The solution to this problem was coded in MATLAB and consists of the single file `problem7.m`, listed below.

```
% Solution to Problem #7 (Hundred-Dollar, Hundred-Digits)
% Kim McInturff  4/5/2002
%
% Requires UMFPACK, obtain from http://www.cise.ufl.edu/research/sparse
%
% To run for other matrix orders than 20000, change the value of nrow.

nrow=20000;
ncol=nrow;

p=zeros(nrow,1);
q=zeros(nrow,1);
s=zeros(nrow,1);

m=1;
k=2;
while m<=nrow
  if (size(factor(k))==[1 1])
    p(m)=m;
    q(m)=m;
```

```
    s(m)=k;
    m=m+1;
  end
  k=k+1;
end
A=sparse(p,q,s,nrow,ncol,nrow);
clear p q s

pmax=0;
test=nrow;
while test>2
  pmax=pmax+1;
  test=0.5*test;
end
noffdiag=2*(pmax+1)*(nrow-2^pmax);
for k=1:pmax
  noffdiag=noffdiag+2*k*2^(k-1);
end

p=zeros(noffdiag,1);
q=zeros(noffdiag,1);
s=zeros(noffdiag,1);

length=0;
mmax=nrow-1;
for m=1:mmax
  power=0;
  while (m+2^power<=nrow)
    length=length+2;
    p(length-1)=m;
    q(length-1)=m+2^power;
    s(length-1)=1;
    p(length)=m+2^power;
    q(length)=m;
    s(length)=1;
    power=power+1;
  end
end
A=A+sparse(p,q,s,nrow,ncol,length);
clear p q s

b=zeros(nrow,1);
b(1)=1;
x=umfpack(A,'\',b);
x1byumf=x(1)
xcheck=A\b;
x1bystandard=xcheck(1)

r=b-A*x;
supresidual=max(r)
```

8 Problem 8

A square plate $[-1, 1] \times [-1, 1]$ is at temperature $u = 0$. At time $t = 0$ the temperature is increased to $u = 5$ along one of the four sides while being held at $u = 0$ along the other three sides, and heat then flows into the plate according to $u_t = \Delta u$. When does the temperature reach $u = 1$ at the center of the plate?

8.1 Solution Method

We will assume that the selected edge is located at $y = -1$. The boundary value problem to be solved is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t} \quad (|x| < 1, |y| < 1, t > 0)$$

where $u = u(x, y, t)$ is subject to the boundary conditions

$$\begin{aligned} u(-1, y, t) &= 0 \\ u(1, y, t) &= 0 \\ u(x, 1, t) &= 0 \\ u(x, -1, t) &= 5, \quad (t > 0) \end{aligned}$$

and the initial condition

$$u(x, y, 0) = 0. \tag{8.1}$$

We will employ the Laplace transform to eliminate the partial derivative with respect to time t. Let U be the Laplace transform of u and let s be the transform variable. Then $U(x, y, s)$ satisfies the partial differential equation

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = sU - u(x, y, 0) = sU \quad (|x| < 1, |y| < 1, s > 0) \tag{8.2}$$

by virtue of (8.1), subject to the boundary conditions

$$U(-1, y, s) = 0 \tag{8.3a}$$
$$U(1, y, s) = 0 \tag{8.3b}$$
$$U(x, 1, s) = 0 \tag{8.3c}$$
$$U(x, -1, s) = \frac{5}{s} \tag{8.3d}$$

Applying the technique of separation of variables, we assume eigenfunctions of the form $U = X(x)\,\Psi(y, s)$ and (8.2) becomes

$$\frac{X''}{X} + \frac{1}{\Psi}\frac{\partial^2 \Psi}{\partial y^2} = s$$

so we may define a separation constant k^2 such that

$$-\frac{X''}{X} = \frac{1}{\Psi}\frac{\partial^2 \Psi}{\partial y^2} - s = k^2$$

or

$$X'' + k^2 X = 0, \quad \frac{\partial^2 \Psi}{\partial y^2} - \phi^2 \Psi = 0 \tag{8.4}$$

where $\phi^2 = s + k^2$. Solutions of (8.4) take the form

$$X = A\cos k(x+1) + B\sin k(x+1), \quad \Psi = C\cosh \phi(1-y) + D\sinh \phi(1-y)$$

From (8.3a) we have $A = 0$, and from (8.3b)

$$k = k_m = \frac{m\pi}{2}, \quad m = 1, 2, 3, \ldots \tag{8.5}$$

From (8.3c) we have $C = 0$ and as a result of (8.5) the solution of the boundary value problem (8.2)–(8.3) is

$$U(x, y, s) = \sum_{m=1}^{\infty} U_m \sin k_m(x+1)\sinh \phi_m(1-y) \tag{8.6}$$

where $\phi_m = \sqrt{s + k_m^2}$.

To evaluate U_m note that (8.3d) implies

$$\sum_{m=1}^{\infty} U_m \sin k_m(x+1)\sinh 2\phi_m = \frac{5}{s}.$$

Multiplying by $\sin k_n(x+1)$ and integrating from -1 to 1 yields

$$U_n \sinh 2\phi_n \int_{-1}^{1} \sin^2 k_n(x+1)\,dx = \frac{5}{s}\int_{-1}^{1} \sin k_n(x+1)\,dx$$

Thus,

$$U_n \sinh 2\phi_n = \frac{5}{s}\sin k_n \int_{-1}^{1} \cos k_n x\,dx = \frac{5}{k_n s}\sin k_n \sin k_n x\Big|_{-1}^{1} = \frac{10}{k_n s}\sin^2 k_n,$$

so

$$U_n = \begin{cases} \frac{10}{k_n s \sinh 2\phi_n} & (n \text{ odd}) \\ 0 & (n \text{ even}). \end{cases} \tag{8.7}$$

Inserting (8.7) into (8.6) yields

$$U(x, y, s) = \frac{10}{s} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin k_m(x+1) \sinh(1-y)\phi_m}{k_m \sinh 2\phi_m} = 10 \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin k_m(x+1)}{k_m} F_m(s) \tag{8.8}$$

where we require the inverse Laplace transform $f_m(t)$ of the function

$$F_m(s) = \frac{\sinh(1-y)\phi_m}{s \sinh 2\phi_m}.$$

We have the Laplace transform pair

$$G(s) = \frac{\sinh(1-y)\sqrt{s}}{\sinh 2\sqrt{s}}, \qquad g(t) = \sum_{n=1}^{\infty} (-1)^{n-1} k_n e^{-k_n^2 t} \sin k_n(1-y).$$

which is tabulated in [2, Appendix B, #124][1]. Then the functions

$$H_m(s) = G(\phi_m^2) = \frac{\sinh(1-y)\phi_m}{\sinh 2\phi_m}$$

and

$$h_m(t) = g(t) e^{-k_m^2 t} = \sum_{n=1}^{\infty} (-1)^{n-1} k_n e^{-(k_m^2 + k_n^2)t} \sin k_n(1-y)$$

also form a Laplace transform pair for each positive integer m. After noting that $F_m(s) = \frac{1}{s} H_m(s)$, we conclude that

$$f_m(t) = \int_0^t h_m(u)\, du = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} k_n}{k_m^2 + k_n^2} \left(1 - e^{-(k_m^2 + k_n^2)t}\right) \sin k_n(1-y) \tag{8.9}$$

From (8.8) and (8.9) we obtain an explicit formula for the temperature $u(x, y, t)$:

$$\begin{aligned} u(x, y, t) &= 10 \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin k_m(x+1)}{k_m} f_m(t) \\ &= 10 \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin k_m(x+1)}{k_m} \sum_{n=1}^{\infty} \frac{(-1)^{n-1} k_n}{k_m^2 + k_n^2} \left[1 - e^{-(k_m^2 + k_n^2)t}\right] \sin k_n(1-y) \\ &= \frac{40}{\pi^2} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin \frac{m\pi(x+1)}{2}}{m} \sum_{n=1}^{\infty} \frac{(-1)^{n+1} n}{m^2 + n^2} \left(1 - e^{-(m^2+n^2)\pi^2 t/4}\right) \sin \frac{n\pi(1-y)}{2}. \end{aligned} \tag{8.10}$$

[1] A sign error in the tabulated formula has been corrected.

The series in (8.10) requires further manipulation before it can be efficiently evaluated, since the sum over n is neither absolutely nor uniformly convergent with respect to y. Towards this end, we rewrite (8.10) as

$$u(x, y, t) = u(x, y, \infty) + v(x, y, t)$$

where

$$u(x, y, \infty) = \frac{40}{\pi^2} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin \frac{m\pi(x+1)}{2}}{m} \sum_{n=1}^{\infty} \frac{(-1)^{n+1} n}{m^2 + n^2} \sin \frac{n\pi(1-y)}{2}, \tag{8.11}$$

$$v(x, y, t) = \frac{40}{\pi^2} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin \frac{m\pi(x+1)}{2}}{m} \sum_{n=1}^{\infty} \frac{(-1)^{n} n}{m^2 + n^2} e^{-(m^2+n^2)\pi^2 t/4} \sin \frac{n\pi(1-y)}{2}. \tag{8.12}$$

The numerical difficulties are now restricted to the late-time series (8.11), since (8.12) exhibits exponential convergence for any $t > 0$. We can find an alternative, more rapidly convergent series for $u(x, y, \infty)$ by appealing to the final value theorem of Laplace transform theory:

$$\begin{aligned} u(x, y, \infty) &= \lim_{s \to 0} sU(x, y, s) \\ &= \frac{20}{\pi} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin[m\pi(x+1)/2] \sinh[m\pi(1-y)/2]}{m \sinh m\pi} \\ &= \frac{20}{\pi} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin[m\pi(x+1)/2]}{m} \frac{1 - e^{-m\pi(1-y)}}{1 - e^{-2m\pi}} e^{-m\pi(1+y)/2} \end{aligned} \tag{8.13}$$

or simply by noting that the inner sum in (8.11) is a Fourier series tabulated in [3, Formula 1.445.4]. The final representation in (8.13) explicitly exhibits the exponential convergence of the series for $y > -1$.

We are interested in the special case $x = y = 0$. Thus we require

$$\begin{aligned} u(0, 0, \infty) &= \frac{20}{\pi} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin m\pi/2}{m} \frac{1 - e^{-m\pi}}{1 - e^{-2m\pi}} e^{-m\pi/2} \\ &= \frac{20}{\pi} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{(-1)^{\frac{m-1}{2}}}{m} \frac{1 - e^{-m\pi}}{1 - e^{-2m\pi}} e^{-m\pi/2} \end{aligned}$$

and

$$v(0,0,t) = \frac{40}{\pi^2} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{\sin\frac{m\pi}{2}}{m} \sum_{n=1}^{\infty} \frac{(-1)^n n}{m^2+n^2} e^{-(m^2+n^2)\pi^2 t/4} \sin\frac{n\pi}{2}$$
$$= -\frac{40}{\pi^2} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{(-1)^{\frac{m-1}{2}}}{m} \sum_{\substack{n=1 \\ n \text{ odd}}}^{\infty} \frac{(-1)^{\frac{n-1}{2}} n}{m^2+n^2} e^{-(m^2+n^2)\pi^2 t/4}$$

which, when combined, yield the desired result:

$$u(0,0,t) = u(0,0,\infty) + v(0,0,t)$$
$$= \frac{20}{\pi} \sum_{\substack{m=1 \\ m \text{ odd}}}^{\infty} \frac{(-1)^{\frac{m-1}{2}}}{m} \left[\frac{1-e^{-m\pi}}{1-e^{-2m\pi}} e^{-m\pi/2} - \frac{2}{\pi} \sum_{\substack{n=1 \\ n \text{ odd}}}^{\infty} \frac{(-1)^{\frac{n-1}{2}} n}{m^2+n^2} e^{-(m^2+n^2)\pi^2 t/4} \right] \tag{8.14}$$

For $t \geq 0.01$ the series in (8.14) converges to 14 digits of accuracy using fewer than 900 terms. A root-finding routine is then used to locate the time at which the midpoint temperature reaches 1. When this is done, the value obtained is $t = 0.42401138703369$, as shown in Figure 8.1. As partial verification we note that the series converges to the correct value of $u(0,0,\infty) = \frac{5}{4}$ for large t.

8.2 Code Listing

The solution to this problem was coded in MATLAB and consists of the single file `problem8.m`, listed below.

```
function problem8
%
% problem8.m
% Solution of problem 8 in the hundred-dollar, hundred-digit
% challenge.
%
% Method of attack: Use Laplace transform to eliminate time. Solve the
% resulting 2-D PDE using separation of variables. After transforming
% solution back to time domain, separate out the steady state solution
% and replace this with a series that converges exponentially, using the
% final value theorem of Laplace transform theory.
%
% Peter Simon
% 3/8/2002
%

% Locate the time at which the temperature is 1.
```

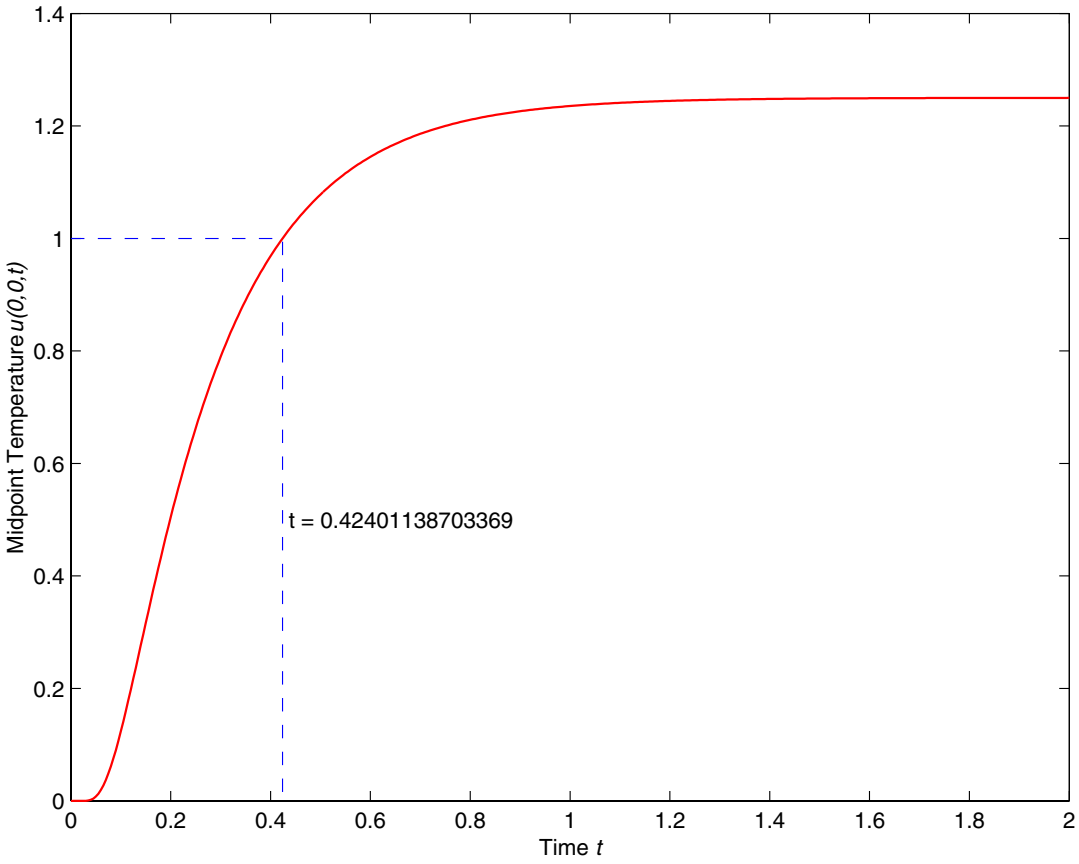

Figure 8.1: Temperature at the center of the square versus time.

```
options = optimset('fzero'); % Get default options.
% Set function tolerance to machine epsilon:
options = optimset(options, 'TolFun', eps);
% Locate the time at which temperature is 1:
[t1, u0, exitflag] = fzero(@u00m1, [0 2], options);
if exitflag < 0
  error('Bad exitflag in findnodes!')
end

% Plot u versus temperature
t = [0:0.005:2];
[umid,terms] = u00(t);
figure
plot(t,umid, 'r-', 'linewidth', 1)
xlabel('Time \it{t}')
ylabel('Midpoint Temperature \it{u}(0,0,\it{t})')
hold on
x = [0 t1 t1];
y = [1 1 0];
plot(x,y,'b--')
text(t1,0.5,sprintf(' t = %16.14f',t1),'horizontalalignment','left')
fprintf('u(0, 0, %16.14f) = 1\n',t1);
% Determine max number of terms needed:
Nmax = max(terms);
index = find(terms == Nmax);
fprintf('Maximum summation terms = %d at t = ',Nmax);
fprintf('%6.2f',t(index));
fprintf('\n');

return

function um1 = u00m1(t)
% Compute the midpoint temperature minus one.  For use in root finding
% routine.
um1 = u00(t) - 1;
return

function [umid, terms] = u00(t)
% Calculate the temperature umid at the midpoint of the square at time
% t. Also return the number of terms summed.

tin = t; % Save original size in case of vector or matrix input.

t = t(:); % Make into a vector.
terms = zeros(size(t)); % Initialize count of terms summed.

umid = zeros(size(t)); % Preallocation

for i = 1:length(t)

  if t(i) <= 5e-4;
    umid(i) = 0;
    continue % Skip to next i value
  end

  pisqto4 = pi^2 * t(i) / 4;
```

```
  twoopi = 2/pi;
  sum_mold = 0;  % Initialize
  m = 1;
  sign_m = 1;

  while 1==1 % Outer loop over m
    mpi = m*pi;
    mpio2 = mpi/2;
    twompi = 2*mpi;
    msq = m*m;

    sum_nold = 0; % Initialize
    n = 1;
    sign_n = 1;

    while 1==1 % Inner loop over n
      msqpnsq = msq + n*n;
      term_n = sign_n * n /(msqpnsq) * exp(-msqpnsq * pisqto4);
      terms(i) = terms(i) + 1; % Bump count of terms summed.
                          % Converge to machine precision:
      sum_n = sum_nold + term_n;
      if sum_nold == sum_n
        break
      else
        sum_nold = sum_n;
        n = n + 2;
        sign_n = -sign_n;
      end
    end % End of loop over n.

    empio2 = exp(-mpio2);
    empi = exp(-mpi);
    emtwopi = exp(-twompi); % compute these separately to avoid rounding
                            % errors

    term_m = sign_m / m * ((1 - empi)/(1 - emtwopi) * empio2 - twoopi*sum_n);
    sum_m = sum_mold + term_m;
    if sum_m == sum_mold % Converge to machine precision.
      break
    else
      sum_mold = sum_m;
      m = m + 2;
      sign_m = -sign_m;
    end

  end % End of loop over m.

  umid(i) = 20/pi * sum_m; % Scale the result properly.

end % End of loop over t values.

umid = reshape(umid,size(tin));
terms = reshape(terms,size(tin));
return
```

9 Problem 9

The integral $I(\alpha) = \int_0^2 [2+\sin(10\alpha)] x^\alpha \sin(\alpha/(2-x))\, dx$ depends on the parameter α. What is the value $\alpha \in [0, 5]$ at which $I(\alpha)$ achieves its maximum?

9.1 Solution Method

For $\alpha \in (0, 5]$, differentiating

$$I(\alpha) = [2 + \sin(10\alpha)] \int_0^2 x^\alpha \sin\left(\frac{\alpha}{2-x}\right) dx$$

yields

$$I'(\alpha) = 10\cos(10\alpha) I_1(\alpha) + [2 + \sin(10\alpha)][I_2(\alpha) + I_3(\alpha)]$$

where

$$\begin{aligned}
I_1(\alpha) &= \int_0^2 x^\alpha \sin\left(\frac{\alpha}{2-x}\right) dx = \alpha \int_{\alpha/2}^\infty \left(2 - \frac{\alpha}{u}\right)^\alpha \frac{\sin u}{u^2} du \\
&= \alpha \left[\int_{\alpha/2}^\pi \left(2 - \frac{\alpha}{u}\right)^\alpha \frac{\sin u}{u^2} du + \sum_{k=1}^\infty \int_{k\pi}^{(k+1)\pi} \left(2 - \frac{\alpha}{u}\right)^\alpha \frac{\sin u}{u^2} du \right], \qquad (9.1) \\
I_2(\alpha) &= \int_0^2 x^\alpha \log x \sin\left(\frac{\alpha}{2-x}\right) dx = \alpha \int_{\alpha/2}^\infty \left(2 - \frac{\alpha}{u}\right)^\alpha \log\left(2 - \frac{\alpha}{u}\right) \frac{\sin u}{u^2} du \\
&= \alpha \left[\int_{\alpha/2}^\pi \left(2 - \frac{\alpha}{u}\right)^\alpha \log\left(2 - \frac{\alpha}{u}\right) \frac{\sin u}{u^2} du \right. \\
&\qquad \left. + \sum_{k=1}^\infty \int_{k\pi}^{(k+1)\pi} \left(2 - \frac{\alpha}{u}\right)^\alpha \log\left(2 - \frac{\alpha}{u}\right) \frac{\sin u}{u^2} du \right], \qquad (9.2) \\
I_3(\alpha) &= \int_0^2 \frac{x^\alpha}{2-x} \cos\left(\frac{\alpha}{2-x}\right) dx = \int_{\alpha/2}^\infty \left(2 - \frac{\alpha}{u}\right)^\alpha \frac{\cos u}{u} du \\
&= \int_{\alpha/2}^{3\pi/2} \left(2 - \frac{\alpha}{u}\right)^\alpha \frac{\cos u}{u} du + \sum_{k=2}^\infty \int_{(k-\frac{1}{2})\pi}^{(k+\frac{1}{2})\pi} \left(2 - \frac{\alpha}{u}\right)^\alpha \frac{\cos u}{u} du. \qquad (9.3)
\end{aligned}$$

Each of the integrals appearing in the final expressions of (9.1), (9.2), and (9.3) can be easily and accurately determined using standard numerical quadrature. Moreover, the convergence of the alternating series in (9.1), (9.2), and (9.3) can be accelerated by repeated averaging of the partial sums.

9.2 Code Listing

The solution to this problem was coded in Fortran 77 and consists of the single file `problem9.f`, listed below. It requires the presence of the routine `DQAG` from the numeric quadrature package `QUADPACK`.

```
c  solution of problem 9 in the hundred-dollar, hundred-digit challenge
c
c  Kim McInturff
c  3/8/2002
c
c  'i1grand' is the integrand of the function I_1 defined in the
c     'Solution to Problem 9' notes
c
c  'I_1' is written to fort.11 as a function of 'alpha'
c  'I_2' is written to fort.12 as a function of 'alpha'
c  'I_3' is written to fort.13 as a function of 'alpha'
c  'I' is written to fort.21 as a function of 'alpha'

      implicit none

      integer limit, lenw, last
      parameter(limit=256, lenw=4*limit)

      integer iwork(limit)
      double precision work(lenw)

      integer key, neval, ier

      double precision i1grand, i2grand, i3grand

      double precision a, b, epsabs, epsrel, result
      double precision abserr
      double precision alpha, pi

      integer m, k, nk, navg
      integer n, nn, nstop

      parameter(nk=35, navg=14)

      double precision term0, partsum(nk), partsacc(navg+1)
      double precision i1, i2, i3, funci, iprime

      external i1grand, i2grand, i3grand

      common /params/ alpha

      pi=3.14159265358979d0

      epsabs=1.d-13
      epsrel=1.d-10
      key=3

      do m=1,5000

c       alpha=0.001d0*m
```

```
c       alpha=0.75d0 + m*1.d-5
c       alpha=0.7855d0 + m*1.d-7
c       alpha=0.78593d0 + m*1.d-9
c       alpha=0.78593365 + m*1.d-11
c       alpha=0.785933674 + m*1.d-13
        alpha=0.78593367435 + m*1.d-14

c loop for I1
        a=0.5d0*alpha
        b=pi

        call dqag(i1grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
        term0=result

        a=pi
        b=2.d0*pi
        call dqag(i1grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
        partsum(1)=result

        do k=2,nk
          a=k*pi
          b=(k+1)*pi
          call dqag(i1grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
          partsum(k)=partsum(k-1)+result
          if (k.ge.(nk-navg)) then
            n=k+1-(nk-navg)
            partsacc(n)=partsum(k)
          end if
        end do
        do n=1,navg
          nstop=navg+1-n
          do nn=1,nstop
            partsacc(nn)=0.5d0*(partsacc(nn)+partsacc(nn+1))
          end do
        end do

        i1=alpha*(term0+partsacc(1))
        write(11,*) alpha, i1
        funci=(2.d0+sin(1.d1*alpha))*i1
        write(21,*) alpha, funci

c loop for I2
        a=0.5d0*alpha
        b=pi

        call dqag(i2grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
        term0=result

        a=pi
        b=2.d0*pi
        call dqag(i2grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
```

```
        partsum(1)=result

        do k=2,nk
          a=k*pi
          b=(k+1)*pi
          call dqag(i2grand,a,b,epsabs,epsrel,key,result,abserr,
     *             neval,ier,limit,lenw,last,iwork,work)
          partsum(k)=partsum(k-1)+result
          if (k.ge.(nk-navg)) then
            n=k+1-(nk-navg)
            partsacc(n)=partsum(k)
          end if
        end do
        do n=1,navg
          nstop=navg+1-n
          do nn=1,nstop
            partsacc(nn)=0.5d0*(partsacc(nn)+partsacc(nn+1))
          end do
        end do

        i2=alpha*(term0+partsacc(1))
        write(12,*) alpha, i2

c loop for I3
        a=0.5d0*alpha
        b=1.5d0*pi

        call dqag(i3grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
        term0=result

        a=1.5d0*pi
        b=2.5d0*pi
        call dqag(i3grand,a,b,epsabs,epsrel,key,result,abserr,
     *            neval,ier,limit,lenw,last,iwork,work)
        partsum(1)=result

        do k=2,nk
          a=(k+0.5d0)*pi
          b=(k+1.5d0)*pi
          call dqag(i3grand,a,b,epsabs,epsrel,key,result,abserr,
     *             neval,ier,limit,lenw,last,iwork,work)
          partsum(k)=partsum(k-1)+result
          if (k.ge.(nk-navg)) then
            n=k+1-(nk-navg)
            partsacc(n)=partsum(k)
          end if
        end do
        do n=1,navg
          nstop=navg+1-n
          do nn=1,nstop
            partsacc(nn)=0.5d0*(partsacc(nn)+partsacc(nn+1))
c            if (nstop.le.4) then
c              write(6,*) 'nn, partsacc(nn): ', nn, partsacc(nn)
c            end if
          end do
```

```
  end do

  i3=term0+partsacc(1)
  write(13,*) alpha, i3
  iprime=1.d1*cos(1.d1*alpha)*i1+(2.d0+sin(1.d1*alpha))*(i2+i3)
  write(22,*) alpha, iprime

end do

stop
end


double precision function i1grand(u)

double precision alpha, u

common /params/ alpha

i1grand=(2.d0-alpha/u)**alpha*sin(u)/(u*u)

return
end


double precision function i2grand(u)

double precision alpha, u

common /params/ alpha

i2grand=(2.d0-alpha/u)**alpha*log(2.d0-alpha/u)*sin(u)/(u*u)

return
end


double precision function i3grand(u)

double precision alpha, u

common /params/ alpha

i3grand=(2.d0-alpha/u)**alpha*cos(u)/u

return
end
```

10 Problem 10

A particle at the center of a 10×1 rectangle undergoes Brownian motion (i.e., 2D random walk with infinitesimal step lengths) till it hits the boundary. What is the probability that it

hits at one of the ends rather than at one of the sides?

10.1 Solution Method

Consider the general case of two-dimensional Brownian motion of a particle P in the x–y plane within a rectangle R of length $2a$ and width $2b$ placed so that its vertices are at $(\pm a, \pm b)$. The two-dimensional Brownian motion may be decomposed into two statistically independent Brownian motions $x(t)$ and $y(t)$ by projection onto the x and y axes, respectively.

It is then of interest to determine the probability that a particle P experiencing Brownian motion along the v axis remains within a given interval $(-p, p)$. Assume P is located at v at time $t = 0$ where $|v| < p$. The probability of remaining in the interval $(-p, p)$ over the time span $[0, t]$ is given by the Feynman-Kac formula as the function $F(t, v; p)$ satisfying the boundary value problem

$$\begin{aligned} \frac{\partial F}{\partial t}(t, v) &= \frac{1}{2}\frac{\partial^2 F}{\partial v^2}(t, v) && t > 0,\ |v| < p, \\ F(t, \pm p) &= 0 && t > 0, \\ F(0, v) &= 1 && |v| < p. \end{aligned}$$

It is clear that $F(t, v)$ should be symmetric in the variable v.

Employing the technique of separation of variables, assume eigenfunctions of the form $F(t, v) = A(t)B(v)$. The separation constant λ leads to

$$\frac{1}{A}\frac{dA}{dt} = \frac{1}{2B}\frac{d^2B}{dv^2} = -\frac{\lambda}{2},$$

so the ordinary differential equations of interest are

$$\frac{dA}{dt} = -\frac{\lambda}{2}A, \qquad \frac{d^2B}{dv^2} = -\lambda B.$$

Eigenfunctions of the ordinary differential equation for A are of the form $A(t) = e^{-\lambda t/2}$ where λ is chosen to be positive to ensure $\lim_{t\to\infty} F(t, v) = 0$. Symmetric eigenfunctions of the ordinary differential equation for B are of the form $B(v) = \cos(\sqrt{\lambda}\, v)$. The boundary conditions at $\pm p$ imply the only eigenfunctions of interest are

$$B_k(v) = \cos\frac{(2k+1)\pi v}{2p} \qquad k = 0, 1, 2, \ldots$$

with corresponding eigenvalues

$$\lambda_k = \frac{(2k+1)^2\pi^2}{4p^2} \qquad k = 0, 1, 2, \ldots.$$

Thus, the desired solution of the boundary value problem takes the form

$$F(t, v) = \sum_{k=0}^{\infty} c_k e^{-\frac{(2k+1)^2\pi^2 t}{8p^2}} \cos\frac{(2k+1)\pi v}{2p}.$$

The initial condition implies

$$1 = \sum_{k=0}^{\infty} c_k \cos\frac{(2k+1)\pi v}{2p}.$$

Multiplying by $\cos\frac{(2\ell+1)\pi v}{2p}$ and integrating over the interval $(-p, p)$ yields

$$\int_{-p}^{p} \cos\frac{(2\ell+1)\pi v}{2p} dv = c_\ell \int_{-p}^{p} \cos^2\frac{(2p+1)\pi v}{2p} dv,$$

or

$$c_\ell = \frac{4(-1)^\ell}{(2\ell+1)\pi} \qquad \ell = 0, 1, 2, \ldots.$$

That is,

$$F(t, v; p) = \frac{4}{\pi}\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} e^{-\frac{(2k+1)^2\pi^2 t}{8p^2}} \cos\frac{(2k+1)\pi v}{2p},$$

and in the specific case that the Brownian motion begins at the center of the interval $(-p, p)$

$$F(t, 0; p) = \frac{4}{\pi}\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} e^{-\frac{(2k+1)^2\pi^2 t}{8p^2}}. \tag{10.1}$$

Note that

$$G(t; p) = 1 - F(t, 0; p) = 1 - \frac{4}{\pi}\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} e^{-\frac{(2k+1)^2\pi^2 t}{8p^2}}$$

is the probability distribution function associated with the exit time of P from the interval $(-p, p)$. The corresponding probability density function is

$$g(t; p) = \frac{dG}{dt}(t; p) = \frac{\pi}{2p^2}\sum_{k=0}^{\infty} (-1)^k (2k+1) e^{-\frac{(2k+1)^2\pi^2 t}{8p^2}}. \tag{10.2}$$

Returning to the general 2D problem, the independence of the motion in the x and y directions implies the desired probability of P hitting an end rather than a side of the rectangle is

$$Q_{ab} = \int_0^{\infty} g(t; a) F(t, 0; b)\, dt. \tag{10.3}$$

Inserting (10.1) and (10.2) into (10.3) yields

$$\begin{aligned} Q_{ab} &= \frac{2}{a^2}\sum_{n=0}^{\infty}\sum_{m=0}^{\infty}(-1)^{m+n}\frac{2m+1}{2n+1}\int_0^{\infty} e^{-\frac{t\pi^2}{2}[\frac{(2m+1)^2}{4a^2}+\frac{(2n+1)^2}{4b^2}]}\,dt \\ &= \frac{4}{a^2\pi^2}\sum_{n=0}^{\infty}\sum_{m=0}^{\infty}\frac{2m+1}{2n+1}\frac{(-1)^{m+n}}{\frac{(2m+1)^2}{4a^2}+\frac{(2n+1)^2}{4b^2}} \\ &= \frac{16}{\pi^2}\sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1}\sum_{m=0}^{\infty}\frac{(-1)^m(2m+1)}{(2m+1)^2+[(2n+1)a/b]^2} \end{aligned} \tag{10.4}$$

The series over m in (10.4) can be summed in closed form. Evaluating the Mittag-Leffler expansion

$$\operatorname{sech} z = \pi\sum_{m=0}^{\infty}\frac{(-1)^m(2m+1)}{[(2m+1)\pi/2]^2+z^2}$$

at $z = \frac{(2n+1)a\pi}{2b}$ yields

$$\begin{aligned} \operatorname{sech}\frac{(2n+1)a\pi}{2b} &= \pi\sum_{m=0}^{\infty}\frac{(-1)^m(2m+1)}{[\frac{(2m+1)\pi}{2}]^2+[\frac{(2n+1)a\pi}{2b}]^2} \\ &= \frac{4}{\pi}\sum_{m=0}^{\infty}\frac{(-1)^m(2m+1)}{(2m+1)^2+[(2n+1)a/b]^2} \end{aligned}$$

or

$$\sum_{m=0}^{\infty}\frac{(-1)^m(2m+1)}{(2m+1)^2+[(2n+1)a/b]^2} = \frac{\pi}{4\cosh\frac{(2n+1)a\pi}{2b}},$$

so that (10.4) reduces to the rapidly convergent, one-dimensional series

$$Q_{ab} = \frac{4}{\pi}\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)\cosh\frac{(2n+1)a\pi}{2b}}.$$

Specifically, in the case where $a = 5$, $b = 1/2$,

$$Q_{5,1/2} = \frac{4}{\pi}\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)\cosh 5(2n+1)\pi} \approx 3.8375879792512\times 10^{-7}.$$

The given approximate numerical result is found to 14 significant digits by summing only two terms of the series.

10.2 Code Listing

The solution to this problem was coded in MATLAB and consists of the file `problem10.m` listed below.

```
function problem10
%
% problem10.m
% Solution of problem 10 in the hundred-dollar, hundred-digit
% challenge.
%
% Kim McInturff and Peter Simon
% 3/27/2002
%

Qsum = 0
sgn = -1;
for n = 0:20
  tn = 2*n + 1;
  sgn = -sgn;
  Qsum = Qsum + 4/pi * sgn / (tn * cosh(tn * 5 * pi));
  fprintf('%d %20.14g\n',[n, Qsum]);
end
```

References

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover Publications, 1972.

[2] M. R. Spiegel, *Theory and Problems of Laplace Transforms*, McGraw-Hill, 1965.

[3] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, corrected and enlarged edition, Academic Press, 1980.

[4] D. Bailey, "Multiprecision translation and execution of Fortran programs," *ACM Trans. Math. Software*, vol. 19, no. 3 (Sept. 1993), pp. 288–319.

[5] Wrench, J. W. Jr. "Concerning two series for the gamma function." Math. Comput. vol. 22, pp. 617–626, 1968.