User-Written MATLAB Functions

We have saved each MATLAB function contained in this appendix as the text file *function_name*.m. Electronic copies of these files can be downloaded from the web site http://www.radford.edu/npsigmon/algebrabook.html.

Contents

Functions for Chapter 1

```
function res = Irreduc(f, p)
    S = symengine;
    v = feval(S, 'indets', f);
    pv = strcat('[', char(v), ']');
    pd = strcat('Dom::IntegerMod(', num2str(p), ')');
    res = feval(S, 'irreducible', feval(S, 'poly', f, pv, pd));


function res = Primitive(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
   if strcmp(class(p), 'sym') == 1
      pd = strcat('Dom::IntegerMod(', char(p), ')');
   else
      pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   end
   fp = feval(S, 'poly', f, pv, pd);
   if ~feval(S, 'irreducible', fp)
      res = 'FALSE';
   else
      d = feval(S, 'degree', fp);
      n = p^d;
      pdiv = feval(symengine, 'numlib::primedivisors', n-1);
      res = 'TRUE';
      xt = feval(S, 'poly', v(1), pv, pd);
      for i = 1:length(pdiv)
          e = double((n-1)/pdiv(i));
          if feval(S, 'expr', feval(S, 'powermod', xt, e, ...
          fp)) == 1
             res = 'FALSE';
          end
      end
   end


function res = Powmod(v, e, f, n, p)
   S = symengine;
   pi = feval(S, 'indets', f);
   pv = strcat('[', char(pi), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   xt = feval(S, 'poly', v, pv, pd);
```

```
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'powermod', xt, e, fp));
```

Functions for Chapter 2

```
function res = Primitive(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
   if strcmp(class(p), 'sym') == 1
      pd = strcat('Dom::IntegerMod(', char(p), ')');
   else
      pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   end
   fp = feval(S, 'poly', f, pv, pd);
   if ~feval(S, 'irreducible', fp)
      res = 'FALSE';
   else
      d = feval(S, 'degree', fp);
      n = p^d;
      pdiv = feval(symengine, 'numlib::primedivisors', n-1);
      res = 'TRUE';
      xt = feval(S, 'poly', v(1), pv, pd);
      for i = 1:length(pdiv)
          e = double((n-1)/pdiv(i));
          if feval(S, 'expr', feval(S, 'powermod', xt, e, ...
          fp)) == 1
             res = 'FALSE';
          end
      end
   end


function res = Powmod(v, e, f, n, p)
   S = symengine;
   pi = feval(S, 'indets', f);
   pv = strcat('[', char(pi), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   xt = feval(S, 'poly', v, pv, pd);
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'powermod', xt, e, fp));
```

Functions Used in Chapter 4

```
function res = Primitive(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
   if strcmp(class(p), 'sym') == 1
      pd = strcat('Dom::IntegerMod(', char(p), ')');
   else
      pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   end
   fp = feval(S, 'poly', f, pv, pd);
   if ~feval(S, 'irreducible', fp)
      res = 'FALSE';
   else
      d = feval(S, 'degree', fp);
      n = p^d;
      pdiv = feval(symengine, 'numlib::primedivisors', n-1);
      res = 'TRUE';
      xt = feval(S, 'poly', v(1), pv, pd);
      for i = 1:length(pdiv)
          e = double((n-1)/pdiv(i));
          if feval(S, 'expr', feval(S, 'powermod', xt, e, ...
          fp)) == 1
             res = 'FALSE';
          end
      end
   end


function res = Powmod(v, e, f, n, p)
   S = symengine;
   pi = feval(S, 'indets', f);
   pv = strcat('[', char(pi), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   xt = feval(S, 'poly', v, pv, pd);
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'powermod', xt, e, fp));


function table = ftable(p,field)
    syms a
    [r,c] = size(field);
    e = find(field == p);
```

```
    if e == c-1
       table = 1;
    elseif e == c
       table = 0;
    else
       table = a^e;
    end
```

```
function res = Factor(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'factor', fp));
```

```
function res = Rem(f, pp, v, p)
   S = symengine;
   pv = strcat('[', char(v), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   fp = feval(S, 'poly', f, pv, pd);
   ppp = feval(S, 'poly', pp, pv, pd);
   resa = feval(S, 'expr', feval(S, 'divide', fp, ppp));
   res = resa(2);
```

```
function res = Expand(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'expand', fp));
```

Functions Used in Chapter 5

```
function res = Primitive(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
```

```
   if strcmp(class(p), 'sym') == 1
      pd = strcat('Dom::IntegerMod(', char(p), ')');
   else
      pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   end
   fp = feval(S, 'poly', f, pv, pd);
   if ~feval(S, 'irreducible', fp)
      res = 'FALSE';
   else
      d = feval(S, 'degree', fp);
      n = p^d;
      pdiv = feval(symengine, 'numlib::primedivisors', n-1);
      res = 'TRUE';
      xt = feval(S, 'poly', v(1), pv, pd);
      for i = 1:length(pdiv)
          e = double((n-1)/pdiv(i));
          if feval(S, 'expr', feval(S, 'powermod', xt, e, ...
          fp)) == 1
             res = 'FALSE';
          end
      end
   end


function res = Powmod(v, e, f, n, p)
   S = symengine;
   pi = feval(S, 'indets', f);
   pv = strcat('[', char(pi), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   xt = feval(S, 'poly', v, pv, pd);
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'powermod', xt, e, fp));


function table = ftable(p,field)
    syms a
    [r,c] = size(field);
    e = find(field == p);
    if e == c-1
       table = 1;
    elseif e == c
       table = 0;
    else
```

```
       table = a^e;
    end


function res = Rem(f, pp, v, p)
   S = symengine;
   pv = strcat('[', char(v), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   fp = feval(S, 'poly', f, pv, pd);
   ppp = feval(S, 'poly', pp, pv, pd);
   resa = feval(S, 'expr', feval(S, 'divide', fp, ppp));
   res = resa(2);


function g = rscoeff(f, x, p, a)
    syms ng
    S = symengine;
    if feval(S, 'degree', f, x) > 0
       fs = sym2poly(2^(feval(S, 'degree', p)));
       for i = 1:fs-1
           field(i) = Powmod(a, i, p, a, 2);
       end;
       field(fs) = 0;
       g = feval(S, 'expand', f);
       ng = 0;
       dg = sym2poly(feval(S, 'degree', g, x));
       for j = 0:dg
           cg = feval(S, 'coeff', g, x, j);
           ncg = feval(S, 'numer', cg);
           dcg = feval(S, 'denom', cg);
           cg = ftable(Rem(ncg, p, a, 2), field) / ...
                ftable(Rem(dcg, p, a, 2), field);
           degdcg = sym2poly(feval(S, 'degree', dcg, a));
           if degdcg > 0
              cg = cg*a^(fs-1);
           end
           ng = ng + cg*x^j;
       end
       g = feval(S, 'expr', feval(S, 'poly', ng, ...
           'Dom::IntegerMod(2)'));
    else
       g = f;
    end
```

```
function sp = sortpoly(p, v)
    S = symengine;
    if feval(S, 'degree', p) > 0
       pv = strcat('[', char(v), ']');
       sp = char(feval(S, 'poly', p, pv));
       sp = sp(6:findstr(sp, ',')-1);
    else
       sp = char(p);
    end


function binvect = binword(cw, n, p, a, ml)
    S = symengine;
    vs = [];
    syms x
    for i= 0:ml
        pco = feval(S, 'coeff', cw, x, i);
        if pco ~= 0
           dga = sym2poly(feval(S, 'degree', pco, a));
           pco = Powmod(a, dga, p, a, 2);
        end
        for j = 0:n-1
            vs = [vs feval(S, 'coeff', pco, a, j)];
        end
    end
    binvect = vs;


function pcoeff = bincoeff(n,bmess)
    syms a
    pcoeff = [];
    [bkr bkc] = size(bmess);
    i = 0;
    k = 0;
    pcoeff = [];
    while i < bkc
          poly = 0;
          for j = 1:n
              poly = poly + bmess(i+j)*a^(j-1);
          end
          pcoeff = [pcoeff poly];
          k = k+1;
          i = k*n;
```

```
    end
    pcoeff;


function e = rseuclid(t, f, g, z, p, a)
    S = symengine;
    rm1 = Expand(f, 2);
    r = Expand(g, 2);
    um1 = 1;
    u = 0;
    vm1 = 0;
    v = 1;
    dr = sym2poly(feval(S, 'degree', r, z));
    fs = sym2poly(2^(feval(S, 'degree', p)));
    mf = a^(fs-1);
    disp(' ');
    while dr >= t
          trp1 = feval(S, 'divide', rm1, r, strcat('[', ...
          char(z), ']'));
          rp1 = rscoeff(feval(S, 'expand', mf*trp1(2)), ...
          z, p, a);
          q = rscoeff(feval(S, 'expand', mf*trp1(1)), ...
          z, p, a);
          vp1 = feval(S, 'expand', vm1-v*q);
          vm1 = v;
          v = vp1;
          v = rscoeff(v, z, p, a);
          up1 = feval(S, 'expand', um1-u*q);
          um1 = u;
          u = up1;
          u = rscoeff(u, z, p, a);
          rm1 = r;
          r = rp1;
          disp(['Q:', blanks(1), sortpoly(q, z), '   R: ', ...
                blanks(1), sortpoly(r, z),'   V: ', ...
                blanks(1), sortpoly(v, z), '   U: ', ...
                blanks(1), sortpoly(u, z)])
          dr = sym2poly(feval(S, 'degree', r, z));
    end
    fprintf('\n')
    e = [q, r, v, u];
```

Functions Used in Chapter 6

```
function let = ltable(message)
    letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    for i = 1:length(message)
        let(i) = find(letters == message(i))-1;
    end
    let;


function res = printletterfreq(mess)
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    flist = [];
    alist = [];
    for i = 1:length(abet)
        ct = 0;
        for j = 1:length(mess)
            if abet(i) == mess(j)
               ct = ct+1;
            end
        end
        if ct > 0
           flist = [flist ct];
           alist = [alist i];
        end
    end
    [flist n] = sort(flist, 'ascend');
    alist = alist(n);
    for i = numel(flist):-1:1
        fprintf('%s %s %s %2.0f %s \n', 'Letter', ...
        abet(alist(i)), 'occurs', flist(i), 'times')
    end
```

Functions Used in Chapter 7

```
function let = ltable(message)
    letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    for i = 1:length(message)
        let(i) = find(letters == message(i))-1;
    end
    let;
```

```
function f = vigenere(x,k)
    for i = 1:numel(x)
        f(i) = mod(x(i) + k(mod(i-1, numel(k))+1), 26);
    end
    f;


function p = stringprint(str,num)
    fprintf('\n');
    q = floor(length(str)/num);
    for i = 1:q
        fprintf('%s \n', str((i-1)*num+1:i*num));
    end
    if mod(length(str), num) ~= 0
       fprintf('%s \n', str(q*num+1:end));
    end
    fprintf('\n');


function flist = letterfreq(mess)
    ms = char(mess);
    flist = [];
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    for i=1:length(abet)
        ct = 0;
        for j=1:length(ms)
            if abet(i) == ms(j)
               ct = ct+1;
            end
        end
        flist = [flist,ct];
    end
    flist = flist/sym(length(ms));


function icoin = ic(cm)
    icoin = [];
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    c = cellstr(cm);
    l = length(abet);
    for i = 1:numel(c)
        n = length(char(c(i)));
        ms = char(c(i));
        f = [];
```

```
        for j=1:l
            ct = 0;
            for k=1:length(ms)
                if abet(j) == ms(k)
                   ct = ct+1;
                end
            end
        f = [f, ct/n];
        end
        icoini = 0;
        for j = 1:numel(f)
            icoini = icoini + n*f(j)*(n*f(j)-1);
        end
        icoini = double(icoini/(n*(n-1)));
        icoin = [icoin,icoini];
    end


function monolist = vigencoset(kwl,mess)
    monolist = [];
    for i=1:kwl
        monolist{i} = [mess(i:kwl:numel(mess))];
    end
```

Functions Used in Chapter 8

```
function cno = tonumber(imess)
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    if isstr(imess)
       im = {imess};
    else
       im = imess;
    end
    for j = 1:numel(im)
        mess = im{j};
        sl = length(mess);
        cn = 0;
        for ii=1:sl
            sn = sym(find(mess(ii) == abet)-1);
            cn = 100*cn + sn;
        end
        cno(j) = cn;
    end
```

```
    cno;


function bans = toletter(num)
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    bans = '';
    for j = 1:numel(num);
        cn = sym(num(j));
        sl = floor(log10(cn)/2)+1;
        ans = '';
        for ii=1:sl
            cn = cn/100;
            cs = abet(double(frac(cn)*100+1));
            ans = strcat(cs,ans);
            cn = floor(cn);
        end
        bans = strcat(bans, ans);
    end


function p = LengthSplitString(str,num)
    q = floor(length(str)/num);
    for i = 1:q
        p{i} = str((i-1)*num+1:i*num);
    end
    if mod(length(str), num) ~= 0
       p{q+1} = str(q*num+1:end);
    end
    p;

function x = mlog(b, a, m)
   S = symengine;
   N = floor(sqrt(m)) + 1;
   Ap(1) = 1;
   for j = 2:N
      Ap(j) = mod(Ap(j-1)*a, m);
   end
   f = 0;
   ani = feval(S, 'powermod', a, N, m);
   ani = mod(ani^(-1), m);
   k = 0;
   r = mod(b, m);
   while (f < 1) & (k < N)
      n = find(Ap == r);
```

```
      if numel(n) > 0
         f = 1;
      else
         k = k + 1;
         r = mod(r*ani, m);
      end
   end
   if numel(n) > 0
      x = (n(1)-1) + N*k;
   else
      x = 'FAIL';
   end
   x;
```

Functions Used in Chapter 9

```
function res = Primitive(f, p)
   S = symengine;
   v = feval(S, 'indets', f);
   pv = strcat('[', char(v(1)), ']');
   if strcmp(class(p), 'sym') == 1
      pd = strcat('Dom::IntegerMod(', char(p), ')');
   else
      pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   end
   fp = feval(S, 'poly', f, pv, pd);
   if ~feval(S, 'irreducible', fp)
      res = 'FALSE';
   else
      d = feval(S, 'degree', fp);
      n = p^d;
      pdiv = feval(symengine, 'numlib::primedivisors', n-1);
      res = 'TRUE';
      xt = feval(S, 'poly', v(1), pv, pd);
      for i = 1:length(pdiv)
          e = double((n-1)/pdiv(i));
          if feval(S, 'expr', feval(S, 'powermod', xt, e, ...
          fp)) == 1
             res = 'FALSE';
          end
      end
```

```
   end


function res = Powmod(v, e, f, n, p)
   S = symengine;
   pi = feval(S, 'indets', f);
   pv = strcat('[', char(pi), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   xt = feval(S, 'poly', v, pv, pd);
   fp = feval(S, 'poly', f, pv, pd);
   res = feval(S, 'expr', feval(S, 'powermod', xt, e, fp));


function res = Rem(f, pp, v, p)
   S = symengine;
   pv = strcat('[', char(v), ']');
   pd = strcat('Dom::IntegerMod(', num2str(p), ')');
   fp = feval(S, 'poly', f, pv, pd);
   ppp = feval(S, 'poly', pp, pv, pd);
   resa = feval(S, 'expr', feval(S, 'divide', fp, ppp));
   res = resa(2);


function ecurve = epoints(ec, x, ub, p)
    S = symengine;
    pct = 0;
    k = 0;
    p = sym(p);
    while k <= double(p-1) & (pct <= ub)
          z = mod(subs(ec, x, k), p);
          if z == 0
             pct = pct+1;
             ecurve{pct} = [k,z];
          end
          if feval(S, 'powermod', z, (p-1)/2, p) ...
          == sym(1)
             z = feval(S, 'powermod', z, ...
             (p+1)/4, p);
             ecurve{pct+1} = [k,z];
             ecurve{pct+2} = [k, mod(-z, p)];
             pct = pct+2;
          end
          k = k+1;
    end
```

```
    if pct > ub
       pct = ub;
    end
    ecurve = ecurve(1:pct);


function s = estring(e)
    s = '';
    ne = numel(e);
    for i = 1:ne-1
        if length(char(e{i})) > 1
           e{i} = [sym(e{i}(1)), sym(e{i}(2))];
           s = strcat(s, '[', char(e{i}(1)), ',', ...
           char(e{i}(2)), ']', ',');
        else
           e{i} = sym(e{i});
           s = strcat(s, char(e{i}), ',');
        end
    end
    if length(char(e{ne})) > 1
       e{ne} = [sym(e{ne}(1)), sym(e{ne}(2))];
       s = strcat(s, '[', char(e{ne}(1)), ',', ...
       char(e{ne}(2)), ']');
    else
       e{ne} = sym(e{ne});
       s = strcat(s, char(e{ne}));
    end


function p = stringprint(str,num)
    fprintf('\n');
    q = floor(length(str)/num);
    for i = 1:q
        fprintf('%s \n', str((i-1)*num+1:i*num));
    end
    if mod(length(str), num) ~= 0
       fprintf('%s \n', str(q*num+1:end));
    end
    fprintf('\n');


function res = addec(le, re, c, p)
    cle = mod(le,p);
    cre = mod(re,p);
```

```
    if isequal(double(cle), 0) | isequal(double(cre), 0)
       res = cle + cre;
    elseif cle(1) == cre(1) & cle(2) == mod(-cre(2), p)
       res = 0;
    else
       if cle(1) == mod(cre(1), p) & cle(2) == mod(cre(2), p)
          lambda = mod(((3*cle(1)^2+c)/2/cle(2)), p);
       else
          lambda = mod((cre(2)-cle(2))/(cre(1)-cle(1)), p);
       end
       x3 = mod((lambda^2-cle(1)-cre(1)), p);
       y3 = mod((lambda*(cle(1)-x3)-cle(2)), p);
       res = [x3,y3];
    end


function t = p3mod4(s)
    S = symengine;
    t = feval(S, 'nextprime', sym(s+1));
    while double(mod(t,4)) ~= 3
          t = feval(S, 'nextprime', sym(t+1));
    end


function y = elgamal(alpha, e, c, p)
    calpha = alpha;
    n = e;
    y = 0;
    while double(n) > 0
          if mod(n, 2) == 1
             y = addec(calpha, y, c, p);
          end
          n = floor(n/2);
          calpha = addec(calpha, calpha, c, p);
    end


function p = LengthSplitString(str,num)
    q = floor(length(str)/num);
    for i = 1:q
        p{i} = str((i-1)*num+1:i*num);
    end
    if mod(length(str), num) ~= 0
       p{q+1} = str(q*num+1:end);
```

```
    end
    p;


function cno = tonumber(imess)
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    if isstr(imess)
       im = {imess};
    else
       im = imess;
    end
    for j = 1:numel(im)
        mess = im{j};
        sl = length(mess);
        cn = 0;
        for ii=1:sl
            sn = sym(find(mess(ii) == abet)-1);
            cn = 100*cn + sn;
        end
        cno(j) = cn;
    end
    cno;


function bans = toletter(num)
    abet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    bans = '';
    for j = 1:numel(num);
        cn = sym(num(j));
        sl = floor(log10(cn)/2)+1;
        ans = '';
        for ii=1:sl
            cn = cn/100;
            cs = abet(double(frac(cn)*100+1));
            ans = strcat(cs,ans);
            cn = floor(cn);
        end
        bans = strcat(bans, ans);
    end


function p = LengthSplitList(list,num)
    q = floor(length(list)/num);
    for i = 1:q
```

```
        p{i} = list((i-1)*num+1:i*num);
    end
    if mod(length(list), num) ~= 0
       p{q+1} = [list(q*num+1:end), ...
       zeros(1:num-numel(list(q*num+1:end)))];
    end
    p;
```

Functions Used in Chapter 10

```
function res = Irreduc(f, p)
    S = symengine;
    v = feval(S, 'indets', f);
    pv = strcat('[', char(v), ']');
    pd = strcat('Dom::IntegerMod(', num2str(p), ')');
    res = feval(S, 'irreducible', feval(S, 'poly', f, pv, pd));
```

```
function [St,iSt] = sboxtable(f,x)
    se = symengine;
    v = [1 0 0 0 1 1 1 1];
    S = [v];
    for i = 2:8
        v = circshift(v, [1,1]);
        S = [S ; v];
    end
    d = double(feval(se, 'degree', f, x));
    l = d/2;
    for i = 0:15
        rt = dec2bin(i);
        while length(rt) < l
              rt = strcat('0', rt);
        end
        for j = 0:15
            ct = dec2bin(j);
            while length(ct) < l
                  ct = strcat('0', ct);
            end
            pv = mod(double(strcat(rt,ct)), 2);
            ffe = poly2sym(pv);
            if ffe ~= sym(0)
               rm1 = ffe;
               r = f;
```

```
            um1 = 1;
            u = 0;
            dr = double(feval(se, 'degree', r, x));
            while dr >= 1
                  trp1 = feval(se, 'divide', rm1, r, ...
                         strcat('[', char(x), ']'));
                  rp1 = mod(trp1(2), 2);
                  q = mod(trp1(1), 2);
                  up1 = mod(um1 - u*q, 2);
                  um1 = u;
                  u = up1;
                  rm1 = r;
                  r = rp1;
                  dr = double(feval(se, 'degree', r, x));
            end
            y = sym2poly(u);
         else
            y = 0;
         end
         while numel(y) < d
               y = [0 y];
         end
         y = fliplr(y);
         z = S * y' + [1 1 0 0 0 1 1 0]';
         St(i+1, j+1) = ...
         mod(poly2sym(fliplr(z')), 2);
         iz = mod(fliplr(z'), 2);
         ii = bin2dec(int2str(iz(1:l)));
         jj = bin2dec(int2str(iz(l+1:d)));
         iSt(ii+1, jj+1) = ffe;
      end
   end
   [St,iSt];


function z = sbox(ffe, St, f, x)
   v = sym2poly(ffe);
   d = double(feval(symengine, 'degree', f, x));
   while numel(v) < d
         v = [0 v];
   end
   r = bin2dec(int2str(v(1:d/2)));
   c = bin2dec(int2str(v(d/2+1:d)));
```

```
    z = St(r+1, c+1);


function messbl = mblocks(message,bl)
    ml = length(message);
    if ml <= bl
       mres = message;
       for i=1:bl-ml
           mres = strcat(mres, 'A');
       end
       messbl = mres;
    else
       if mod(ml, bl) ~= 0
          mb = floor(ml/bl)+1;
       else
          mb = floor(ml/bl);
       end
       mres = [];
       for i = 1:mb-1
           mres = [mres; message((i-1)*bl+1:i*bl)];
       end
       lblock = message((mb-1)*bl+1:ml);
       ml = length(lblock);
       if ml < bl
          for i=1:bl-ml
              lblock = strcat(lblock, 'A');
          end
       end
       mres = [mres; lblock];
       messbl = cellstr(mres)';
    end


function lpoly = topoly(mess,x)
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    sl = length(mess);
    for i = 1:sl
        let =  find(alphabet == mess(i:i))-1;
        bn = dec2bin(let);
        for j = 1:length(bn)
            bn = mod(double(bn), 2);
            lpoly(i) = poly2sym(bn,x);
        end
```

```
    end
    lpoly;


function wmat = keysched(key, St, f, x)
    nk = length(key);
    nr = nk+6;
    wmat = key;
    for i = nk+1:4*(nr+1)
        v = wmat(:, i-nk);
        j = i-1;
        w = wmat(:,j);
        if mod(j,nk) == 0
           w = circshift(w, [-1,1]);
           for k = 1:numel(w)
               w(k) = sbox(w(k), St, f, x);
           end
           tr = feval(symengine, 'divide', x^((j-nk)/nk), ...
                f, strcat('[', char(x), ']'));
           r = mod(tr(2), 2);
           w(1) = mod(w(1)+r, 2);
        end
        if nk > 6 & mod(j, nk) == 4
           for k = 1:numel(w)
               w(k) = sbox(w(k), St, f, x);
           end
        end
        a = v + w;
        for k = 1:numel(a)
            a(k) = mod(a(k), 2);
        end
        wmat = [wmat,a];
    end
    wmat;


function r = addrkey(M,K)
    r = mod(M+K, 2);
    [m n] = size(r);
    for i = 1:m
        for j = 1:n
            r(i,j) = sort(r(i,j));
        end
```

```
    end
    r;


function r = bytesub(M, St, p, x)
    [m,n] = size(M);
    for i=1:m
        for j = 1:n
            r(i,j) = sbox(M(i,j), St, p, x);
        end
    end
    r;


function res = shiftrow(M)
    [m,n] = size(M);
    res = [];
    for i = 0:size(M)-1
        v = circshift(M(i+1,:), [1,-i]);
        res = [res; v];
    end
    res;


function r = mixcolumn(M, f, x)
    L = [[x,1+x,1,1]; [1,x,1+x,1]; [1,1,x,1+x]; [1+x,1,1,x]];
    r = L*M;
    [m,n] = size(r);
    for i = 1:m
        for j = 1:n
            tr = feval(symengine, 'divide', r(i,j), ...
            f, strcat('[', char(x), ']'));
            r(i,j) = mod(tr(2),2);
        end
    end
    r;


function r = invmixcolumn(M, f, x)
    iL = [[x+x^2+x^3, 1+x+x^3, 1+x^2+x^3, 1+x^3]; [1+x^3, ...
    x+x^2+x^3, 1+x+x^3, 1+x^2+x^3]; [1+x^2+x^3, 1+x^3, ...
    x+x^2+x^3, 1+x+x^3]; [1+x+x^3, 1+x^2+x^3, 1+x^3, ...
    x+x^2+x^3]];
    r = iL*M;
```

```
    [m,n] = size(r);
    for i = 1:m
        for j = 1:n
            tr = feval(symengine, 'divide', r(i,j), ...
            f, strcat('[', char(x), ']'));
            r(i,j) = mod(tr(2),2);
        end
    end
    r;


function r = invaddrkey(M, K, f, x)
    iL = [[x+x^2+x^3, 1+x+x^3, 1+x^2+x^3, 1+x^3]; [1+x^3, ...
    x+x^2+x^3, 1+x+x^3, 1+x^2+x^3]; [1+x^2+x^3, 1+x^3, ...
    x+x^2+x^3, 1+x+x^3]; [1+x+x^3, 1+x^2+x^3, 1+x^3, ...
    x+x^2+x^3]];
    r = M + iL*K;
    [m,n] = size(r);
    for i = 1:m
        for j = 1:n
            tr = feval(symengine, 'divide', r(i,j), ...
            f, strcat('[', char(x), ']'));
            r(i,j) = mod(tr(2),2);
        end
    end
    r;


function res = invshiftrow(M)
    [m,n] = size(M);
    res = [];
    for i = 0:size(M)-1
        v = circshift(M(i+1,:), [1,i]);
        res = [res; v];
    end
    res;


function res = frompoly(polyv,x)
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    res = '';
    for i = 1:numel(polyv)
        s = bin2dec(int2str(sym2poly(polyv(i))));
        let = alphabet(s+1);
```

```
        res = strcat(res,let);
    end
    res;


function res = aesencipher(message, key, St, p, x)
    A = reshape(topoly(message,x), 4, 4);
    Keyin = reshape(topoly(key,x), 4, length(key)/4);
    KS = keysched(Keyin, St, p, x);
    nk = size(Keyin,2);
    nr = nk+6;
    K0 = KS(1:4, 1:4);
    Ai = addrkey(A,K0);
    for i=1:nr-1
        B = bytesub(Ai, St, p, x);
        C = shiftrow(B);
        Dm = mixcolumn(C, p, x);
        Ki = KS(1:4, 4*i+1:4*(i+1));
        Ai = addrkey(Dm,Ki);
    end
    B = bytesub(Ai, St, p, x);
    C = shiftrow(B);
    Ki = KS(1:4, 4*nr+1:4*(nr+1));
    Ai = addrkey(C,Ki);
    [m n] = size(Ai);
    for i = 1:m
        for j = 1:n
            res(i,j) = bin2dec(int2str(sym2poly(Ai(i,j))));
        end
    end
    res = reshape(res, 1, numel(res));


function res = aesdecipher(ctext, key, St, iSt, p, x)
    for i = 1:numel(ctext)
        bn = dec2bin(ctext(i));
        for j = 1:length(bn)
            bn = mod(double(bn), 2);
            ml(i) = poly2sym(bn,x);
        end
    end
    Ai = reshape(ml, 4, 4);
    Keyin = reshape(topoly(key,x), 4, length(key)/4);
```

```
    KS = keysched(Keyin, St, p, x);
    nk = size(Keyin,2);
    nr = nk+6;
    Ki = KS(1:4, 4*nr+1:4*(nr+1));
    Ai = addrkey(Ai,Ki);
    for i = nr-1:-1:1
        Dm = bytesub(Ai, iSt, p, x);
        C = invshiftrow(Dm);
        B = invmixcolumn(C, p, x);
        Ki = KS(1:4, 4*i+1:4*(i+1));
        Ai = invaddrkey(B, Ki, p, x);
    end
    Ai = bytesub(Ai, iSt, p, x);
    Ai = invshiftrow(Ai);
    K0 = KS(1:4, 1:4);
    A = addrkey(Ai,K0);
    polyv = reshape(A, 1, numel(A));
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    res = '';
    for i = 1:numel(polyv)
        s = bin2dec(int2str(sym2poly(polyv(i))));
        let = alphabet(s+1);
        res = strcat(res,let);
    end
    res;
```

Functions for Chapter 11

```
function cres = cicn(n, x)
   S = symengine;
   d =  feval(S, 'numlib::divisors', n);
   p = sym('0');
   for i = 1:numel(d)
       v = sym(strcat(char(x),char(d(i))));
       p = p + feval(S, 'numlib::phi', d(i))*v^(n/d(i));
   end
   p = 1/n*p;
   for i = 1:n
       var{i} = sym(strcat(char(x),num2str(i)));
   end
   cres{1} = p;
   cres{2} = var;
```

```
function cterm = cpinv(pinv, term)
   S = symengine;
   cterm = pinv;
   if feval(S, 'degree', term) ~= feval(S, 'degree', pinv)
      cterm = 0;
   else
      st = sym(strsplit(char(term), '*'));
      for i=1:numel(st)
          v = sym(char(feval(S, 'indets', st(i))));
          d = feval(S, 'degree', st(i));
          cterm = feval(S, 'coeff', cterm, v, d);
      end
   end
```

```
function cres = cidn(n, x)
   S = symengine;
   d =  feval(S, 'numlib::divisors', n);
   p = sym('0');
   for i = 1:numel(d)
       v = sym(strcat(char(x),char(d(i))));
       p = p + feval(S, 'numlib::phi', d(i))*v^(n/d(i));
   end
   for i = 1:n
       var{i} = sym(strcat(char(x),num2str(i)));
   end
   if mod(n, 2) == 1
      p = 1/(2*n)*p + 1/2*var{1}*var{2}^((n-1)/2);
   else
      p = 1/(2*n)*p + 1/4*(var{1}^2*var{2}^((n-2)/2) ...
          + var{2}^(n/2));
   end
   cres{1} = p;
   cres{2} = var;
```

Functions Used in Chapter 12

```
function p = stringprint(str,num)
    fprintf('\n');
    q = floor(length(str)/num);
    for i = 1:q
```

```
        fprintf('%s \n', str((i-1)*num+1:i*num));
    end
    if mod(length(str), num) ~= 0
       fprintf('%s \n', str(q*num+1:end));
    end
    fprintf('\n');


function p = partition(n)
    if n < 0
       p = strcat('Error: Input must be a non-negative ...
       integer but received', blanks(1), char(n));
    elseif n == 0
       p = '[[]]';
    else
       pn = feval(symengine, 'combinat::partitions', n);
       part = firstpart(n);
       p = part;
       for i=2:pn
           part = nextpart(part);
           p = strcat(p, ',', char(1), part);
       end
       p = strcat('[', p, ']');
    end


function r = firstpart(n)
    if n < 0
       r = strcat('Error: Input must be a non-negative ...
       integer but received', blanks(1), char(n));
    elseif n == 0
       r = '[]';
    elseif n == 1
       r = '[1]';
    else
       a = zeros(1, n);
       k = 2;
       a(2) = n;
       x = a(k-1) + 1;
       y = a(k) - 1;
       k = k - 1;
       while x <= y
          a(k) = x;
```

```
            y = y-x;
            k = k+1;
        end
        a(k) = x+y;
        s = 0;
        j = 0;
        r = '[';
        for i = 1:numel(a)
            s = s + a(i);
            if s <= n
               j = j + 1;
               r = [r, int2str(a(i)), ', '];
            end
        end
        r(numel(r)-1) = '';
        r(numel(r)) = ']';
    end


function p = partlist(y,p)
    y = sym(y);
    for i = 1:numel(y)
        p(double(y(i))) = p(double(y(i)))+1;
    end


function r = nextpart(ip)
    pp = sym(ip);
    n = 0;
    for i = 1:numel(pp)
        a(i) = double(pp(i));
        n = n + double(pp(i));
    end
    k = numel(pp);
    x = a(k-1) + 1;
    y = a(k) - 1;
    k = k - 1;
    while x <= y
        a(k) = x;
        y = y-x;
        k = k+1;
    end
    a(k) = x+y;
```

```
    s = 0;
    r = '[';
    for i = 1:numel(a)
        s = s + a(i);
        if s <= n
           r = [r, int2str(a(i)), ', '];
        end
    end
    r(numel(r)-1) = '';
    r(numel(r)) = ']';


function cres = cisn(n,x)
    S = symengine;
    x = sym(x);
    cip = 0;
    for i = 1:double(feval(S, 'combinat::partitions', n))
        P = zeros(1,n);
        if i == 1
           part = firstpart(n);
        else
           part = nextpart(part);
        end
        P = partlist(part,P);
        vt = 1;
        for k = 1:n
            if P(k) > 0
               xk = sym(strcat(char(x),num2str(k)));
               var{k} = xk;
               vt = vt*(1/(sym(k)^P(k)* ...
               feval(S, 'fact', sym(P(k)))))*xk^P(k);
            end
        end
        cip = cip + vt;
    end
    cres{1} = cip;
    cres{2} = var;


function cres = igraph(n,x)
    S = symengine;
    x = sym(x);
    maxv = n;
```

```
cires = 0;
for i = 0:floor((n-1)/2)
    os{i+1} = 2*i+1;
end
for i = 1:floor(n/2)
    es{i} = 2*i;
end
for pnum = 1:double(feval(S, 'combinat::partitions', n))
    if pnum == 1
       part = firstpart(n);
    else
       part = nextpart(part);
    end
    P = zeros(1,n);
    P = partlist(part,P);
    w = 1;
    for k = 1:n
        w = w*(1/(sym(k)^P(k)* feval(S, 'fact', ...
        sym(P(k)))));
    end
    for i = 1:numel(es);
        xid2 = sym(strcat(char(x), num2str(es{i}/2)));
        xi = sym(strcat(char(x), num2str(es{i})));
        w = w*((xid2*xi^((es{i}-2)/2))^P(es{i}));
    end;
    for i = 1:numel(os);
        xi = sym(strcat(char(x), num2str(os{i})));
        w = w*(xi^((os{i}-1)/2*P(os{i})));
    end
    for i = 1:floor(n/2)
        xi = sym(strcat(char(x), num2str(i)));
        w = w*(xi^(i*double(feval(S, 'binomial', P(i), ...
        2))));
    end
    for r=1:n-2
        for s=r+1:n-1
            if (P(r) > 0) & (P(s) > 0)
               xi = sym(strcat(char(x), ...
               num2str(lcm(r,s))));
               w = w*(xi^(gcd(r,s)*P(r)*P(s)));
               maxv = max([maxv, lcm(r,s)]);
            end
        end
```

```
        end
        cires = cires + w;
    end
    for i = 1:maxv
        xi = sym(strcat(char(x), num2str(i)));
        var{i} = xi;
    end
    cres{1} = cires;
    cres{2} = var;
```

Functions for Chapter 13

```
function cres = cicn(n, x)
   S = symengine;
   d =  feval(S, 'numlib::divisors', n);
   p = sym('0');
   for i = 1:numel(d)
       v = sym(strcat(char(x),char(d(i))));
       p = p + feval(S, 'numlib::phi', d(i))*v^(n/d(i));
   end
   p = 1/n*p;
   for i = 1:n
       var{i} = sym(strcat(char(x),num2str(i)));
   end
   cres{1} = p;
   cres{2} = var;


function cres = cidn(n, x)
   S = symengine;
   d =  feval(S, 'numlib::divisors', n);
   p = sym('0');
   for i = 1:numel(d)
       v = sym(strcat(char(x),char(d(i))));
       p = p + feval(S, 'numlib::phi', d(i))*v^(n/d(i));
   end
   for i = 1:n
       var{i} = sym(strcat(char(x),num2str(i)));
   end
   if mod(n, 2) == 1
      p = 1/(2*n)*p + 1/2*var{1}*var{2}^((n-1)/2);
   else
      p = 1/(2*n)*p + 1/4*(var{1}^2*var{2}^((n-2)/2) ...
```

```
          + var{2}^(n/2));
   end
   cres{1} = p;
   cres{2} = var;


function res = mulperms(p1, p2)
    ct = 0;
    res = [];
    if numel(p1) == 0
       res = p2;
    elseif numel(p2) == 0
       res = p1;
    else
       e = sort(unique([p1{:}, p2{:}]));
       for i=1:e(numel(e))
           r{i} = i;
       end
       for w = 1:numel(e)
           fd2 = 0;
           for h=numel(p2):-1:1
               for i=1:numel(p2{h})
                   if e(w) == p2{h}(i)
                      s = p2{h}(i);
                      if i < numel(p2{h})
                         t = p2{h}(i+1);
                      else
                         t = p2{h}(1);
                      end
                   fd2 = 1;
                   break;
                   end
               end
               if fd2 == 1
                  break;
               end
           end
           if fd2 ~= 1
              fd1 = 0;
              for h=numel(p1):-1:1
                  for i=1:numel(p1{h})
                      if e(w) == p1{h}(i)
                         s = p1{h}(i);
```

```
                    if i < numel(p1{h})
                       t = p1{h}(i+1);
                    else
                       t = p1{h}(1);
                    end
                    fd1 = 1;
                    break;
                end
            end
            if fd1 == 1
               break;
            end
        end
    else
        fd1 = 0;
        for j=numel(p1):-1:1
            for k=1:numel(p1{j})
                if t == p1{j}(k)
                    fd1 = 1;
                    if k < numel(p1{j})
                       t = p1{j}(k+1);
                    else
                       t = p1{j}(1);
                    end
                    break;
                end
            end
            if fd1 == 1
               break;
            end
        end
    end
    r{s} = t;
end
ct = 0;
k = 1;
u = [1:numel(r)];
while k <= numel(r)
    if k ~= r{k} && ismember(k, u)
        p = [];
        j = k;
        if numel(p) == 0;
           p = [p, j, r{j}];
```

```
                j = r{j};
            end
            while r{j} ~= p(1)
                p = [p, r{j}];
                j = r{j};
            end
            k = k + numel(p);
            while p(1) ~= min(p)
                p = circshift(p, 1, 2);
            end
            ct = ct + 1;
            res{ct} = p;
            u = setdiff(u, p);
        else
            u = setdiff(u, [k]);
            k = k + 1;
        end
    end
end
res;


function k = fperm(p, a)
    fd = 0;
    k = a;
    for i=1:numel(p)
        tp = p{i};
        for j = 1:numel(tp)- 1
            if k == tp(j)
               k = tp(j+1);
               fd = 1;
               break;
            end
            if k == tp(numel(tp)) && fd ~= 1
               k = tp(1);
               fd = 1;
            end
        end
        if fd == 1
           break;
        end
    end
```

```
function n = ntable(l, notes)
    for i = 1:numel(notes)
        if l == notes{i}
           n = i;
           break;
        end
    end


function eqnotes = midichords(fn, instr, poly, d)
   if sum(coeffs(poly)) == 352
      group{1} = [];
      for i = 2:12
          group{i} = mulperms({[1,2,3,4,5,6,7,8,9,10,11,12]}, ...
                     group{i-1});
      end
   else
      group{1} = [];
      for i = 2:12
          group{i} = mulperms({[1,2,3,4,5,6,7,8,9,10,11,12]}, ...
                     group{i-1});
      end
      for i = 13:24
          group{i} = mulperms(group{i-12}, {[1,11], [2,10], ...
                     [3,9], [4,8], [5,7]});
      end
   end
   s = group;
   nm = nchoosek(1:12, d);
   snm = size(nm);
   for i = 1:snm(1)
       notes{i} = nm(i,:);
   end
   nct = coeffs(poly);
   nct = nct(d+1);
   ct = 0;
   for j = 1:numel(notes)
       res = [];
       for i = 1:numel(s)
           pt = [];
           for h = 1:numel(notes{j})
               pt = [pt, fperm(s{i}, notes{j}(h))];
           end
```

```
            res = sort(unique([res, ntable(sort(pt), notes)]));
        end
        if j == res(1)
           ct = ct + 1;
           eqnotes{ct} = notes{j} - 1;
        end
        if ct == nct
           break;
        end
    end
    num = 2^32 + 9*nct + 7;
    b = [];
    while num > 0
        s = mod(num, 256);
        num = floor(num/256);
        b = [s, b];
    end
    b = b(2:5);
    o = [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71];
    track = [];
    for j = 1:d
        ntrack = [];
        for i = 1:nct
            ntrack = [ntrack, 0, 144, o(eqnotes{i}(j)+1), 96, ...
                      130, 30, 144, o(eqnotes{i}(j)+1), 0];
        end
         track = [track, 77, 84, 114, 107, b, 0, 192, instr, ...
                  ntrack, 0, 255, 47, 0];
    end
    m = [77, 84, 104, 100, 0, 0, 0, 6, 0, 1, 0, d, 0, 128, track];
    fname = fopen(char(fn), 'w');
    fwrite(fname, double(m));
    fclose(fname);


function s = pstring(p)
    if numel(p{1}) == 0
        s = '[]';
    else
       s = '';
       for i = 1:numel(p)-1
           t =  char(sym(p{i}));
           s = strcat(s, t(9:length(t)-2), ',');
```

```
       end
       t =  char(sym(p{numel(p)}));
       s = strcat('[', s, t(9:length(t)-2),']');
    end


function p = stringprint(str,num)
    fprintf('\n');
    q = floor(length(str)/num);
    for i = 1:q
        fprintf('%s \n', str((i-1)*num+1:i*num));
    end
    if mod(length(str), num) ~= 0
       fprintf('%s \n', str(q*num+1:end));
    end
    fprintf('\n');


function p = notetoperm(n)
   s = n + 1;
   sn = sort(n+1);
   ct = 0;
   i = 0;
   p{1} = [];
   while ct < numel(n)
      c = [sn(1)];
      k = sn(1);
      while s(k) ~= c(1)
          c = [c, s(k)];
          k = s(k);
      end
      ct = ct + numel(c);
      sn = setdiff(sn, c);
      if numel(c) > 1
         i = i + 1;
         p{i} = c - 1;
      end
   end


function t = Tn(n, p)
   for i=1:numel(p)
       ires{i} = p{i} + 1;
   end
```

```
    tn = [];
    for i=1:n
        tn = mulperms({[1,2,3,4,5,6,7,8,9,10,11,12]}, tn);
    end
    ires = mulperms(tn, ires);
    for i=1:numel(ires)
        ores{i} = ires{i} - 1;
    end
    t = ores;


function r = Ret(n, p)
    for i=1:numel(p)
        ires{i} = p{i} + 1;
    end
    rn = [];
    for i=1:n
        rn = mulperms({[1,12], [2,11], [3,10], ...
             [4,9], [5,8], [6,7]}, rn);
    end
    ires = mulperms(ires, rn);
    for i=1:numel(ires)
        ores{i} = ires{i} - 1;
    end
    r = ores;


function i = Inv(n, p0, p)
    for i=1:numel(p)
        ires{i} = p{i} + 1;
    end
    t = [];
    for i=1:2*p0+1
        t = mulperms(t, { [1, 2, 3, 4, 5, 6, ...
            7, 8, 9, 10, 11, 12] });
    end
    for i = 1:n
        ires = mulperms({ [1,12], [2,11], [3,10], ...
               [4,9], [5,8], [6,7] }, ires);
        ires = mulperms(t, ires);
    end
    for i=1:numel(ires)
        ores{i} = ires{i} - 1;
```

```
   end
   i = ores;


function f = midinotes(fn, inotes, instr, b12)
   n = inotes;
   nnum = numel(n);
   o = [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71];
   if b12 == 1
      while mod(nnum, 12) ~= 0
          n = [n, 0];
          nnum = numel(n);
      end
      notes = [];
      for j = 1:nnum/12
          nn = [];
          for i = (j-1)*12+1:j*12
              nn = [nn, 0, 144, o(n(i)+1), 96, ...
              129, 110, 144, o(n(i)+1), 0];
          end
          notes = [notes, nn,  127, 144, o(1),0];
      end
      num = 2^32 + numel(notes) + 7;
      b = [];
      while num > 0
          s = mod(num, 256);
          num = floor(num/256);
          b = [s, b];
      end
      b = b(2:5);
      m = [77, 84, 104, 100, 0, 0, 0, 6, 0, 0, 0, 1, 0, ...
           128, 77, 84, 114, 107, b, 0, 192, instr, notes, ...
           0, 255, 47, 0];
   else
      num = 2^32 + 9*nnum + 7;
      b = [];
      while num > 0
          s = mod(num, 256);
          num = floor(num/256);
          b = [s, b];
      end
      b = b(2:5);
      notes = [];
```

```
    for i = 1:nnum
        notes = [notes, [0, 144, o(n(i)+1), 96, ...
        129, 110, 144, o(n(i)+1), 0]];
    end
    m = [77, 84, 104, 100, 0, 0, 0, 6, 0, 0, 0, 1, 0, ...
          128, 77, 84, 114, 107, b, 0, 192, instr, notes, ...
          0, 255, 47, 0];
end
fname = fopen(fn, 'w');
fwrite(fname, m);
fclose(fname);
f = numel(m);
```