~/summer2024_Research/week_2_ben_tiwai_interpolation/10_unknownT.mpl

```
# solution 1 - try catch?
with(LinearAlgebra):
with(ArrayTools):
# 1. Black box for some polynomial f in Q[x_1,x_2,....x_n] of some degree m
B:=proc(var,point_)
    local u,v,a:
    a:=randpoly(var,degree=5);
    return [seq(eval(a,{seq(var[v]=point_[u][v],v=1..numelems(point_[u]))}),u=1..numelems(point_))]:
end proc:
# 2. Generating a prime for each variable
generate_evaulation_primes:=proc(n)
    local p,m,i:
    m:=1:
    p:=Vector(n,0):
    for i from 1 to n do
        p[i]:=nextprime(m):
        m:=p[i]:
    end do:
return convert(p,list):
end proc:
# 3. Generating a list of list powers of prime.
generate_prime_powers:=proc(T,prime_points,num_var)
    local i,j:
    return [seq([seq(prime_points[j]^i,j = 1..num_var)], i = 0..2*T-1)]:
end proc:

# 4. Getting the number of terms in the polynomial
get_num_terms:=proc(v,T)
    local H,i:
    H:=Matrix([seq(v[i..i+(T-1)],i=1..T)]):
    return H,Rank(H):
end proc:
# 5. Getting the roots of the lambda polynomial
get_rootsOf_lambda_polynomial:=proc(M,v,terms)
    local H,b,X,num_row,Lambda,R,i,r:
    H:=M[1..terms,1..terms]:
    b:=-Vector(v[terms+1..terms+terms]):
    X:=LinearSolve(H,b):
    num_row:=Size(X)[1]:
    Lambda:=Z^num_row:
    for i from 1 to num_row do
        Lambda:=Lambda+X[i]*Z^(i-1):
    end do:
    R:=roots(Lambda):
    return [seq(r[1],r in R)]:
end proc:

# 6.Generating monomials from the roots of the lambda polynomial
generate_monomials:=proc(roots_,num_var,prime_points,vars)
     local ff,l,l2,i,prime_var_map,monomials,j:
     prime_var_map:= table([seq(prime_points[i]=vars[i],i=1..num_var)]):
     print(prime_var_map):
     monomials:=Vector(numelems(roots_),0):
     for j from 1 to numelems(roots_) do
        #   print(roots_[j]):
```

```
            ff:=ifactor(roots_[j]):
          #   print(ff):
            l:=nops(ff):
            for i from 1 to l do
                 l2:=nops(op(i,ff)):
                 if l2=1 then
                      ff:=subs(op(i,ff)=prime_var_map[op(1,op(i,ff))],ff):
                 else
                      ff:=subs(op(1,op(i,ff))=prime_var_map[op(1,(op(1,op(i,ff))))]
,ff):
                 fi:
            end do:
            monomials[j]:=ff:
       end do:
       return convert(monomials,list):
end proc:

# Step 2 of BT interpolation
# 7. Constructing the Vandermonde matrix
Construct_Vandermonde:=proc(terms,Roots_)
    local i,j:
    return Matrix([seq([seq(Roots_[j]^i,j = 1..numelems(Roots_))], i =
0..terms-1)]):
end proc:

# 8. Getting the coefficients of the polynomial
get_coefficients:=proc(terms,Roots_,v)
    local Van,b:
    b:=<v[1..terms]>:
    Van:=Construct_Vandermonde(terms,Roots_):
    return LinearSolve(Van,b):
end proc:
# 9. Constructing the final polynomial
construct_final_polynomial:=proc(coeff_,Monomials)
    local i,f,n:
    f:=0:
    for i from 1 to numelems(coeff_) do
        f:=f+coeff_[i]*Monomials[i]:
    end do:
    return f:
end proc:

num_var:=3:
vars:={x,y,z}:
# f:=randpoly(vars,degree=5):
deg:=5:
# Try T:=1..2^n until we find a T that works(O(log(n)) time complexity)
prime_points:=generate_evaulation_primes(num_var):
# T:=deg-1:
TT:=seq(2^i,i=1..num_var+1);
for T in TT do
    prime_powers:=generate_prime_powers(T,prime_points,num_var);
    y_:=B(vars,prime_powers):
    Y,terms:=get_num_terms(y_,T):
    terms;
end do;


# Roots_:=get_rootsOf_lambda_polynomial(Y,y_,terms):
```

```
# Monomials:=generate_monomials(Roots_,num_var,prime_points,vars):
# coeff_:=get_coefficients(terms,Roots_,y_):
# f1:=construct_final_polynomial(coeff_,Monomials);

```