Et billede, der indeholder tekst

Automatisk genereret beskrivelse

1a: A

Kan det skrives som Nix. Nej, ikke hvor det bliver finit

1b: A

Kan det skrives som

Nej, ikke hvor det bliver finit

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Der mistes altså maksimalt 3 og mindst 2 betydningsfulde cifre i base-10. Det giver i log2 , hvorved at der mindst mistes 8 og maksimalt 9 bits. C

”Loss of precision theorem” –

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

måske p-1? Dvs. B)

Implicit first-digit for floating point. and

er alle tal mellem 1 og 2. Derfor

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

D)

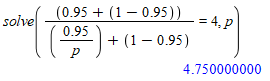
Et billede, der indeholder tekst

Automatisk genereret beskrivelse

b) No.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse



5 kerner – B.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

C)

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

C)

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

B)

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

A). Du afleder, og bruger

Når du skal se, hvordan den opfører sig når det går mod nul kan du bruge lhospitals regel og skide hul i absolutværdierne->

Det ses herved, at der er tale om en kontinuert funktion, da den når den går mod sker der ikke noget ”uventet”. Derfor er det veldefineret, og numerisk well-conditioned.

Et condition-number er aldrig afhængigt af implementering, men udelukkende af funktionen. Så fra start ved du, at det aldrig er C.

Alternativt brug lhospitals:

Den opfører sig finnobambino.

Big cond= bad.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

B) – når x er tæt på nul giver det meget forkert.   
Man kunne fixe det med expm1(x)/x fra <math.h>.

YOu could also do a series expansion of .

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

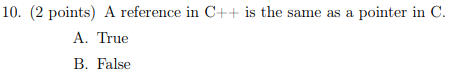
A)  
elementerne i arrayet er gemt fysisk ved siden af hinanden i hukommelsen.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

b) Vi kalder elementet (adressen til) sum igen og igen over tid. Derfor er denne temporally local.

”means that you are using the same location in memory (variable) over and over again over a period of time”. Skal ikke kaldes “på ny” fra hukommelse, kan få lov at blive i hukommelsen.



B.

References can not be changed after initialization in the way a pointer can. Values to references can though.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

B)

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

B. n’th polynomial calls the two earlier polynomials. Thus, it is a nested function (calls itself twice in each call)

Et billede, der indeholder tekst, bord

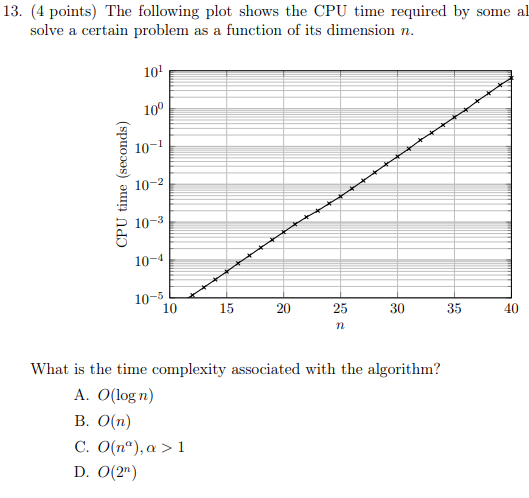
Automatisk genereret beskrivelse

Tegn kalde-træet.

Slutter I en forgrening H(x,2)->(H(x,1) & H(x,0)). Hver gang behøves der to lokale variable altid. Et funktionskald er derfor operationer. Det største antal kald er dybden af træet, og det er maksimalt . Derfor er space-complexity .

Spatially it grows as a tree with depth . Thus, it is . Temporally, have look at notes for Fibonacci .

En gang man går ned af træet fordobler man antallet af funktionskald, altså en potens-sag ->



D)

lineaært I semilog bliver kun eksponentiel vækst hehe.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

B)

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

A)

Dynamic is if you use malloc/calloc/XYZlloc

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Rewriting as when using double-angle identity:

Du kunne også have lavet noget med på en eller anden måde. Tjek det lige ud.

#include <math.h>

#include <stdio.h>

double feval(double x) {

    if(sin(x)==0){

        return 0;

    }

    else{

        return 2\*pow(sin(x/2),2)/sin(x);

    }

}

int main(void){

    double x = 0.0;

    double outers = feval(x);

    printf("%f",outers);

    return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

If cancellation baby

Rewrite as

And this would take care of cancellation. If you want to prevent overflow you should simplify it when is BIGGIBIGGI. Then denominater would be , for as distance to closest number in this range is appx 2.

#include <math.h>

#include <stdio.h>

double geval(double x) {

    return cbrt(pow(x,2)+2)-cbrt(pow(x,2));

}

You need to do more. When it will give cancellation.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

For summation of same sign:   
You need to make check if all elements .

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

double weighted\_harmonic\_mean(int n, double \* x, double \* w) {

    if(n<=0||!(x)||!(w)){

        return NAN;

    }

    else{

        double seqsumW,seqsumX;

        seqsumW = 0.0;

        seqsumX = 0.0;

        for(int i=0;i<n;i++){

            if(x[i]==0||w[i]==0){

                return NAN;

            }

            seqsumW += w[i];

            seqsumX += w[i]/x[i];

        }

        if(seqsumX==0||isnan(seqsumX)||isnan(seqsumW)){

            return NAN;

        }

        else{return seqsumW/seqsumX;}

    }

}

int main(){

    int n = 4;

    double \* x, \* w,x\_arr[4]={1,2,3,100},w\_arr[4]={2,1,4,12},outers;

    x = x\_arr;

    w = w\_arr;

    outers = weighted\_harmonic\_mean(n,x,w);

    printf("%f",outers);

    return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

int darmv(int n, double \* a, double \* b, double \* x) {

    double \*x\_holder = (double\*)malloc(n\*sizeof(double));

    if(x\_holder==NULL||n<=0||b==NULL||x==NULL||a==NULL){

        return -1;

    }

    for(int i=0;i<n-1;i++){

        x\_holder[i] = x[i]\*a[i];

        if(isnan(x\_holder[i])||isinf(x\_holder[i])){

            return -1;

        }

    }

    double seqsum = 0.0;

    for(int i=0;i<=n-1;i++){

        seqsum += b[i]\*x[i];

        if(i==n-1){

            seqsum += a[i]\*x[i];

            x[i] = seqsum;

        }

    }

    for(int i=0;i<n-1;i++){

        x[i] = x\_holder[i];

    }

    free(x\_holder);

    printf("\nfinal vector x:\n");

    for(int i=0;i<n;i++){

        printf("%f\n",x[i]);

    }

    return 0;

}

int main(){

    double a\_val[4] = {1,2.7,16.8,11},\*a,b\_val[3] = {2400,12,67.6},\*b,x\_val[4] = {0.01,300,30,17},\*x;

    int n = 4;

    a = a\_val;

    b = b\_val;

    x = x\_val;

    int returner = darmv(n,a,b,x);

    return returner;

}

Use that

Second part:

darsv

You need to make sure determinant -> .

Using forward subs -> diagonal elements.

What you want to do

Last element:

Last coefficient.   
Then you have solved system .

You should check if diagonal contains s.

Pin code is five zeroes.

implementation 1

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

How many flops?

You have floating point additions, operations, and additions in the end if row major.

This gives: (if row major). In case , you have ops, and if you time this:

And if you want GFLOPS:

implementation 2

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Then you only have

If you go in else loop you have .

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Double precision har basen . Herved skal vi undersøge, om der findes en mulig repræsentation som opfylder:

Hvor og er heltallige. Det er klart, at hvis vi bruger , og , så får vi .

Derfor må det være sandt.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Hvis man evalfer så får man at er MEGET tæt på at være 1. Men det er *ikke* 1. Derfor er det ikke et heltal, og tallet kan ikke repræsenteres.

Kig på hele eksponent-grafen.

Et billede, der indeholder tekst

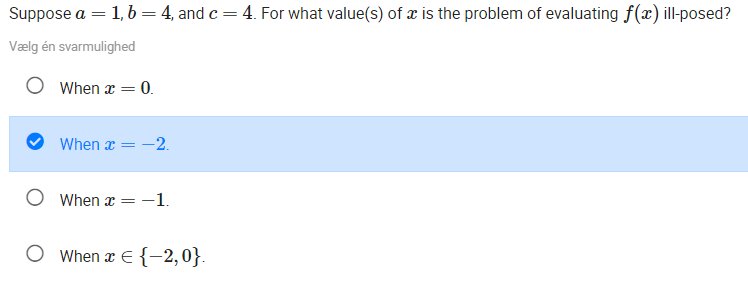
Automatisk genereret beskrivelse



Sååååå vi afleder

Et billede, der indeholder tekst

Automatisk genereret beskrivelse



A problem is ill-posed only when the condition-number is . If it is just large, it is ill-conditioned.

If we need conditionnumber to be inf, we would have to divide by zero. Substituting into the earlier found condition-number, and checking when denominator is zero shows that this is in -2.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

I would guess.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

What can you say about the relative condition number when abs(**x)=0?**

Fcond Rel is close to 1.

The naïve implementation would be prone to underflow I believe. One should use e.g. log1p.

Cancellation only occurs for subtraction. The condition-number of log(x) is when . One could use l’hospital and show that condition-number goes to zero. But this seems odd, as log(x) is not the function we are looking at.

Underflow most possible.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

One call calls itself twice. Multiple recursion.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Drawing tree, we see that time must be . Space must be , as maximal layer depth is k, and number of stored variables is at max 2k

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Vi ved ikke rigtigt hvad y er. Men i hver løkke holdes y[i] i hvert fald og lægges til, så der er en grad af temporal lokalitet (da den er i inder-løkken og derved holdes).



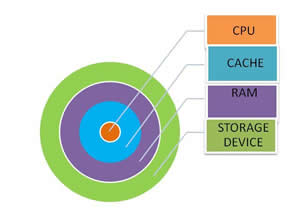
x iteres igennem af inder-løkken, og derfor kaldes den samme hukommelsesadresse aldrig flere gange.

Da det er et array ligger adresserne dog ved siden af hinanden igennem løkken, og der er spatiel lokalitet.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Cache er det hurtigste vi har. Main memory er RAM, og den er langsommere.



Et billede, der indeholder tekst

Automatisk genereret beskrivelse

(vi kan parallelisere 60 af sekunderne, de resterende 20 kan vi ikke røre ved).

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

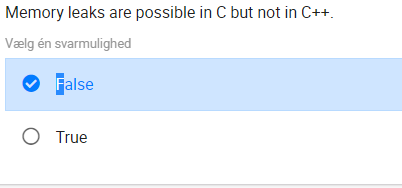


Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Nikolines pointe:

det svarer til at man skriver double \*arr; arr = malloc(xxx) på to linjer. Altså automatisk allokering for selve pointeren og dynamisk allokering af hvad arr pointer til.



Also c++

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

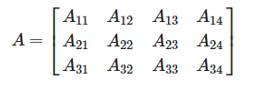
Det sendes som array gemt i row-major. Derfor (3,-1.0,pa+5,1)

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Forkert.

RowMajor array:

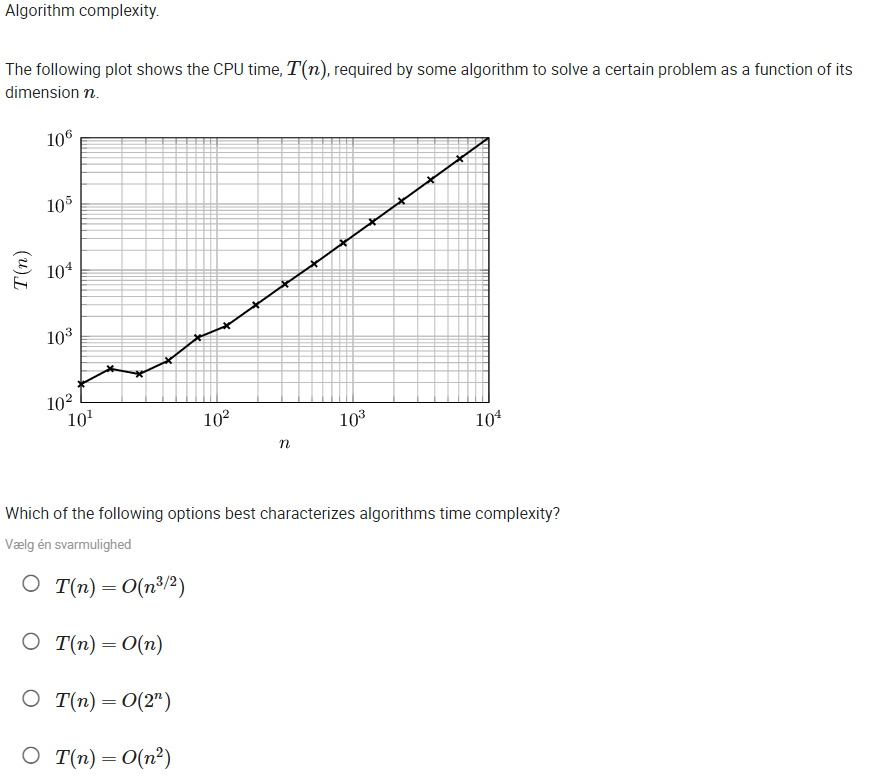


Et billede, der indeholder tekst

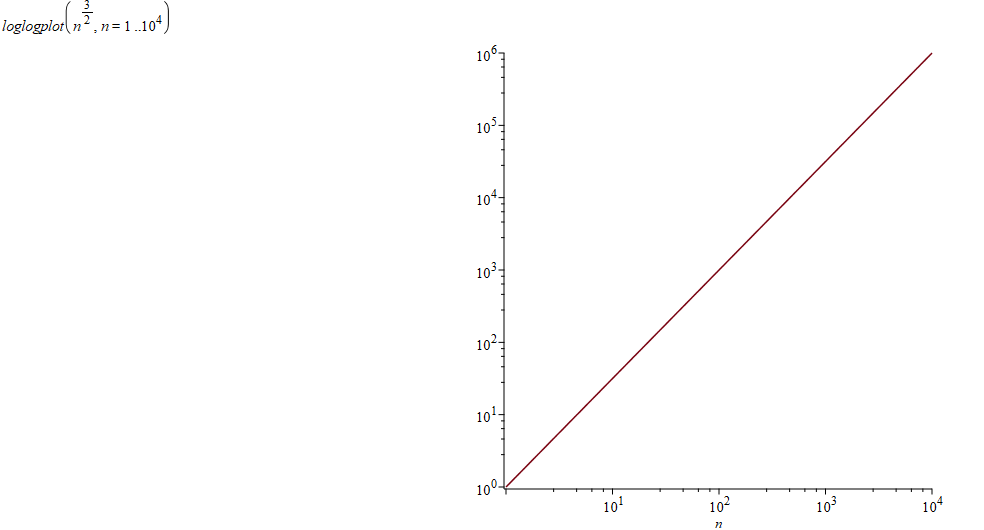
Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse



Flere er lineære I loglogland. Men akserne passer kun med



Et billede, der indeholder tekst

Automatisk genereret beskrivelse

**OBS. triplet form = COO (**Coordinate representation)

#include "sparse\_triplet.h"

#include <stdlib.h>

#include <stdio.h>

void sparse\_triplet\_mv\_trans(

    const struct sparse\_triplet \*A,

    const double \*x,

    double \*y) {

        //module 1 for transposing COO cordinate format

        size\_t \*rowIdx,\*colIdx,m,n;

        rowIdx = A->J;

        colIdx = A->I;

        n = A->n;

        m = A->m;

        //transposing finished

        //multiplication module

        double \* y\_val = (double\*)malloc(n\*sizeof(double));

        for(size\_t i = 0;i<A->nnz;i++){

                y[rowIdx[i]] += A->V[i]\*x[colIdx[i]];

            }

        //printing module

        for(size\_t i=0;i<A->n;i++){

            printf("%f, ",y[i]);

        }

    }

//for debugging

int main(){

    double A[3]={2,3,5};

    size\_t nnz = 3;

    struct sparse\_triplet \*arr = (struct sparse\_triplet \*)malloc(nnz\*sizeof(struct sparse\_triplet));

    arr->V = A;

    size\_t m = 2,n=3;

    size\_t rower[3] = {0,1,1};

    size\_t columna[3] = {1,0,2};

    arr->m = m;

    arr->n = n;

    arr->nnz = nnz;

    arr->I = rower;

    arr->J = columna;

    double x\_val[2] = {1,2};

    double y\_val[3] = {9,3,3};

    double \*x, \*y;

    x = x\_val;

    y = y\_val;

    sparse\_triplet\_mv\_trans(arr,x,y);

    return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

rewrite by multiplying by 1

Using to avoid underflow

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

double poisson\_pmf(unsigned long k, double lambda) {

  if(lambda<=0 || isnan(lambda)){

    return NAN;

  }

  else{

    return exp(k\*log(lambda)-lgamma(k+1)-lambda);

  }

}

int main(){

  double lambda = 2.0;

  unsigned long k = 2;

  double out = poisson\_pmf(k,lambda);

  printf("%f",out);

  return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

En centering-matrix har den funktion, at den trækker gennemsnittet af komponenterne af vektoren fra hvert element i vektoren aka

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

int dcemv(int n, double \* x) {

  double mean = 0.0;

  for(int i=0;i<n;i++){

    mean += x[i]; //lægger alle vektor-værdier sammen

  }

  mean /= n; //finder gennemsnittet af dem

  for(int i=0;i<n;i++){

    x[i]-=mean; //trækker gennemsnittet fra hvert element

    printf("%f, ",x[i]);

  }

  return 0;

}

int main(){

    double A[3]={2,3,5};

    double \*arr;

    arr = A;

    int n = 3; //nxn matrix boi

    int call = dcemv(n,arr);

    return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

void rotg(double a, double b, double \* c, double \* s){

  /\* Insert code here \*/

  double r = hypot(a,b);

  double cc = a/r;

  double cs = -b/r;

  c = &cc;

  printf("\n%f",\*c);

  s = &cs;

  printf("\n%f",\*s);

}

int main(){

  double a = 2.0;

  double b = 3.0;

  double \* c;

  double \* s;

  rotg(a,b,c,s);

  return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, skærmbillede, person

Automatisk genereret beskrivelse

deriving gives . This is prone to underflow if very small, but when it is very small it approaches zero.

You can rewrite as:

En nok bedre løsning ville være at

math.h special-funktioner

log1p(x) =

lgamma(n+1) = log(n!)

gamma(n+1)=n!

expm1(x) =

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

int dhhmv(int n, const double \*u, double \*x){

    double \*uut = (double\*)malloc(n\*n\*sizeof(double));

    double \*J = (double\*)malloc(n\*n\*sizeof(double));

    if(uut==NULL || J==NULL || n<2 || u==NULL || x==NULL){

        return -1; //sadly, an initialized pointer without value is undefined and cant be checked for.. It could evaluate to NULL, but often wont.

    }

    J[0] = 1.0;

    double den = u[0]\*u[0];

    for(int i=0;i<n;i++){

        for(int j=0;j<n;j++){

            uut[j+i\*n] = u[i]\*u[j];

            if(j==i && j>0){

                den -= uut[j+i\*n];

                J[j+i\*n] = -1;

            }

            else if (j!=0 && i!=0 && i!=j){ //is enough with just j!=0

                J[j+i\*n] = 0.0;

            }

        }

    }

    den = 2/den;

    for(int i = 0;i<n\*n;i++){

        J[i] -= den\*uut[i];

    }

    for(int i=0;i<n;i++){

        double seqsum = 0.0;

        for(int j=0;j<n;j++){

            seqsum += x[j]\*J[j+i\*n];

        }

        J[i] = seqsum; //NEED TO HOLD VALS SOMEWHERE, MIGHT ASWELL OVERWRITE FIRST ROW

    }

    for(int i=0;i<n;i++){

        x[i] = J[i];

    }

    printf("Multiplication product Tx is: ");

    for(int i=0;i<n;i++){

        printf("%f, ",x[i]);

    }

    free(uut);

    free(J);

    return 0;

}

int main(){

    double u\_val[2] = {2,3};

    double \*u;

    u = u\_val;

    double x\_val[2] = {2,4};

    double \*x;

    x = x\_val;

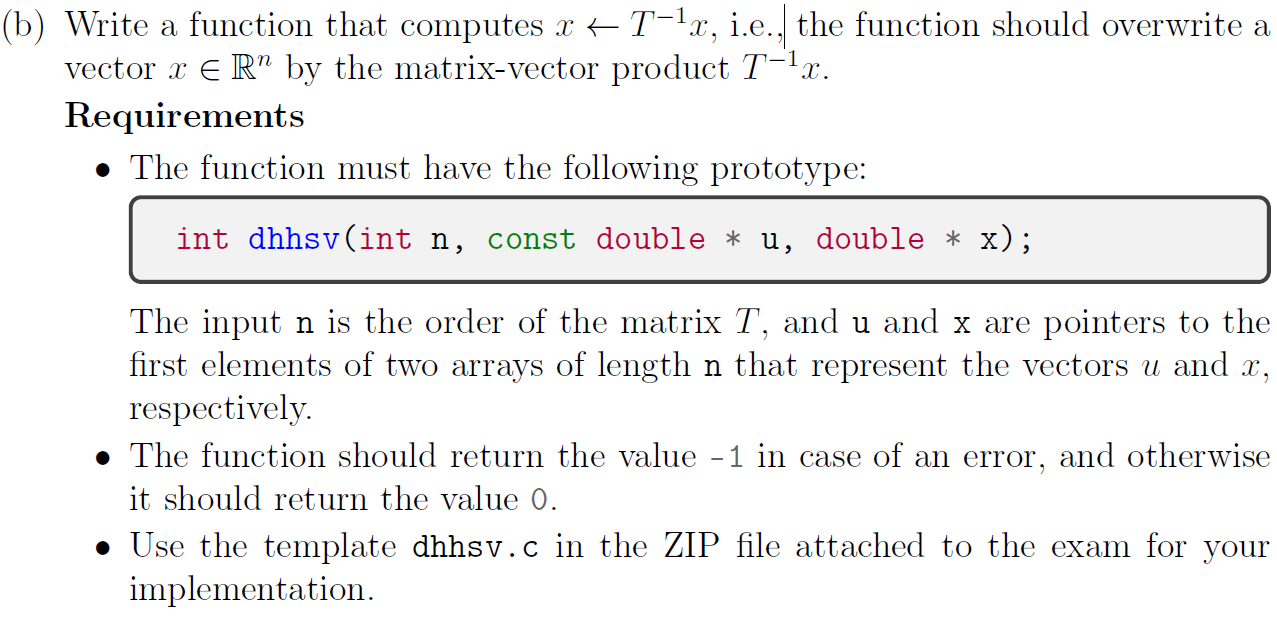
    int n = 2;

    int outers = dhhmv(n,u,x);

    printf("\n Exit code: %d\n",outers);

    return 0;

}



Much the same implementation. Difference is in how the uut matrix is stored. In this assignment, J\*u\*uT\*J (which is tæller), you see that J\*u\*uT\*J = u\*uT (but with MINUS in first row and column except element 0). Thus, the only thing you must implement is the change in sign, and the rest is the same.

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

int dhhmv(int n, const double \*u, double \*x){

    double \*uut = (double\*)malloc(n\*n\*sizeof(double));

    double \*J = (double\*)malloc(n\*n\*sizeof(double));

    if(uut==NULL || J==NULL || n<2 || u==NULL || x==NULL){

        return -1; //sadly, an initialized pointer without value is undefined and cant be checked for.. It could evaluate to NULL, but often wont.

    }

    J[0] = 1.0;

    double den = u[0]\*u[0];

    for(int i=0;i<n;i++){

        for(int j=0;j<n;j++){

            if((i==0 && i!=j)||(j==0&&j!=i)){

                uut[j+i\*n] = -u[i]\*u[j];

            }

            else{

                uut[j+i\*n] = u[i]\*u[j];

            }

            if(j==i && j>0){

                den -= uut[j+i\*n];

                J[j+i\*n] = -1;

            }

            else if (j!=0 && i!=0 && i!=j){ //is enough with just j!=0

                J[j+i\*n] = 0.0;

            }

        }

    }

    if (den==0){return -1;}

    den = 2/den;

    for(int i = 0;i<n\*n;i++){

        J[i] -= den\*uut[i];

    }

    for(int i=0;i<n;i++){

        double seqsum = 0.0;

        for(int j=0;j<n;j++){

            seqsum += x[j]\*J[j+i\*n];

        }

        J[i] = seqsum; //NEED TO HOLD VALS SOMEWHERE, MIGHT ASWELL OVERWRITE FIRST ROW

    }

    for(int i=0;i<n;i++){

        x[i] = J[i];

    }

    printf("Multiplication product Tx is: ");

    for(int i=0;i<n;i++){

        printf("%f, ",x[i]);

    }

    free(uut);

    free(J);

    return 0;

}

int main(){

    double u\_val[4] = {1,3,4,3};

    double \*u;

    u = u\_val;

    double x\_val[4] = {2,4,1,3};

    double \*x;

    x = x\_val;

    int n = 4;

    int outers = dhhmv(n,u,x);

    printf("\n Exit code: %d\n",outers);

    return 0;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

**Floating point operations er kommutative. De er dog ikke nødvendigvis associative (hvilket er hvad b egentligt spørger op. distributiv er noget andet).**

**Altså   
a true  
b false**

**Et billede, der indeholder tekst

Automatisk genereret beskrivelse**

**A) Binary 64 has the smallest machine epsilon. As you see, it has a way bigger precision and smaller , thus giving smaller errors.**

You can calculate machine-epsilon:

B64:

vs

D64:

In human numbers:

’

Et billede, der indeholder tekst

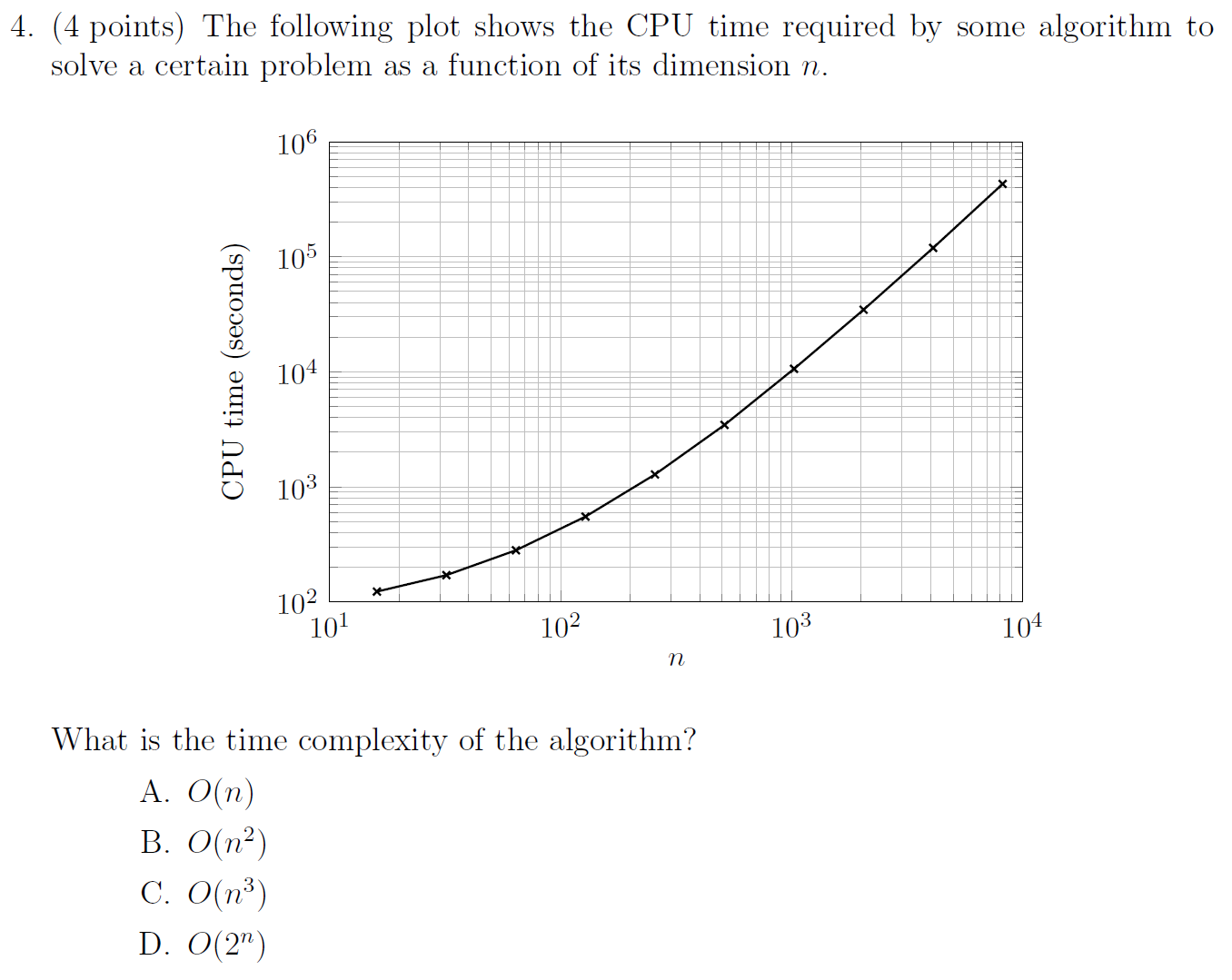
Automatisk genereret beskrivelse

I would say D.

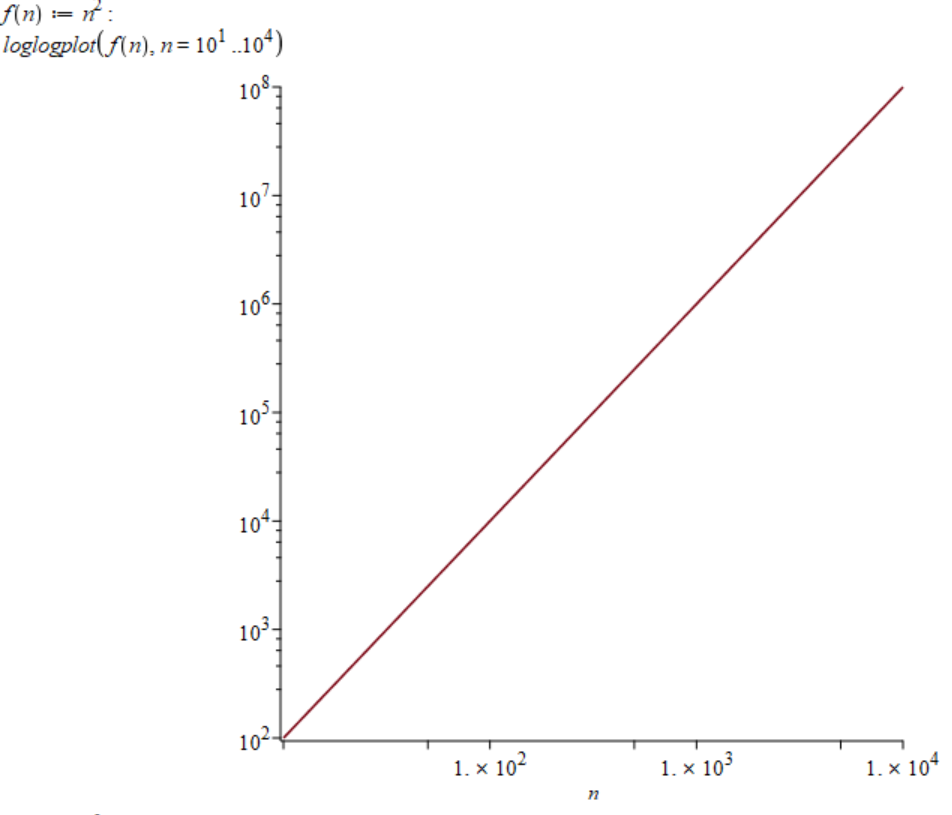
Et billede, der indeholder tekst

Automatisk genereret beskrivelse

When x is close to 0, you are subtracting two numbers which are *very* close, and thus you get catastrophic cancellation. Condition number has nothing to do with LOS.



Du kan se at den starter med at være hurtig, men efter lidt tid tager kompleksiteten så til og bliver upper bounded af . ville give en for voldsom increase i CPU-time.



Et billede, der indeholder tekst

Automatisk genereret beskrivelse

C)   
Det må være en pointer, da arrayet er en pointer til en pointer. Værdien af pointeren er herved double \*.

D)

Du vil have i’te element ud. Derfor \*(pA+i) giver dig en pointer til det i’te element. Til denne pointer lægges så j.   
Jeg ville gå med D, resten giver ikke en double men en pointer.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

a)   
An object is an instance of a class or class template. This must be the case, D.

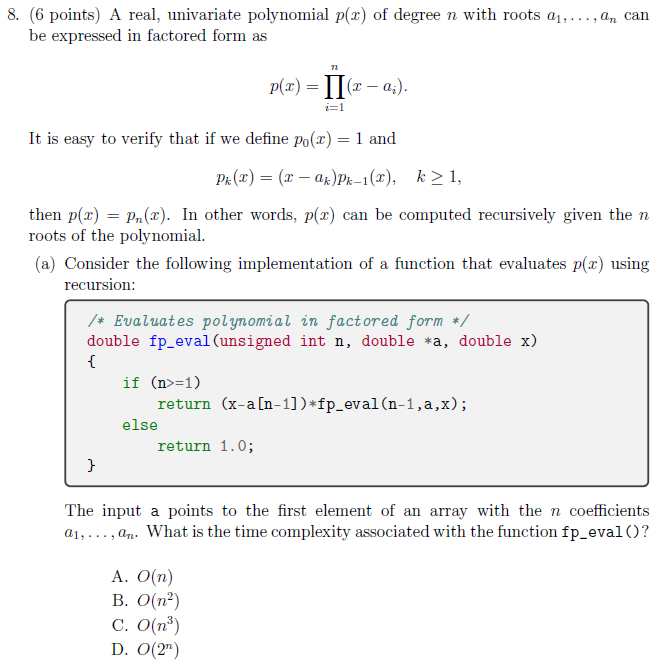
b) This is a class template, as type is defined in the vector call.

c) C it is a method (function of a class template).

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

A)   
Du fylder ikke v med noget som helst. Derfor vil det oftest være NULL, og du returner NULL. Det er i og for sig også fint nok, men her vist ikke intentionen, da det skal fungere som et fejlkald. Når det er et fejlkald, så skal der være free(arr.v).



If i draw the tree it is linear, as it is single recursion. Thus, .

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Also .

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Drawing a tree shows it now has two function calls in every ’th call, and these calls occur n times. Thus the time is going to be .

The space complexity is still going to be the maximal layer depth, which is going to be maximally . At any time we maximally have two variables saved on the stack, and this happens n times.  
Thus, still .

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

C   
1.88

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

You want 3 elements, . You want scalar -1.0. You want to start in ie pointer must be incremented +4. You have stride 3 (stride ), as it is colmaj.

Thus, B).

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

C

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

HVORFOR

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Hver operation I float er bounded af en fejl;

Worst case:

For subtraktionen er det ligeledes:

Hvilket kan skrives som

The error is given as:

Indsæt vores værdier:

Det må altså gælde at:

Den samlede fejl er nu den kumulative fejlsum:

Størrelsen af determinanten er ca. 4. Derfor er fejlen i determinant-beregningen .  
I hver multiplikation indføres der en fejl på

En bedre måde ville nok være:

Det gøres to gange, hvoraf vi fra multiplikationerne får fejl. Subtraktionen giver i sig selv u fejl.

Heraf propagerer den samlede fejl 3u.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Og herved er unit-round-off:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

double logfactorial(int n){

    //double res = log(faculty(n)); //bad idea as factorial overflows to quickly

    double res = lgamma(n+1);

    return res;

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

#include <stdio.h>

double linear\_max(const double x, const double \*a, const double \*b, unsigned int n){

    if(!x||!b||n<=0){return -1;}

    double max = a[0]\*x+b[0];

    double feval = max;

    for(int i=1;i<n;i++){

        feval = a[i]\*x+b[i];

        // if(feval>max){ //non ternary form

        //     max = feval;

        // }

        max = (feval>max)?feval:max; //ternary form

    }

    return max;

}

int main(){

    double x = 2.0;

    double a\_arr[4] = {1000,2,3,4};

    double \* a = a\_arr;

    double b\_arr[4] = {17,700,40000,1000};

    double \*b = b\_arr;

    unsigned int n = 4;

    double out = linear\_max(x,a,b,n);

    printf("%f",out);

}

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Det kan gøres på flere måder.   
mat\_arr[1]->m = 4, men det er ikke en valgmulighed. Samme funktion at gøre (mat\_arr+1->m) = 4 (D)