```
> restart:
```

Musterlösung Aufgabe 5

Rekursive Programmierung

a) Definieren Sie eine Funktion $A(n)$ zur Konstruktion quadratischer Matrizen der Dimension n mit der Eigenschaft: $A_{i,i} = i, i = 1..n$ (Hauptdiagonalelemente), die untere und obere Nebendiagonale sei konstant 1 und der Rest der Matrix gleich 0.

Hinweis: Verwenden Sie das Kommando Matrix und erzeugen Sie die Matrixelemente mit Hilfe einer anonymen Funktion.

```
> A:= (n::nonnegint) -> Matrix(n,n,(i,j) -> if abs(i-j)=1 then 1
  else if i=j then i else 0 end if end if);
```

$$A := n::\mathbb{Z}^{(0,+)} \mapsto Matrix(n, n, (i,j) \mapsto \textbf{if } |i-j| = 1 \textbf{ then } 1 \textbf{ else if } i = j \textbf{ then } i \textbf{ else } 0 \textbf{ end if end if}) \quad (1)$$

```
> A(5);
```

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 3 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 & 5 \end{bmatrix} \quad (2)$$

b) Die Laplace-Entwicklung (Pierre-Simon Laplace) der Determinante von $A(n)$ bzgl. der letzten Reihe zeigt, dass $det\,A(n)$ durch folgende Rekursion berechnet werden kann:

$$det\,A(n) = n\,det\,A(n-1) - det\,A(n-2)$$

Schreiben Sie mit Hilfe dieser Rekursion eine Prozedur zur Berechnung von $det\,A(n)$.

```
> dt1:=proc(n::nonnegint)
    if n <= 2 then
       1
    else
       n*dt1(n-1) - dt1(n-2)
    end if
  end proc;
```

$$dt1 := \textbf{proc}(n::nonnegint)\ \textbf{if } n <= 2 \textbf{ then } 1 \textbf{ else } n * dt1(n-1) - dt1(n-2) \textbf{ end if end proc} \quad (3)$$

```
> seq(dt1(n),n=1..10);
  LinearAlgebra:-Determinant(A(10));    # Probe.
```

$$1, 1, 2, 7, 33, 191, 1304, 10241, 90865, 898409$$

898409 **(4)**

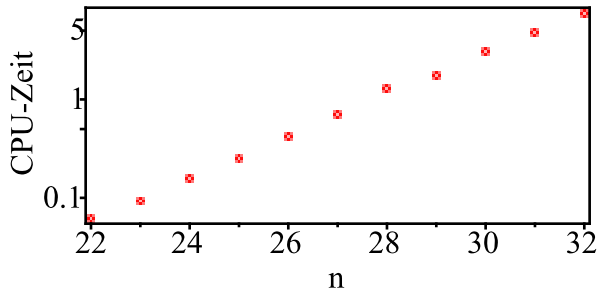
c) Berechnen Sie die Determinanten $det\,A(n), n = 1..32$ und messen Sie die benötigten CPU-Zeiten. Plotten Sie die Zeiten für $n = 22..32$ in ein semi-logarithmisches Diagramm mit logarithmischer Skalierung in y-Richtung.

Hinweis: Nutzen Sie time für die Zeitmessung, seq für die Erzeugung der Liste mit den Zeitwerten und logplot aus dem Paket plots für die Darstellung der Datenpunkte (`style=point`).

```
> times:=[seq(time(dt1(n)), n=1..32)];
```

$$times := [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.015, 0., 0., 0.015, 0.015, 0.015, 0.046, 0.062, 0.093, 0.156, 0.250, 0.421, 0.703, 1.281, 1.750, 3.046, 4.812, 7.546] \quad \textbf{(5)}$$

```
> with(plots):
> d:=logplot([seq([n,times[n]],n=22..32)],style=point,symbol=
  circle,axes=boxed,labels=["n","CPU-Zeit"],labeldirections=
  [horizontal,vertical]):
> d;
```



d)
i) Berechnen Sie mit der Funktion LeastSquares aus dem CurveFitting-Paket die Ausgleichsgerade für die logarithmierten Zeitwerte von $n = 22..32$ und plotten Sie die Datenpunkte und die Ausgleichsgerade in ein Schaubild.

Hinweis: Kapitel 1.8 des Skripts zeigt ein Beispiel zur Anwendung der Funktion `Leastsquares`.

```
> with(CurveFitting):
> Xvalues:=[seq(n,n=22..32)];
```

$$Xvalues := [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32] \quad \textbf{(6)}$$

```
> Yvalues:=[seq(log(times[n]),n=22..32)];
```

$$Yvalues := [-2.780620894, -2.375155786, -1.857899272, -1.386294361, -0.8651224453, -0.3523983872, 0.2476410229, 0.5596157879, 1.113829254, 1.571112798, 2.021017622] \quad \textbf{(7)}$$

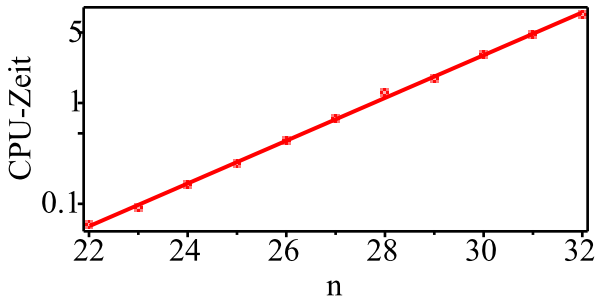
```
> LeastSquares(Xvalues,Yvalues,n);
```

$$-13.5572247784273 + 0.488300329636364\,n \quad \textbf{(8)}$$

```
> Zeit:=unapply(exp((8)),n);
```

$$Zeit := n \mapsto e^{-13.5572247784272903 + 0.488300329636364328 \cdot n} \quad (9)$$

```
> g:=logplot(Zeit(n),n=22..32,labels=["n","CPU-Zeit"],
  labeldirections=[horizontal,vertical]):
> display({d,g});
```



ii) Lösen Sie die Aufgabe i) mit der Maplet-Oberfläche Interactive aus dem Paket CurveFitting.

```
> Interactive([seq([n,log(times[n])],n=22..32)],variable=n);
```

$$-13.5572247784273 + 0.488300329636366\, n \quad (10)$$

e) Bestimmen Sie mit Hilfe der Ausgleichsgeraden die CPU-Zeit, die Ihr Computer für die Berechnung von $det\, A(100)$ benötigen würde.

Hinweis: Ein Jahr hat etwa *365.25*24*3600* Sekunden.

```
> Zeit(100);
```

$$2.08346378578882 \times 10^{15} \quad (11)$$

```
> (11)/(365.25*24*3600)*Jahre;
```

$$6.60209834014252 \times 10^{7}\ Jahre \quad (12)$$

```
> evalf( (12) / 10^6, 2) * Millionen_Jahre / Jahre;
```

$$66.\ Millionen\_Jahre \quad (13)$$

f) Erweitern Sie die Prozedur unter b) mit der *option remember* und messen Sie wiederum die CPU-Zeit für die Berechnung von $det\, A(100)$.

```
> dt2:=proc(n::nonnegint)
     option remember;
     if n <= 2 then
        1
     else
        n*dt2(n-1) - dt2(n-2)
     end if
  end proc;
```

$$dt2 := \mathbf{proc}(n::nonnegint) \quad (14)$$

```
    option remember;
    if n <= 2 then 1 else n * dt2(n - 1) - dt2(n - 2) end if
end proc
```

```
> forget(dt2);
```

```
> st:=time();
```

$$st := 67.406 \quad (15)$$

```
> dt2(1000);
```

$$9018093307789586347750663148143511827086519211581460916777428906664330304816214\ (16)$$
$$8393094283458603062703733007891856894874991430105691786244947864305430660878\$$
$$8490897019031960971950645865370671688431345141630865693683366852679915735958\$$
$$2355557813122189487293677830242573907534927917650545545588766428844596926199\$$
$$9997331809943173113595905591603779555034311914741445025805326605021771042958\$$
$$4376867085368228467880486355210869692411627660372720902169518223665021320350\$$
$$5007666786873534226618767410668083818285001499623289680267416335049658727971\$$
$$4488718162329914735168190249461141738149556003545207541363782633860604448102\$$
$$5686337015050911880423397639589409121217763439906219869222809648083477236391\$$
$$1508510551227695390521979863153180202490459806583299166356812280087931373072\$$
$$4336817462997067507190333808099552393900984169969762139494900421281411650557\$$
$$2862036509559210844686431292534577111189209316672076494698319793058409675233\$$
$$0189899165864152999197415639248919465004484464871991129276426991544726410053\$$
$$1715897971083845744863585820758337872460825959715063974226070028071701461101\$$
$$3577836385470455769144153252151094003490684360603999324661112580644149260597\$$
$$2742348288674844659176947935119323172120086340097959428387672912535031571667\$$
$$2198665615329632219574735386655598935042941003248644248432195815201272945870\$$
$$1757855141474265191053195924268649541351976199027278017749068084205848096398\$$
$$3329449971718235643976242201581326300903722356464740390846280124223908731479\$$
$$9213783541768183196303043162286847590241158243195897935182259458194007801265\$$
$$4589262085332890863863093947741372464300992068424358063882166248114829297754\$$
$$5667055640891387431289994850740647165944219010313893686248114292525403777054\$$
$$2558438620249873930830890924460803480479230699128035190628529624504476104485\$$
$$4802306073638585559932425942290092548923116415011462947197965632903715536172\$$
$$8849366269258258397955955683322185606947869993582454010850882976628226211481\$$
$$3448070880298878332128122866648496047897207447500938003907769643364173949804\$$

```
617143022270850545809227431795409111836263832903804370930347448871513884240\
706023105841671996856640639414065954186409767006504639508228513159191279306 3\
690549667002038013875755344835773890796535725178198831939796506178418347569 7\
806434508297817996381573353055016005651440008080335877614313641135311687753 2\
814093693118628492488880893814372330273116929898708260609007066574556926457 4\
424772081152636784405482783023475644823662013839763343256806852429190407827 1\
168950416407764146988060168736090049417887093256116045339396607856281932629 1\
280339432969658808136046603758930873304414396119924000 01
```

```
> time()-st;
```

0.015 **(17)**