POGLAVJE I

1. Osnovni koraki v *Mathematici*

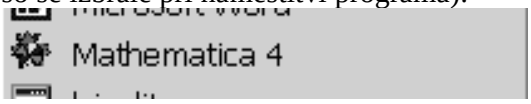
1. Predstavitev programa Mathematica
2. Aktiviranje programa
3. Spoznavanje osnovnih ukazov programa Mathematica
4. Definiranje uporabniških funkcij
5. Shranjevanje datotek
6. Uporaba obstoječih datotek

1. Predstavitev programa Mathematica

Programski paket *Mathematica* predstavlja izredno močno orodje za reševanje široke palete inženirskih problemov, saj uporabniku nudi obsežno množico uporabnih možnosti za analitično, simbolično in numerično analizo matematičnih problemov, hkrati pa je na razpolago tudi grafična predstavitev rezultatov analize. Pojavlja se na različnih platformah (PC, Mac, Unix) in čeprav je v tem delu uporabljana predvsem verzija *Mathematica* 4 v okolju Windows, je delo uporabno tudi za ostale verzije (praktično od MS DOS dalje vse do aktualne verzije 5).

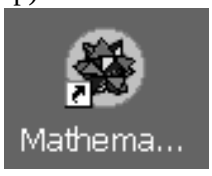
2. Aktiviranje programa

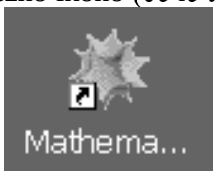
Program se aktivira npr. v meniju *Start* s klikom na ustrezno ikono (odvisno od verzije in nastavitev, ki so se izbrale pri namestitvi programa):



ali


ali v namizju (desktop) s klikom na ustrezno ikono (če le ta obstaja):

ali


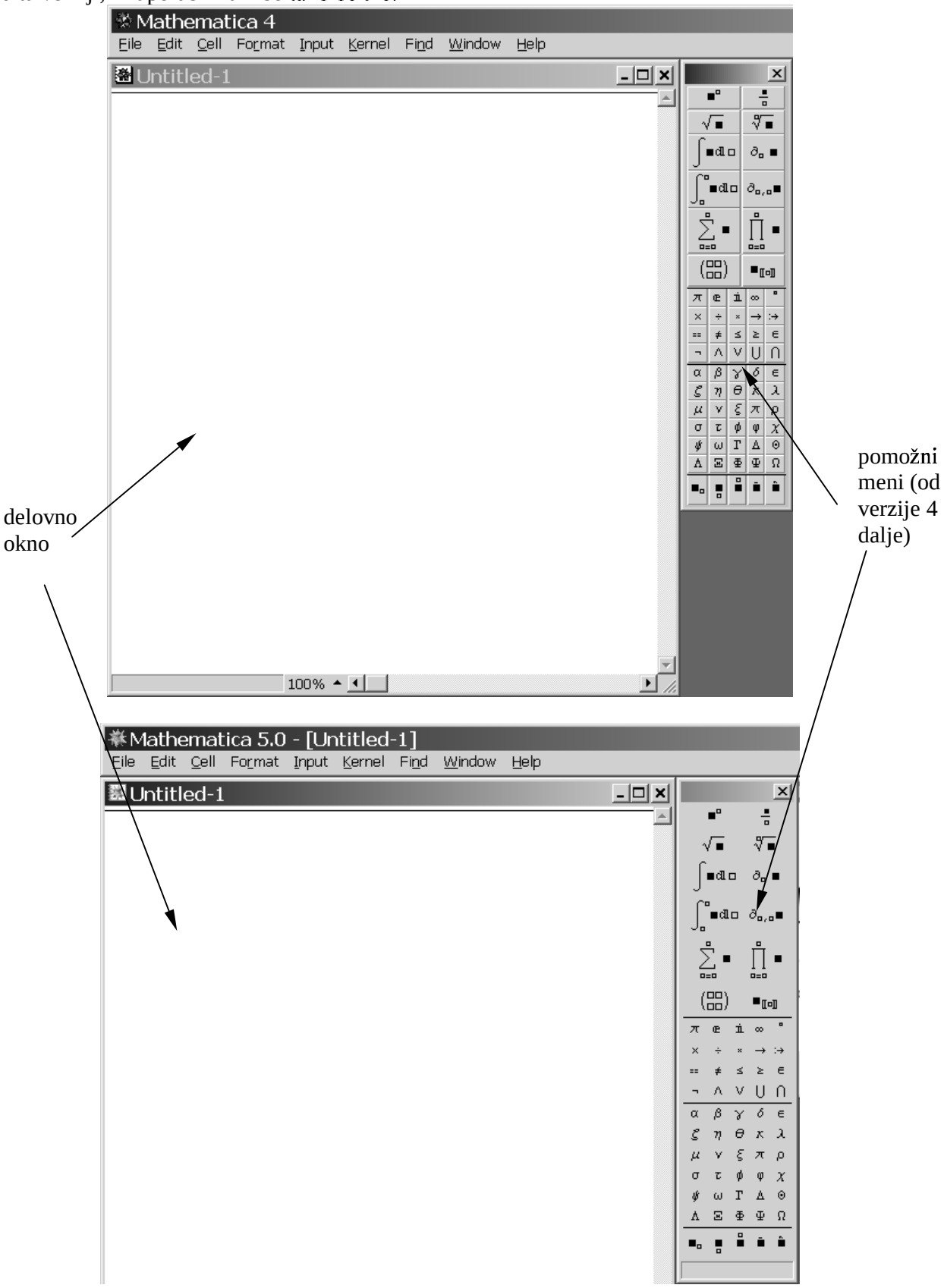
ali


Po aktiviranju programa se pojavi delovno okno. Čeprav je osnovna ideja podajanja vhodnih podatkov v vseh verzijah vedno enaka, je pri izgledu delovnega okna prišlo skozi zadnje verzije (in predvsem s prehodom na platformo Windows) do velikega napredka. V DOS verziji (in ostalih negrafičnih verzijah) je namreč začetno okno izgledalo približno takole:

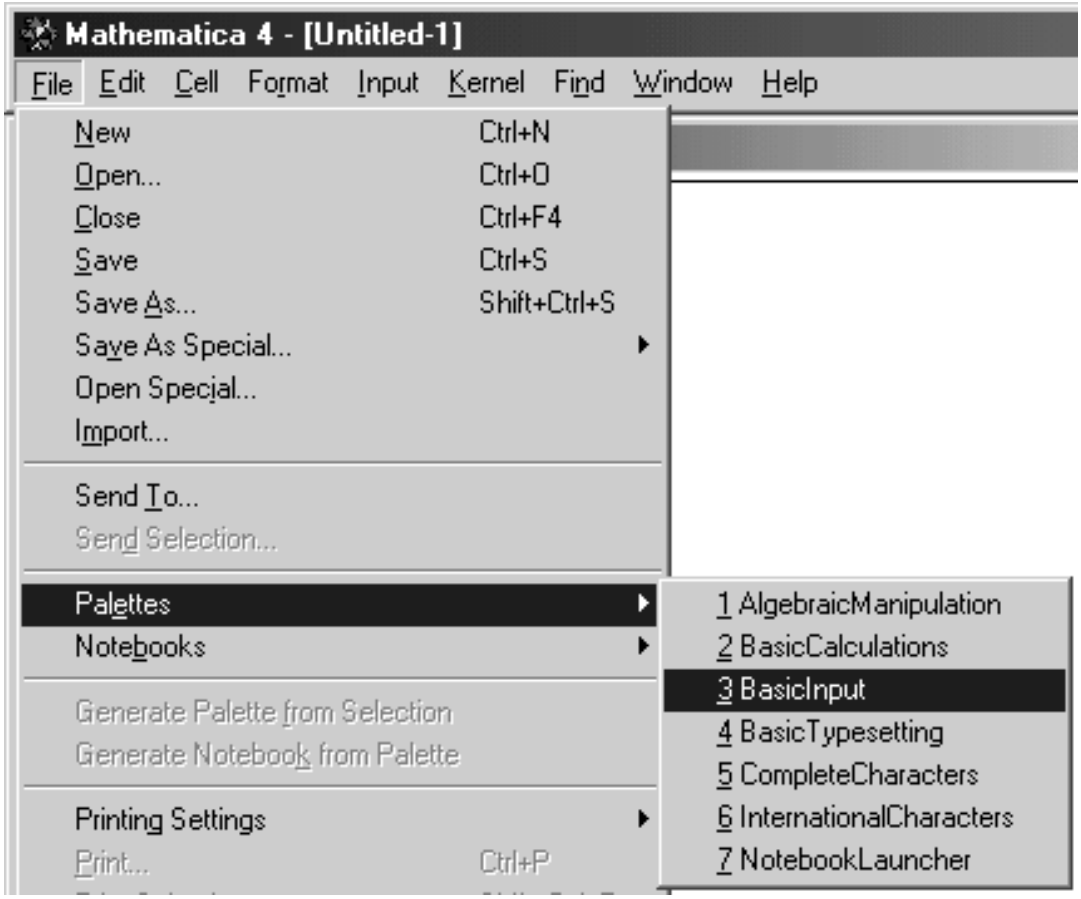
```
Mathematica 2.0 for IBM RISC System 6000
Copyright 1988-91 Wolfram Research, Inc.
 -- X11 windows graphics initialized --

In[1]:=
```

Na ekranu se v okolju Windows pojavi naslednje delovno okno (od verzije 4.0 s pomožnim menijem), ki se, odvisno od verzije programa in operacijskega sistema, lahko nekoliko razlikuje. Na naslednjih slikah sta prikazani delovni okni zadnjih dveh verzij (4 in 5) in razvidno je, da gre bolj ali manj za

kozmetične popravke med njima, čeprav se bistvene razlike skrivajo v možnostih in algoritmih, ki jih nudita verziji, in uporabniku niso tako očitne:



Pomožne menije je mogoče tudi izključiti. V primeru, ko so kasneje potrebni, se lahko prikličejo z ukazom *File* → *Palletes* → *BasicInput*:



V verziji 5 je uporabniku na voljo tudi posebno okno, kjer lahko izbere opcijo *Ten-Minute Tutorial*, ki ga uvede v svet zmožnosti *Mathematice*:

3. Osnovni ukazi

V tem poglavju bo predstavljenih zgolj nekaj osnovnih ukazov (ostali potrebni ukazi bodo prikazani pri reševanju kontretnih inženirskih problemov), opozoriti pa je potrebno, da je pri njihovi uporabi potrebno paziti na dejstvo, da *Mathematica* loči med malimi in VELIKIMI črkami.
Podajanje ukazov ali operacij, ki naj jih program izvrši, se izvede v vhodnih vrsticah, označenih z *In[]*, kjer med oklepajema stoji zaporedna številka ukaza. Opozoriti je potrebno, da med samim pisanjem vhodnega niza le ta še ni opredeljen kot vhodna vrstica in tako označen z označbo *In[]* (saj so posamezne vrstice lahko tudi besedilo, naslovi..), ki se pojavi šele, ko se niz izvede. Ko je podajanje ukaznega niza končano, se torej izvede njegova izvršitev s hkratnim pritiskom na tipki *Shift+Enter*.
Rezultat (ali pa obvestilo o morebitni napaki) se pojavi v izhodni vrstici, označeni z *Out[]*, kjer med oklepajema stoji zaporedna številka izhodnega podatka – rezultata vhodne vrstice. Tako je vedno jasno, kateremu vhodnemu nizu pripada izhodni rezultat.

```
In[1]:= (x + 3) ^ 2

Out[1]= (3 + x)^2
```

V prikazanem primeru je bil kot vhodni ukaz podan izraz $(x+3)^2$. Videti je mogoče, da je zapis v *Mathematici* zelo podoben zapisu, uporabljenem v tem urejevalniku besedil, seveda ob upoštevanju, da znak ^ pomeni na potenco. Z uporabo predefiniranih menijev na desnem robu pa bi lahko zapis že v vhodni vrstici oblikovno še približali zapisu iz urejevalnika besedil. Z uporabo menija  se namreč v ukazni vrstici pojavi nastavek  , kamor se v večji kvadrat (s črnim ozadjem) najprej zapiše argument, v manjši kvadratek (beli kvadratek s črnim robom) pa potenca, kar vodi do:

```
In[2]:= (x + 3)^2

Out[2]= (3 + x)^2
```

Pri primerjavi rezultata analize in vhodnega podatka je razvidno, da se je (ne glede na način vnosa potence) spremenil zgolj vrstni red členov v argumentu – oklepaju, kar pa seveda ne vpliva na rezultat. Z rezultati je mogoče tudi dalje manipulirati. Če želimo npr. razviti posamezni izraz, se to stori z ukazom *Expand[]*, kjer se v oglatem oklepaju (opozorilo: Mathematica loči med okroglimi ( ), oglatimi [ ] in zavitimi { } oklepaji) zapiše argument ukaza. Ker želimo v obravnavanem primeru razviti niz $(x+3)^2$, ki je že bil vnesen v zadnjem vnesenem ukaznem nizu, se lahko uporabi naslednja možnost:

```
In[3]:= Expand[%]

Out[3]= 9 + 6 x + x^2
```

kjer *%* pomeni zadnji dobljeni rezultat (slično pomeni *%%* predzadnji rezultat, itd.). Seveda se po izvedbi novega ukaza vrednost *%* nanaša na nov rezultat. Rezultat je seveda enak, kot če bi se zapisalo *Expand[(x+3)^2]*

ali
Expand[Out[2]]

kjer se številka v notranjem oglatem oklepaju nanaša na drugi rezultat.

V inženirskih problemih, vezanih na dinamiko konstrukcij (kot tudi v mnogih drugih problemih mehanske analize), se pogosto pojavlja integriranje funkcij. V *Mathematici* je tako mogoče analitično izvesti integriranje nedoločenih in določenih integralov, kot tudi numerično integriranje.
Za potrebe zgleda bo tako definirana neka funkcija argumenta x, npr. sin(x).

```
In[4]:= f[x_] := Sin[x]
```

Izbrano ime funkcije je *f*, njen argument (ki ima enako vlogo kot pri klasičnih funkcijskih podprogramih, znanih iz programskih jezikov, kar pomeni, da predstavlja zgolj lokalno spremenljivko znotraj definicije funkcije, kar se doseže z _), zapisan med oglata oklepaja (obvezno), je *x* (zapisan sicer kot x_), := pomeni definiranje, sin(x) pa je zgolj informacija, kako se v obravnavanem primeru izračuna vrednost definirane funkcije *f* (paziti potrebno na zapis, saj Sin(x)≠sin(x)). Opozoriti je tudi potrebno, da takemu definiranju funkcije ne sledi izhodna vrstica (*Out[]*). Funkcijo pa je mogoče definirati tudi nekoliko drugače, kar bo predstavljeno v nadaljevanju tega poglavja.

Definirano funkcijo (kot tudi že originalne programsko predefinirane funkcije, kot npr. trigonometrične, logaritemske ali podobno) je mogoče integrirati analitično, z ukazom *Integrate[funkcija, argument]*. Čeprav je inženirsko jasno, da je edino smiselno integriranje po argumentu funkcije, je to potrebno eksplicitno navesti, saj *Mathematica* omogoča integriranje po poljubni spremenljivki:

```
In[5]:= Integrate[f[x], x]

Out[5]= -Cos[x]
```

saj le to vodi do pričakovanega rezultata (le da brez integracijske konstante). Enakovreden je seveda zapis s pomočjo ukaza iz pomožnega menija:

```
In[5]:= ∫ f[x] dx

Out[5]= -Cos[x]
```

Seveda je potrebno vedno navesti integracijsko spremenljivko, saj se poskus integracije brez navedbe argumenta namreč konča z obvestilom o napaki:

```
In[6]:= Integrate[f[x]]

      Integrate::argmu :  Integrate called with
         1 argument; 2 or more arguments are expected.

Out[6]= Integrate[Sin[x]]
```

integracija po drugi spremenljivki, npr. y, pa se izvede sicer korektno (definirana funkcija se smatra kot konstanta), vendar dobljeni rezultat običajno ne predstavlja želenega:

```
In[7]:= Integrate[f[x], y]

Out[7]= y Sin[x]
```

Seveda pa je mogoče izračunati tudi določeni integral, le da je potrebno definirati še spodnjo in zgornjo mejo, ki sledita v zavitem oklepaju zapisanem argumentu - *Integrate[funkcija, {argument, spodnja meja argumenta, zgornja meja argumenta}]*:

```
In[8]:= Integrate[f[x], {x, 0, π/4}]

Out[8]= 1 - 1/√2
```

kar je seveda ekvivalent (zgolj preglednejšemu) zapisu:

```
In[8]:= ∫_0^(π/4) f[x] dx

Out[8]= 1 - 1/√2
```

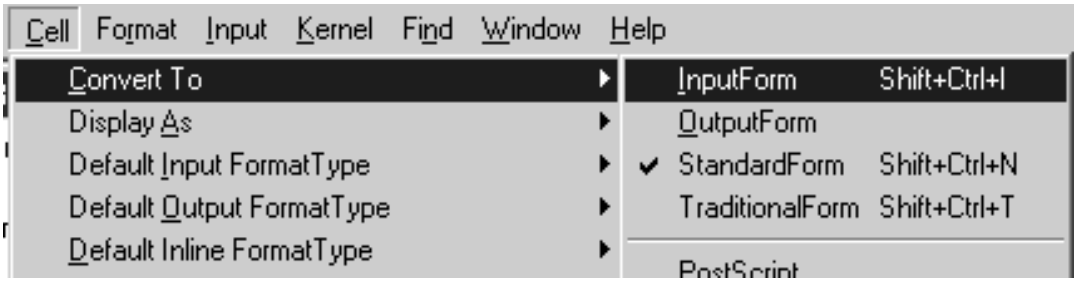
Čeprav je jasno, da je rezultat določenega integrala numerična vrednost, se *Mathematica* trudi, da jo zapiše v čimbolj splošni obliki (npr. ulomki, koreni,...), saj je tudi zgornja meja podana v obliki ulomka. Če pa bi se zgornja meja podala v inženirsko bolj »domačem« zapisu z decimalnimi mesti, npr. kot 0.7854, bi pa seveda sledil (matematično manj natančen, a inženirsko popolnoma uporaben) rezultat v številski obliki, brez poskusa tvorbe morda možnega racionalnega zapisa.
Če pa je rezultat potrebno iz splošne oblike predstaviti v številski obliki, se uporabi ukaz *N[argument]*, kjer se v oklepaju zapiše argument:

```
In[9]:= N[%]

Out[9]= 0.292893
```

Izpisana vrednost se od dejansko izračunane vrednosti razlikuje po številu izpisanih decimalnih mest, saj izpisana vrednost predstavi samo nekaj decimalnih mest. Če uporabnik potrebuje še ostala računska decimalna mesta, mora izhodno vrstico označiti:

```
In[9]:= N[%]

Out[9]= 0.292893
```

ter nato uporabiti ukaz *Cell → Convert To → Input Form* (oziroma *Ctrl + Shift + I*):



kar vodi do izpisa dodatnih decimalnih mest (če seveda le ta obstajajo):

```
In[9]:= N[%]

Out[9]= 0.29289321881345254
```

V nekaterih primerih pa določene integracije ni mogoče (ali pa ni potrebno) izvesti analitično, in se lahko uporabi kar numerična integracija. Ukaz je pravzaprav enak kot pri določenem integralu, le da se doda črka *N* neposredno pred ukaz - *NIntegrate[funkcija, {argument, spodnja meja argumenta, zgornja meja argumenta}]*:

```
In[10]:= NIntegrate[f[x], {x, 0, π/4}]

Out[10]= 0.292893
```

V tem primeru je rezultat podan direktno z numerično vrednostjo. Natančnost dobljenega izraza je npr. mogoče preveriti z iskanjem razlike med rezultatoma obeh integracij:

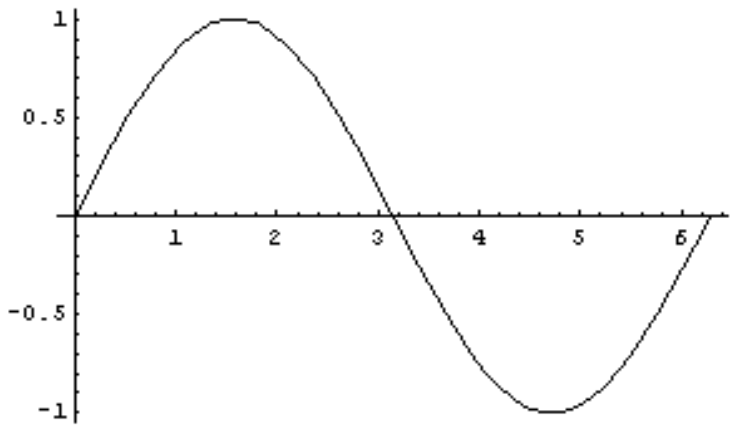
```
In[11]:= %% - %

Out[11]= 1.11022 × 10^-16
```

Dobljena vrednost je zelo majhna, v *Mathematici 5* pa je dobljeni rezultat celo 0. Očitno gre za razliko, ki inženirsko ni zanimiva. Takšnim (majhnim) rezultatom se je mogoče izogniti z ukazom *Chop[argument]*, ki vsa realna števila v argumentu, manjša od (predefinirane) vrednosti 10^{-10} nadomesti z vrednostjo 0. Pri uporabi tega ukaza sta potrebna posebna previdnost in dobro poznavanje velikostnega razreda rezultata analiziranega problema, da ne pride do napake (če se seveda operira z izredno majhnimi vrednostmi).

```
In[12]:= Chop[%]

Out[12]= 0
```

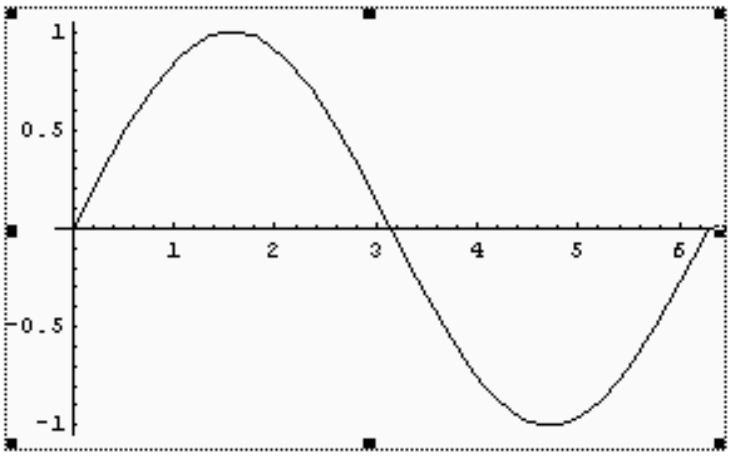
Seveda pa je mogoče funkcijo tudi izrisati. V primeru enostavne funkcije je to mogoče narediti z ukazom *Plot[funkcija, {argument, začetna meja argumenta, končna meja argumenta}];*.

```
In[13]:= Plot[f[x], {x, 0, 2 π}];
```



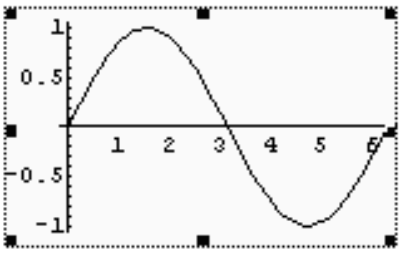
Velikost izrisane slike je mogoče spreminjati direktno po izrisu. Slika se najprej označi s klikom z miško:

```
In[13]:= Plot[f[x], {x, 0, 2 π}];
```



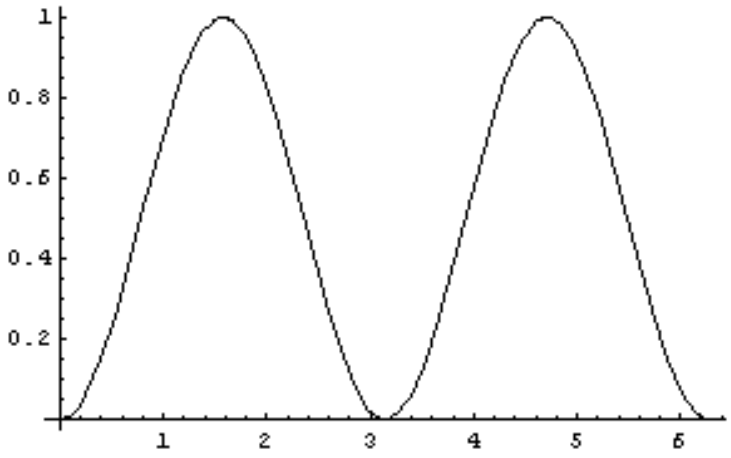
nato pa se s spreminjanjem robov slika lahko proporcionalno poveča ali pomanjša (pri ponovni izvedbi vrstice z izrisom se nato ohrani izbrana velikost slike):

```
In[13]:= Plot[f[x], {x, 0, 2 π}];
```



Če se za ukazom *Plot* podpičje izpusti, se za sliko pojavi še zapis - *Graphics* -, ki pa ne vpliva na nadaljnje delo:

```
In[14]:= Plot[f[x]^2, {x, 0, 2π}]
```

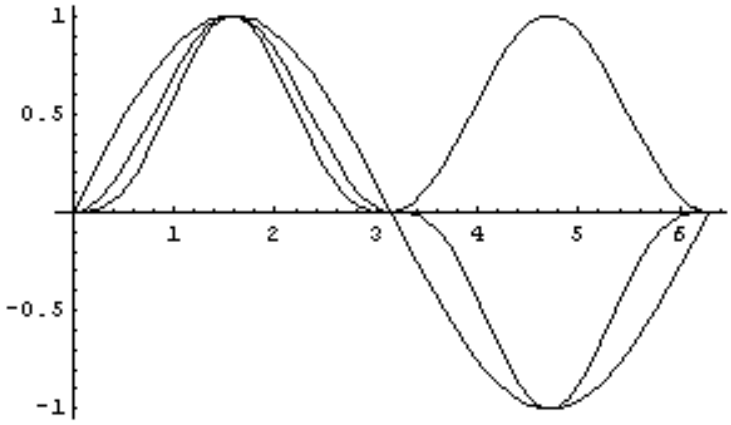



```
Out[14]= - Graphics -
```

Zadnji primer je tudi pokazal, da je ukaz Plot sposoben izrisati ne samo direktno podano funkcijo, ampak tudi posredno podano, npr. v obliki matematične manipulacije več funkcij neposredno pred izrisom, saj je sedaj izrisan kvadrat funkcije $f(x)^2$.

Možen je tudi hkraten izris več funkcij, podanih v zavitem oklepaju in medsebojno ločenih z vejicami. Seveda se vse funkcije izrišejo v istem intervalu:

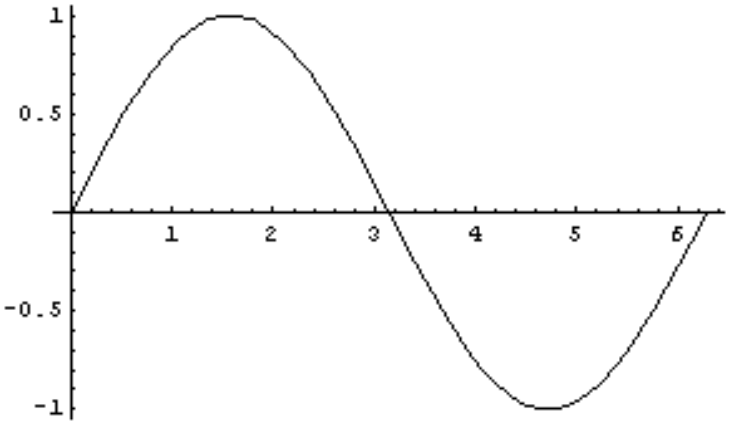
```
In[15]:= Plot[{f[x], f[x]^2, f[x]^3}, {x, 0, 2π}];
```



Tak izris ni najbolj praktičen, saj so vse funkcije izrisane v isti barvi in zato ni jasno, kateri funkciji pripada posamezna krivulja. Zato je primerneje vsako funkcijo izrisati v svoji barvi. Kadar pa ima vsaka funkcija celo svoj interval izrisa, je potrebno vsako funkcijo izrisati posebej v samostojni sliki z izbrano barvo, to sliko primerno imenovati, ter nato vse izrisane funkcije prikazati skupaj na novi sliki.

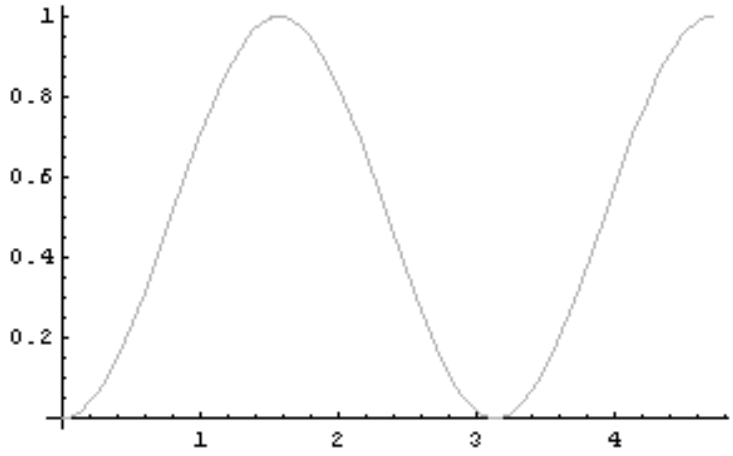
Osnovno funkcijo tako narišemo npr. v predefinirani barvi (črni) in sliko imenujemo npr. *original*:

```
In[16]:= original = Plot[f[x], {x, 0, 2 π}];
```



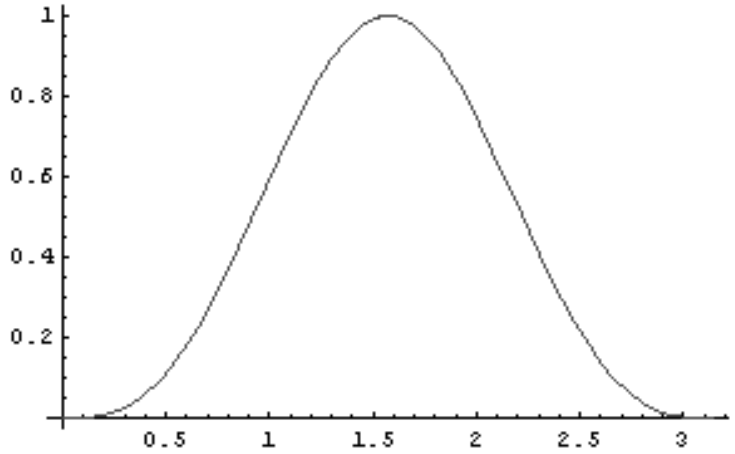
Kvadrat osnovne funkcije narišemo npr. v svetlo modri barvi z opcijo *PlotStyle-> RGBColor[0,1,1]*, kjer vrednosti v oklepaju podajajo vsebnost (med 0 in 1) treh osnovnih barv: rdeče (R), zelene (G) in modre (B). Opcija PlotStyle nudi uporabniku še mnogo drugih možnosti. Tako se lahko definira tip (polna črta, črtkana črta, ...) ali debelina krivulje Sliko imenujemo npr. *kvadrat* (opcija *RGBColor[0,1,1]* pa bo povzročila, da bosta v barvi izrisa enakomerno zastopani zelena in modra barva)

```
In[17]:= kvadrat = Plot[f[x]^2, {x, 0, 1.5 π},
           PlotStyle -> RGBColor[0, 1, 1]];
```



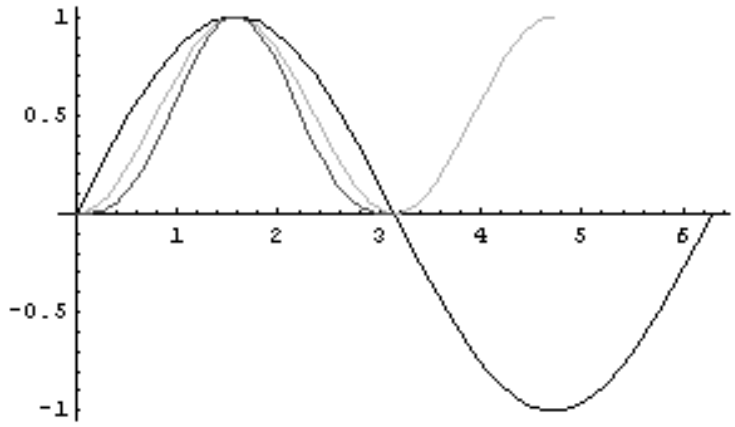
Kubno vrednost osnovne funkcije narišemo npr. v osnovni rdeči barvi z opcijo in imenujemo npr. *kub*:

```
In[18]:= kub = Plot[f[x]^3, {x, 0, π},
             PlotStyle -> RGBColor[1, 0, 0]];
```

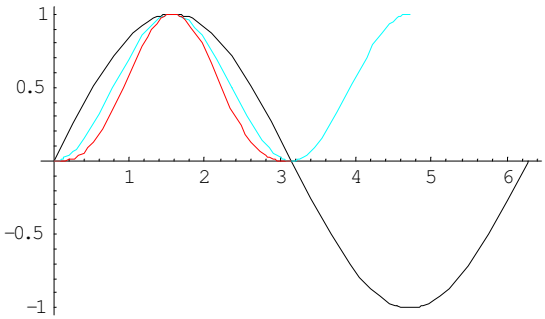


Vse slike hkrati pa se nato prikažejo s pomočjo ukaza *Show[slika1,slika2,slika3]*:

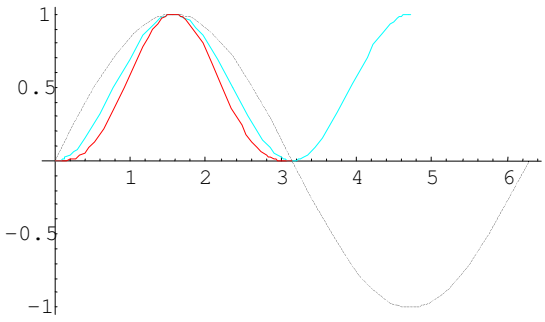
```
In[19]:= Show[original, kvadrat, kub];
```



Prenos slike npr. v urejevalnik besedil je mogoč kar preko *odložišča* (*clipboard*), saj se slika označi z miško, shrani v odložišče (s *Ctrl+C*) in se nato prilepi (s *Ctrl+V*) v urejevalnik besedil. Tako prenesena funkcija izgleda takole:



Tako sliko je mogoče v urejevalniku besedil nato tudi dodatno urejati, npr. z ukazom *Uredi sliko* (*Edit picture*)



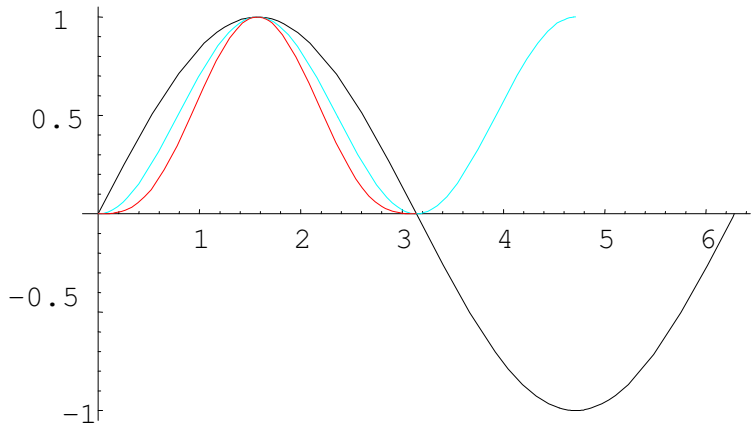
Možen pa je tudi prenos slike v obliki bitnega zapisa (*bmp*), ki se doseže tako, da se v *Mathematici* najprej klikne na sliko, nato pa se izbere opcija *Copy As*, ter se nato izbere opcija *Bitmap* (možne so tudi druge oblike).

Za prenos slik v aplikacije, kamor to ni mogoče storiti preko odlagališča, ali pa se želi kvalitetnejša slika (preko odložišča prenešena slika v verziji 4 ni na zgledno visokem kvalitetnem nivoju), se slika v *Mathematici* lahko shrani na trdi disk tudi z ukazom (ki v zapisani obliki stoji neposredno za izrisom slike) *Display[”izbrano ime slike”,%, ”Metafile”];*.

```
In[20]:= Display["slika", %, "Metafile"];
```

Prikazana oblika ukaza se uporabi, če se slika shrani na disk neposredno po izrisu (zato ukaz %). Če pa se želi na disk shraniti druga, prej izrisana slika, se uporabi ukaz v obliki *Display[”izbrano ime slike”,Out[številka vrstice z rezultatom], ”Metafile”];*, pa čeprav slike, izrisane s podpičjem na koncu ukaza za izris (v vhodni vrstici), nimajo označbe *Out[]*. Še elegantnejša in očitnejša je opcija *Display[”izbrano ime slike”,ime slike, ”Metafile”];* kjer *ime slike* podaja ime spremenljivke, kjer je shranjena izrisana slika, npr. *Display[”izbrano ime slike”,kvadrat, ”Metafile”];*.

Shranjena slika se nato lahko v novo aplikacijo včita z diska z naslova c:\Program Files\Wolfram Research\Mathematica\4.0\ime slike. Tako shranjena in nato včitana slika je bistveno kvalitetnejša kot slika, neposredno prenešena preko odložišča. Tudi ta slika se prav tako dalje lahko manipulira oz. oblikuje.



V inženirski aplikacijah pa pogostno nastopajo tudi odvodi izrazov ali funkcij. Odvod funkcije ene same spremenljivke se lahko označi kar s ', in *Mathematica* bo avtomatično izvedla odvajanje po funkcijski spremenljivki oz. argumentu funkcije (tukaj je torej stopnja »umetne inteligence« programa nekoliko višja kot pri integriranju, kjer je potrebno argument eksplicitno navesti):

```
In[21]:= f'[x]

Out[21]= Cos[x]
```

Če pa je funkcija funkcija več spremenljivk, se odvajanje z ukazom ' ne izvede, ampak zgolj nakaže v izpisu.

Slično je mogoče izračunati tudi višje odvode:

```
In[22]:= f'''[x]

Out[22]= -Cos[x]
```

Na enak način je seveda mogoče izračunati odvode tudi programsko predefiniranih (in ne samo uporabniško definiranih) funkcij, kot npr. *Log[]*. Opozoriti je potrebno, da ukaz *Log[argument]* v resnici izračuna naravni algoritem argumenta, torej *Ln(argument)*:

```
In[23]:= Log''[x]

Out[23]= -1/x^2
```

Kadar pa je zahtevan višji odvod funkcije in postane takšno označevanje nepregledno, ali pa se računa parcialni odvod, se uporabi ukaz *D[funkcija, argument]*, kjer se kot *funkcija* zapiše predefinirana ali uporabniško definirana funkcija, ali izraz, katerega odvod po eksplicitno navedenem parametru *argument* je iskan:

```
In[24]:= D[x^3 + Sin[x], x]

Out[24]= 3 x^2 + Cos[x]
```

Če pa je iskan višji odvod (npr. triindvajseti), pa se le ta zapiše v zaviti oklepaj, takoj za parametrom *spremenljivka*:

```
In[25]:= D[x^3 + Sin[x], {x, 23}]

Out[25]= -Cos[x]
```

4. Definiranje uporabniških funkcij

Kot je bilo že omenjeno in pokazano, je v *Mathematici* mogoče tudi definirati svoje, torej uporabniške funkcije. Pri tem je mogoče izbirati med dvema navidezno podobnima, a vsebinsko različnima načinoma definiranja funkcij: z takojšnjo izvedbo definicije v obliki *ime_funkcije[argument(i)]=definicija* (shrani se vrednost spremenljivk) ali z zakasnelo oz. odloženo (sprotno) izvedbo definicije v obliki *ime_funkcije[argument(i)]:=definicija* (shrani se označba spremenljivk).
Najbolj očitna razlika nastopi, če se v definiciji poleg argumentov pojavljajo še druge spremenljivke, ki že imajo definirano vrednost. V prvem načinu definicije (takojšnja izvedba) se namreč shranijo trenutne oz. aktualne vrednosti definiranih spremenljivk, v drugem (odložena izvedba) pa zgolj označbe (imena) spremenljivk, (takratne) vrednosti spremenljivk pa se upoštevajo šele pri uporabi

funkcije. Če po definiranju funkcije ni prišlo do spremembe vrednosti v definiciji uporabljenih spremenljivk, med definicijama ni razlike, če pa pride do spremembe vrednosti spremenljivk, pa definiciji definirata različni funkciji in posledično vodita do različnih rezultatov, kar demonstrira spodnji primer.

Naj bo tako spremenljivka, ki se bo uporabila v definiciji, npr. A:

```
In[26]:= A = 3;
```

Sedaj se definira funkcija (presledek med členoma pomeni množenje) z takojšnjo izvedbo definicije (*Mathematica* v definiciji funkcije shrani vrednost spremenljivke A, ime spremenljivke pa tako ni več pomembno, saj se zapis shrani kot 3 x):

```
In[27]:= funkcija1[x_] = A x

Out[27]= 3 x
```

ter funkcija z zakasnelo, odloženo izvedbo definicije:

```
In[28]:= funkcija2[x_] := A x
```

Izračun s prvo funkcijo (s takojšnjo izvedbo definicije) vodi do pričakovanega rezultata ob argumentu npr. 3:

```
In[29]:= funkcija1[3]

Out[29]= 9
```

prav tako pa tudi z drugo funkcijo (ob istem argumentu):

```
In[30]:= funkcija2[3]

Out[30]= 9
```

Če pa sedaj pride v procesu analize nekega problema do spremembe spremenljivke A, npr.:

```
In[31]:= A = 4;
```

pa funkciji vrneta različna rezultata. Izračun s prvo funkcijo (s takojšnjo izvedbo definicije) tako ohrani že znani rezultat (saj v zapisu definicije funkcije ni prišlo do sprememb):

```
In[32]:= funkcija1[3]

Out[32]= 9
```

spremeni pa se rezultat, dobljen s funkcijo, kjer pride do odložene oz. sprotne izvedbe definicije, saj se upošteva trenutna vrednost spremenljivke A:

```
In[33]:= funkcija2[3]

Out[33]= 12
```

Če se torej želi zagotoviti, da se bo zapis definicije ohranil skozi ves izračun, je bolje uporabiti definiranje z takojšnjo definicijo funkcije, če pa se funkcija najprej definira v splošnem zapisu z zaenkrat še neznanimi koeficienti (ki pa bodo pridobljeni z nadaljnjo analizo), pa je primernejši drugi način definiranja.

Pri definiranju več uporabniških funkcij pa lahko nastane nepregledno, kako je posamezna funkcija definirana. Seveda se je mogoče vrniti v delovni datoteki nazaj do definicije posamezne funkcije in preveriti njen način definiranja, mogoča pa je tudi direktna informacija o funkciji ali pa tudi o vsakem uporabljenem simbolu. Ta informacija se dobi tako, da se pred imenom funkcije ali spremenljivke zapiše ?:

```
In[34]:= ? funkcija1

         Global`funkcija1

         funkcija1[x_] = 3 x
```

Iz rezultata je sedaj jasno, za kakšno definicijo funkcije gre (jasno se vidi, da je namesto spremenljivke A, uporabljene pri definiranju funkcije, bila shranjena njena vrednost, torej 3), saj za drugo funkcijo sledi:

```
In[35]:= ? funkcija2

         Global`funkcija2

         funkcija2[x_] := A x
```

Seveda je mogoče preveriti tudi navadno spremenljivko (kot tudi spremenljivko, v kateri je shranjena slika, npr. *?original*) in njeno trenutno vrednost:

```
In[36]:= ? A

         Global`A

         A = 4
```

Če se zahteva informacija o še ne uporabljeni spremenljivki (ki tako še ni definirana, in tako ni jasno, ali gre za spremenljivko, definicijo, ime slike,....), sledi:

```
In[37]:= ? B

         Information::notfound :  Symbol B not  found.
```

Obstajajo pa tudi predefinirane sistemske spremenljivke, kot npr. C (ter N, D in *i*), katerim ni mogoče dodeliti vrednosti:

```
In[38]:= C = 3

         Set::wrsym :  Symbol C is Protected.

Out[38]= 3
```

Čeprav na prvi pogled izgleda, kakor da je definiranje, kljub opozorilu, da je C zaščitena spremenljivka, uspelo, nadaljnje manipuliranje s spremenljivko C pokaže, da to ni res, ampak da se je dejansko zgolj informativno izpisala vrednost, ki se je poskušala definirati spremenljivki:

```
In[39]:= 2 C

Out[39]= 2 C
```

Vpogled v spremenljivko C pokaže, da gre v bistvu za (integracijsko) konstanto, ki se pojavi pri reševanju diferencialnih enačb:

```
In[40]:= ?C

C[i] is the default form for the
  ith constant of integration produced in
  solving a differential equation with DSolve.
```

Vpogled v spremenljivko pa ima poleg osnovnega nivoja (z ?) še globlji nivo, namreč z ??, ki pokaže, da je C zaščitena spremenljivka:

```
In[41]:= ??C

C[i] is the default form for the
  ith constant of integration produced in
  solving a differential equation with DSolve.

Attributes[C] = {Protected, ReadProtected}
```

Oba nivoja informacij je mogoče uporabiti tudi na sistemsko predefiniranih matematičnih funkcijah, npr.

```
In[42]:= ?Sin

Sin[z] gives the sine of z.
```

ali

```
In[43]:= ??Sin

Sin[z] gives the sine of z.

Attributes[Sin] = {Listable, NumericFunction, Protected}
```

Mogoče pa je pridobiti tudi splošno informacijo o grafičnih funkcijah, kar lahko služi kot pomoč:

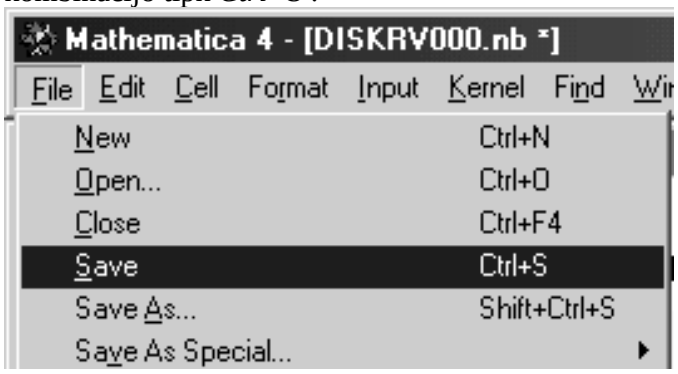
```
In[44]:= ?Plot

Plot[f, {x, xmin, xmax}] generates a plot of f as a
  function of x from xmin to xmax. Plot[{f1, f2, ... },
  {x, xmin, xmax}] plots several functions fi.
```

ali celo informacije o vseh parametrih, ki vplivajo na izris (vse te informacije o predefiniranih funkcijah, matematičnih ali grafičnih, je mogoče dobiti tudi v vgrajeni pomoči – meniju *Help*):

```
In[45]:= ?? Plot

Plot[f, {x, xmin, xmax}] generates a plot of f as a
  function of x from xmin to xmax. Plot[{f1, f2, ... },
  {x, xmin, xmax}] plots several functions fi.

Attributes[Plot] = {HoldAll, Protected}

Options[Plot] = {AspectRatio → 1/GoldenRatio,
  Axes → Automatic, AxesLabel → None,
  AxesOrigin → Automatic, AxesStyle → Automatic,
  Background → Automatic, ColorOutput → Automatic,
  Compiled → True, DefaultColor → Automatic,
  Epilog → {}, Frame → False, FrameLabel → None,
  FrameStyle → Automatic, FrameTicks → Automatic,
  GridLines → None, ImageSize → Automatic, MaxBend → 10.,
  PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
  PlotRange → Automatic, PlotRegion → Automatic,
  PlotStyle → Automatic, Prolog → {}, RotateLabel → True,
  Ticks → Automatic, DefaultFont :→ $DefaultFont,
  DisplayFunction :→ $DisplayFunction,
  FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

5. Shranjevanje datotek

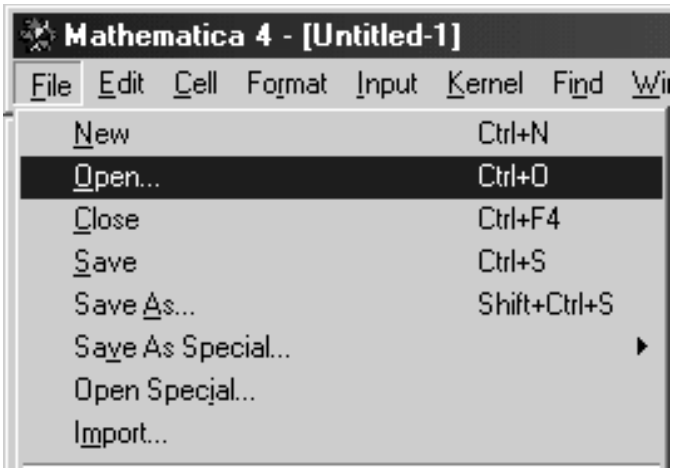
Na koncu dela (ali po potrebi tudi med delom) je svoje delo mogoče shraniti z ukazom *File/Save* ali s kombinacijo tipk *Ctrl+S* :



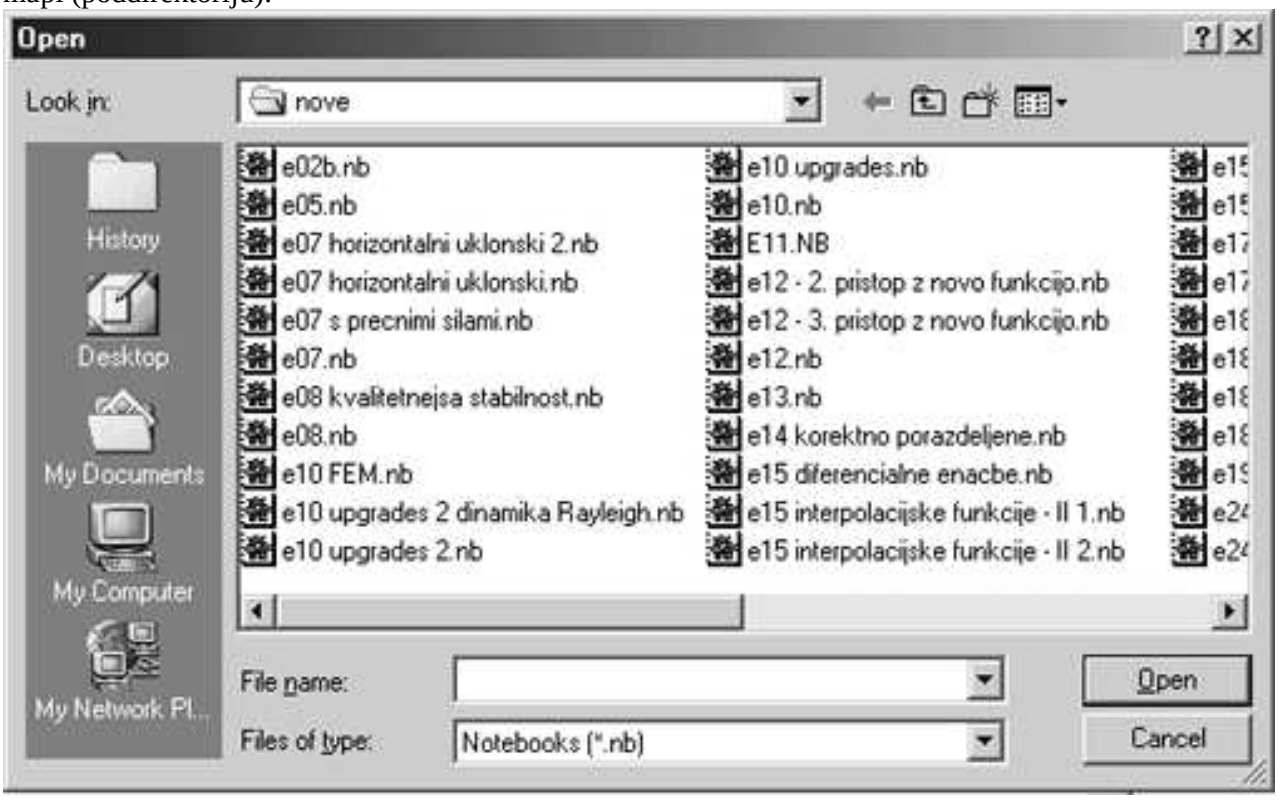
Končnica izhodnih datotek (od verzije 3 naprej) je *nb* (notebook), v prejšnjih (nižjih) verzijah pa je bila *ma*.

6. Uporaba obstoječih datotek

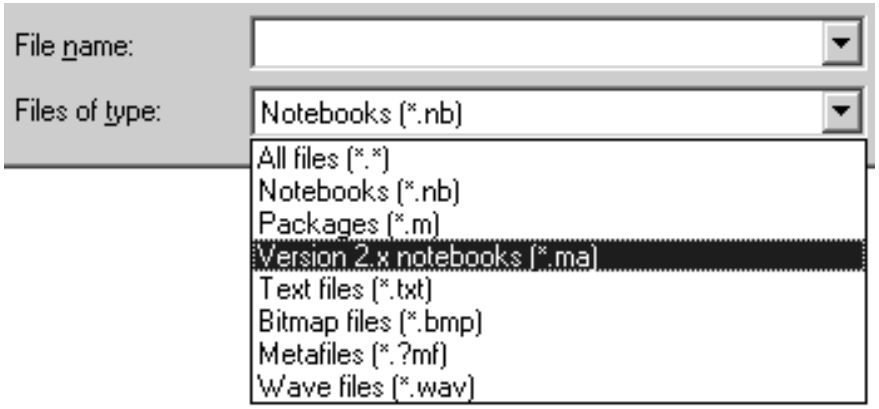
V primerih, ko se želi uporabiti na disku shranjena datoteka, se le ta včita z ukazom *File/Open* ali s kombinacijo tipk *Ctrl+O* :



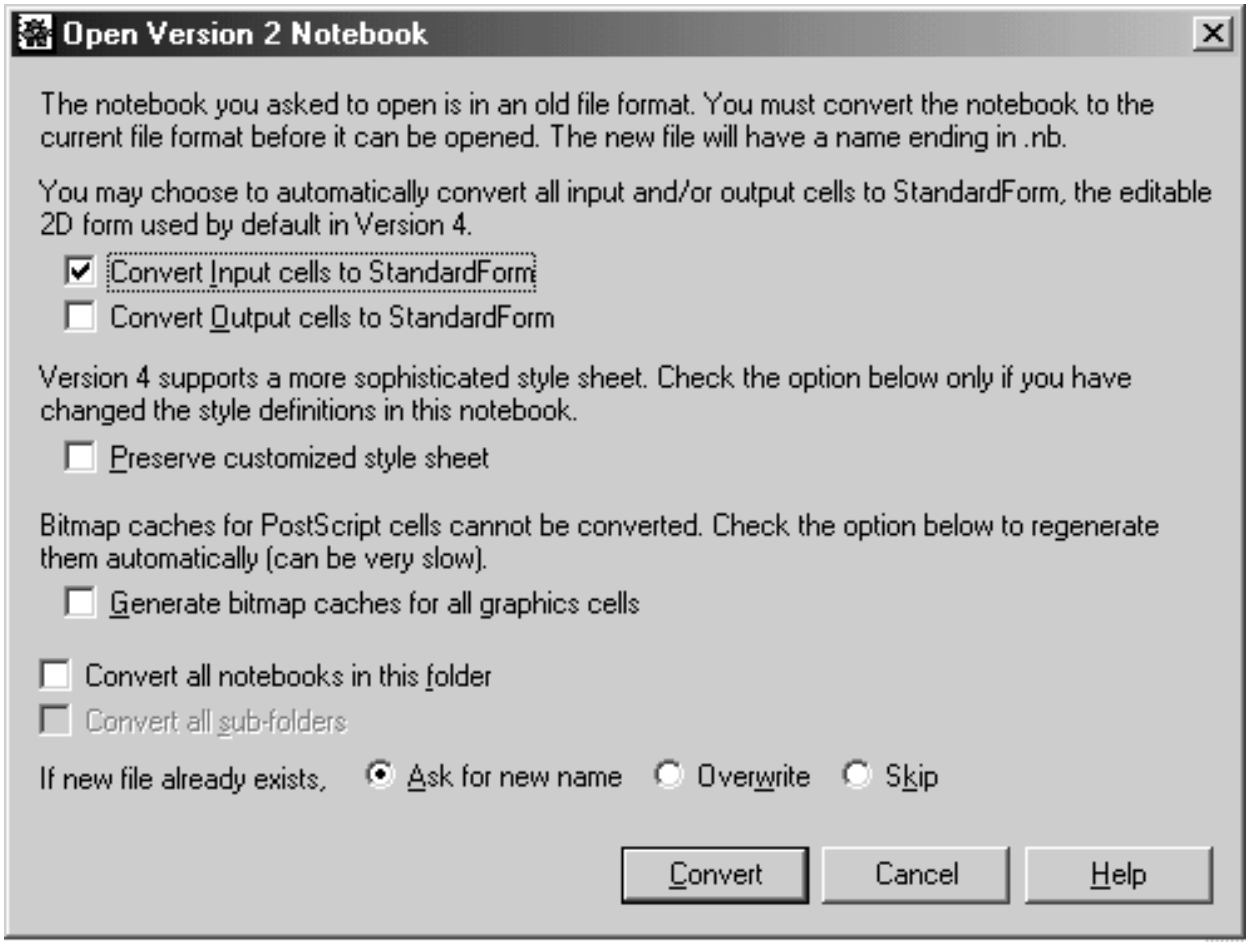
Ukazu seveda sledi prikaz vseh izhodnih datotek s končnico *nb* (notebook), ki se nahajajo v trenutni mapi (poddirektoriju):



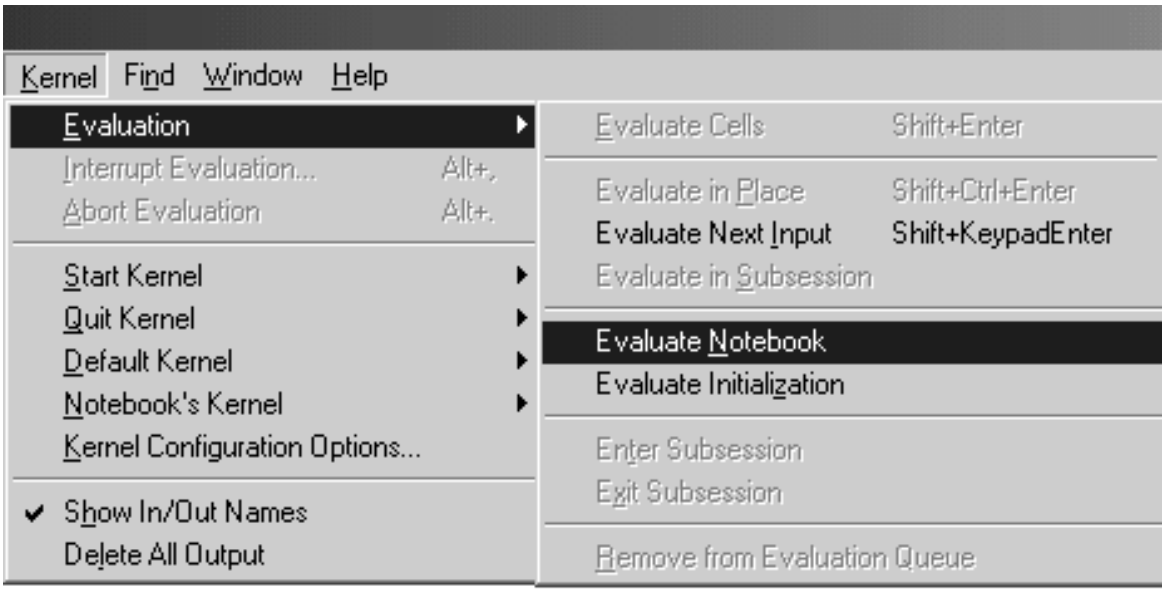
Če pa se želi včitati datoteka, narejena z verzijo Mathematice, manjšo od 3, ki ima končnico *ma*, je potrebno izbrati ustrezno opcijo *Files of type*:



Po izbiri ustrezne datoteke se pojavi okno, kjer se nastavijo opcije, ki vplivajo na pretvorbo starega zapisa v zapis, ki ga uporablja aktualna verzija (tudi zapisa med verzijama 4 in 5 se nekoliko razlikujeta, vendar je načeloma verzija 4 tudi sposobna včitati datoteke, narejene z verzijo 5).



Ko je datoteka včitana, je mogoče ponoviti izvedbo posameznih vrstic z ukazom *Shift+Enter*. Če se želi izvesti izvedba vseh vrsti (celic), pa se uporabi zaporedje ukazov *Kernel* → *Evaluation* → *Evaluate Notebook.*


Kernel Find Window Help
Evaluation
Interrupt Evaluation... Alt+,
Abort Evaluation Alt+.
Start Kernel
Quit Kernel
Default Kernel
Notebook's Kernel
Kernel Configuration Options...
Show In/Out Names
Delete All Output
Evaluate Cells Shift+Enter
Evaluate in Place Shift+Ctrl+Enter
Evaluate Next Input Shift+KeypadEnter
Evaluate in Subsession
Evaluate Notebook
Evaluate Initialization
Enter Subsession
Exit Subsession
Remove from Evaluation Queue