NTNU – Trondheim
Norwegian University of
Science and Technology

Modeling and Analysis of Noise

Hege Auglænd

Master of Science in Engineering and ICT
Submission date: June 2013
Supervisor: Tor Guttorm Syvertsen, KT

Norwegian University of Science and Technology
Department of Structural Engineering

NTNU
Department of Structural Engineering

MASTER THESIS 2013

for

stud.techn. Hege Auglænd

Modeling and Analysis of Noise

Modellering og beregning av støy

Background

Norwegian mines are required to present a noise map over their areas. Quite often these maps are made by external contractors, however, it is desirable to be able to create noise maps using in-house software. The software used today is MicroStation, and an extension for noise computations would save time and money. The objective of this thesis is to develop a plug-in for MicroStation in cooperation with Norconsult Informasjonssystemer, SINTEF and NTNU.

Approach

- The extension to MicroStation will be programmed in C# using Microsoft Visual Studio.
- A computational model from SINTEF will be used as the foundation for the computations.
- A course in MicroStation programming will be taken in addition to an introduction to noise modeling and computation.
- Visnes Kalk AS, a limestone mine, will be used as a reference, as both a model and a noise map of the area are available.

Result

The thesis will result in an extension to MicroStation and a digital report, which will be graded. The extension will have the following features:

- Import a terrain model - already supported in MicroStation
- Create one or more noise sources
- Create one or more buildings
- Create noise abatements
- Compute the noise distribution
- Display a noise map
- Display exact values for buildings and specific points.

It is desired that the system is interactive for easy adding of noise abatement, rerunning the computation and presenting an updated noise map.

The report is to be handed in to the Department of Structural Engineering by June 10, 2013.
The thesis may be adjusted during the project due to the progress of work and the interests of the student.
The report is to be organized in accordance with the current instructions (http://www.ntnu.no/kt/studier/masteroppgaven).

Contacts at Norconsult Infomasjonssystemer:
Ole Magne Kvindesland (Ole.Magne.Kvindesland@norconsult.com) and
Frode Tørresdal (Frode.Torresdal@norconsult.com)
Contacts at SINTEF Akustikk:
Herold Olsen (Herold.Olsen@sintef.no) and
Rolf Tore Randeberg (Rolf.T.Randeberg@sintef.no)
Contact at NTNU, Department og Geology and Mineral Resources Engineering:
Erik Ludvigsen (Erik.Ludvigsen@ntnu.no)

Professor: Tor G. Syvertsen (torgsyv@gmail.com)

Trondheim, January 19, 2013

Tor G. Syvertsen (sign.)
Professor

Norwegian Abstract: Sammendrag

Støy er et miljøproblem som kan ha negativ effekt på helse og velferd. Norsk lov har fastslått anbefalte støygrenser for bebodde og andre støysensitive områder. Støybelastede områder skal ha kartlagt støysoner for å kunne visualisere og finne behov for støyskjerming.

Den norske gruveindustri genererer en betydelig mengde støy og må kartlegge effekten av denne støyen på omkringliggende områder. I dag blir dette ofte gjort ved å leie inn eksterne konsulenter som lager støykart over områdene. Dette er både dyrt og kan ta veldig lang tid og noen selskaper ønsker derfor å kunne gjøre kartleggingen selv. Det finnes programmer som kan gjøre dette i dag, men disse er både dyre og omfattende. Noen av disse vil trenge et komplett bytte av systemer, noe som er veldig dyrt og tidskrevende. Siden MicroStation er et mye brukt program i dag, er det ønskelig med et programvaretillegg som håndterer støyanalyse og modellering til dette programmet.

Programvaretillegget laget i dette prosjektet kan brukes både i gruveindustrien og i andre prosjekter, som vei- og industriprosjekter. Den viser støy kartlagt i henholdt til fargekoder gitt i norsk lov. I tillegg vises støykart med kotelinjer med 3 dB mellom hver linje og detaljerte verdier for spesifiserte punkter og fasadepunkter på hus.

Abstract

Noise is an environmental issue which may have a negative impact on health and wellbeing. Norwegian legislation states recommended noise limits for inhabited and other noise sensitive areas. Areas susceptible to noise problems should have mapped noise zones, to visualize and clarify any noise abatement needs.

The mining industry in Norway generates a substantial amount of noise. As a result, they are required to map the noise level of the surrounding area. Today many companies hire external consultants to create noise maps for their areas. This is very expensive and may take a substantial amount of time and some companies would prefer to be able to do the mapping themselves. There are software solutions available for this purpose, however these are expensive and comprehensive solutions. Some of these solutions would require a change of software system. As MicroStation is used by many today, a plug-in that handles noise analysis and modeling in MicroStation has been requested.

The plug-in created in this project may be used in the mining industry and in other projects, such as road or industry projects. It shows noise mapped as color coded contour lines in accordance with Norwegian legislation. In addition, it shows noise maps with contour lines for every 3 dB and exact values for selected points and house facades.

Preface

This report, a user manual and the plug-in developed for MicroStation is the end result of the course TKT4915 Computational Mechanics, Master Thesis. This is a part of my Masters Degree in Engineering and ICT, Structural Engineering at the Norwegian University of Science and Technology, NTNU, in Trondheim.

I would like to thank Norconsult Informasjonssystemer for the opportunity to work with this project, in addition to helping me and providing what I needed when it came to courses, software and hardware. Special thanks to my two contacts: Ole Magne Kvindesland who found the project for me and Frode Tørresdal who has been helping me every step of the way.

In addition, I would like to thank Erik Ludvigsen and Herold Olsen for teaching me about noise, noise modelling and measurement. SINTEF ICT, Acoustics and Norwegian Public Roads Administration (Statens Vegvesen) has been kind to let me use their base computational model - SoundKernel. Rolf Tore Randeberg at SINTEF ICT, Acoustics have been giving me good code examples and helping me when I had any questions. Torunn Moltumyr at Norwegian Public Roads Administration helped me with road and vehicle details.

Tor G. Syvertsen has been a great asset to my thesis, and I would like to thank him for his guidance and enthusiasm.

Hege Auglænd
Trondheim, June 6, 2013

Contents

Terms and Definitions

Abbreviations

dB	Decibel, logarithmic unit
lg	logarithm base 10
X3D	an XML-based file format for representing 3D computer graphics

Acronyms

AADT	Annual Average Daily Traffic
CAD	Computer Aided Design
DLL	Dynamic Link-Library
DNV	Det Norske Veritas
GUI	Graphical User Interface
IDE	Integrated Development Environment
NPRA	Norwegian Public Roads Administration (Statens Vegvesen)
SOSI	Samordnet Opplegg for Stedfestet Informasjon
XML	Extensible Markup Language

Latin Letters

L_{5AF}	the A-weighted level measured with fast response, only including the top 5 % incidents.
L_{AFmax}	the average of the 5-10 highest occurring noise levels L_{AF} defined in section 3.1, for the night hours of 23-07.
L_{den}	A-weighted Equivalent Level for day-evening-night, with 5/10 dB added for evening and night.
L_{eq}	Equivalent Continuous Sound Level
$L_{evening}$	A-weighted Equivalent Level for the evening hours of 19-23.
lg	logarithm base 10
L_{night}	A-weighted Equivalent Level for the night hours of 23-07.
L_p	Sound Pressure Level
L_W	Sound Power Level

1 Introduction

1.1 Background

What is noise? A simple definition of noise is that noise is unwanted sound. Noise a has negative effect on humans. A report from WHO[1] states that in Europe one in three individuals is annoyed by noise from traffic during the daytime and one in five during night-time. The same report finds evidence to support that serious health problems such as discomfort, stress, sleep disturbance, cardiovascular disease, hearing impairment and tinnitus may be caused or worsened by noise. According to a Norwegian report[2], approximately one in twenty individuals in Norway have large problems with traffic noise and may be facing health risks as a result. As a precaution, regulations and legislations have been created to prevent damaging levels of noise.

Several industries need to comply to these laws and regulations. There are limits for traffic noise related to roads, railways and airports, and limits for various types of industries, ports and terminals. Moreover, there are limits for wind turbines and particular noise sources such as firing ranges. To be able to document compliance of noise regulations, a noise mapping has to be performed. This may be done by using existing software, such as SoundPLAN[3] and NovaPoint Noise[4]. However, these systems tend to be large and expensive and require a large amount of time and training.

If the necessary software is not owned by the company wanting to do the noise mapping, an external consultant has to be hired, which may be both expensive and time consuming. It is therefore desired to be able to do the noise mapping within an already used software, in this case MicroStation. Visnes Kalk AS is a mine in Norway which had a noise mapping performed in 2012[5]. These results will be used to verify the plug-in results in section 4.3.

1.2 Scope

The plug-in should be able to add roads as noise sources and compute a noise map based on a terrain model. The output should be a map showing yellow and red noise limits, as explained in section 3.3: Legislation and Regulations.

Additional features:

- Detailed and accurate grid based on terrain mesh
- Noise sources:
  - Drilling rig as a volume source. The area will be selected from MicroStation, a rig is chosen, time and frequency is given as input
  - A Hammer, which is similar to the drilling rig. It has as impulse or continuous noise
  - Industry noise
  - Loading and unloading of cargo ships
  - Point noise sources in general
- Noise Mitigation elements such as screen, wall and/or mound.
- Graded noise map, showing the noise in the area as color gradings
- Exact values: The ability to select specific points, areas or volumes to get exact calculated noise values on for instance critical buildings.

The actual implementation may be limited by time, technologies, calculation time or SoundKernel - the computation model.

1.3 Outline of the report

This report is composed of four main parts:

- Overview of the relevant software components, including MicroStation and programming tools
- Simple sound theory, information about current legislation and Visnes Kalk AS
- Implementation: what has been done, what functionality has been added and results obtained.
- Discussions and conclusions

In addition, references and appendices may be found at the end of the report.

2 Technologies

2.1 Programming Tools

The main editing tool used to create the plug-in is Microsoft Visual Studio[6] which is an IDE - Integrated Development Environment. This means that it includes a code editor, debugger and designer, all that is needed to create a software application. It includes a debugger to help fix errors in the code and a forms designer to create GUI applications. In addition there are many other tools available that have not been used in this thesis.

Team Foundation Service from Microsoft was used to safely save the code and to provide source control. The server used belonged to Norconsult Informasjonssystemer and enabled the contacts there to view and comment on the code.

The main language used is `C#`[7][8] which is developed by Microsoft and is a general-purpose object-oriented programming language fairly similar to Java. In addition, XML[9] was used to define commands in MicroStation.

2.2 MicroStation

MicroStation is a CAD-application developed by Bentley Systems used for 2D and 3D design and drafting. It is one of the dominant CAD packages today, and competes with AutoCAD[10] for market shares. As a tool based program, the buttons on the left menu starts most functions. In addition there is a command line based interface, where it is possible to find the same functions and a few more. The screen shot below shows a basic view of MicroStation with the terrain model for Visnes Kalk AS loaded.

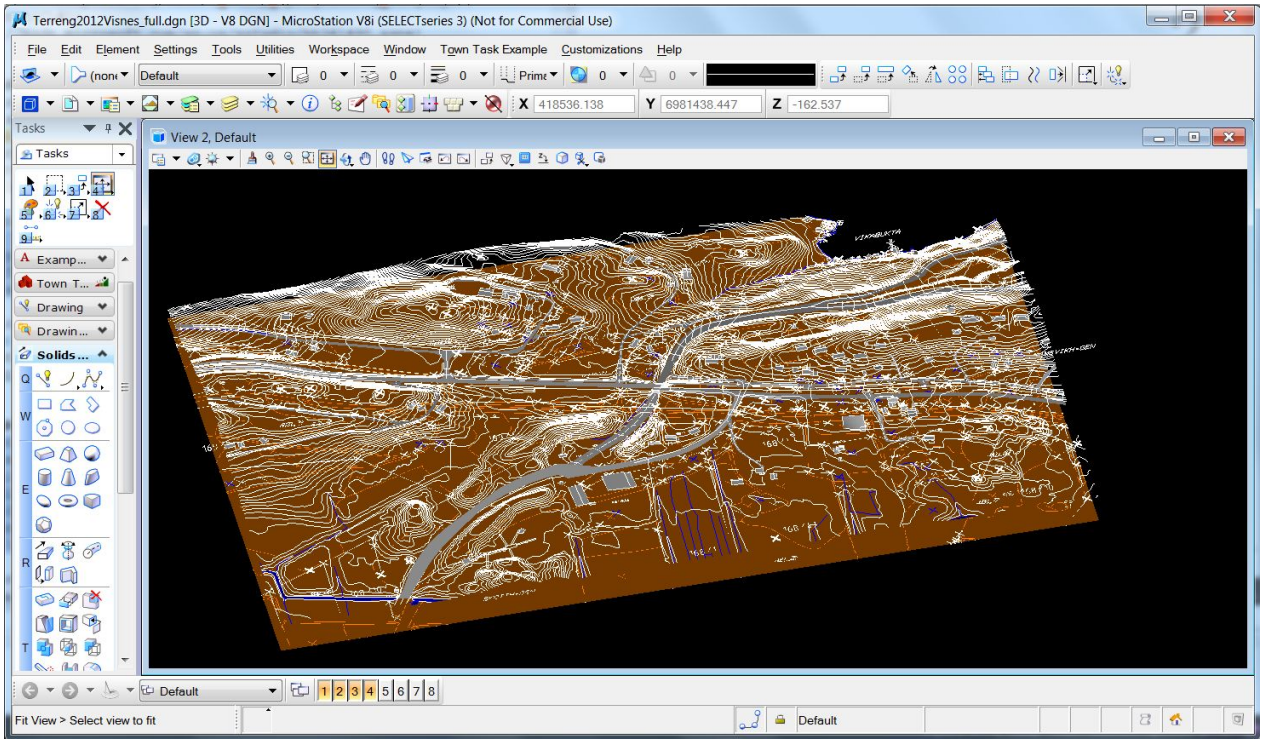


Figure 2.1: Terrain model displayed in MicroStation

MicroStation has all of the basic functions included in most CAD programs, the most relevant to this project are:

Table 2.1: Important functions in MicroStation

Function	***Description***
Import of terrain models	obtains a model of the surrounding area, without having to make it from scratch. SOSI data, which is terrain models from The Norwegian Mapping Authority (Kartverket), is compatible with MicroStation. When a SOSI file is imported into MicroStation, it will look as in figure 2.1 and all existing roads, buildings, etc. will be displayed.
Creation of lines, shapes and solids	manipulates the loaded terrain model. It might be necessary to change elevation lines or add new buildings, noise abatement screens or mounds, etc.
Creation of meshes	A mesh has to be created to be able to find grid points to be used in the computation. In MicroStation a mesh looks like a smooth surface created based on elevation or other lines defining the shape of the model.
Level information	are used to distinguish drawing objects. A level may be "House", "Road", etc. and you may choose if objects with the specified level is visible. For large models, such as SOSI, this is very useful. New default levels may be created, for instance to indicate noise level.
Programmatic extensibility	The possibility to extend MicroStation, using several well known programming languages and frameworks such as Microsoft .NET, `C++`, `C#`, Visual Basic, etc. In this thesis .NET and `C#` will be used. Data may be easily exchanged to and from MicroStation, and entire plug-ins may be created. This is the most important feature, and essential to this thesis.

2.3 SoundKernel

The base computational work for the noise mapping will be SoundKernel. SoundKernel is a small software module initially developed by SINTEF ICT, Acoustics as a part of a noise computation package for The Norwegian Public Roads Administration called NorStøy. Some changes have been made to make SoundKernel more general and to add support for mining and industry-specific noise sources.

SoundKernel was developed using `C#` and the source code was provided. Several dynamic link-libraries(DLLs) are combined to provide the required functionality. A DLL is a library that may contain code, data and resources and can be used by more than one application at a time. An application normally consist of several modules, these may be divided into separate DLL files to promote code reuse and efficient memory and disc space usage. There are several DLL files provided by SINTEF that have to be included to run SoundKernel. The most important one, and the only one with source code provided is the SoundKernel DLL.

The code is too large and there are too many classes and objects to list them here, so only the most important classes are mentioned:

Table 2.2: The most important classes in SoundKernel

Class	***Functionality***
skContour	defines noise limits and output computed contour lines.
skTopography	contains the surrounding terrain model as a grid in addition to lists of buildings, mounds, roads and noise zones.
skTopScreen	contains mounds or noise mitigation screens. These elements are placed to mitigate noise or to correct any errors in the terrain model.
skTopRoad	define roads as noise sources with information about the traffic on the road.
skTopBuilding	contains buildings defined by points.
skMetClass	contains weather information, a default weather is always used.
skSource	is a general class to contain the relevant parameters for a noise source.
skTask	contains information about the computation that will be performed.

Information will be gathered from MicroStation and used as input to SoundKernel. When the specified task in SoundKernel is run, the results are gathered and shown in MicroStation. Details about how SoundKernel is used and the functionality obtained may be found in the Software Development section in sub section 4.1.

3 Background Information

3.1 Sound

Sound is physical compression waves that are oscillating rapidly in either a gas, liquid or solid. Humans perceive sound waves within the sonic range, between 20 Hz and 20 kHz. Sound is created at a sound source and is transmitted in all directions from the source through a compressible media. There are many ways of quantifying, measuring and processing sound and a few of the most important ones will be briefly discussed in this section.

Sound is measured in decibel (dB) and its spectrum frequency is measured in Hz. A sound source is quantified by sound power (Watt), the sound in the surroundings is quantified by sound pressure (Pascal) and both are expressed in decibel[11]. A decibel is a logarithmic unit based on a physical quantity and a reference level. The physical unit is usually either power, intensity or pressure. The reference levels are different for these quantities, which means that a pressure based decibel is not the same as a power based decibel.

Shown below are the equations for calculating sound pressure and sound power levels[12].

Sound Power Level:

$$L_W = 10lg\frac{P}{P_0}\ dB,$$

where the reference value $P_0 = 1 * 10^{-12}\ watt$ and P is the sound power of a source.

Sound Pressure Level:

$$L_p = 10lg\frac{p^2}{p_0^2}\ dB,$$

where the reference value $p_0 = 2 * 10^{-5}\ Pa$ and p is the sound pressure.

There are many variables that quantify a sound signal: frequency, pressure, intensity, speed, amplitude, etc. A sound frequency spectrum shows the oscillation frequency of the sound signal. It is found by using a Fourier transform and usually plotted as amplitude, power, intensity or phase vs. frequency.

A sound wave's physical characteristics are directly related to how the listener perceives the sound. For a specific frequency, a larger pressure amplitude will increase the perceived loudness[13]. The sound the human ear is able to hear depends on the sound pressure level and the frequency.

A high frequency of 20 000 Hz is a very high pitched sound, while a low frequency of 20 Hz is a deep base sound. A normal conversation is generally at 60 dB and 100-1000 Hz. Figure 3.1 shows the correlation between frequency, sound pressure level and audible sound[14]. A line in the image has equal loudness for the different combinations of frequency and sound pressure level.

A change in sound pressure level of 2 or 3 dB is noticeable. A 10 dB increase makes the sound seem twice as strong and equals the difference between having one and ten machines running at the same time.

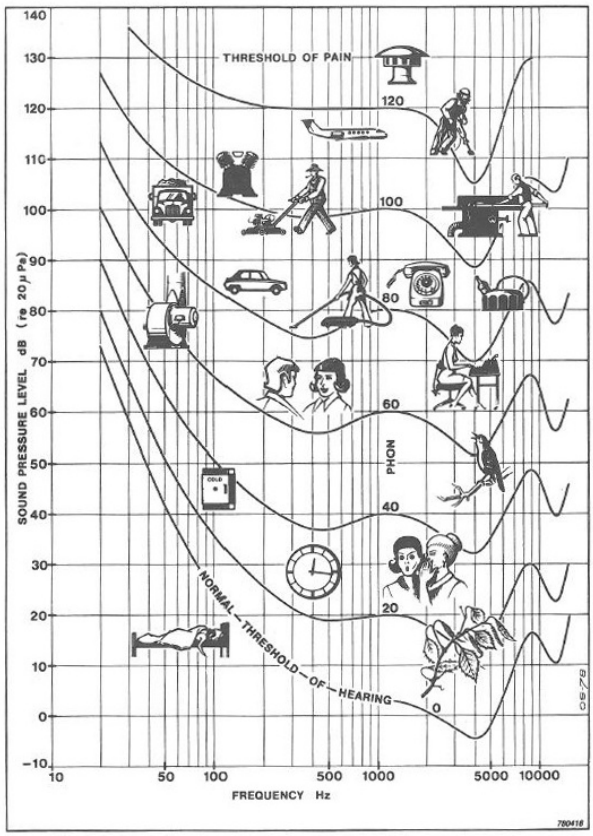


Figure 3.1: Typical sound pressure levels of common noise sources

When measurements of sound pressure levels are made, the measurements are grouped into octaves. An octave is a frequency interval, commonly called a bandwidth[13], where $f_2 = 2f_1$[15].

Measured sounds are filtered so that the sensitivity varies with frequency, such as for a human ear. There are three different types of filters, called weighting networks:

A - Low sound pressure levels

B - Medium sound pressure levels

C - High sound pressure levels

A-weighting is widely used as a measure of possible hearing damage, annoyance caused by noise and compliance with legislation and regulations such as in Section 3.3. The equation for weighing is given in ISO 3744[12]:

$$L_{WA} = 10lg \sum 10^{0,1(L_W+C)} dB,$$

where L_W is the sound power at the specified frequency, C is the attenuation value for the frequency and L_{WA} is the A-weighted sound power level.

The attenuation values for A-weighting are given in Table 3.1:

Table 3.1: A-weighting Attenuation Values

Mid-band frequency [Hz]	63	125	250	500	1000	2000	4000	8000
C [dB]	-26,2	-16,1	-8,6	-3,2	0,0	1,2	1,0	-1,1

Equivalent continuous sound level L_{eq} is given by the A-weighted level of steady noise that has the same A-weighed energy as the actual time-varying noise in question. ISO 1996[16] gives the equation:

$$L_{Aeq,T} = 10lg\left[\frac{1}{T}\int_T p_A^2(t)/p_0^2 dt\right] dB,$$

where $p_A(t)$ is the A-weighted instantaneous sound pressure at running time t.

Rating levels are created for the different periods of the day. These rating levels are used to describe a noise environment by creating a whole-day composite rating level. This is described in ISO 1996[16].

$$L_{Rden} = 10lg\left[\frac{d}{24} * 10^{(L_{Rd}+K_d)/10} + \frac{e}{24} * 10^{(L_{Re}+K_e)/10} + \frac{24-d-e}{24} * 10^{(L_{Rn}+K_n)/10}\right] dB,$$

where:

d	number of daytime hours
e	number of evening hours
L_{Rd}	rating level for daytime
L_{Re}	rating level for evening-time
L_{Rn}	rating level for night time
K_d	adjustment for weekend daytime, if applicable
K_e	adjustment for evening-time
K_n	adjustment for night time

L_{den} is often used as an improvement on L_{Aeq} and takes into account the increased annoyance of noise on evenings and nights. There are many other measurements of noise and noise annoyance, only the most relevant are mentioned in this report. Additional measurements not previously mentioned but important for environmental noise and in section 3.3: Legislation and Regulations, are listed below:

$L_{evening}$	A-weighted equivalent level for the evening hours of 19-23.
L_{night}	A-weighted equivalent level for the night hours of 23-07.
L_{AFmax}	the average of the 5-10 highest occurring noise levels L_{AF} for the night hours of 23-07.
L_{AF}	the A-weighted noise level with fast (125 ms) response.
L_{5AF}	the A-weighted level measured with fast response, only including the top 5% incidents, giving a statistical maximum level.

3.2 Mitigating noise

To reduce noise levels in areas, different types of noise mitigation are used. The main categories are[11]:

- Traffic reduction
- Speed limit reduction
- Reduced noise emission from each vehicle
- Hide the noise sources from the general public
- Hide the general public from the noise sources
- Reduce peoples reactions to the noise
- Change the behavior of the general public
- Change the behavior of the travelers

The usual technologies involved in noise mitigation are traffic management, cars, tires and road covers. In addition noise barriers and sound proofing are used with noise monitoring, new technologies and various tools for calculation and analysis. There are several instruments that may be used to reduce noise, including legislation, economy, globalization, political will and prioritizing. For Visnes mine, mentioned in Section 3.3, legislation is a drive to make sure that the surrounding housing areas have acceptable noise levels.

In the surroundings the most common noise mitigation measures are noise barriers which normally give 5-10 dB reduction in noise level. A well placed row of buildings may give up to 20 dB noise reduction for surrounding buildings.

For an industry area the most relevant ways to mitigate noise are:

- speed and vehicle reduction on industry roads
- more efficient use of machinery, to decrease total number of hours used every day
- noise mitigation walls, screens and mounds
- reduced activity outside normal workdays and work hours
- sound proofed production buildings

Visnes Kalk AS has used noise mitigation mounds in several areas, with good results.

3.3 Legislation and Regulations

The Norwegian pollution regulations, Forurensningsforskriften[17], states noise limits around residential areas, hospitals, educational institutions, kindergartens and other care institutions. These are as follows:

Table 3.2: Noise limits, from Forurensningsforskriften(2004)

Monday - Friday	***Evening Mon-Fri***	***Saturday***	***Sunday and Publ.Holidays***	***Night (23-07)***	***Night (23-07)***
$55L_{den}$	$50L_{evening}$	$50L_{den}$	$45L_{den}$	$45L_{night}$	$60L_{AFmax}$

where L_{den} and $L_{evening}$ are defined in section 3.1.

The guidelines for management of noise in land planning, T-144/2012[18], recommend the use of two noise zones, one yellow and one red. In yellow zones residential areas may be put up if noise regulations are satisfied and in red zones residential areas are not allowed. Criteria for the zones are given on the next page. If at least one of the criteria are met the specific area will fall under that zone. The zones should be computed 4 m above the terrain.

The impulse sound mentioned in table 3.3 is characterized by brief bursts of sound pressure lasting less than 1 second. In ISO 1996[16] there are three definitions of impulse noise:

- *High-energy impulsive sound source*
  Explosive sound due to firing heavy weapons, explosives, etc.
- *Highly impulsive sound source*
  Highly impulsive, highly intrusive noise, such as firing light weapons, hammering, using pneumatic tools, etc.
- *Regular impulsive sound source*
  Impulsive sounds that are not highly impulsive or intrusive, such as car doors, church bells, etc.

Only highly impulsive sounds count as impulsive noise lowering the limits of table 3.2 and 3.3 with 5 dB. The variables in table 3.3 are defined in section 3.1.

Table 3.3: Criteria for noise zones, from T-144/2012[18]

		Yellow Noise Zone	
		Outdoor noise level	
Noise Source	Monday-Friday	Sunday, Saturday & Public Holidays	Night 23-07
Road	L_{den} 55 dB		L_{5AF} 70 dB
Railway	L_{den} 58 dB		L_{5AF} 75 dB
Airport	L_{den} 52 dB		L_{5AF} 80 dB
Wind turbines	L_{den} 45 dB		-
Industry with continuous operation	Without impulse L_{den} 55 dB With impulse L_{den} 50 dB		L_{night} 45 dB L_{AFmax} 60 dB
Other industries	Without impulse L_{den} 55 dB and $L_{evening}$ 50 dB With impulse L_{den} 50 dB $L_{evening}$ 45 dB	Without impulse Saturday: L_{den} 50 dB Sunday: L_{den} 45 dB With impulse Saturday: L_{den} 45 dB Sunday: L_{den} 40 dB	L_{night} 45 dB L_{AFmax} 60 dB
Ports and terminals	Without impulse L_{den} 55 dB With impulse L_{den} 50 dB		L_{night} 45 dB L_{AFmax} 60 dB
Motor sports	L_{den} 45 dB L_{5AF} 60 dB		No activity No activity
Firing ranges	L_{den} 30 dB L_{5AF} 60 dB		No activity No activity
		Red Noise Zone	
Road	L_{den} 65 dB		L_{5AF} 85 dB
Railway	L_{den} 68 dB		L_{5AF} 90 dB
Airport	L_{den} 62 dB		L_{5AF} 90 dB
Wind turbines	L_{den} 55 dB		-
Industry with continuous operation	Without impulse L_{den} 65 dB With impulse L_{den} 60 dB		L_{night} 55 dB L_{AFmax} 80 dB
Other industries	Without impulse L_{den} 65 dB and $L_{evening}$ 60 dB With impulse L_{den} 60 dB $L_{evening}$ 55 dB	Without impulse Saturday: L_{den} 60 dB Sunday: L_{den} 55 dB With impulse Saturday: L_{den} 55 dB Sunday: L_{den} 50 dB	L_{night} 55 dB L_{AFmax} 80 dB
Ports and terminals	Without impulse L_{den} 65 dB With impulse L_{den} 60 dB		L_{night} 55 dB L_{AFmax} 80 dB
Motor sports	L_{den} 55 dB L_{5AF} 70 dB		No activity No activity
Firing ranges	L_{den} 35 dB L_{5AF} 70 dB		No activity No activity

3.4 Visnes Kalk AS

Visnes Kalk AS is located in Visnes by Kornstadfjorden in Eide kommune. The main products are limestone, marble and eclogite. The total deposit is about 40 million tons and the annual production is about 600 000 tons, giving an annual turnover of approximately 60 million NOK[19]. An extension of the current zoning plan is desired. As a result an impact assessment has been made in addition to a noise evaluation[5] performed by Det Norske Veritas (DNV).

This mine and the noise evaluation will be a base for verifying the results obtained by the developed plug-in for MicroStation. DNV found that transportation is the main noise source for most of the mine. However, mining activity are critical in some areas and have to be taken into account. The noise model developed by DNV is based on measurements performed on site of noise from processing, transport, port and mining activity. Noise levels are presented as L_{den}, which is A-weighted noise levels for day, evening and night, with 5/10 dB added for evening/night values. Two situations were examined, both giving two noise maps, one with yellow and red noise limits as mentioned in Section 3.3 and one with gradient noise limits. The two situations considered were:

1. Only daytime work and 24 hour loading of ships
2. Only daytime work, transport to the port and 24 hour loading of the ships

Situation one describes the everyday noise better than situation two and will be used in the Testing and Results section. All of the noise maps may be found in Appendix A.

The housing in the area close to the mine is critical to this noise evaluation. There are a few houses inside the red limit and some inside the yellow limit. Noise mitigation action may be necessary to ensure that these follow current legislation. As previously mentioned, an interactive plug-in would be useful in this situation. Noise mitigation may be used to shield the buildings from the noise and a new map may be computed to see the effect of the mitigation.

The mine usually operates from 7.30 to 15.30, Monday to Friday. When new ships arrive they have to be loaded quickly which causes loading to be performed over the entire 24-hour period. However, the loading is limited to the port area only. On the rare occasion that the storage areas near the port are insufficiently large, transport of material is performed during night and evening time in addition to daytime. This greatly increases the noise levels even if it rarely happens. Years may go by between each such incident.

3.5 Noise Measurement

When measuring noise, there are several important things to consider:

- The instrument used must be suited for the desired type of measurement
- Weather and wind should be noted, especially wind may have a large impact on the measurements. Temperature, humidity and precipitation influence how fast the sound travel. Wind may both increase and decrease the measured noise.
- The area surrounding the noise source are very important. The measurement should be performed at an appropriate distance from the noise source, to be sure to capture it accurately. The surrounding area should ideally be fairly flat and free from noise reflecting surfaces.
- Background noise should be estimated to see if it needs to be considered as large enough to influence the measurement.

On a trip to Visnes Kalk AS, several noise measurements were made. Measurements were made for some of the machines and equipment without any noise data. These may be used in SoundKernel.

L_{eq} was measured and used to calculate L_w, the sound power level used to define the sound source. The equation used is for a free source on or close to terrain level, without any close reflection surfaces[20]:

$$L_{w,okt} = L_{r,okt} + 20log(R) + 8$$

where $L_{r,okt}$ is the measured noise level, L_{eq} and R is the distance from the source to the measurement point.

The measurements and calculated noise levels are shown in full in Appendix B: Noise Measurements from Visnes Kalk AS.

4 Software Development

4.1 Implementation

To be able to properly run the solution and to allow code reuse, two DLL files were created: NoiseComputation and ISY.CAD.Noise. These are connected as shown in the figure below, and the classes and function of NoiseComputation and ISY.CAD.Noise are explained in detail on the next pages.

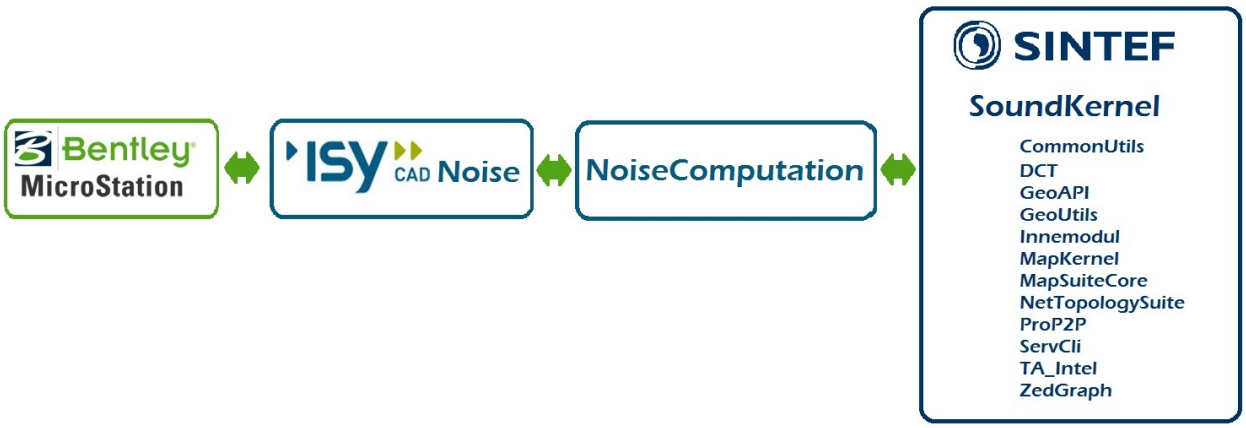


Figure 4.1: Overview over interactions between MicroStation, ISY.CAD.Noise, NoiseComputation and SoundKernel

NoiseComputation
This DLL has a strong connection to SoundKernel and its corresponding DLLs and operates as a link between SoundKernel and ISY.CAD.Noise. NoiseComputation may be used in other projects to connect other programs to SoundKernel. It is general and provides a simple interface for exchanging values.

The classes in NoiseComputation are as listed below. The NoiseRoad, NoiseSource, NoiseMound and NoiseBuilding classes all have constructors and used in ISY.CAD.Noise to give NoiseComputation information about the elements created in MicroStation.

Table 4.1: Overview of NoiseComputation classes

Class	***Function***
NoiseRoad	contains the points, width, velocity, traffic type and distribution type of the road. In addition to if it is a tunnel or bridge and user defined noise spectra and/or traffic distribution, if there are any.
NoiseSource	contains the points of the source in addition to the source type, activity type, geometry, distribution type, amount, amount unit and user defined frequency spectra and time distributions.
NoiseMound	contains the points of the line that is the base of the mound, width at the top and bottom, in addition to left and right height of the mound.
NoiseContour	contains a list of contour line points and the level of the contour line in addition to a constructor. NoiseContour objects are used to save the contour line output from SoundKernel and send them to ISY.CAD.Noise to be drawn in MicroStation.
NoiseBuilding	contains inline and outline points of a building in addition to if facade points should be computed for the building. Inline points are usually the top lines of the building while outline points are the outline of the building at the height where the roof starts.
ComputationMgr	is the main class of the library. It contains methods for creating SoundKernel roads, buildings, topography, mounds, etc. In addition it contains public lists of NoiseRoads, NoiseBuildings, NoiseSources, NoiseMounds and NoiseContours that are accessible from ISY.CAD.Noise and may be altered or used there. The class is too large to explain in detail, but in short the class creates the environment necessary for SoundKernel to run. It creates a topography from the grid information, adds roads, buildings, calculation points, weather, mounds, triangulates the topography model, etc., and runs the computation. This may take some time, and when the computation is finished, contour lines are written to the list of NoiseContours and a summary are written to file.

ISY.CAD.Noise
This DLL deals with MicroStation and deliver information to NoiseComputation so that it may run the computation. It extracts data from MicroStation and manipulates it to suit the objects shared from NoiseComputation. The most important classes shown in table 4.2.

Table 4.2: Overview of ISY.CAD.Noise classes

Class	***Function***
AddBuildingMgr	creates a form to guide the user when adding buildings in MicroStation. When a building has been successfully selected in MicroStation it is added to a list of NoiseBuildings in the RunCalc class.
CalcPointsMgr	enables the user to select a line, line string, shape or solid and add the points to SoundKernel. Exact noise values will be computed for the points. This class extracts points from the chosen element, and adds them to an array of points in RunCalc.
commands	is an XML file mapping commands in MicroStation to functions in the Main class in ISY.CAD.Noise. Commands are added here, and attached to a method in Main.
GridMgr	searches the model for grid lines (Primary and Secondary) and extracts important values such as the smallest and largest x- and y-values in addition to intersection points where the lines cross. These values are later sent to NoiseComputation to be used to create the topography.
IndustryNoiseMgr	enables industry noise sources to be created in MicroStation and saves point coordinates, geometry, time distribution, noise type, amount and frequency spectra to a NoiseSource object that is accessed in NoiseComputation.
Main	contains the methods corresponding to the commands in commands.xml. These methods generally start main methods in the different classes. For instance, the method for the command addBuilding runs the method AddBuilding.StartCommand();
Mesh2GridMgr	creates Primary and Secondary grid lines based on a mesh. This is done by using some integrated MicroStation commands and a lot of code to join line pieces together and find the right z-value for each point in the line. The lines are then drawn in the MicroStation model.
MoundMgr	finds the points of the selected base line of the mound and creates an instance of NoiseMound. The instance contains the points and the heights/widths of the mound and is added to the array of mounds in RunCalc.
RoadMgr	creates a form to guide the user and to enable import of values such as width, velocity, traffic type, AADT, traffic distribution and user defined values. When a line is selected, the points are extracted and a NoiseRoad instance is created and added to the array of roads in RunCalc.
RunCalc	is the "main" class in ISY.CAD.Noise. It collects and interchanges data with NoiseComputation and starts the process of running SoundKernel through NoiseComputation. It contains lists of mounds, roads, sources, calculation points and buildings which it sends to NoiseComputation. When a computation has been run, RunCalc draws contour lines in MicroStation.

4.2 User Manual

Overview

The features of the developed plug-in for MicroStation are as stated in this section. A more detailed ISY.CAD.Noise User Manual[23] is available. This section gives an overview, while the ISY.CAD.Noise User Manual is directed towards more advanced users.

The command line interface in MicroStation, called Key-in, is probably known for many users of MicroStation, however a brief introduction will be provided here. As of now, the new functions created in the ISY.CAD.Noise plug-in may only be accessed using the Key-in.

Accessing
The command line interface may be accessed either by choosing Utilities -> Key-in in MicroStation, or by pressing the Enter-key on the keyboard.

Window explanation
The Key-in window is shown below, the most recent command are shown at the bottom part of the form, and available commands are shown in the scrollable window above.

Executing command
The selected command: "mdl load isycadnoise" is the command to load ISY.CAD.Noise. When this is executed the plug-in may be used. All the commands in the following sections have to be executed in this Key-in form.

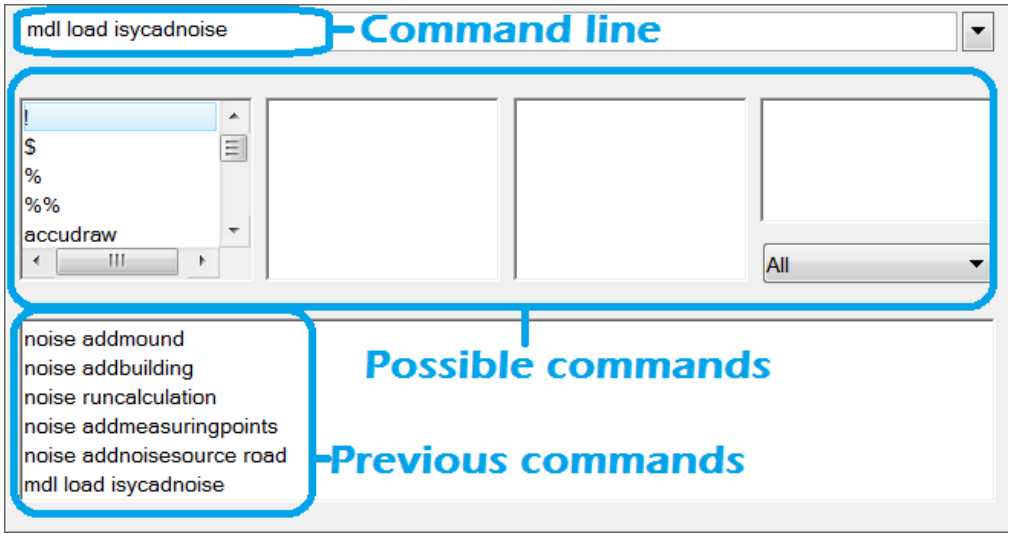


Figure 4.2: MicroStation Command Line Interface

Getting Started

Start MicroStation and establish terrain and mesh:

Terrain model
First establish a terrain model. The easiest way of doing this is to import a terrain model, preferably from SOSI data[22].

Mesh
Then a mesh needs to be established. A mesh connects all the points of the selected elements and creates a surface. This is done by selecting "create mesh from contours" in MicroStation and choosing the lines in the terrain model that are to be the base of the mesh. It is recommended to choose elevation lines (called "Høydekurver" in Norwegian SOSI data). These are displayed by selecting that level using MicroStation Level Display. This will create a nice and even mesh as shown below. The result is shown in figure 4.3. The gray surface is the actual mesh and the white lines are the elevation lines.

The example used in this User Manual is Visnes Kalk AS. As it is a large mine, only a part of the terrain model are used in this example.

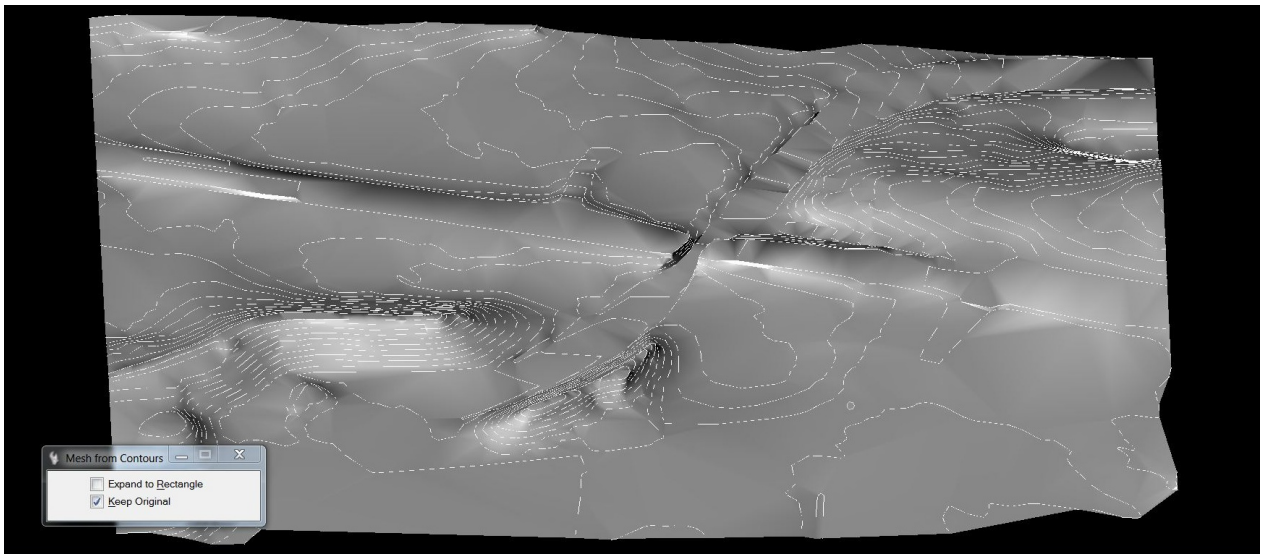


Figure 4.3: A mesh of a part of Visnes Kalk AS

Creation of grid

A grid has to be defined for SoundKernel to develop a topography model for the computation.

Activate the function "Noise Mesh2Grid" in the Key-in to create grid lines based on the mesh in section 4.2. These are straight lines going north to south (Primary grid lines), and east to west (Secondary lines), creating a grid.

The complete result with grid lines are shown below. These lines are saved in the MicroStation model, and this command only have to be executed once for each model. The intersection points of the lines are used as the base of the topography model mentioned in the SoundKernel section.

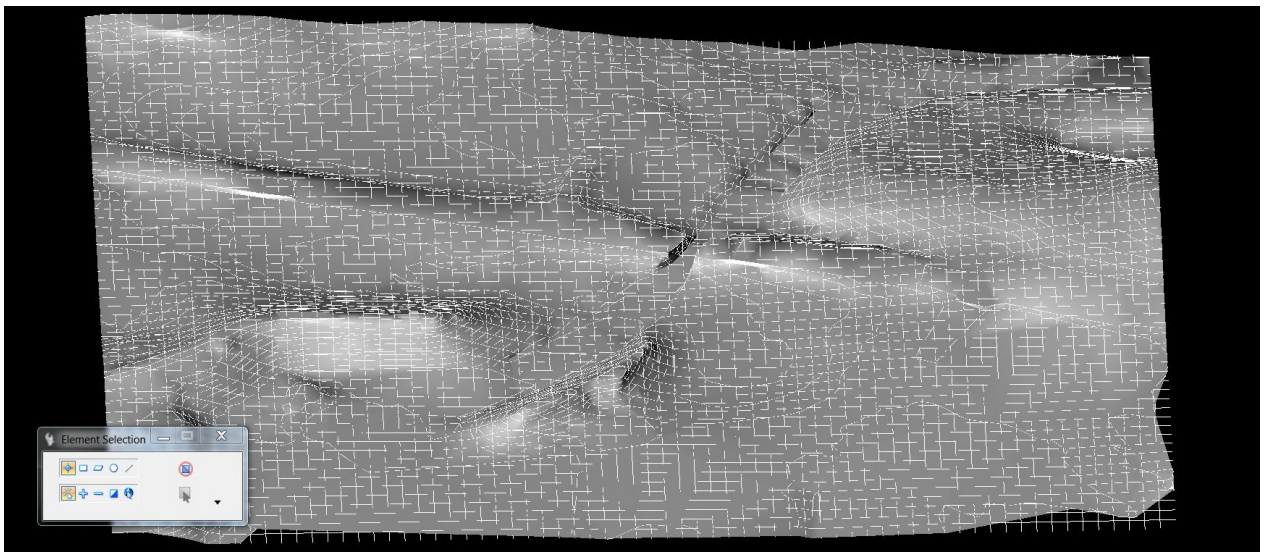


Figure 4.4: Mesh with Primary and Secondary Grid Lines.

The computation will run without errors as soon as a grid is established. However, the results will be zero unless one or more noise sources are added. There are two types of noise sources that may be added: Road and Industry noise source. These are described in the next sections.

Road noise sources

A road noise source is added by activating the "Noise AddNoiseSource" -> "Road" function in the command line interface. The road options form is shown to the right.

To add the road, the center line of the road is selected in the MicroStation model.

The level of the line will be changed to "StøyVeg" - Noise Road in Norwegian. The possible options for a road are explained below:

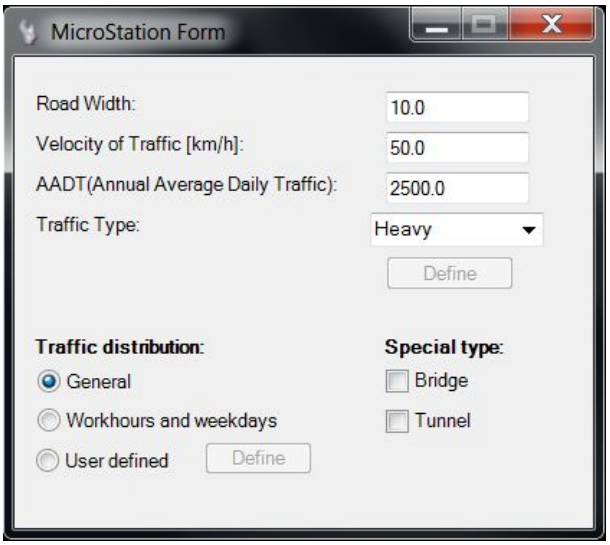


Figure 4.5: Form for adding a Road Noise Source

Table 4.3: Road Options

Option	***Description***
Road width	is defined in meters and as the entire width of the road. For some roads width may be found on the Norwegian Public Roads Administration (NPRA) website[24].
Velocity	is defined in km/h. Values may be found on the NPRA website[24].
AADT	Annual Average Daily Traffic is the number of vehicles passing the road on an average day. Values may be found on the NPRA[24] or Norwegian Directorate for Civil Protection (DSB)[25] websites.
Traffic Type	defines the vehicles driving on the road. There are currently four possible traffic types: Heavy, Normal, Dumper or User Defined. These are defined in Table 4.4.
Traffic Distribution	defines a distribution of AADT over time. The distribution is created for every hour of every day in a week and is the same for all weeks in a year. This is a simplification as traffic may vary throughout a year. There are three possible options: General, Work hours and weekdays and User Defined. These are defined in Table 4.5.
Special type	may be used if the road is a bridge or a tunnel. These elements will have different parameters, for instance will a tunnel generate less noise, except at its ends.

Table 4.4: Traffic Types

Option	***Description***
Normal	this traffic type indicates a common road, for instance a highway, with continuous traffic throughout both day and night. There is a distribution of light, medium and heavy vehicles on the road.
Heavy	this traffic type consist of only heavy vehicles(buses, trucks, etc.).
Dumper	is a traffic type created from noise measurements of a Euclid dumper[5]. A dumper is a large and heavy vehicle common in many industry and construction areas.
User Defined	if there is a specific traffic type on the road, the User Defined traffic type may be used, and the "Define" button opens the form shown below. Decibel values for each frequency may be input here and a user defined traffic type is created.

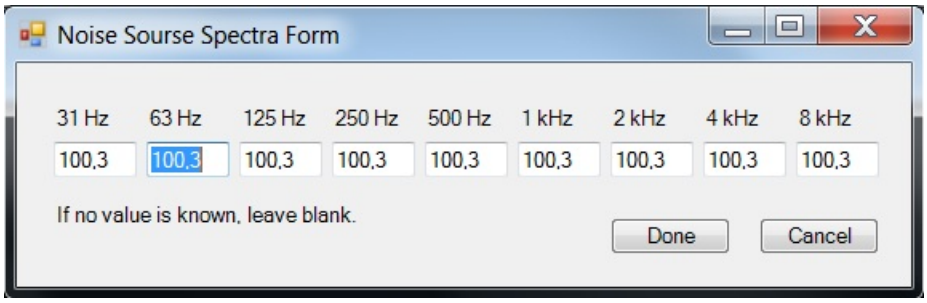


Figure 4.6: User Defined Noise Spectrum

Table 4.5: Traffic Distribution

Option	***Description***
General	defines the distribution on most common roads, such as highways.
Work hours and weekdays	all traffic is distributed between 08 and 16, Monday - Friday.
User Defined	if the traffic distribution is known for the road and does not fit any of the two options above, the user defined option may be used. When the "Define" button is pressed, the form in figure 4.7 is shown. Specific values for daytime, evening and night time on Monday-Friday, Saturday and Sunday may be input. It is important to make sure that the AADT-value is the same as the number of vehicles for a day.

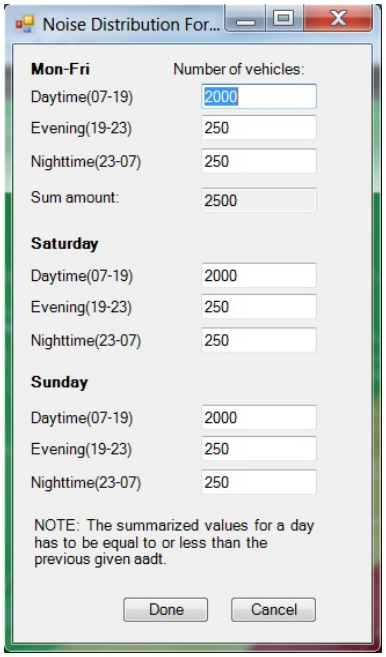


Figure 4.7: User Defined Noise Distribution

The image below shows the form with the options for the road, and a selected road. The terrain model is the same as before, however the levels have been changed to fit this demonstration.

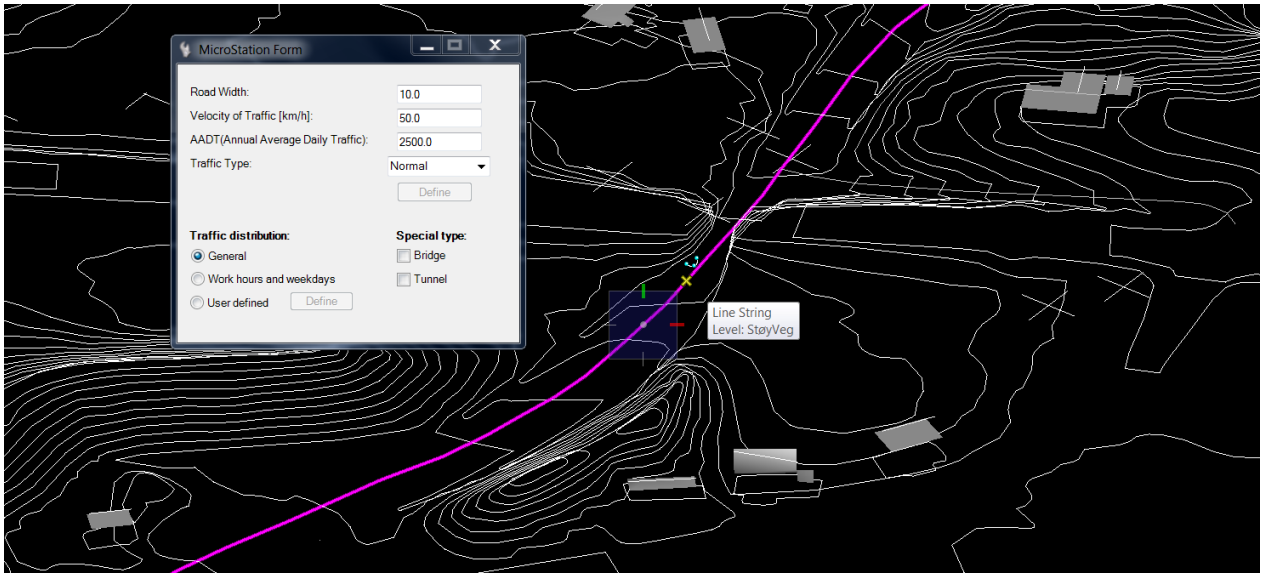


Figure 4.8: Adding a Road Noise Source

Industry noise sources

There are many different types of industry noise sources: machinery, heavy work, construction etc. The currently available noise types in ISY.CAD.Noise are more suited for mining than other industries. However, many other noise types may be added. User defined noise sources enable the use of all types of noise, provided the noise source may be specified or measured.

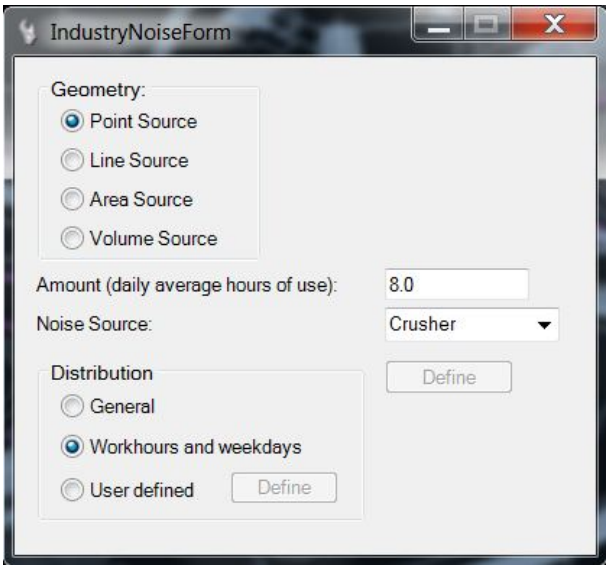


Figure 4.9: Adding an Industry Noise Source

An industry noise source is created by first entering "Noise AddNoiseSource" -> "Industry" in the Key-in. This invokes the form in figure 4.9. The options available are described in table 4.6.

After the options for the noise source have been entered, a point, line or shape is selected in MicroStation and the industry noise source is created.

Table 4.6: Industry Options

Option	***Description***
Geometry	defines what the noise source will look like. There are four choices: Point, Line, Area and Volume Source. A point source is defined by a single point and a line source is a line with two or more points. For the area source, a line or shape should be chosen and the height of the line or shape will be the height of the noise source. A volume source should be created in the same way. For a volume source the noise will be distributed over the volume below the chosen line or shape.
Amount	daily average hours of use defines how long for instance a machine runs throughout a day. This number has to be between 0 and 24. The number of hours is distributed according to the distribution described below.
Noise Source	defines a few types of industry noise. There are currently four possible types: Crusher, Drilling, Dumper or User Defined. These are defined in Table 4.7.
Distribution	this option defines how the hours are distributed over time. The distribution is created for every hour of all the days in a week and is the same for all weeks in a year. This is a simplification. There are three possible options: General, Work hours and weekdays and User Defined. These are defined in Table 4.8.

Table 4.7: Noise Source Types

Option	***Description***
Crusher	this is a large machine used for crushing larger rocks into smaller ones.
Drilling	when a drilling rig is drilling a hole in rock, a lot of noise is created. The frequency spectra for this noise source originates from the measurements mentioned in section 3.5 and shown in Appendix B.
Dumper	is a large and heavy vehicle common in many industry and construction areas.
User Defined	industry noise sources may have a great variation in noise level and frequency range and the user defined option enables an advanced user to enter specific values for the different frequencies. The form is the same as in figure 4.6.

Table 4.8: Time Distribution

Option	***Description***
General	this distributes the noise over day, evening and night, and all days in a week are the same.
Work hours and weekdays	the hours are distributed between 08 and 16 and Monday - Friday.
User Defined	if none of the two options above are suited for the specified noise source a user defined distribution may be created. The time distribution form looks like the one in figure 4.7. It is important that the number of hours input to the form matches the daily amount.

Noise Mitigation

A mound, screen or wall may be added by selecting the "Noise AddMound" function and choosing a line to be the base of the mound. The options form for a mound are in figure 4.10.

It is very important that the line chosen for the new element is based on elevation lines or other elements that are based on the terrain, to make sure no errors occur. This function is quite sensitive to intersecting elements, etc. The options for the noise mitigation elements are described in table 4.9. The radio buttons decide which of the values below may be input.

The level of the base line is changed to "Voll" - Mound in Norwegian and a line indicating the height of the element is added with the level "VollHøyde" - Mound Height.

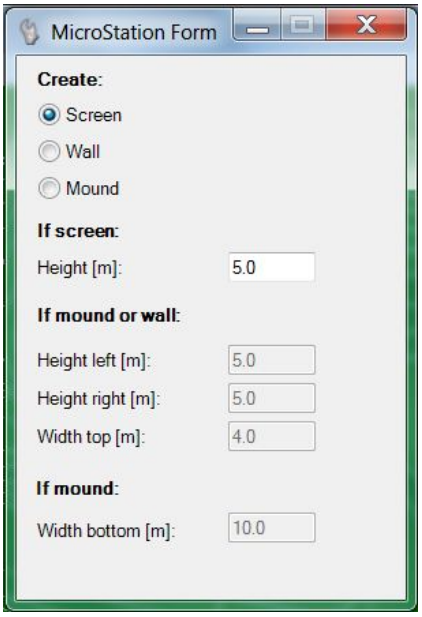


Figure 4.10: Noise Mitigation Form

Table 4.9: Noise Mitigation Options

Option	*Description*
Screen	only the height output needs to be set, the others are zero by default.
Wall	height at the left and right side of the wall and the width at the top is input. Left and right side of the wall is defined based on how the base line for the wall is drawn. A wall is actually a mound with the base being 20 cm wider than the top.
Mound	width at the top and bottom of the mound will be an input in addition to the height at the left and right side of the mound.

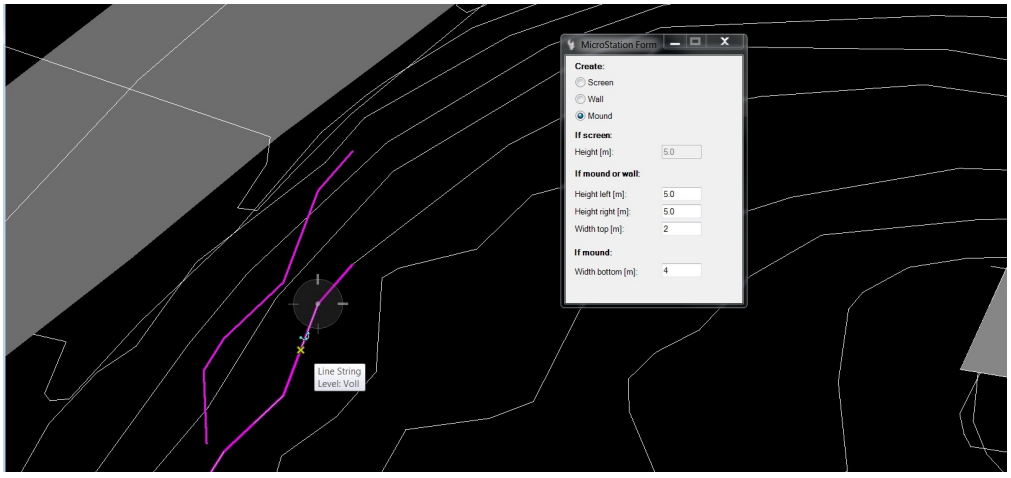


Figure 4.11: Adding a Mound, Screen or Wall

Buildings

To add a building the "Noise AddBuilding" function in the Key-in is selected.

Then a cell or a line string in MicroStation is chosen as the outline of the building. An outline of the building should be a closed line or a shape at the height of the start of the roof of the building.

Any top lines of the building will be automatically searched for and added. These have to have "Mønelinje" as Level.

Noise at facade points around the building is computed if the check box is checked. Many facade points will be added for each building, the house below will for instance have 52 facade points computed. These are only output if any of the values have a decibel value of 50 or larger.

The data provided by SOSI (Samordnet Opplegg for Stedfestet Informasjon[22]) is compatible with this function. Below is an image showing the process of adding a building.

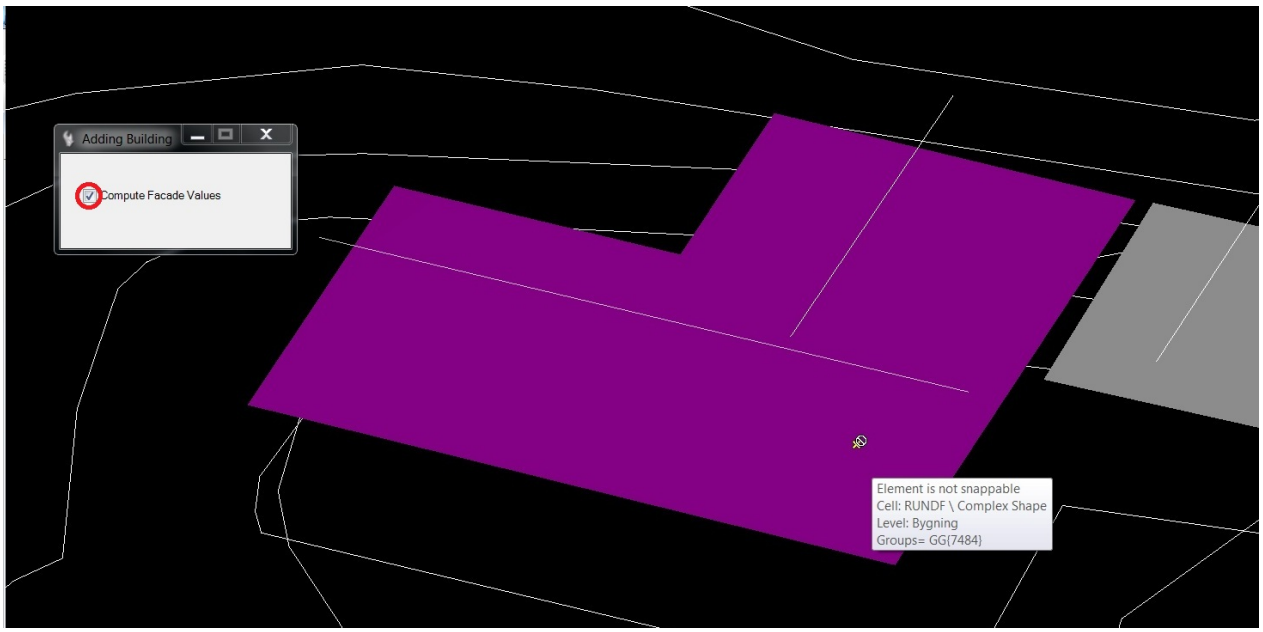


Figure 4.12: Adding a Building

Calculation points

If exact noise values are wanted for a house, the building could be added as shown above, and facade points computation selected. However, if exact values are desired for any other line, shape or solid, the "Noise AddCalculationPoints" function may be used. The function is selected the same way as the others, and the specific line, shape or solid is chosen. The noise at the points will then be calculated and the results output to a file.

The main difference between using facade and calculation points to compute noise levels is that facade points takes reflections from its building wall into account.

Running the computation

By activating "Noise RunCalculation" in the Key-in the form on the right is shown. The unit(s) to be computed may be chosen here. These units should be known for a user with some noise experience. The "Run Computation" button starts the computation process, which may take from seconds to several hours depending on how large the model is, how many noise sources there are and computer performance.

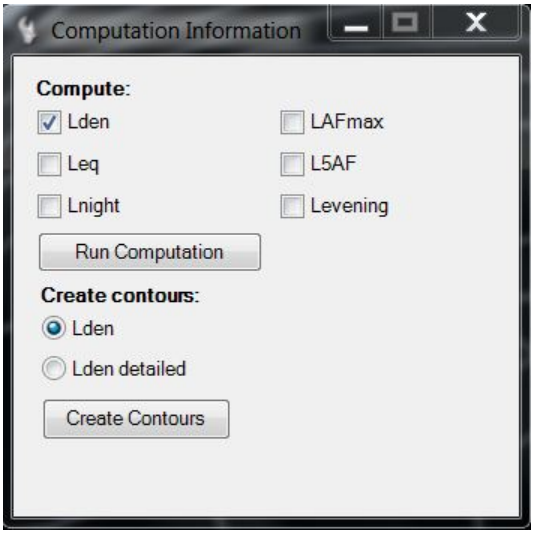


Figure 4.13: Run Computation Form

Topography Model

SoundKernel creates its own topography model based on the grid points created in sub section *Creation of grid* and the buildings, noise mitigation elements and roads added. This model is output as an X3D file and may be viewed using MeshLab[26], a free open source software. The file will look similar to the image below, depending on the added elements and the terrain.

It is recommended to check this model to see if all elements are where they should be and look like they should.

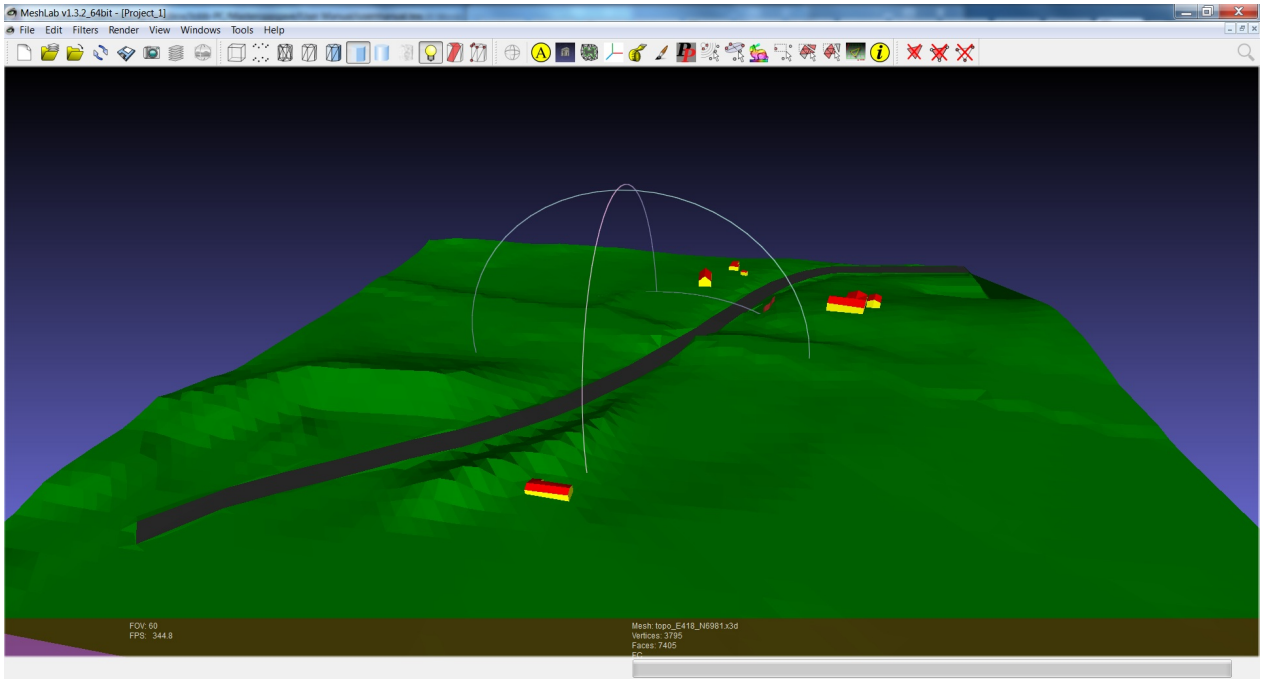


Figure 4.14: Finished Topography Model

Result File

When the computation is finished a form will pop up, showing how long the computation took. The results and details of the computation, such as point coordinates for the grid, noise sources, buildings, mounds, computation points, etc have been written to a file called Summary.txt, in the folder "C:\TEMP". More information about what is output in that file may be found below:

Table 4.10: Information in Summary.txt

Element	***Information written to Summary.txt***
Noise sources	the points of the noise sources in addition to if it is a road or industry noise source. Amount/AADT, distribution, noise type, frequency spectra and other important parameters are written as well.
Building	the inline and outline points of a building.
Noise Mitigation Element	the points of the base line, left/right heights and top and bottom width are output if relevant for the element.
Facade Point results	the points, the resulting values and computation unit.
Grid	the start and end points of the grid, in addition to number of grid lines in each direction and total number of points.
Computation points	the exact computation points, resulting values and computation unit.

Contour Lines

If the computation has been run for the computation value L_{den}, the contour lines may be drawn. Either simple or detailed contours may be chosen and upon pressing the "Create Contours" button, the contours will be drawn. The contours are color coded and look like the images in figure 4.15 and 4.16 on the following page.

Figure 4.15 shows the simple contour lines, with noise at 65 dB or higher inside the red area, and 55 dB or higher inside the yellow limit. Figure 4.16 shows detailed contour lines, where there are 3 dB between each color.

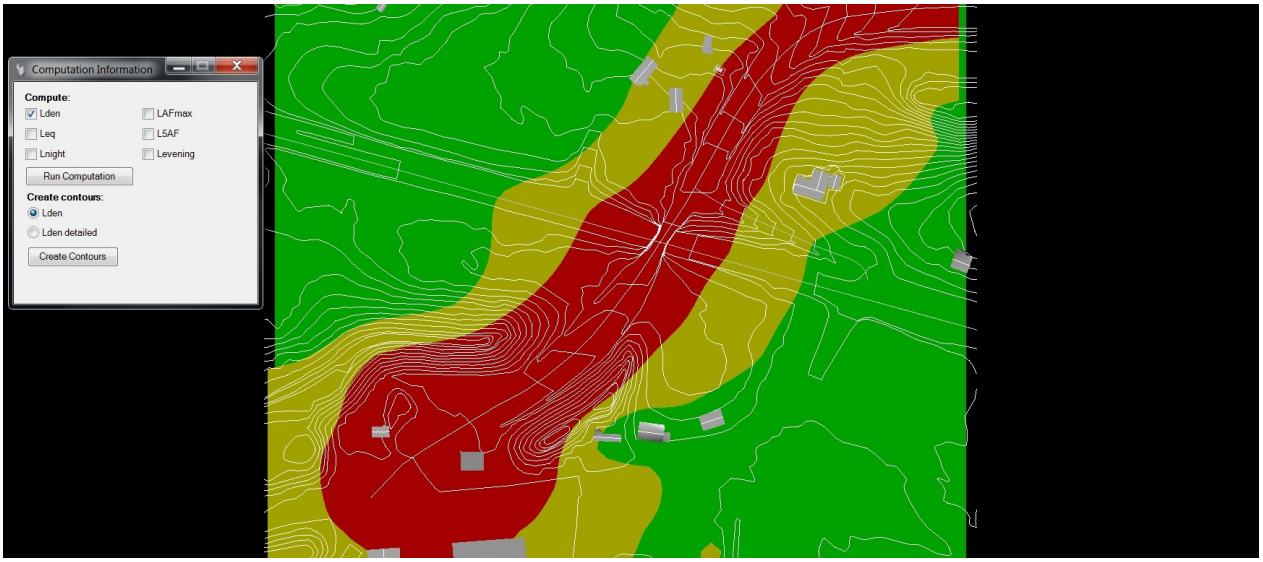


Figure 4.15: Drawn Contour Lines

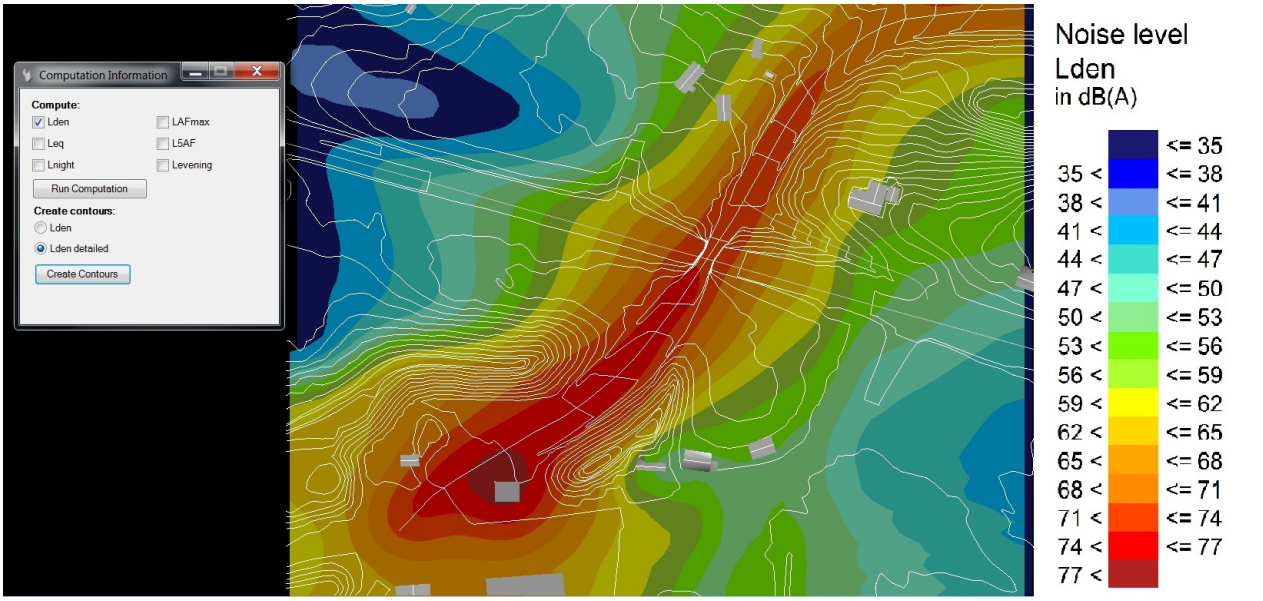
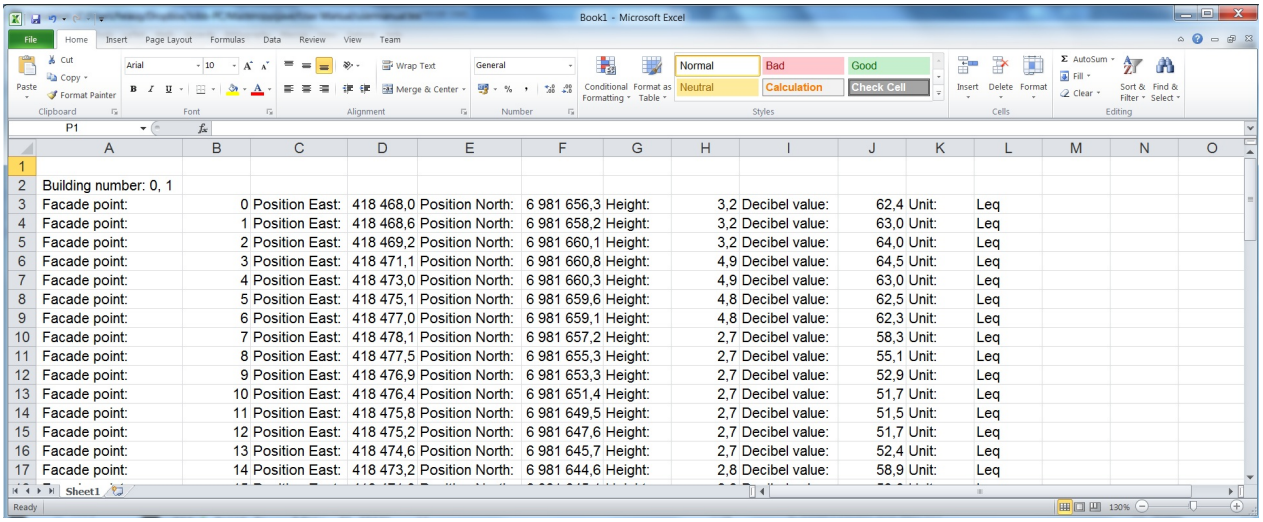


Figure 4.16: Drawn Detailed Contour Lines

Facade Points

The values for facade points are written to the summary file, and may easily be used in for instance an Excel document. The points are listed with building number, facade point number, position east and north, decibel value(s) and unit(s). Copy-pasted into Excel the values will look something like figure 4.17.



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	Building number: 0, 1											
3	Facade point:	0	Position East:	418 468,0	Position North:	6 981 656,3	Height:	3,2	Decibel value:	62,4	Unit:	Leq
4	Facade point:	1	Position East:	418 468,6	Position North:	6 981 658,2	Height:	3,2	Decibel value:	63,0	Unit:	Leq
5	Facade point:	2	Position East:	418 469,2	Position North:	6 981 660,1	Height:	3,2	Decibel value:	64,0	Unit:	Leq
6	Facade point:	3	Position East:	418 471,1	Position North:	6 981 660,8	Height:	4,9	Decibel value:	64,5	Unit:	Leq
7	Facade point:	4	Position East:	418 473,0	Position North:	6 981 660,3	Height:	4,9	Decibel value:	63,0	Unit:	Leq
8	Facade point:	5	Position East:	418 475,1	Position North:	6 981 659,6	Height:	4,8	Decibel value:	62,5	Unit:	Leq
9	Facade point:	6	Position East:	418 477,0	Position North:	6 981 659,1	Height:	4,8	Decibel value:	62,3	Unit:	Leq
10	Facade point:	7	Position East:	418 478,1	Position North:	6 981 657,2	Height:	2,7	Decibel value:	58,3	Unit:	Leq
11	Facade point:	8	Position East:	418 477,5	Position North:	6 981 655,3	Height:	2,7	Decibel value:	55,1	Unit:	Leq
12	Facade point:	9	Position East:	418 476,9	Position North:	6 981 653,3	Height:	2,7	Decibel value:	52,9	Unit:	Leq
13	Facade point:	10	Position East:	418 476,4	Position North:	6 981 651,4	Height:	2,7	Decibel value:	51,7	Unit:	Leq
14	Facade point:	11	Position East:	418 475,8	Position North:	6 981 649,5	Height:	2,7	Decibel value:	51,5	Unit:	Leq
15	Facade point:	12	Position East:	418 475,2	Position North:	6 981 647,6	Height:	2,7	Decibel value:	51,7	Unit:	Leq
16	Facade point:	13	Position East:	418 474,6	Position North:	6 981 645,7	Height:	2,7	Decibel value:	52,4	Unit:	Leq
17	Facade point:	14	Position East:	418 473,2	Position North:	6 981 644,6	Height:	2,8	Decibel value:	58,9	Unit:	Leq

Figure 4.17: Facade Points Results

Calculation Points

The calculation points are written to the same text file as the facade point results. The x, y and z-values of the points are given in addition to the point number, the units computed and the decibel values for the units. The text file may look like below:

Point 0: x: 418 495,9 y: 6 981 606,7 z: 50,4 Lden: 57,0 Lequ: 53,6
Point 1: x: 418 494,9 y: 6 981 610,9 z: 50,3 Lden: 57,3 Lequ: 53,8
Point 2: x: 418 493,9 y: 6 981 612,1 z: 50,3 Lden: 57,4 Lequ: 53,9
Point 3: x: 418 492,7 y: 6 981 613,4 z: 50,1 Lden: 57,5 Lequ: 54,1
Point 4: x: 418 491,6 y: 6 981 614,5 z: 50,1 Lden: 57,6 Lequ: 54,1

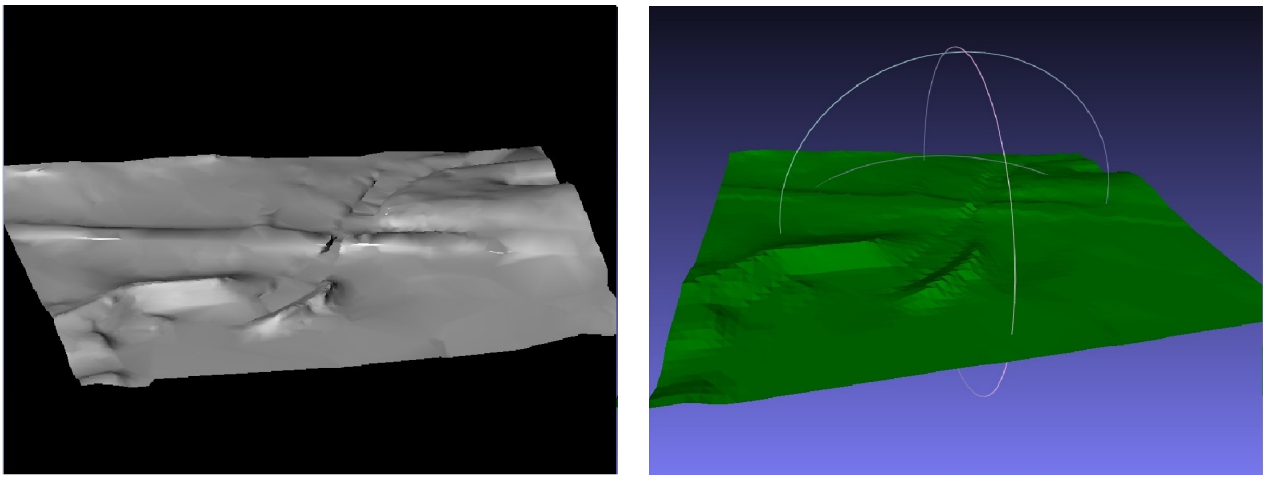
4.3 Testing and Results

In this section the functions in ISY.CAD.Noise will first be tested one by one, and then there will be two larger scale tests. All the tests are listed below:

1. Comparing mesh to topography
2. Noise Mitigation Test
3. Building and Calculation Points
4. Road Noise Source, testing traffic type, time distribution and tunnel
5. Industry Noise Source, testing point, line, area and volume source
6. SINTEF Test Case
7. DNV Test Case

Grid and Terrain Model

First the grid should be verified. The grid used in SoundKernel should be similar to the actual terrain model. This is easily tested by running the computation without any noise sources, buildings or mounds. The results are shown below:



(a) MicroStation mesh (b) SoundKernel Topography model

Figure 4.18: Comparing MicroStation mesh to Topography model

The two models are very similar, the topography model is a bit more rough, however it is still a good approximation for the mesh.

Noise Mitigation

A mound, screen or wall should decrease the noise on the side facing away from the noise source. This will be tested by running a computation with a road noise source twice, once with a screen and once without. The screen will have a height of 5 m.

After the computation has run the mound, screen or wall should be included in the SoundKernel topography model. To the right is the topography model with the 5 m high screen.

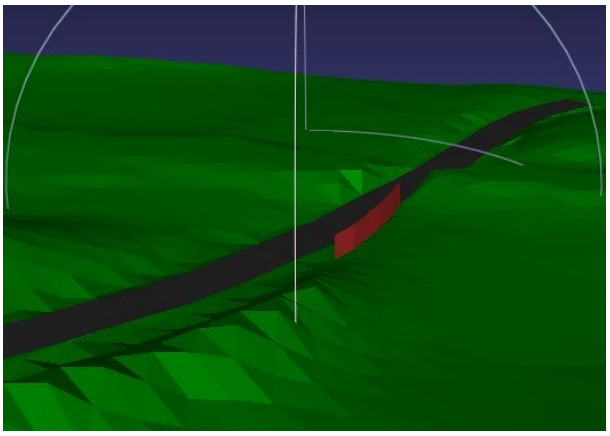
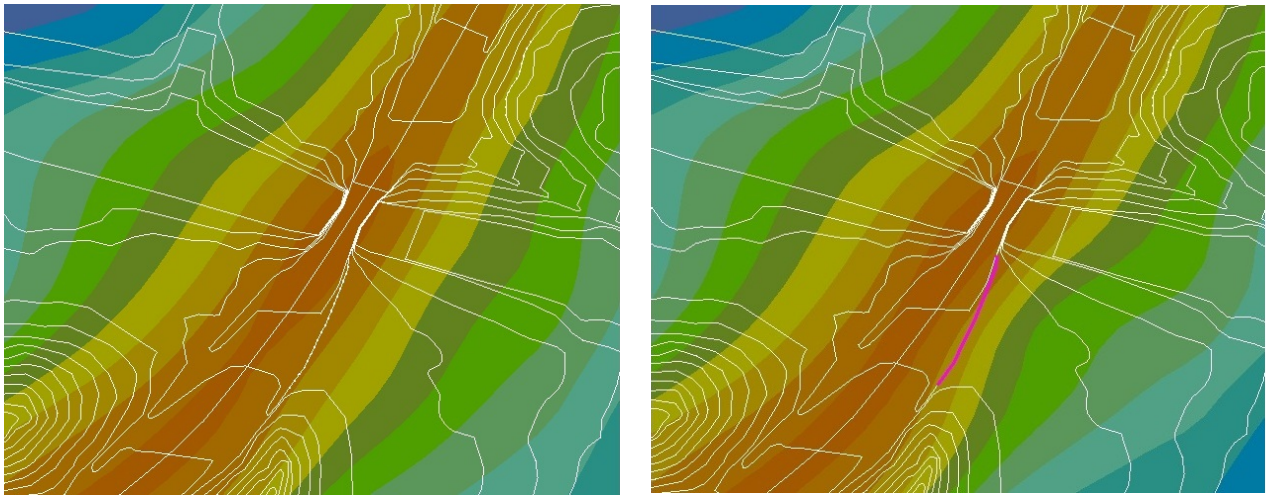


Figure 4.19: Screen in Topography model

The resulting contour noise lines for the two computations mentioned above are shown in figure 4.20:



(a) Without screen

(b) With screen

Figure 4.20: Test results for added screen

The screen is drawn in pink. The contour lines are drawn with 3 dB between each.

The effect of the mound is noticeable, the noise levels are lower on the right side of the screen in (b). Notice the wide green areas on the right side of the screen. If there was a house here, the screen would probably decrease the outside noise level by 3-6 dB.

Note that the contour lines are computed 4 m off the terrain level and that the screen might have a larger impact on sound levels closer to the ground. These results correspond well with the values mentioned in section 3.2.

Building and Calculation Points

A computation was run for a fairly noisy road, a building with facade points and computation points. The computation points were the outline of the building.

The building is shown in figure 4.21, and the six points of the outline are numbered. The noise source is to the left of the building. An extract from the results may be viewed in table 4.11.

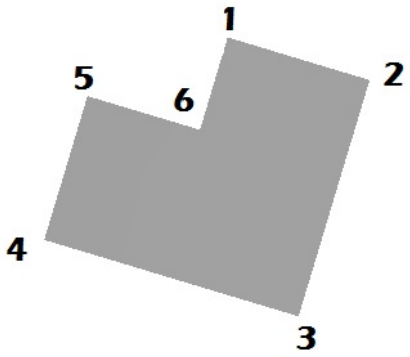


Figure 4.21: Building with Points

Table 4.11: Facade and Computation Point Results

Point	***Type***	***X [m]***	***Y [m]***	***Height [m]***	L_{eq}***[dB]***
1	Computation	418 469.9	6 981 660.7	3.5	49.0
	Facade	418 471.1	6 981 660.8	4.1	48.9
	Facade	418 471.1	6 981 660.0	1.5	44.7
3	Computation	418 473.9	6 981 644.9	3.5	40.2
	Facade	418 473.2	6 981 644.6	2.5	43.0
	Facade	418 473.2	6 981 644.6	1.5	40.8
4	Computation	418 459.5	6 981 649.3	3.5	43.2
	Facade	418 459.3	6 981 650.5	4.1	50.1
	Facade	418 459.3	6 981 650.5	1.5	47.4
6	Computation	418 468.3	6 981 655.5	3.5	49.7
	Facade	418 468.0	6 981 656.3	2.5	44.8
	Facade	418 468.0	6 981 656.3	1.5	43.1

For point 1 and 3 the results are similar, while for point 4 and 6 there is a larger difference between the results. For point 4 the facade values are the largest, while for point 6 they are the smallest.

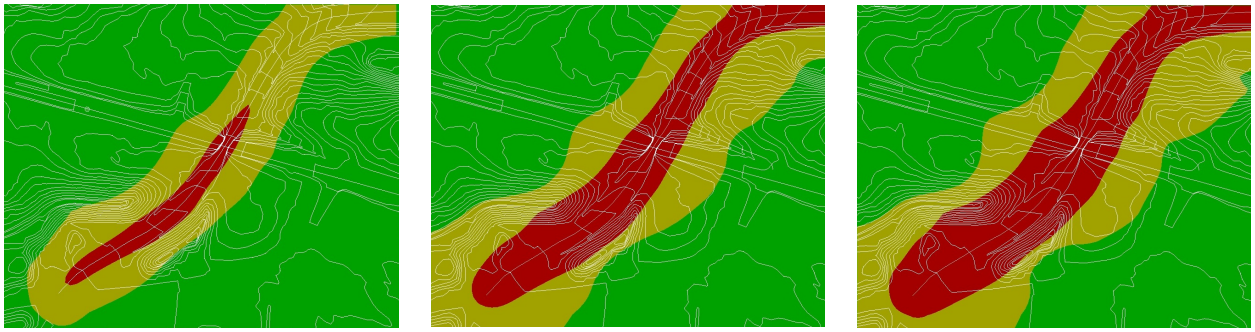
There is a very good explanation for this: The facade points are computed with reflections included. This means that points on the back of the building and inside corners will get less noise, such as for point 6. Point 4 is directly exposed to the noise source and will probably have a lot of reflection on the wall, causing it to have more noise than the corresponding computatioln point.

The computation points does not take into account that there is a building there and will generally give more even values for the area. In addition, the height of the point has a large influence on the noise level, generally the noise is lower when the point is closer to the ground.

Road Noise Source

There are many ways to verify and test the road noise sources, the most important tests will be performed in the *SINTEF Test Case* and *DNV Test Case* sub sections. In this section a few simple roads will be tested.

Traffic Type
First, three computations will be run, one for Normal traffic type, one for Heavy and one for Dumper, results are shown below.



(a) Normal (b) Heavy (c) Dumper

Figure 4.22: Test results for different traffic types

As expected, the Dumper traffic type creates more noise than the Heavy and Normal types.

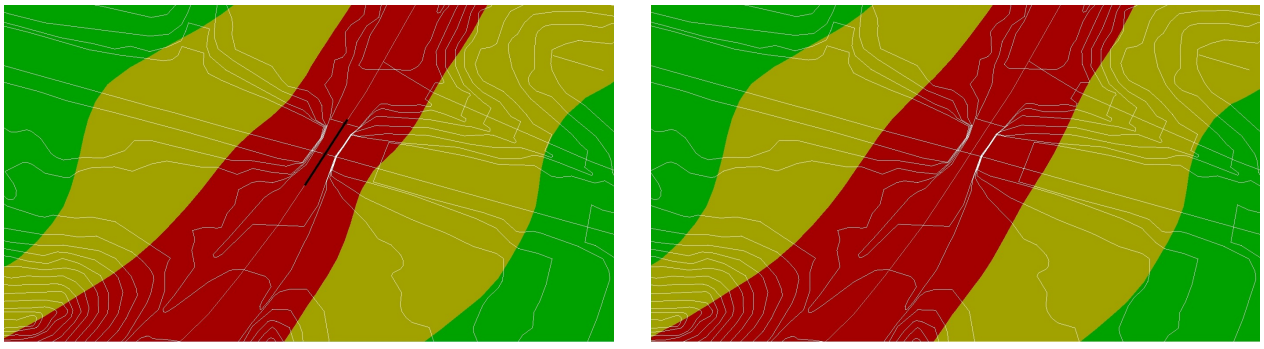
Time Distribution
The time distribution will have different effects on the results for the various computation units. For the general traffic distribution, L_{night} and $L_{evening}$ should not be zero, and L_{den} should be higher than L_{eq}. For the Weekday and work hours distribution L_{night} and $L_{evening}$ should be zero and L_{den} should have the same value as L_{eq}. Two tests with the two different distributions were run to check this. The results for the computation points chosen are shown below.

Table 4.12: Results for computation with different Traffic Distributions

Point	L_{den}	L_{eq}	L_{night}	$L_{evening}$
General traffic distribution				
0	60.3	56.8	51.6	56.8
1	60.4	56.9	51.7	56.9
2	60.5	57.0	51.8	57.0
Workday and work hours traffic distribution				
0	52.5	52.5	-50.0	52.5
1	52.3	52.3	-50.0	52.3
2	52.4	52.4	-50.0	52.4

The results are as expected with the exception of $L_{evening}$. For Workday and work hours distribution the expected value is -50.0, which essentially means that there is no value. For General distribution it should not be the same as L_{eq}. However, as L_{den} and L_{eq} are the same, they have not been affected by the $L_{evening}$ value. The reason for the wrong $L_{evening}$ values is that it has not yet been implemented in SoundKernel.

Tunnel
For a tunnel the noise is lower along the tunnel, but possibly higher at the ends of the tunnel. This is a result of noise accumulating inside the tunnel. A test was run for a Heavy traffic distribution and the results are shown in figure 4.23.



(a) Road with tunnel (b) Road without tunnel

Figure 4.23: Test results for Tunnel

The black line in (a) shows the part of the road defined as a tunnel. It is easy to see that the noise is reduced where the tunnel is, and quickly goes back to a level similar to (b) where the tunnel ends.

Industry Noise Source

Point Source
This noise source is concentrated in a single point, and should have circular contour lines radiating from it. The image below shows the resulting contour lines for a point source centered in the red circle. As expected the noise radiates in a circle close to the source. When the distance from the noise source increases, the terrain makes the noise distribute differently.

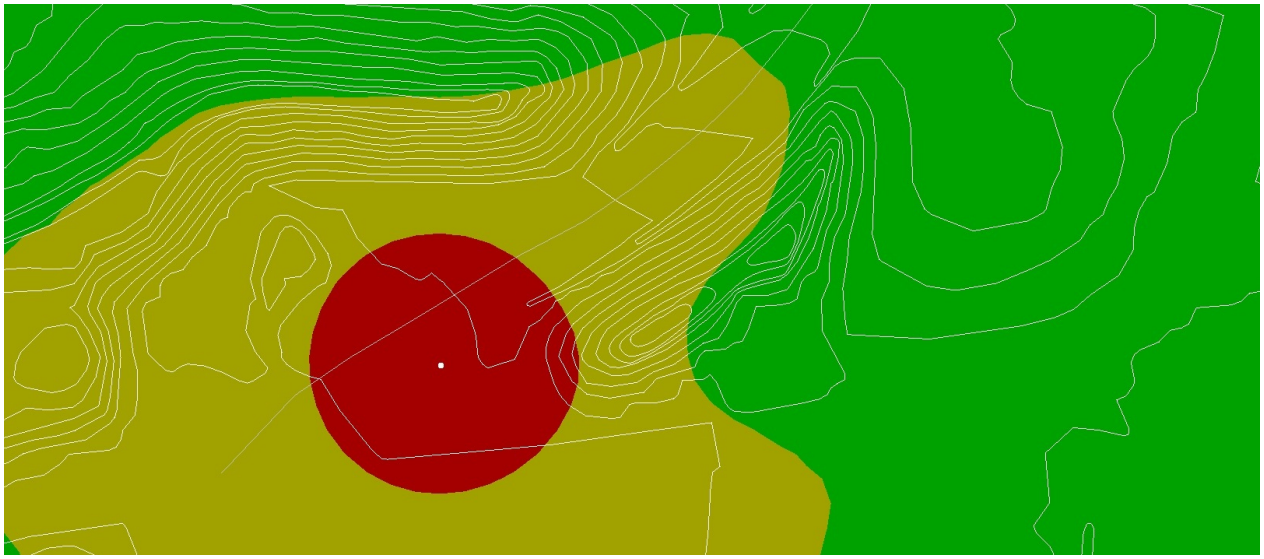
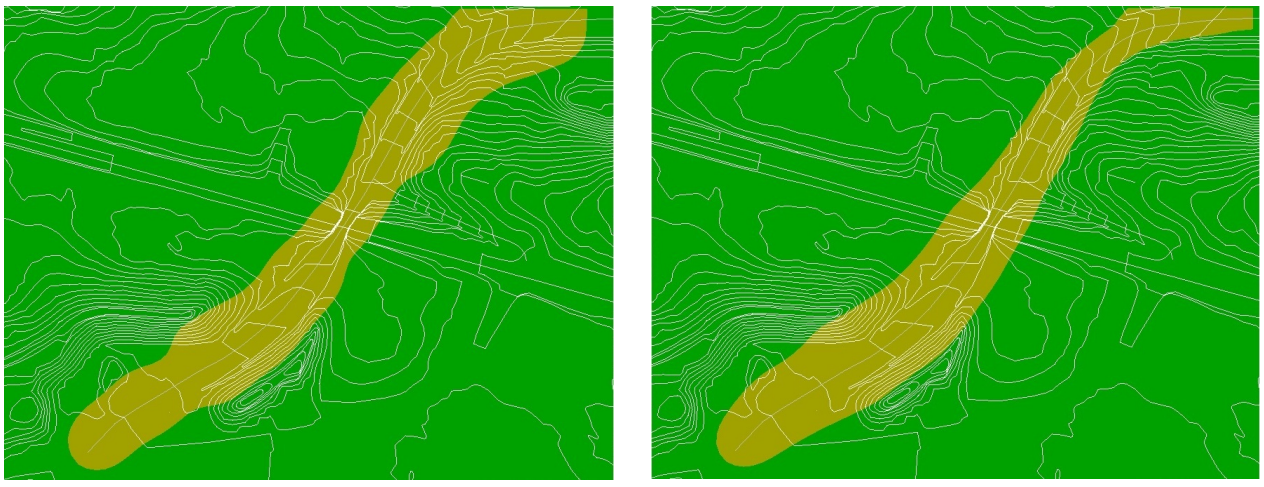


Figure 4.24: Test results for Point industry source

Line Source
For a line source, the noise is dispersed over the length of the line. A dumper line noise source should therefore be similar to a road noise source with dumper traffic type. Two computations were run, one for a road noise source of Dumper traffic type, with 200 AADT. The second was an industry noise source with line geometry, Dumper noise type and an amount of 8 hours. Both had Workday and work hours distribution. The results are shown in figure 4.25.



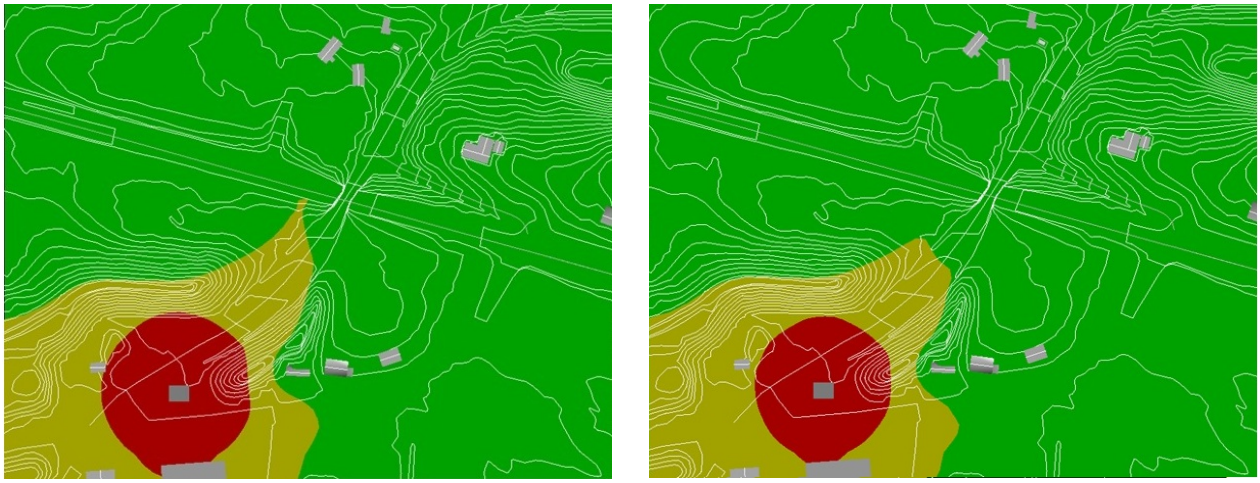
(a) Industry (b) Road

Figure 4.25: Comparison of Industry and Road Dumper Noise Source

The contour lines are similar, however there are some differences. The major difference is that the industry noise source is defined at terrain height, while the road source has heights at terrain level and at 75 cm above terrain level. This gives approximately the same noise levels, but slightly different distribution of noise.

Area and Volume Source
For an area source the noise is based at the height of the area, and noise should be radiating out from it.

For a volume source the noise is distributed from the height of the area to the terrain model. It will therefore be more dispersed than an area source. A comparison between an area and a volume source is shown below. These have been created based on the same shape, noise type and distribution type.



(a) Area (b) Volume

Figure 4.26: Area and Volume Noise Source

These are very similar, as they have the same noise and distribution type, and are not too high off ground level. It is noticable that the area noise source creates more noise. This is, as mentioned before, a result of the volume noise source being dispersed.

SINTEF Test Case

An example program was created, which included a grid, a road, a building and some calculation points. The base height of these elements are set to zero to simplify the computation. Details about the elements are shown below:

Table 4.13: SINTEF Test Information

Element	***Description***
Terrain model	is based on the previously used model of Visnes Kalk AS. The model is approximately 335 * 275 m, and the z-values were set to 0 for all grid points giving an even terrain.
Building	one building was added, with 7 outline points and two top lines. The outline had a height of 3.5 m while the top lines had a height of 5.1 m. Facade point computation was chosen for the building.
Road Noise Source	a Dumper traffic type road was created, with width of 10 m, velocity 50 km/h and 200 AADT. Workdays and work hours traffic distribution was chosen.
Computation Points	three lines were chosen and their points were the computation points for this test. The first line is parallel to the road and quite close to it, and at terrain height. The second line is the same as the first, except that it is 4 m off terrain height. The third line is orthogonal to the road. The lines, building and road are drawn in figure 4.30.

The topography model looks like the image below:

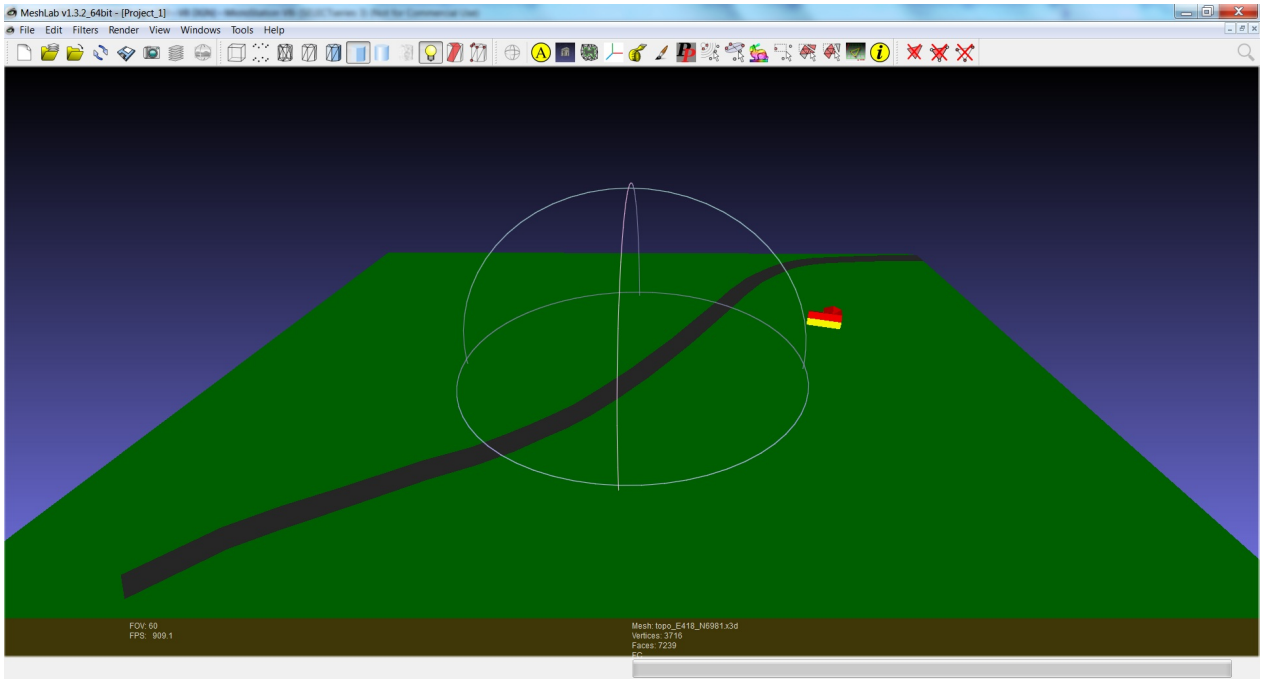


Figure 4.27: Topography for SINTEF Verification

Comparing results: Contour lines

The contour lines created by SINTEF's code look like the image below.

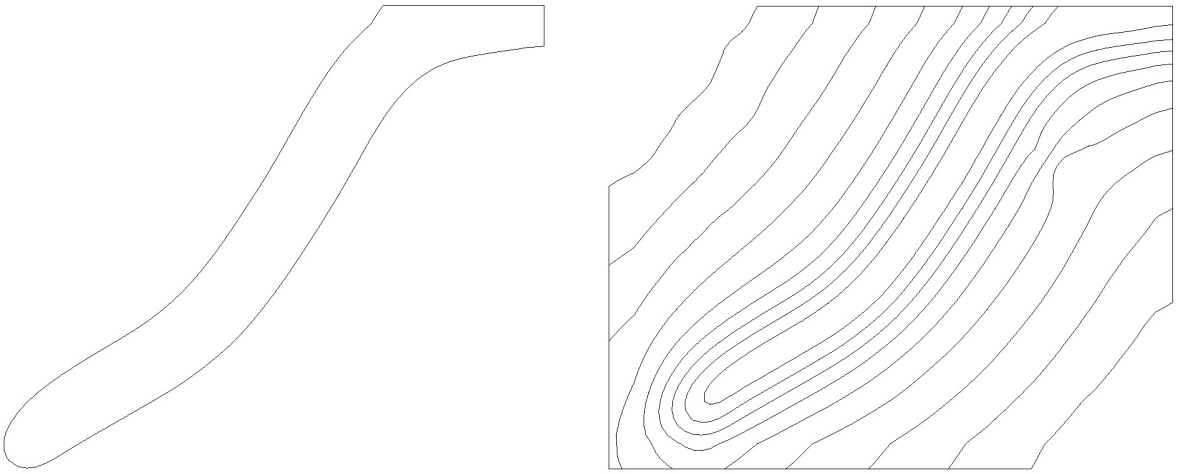


Figure 4.28: SINTEF Contour Lines

SoundKernel is assumed to be thoroughly tested and reliable and produces correct contour lines. By comparing figure 4.28 to the contour lines created by code from ISY.CAD.Noise, in figure 4.29, it is easy to see that these are correct. The highest contour line for the detailed contours are at 59 dB.

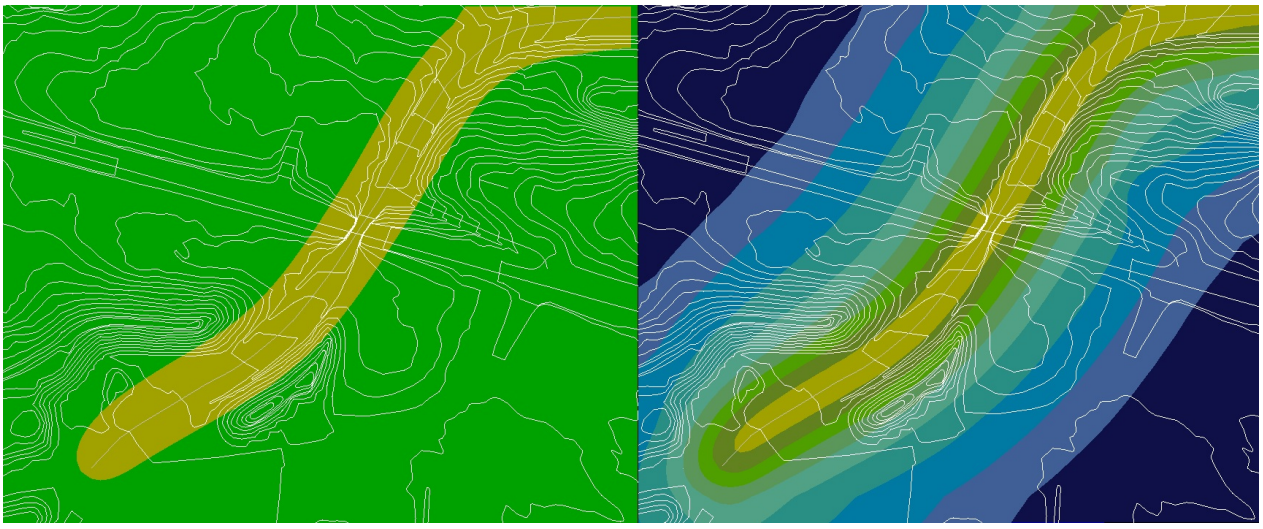


Figure 4.29: ISY.CAD.Noise Contour Lines

Comparing results: Computation and Facade points

SINTEF ICT, Acoustics ran a test based on values from the program in Appendix C and got values around 51-52 dB for line 1 and 58-59 dB for line 2.

The computation point results from the test program are shown in table 4.14, while the first 40 facade point results are shown in table 4.15.

Table 4.14: Computation Point Results

Point	***x***	***y***	***z***	L_{den} ***[dB]***
0	418 440.0	6 981 677.9	0	51.9
1	418 437.4	6 981 674.1	0	52.4
2	418 434.8	6 981 670.2	0	52.7
3	418 432.5	6 981 666.0	0	52.6
4	418 429.8	6 981 660.2	0	52.1
5	418 427.2	6 981 654.5	0	51.6
6	418 440.0	6 981 677.9	4	58.3
7	418 437.4	6 981 674.1	4	58.5
8	418 434.8	6 981 670.2	4	58.7
9	418 432.5	6 981 666.0	4	58.6
10	418 429.8	6 981 660.2	4	58.4
11	418 427.2	6 981 654.5	4	58.1
12	418 495.9	6 981 606.7	0	29.9
13	418 494.9	6 981 610.9	0	30.2
14	418 493.9	6 981 612.1	0	30.5
15	418 492.7	6 981 613.4	0	30.7
16	418 491.6	6 981 614.5	0	30.7
17	418 490.3	6 981 615.4	0	30.9
18	418 486.4	6 981 617.3	0	31.8
19	418 471.8	6 981 622.8	0	33.8
20	418 446.1	6 981 634.0	0	39.7
21	418 426.3	6 981 642.7	0	47.9
22	418 417.3	6 981 648.1	0	59.7
23	418 412.0	6 981 651.5	0	72.1

Table 4.15: Facade Results

Facade point	***Position East***	***Position North***	***Height***	L_{den} ***[dB]***
0	418 468,0	6 981 656,3	2,5	43,7
1	418 468,6	6 981 658,2	2,5	44,7
2	418 469,2	6 981 660,1	2,5	45,4
3	418 471,1	6 981 660,8	4,1	46,8
4	418 473,0	6 981 660,3	4,1	45,5
5	418 475,1	6 981 659,6	4,1	44,7
6	418 477,0	6 981 659,1	4,1	44,4
7	418 478,1	6 981 657,2	2,5	37,2
8	418 477,5	6 981 655,3	2,5	34,6
9	418 476,9	6 981 653,3	2,5	33,7
10	418 476,4	6 981 651,4	2,5	33,2
11	418 475,8	6 981 649,5	2,5	32,6
12	418 475,2	6 981 647,6	2,5	32,5
13	418 474,6	6 981 645,7	2,5	32,4
14	418 473,2	6 981 644,6	2,5	39,9
15	418 471,3	6 981 645,1	2,5	40,7
16	418 469,4	6 981 645,7	2,5	40,5
17	418 467,5	6 981 646,3	2,5	41,5
18	418 465,6	6 981 646,9	2,5	41,9
19	418 463,7	6 981 647,5	2,5	42,4
20	418 461,8	6 981 648,1	2,5	41,7
21	418 459,8	6 981 648,6	2,5	45,0
22	418 459,3	6 981 650,5	4,1	47,7
23	418 459,9	6 981 652,4	4,1	47,9
24	418 460,6	6 981 654,5	4,1	48,1
25	418 461,2	6 981 656,4	4,1	48,3
26	418 463,4	6 981 657,5	2,5	45,3
27	418 465,3	6 981 656,9	2,5	44,5
28	418 468,0	6 981 656,3	1,5	41,9
29	418 468,6	6 981 658,2	1,5	42,9
30	418 469,2	6 981 660,1	1,5	43,6
31	418 471,1	6 981 660,8	1,5	42,9
32	418 473,0	6 981 660,3	1,5	41,9
33	418 475,1	6 981 659,6	1,5	41,3
34	418 477,0	6 981 659,1	1,5	41,1
35	418 478,1	6 981 657,2	1,5	34,7
36	418 477,5	6 981 655,3	1,5	32,0
37	418 476,9	6 981 653,3	1,5	31,0
38	418 476,4	6 981 651,4	1,5	30,6
39	418 475,8	6 981 649,5	1,5	30,6
40	418 475,2	6 981 647,6	1,5	30,2

Line 1 consist of points 0-5 at height 0 m while line 2 consist of points 6-11 at height 4 m, line 2 are directly on top of line 1. Line 3 consists of points 12-23. The lines, point numbers, building and contour lines are shown below:

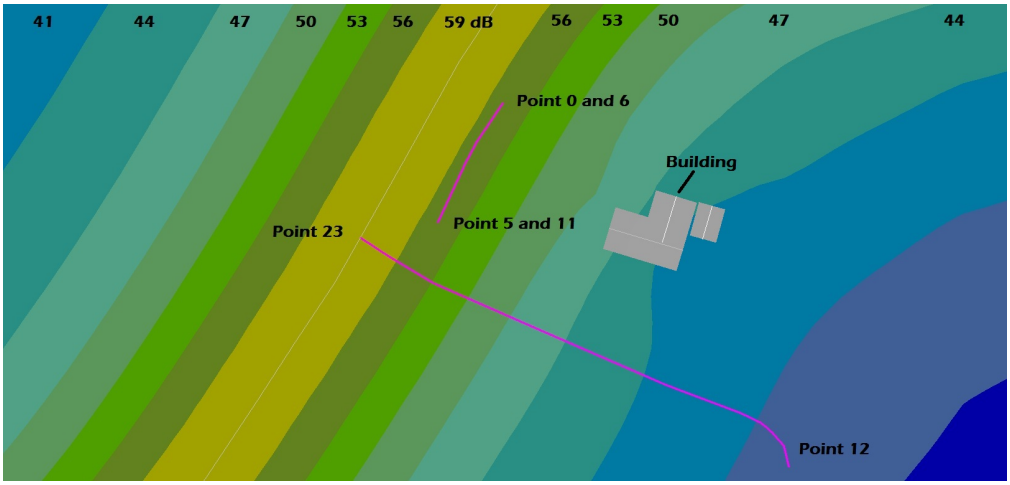


Figure 4.30: Facade, Computation Points and Contour Results

The results in table 4.14 and 4.15 are very good. The following table explains why this test verifies the results:

Table 4.16: Verification of Computation Results

Element	***Verification***
Line 1	has values corresponding to the SINTEF results as they are all between 51.0 and 52.9 dB. They are inside a contour line with values from 56-59 dB. These values are higher because they have been computed at a height of 4 meters.
Line 2	has values between 58.0 and 58.9 dB and are within the SINTEF results. They are computed at a height of 4m and fit nicely within the contour line which has values from 56-59 dB.
Line 3	has values from 29.9 dB to 72.1 dB. The point closest to the road (point 23), are actually on the road, and it makes sense that it is that high, especially since the road is defined at terrain height. As the points are further away from the noise source, the noise levels drop. These are lower than the contour line values because the computation points are at terrain height while the contours are computed at 4m height. For line 1 and 2 the difference was approximately 7 dB, if this assumption is used on point 12, which has a value of 29.9 dB, the result would be approximately 37 dB. The point is in a contour with a value of 38 dB or higher, which is close to the estimated value.
Facade point	values range from 30 dB to 50 dB. The building is within contour lines ranging from 41 to below 50 dB. As mentioned in sub section *Building and Calculation Points*, the facade values vary depending on where on the building the points are. The important thing to look at when verifying the facade points is that they are within a reasonable range, and that points higher off the ground have a larger value than corresponding lower points. These points satisfy both terms.

DNV Test Case

To test ISY.CAD.Noise against the DNV noise map[5], an altered model of Visnes Kalk AS was used. This model was provided by Erik Ludvigsen at NTNU, Department of Geology and Mineral Resources Engineering, and was used by DNV when they did their computation. This a very detailed and accurate model.

As there are no detailed information about how DNV did their computation, assumptions and simplifications have to be made. One major simplification is that only a part of the mine is taken into account. If the whole mine had been used, the testing would become very detailed and time consuming. A critical part of the mine has been chosen. This area has traffic noise, loading and tipping activity and is shown in figure 4.31.

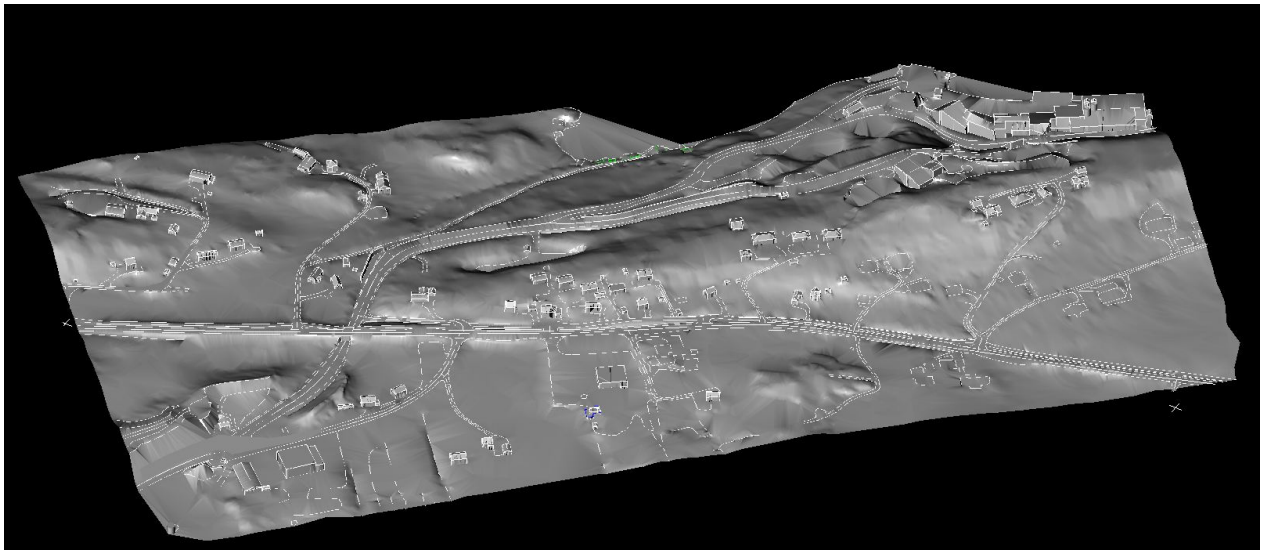


Figure 4.31: Model used when comparing with DNV

The model is very detailed, so no additional noise mitigation elements were needed. Buildings had already been added to the mesh in MicroStation, and since facade values are not needed no additional buildings were added.

Two test cases were used:

1. The roads had an industry noise line source with dumper noise type. There were three of these with 8 hour workday and work hours distribution added. In addition industry noise point and area sources were added at tipping and loading areas. These had a user defined tipping noise source from Appendix B, and 8 hours workday and work hours distribution.

2. Only the roads had defined noise sources. These were road noise sources with dumper traffic type, 50 km/h velocity and 10 m width. The distribution was workday and work hours with 2500 AADT.

These two cases were only educated guesses and will probably not give exactly the same results as the DNV mapping. However, the point of this test is not to reproduce the DNV mapping, but to show that a similar mapping with similar results may be performed using ISY.CAD.Noise.

The relevant area of the noise map for condition 1 of the DNV noise mapping is shown in figure 4.32.

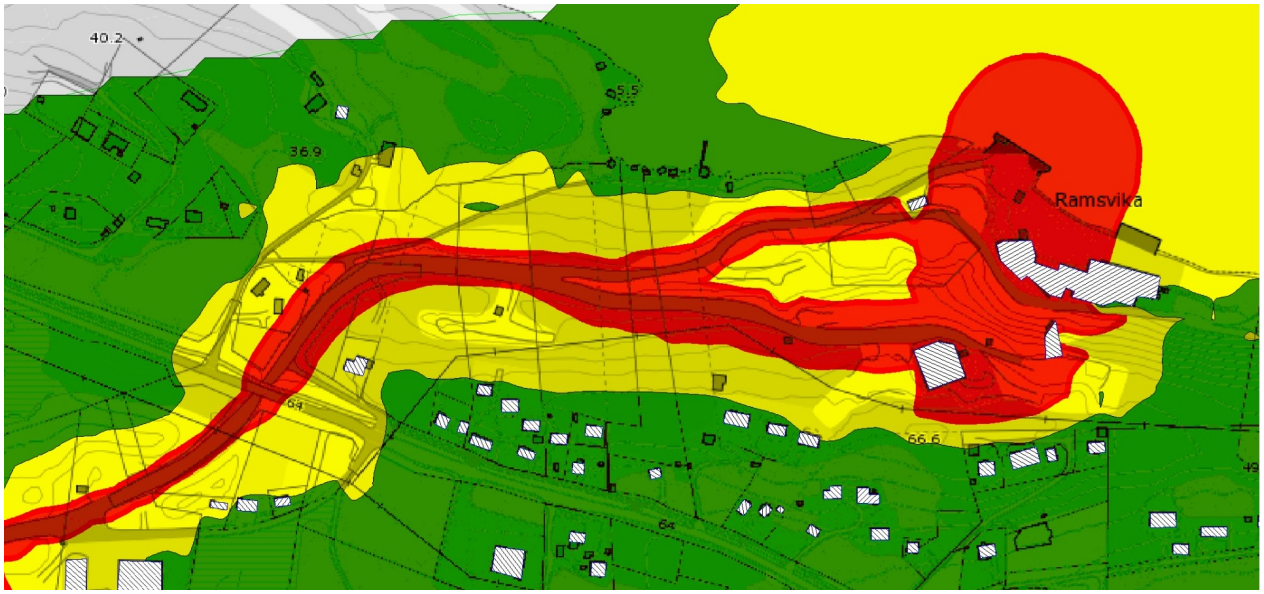


Figure 4.32: Contour lines from DNV report

Test case 1 focus mainly on the port and tipping area at the northeast part of the mine, results are shown in figure 4.33. By comparing that area to figure 4.32 one might see that the results are similar, but not exactly the same. The noise sources in the DNV mapping are lounder than the test case, and the effect of the water is more visible. If functionality for more detailed ground type information is implemented in ISY.CAD.Noise (mentioned in section 5.3), the water could be classified according to how it reflects sound, and the results might be more similar.

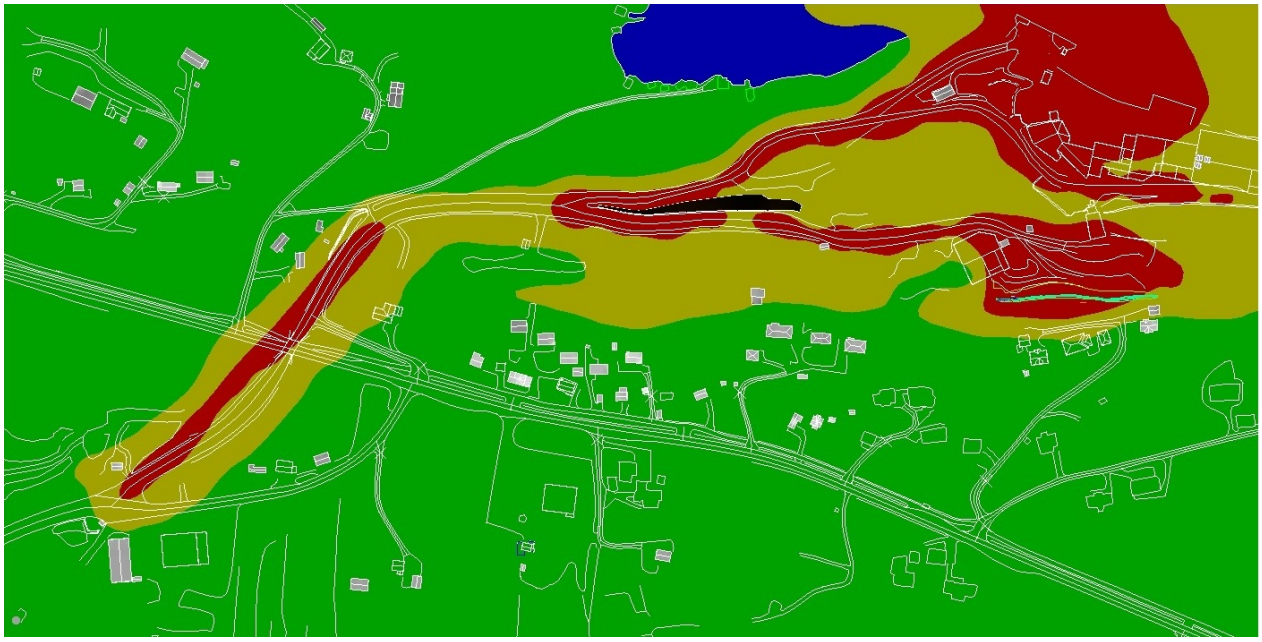


Figure 4.33: Test results for case 1

Test case 2 focus more on the road of the mine and results are shown in figure 4.34. By comparison the noise level is quite similar, however the distribution is a bit different. The noise source used is road, and the difference in distribution is probably because it is not an industry noise source. By referring to figure 4.25 it is easy to see that the industry noise source has a distribution more similar to the DNV mapping.

Figure 4.34: Test results for case 2

The results for test case 1 and 2 are not exactly the same as the results from the DNV mapping, however there are many similarities. As it is not possible to know which noise sources DNV used where, and how they were distributed it would be impossible to exactly reproduce their mapping. The results for the ISY.CAD.Noise are quite good considering the circumstances and could definitely be used for a new mapping of Visnes Kalk AS.

Two additional notes were discovered during the testing:

1. It seems that DNV used quite high values for the noise mapping. For instance for the road noise source in test case 2, the AADT values in ISY.CAD.Noise had to be as high as 2500 vehicles to approximate the results. Representatives from Visnes Kalk AS has estimated an AADT of 200-300 vehicles. Without knowing the details of the DNV noise mapping, it is difficult to know exactly how the noise sources were defined, so it is impossible to conclude anything.

2. Computation time increased quite drastically for the larger model and especially for many road noise sources. A few tests were run before the two test cases above were established, and one of these included several industry noise line sources with dumper type on the roads. The computation time doubled for two dumper sources on a road compared to just one. These computations are done in SoundKernel, however it is recommended to see if there is a way to improve the computation time.

5 Discussion and Conclusion

5.1 Discussion

NorStøy is a software application for the computation of traffic noise in Norway, created by SINTEF, Triona and the Norwegian Public Roads Administration[28]. Norstøy and ISY.CAD.Noise both use SoundKernel, which is assumed to be thoroughly tested and reliable.

Most of the features mentioned in Scope have been successfully implemented:

Table 5.1: Features in ISY.CAD.Noise

Element	***Discussion***
Noise road	was a minimum requirement and is fully implemented with choices for traffic type and distribution in addition to user defined values.
Noise map	this was another minimum requirement and is fully implemented with two types of contour lines: basic lines with red and yellow limits, and detailed lines with 3 dB between each line.
Grid	have been implemented and is based on a terrain mesh created from elevation lines.
Noise sources	several specific noise sources were mentioned, including drilling rig, hammer, industry noise, loading/unloading of freighters and point noise sources. Some of these have been implemented in the industry noise function. This function support point, line, area and volume sources and user defined frequency spectra. The noise sources mentioned in section 1.2 may be used if the user has the corresponding frequency spectra.
Noise mitigation	screen, wall and mound elements have been implemented.
Exact values	for selected elements may be computed by using the calculation point function. The building function gives exact values for facade points on the building wall.

5.2 Conclusion

The main product of this thesis is the ISY.CAD.Noise plug-in for MicroStation and the NoiseComputation DLL. Neither of these are 100 % finished and ready for professional use, however, successful connections to both SoundKernel and MicroStation have been created. Not only have the minimum requirements been met, but additional functions have also been created.The plug-in works well and gives good results.

This report and a user manual for ISY.CAD.Noise are additional results of this thesis. These explain the theoretical background of noise, how the plug-in is used, and to some extent how it works and how the results may be interpreted.

As shown in section 4.3, the test results for the functions in ISY.CAD.Noise are satisfactory, the grid is fairly accurate, the computation and facade points correspond well with the contour lines, and the road and industry noise sources behave as expected.

However, there are still many things mentioned in Further Work, that need to be done before ISY.CAD.Noise may be used professionally. The most important is further error checking and handling as this is absolutely necessary for an application.

5.3 Further Work

ISY.CAD.Noise is not a completely finished plug-in, there are still several things that need to be done before it may be distributed professionally. These are listed in the two forms below:

Table 5.2: Essential further work

Element	***Description***
Installation	An installation routine should be created, to make ISY.CAD.Noise easily accessible.
DLL management	As of now, the SoundKernel DLL files, NoiseComputation.dll and ISY.CAD.Noise.dll have to be in the folder of the MicroStation startup file. This is inconvenient and untidy, and all the DLLs required for the plug-in should be in a separate folder.
Saving and loading	Every time a MicroStation model is closed, all the buildings, noise sources, etc are deleted. There should be a save function, to enable the user to save all progress, and to load the elements when opening a model. With this function, the user would not have to start from scratch every time he/she opens a MicroStation model.
Deleting and editing	If a wrong line is selected to be a noise source, a mound, etc, there should be a way of un-doing the selection or deleting the element. A function for deleting and editing elements would be very helpful if the user wants to change the environment.
Testing and verification	The methods in both ISY.CAD.Noise and NoiseComputation should undergo thorough testing, to make sure there are no hidden errors. Some tests have been performed in section 4.3, however these are insufficient for a professional product.
Error handling	There should be added more error and exception handling, to make sure that the user gets proper information in case the program encounters any problems. In addition it is necessary to make sure that no invalid values are used in the computation or the set-up of the environment. The plug-in as of now assumes an intelligent user, and would for instance fail if a letter is input where a number should be.
Buttons	There should be a row of buttons that the user of MicroStation may use instead of the Key-in. These should be simple and clear, and there should be a button for each function in ISY.CAD.Noise. This will increase the usability of the program.

Table 5.3: Additional further work

Element	***Description***
Save Frequency Spectra	A user might have many noise frequency spectra for different vehicles or machines, relevant only to that user or company. It should be possible to create a user defined database of noise sources and their frequency spectra so that the user may select these from the road and industry noise forms.
Import measured values	When noise measurements are made, a lot of variables are created. It might be useful to create an input and convert function where the user can enter raw measurement data. Sound power levels may be calculated from these data and added to the database of saved frequency spectra. This would greatly simplify the process of going from noise measurements to doing a noise mapping.
User Manual	There are many values for the grid, road, noise sources and mounds that are set to default values in the code. It might be useful to create functions to set some of these. More information may be found in the ISY.CAD.Noise User Manual[23].
Ground Types and Forest	There are different ground types that may be used for the grid points. There are more information about these in the ISY.CAD.Noise User Manual[23]. The ground types may influence how the noise is distributed over the terrain, and a function to set specific ground types for areas with trees, water, etc could be useful.
Impulse noise	Impulse noise are not supported in SoundKernel, however it might be a useful feature in new versions. Many industries have impulse noise and have to include it in their noise maps.
Computation Time	The run time for the computation should be checked for different noise sources and models, and an attempt to optimize the run time could be made. As the computation is performed in SoundKernel, it is not necessarily possible to do a lot about this, however there is usually always room for speed improvements in any code.

Bibliography

[1] **WHO and European Commission**, *Burden of disease from environmental noise*, Quantification of healthy life years lost in Europe. Published by World Health Organization(WHO), 2011. Link: `http://www.euro.who.int/__data/assets/pdf_file/0008/136466/e94888.pdf`, accessed 24.04.2013.

[2] **Aasvang, G.M.**, *Beregning av helsebelastning som skyldes vegtrafikkstøy i Norge.*, Published (in Norwegian) by the Norwegian Institute of Public Health in Oslo, 2012. Link: `http://www.fhi.no/dokumenter/9e0c464e02.pdf`, accessed 24.04.2013.

[3] **SoundPLAN**, Link: `http://soundplan-uk.com/`, accessed 24.04.2013.

[4] **NovaPoint Noise**, Link: `http://www.vianovasystems.com/Products/Novapoint-products/Novapoint-Noise#.UXfGQLV7Jg0`, accessed 24.04.2013.

[5] **Mayer, A. and Bjørklund, P. O.** , *TN12-51 Visnes Kalk noise evaluation*, by Det Norske Veritas(DNV), 2012. Not published.

[6] **Wikipedia, Microsoft Visual Studio**, Link: `http://en.wikipedia.org/wiki/Microsoft_Visual_Studio`, accessed 24.04.2013.

[7] **Wikipedia, C#**, Link: `http://en.wikipedia.org/wiki/C_Sharp_(programming_language)`, accessed 24.04.2013.

[8] **Microsoft, C#**, Link: `http://msdn.microsoft.com/en-us/vstudio/hh341490.aspx`, accessed 24.04.2013.

[9] **Wikipedia, XML**, Link: `http://no.wikipedia.org/wiki/XML`, accessed 02.06.2013.

[10] **AutoCAD**, Link: `http://www.autodesk.com/products/autodesk-autocad/overview`, accessed 14.05.2013.

[11] **Olsen, H.** , *Noise. Characteristics and effects*, Lecture Notes in Norwegian, 2013. Not published.

[12] **ISO 3744**, *Acoustics - Determination of sound power levels and sound energy levels of noise sources using sound pressure*, International standard, Third edition 2010-10-01. International Organization for Standardization(ISO) in Geneva, 1994.

[13] **Young, H.D. and Freedman, R.A.**, *University Physics, with modern physics*, , tenth edition, Sears and Zemansky's. Published by Addison Wesley Longman, Inc., 1999.

[14] **Hassall, J.R. and Zaveri, K.**, *Acoustic Noise Measurements*, Brüel & Kjær, Published by K.Larsen & Søn A/S in Glostrup, Denmark, 1988.

[15] **Barron, R.F.** , *Industrial Noise Control and Acoustics*, Published by Marcel Dekker, Inc. in New York, 2003.

[16] **ISO 1996-1**, *Acoustics - Description, measurement and assessment of environmental noise*, International standard, Second edition 2003-08-01. International Organization for Standardization(ISO) in Geneva, 2003.

[17] **Forurensningsforskriften**, *FOR 2004-06-01 nr 931: Forskrift om begrensning av forurensning* , Kapittel 20. Forurensninger fra produksjon av pukk, grus, sand og singel. Fastsatt av Miljøverndepartementet(Ministry of the Environment) 17. september 2009.
Link: `http://www.lovdata.no/cgi-wift/ldles?doc=/sf/sf/sf-20040601-0931.html`, accessed 24.04.2013.

[18] **T-1442/2012**, *Retningslinje for behandling av støy i arealplanlegging*, By Miljøverndepartementet(Ministry of the Environment), in Norwegian, 2012.
Link: `http://www.regjeringen.no/pages/37952459/T-1442_2012.pdf`, accessed 24.04.2013.

[19] **Visnes Kalk AS**, Link: `www.visneskalk.no`, accessed 24.04.2013.

[20] **Storeheier, S. Å.**, *Brukerveiledning for programmet XISTØY v.1.2*, Beregning av ekstern industristøy. SINTEF Tele og data, Trondheim, 1996.

[21] **The Engineering Toolbox, Sound Pressure**, Link: `http://www.engineeringtoolbox.com/sound-pressure-d_711.html`, accessed 15.05.2013

[22] **SOSI Standard in English, Kartverket**, Link: `http://www.statkart.no/en/Standarder/SOSI/SOSI-Standard-in-English/`, accessed 20.05.2013.

[23] **Auglænd, H.**, Unpublished user manual: ISY.CAD.Noise User Manual, Department of Structural Engineering, NTNU, Trondheim, Norway, 2013.

[24] **Norwegian Public Roads Administration, Road Map**, Link: `https://www.vegvesen.no/vegkart/vegkart/`, accessed 26.05.2013.

[25] **Norwegian Directorate for Civil Protection (DSB) Map**, Link: `http://kart.dsb.no/default.aspx?gui=1&lang=2`, accessed 16.05.2013.

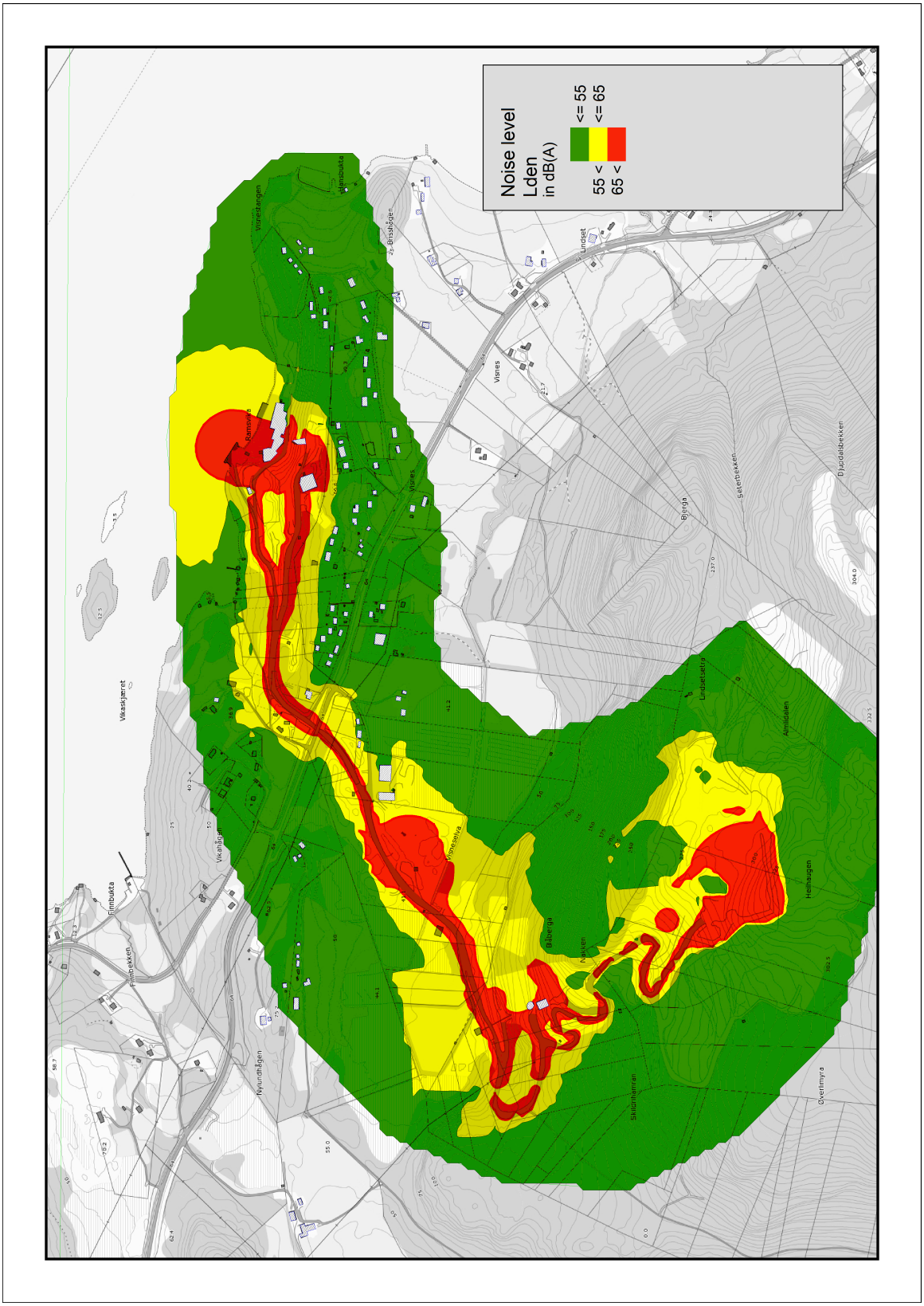
[26] **MeshLab**, Link: `http://meshlab.sourceforge.net/`, accessed 16.05.2013.

[27] **Veileder, støykartlegging og trafikkdata**, Link: `http://www.klif.no/arbeidsomr/stoy/stoykartlegging/Veileder_Stoy_trafikkdata300610.pdf`, accessed 15.05.2013

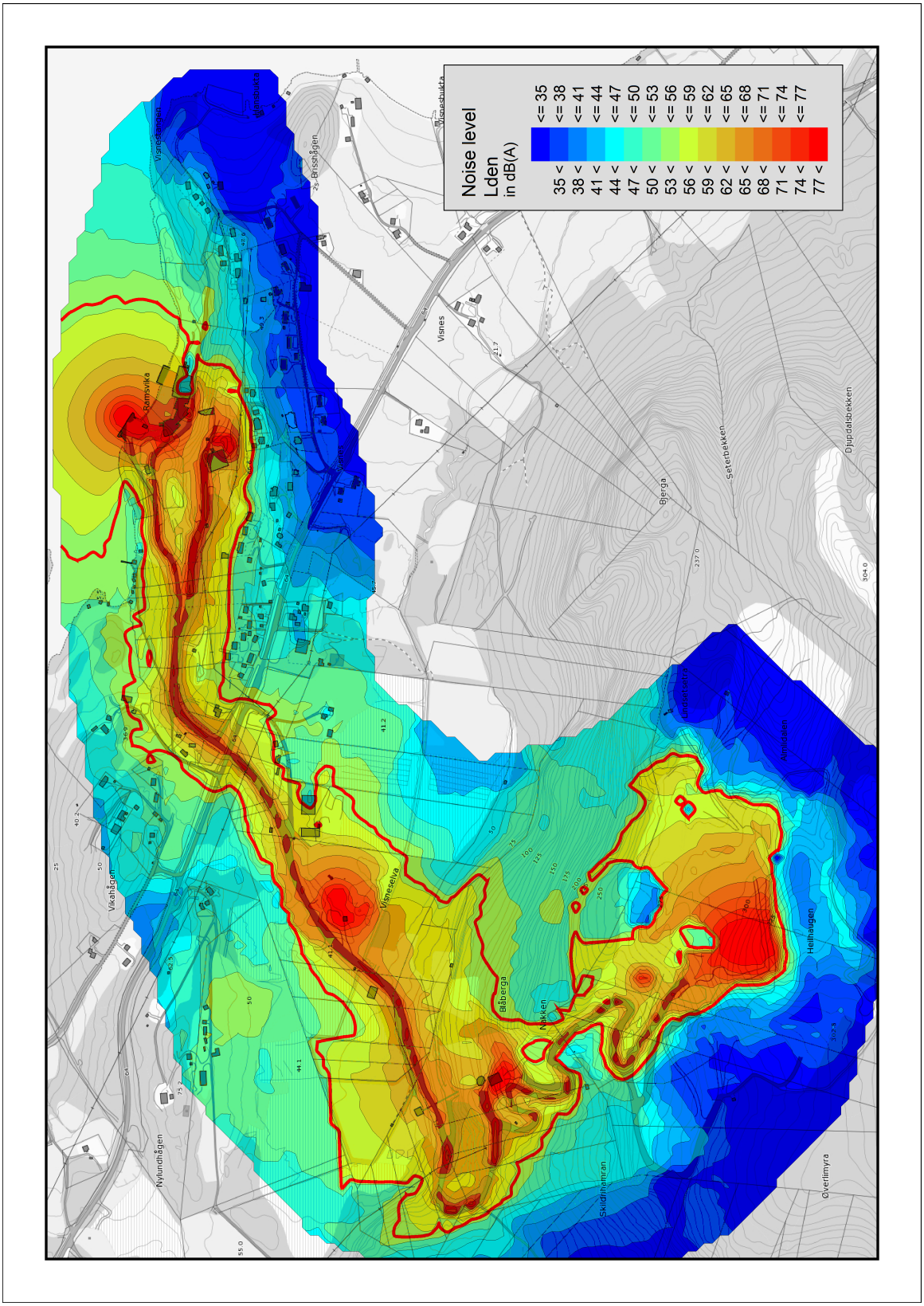
[28] **Gemini, Trafikkstøy inn i stua?**, Link: `http://gemini.no/2013/03/trafikkstoy-inn-i-stua/`, accessed 15.05.2013

A

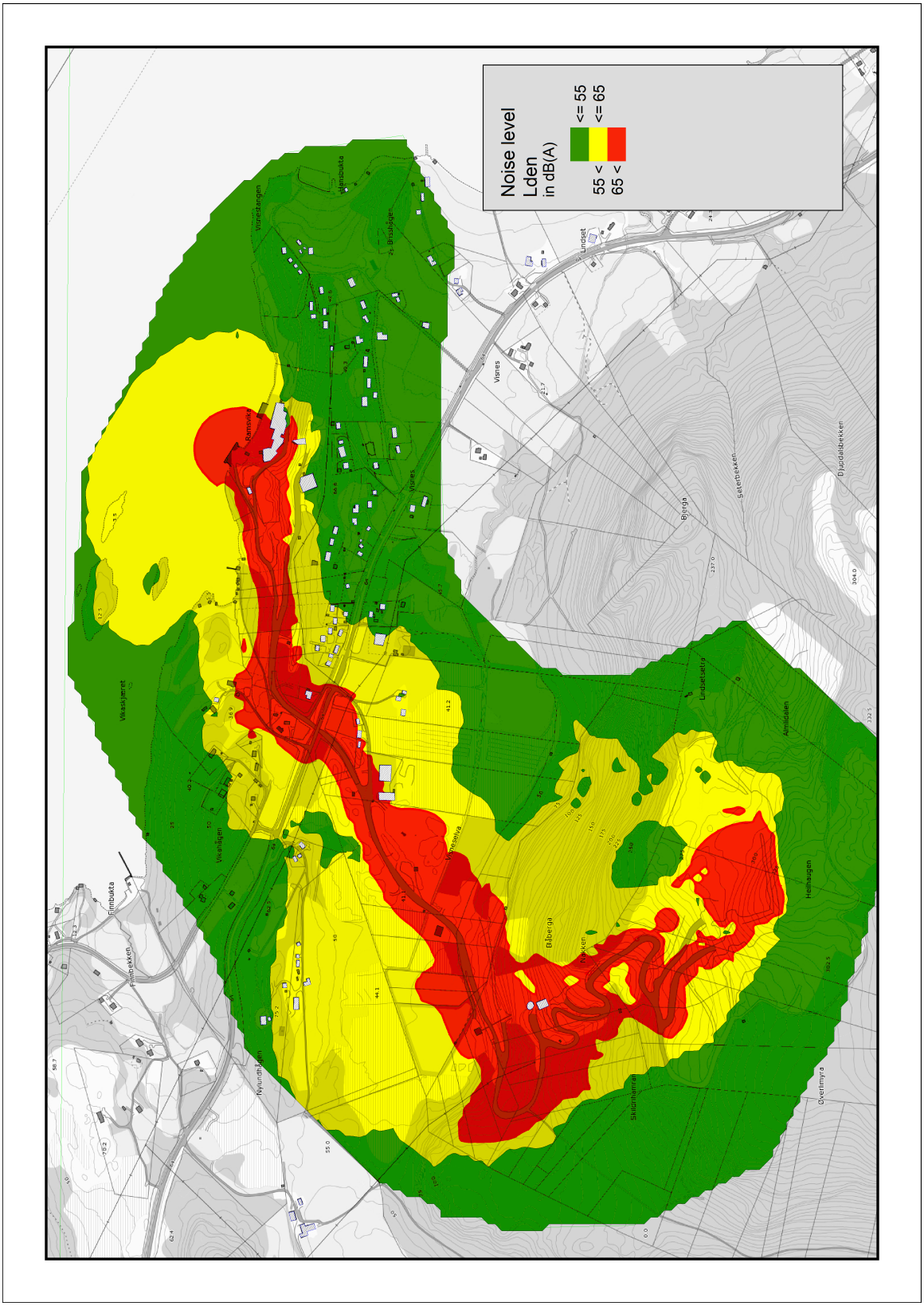
Noise maps from DNV



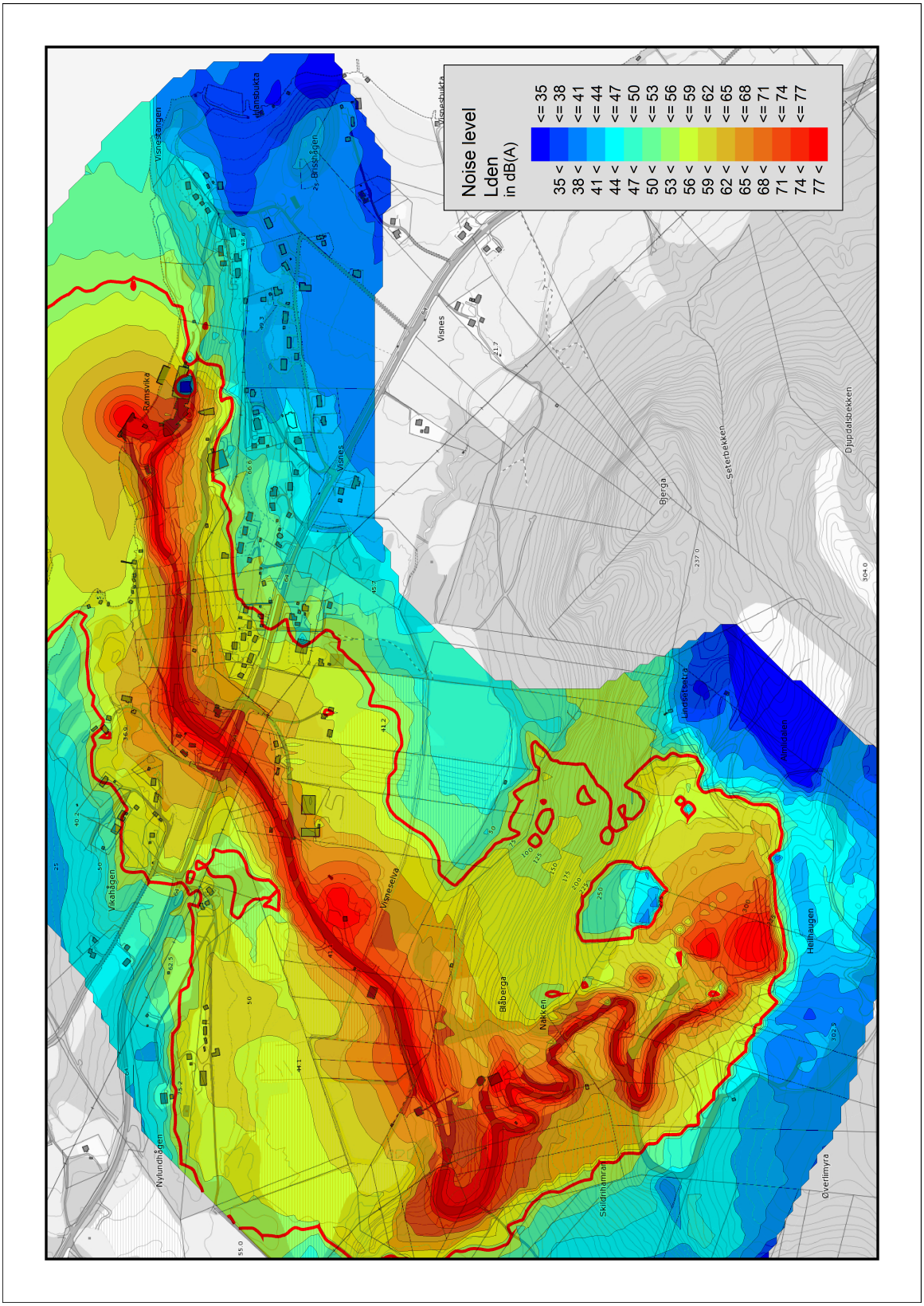
Noise map, condition 1



Noise map, condition 1, gradient



Noise map, condition 2



Noise map, condition 2, gradient

B

Noise Measurements from Visnes Kalk AS

Table B.1: Information about the noise sources

Number	***Noise Source***	***Distance [m]***	***Measurement duration [h:m:s]***
1	Tipping	18	(0:0:23)
2	Crusher, mechanical digger and dumper	17	(0:1:0)
3	Crusher, mechanical digger and dumper	24	(0:0:8)
4	Crusher and dumper	24	(0:1:0)
5	Drilling	9	(0:1:0)
6	Drilling	9	(0:0:40)
7	Drilling	9	(0:1:0)
8	Drilling - when changing drill	9	(0:0:19)
9	Moving medium sized rock	17,5	(0:1:0)
10	Moving medium sized rock	17,5	(0:1:0)
11	Moving medium sized rock	22	(0:0:26)
12	Moving medium sized rock	13	(0:0:4)

Table B.2: Measured and Calculated Sound Levels

Measured Noise Level: L_{eq} [dB]									
Number	***31.5***	***63***	***125***	***250***	***500***	***1000***	***2000***	***4000***	***8000 [Hz]***
1	70,1	75	83,4	82,9	77,2	74,9	70,6	65,6	58
2	84,6	93,2	95,1	90,3	91,5	88,2	86,1	80,1	73,2
3	79,5	89,1	92,8	84	84,9	81,2	76,9	72	66,5
4	81	90,3	92,5	84,9	85,3	81,4	77,2	72,7	67,2
5	76,9	79,4	88,8	80,8	85,6	85,1	88,2	88,4	88,3
6	81,2	79,6	88,6	78,8	85,5	84	86	86,7	87,5
7	78,3	79,4	88,3	79,9	86,4	86,4	90,1	89,6	90,9
8	80,8	85,4	81,8	75,5	72,7	70,2	71	70,2	71,9
9	73,7	75,8	80,5	83,4	81,2	82,9	84,1	79,3	68,7
10	73,5	76,5	81,2	82,7	79,7	82,7	84,6	80,1	68,9
11	74,5	78,3	83	82,5	77	74,3	74	69,3	60,8
12	74,1	75,4	77,1	80	83,7	88,5	92,2	90,3	81

Sound Power Level: L_w [dB]									
Number	***31.5***	***63***	***125***	***250***	***500***	***1000***	***2000***	***4000***	***8000 [Hz]***
1	103,2	108,1	116,5	116,0	110,3	108,0	103,7	98,7	91,1
2	117,2	125,8	127,7	122,9	124,1	120,8	118,7	112,7	105,8
3	115,1	124,7	128,4	119,6	120,5	116,8	112,5	107,6	102,1
4	116,6	125,9	128,1	120,5	120,9	117,0	112,8	108,3	102,8
5	104,0	106,5	115,9	107,9	112,7	112,2	115,3	115,5	115,4
6	108,3	106,7	115,7	105,9	112,6	111,1	113,1	113,8	114,6
7	105,4	106,5	115,4	107,0	113,5	113,5	117,2	116,7	118,0
8	107,9	112,5	108,9	102,6	99,8	97,3	98,1	97,3	99,0
9	106,6	108,7	113,4	116,3	114,1	115,8	117,0	112,2	101,6
10	106,4	109,4	114,1	115,6	112,6	115,6	117,5	113,0	101,8
11	109,3	113,1	117,8	117,3	111,8	109,1	108,8	104,1	95,6
12	104,4	105,7	107,4	110,3	114,0	118,8	122,5	120,6	111,3

Table B.3: Averaged Measured and Calculated Sound Levels

Averaged Measured Noise Level: L_{eq} [dB]									
Number	***31.5***	***63***	***125***	***250***	***500***	***1000***	***2000***	***4000***	***8000 [Hz]***
5, 6 and 7	78,8	79,5	88,6	79,8	85,8	85,2	88,1	88,2	88,9
9 and 10	73,6	76,2	80,9	83,1	80,5	82,8	84,4	79,7	68,8

Averaged Sound Power Level: L_w [dB]									
Number	***31.5***	***63***	***125***	***250***	***500***	***1000***	***2000***	***4000***	***8000 [Hz]***
5, 6 and 7	105,9	106,6	115,7	106,9	112,9	112,3	115,2	115,3	116,0
9 and 10	106,5	109,0	113,7	115,9	113,3	115,7	117,2	112,6	101,7

C

Code from Test Program

The code from the test program is shown on the following page, and is a combination of example code from SINTEF ICT, Acoustics and code produced in this thesis. It shows, in a basic way, all that is needed to perform a simple noise mapping.

```
using System;
using System.IO;
using System.Text;
using System.Collections;
using System.Collections.Generic;

using CommonUtils;
using GeoUtils;
using SoundKernel;
using MapKernel;
using Innemodul;
using System.Windows.Forms;

namespace NoiseComputation
{
    public static class Program
    {
        /// <summary>Kotegenereringsobjekt</summary>
        static mkTriMesh MK = new mkTriMesh();

        /// <summary>Rektangel som omslutter ønsket utsnitt av kartet</summary>
        static typBBox MapBox;

        static int numCalcPoints;


        static typmkLayer[][] layers;

        public static List<NoiseContour[]> contours = new List<NoiseContour[]>();

        public static void Main(string[] args)
        {

            Queue<typError> Errors = new Queue<typError>();
            Queue<typError> Warnings = new Queue<typError>();

            double xorigo = 418208.5783;
            double yorigo = 6981471.4255;

            // Oppretter en (tom) topografimodell
            skTopography Topo = new skTopography();

            #region Creating Grid
            int nx = 68;
            int ny = 56;
            double x1 = 418208.5783;
            double y1 = 6981471.4255;
            double x2 = 418543.5783;
            double y2 = 6981746.4255;
            /// <summary>Vinkel for skråplan mot underliggende grid (0=hor., 90=vert.)</summary>
            int a = 90;
            /// <param name="m_">Default marktype</param>
            /// unknown, hard, soft, A, B, C, D, E, F, G, H, Index6
            enGroundType groundType = enGroundType.soft;
            /// <param name="s_">Default skoghøyde</param>
            int s = 0;
            /// <param name="r_">Default ruhet</param>
            int r = 0;

            skTopGrid Grid = new skTopGrid(x1, y1, x2, y2, a, nx, ny, a, groundType, s, r);

            Grid.zGrid = new float[nx, ny];
            Grid.Refresh();
            Topo.TopGrid.Add(Grid);
            #endregion

            #region Adding Calculation Points
            int numUnits = 2;
            numCalcPoints = 24;
            skSubTask currentSubTask = new skSubTask(0, 0, numCalcPoints, numUnits);

            currentSubTask.points[0, 0] = 418439.97; currentSubTask.points[0, 1] = 6981677.93;
currentSubTask.hpoints[0] = 0;
            currentSubTask.points[1, 0] = 418437.375; currentSubTask.points[1, 1] = 6981674.085;
currentSubTask.hpoints[1] = 0;
            currentSubTask.points[2, 0] = 418434.78; currentSubTask.points[2, 1] = 6981670.24;
currentSubTask.hpoints[2] = 0;
            currentSubTask.points[3, 0] = 418432.53; currentSubTask.points[3, 1] = 6981666.01;
```

```
    currentSubTask.hpoints[3] = 0;
76              currentSubTask.points[4, 0] = 418429.845; currentSubTask.points[4, 1] = 6981660.245;
    currentSubTask.hpoints[4] = 0;
77              currentSubTask.points[5, 0] = 418427.16; currentSubTask.points[5, 1] = 6981654.48;
    currentSubTask.hpoints[5] = 0;
78
79              // Same line, 4m higher up:
80              currentSubTask.points[6, 0] = 418439.97; currentSubTask.points[6, 1] = 6981677.93;
    currentSubTask.hpoints[6] = 4;
81              currentSubTask.points[7, 0] = 418437.375; currentSubTask.points[7, 1] = 6981674.085;
    currentSubTask.hpoints[7] = 4;
82              currentSubTask.points[8, 0] = 418434.78; currentSubTask.points[8, 1] = 6981670.24;
    currentSubTask.hpoints[8] = 4;
83              currentSubTask.points[9, 0] = 418432.53; currentSubTask.points[9, 1] = 6981666.01;
    currentSubTask.hpoints[9] = 4;
84              currentSubTask.points[10, 0] = 418429.845; currentSubTask.points[10, 1] = 6981660.245;
    currentSubTask.hpoints[10] = 4;
85              currentSubTask.points[11, 0] = 418427.16; currentSubTask.points[11, 1] = 6981654.48;
    currentSubTask.hpoints[11] = 4;
86
87              // A different line, moving away from the noise source
88              currentSubTask.points[12, 0] = 418495.94; currentSubTask.points[12, 1] = 6981606.73;
    currentSubTask.hpoints[12] = 0;
89              currentSubTask.points[13, 0] = 418494.93; currentSubTask.points[13, 1] = 6981610.87;
    currentSubTask.hpoints[13] = 0;
90              currentSubTask.points[14, 0] = 418493.86; currentSubTask.points[14, 1] = 6981612.06;
    currentSubTask.hpoints[14] = 0;
91              currentSubTask.points[15, 0] = 418492.71; currentSubTask.points[15, 1] = 6981613.4;
    currentSubTask.hpoints[15] = 0;
92              currentSubTask.points[16, 0] = 418491.58; currentSubTask.points[16, 1] = 6981614.45;
    currentSubTask.hpoints[16] = 0;
93              currentSubTask.points[17, 0] = 418490.33; currentSubTask.points[17, 1] = 6981615.42;
    currentSubTask.hpoints[17] = 0;
94              currentSubTask.points[18, 0] = 418486.43; currentSubTask.points[18, 1] = 6981617.28;
    currentSubTask.hpoints[18] = 0;
95              currentSubTask.points[19, 0] = 418471.8; currentSubTask.points[19, 1] = 6981622.78;
    currentSubTask.hpoints[19] = 0;
96              currentSubTask.points[20, 0] = 418446.08; currentSubTask.points[20, 1] = 6981633.97;
    currentSubTask.hpoints[20] = 0;
97              currentSubTask.points[21, 0] = 418426.3; currentSubTask.points[21, 1] = 6981642.67;
    currentSubTask.hpoints[21] = 0;
98              currentSubTask.points[22, 0] = 418417.26; currentSubTask.points[22, 1] = 6981648.1;
    currentSubTask.hpoints[22] = 0;
99              currentSubTask.points[23, 0] = 418412; currentSubTask.points[23, 1] = 6981651.54;
    currentSubTask.hpoints[23] = 0;
100
101             #endregion
102
103             currentSubTask.Sources = new AListWrap(enFC.Source);
104
105
106             // Trenger å lese denne filen for at fasadepunkt skal håndteres korrekt.
107             if (!clsImport.LesKonfigurasjon("C:\\Dev\\Microstation_v8i\\DLL\\NoiseComputation\\bin\\Debug\
    \konfig.dat", out currentSubTask.config))
108                 throw new Exception("Feil ved lesing av konfig.dat");
109
110             #region Adding Road Noise Source
111
112             skTopRoad newRoad = new skTopRoad("Road0", 10, enHeightBase.sea, enGroundType.hard, "AB16", 1, 30);
113
114             newRoad.pGon[0, 0] = 418548.641875835; newRoad.pGon[0, 1] = 6981739.26413355;
115             newRoad.pGon[1, 0] = 418541.31; newRoad.pGon[1, 1] = 6981739.42;
116             newRoad.pGon[2, 0] = 418518.34; newRoad.pGon[2, 1] = 6981738.37;
117             newRoad.pGon[3, 0] = 418498.99; newRoad.pGon[3, 1] = 6981736.21;
118             newRoad.pGon[4, 0] = 418485.44; newRoad.pGon[4, 1] = 6981734.39;
119             newRoad.pGon[5, 0] = 418476.6; newRoad.pGon[5, 1] = 6981731.48;
120
121             newRoad.pGon[6, 0] = 418468.61; newRoad.pGon[6, 1] = 6981727.44;
122             newRoad.pGon[7, 0] = 418462.49; newRoad.pGon[7, 1] = 6981723.12;
123             newRoad.pGon[8, 0] = 418456.12; newRoad.pGon[8, 1] = 6981717.62;
124             newRoad.pGon[9, 0] = 418448.61; newRoad.pGon[9, 1] = 6981709.9;
125             newRoad.pGon[10, 0] = 418442.67; newRoad.pGon[10, 1] = 6981703.08;
126
127             newRoad.pGon[11, 0] = 418437.45; newRoad.pGon[11, 1] = 6981695.18;
128             newRoad.pGon[12, 0] = 418431.36; newRoad.pGon[12, 1] = 6981685.99;
129             newRoad.pGon[13, 0] = 418420.82; newRoad.pGon[13, 1] = 6981667.12;
130             newRoad.pGon[14, 0] = 418412; newRoad.pGon[14, 1] = 6981651.54;
131             newRoad.pGon[15, 0] = 418406.9; newRoad.pGon[15, 1] = 6981642.59;
```

```
132
133             newRoad.pGon[16, 0] = 418399.49; newRoad.pGon[16, 1] = 6981631.51;
134             newRoad.pGon[17, 0] = 418392.47; newRoad.pGon[17, 1] = 6981620.68;
135             newRoad.pGon[18, 0] = 418381.75; newRoad.pGon[18, 1] = 6981604.59;
136             newRoad.pGon[19, 0] = 418370.59; newRoad.pGon[19, 1] = 6981589.89;
137             newRoad.pGon[20, 0] = 418361.81; newRoad.pGon[20, 1] = 6981579.08;
138
139             newRoad.pGon[21, 0] = 418354.52; newRoad.pGon[21, 1] = 6981571.99;
140             newRoad.pGon[22, 0] = 418347.62; newRoad.pGon[22, 1] = 6981566.21;
141             newRoad.pGon[23, 0] = 418339.67; newRoad.pGon[23, 1] = 6981559.76;
142             newRoad.pGon[24, 0] = 418329.42; newRoad.pGon[24, 1] = 6981552.74;
143             newRoad.pGon[25, 0] = 418315.84; newRoad.pGon[25, 1] = 6981545.57;
144
145             newRoad.pGon[26, 0] = 418295.84; newRoad.pGon[26, 1] = 6981533.39;
146             newRoad.pGon[27, 0] = 418281.46; newRoad.pGon[27, 1] = 6981524.83;
147             newRoad.pGon[28, 0] = 418270.18; newRoad.pGon[28, 1] = 6981517.09;
148             newRoad.pGon[29, 0] = 418255.2; newRoad.pGon[29, 1] = 6981501.19;
149
150             newRoad.hs = new float[33];
151
152             newRoad.Refresh();
153             Topo.TopRoads.Add(newRoad);
154
155             // adding the noise source
156             float amount = 200;
157             float velocity = (float)(50 / 3.6); // 50km/t
158             int numClasses = 3;
159             skSource newSource = new skSource(enActType.road, enPlaceType.line, enQuantityUnit.ydt, amount,
    "Road0", 0, numClasses);
160
161             float[] sourcePeriodTraffic = new float[3];
162             sourcePeriodTraffic[0] = amount;         // Distribution Day
163             sourcePeriodTraffic[1] = 0;              // Distribution Evening
164             sourcePeriodTraffic[2] = 0;              // Distribution Night
165
166             float lightVehicles, mediumVehicles, heavyVehicles;
167             for (int period = 0; period < 3; ++period) //Døgnsegment: Dag, Kveld, Natt
168             {
169                 float periodTraffic = sourcePeriodTraffic[period] / newSource.Q;
170
171                 heavyVehicles = periodTraffic;              // all vehicles are heavy
172                 lightVehicles = 0;
173                 mediumVehicles = 0;
174
175                 for (int vehicleType = 0; vehicleType < 3; ++vehicleType) //Kategorier: Lett, Middels og Tung
176                 {
177                     float categoryTraffic = (vehicleType == 0 ? lightVehicles : ((vehicleType == 1) ?
    mediumVehicles : heavyVehicles));
178                     for (int day = 0; day < 7; ++day) //Ukedag: Mandag til Søndag
179                         for (int month = 0; month < 12; ++month) //Måned: 1-12
180                         {
181                             int numberOfWeekdays = 5;
182                             float thisContribution = categoryTraffic / (numberOfWeekdays * 12);
183                             if (day == 5 || day == 6)
184                                 thisContribution = 0;
185                             newSource.TFac[period, day, month][vehicleType] = thisContribution;
186                         }
187                 }
188             }
189
190             double[] frequencySpectra = new double[27];
191             frequencySpectra[0] = 3.33333333333333E-21; frequencySpectra[1] = 3.33333333333333E-21;
192             frequencySpectra[2] = 3.33333333333333E-21; frequencySpectra[3] = 12102597.5734628;
193             frequencySpectra[4] = 12102597.5734628;     frequencySpectra[5] = 12102597.5734628;
194             frequencySpectra[6] = 635153572.654415;     frequencySpectra[7] = 635153572.654415;
195             frequencySpectra[8] = 635153572.654415;     frequencySpectra[9] = 2903211966.52028;
196             frequencySpectra[10] = 2903211966.52028;    frequencySpectra[11] = 2903211966.52028;
197             frequencySpectra[12] = 4007548115.39138;    frequencySpectra[13] = 4007548115.39138;
198             frequencySpectra[14] = 4007548115.39138;    frequencySpectra[15] = 4818132569.15309;
199             frequencySpectra[16] = 4818132569.15309;    frequencySpectra[17] = 4818132569.15309;
200             frequencySpectra[18] = 3571731017.45868;    frequencySpectra[19] = 3571731017.45868;
201             frequencySpectra[20] = 3571731017.45868;    frequencySpectra[21] = 856798594.256288;
202             frequencySpectra[22] = 856798594.256288;    frequencySpectra[23] = 856798594.256288;
203             frequencySpectra[24] = 71265402.9834077;    frequencySpectra[25] = 71265402.9834077;
204             frequencySpectra[26] = 71265402.9834077;
205
206             newSource.Params["RoadSrcSpec"] = populateRoadSourceSpec(frequencySpectra);
207
```

```
208             newSource.RefID = "0 0.000 0.999"; // Dette vil vanligvis være "ReferanselenkeID FromMeasure
    Tomeasure"
209             newSource.Params["RoadSurf"] = newRoad.rs;      // Vegdekke
210             newSource.Params["RoadWidth"] = newRoad.b;      // Vegbredde
211             newSource.Params["RoadSpeedLim"] = velocity;         // 40 km/t
212 
213             newSource.Vectors = newRoad.Vectors;
214             newSource.h = newRoad.hs;
215             newSource.Refresh();
216             currentSubTask.Sources.Add(newSource);
217 
218             #endregion
219 
220             #region Defining a Building
221             skTopBuilding newBuilding = new skTopBuilding(enHeightBase.sea, 7, 4, 1, 1);
222             newBuilding.ID = 0;
223 
224             newBuilding.OutLines[0].X = 418468.3; newBuilding.OutLines[0].Y = 6981655.5; newBuilding.OutLines
    [0].Z = 3.5f;
225             newBuilding.OutLines[1].X = 418469.9; newBuilding.OutLines[1].Y = 6981660.7; newBuilding.OutLines
    [1].Z = 3.5f;
226             newBuilding.OutLines[2].X = 418477.9; newBuilding.OutLines[2].Y = 6981658.3; newBuilding.OutLines
    [2].Z = 3.5f;
227             newBuilding.OutLines[3].X = 418473.9; newBuilding.OutLines[3].Y = 6981644.9; newBuilding.OutLines
    [3].Z = 3.5f;
228             newBuilding.OutLines[4].X = 418459.5; newBuilding.OutLines[4].Y = 6981649.3; newBuilding.OutLines
    [4].Z = 3.5f;
229             newBuilding.OutLines[5].X = 418462; newBuilding.OutLines[5].Y = 6981657.4; newBuilding.OutLines[5].Z
    = 3.5f;
230             newBuilding.OutLines[6].X = 418468.3; newBuilding.OutLines[6].Y = 6981655.5; newBuilding.OutLines
    [6].Z = 3.5f;
231 
232             newBuilding.InLines[0].X = 418475.1; newBuilding.InLines[0].Y = 6981649; newBuilding.InLines[0].Z =
    5.1f;
233             newBuilding.InLines[1].X = 418460.7; newBuilding.InLines[1].Y = 6981653.3; newBuilding.InLines[1].Z
    = 5.1f;
234             newBuilding.InLines[2].X = 418471.2; newBuilding.InLines[2].Y = 6981650.6; newBuilding.InLines[2].Z
    = 5.1f;
235             newBuilding.InLines[3].X = 418473.9; newBuilding.InLines[3].Y = 6981659.5; newBuilding.InLines[3].Z
    = 5.1f;
236 
237             newBuilding.DoFPnts = true;
238             newBuilding.Refresh();
239             Topo.TopBuildings.Add(newBuilding);
240 
241             currentSubTask.sitidx.Add(0);
242             #endregion
243 
244             #region All the other stuff
245 
246             // Genererer trekantmodell basert på innleste objekter
247             Top2Tri T2T = new Top2Tri(Topo, false);
248             T2T.SetLimits(x1, x2, y1, y2, xorigo, yorigo);
249             T2T.SetBldIDTrans(currentSubTask.buildings, currentSubTask.bldIDTrans, currentSubTask.ih, 0f,
    currentSubTask.config);
250             if (!T2T.CreateTriangles(true, true, true, true, true, true, true, null))
251             {
252                 //  Noe galt skjedde i forbindelse med triangulering
253                 MessageBox.Show("Error in triangulation!", "Error", MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation);
254             }
255 
256             List<conline> ConLines = new List<conline>();
257             List<conpoly> ConPolys = new List<conpoly>();
258             // Henter kanter fra triangulering av topografi (er nødvendig for å bygge opp trekantsamling
259             // for beregningspunkt, for at kotegenerering ikke skal krysse kanter, som f.eks. skjermer og
    bygninger.
260             // Dersom dette IKKE er et krav, kan dette utelates. ConLines og ConPolys skal da være tomme.
261 
262             // Henter alle linjer og alle TOMME polygoner ( = bygninger)
263             T2T.GetLinesPolys(true, true, true, out ConLines, out ConPolys);
264             // Tar med bare et utvalg av linjer og polygoner; Bare skjermer og bygningsomriss; Ikke taklinjer,
    etc
265             // Må også sikre at evt. endringer i punktene i forbindelse med trekantsamling for beregning
266             // IKKE påvirker punktene i topografien. Tar derfor KOPIER av linjene og polygonene
267             List<conline> TmpLines = new List<conline>(ConLines.Count);
268             List<conpoly> TmpPolys = new List<conpoly>(ConPolys.Count);
269             for (int i = 0; i < ConLines.Count; i++)
```

```
270             {
271                 if ((string)ConLines[i].par["Otyp"] != "TopoScreen") continue;
272                 TmpLines.Add(ConLines[i].Copy(false));
273             }
274             for (int i = 0; i < ConPolys.Count; i++)
275             {
276                 if ((string)ConPolys[i].par["Otyp"] != "TopoBuilding") continue;
277                 TmpPolys.Add(ConPolys[i].Copy(false));
278             }
279             ConLines = TmpLines; ConLines.TrimExcess(); TmpLines = null;
280             ConPolys = TmpPolys; ConPolys.TrimExcess(); TmpPolys = null;
281 
282             // Bestemmer de unike punktene i trekantsamlingen
283             Topo.TopTriang.FindUniquePoints();
284             // Må konvertere alle genererte objekter (trekanter og speil) til globale koordinater
285             // (ConLines og ConPolys forblir i lokale koordinater)
286             Topo.TopTriang.ConvertToGlobal(xorigo, yorigo);
287             Topo.ConvertMirrorsToGlobal(xorigo, yorigo);
288             Topo.TopTriang.RefreshHash();
289             Topo.Refresh();
290 
291             // Definerer beregningsenheter: Lden
292             currentSubTask.Units[0].Unit = enSoundUnit.Leq;
293             currentSubTask.Units[0].TW = enTimeWeight.DEN;
294             currentSubTask.Units[0].FW = enFreqWeight.A;
295             currentSubTask.Units[0].L1 = 10.0f;
296             currentSubTask.Units[0].Toler = 0.2f;
297 
298             // Definerer Leq
299             currentSubTask.Units[1].Unit = enSoundUnit.Leq;
300             currentSubTask.Units[1].TW = enTimeWeight.none;
301             currentSubTask.Units[1].FW = enFreqWeight.A;
302             currentSubTask.Units[1].L1 = 10.0f;
303             currentSubTask.Units[1].Toler = 0.2f;
304 
305             // Spesifiserer værforhold
306             skMetClass Met = new skMetClass();
307             Met.TGrad = 0.0f;
308             Met.Refresh(); // Sluttberegner interne egenskaper for været
309             currentSubTask.MET = new AListWrap(enFC.MET, Met); // Legger været til oppdraget
310             currentSubTask.MET.Refresh();
311 
312             // Dumper 3D-modell til fil (for debug innsyn)
313             try //se if it is possible with activityType timecheck for this one
314             {
315                 TriGrid2File.WriteParts(Topo, 1.0, true, false, -1);
316             }
317             catch
318             {
319                 MessageBox.Show("Some parts are not defined correctly or not on the terrain", "Info",
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
320             }
321 
322             // Definerer beregningsgrid, 20 meter gridstørrelse
323             griddef GrdDef = new griddef();
324             GrdDef.x0 = x1; GrdDef.y0 = y1;
325             GrdDef.xi = 1.0f; GrdDef.yi = 0.0f;
326             GrdDef.Lu = x2 - x1; GrdDef.Lv = y2 - y1;
327             GrdDef.nu = nx; GrdDef.nv = ny;
328             currentSubTask.GrdDef = GrdDef;
329 
330             // Bygger trekantmodell for beregningspunkt.
331             // Merk at hvis ConLines og ConPolys IKKE er tomme (men inneholder skjermer og bygningsomriss)
332             // så vil beregningsgridet IKKE være regulært; Det vil bli etablert beregningspunkt BÅDE for
333             // punktene definert i GrdDef OG for alle punktene i ConLines og ConPolys.
334             Grid2Tri G2T = new Grid2Tri(currentSubTask);
335             if (!G2T.CreateCalcTri(GrdDef, ConLines, ConPolys, xorigo, yorigo, null))
336                 // Noe galt skjedde i forbindelse med triangulering
337                 throw new Exception("Feil ved triangulering av beregningsgrid");
338 
339             #endregion
340 
341             #region Computation
342             currentSubTask.SubTopoHash = Topo.Hash;
343             currentSubTask.Method = enMethod.N2000R;
344             currentSubTask.ih = enHeightBase.terr;
345             currentSubTask.Sources.Refresh();
346             currentSubTask.Refresh();
```

```
347
348             // Utfører beregning for beregningspunktene
349             SK.TOPO = Topo;
350
351             // Triks for å sikre at resultatene lagres korrekt
352             SK.Task = skSubTask.Load(currentSubTask.Save());
353
354             SK.Task.bldDetailedFP = new Dictionary<long, int>();
355             SK.Task.bldDetailedFP[0] = 3;
356
357             //resetting and enabling the computation to be run several times
358             SK.Stopped = false;
359             SK.Stopping = false;
360             SK.Errors.Clear();
361             SK.PercentFinished = 0;
362
363             SK.MainStatus = enMStat.active;
364             SK.threadcontrol = new System.Threading.EventWaitHandle(false,
    System.Threading.EventResetMode.ManualReset);
365             SK.threadcontrol.Set();
366
367             SK.RunSubTask();
368
369             if (SK.MainStatus == enMStat.error)
370             {
371                 //  Noe galt skjedde i forbindelse med beregning
372                 MessageBox.Show("Feil oppsto under beregning: \n" + SK.Errors.Peek().Desc
373                     + ", \n" + SK.Errors.Peek().ErrMsg + ", \n in module: "
374                     + SK.Errors.Peek().Module + ", \n and in function: " + SK.Errors.Peek().Func, "Info",
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
375             }
376             else
377             {
378                 float mscale = 5000f;
379                 SK.Contours = new List<skContour>();
380
381                 // Definerer kotesett for Lden
382                 skContour Cont = new skContour();
383                 Cont.UnitIdx.Add(0); // Indeks til støyenhet støykotene skal lages for
384                 Cont.SmoothRadius = 10f; // Glatteradius 10 m
385                 Cont.CellSize = mscale / 1000f;
386                 Cont.Levels.Add(new float[2]); // Lager koter for Lden 55 og 65 dB
387                 Cont.Levels[0][0] = 55f;
388                 Cont.Levels[0][1] = 65f;
389                 SK.Contours.Add(Cont);
390
391                 // Definerer detaljert kotesett for Lden
392                 skContour Cont2 = new skContour();
393                 Cont2.UnitIdx.Add(0); // Indeks til støyenhet støykotene skal lages for
394                 Cont2.SmoothRadius = 10f; // Glatteradius 10 m
395                 Cont2.CellSize = mscale / 1000f;
396                 Cont2.Levels.Add(new float[15]); // Lager koter for Lden 35, 38, 41, 44, 47, 50, 53, 56, 59, 62,
    65, 68, 71, 74 og 77dB
397                 for (int i = 0; i < 15; i++)
398                     Cont2.Levels[0][i] = 35 + i * 3;
399                 SK.Contours.Add(Cont2);
400
401                 layers = new typmkLayer[2][];
402
403                 bool stError, SaveOK;
404                 SaveResults((skSubTask)SK.Task, Errors, out stError, out SaveOK);
405
406                 calculateValues(SK.Task, numUnits);
407                 saveFacadeResults((skSubTask)SK.Task, Errors);
408
409                 for (int index = 0; index < 2; index++)
410                 {
411                     if (layers[index] == null)
412                         continue;
413                     NoiseContour[] newContour = new NoiseContour[layers[index].Length];
414
415                     for (int i = 0; i < layers[index].Length; i++) // layers
416                     {
417                         float level;
418                         List<double[,]> p = new List<double[,]>();
419                         //double[,,] points= new double[k,layers[i].PolyLines[k].Length,2];
420                         level = layers[index][i].Level;
421                         for (int k = 0; k < layers[index][i].PolyLines.Count; k++) // lines in layer
```

```
                        {
                            double[,] points = new double[layers[index][i].PolyLines[k].Length, 2];
                            for (int j = 0; j < layers[index][i].PolyLines[k].Length; j++) // points in lines
                            {
                                try
                                {
                                    points[j, 0] = layers[index][i].PolyLines[k][j].x;
                                    points[j, 1] = layers[index][i].PolyLines[k][j].y;
                                }
                                catch { }
                            }
                            p.Add(points);
                        }
                        newContour[i] = new NoiseContour(p, level);
                    }
                    contours.Add(newContour);
                }
            }


            #endregion

        }

        private static double[, , ,] populateRoadSourceSpec(double[] frequencySpectra)
        {
            // 3 kjøretøykategorier, 3 kildehøyder [1 cm, 30 cm, 75 cm], 27 frekvenser, 3 enheter [Lw, L5, Lmax]
            double[, , ,] sourceSpec = new double[3, 3, 27, 3];
            // Verdier trenger bare fylles ut for kjøretøykategorier og kildehøyder som skal brukes.
            // NB: Kategori 0 ("lette") bruker kildehøyde 0 (1 cm) og 1 (30 cm).
            // Kategori 1 ("middels") og 2 ("tunge") bruker kildehøyde 0 (1 cm) og 2 (75 cm)!
            // Pass på at ulike kategorier har utfylt data for sin(e) kildehøyder.
            // Enhet 0 må fylles ut for å få beregnet Lden og Leq.
            // Enhet 2 må fylles ut for å få beregnet Lmax (hvis det er aktuelt).
            // Enhet 1 (L5) kan utelates.

            for (int vehicle_i = 0; vehicle_i < 3; vehicle_i++)
            {
                if (vehicle_i == 0 || vehicle_i == 1)
                    continue;
                for (int sourceheight_i = 0; sourceheight_i < 3; sourceheight_i++)
                {
                    if (sourceheight_i == 1)
                        continue;
                    for (int freq_i = 0; freq_i < 27; freq_i++)
                    {
                        for (int unit_i = 0; unit_i < 3; unit_i++)
                        {
                            if (unit_i == 1 || unit_i == 2)
                                sourceSpec[vehicle_i, sourceheight_i, freq_i, unit_i] = 0;
                            //else fylle ut for unit = 1, og frekvenser
                            else
                                sourceSpec[vehicle_i, sourceheight_i, freq_i, unit_i] = frequencySpectra[freq_i];

                        }
                    }
                }
            }
            return sourceSpec;
        }

        private static void calculateValues(skTask task, int numUnits)
        {
            string lines = "";
            float[,] p = task.resultpoints;
            for (int i = 0; i < numCalcPoints; i++)
            {

                lines += "\r\n Point " + i + ":\t x: \t" +
                    task.points[i, 0] + "\t y: \t" + task.points[i, 1] +
                    "\t z: \t" + task.hpoints[i] + "\t Lden: \t" + p[i, 0] + "\t" +
                    "\t Leq: \t" + p[i, 1] + "\t";
            }

            System.IO.File.WriteAllText(@"C:\TEMP\WriteLines.txt", lines);
        }

        private static void saveFacadeResults(skSubTask subtask, Queue<typError> Errors)
```

```
500         {
501             double dbval;
502             Bygning B;
503             double mmin = double.PositiveInfinity;
504             double mmax = double.NegativeInfinity;
505 
506             int iBld, ifp, iu = 0, nu = subtask.numUnits;
507             string lines = string.Empty;
508 
509             for (iBld = 0; iBld < subtask.buildings.Count; iBld++)
510             {
511                 B = subtask.buildings[iBld];
512 
513                 if (B.fasadepunkt.Count == 0) continue; // Skriver ikke "tomme" bygninger
514                 lines += "\r\n Building number: " + iBld + ", " + B.bygningsnummer;
515 
516                 for (ifp = 0; ifp < B.fasadepunkt.Count; ifp++)
517                     //recnum++;
518                     for (iu = 0; iu < nu; iu++)
519                     {
520                         lines += "\r\n Facade point: \t" + ifp + "\tPosition East: \t" + B.fasadepunkt
    [ifp].sted.posisjonE +
521                             "\tPosition North: \t" + B.fasadepunkt[ifp].sted.posisjonN + "\tHeight: \t" +
    B.fasadepunkt[ifp].sted.h +
522                             "\tDecibel value: \t" + B.fasadepunkt[ifp].utesituasjoner[0].utestøy.nivå[iu].verdi +
523                             "\tUnit: \t" + subtask.Units[iu].Unit.ToString();
524 
525                         dbval = B.fasadepunkt[ifp].utesituasjoner[0].utestøy.nivå[iu].verdi;
526                         mmin = Math.Min(dbval, mmin);
527                         mmax = Math.Max(dbval, mmax);
528                     }
529                 lines += "\r\nMin Value: \t" + mmin + "\tMax Value: \t" + mmax;
530                 mmin = double.PositiveInfinity; mmax = double.NegativeInfinity;
531             }
532             System.IO.File.WriteAllText(@"C:\TEMP\WriteFacadePoints.txt", lines);
533         }
534 
535         public static void SaveResults(skSubTask subtask, Queue<typError> Errors, out bool stError, out bool
    SaveOK) {}
536         // ************************************************************
537         // DISSE RUTINENE ER NØDVENDIG FOR Å LAGRE RESULTATER TIL FILER
538         // DE ER LAGET AV SINTEF OG BRUKES KUN FOR Å SJEKKE RESULTATER
539         // DE ER IKKE RELEVANTE FOR FORSTÅELSEN AV PROGRAMMET OG
540         // INKLUDERES IKKE HER
541         // ************************************************************
542 }
```