Spatial interpolation of accessibilities

Theresa Thunig
Transport Planning and Transport Telematics Group,
Berlin Institute of Technology (TU Berlin), Berlin, Germany

Thomas W. Nicolai
Transport Planning and Transport Telematics Group,
Berlin Institute of Technology (TU Berlin), Berlin, Germany

June 24, 2013

Abstract

This paper is a technical report about the spatial interpolation of structured data. There are many different interpolation methods. This report provides a general overview about interpolation methods, discusses three methods *bilinear interpolation*, *bicubic spline interpolation* and *inverse distance weighting* in more depth and compares them based on different testing criteria.

1 Introduction

This work is motivated by an attempt to couple a land use simulation model, UrbanSim (Miller et al., 2005; OPUS User Guide, 2011; Waddell, 2002), with a transport simulation model, MATSim (Balmer et al., 2005, 2009; Raney and Nagel, 2006). The transport model is to feed back information to the land use model such as accssibilities to certain activity locations, e.g. work places. Normally, accessibilities are attached to spatial units like zones or parcels. However, the computational effort on the disaggregated parcel level would be rather large. Nicolai and Nagel (2012a, submitted in 2013) are introducing an approach to calculate accessibilities at high-resolutions where the study area is approximated by a grid of configurable size. In this approach accessessibilities are measured at and attached to cell centroids. Accessibilites at arbitrary locations, e.g. individual parcels or buildings, may be interpolated based on the grid.

The present paper discusses different interpolation methods and strategies to obtain an interpolated value from the grid at an arbitrary location. At this point no previous knowledge about the MATSim for UrbanSim integration is required. Readers who are interested in the integration are referred to (Nicolai, 2013; Nicolai and Nagel, 2013; Nicolai et al., 2011, 2013).

2 Spatial interpolation on a grid

Interpolation is the estimation of function values at unknown points with the aid of known function values at other points. In case of spatial interpolation, the point-value pairs are located in a three dimensional space. We are working with structured data, since the known sampling points are located in a two dimensional grid. Some interpolation methods only work with structured data. Before we discuss different interpolation possibilities, we define our problem formally:

Given are function values at grid points

$$f(x_i, y_i) = z_{ij} \text{ for } i = 0, \ldots, n, j = 0, \ldots, m.$$

With interpolation it is possible to estimate the function value at an arbitrary point (x, y) inside the grid. Therefore a function that passes through the given function values is defined.

3 Spatial interpolation methods

There are many different interpolation methods, since there is not the only one correct solution. You may distinguish between local and global interpolation, where global interpolation uses all sampling points for the calculation and local interpolation only uses a selected amount of them. Further possibility is the consideration of abrupt interpolation, which admits points of discontinuity in the interpolation function, and continuous interpolation, which requires a continuous function. Besides exact interpolation, which is used in this case, there also exists approximative interpolation, which admits a certain mistake on the interpolation of sampling points but has other advantages instead. Approximation is reasonable, if, for example, the underlying data is only an estimation. Further there are differences between deterministic and stochastic interpolation. Deterministic interpolation is based on stated mathematical functions. This interpolation assumes a simple spatial coherence, where neighboring points are interpolated by related values. Therefore this kind of interpolation is predictable. On the other hand, stochastic or similar geostatistic interpolation assumes a distribution function and is able to consider exact spatial coherences, like altitude differences or coherences applied to direction (see e.g. (Chang, 2012, pp. 315) or (Peters, 2009, p. 15)).

Generally you have to think about which interpolation method is the most appropriate one for your usage. As the statistician Georg Box once said: "All models are wrong, but some are useful" (Box, 1979, p. 2). This paper will present selected interpolation methods. There are many more.

3.1 Nearest neighbor

Nearest neighbor interpolation assigns the value of the nearest sampling point to the point to interpolate. In Fig. 1 for example, where the sampling points build a grid, every value in a grid cell will be interpolate with the value of the grid cell sampling point. This method is probably the most popular abrupt interpolation method. It is also known as Thiessen or Voronoi polygon method (see e.g. Chang, 2012, pp. 119 and pp. 319). The algorithm needs a very low calculation time, but only offers a low precision.

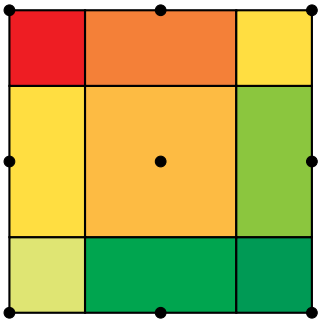


Figure 1: Nearest neighbor interpolation

3.2 Triangulation with linear Interpolation

Linear interpolation with triangulation is an interpolation procedure, which divides the testing region into triangles with the sampling points as vertices. This division is not necessarily unique. Normally the division, where the minimum angle of the triangles is maximized, is used for interpolation. It is called the Delaunay-Triangulation. On each triangle the procedure assumes a linear function, i.e. points to interpolate are assigned to the value at the plane, which goes through the three vertices of the corresponding triangle (see e.g. de Berg et al., 2000, pp. 191). This continuous interpolation method does not need a lot of calculation time. But if you compare it to bilinear interpolation (see Sec. 3.4) it does not offer a high precision.

3.3 Polynomial interpolation

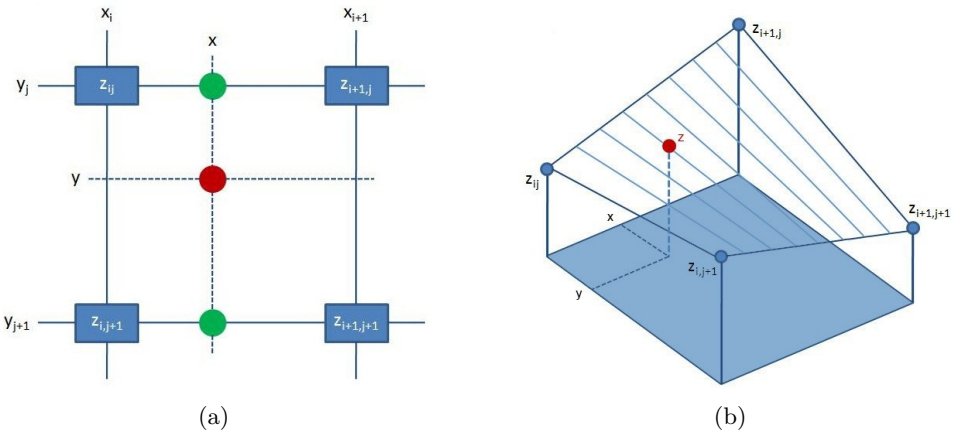
Polynomial interpolation defines a polynomial function, passing through all sampling points. Thus it is a global and exact interpolation method. There are different techniques to find such a polynomial (see e.g. Burden and Faires, 1989, pp. 91 for Lagrange interpolation polynomials). It is a strong claim, that the polynomial has to pass exactly through all sampling points. Therefore strong oscillations may appear at the boundary. Additionally oscillations inside the grid may occur if neighboring sampling points values differ a lot.

For the interpolation of two dimensional data this method is almost never used, because the strong oscillations cause a mistake which is too high.

3.4 Bilinear interpolation

Consider first linear spline interpolation in the plane, i.e. for one dimensional data. This method assumes a linear function on each interval between two sampling points.

Bilinear interpolation extends linear spline interpolation to two dimensional grid data: It interpolates first horizontal with linear spline interpolation, that is, it calculates the values $f(x, y_j)$ and $f(x, y_{j+1})$. After that it interpolates $f(x, y)$ vertically with the same interpolation method (see e.g. Chang, 2012, pp. 119 and Fig. 2). The result is the same, if you interpolate first vertically, then horizontally.



(a) (b)

Figure 2: Figure 2(a) illustrates the principle of bilinear interpolation with linear interpolation in horizontal and vertical direction. Figure 2(b) visualizes it in the three dimensional space. Figure 2(a) and 2(b) adapted from Smith (1997), p. 396.

Hence bilinear interpolation considers only the proportion of the distances to the sampling points. An interpolation method which considers exact distances to the sampling points is the inverse distance weighting (see Sec. 3.6). An important property of bilinear interpolation is that the interpolation of one value only uses the four sampling points of the corresponding grid cell. This has the effect of lower computation time when compared with global interpolation methods.

3.5 Bicubic spline interpolation

Consider again first cubic spline interpolation in the plane. This interpolation method divides the testing area into intervals between the sampling points, like linear spline interpolation, and assumes a third degree polynomial on each interval. The separate functions have to blend smoothly into

each other at the interval boundaries, hence the first and second derivatives of the neighboring functions have to coincide (see e.g. Burden and Faires, 1989, pp. 127).

Bicubic spline interpolation extends cubic spline interpolation to two dimensional data: With cubic splines, which are calculated for all rows and columns, the values of the partial derivatives may be approximated at the sampling points. With these approximated values and the known values at the sampling points the coefficients a_{ij} of the generally bicubic function

$$f(x,y) = \sum_{\substack{i=0,\ldots,3 \\ j=0,\ldots,3}} a_{ij} x^i y^j$$

may be calculated for every grid cell (see e.g. Späth, 1991, pp. 76). Therefore you get a smoother surface than with bilinear interpolation, but you need, on the other hand, more calculation time instead because bicubic spline interpolation uses all sampling points for the calculation and has to calculate derivatives additionally. There are some oscillations, in contrast to bilinear interpolation, but much less than with polynomial interpolation (see the example in Appendix A.1).

3.6 Inverse distance weighting

The inverse distance weighting method (or IDW for short) takes the exact distances between the point to interpolate and the sampling points into account and weights accordingly. The bigger the distance between a sampling point and the point to interpolate, the less its weight for the computation. This interpolation method may also be used for unstructured data. For grid data the following formula is used:

$$f(x,y) = \sum_{\substack{i=0,\ldots,n \\ j=0,\ldots,m}} w_{ij}\ f(x_i, y_j) \tag{1}$$

where the factor w_{ij} weights the sampling points by its distance to the point to interpolate and scales them. It is

$$w_{ij} = \frac{{d_{ij}}^{-a}}{\sum_{\substack{k=0,\ldots,n \\ l=0,\ldots,m}} {d_{kl}}^{-a}},$$

where $d_{ij} = \sqrt{(x-x_i)^2 - (y-y_j)^2}$ is the euclidean distance between (x,y) and (x_i, y_j) in the plane. The user may decide, how many known values, i.e. sampling points, should be used in the calculation. This decision has a big influence on the calculation time of the interpolation. In formula (1) all sampling points are used. Moreover the exponent a for the distance

weighting may be choosen. With this interpolation method, circular spreads occur around the sampling points, which are called bull eyes. Especially the interpolation of regions, with only a few known data values, achieves this effect (see e.g. (Chang, 2012, pp. 321) and (Peters, 2009, pp. 15)).

3.7 Kriging

Kriging is a geostatistic interpolation procedure, which also works for unstructured data. In constrast to the inverse distance weighting, the weight of particular sampling points does not depend only on their distance to the point to interpolate. The distribution of sampling points in the plane and the correspondence of their values is used additionally. Kriging is a complex interpolaion method, which is why we only describe it roughly here (for more details see e.g. Chang, 2012, pp. 324).

The correspondences of neighboring sampling points may be analysed with variography. A variogram which presents the spatial coherences may be created. With it the weights w_{ij} of the sampling points may be calculated. They are used in the interpolation formula

$$f(x,y) = \sum_{\substack{i=0,\dots,n \\ j=0,\dots,m}} w_{ij} f(x_i, y_j)$$

for

$$\sum_{\substack{i=0,\dots,n \\ j=0,\dots,m}} w_{ij} = 1.$$

4 Implementation of selected methods

We implemented three of the presented interpolation methods: Bilinear interpolation, bicubic spline interpolation and inverse distance weighting.

Since we interpolate data from a grid, it seems to be reasonable for us to consider at least the four nearest neighboring sampling points, i.e. in most cases the corners of the respective grid cell. Therefore the interpolation methods nearest neighbor and triangulation drop out. Additionally, we need good solutions at the boundary too, so polynomial interpolation drops out due to its high oscillations there. Kriging drops out as it is a too complex interpolation method for our case and requires too much calculation time.

Because we want to combine the interpolation with MATSim4UrbanSim we decided to use the same programming language - Java.

You will get an overview about our implementation in the MATSim API[1] under the MATSim4UrbanSim extension. There is a general class Interpolation.java, which works as a wrapper to use the three named interpolation methods, whose source code you may additionally find in the Appendix A.2.

[1] http://www.matsim.org/javadoc

4.1 Implementation of bilinear interpolation

We could not find a suitable implementation of this interpolation method, therefore we implemented it ourselves. As described in Sec. 3.4, our implementation interpolates with the aid of linear spline interpolation, first vertical, then horizontal. The implementation needs a low calculation time and interpolates averages between the sampling points as expected. Hence the data looks a bit blurry when you plot it. Because of this, bilinear interpolation is seldemly used for imaging processing. Although in our application, i.e. the interpolation of accessibilities, it is reasonable.

4.2 Implementation of bicubic spline interpolation

For bicubic spline interpolation we worked with the classes BicubicSplineInterpolator and BicubicSplineInterpolatingFunction from the interpolation package provided by Apache (see Apache Software Foundation, 2013). The Apache License is an open source license and compatible with the Gnu Public License (GPL) from MATSim. The interpolation classes BicubicSplineInterpolator and BicubicSplineInterpolatingFunction work with the bicubic spline interpolation algorithm described in Sec. 3.5. This interpolation method induces mild oscillations between neighboring values which differ a lot. Therefore it is possible that a point at the boundary between a mean accessibility region and a low accessibility region is interpolated with a high accessibility, since the difference between the regions values causes oscillations. If the difference between two neighboring values is very high, oscillations may occur not only between them, but also in their neighborhood (see the example in Appendix A.1).

4.3 Implementation of inverse distance weighting

We could not find a suitable implementation of this interpolation method too and implemented it ourselves. Our implementation interpolates such as it is described in Sec. 3.6, but only uses the four nearest neighboring sampling points for the calculation. The class also provides the interpolation with all known sampling points, but we advise against the use of this functionality, because the interpolation is to slowly, since the procedure has to read all sampling points again for every point to interpolate. The implemented interpolation should work for large study areas with a high resolution (see (Nicolai and Nagel, 2012a), (Nicolai and Nagel, 2012b) and (Nicolai and Nagel, submitted in 2013)), so we will consider only the inverse distance weighting with consideration of the four nearest neighboring sampling points hereafter.

However, you have to choose the second parameter of the inverse distance weighting by using our implementation - the exponent for the calculation (see section 3.6).

The exponent modifies the influence of the sampling points which are further away from the point to interpolate. The higher the exponent, the lower the weight of these sampling points. Thus the interpolated value depends a lot on the sampling points in the nearest neighborhood. With a low exponent, the interpolated values are globally more similar instead. Thereby peaks and valleys occur, since the interpolated values differ from the values at the sampling points. Often an exponent of one or two is used (see Peters, 2009, pp. 15).

5 Testing the implemented interpolation methods

To compare the implemented interpolation methods according to special testing criteria, we tested them with accessibility data of a real-world scenario, the city of Zurich (Switzerland) (see (Nicolai and Nagel, 2012a), (Nicolai and Nagel, 2012b) and (Nicolai and Nagel, submitted in 2013)). The accessibility data of the city of Zurich is available at different grid resolutions, i.e. different grid cell sizes, for example at a resolution of $200m \times 200m$ or $100m \times 100m$. The low resolution data, i.e. $200m \times 200m$, is used as input for the interpolation methods in the Zurich testing scenario and is interpolated to the higher resolution of $100m \times 100m$. The resulting output is compared with the original accessibility data at the same resolution (see Sec. 5.2).

In MATSim4UrbanSim there are two methods to capture the study area, i.e. to save the accessibility grid data:

- Bounding box: The study area is defined as a rectangular area, a so called bounding box. This bounding box is approximated by the grid. It subdivides the area into squared cells, where the resulting cell centroids serve as measuring points for the accessibility calculations and for the interpolation. The spatial resolution depends on the selected cell size, which is configurable.

- Shapefile: The study are is specified very precisely by a shapefile. Analog to the bounding box approach the study area is given by a rectangular area that is approximated by a grid. However, it only contains values (or measures) that lie within the shapefile boundaries. Measuring points outside this boundary are marked as 'not a number', i.e. they do not have a numeric value. The spatial resolution again depends on the selected cell size, which is configurable.

In the Zurich scenario we tested the interpolation methods with both kinds of data representations. Even if we interpolate only grid points in this case, the interpolation procedures still may interpolate the value of an arbitrary point. We considered the following testing criteria by interpolating the Zurich scenario.

5.1 Interpolation time

The calculation time of the interpolation method is a very important testing criterion in our application since the implemented interpolation should work for large study areas with a high resolution (see (Nicolai and Nagel, 2012a), (Nicolai and Nagel, 2012b) and (Nicolai and Nagel, submitted in 2013)).

We measured the calculation time of the different interpolation methods by interpolating the Zurich scenario and compared the values in combination with the second testing criterion:

5.2 Interpolation error

To compare the implemented interpolation methods based on their interpolation quality we summed up the absolute difference between the interpolated and the original values at the resolution of $100m \times 100m$ for each interpolation method, after interpolating the accessibility data of the city of Zurich described above. Since many small differences to the original data are better than a few big ones, we also calculated the relative interpolation error by dividing the summed absolute differences by the number of points which differ from the original data. The solutions are compared in table 3 in combination with the interpolation time required.

5.3 Testing results

Analysing the interpolation results you will make the important discovery, that bicubic spline interpolation does not work with shapefile data (see Fig. 4(d)). The reason is that bicubic spline interpolation is a global interpolation method, which uses all sampling points for the interpolation of one value. Since the shapefile representation of the data includes 'not a number' entries outside the study area, as discussed before in the definition of the shapefile representation, all interpolated values will become 'not a number'.

Using the inverse distance weighting, we had to choose an exponent for the weights. Comparing the interpolation results of the Zurich scenario with different exponents, we selected an exponent of 10.0 in this particular scenario, because it produced low interpolation errors compared to other exponents. In comparison to the results with an exponent of 2.0 for example, we reduced the relative interpolation error from 0.1733 to 0.1617 with shapefile data and from 0.1616 to 0.1498 with bounding box data. The improvement is therefore about seven percent. Using a high exponent like 10.0 means, that the interpolated accessibilities depends a lot on the next neighboring values, which is desired in our application - the interpolation of accessibilities. An interpolation of a global trend between the sampling points with a lower exponent would in this case not be helpful. Nevertheless the exponent

should be choosen independently for every single scenario, since in other scenarios other exponents than 10.0 may cause better solutions.

We visualized the resulting accessibility data by interpolating the Zurich scenario with the different interpolation methods in Fig. 4 and 5.

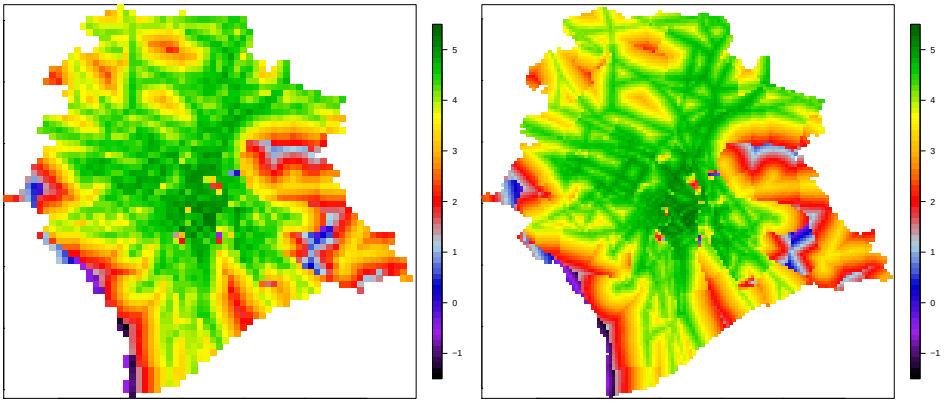
Interpolation method	Interpolation time	Sum of absolute differences	Relative difference
Bilinear interp.	31 ms	1419.00	0.1612
Bicubic spline interp.	48 ms	not comparable	not comparable
IDW with exp. 10.0	16 ms	1398.21	0.1617

(a) Interpolation of the Zurich scenario with shapefile data

Interpolation method	Interpolation time	Sum of absolute differences	Relative difference
Bilinear interp.	31 ms	2530.47	0.1498
Bicubic spline interp.	68 ms	2530.25	0.1498
IDW with exp. 10.0	16 ms	2530.76	0.1498

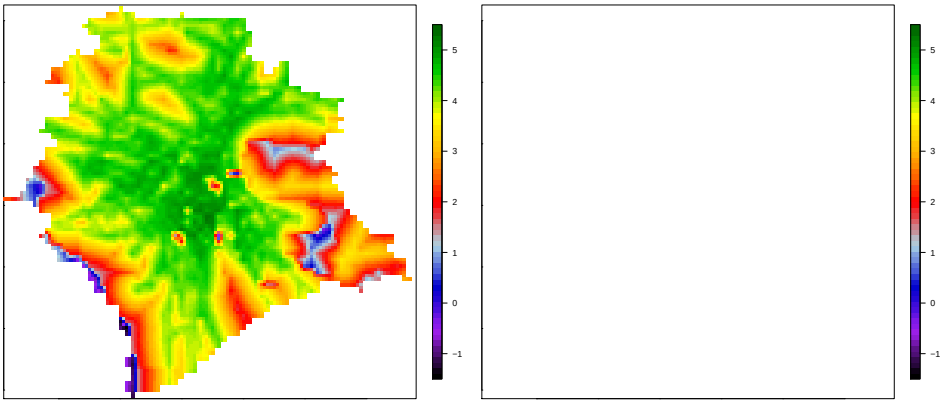
(b) Interpolation of the Zurich scenario with bounding box data

Figure 3: Comparison of interpolation time and error of the different interpolation methods by interpolating the Zurich testing scenario with shapefile- and bounding box data as described above.



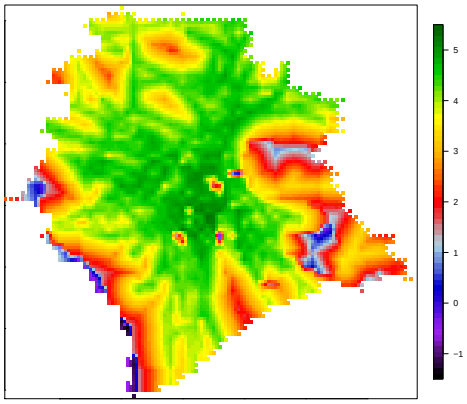
(a) Original accessibility data at a resolution of 200 meters

(b) Original accessibility data at a resolution of 100 meters



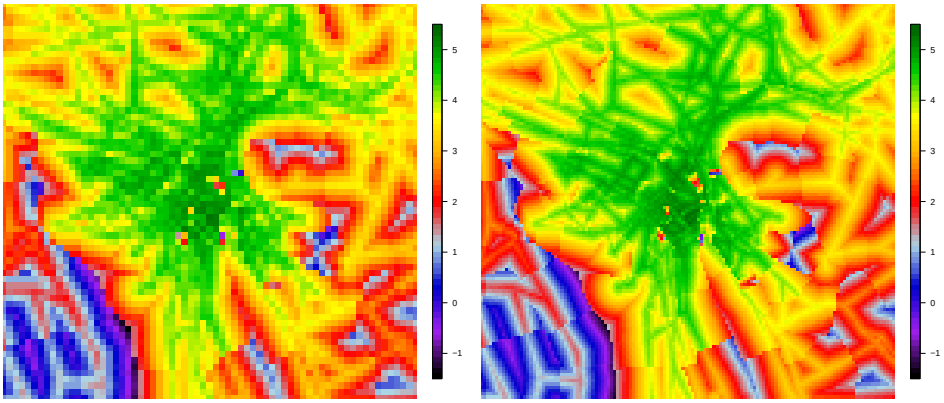
(c) Bilinear interpolation

(d) Bicubic interpolation



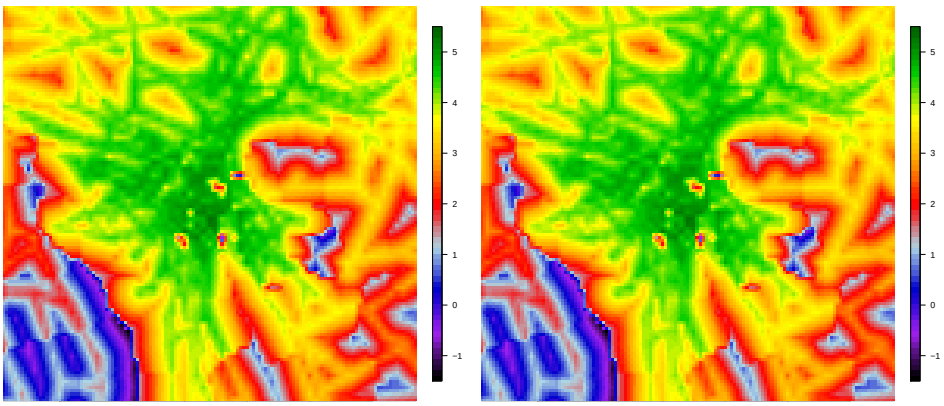
(e) IDW with exponent 10.0

Figure 4: The figures show the resulting accessibility plots from the Zurich testing scenario with shapefile data. The color bar on the right hand side indicates the accessibility level, where a good accessibility is indicated by green areas and poor accessibility is indicated by dark blue or black areas. More in-depth information on this can be found e.g. in Nicolai and Nagel (2012a).



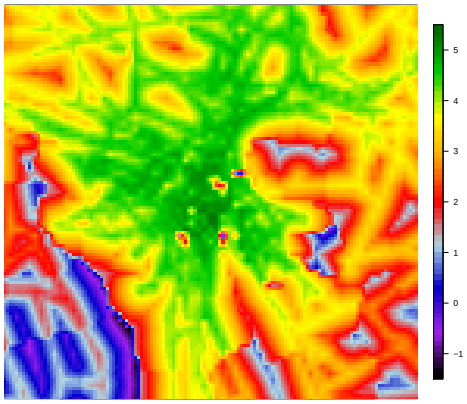
(a) Original accessibility data at a resolution of 200 meters

(b) Original accessibility data at a resolution of 100 meters



(c) Bilinear interpolation

(d) Bicubic interpolation



(e) IDW with exponent 10.0

Figure 5: The figures show the resulting accessibility plots from the Zurich testing scenario with bounding box data. The color bar on the right hand side indicates the accessibility level, where a good accessibility is indicated by green areas and poor accessibility is indicated by dark blue or black areas. More in-depth information on this can be found e.g. in Nicolai and Nagel (2012a).

6 Comparison of the implemented interpolation methods

Testing the described criterions of the implemented interpolations you may say that all interpolation methods were implemented successfully. With the tabular presentation of interpolation time and error of the Zurich testing scenario in Tab. 3 and the graphic representation of the testing results in Fig. 4 and 5, the three implemented methods may be easily compared. Altogether advantages and disadvantages of the specific methods may be found.

6.1 Evaluation of bilinear interpolation

A well known advantage of bilinear interpolation is the very low interpolation time. Also no oscillations occur in contrast to bicubic spline interpolation since the interpolated value is always an intermediate value between the neighboring sampling points. That is why the interpolated data seems to be a bit blurry if you interpolate grid data to one resolution higher and plot them. But this is no disadvantage because our application requires no image processing but the interpolation of accessibilities. Bilinear interpolation works for shapefile data and bounding box data. Hence this interpolation method is an easy continuous interpolation which has no disadvantages in our application.

6.2 Evaluation of bicubic spline interpolation

Bicubic spline interpolation needs more calculation time than bilinear interpolation. In other applications this disadvantage often is compensated with a higher interpolation precision. In our application this is not the case, because of the oscillation effect which we described in section 4.2. The observed interpolation error by testing the interpolation method is the same as with bilinear interpolation. Additionally, bicubic spline interpolation does not work with shapefile data, i.e. data with 'not a number' entries (see Sec. 5.3). You may try to solve this problem by replacing all 'not a number' entries with a number, e.g. zero. After this bicubic spline interpolation will work, but the values at the boundary regions will be falsified, which is why this is no satisfying solution. Hence the provided bicubic spline interpolation only works with bounding box data without 'not a number' entries. Despite its good estimation in other applications, in our case bicubic spline interpolation only should be used if the user is able to exclude the described problems for his application.

6.3 Evaluation of inverse distance weighting

The inverse distance weighting is a flexible interpolation method where the user has to choose the exponent for the distance weighting. As described in

Sec. 5.3 the choice of the exponent has a big influence on the interpolation error. With a suitable exponent, the inverse distance weighting produces the same interpolation error as bilinear and bicubic spline interpolation in the Zurich testing scenario. Since the quality of an exponent depends a lot on the particular scenario, we did not choose a default value, but left its choice up to the user. So this interpolation method rather is recommended for experienced users.

The consideration of only four known sampling points, restricts the quality possibilities of the interpolation in comparison to the consideration of all known sampling points. But the resulting interpolation error in the Zurich testing scenario with a suitable exponent is very low anyway and comparable with the one of the other two methods. Inverse distance weighting is therefore an alternative choice to bilinear interpolation for the interpolation of bounding box data.

But the inverse distance weighting causes problems with shapefile data, like bicubic spline interpolation: Interpolating shapefile data with consideration of the four nearest neighboring sampling points, the boundary will be fringed, since points lying in a grid cell with at least one unknown sampling point will be interpolated with a 'not a number' value. You may try to solve this problem by considering only the known neighboring sampling points. If you do so, you assume that the point to interpolate lies within the shapefile boundary of your study area. But the grid based representation of the area does not contain this information. So the suggested solution would create additionally information, which is why we did not correct the effect of the fringed boundary. The user should implicitly take this into account if he uses the inverse distance weighting for the interpolation of shapefile data. We recommend to use the implemented inverse distance weighting only with bounding box data, whithout 'not a number' entries.

6.4 Conclusion

Based on the presented comparison of the three implemented interpolation methods, you may say that bilinear interpolation is most suitable in our application - the interpolation of accessibilities. This method supplies both: fast computation and high accuracy. In addition, it works for shapefile data too. This conclusion is unexpected, since in other applications, e.g. image processing, other interpolation methods are favoured. However, for the interpolation in the software MATSim4UrbanSim we use bilinear interpolation as default. Because in specific cases it may be better to use one of the other interpolation methods, we leave to the user the possibility to switch between them. The following table may give a summarized overview on the applications of the three implemented interpolation methods.

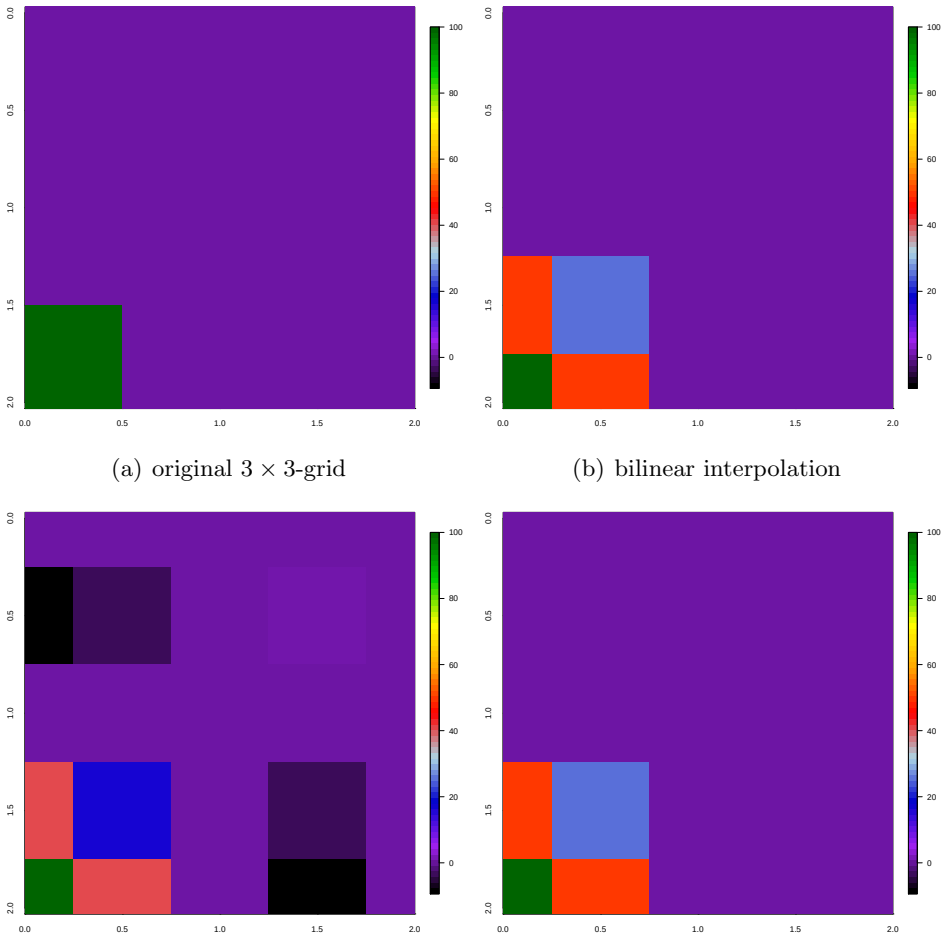
	Bilinear interpolation	Bicubic spline interpolation	IDW with four sampling points
Shapefile data	fast computation, high accuracy	impossible	unrecommended: boundary fringed
Bounding box data	fast computation, high accuracy	suitable, but slower	suitable with right exponent

Table 1: Tabular conclusion of the testing results to give an overview on the usage of the different methods.

A Appendix

A.1 Minimal Example

To be able to analyse the interpolation of specific values, we also tested the interpolation methods with a self created minimal scenario. This mininmal scenario demonstrates a testing region as a 3×3-grid with low accessibility everywhere, except one point with a high accessibility value (see Fig. 6(a)). We interpolated this 3×3-grid to the next higher grid resolution, as we did in the Zurich testing scenario. The results are 5×5-grids for the three different interpolation methods (see Fig. 6). With this minimal scenario you may test the exact interpolation at known sampling points and the range of interpolated values at unkown points. As we described in Sec. 3, our objective is the exact interpolation of known values. Testing the interpolation of unknown data is generally more difficult, since there are different possible solutions for an interpolation problem, as we described in section 3 too. Comparing the grids in Fig. 6 you may see that the methods partially produce different results. In Fig. 6(c) you may see the mild oscillations of bicubic spline interpolation described in Sec. 4.2.



(a) original 3×3-grid

(b) bilinear interpolation

(c) bicubic interpolation

(d) inverse distance weighting

Figure 6: These figures show the original 3×3-grids of the minimal scenario and its interpolations to one resolution higher with the different interpolation methods. In Fig. 6(c) you may see the oscillations of bicubic spline interpolation described in Sec. 4.2. The inverse distance weighting in Fig. 6(b) uses the four nearest sampling points for the interpolation and an exponent of 10.0. In this case a modification of the exponent has no effect.

A.2 Source code of implemented interpolation methods

The following pages show the source code of the implemented interpolation methods, discussed in Sec. 4. The methods are bilinear and bicubic spline interpolation as well as inverse distance weighting. The code can also be found online at https://matsim.svn.sourceforge.net/svnroot/matsim/contrib/trunk/matsim4urbansim.

```java
package org.matsim.contrib.matsim4opus.interpolation;

import org.matsim.contrib.matsim4opus.gis.SpatialGrid;

/**
 * Implements bilinear interpolation.
 * Uses linear spline interpolation with separation: first horizontal then vertical.
 * Own implementation (no suitable implementation found).
 *
 * Requires values on a SpatialGrid.
 *
 * @author tthunig
 *
 */
class BiLinearInterpolator {

    private SpatialGrid sg = null;

    /**
     * Prepares bilinear interpolation.
     *
     * @param sg the SpatialGrid to interpolate
     */
    BiLinearInterpolator(SpatialGrid sg){
        this.sg= sg;
    }

    /**
     * Interpolates the value on a arbitrary point with bilinear interpolation.
     *
     * @param xCoord the x-coordinate of the point to interpolate
     * @param yCoord the y-coordinate of the point to interpolate
     * @return interpolated value on the point (xCoord, yCoord)
     */
    double biLinearInterpolation(double xCoord, double yCoord){
        return biLinearValueInterpolation(this.sg, xCoord, yCoord);
    }

    /**
     * Interpolates the value on a arbitrary point with bilinear interpolation.
```

```
41      * Requires values on a grid as SpatialGrid.
42      *
43      * @param sg the values on the grid as SpatialGrid
44      * @param xCoord the x-coordinate of the point to interpolate
45      * @param yCoord the y-coordinate of the point to interpolate
46      * @return interpolated value on the point (xCoord, yCoord)
47      */
48     static double biLinearValueInterpolation(SpatialGrid sg, double xCoord, double yCoord){
49         double xDif= (xCoord-sg.getXmin()) % sg.getResolution();
50         double yDif= (yCoord-sg.getYmin()) % sg.getResolution();
51
52         //corner coordinates of the grid cell
53         double x1= xCoord-xDif;
54         double x2= x1+sg.getResolution();
55         double y1= yCoord-yDif;
56         double y2= y1+sg.getResolution();
57
58         double xWeight= xDif/sg.getResolution();
59         double yWeight= yDif/sg.getResolution();
60
61         //case differentiation important for boundary data of shapefiles, because of neighboring NaN values
62         if (xDif==0){
63             if (yDif==0){
64                 //known value
65                 return sg.getValue(xCoord, yCoord);
66             }
67             //point to interpolate lies at the grid cell boundary
68             return sg.getValue(x1, y1)*(1-yWeight) + sg.getValue(x1, y2)*yWeight;
69         }
70         if (yDif==0){
71             //point to interpolate lies at the grid cell boundary
72             return sg.getValue(x1, y1)*(1-xWeight) + sg.getValue(x2, y1)*xWeight;
73         }
74
75         //interpolates first in y-direction then in x-direction with linear splines
76         return (sg.getValue(x1, y1)*(1-yWeight) + sg.getValue(x1, y2)*yWeight) * (1-xWeight)
77                 + (sg.getValue(x2, y1)*(1-yWeight) + sg.getValue(x2, y2)*yWeight) * xWeight;
78     }
79 }
80
```

```java
package org.matsim.contrib.matsim4opus.interpolation;

import org.apache.commons.math.FunctionEvaluationException;
import org.apache.commons.math.MathException;
import org.apache.commons.math.analysis.BivariateRealFunction;
import org.apache.commons.math.analysis.interpolation.BicubicSplineInterpolator;
import org.apache.commons.math.analysis.interpolation.BivariateRealGridInterpolator;
import org.apache.log4j.Logger;

import org.matsim.contrib.matsim4opus.gis.SpatialGrid;

/**
 * Interpolates data on a SpatialGrid with bicubic spline interpolation from apache (http://commons.apache.org).
 *
 * Requires values on a SpatialGrid.
 *
 * Problem: Wave effects may occur.
 *
 * @author tthunig
 *
 */
class BiCubicInterpolator {

    private static final Logger log = Logger.getLogger(BiCubicInterpolator.class);

    private BivariateRealFunction interpolatingFunction = null;

    private SpatialGrid sg = null;

    /**
     * Prepares bicubic spline interpolation:
     * Generates an interpolation function with BicubicSplineInterpolator from apache
     * (http://commons.apache.org/math/apidocs/org/apache/commons/math3/analysis/interpolation/BicubicSplineInterpolator.html).
     *
     * @param sg the SpatialGrid to interpolate
     */
    BiCubicInterpolator(SpatialGrid sg){
        this.sg= sg;
        sgNaNcheck();

```

```
41          //create coordinate vectors for interpolation and a compatible array of values
42          double[] x_coords= coord(sg.getXmin(), sg.getXmax(), sg.getResolution());
43          double[] y_coords= coord(sg.getYmin(), sg.getYmax(), sg.getResolution());
44          double[][] mirroredValues= sg.getMatrix();
45
46          BivariateRealGridInterpolator interpolator = new BicubicSplineInterpolator();
47          try {
48              interpolatingFunction = interpolator.interpolate(y_coords, x_coords, mirroredValues);
49          } catch (MathException e) {
50              e.printStackTrace();
51          }
52      }
53
54      private void sgNaNcheck() {
55          for (double y = this.sg.getYmin(); y <= this.sg.getYmax(); y += this.sg.getResolution()) {
56              for (double x = this.sg.getXmin(); x <= this.sg.getXmax(); x += this.sg.getResolution()) {
57                  if (Double.isNaN(this.sg.getValue(x, y))){
58                      log.error("Bicubic spline interpolation doesn't work with NaN entries. " +
59                              "Please use bounding box data or shapefile data without NaN entries.");
60                      return;
61                  }
62              }
63          }
64      }
65
66      /**
67       * Interpolates the value at an arbitrary point with bicubic spline interpolation from apache.
68       *
69       * @param xCoord the x-coordinate of the point to interpolate
70       * @param yCoord the y-coordinate of the point to interpolate
71       * @return interpolated value on the point (xCoord, yCoord)
72       */
73      double biCubicInterpolation(double xCoord, double yCoord){
74          try {
75              return interpolatingFunction.value(yCoord, xCoord);
76          } catch (FunctionEvaluationException e) {
77              e.printStackTrace();
78          }
79          return Double.NaN;
80      }
```

```
 81
 82     /**
 83      * Creates a coordinate vector.
 84      *
 85      * @param min the minimum coordinate
 86      * @param max the maximum coordinate
 87      * @param resolution
 88      * @return coordinate vector from min to max with the given resolution
 89      */
 90     private static double[] coord(double min, double max, double resolution) {
 91         double[] coord = new double[(int) ((max - min) / resolution) + 1];
 92         coord[0] = min;
 93         for (int i = 1; i < coord.length; i++) {
 94             coord[i] = min + i * resolution;
 95         }
 96         return coord;
 97     }
 98
 99 }
100
```

```java
package org.matsim.contrib.matsim4opus.interpolation;

import org.matsim.contrib.matsim4opus.gis.SpatialGrid;

/**
 * Implements inverse distance weighting for interpolation. Own implementation (no suitable implementation found).
 *
 * Requires values on a SpatialGrid.
 *
 * Problem: Peaks and valleys occur.
 *
 * For more information see e.g.:
 * http://www.geography.hunter.cuny.edu/~jochen/GTECH361/lectures/lecture11/concepts/Inverse%20Distance%20Weighted.htm
 * or: http://gisbsc.gis-ma.org/GISBScL7/de/html/VL7a_V_lo7.html (German).
 *
 * @author tthunig
 *
 */
class InverseDistanceWeighting {

    private SpatialGrid sg = null;

    /**
     * Prepares the interpolation with the inverse distance weighting method.
     *
     * @param sg the SpatialGrid to interpolate
     */
    InverseDistanceWeighting(SpatialGrid sg){
        this.sg= sg;
    }

    /**
     * Interpolates the value on a arbitrary point with inverse distance weighting.
     * Considers only four neighboring values because this method needs less time for calculation than considering all known values
     * and the result is even more suitable for accessibility interpolation.
     *
     * @param xCoord the x-coordinate of the point to interpolate
     * @param yCoord the y-coordinate of the point to interpolate
     * @param exponent the exponent for the inverse distance weighting
     * @return interpolated value on the point (xCoord, yCoord)  *
```

```
41      */
42     double inverseDistanceWeighting(double xCoord, double yCoord, double exponent){
43         return fourNeighborsIDW(this.sg, xCoord, yCoord, exponent);
44     }
45
46     /**
47      * Interpolates a value at the given point (xCoord, yCoord) with the inverse distance weighting with variable power of weights.
48      * Considers only four neighboring values.
49      *
50      * @param sg the SpatialGrid with the known values
51      * @param xCoord
52      * @param yCoord
53      * @param exp the exponent for the weights. standard values are one or two.
54      * @return interpolated value at (xCoord, yCoord)
55      */
56     static double fourNeighborsIDW(SpatialGrid sg, double xCoord, double yCoord, double exp) {
57         double xDif= (xCoord-sg.getXmin()) % sg.getResolution();
58         double yDif= (yCoord-sg.getYmin()) % sg.getResolution();
59
60         //known value
61         if (xDif==0 && yDif==0){
62             return sg.getValue(xCoord, yCoord);
63         }
64
65         double x1= xCoord-xDif;
66         double x2= x1+sg.getResolution();
67         double y1= yCoord-yDif;
68         double y2= y1+sg.getResolution();
69
70         //calculate distances to the 4 neighboring sampling points
71         double d11= Math.pow(distance(x1, y1, xCoord, yCoord), exp);
72         double d12= Math.pow(distance(x1, y2, xCoord, yCoord), exp);
73         double d21= Math.pow(distance(x2, y1, xCoord, yCoord), exp);
74         double d22= Math.pow(distance(x2, y2, xCoord, yCoord), exp);
75
76         //interpolation at the boundary
77         if (xCoord == sg.getXmax()){
78             //consider only 2 neighboring sampling points (up and down)
79             return (sg.getValue(x1, y1)/d11 + sg.getValue(x1, y2)/d12) / (1/d11 + 1/d12);
80         }
```

```
81          if (yCoord == sg.getYmax()){
82              //consider only 2 neighboring sampling points (left and right)
83              return (sg.getValue(x1, y1)/d11 + sg.getValue(x2, y1)/d21) / (1/d11 + 1/d21);
84          }
85
86          //interpolation with 4 neighboring sampling points
87          return (sg.getValue(x1, y1)/d11 + sg.getValue(x1, y2)/d12 + sg.getValue(x2, y1)/d21 + sg.getValue(x2, y2)/d22)
88                  / (1/d11 + 1/d12 + 1/d21 + 1/d22);
89      }
90
91      /**
92       * Calculates the distance between two given points in the plane.
93       *
94       * @param x1 the x-coordinate of point 1
95       * @param y1 the y-coordinate of point 1
96       * @param x2 the x-coordinate of point 2
97       * @param y2 the y-coordinate of point 2
98       * @return distance between the points (x1,y1) and (x2,y2)
99       */
100     private static double distance(double x1, double y1, double x2, double y2) {
101         return Math.sqrt((y2-y1)*(y2-y1) + (x2-x1)*(x2-x1));
102     }
103
104     /**
105      * Attention: Experimental version. Not tested sufficiently. Requires too much calculation time.
106      *
107      * Interpolates a value at the given point (xCoord, yCoord) with the inverse distance weighting with variable power of weights:
108      * z(u_0)= Sum((1/d_i^exp)*z(u_i)) / Sum (1/d_i^exp).
109      * Needs more time for calculation than fourNeighborsIDW and the result is even less suitable for accessibility interpolation.
110      *
111      * @param sg the SpatialGrid with the known values
112      * @param xCoord
113      * @param yCoord
114      * @param exp the exponent for the weights. standard values are one or two.
115      * @return interpolated value at (xCoord, yCoord)
116      */
117     @Deprecated
118     static double allValuesIDW(SpatialGrid sg, double xCoord, double yCoord, double exp) {
119         double xDif= (xCoord-sg.getXmin()) % sg.getResolution();
120         double yDif= (yCoord-sg.getYmin()) % sg.getResolution();
```

```
121
122         //known value
123         if (xDif==0 && yDif==0){
124             return sg.getValue(xCoord, yCoord);
125         }
126
127         //interpolation with all known sampling points
128         double distanceSum=0;
129         double currentWeight=1;
130         double weightSum=0;
131         for (double y = sg.getYmin(); y <= sg.getYmax(); y += sg.getResolution()){
132             for (double x = sg.getXmin(); x <= sg.getXmax(); x += sg.getResolution()){
133                 currentWeight= Math.pow(distance(x, y, xCoord, yCoord), exp);
134                 distanceSum+= sg.getValue(x, y)/currentWeight;
135                 weightSum+= 1/currentWeight;
136             }
137         }
138         return distanceSum/weightSum;
139     }
140
141 }
142
```

References

The Apache Software Foundation. Package org.apache.commons.math3.analysis.interpolation, 2013. URL http://commons.apache.org/math/apidocs/org/apache/commons/math3/analysis/interpolation/. Accessed March 2013.

M. Balmer, B. Raney, and K. Nagel. Adjustment of activity timing and duration in an agent-based traffic flow simulation. In H.J.P. Timmermans, editor, *Progress in activity-based analysis*, pages 91–114. Elsevier, Oxford, UK, 2005.

M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K. W. Axhausen. MATSim-T: Architecture and simulation times. In A.L.C. Bazzan and F. Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 57–78. IGI Global, 2009.

G.E.P. Box. Robustness in the strategy of scientific model building. In *Robustness in Statistics*, pages 201–236. Academic Press, 1979.

R.L. Burden and J.D. Faires. *Numerical analysis*. PWS-KENT Publishing Company, Boston, 1989.

K.-T. Chang. *Introduction to geographic information systems*. McGraw-Hill, New York, 2012.

M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 2000.

E. Miller, K. Nagel, H. Ševčíková, D. Socha, and P. Waddell. OPUS: An open platform for urban simulation. 2005. URL http://128.40.111.250/cupum/searchpapers/detail.asp?pID=428.

T.W. Nicolai. *MATSim for UrbanSim: Integrating an urban simulation model with a travel model*. PhD thesis, Berlin Institute of Technology, 2013. forthcomming.

T.W. Nicolai and K. Nagel. Coupling transport and land-use: Investigating accessibility indicators for feedback from a travel to a land use model. In *Latsis Symposium 2012 – 1st European Symposium on Quantitative Methods in Transportation Systems*, Lausanne, Switzerland, 2012a. VSP Working Paper 12-16. See www.vsp.tu-berlin.de/publications.

T.W. Nicolai and K. Nagel. Sensitivity tests with high resolution accessibility computations. VSP Working Paper 12-22, TU Berlin, Transport Systems Planning and Transport Telematics, 2012b. See www.vsp.tu-berlin.de/publications.

T.W. Nicolai and K. Nagel. *Handbook on Sustainable Land Use Modelling*, chapter Integration of agent-based transport and land use models. EPFL Press, 2013. forthcoming.

T.W. Nicolai and K. Nagel. High resolution accessibility computations. In A. Condeço, A. Reggiani, and J. Gutiérrez, editors, *Accessibility and spatial interaction*. Edward Elgar, submitted in 2013. Also VSP Working Paper 13-02. See www.vsp.tu-berlin.de/publications.

T.W. Nicolai, L. Wang, K. Nagel, and P. Waddell. Coupling an urban simulation model with a travel model – A first sensitivity test. In *Computers in Urban Planning and Urban Management (CUPUM)*, Lake Louise, Canada, 2011. Also VSP Working Paper 11-07. See www.vsp.tu-berlin.de/publications.

T.W. Nicolai, C. Zöllig Renner, and K. Nagel. *Handbook on Sustainable Land Use Modelling*, chapter General description of the state of the art of integrated transport land use modeling. EPFL Press, 2013. forthcoming.

OPUS User Guide. *The Open Platform for Urban Simulation and UrbanSim Version 4.3*. University of California Berkley and University of Washington, January 2011. URL http://www.urbansim.org.

S. Peters. Ein Vergleich räumlicher Interpolationsverfahren für Ertragswerte im Weinanbau. In *gis Science, ISSN: 1430-3663, Nr.2*, pages 50–56, 2009.

B. Raney and K. Nagel. An improved framework for large-scale multi-agent simulations of travel behaviour. pages 305–347. 2006.

S.W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, 1997. See www.DSPguide.com.

H. Späth. *Zweidimensionale Spline-Interpolations-Algorithmen*. R. Oldenbourg Verlag GmbH, München, Wien, 1991.

P. Waddell. Urbansim: Modeling urban development for land use, transportation, and environmental planning. *Journal of American planning Association*, 68(3):297 – 314, 2002.